HAL
open science

An IoT Attribute-Based Security Framework for Topic-based Publish/Subscribe Systems

Olivier Blazy, Emmanuel Conchon, Mathieu Klingler, Damien Sauveron

To cite this version:

Olivier Blazy, Emmanuel Conchon, Mathieu Klingler, Damien Sauveron. An IoT Attribute-Based Security Framework for Topic-based Publish/Subscribe Systems. IEEE Access, 2021, pp.1-1. 10.1109/ACCESS.2021.3051469 . hal-03133781

HAL Id: hal-03133781

https://unilim.hal.science/hal-03133781

Submitted on 8 Dec 2023

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.

IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received December 9, 2020, accepted December 22, 2020, date of publication January 13, 2021, date of current version February 2, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3051469

An IoT Attribute-Based Security Framework for Topic-Based Publish/Subscribe Systems

OLIVIER BLAZY, EMMANUEL CONCHON, MATHIEU KLINGLER, AND DAMIEN SAUVERON
XLIM, UMR CNRS 7252, Université de Limoges, 87032 Limoges, France

Corresponding author: Damien Sauveron (damien.sauveron@unilim.fr)

This work was supported in part by the MIRES research federation under grants for project "CANIoT," in part by the Région Nouvelle-Aquitaine under the grant for project "SVP-IoT," and in part by the ID-Fix project, an ANR funded project, under Grant ANR-16-CE39-0004.

ABSTRACT Publish/subscribe is a widely used paradigm in the Internet of Things (IoT). It allows a loose coupling between data producers and data consumers using a network of interconnected brokers. However, sensitive data could be exposed if a broker is compromised or if the broker itself is curious about the information that is exchanged. In this paper, we present a complete security framework for topic-based publish/subscribe systems to ensure both security and privacy at the broker level, going beyond the naive encryption of information while keeping the loose coupling between publishers and subscribers. Furthermore, the proposed solution enables user revocation at the broker level; i.e. a revoked user can no longer subscribe to published data. To achieve that, we propose a unified solution relying on attribute-based cryptography with: (1) Attribute-Based Encryption (ABE) for data encryption; (2) a new construction of Attribute-Based Keyword Search (ABKS) to allow the broker to perform an encrypted matching that enforces privacy; and (3) an Attribute-Based Signature (ABS) to enforce the data authentication.

INDEX TERMS Secure publish/subscribe, attribute-based cryptography, publications/subscriptions confidentiality, user revocation.

I. INTRODUCTION

With the massive adoption of IoT in many application areas, using appropriate communications technology to enable a seamless evolution of the systems (e.g. network addition or removal of devices), while ensuring a relevant level of security, is mandatory. From a communication perspective, the publish/subscribe (a.k.a. pub/sub) paradigm has been proposed to enable communication between IoT and users. From a security perspective, this paper fills existing gaps by proposing an IoT attribute-based security framework for topic-based pub/sub systems.

Pub/sub systems communication architecture relies on three different entities: broker, publishers and subscribers. The broker routes data produced by publishers to subscribers, which provides a loose coupling between them. Publishers and subscribers do not have to communicate directly with each other to exchange data. Data sent by publishers are defined as publications and the interests that subscribers give to the broker are defined as subscriptions.

Upon receiving publications, the broker matches them with subscriptions. Thus, publishers do not know about subscribers' interests, and subscribers do not have information about publishers.

Pub/sub solutions can be split into two main categories: content-based and topic-based. While content-based systems give subscribers more possibilities for their subscriptions, topic-based pub/sub systems are the most widespread. Indeed, at the time of writing no industrial standard is available for content-based pub/sub systems while several topic-based systems such as MQTT [1] or Kafka [2] are widely used in IoT. In this article, we will focus on topic-based pub/sub systems where: 1) publications are defined as a couple (topic, payload) in which the payload contains the application data and the topic describes the payload content; and 2) subscriptions are defined as a request on a topic.

Thanks to their interesting features, pub/sub systems are used in different scenarios using IoT devices such as e-health

The associate editor coordinating the review of this manuscript and approving it for publication was Vyasa Sai.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

to monitor information from medical equipment [3], [4]. In a smart home context, applications or actuators can subscribe to sensors' publications to provide new services to improve occupiers' quality of life. Most of the time, data from sensors are collected by a gateway (i.e. a more powerful device) that publishes them on the broker on their behalf.

However, in most real-life use cases, the adoption of pub/sub systems is limited to private network usage due to security and privacy concerns.

Very few pub/sub solutions offer security mechanisms (e.g. SSL/TLS with MQTT or Kafka) to protect underlying communication channels between every participant (publisher, broker and subscriber), and when considering deployment in public networks, none of them are able to ensure publication confidentiality. In contrast with private networks, in public networks, a broker cannot be considered fully trusted as it can be controlled by external entities/companies. Thus a broker can view every publication and subscription it deals with.

A first naive solution to enforce publication confidentiality is to encrypt payloads (i.e. applications data) prior to publication so that the broker cannot read them. However this is not sufficient to avoid leakage of private information, from both subscribers' and publishers' perspectives, since a broker has to perform matchings between publication topics and subscription topics. A requirement is then to provide security features that allow the broker to still use subscription topics, but without gathering information about the associated subscribers (apart from routing information), and the publication topics. Another important requirement that may be considered to improve trust in the overall system is the ability to revoke malicious users from the subscriber list with perfect forward secrecy.

A. CONTRIBUTIONS

In this paper, our goal is to improve the overall security of topic-based pub/sub systems through three main aspects: *Subscription confidentiality*, *Publication confidentiality* and *Payload confidentiality*.

Our salient contributions are to propose a security framework for topic-based pub/sub systems that relies on attribute-based cryptography allowing: (1) private fine-grained access control over data in a multi-publishers, multi-subscribers scenario (i.e. typical IoT scenario); (2) data authentication; and (3) user revocation. To achieve that, we have orchestrated several existing schemes in a consistent manner and have also modified the construction of one of them to satisfy the constraints of pub/sub systems while providing a formal proof of security. We also have conducted comprehensive experiments establishing that the proposed framework is suitable for real-world deployments. Our work and experiments are based on the topic-based publish/subscribe implementation MQTT that is a standard in IoT communication [5]. Our modifications of the cryptographic scheme presented in Section V make it suitable for any distributed environment, not merely for pub/sub systems.

B. STRUCTURE OF THE PAPER

Section II reviews the literature related to pub/sub systems confidentiality. Section III presents the framework system model to introduce the preliminaries about attribute-based schemes and to state the security requirements and their fulfillment, as well as the considered threat and security models. Section IV details the different parts of our framework and discusses the security it provides. Section V focuses on modifications we have made to an existing cryptographic attribute-based scheme to make the private matching algorithm suitable for pub/sub systems. This Section also establishes the correctness of the modified scheme. In addition, a formal security proof that our modifications do not decrease the security of the original scheme is provided in Section VI. Section VII then provides both a complexity analysis and a performance evaluation of the framework implementation on both a regular computer and a Raspberry Pi 3 as an IoT-like device. Finally, Section VIII concludes the paper with suggestions for future work.

II. RELATED WORK

In this section, the most relevant proposed solutions for providing confidentiality in pub/sub systems are presented, along with their main drawbacks compared to our proposal.

A. ACCESS CONTROL SOLUTIONS

Studies around the confidentiality of pub/sub systems have first focused on brokers' access control. Several solutions have been proposed using Role-Based Access Control (RBAC) [6], [7], Attribute-Based Access Control (ABAC) [8], [9] and Policy-Based Access Control (PBAC) [10], [11]. In these schemes, the authentication has to be done by a trusted (or at least semi-trusted) entity (i.e. a proxy) but data that transit via the broker are not encrypted. Access control over encrypted data is more scalable than access control via proxy because the latter requires the use of a proxy to ensure the authentication; this is a bottleneck compared to the former, which does not need such an entity.

B. CLASSIC ENCRYPTION SOLUTIONS

Another approach has been considered on end-to-end encryption with a secret key per subscriber [12], and using a symmetric key per group of users [13]–[16]. However, all of these solutions need interaction between publishers and subscribers, which is not satisfactory in a pub/sub system where loose coupling is a key property. Furthermore, both symmetric schemes and classic asymmetric schemes present limitations in term of scalability since publishers have to encrypt their publications for each subscriber/group of subscribers.

C. ATTRIBUTE-BASED ENCRYPTION SOLUTIONS

Some pub/sub solutions use attribute-based cryptosystems as presented in [17]. In [18], Yuen *et al.* define a security model for pub/sub systems and describe a solution to

secure it. They use Attribute-Based Encryption (ABE) on the payload and the sanitizable signature scheme developed by Suzuki *et al.* [19]. More recently, Thatmann *et al.* [20] applied an ABE combined with an AES symmetric scheme to encrypt the payload and manage users' revocation with a global update of the AES keys. Like us, their solution is for topic-based pub/sub systems and is applied to MQTT in an IoT context. Those two schemes ensure payload confidentiality but do not provide a solution for the topic confidentiality. In a content-based pub/sub system, Ion *et al.* [21] provide payload confidentiality with ABE and deal with encrypted matching to hide publications and subscriptions from the broker using the Searchable Data Encryption (SDE) scheme developed by Dong *et al.* [22]. However, an online proxy is required to perform SDE, which increases the overall complexity and moves the requirement for privacy onto it instead of the broker. In this paper, we adopt an encrypted matching scheme for topic-based pub/sub systems without requiring a proxy.

III. FRAMEWORK SYSTEM MODEL

Our goal being to ensure security and privacy in a topic-based pub/sub system while maintaining loose coupling between publishers and subscribers when facing an honest-but-curious broker, we propose an attribute-based security framework to provide data confidentiality, data authentication and encrypted matching with the same set of attributes to ease the deployment process in real-world environments.

In this section, we first introduce the basics and describe the cryptographic schemes used in our framework for topic-based pub/sub systems. Then, the different stakeholders involved in the system are described before specifying the security requirements along with fulfilments, the threat model and the security model.

A. PRELIMINARIES

In this section, first we set the security hypothesis and then describe the three attribute-based cryptographic schemes that are used in our framework. We uniformised the notations used in the different pre-existing schemes to provide more consistent explanations.

1) SECURITY HYPOTHESIS

We rely on the 3-party decision Diffie-Hellman assumption (TDH) as defined in [23].

In a bilinear group $\mathbb{G}$, with a bilinear map e, given g^a, g^b, g^c and g^d, it is hard to decide whether $d = abc$.

2) ABE

Attribute-based encryption is an asymmetric encryption scheme. It allows users to encrypt information based on the attributes they own. It offers producers access control over encrypted data. There are two approaches: Key-Policy Attribute-Based Encryption (KP-ABE), and Ciphertext-Policy Attribute-Based Encryption (CP-ABE). In KP-ABE, a set of attributes is attached to the data and users are registered to the trusted authority (TA) with an access structure. In CP-ABE, users are defined by the TA with a set of attributes and an access structure is attached to the data. Both can be used to provide access control by encrypting a message for a group of users. In our framework, we chose to rely on the CP-ABE model because it is more suitable for a broadcast environment such as pub/sub.

CP-ABE consists of the following four algorithms:

- $\mathsf{Setup}(d) \rightarrow (PK, MK)$: Given an input security parameter d, the **Setup** algorithm generates the master public key (PK) and the master secret key (MK).
- $\mathsf{KeyGen}(MK, \mathcal{A}) \rightarrow (SK)$: Given a set of attributes $\mathcal{A}$ and the master secret key MK, the **KeyGen** algorithm produces a secret key SK for a user.
- $\mathsf{Encrypt}(PK, GT, m) \rightarrow (c)$: When a user wants to encrypt a plaintext message m, he first chooses an access structure GT. Then, he runs the **Encrypt** algorithm with the master public key PK, his access structure GT and m to produce a ciphered message c.
- $\mathsf{Decrypt}(SK, c) \rightarrow (m)$: When a user wants to decrypt a message, he runs the **Decrypt** algorithm with his secret key SK and the ciphered message c. The plaintext message m is recovered if and only if his attributes $\mathcal{A}$ match the access structure GT of the ciphered message.

3) ABKS-UR

Sun *et al.* designed the Attribute-Based Keyword Search with User Revocation (ABKS-UR) [24], which provides searchable encryption incorporating a user revocation solution. First, the data producer sets an access structure. Then, keywords describing the data are encrypted with this access structure and sent to a server. Second, the consumer generates a trapdoor, which is based on his attributes and targeted keywords. This trapdoor is sent to the same server as in the previous step. If the trapdoor matches the access structure, data are sent to the consumer. If not, nothing happens.

ABKS-UR is composed of nine fundamental algorithms to perform an attribute-based keyword search with user revocation for keyword space $\mathcal{W}$ and access structure space $\mathcal{G}$. Let $\mathcal{N}$ denotes the universe of attributes $\{1, \ldots, n\}$ with $n \in \mathbb{N}$. The set of attributes used for the user's secret key is depicted as $\mathcal{A}$ with $\mathcal{A} \subseteq \mathcal{N}$.

- $\mathsf{Setup}(d, \mathcal{N}) \rightarrow (PK, MK)$: The **Setup** algorithm receives as input the security parameter d and the universe of attributes $\mathcal{N}$. It defines a bilinear group $\mathbb{G}$ of prime order p with a generator g and a non-degenerative bilinear map defined as $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_1$. It outputs the master public key PK and the master secret key MK. It also initializes the version numbers of master keys ver as 1.
- $\mathsf{CreateUL}(PK, ID) \rightarrow (UL)$: This algorithm takes as input the master public key PK and the user identity ID. It outputs UL, an updated list of users registered by the data producer.
- $\mathsf{EncIndex}(PK, GT, W) \rightarrow (D)$: This algorithm receives as input parameters the master public key PK, the access

structure $GT \in \mathcal{G}$ and a set of keywords $W \subseteq \mathcal{W}$. It outputs the encrypted index D.

- KeyGen$(PK, MK, \mathcal{A}) \rightarrow (SK)$: This algorithm receives as input the master public key PK, the master secret key MK and the set of user attributes $\mathcal{A}$. It then outputs the user's secret key SK.
- ReKeyGen$(\Phi, MK) \rightarrow (rk, MK', PK')$: This algorithm is used to perform re-encryption key generation. It receives as input the attribute set Φ, which contains the attributes to update and the current master secret key MK. It outputs a re-encryption key rk, which is used to recompute the key after user revocation, for all attributes in $\mathcal{N}$, the updated master secret key MK' and the updated master public key PK' where both version numbers are increased by 1. In rk, attributes that are not present in Φ have their re-encryption value set to 1.
- ReEncIndex$(rk, D, \Delta) \rightarrow (D')$: This algorithm receives as input the re-encryption key rk, the encrypted index D and the attribute set Δ that includes all the attributes in the access structure GT of D where the re-encryption values in rk are not 1. It outputs the new re-encrypted index D'.
- ReKey$(\Omega, rk, PSK) \rightarrow (PSK')$: This algorithm receives as input the attribute set Ω that contains all the attributes to update, and the partial secret key PSK that corresponds to the attribute part of the user's secret key. It also receives the re-encryption key rk. It outputs the new partial secret key PSK'.
- GenTrapdoor$(PK, SK, w') \rightarrow (Q)$: This algorithm is used to generate the trapdoor. It receives as input the master public key PK, the user's secret key SK and a set of keywords of interest $W' \subseteq \mathcal{W}$. It outputs the trapdoor Q associated with the set of keywords W'.
- Search$(UL, D, Q) \rightarrow$ {search result, $\perp$}: This algorithm receives as input the list of users UL, the encrypted index D and the trapdoor Q. It outputs the search result or notifies a failure with $\perp$.

4) ABS

Attribute-Based Signature (ABS) is an extension of the concept of Identity-Based Signature (IBS) [25], where users can sign with identifying information instead of their identity. In other words, they sign with any predicate of their attributes delivered by the TA. It allows users to do an anonymous authentication since they are indistinguishable from any other user who signs with the same predicate. The ABS with flexible threshold predicate support proposed by Li *et al.* [26] allows users to prove the ownership of attributes among those of the set given by the TA.

A signature key corresponding to a set $\alpha \subseteq \mathcal{N}$ of attributes is given to a user by the TA. To sign a message, a user chooses a subset $\alpha^* \subseteq \mathcal{N}$ associated with the message. This scheme consists of four algorithms:

- Setup$(d, \mathcal{N}) \rightarrow (PK, MK)$: To generate a master public key PK and a master secret key MK according to the security parameter d and the universe of attributes $\mathcal{N}$.

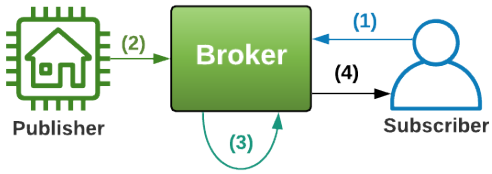


FIGURE 1. Publish/Subscribe Architecture.

- Extract$(PK, MK, \alpha) \rightarrow (SK)$: To generate a secret key SK for a user who possesses an attributes set α.
- Sign$(SK, \alpha^*, m) \rightarrow (\sigma)$: Assuming that a user has his secret key SK corresponding to his attributes α, to sign a message m and to prove he possesses k attributes from a set α^*, he selects a subset α' of size k such that $\alpha' \subseteq \alpha \cap \alpha^*$.
- Verify$(m, \alpha^*, PK, \sigma) \rightarrow \{0, 1\}$: To verify a signature σ associated with a message m, the master public key PK is used. It proves that the signer owns at least k attributes of α^*.

B. FRAMEWORK OVERVIEW

In this section, first, the basic topic-based pub/sub system is described. Then, the security requirements are introduced and how our framework fulfills them is explained. The threat model is also clearly stated, along with the security model.

1) SYSTEM MODEL

In a topic-based pub/sub system, there are three main entities as illustrated in Figure 1: *Broker*, *Publisher* and *Subscriber*. Publishers and subscribers interact through the broker.

Publishers send publications to subscribers via the broker. A publication `P` is defined by two elements: topic and payload. The topic is a string describing the publication content and the payload is the associated data. For instance, a publication sent by a temperature sensor could be `P:=(topic="temperature", payload="23°C")`. To subscribe, users have to send a subscription containing topics that correspond to their interests. The overall communication process can be depicted in four steps: (1) subscribers first subscribe to one or multiple topics via the broker (i.e. subscriptions); (2) a publisher sends a publication (i.e. data labelled with a topic); (3) the broker receives the publication and checks whether the topic of the publication matches some subscriptions; (4) if there is at least one match, the broker forwards the publication to the relevant subscribers.

2) SECURITY REQUIREMENTS

To provide our secure framework for topic-based pub/sub system, the following security requirements are:

- R1: The payload of the publication should be encrypted to be hidden from the broker and any unauthorized parties.
- R2: The broker should be able to match publications with subscriptions without gaining information about

their content or interests (i.e. publication topic and subscription topic confidentiality should be ensured).
- R3: If a subscriber is revoked from the system, he should not be able to access any future publications.
- R4: The publisher should be able to ensure publication authentication.

3) SECURITY REQUIREMENT FULFILLMENTS

First, we solve the payload confidentiality requirement (R1) by using a Ciphertext-Policy Attribute-Based Encryption (CP-ABE) scheme [27] on the publication payload. The publisher defines an access structure corresponding to the attributes needed to decrypt the message. With this policy and the master public key (PK_{ABE}) that is spread to all users, a publisher can then encrypt data so that only subscribers with the matching attributes can decrypt them using their own secret keys (SK_{ABE}).

Second, we address the issue of publication and subscription confidentiality (i. e. the topic published or subscribed) by modifying the Attribute-Based Keyword Search with a User Revocation (ABKS-UR) encryption scheme designed for a centralized system relying on a central cloud server [24] while maintaining the security level as explained in Section V. The broker can match publications with subscriptions without learning any information (R2). To achieve that, publishers encrypt topics with the master public key $PK_{ABKS-UR}$ and subscribers generate trapdoors for the desired topics (i.e. subscriptions) based on their own secret key ($SK_{ABKS-UR}$). This solution avoids an unauthorized subscriber subscribing to a topic that does not match his attributes as he is not able to generate the correct trapdoor for the matching to occur.

Third, thanks to user revocation, a user can no longer subscribe to a topic when he is revoked since his key is outdated upon revocation (R3). His past subscriptions are no longer effective either.

Fourth, we enable publishers to authenticate publications with their attributes thanks to an Attribute-Based Signature (ABS) [26] (R4). They sign their publications with their own secret key SK_{ABS}, which relies on the same universe of attributes as previous schemes that enforce an homogeneous vocabulary from a cryptographic standpoint.

Finally, it must be noted that the proposed security framework does not change the loose coupling between publishers and subscribers, which is a mandatory functional requirement of pub/sub systems. To use those schemes, compared to the entities illustrated in Figure 1, an entity was missing, which has been added in our security framework: the Trusted Authority (TA) which manages the universe of attributes and key distribution to every user in the system, as illustrated in Figure 2.

4) THREAT MODEL

We consider the following threats:

- Everyone can publish on the broker, and everyone can also subscribe (it does not reject communications).

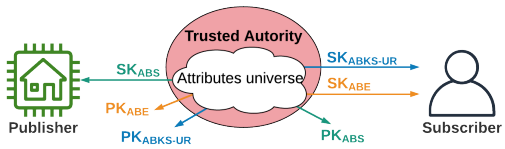


FIGURE 2. Keys provided by the trusted authority.

- Malicious subscribers can try to collude together to access unauthorized data.
- A revoked subscriber can still try to access the broker after revocation.

5) SECURITY MODEL

a: INDISTINGUISHABILITY WITH RESPECT TO SUBSCRIBERS

In this experiment, we aim to show that colluding subscribers cannot generate a valid ABKS key to access data that none of them are allowed to access on their own. In practice, the adversary picks a set of corrupted subscribers, a policy none of them meets, two messages M_0, M_1, and receives an encryption of one of those messages according to the policy. He succeeds if he manages to guess the encrypted message with non-negligible advantage.

b: TRAPDOOR UNLINKABILITY

In this experiment, we aim to show that the broker cannot distinguish whether two trapdoors Q_0, Q_1 are generated independently.

When generating the trapdoor, the user picks a different random number u to obfuscate the trapdoor. This means that an observer is unable to differentiate two or more trapdoors even when they are produced with the same keyword. Thus, our construction provides a trapdoor unlinkability property for free.

In case of loose coupling, it seems nearly impossible to achieve a full fledged trapdoor privacy [28], as this would require to provide a non reusable trapdoor to prevent the broker from correlating them. To generate such non reusable trapdoor, interaction between publishers and subscribers would be needed, which would break the loose coupling property of publish/subscribe systems.

c: KEYWORD SEMANTIC SECURITY

In this experiment, we aim to show that malicious brokers cannot guess which topic is currently targeted. In practice, the adversary picks a challenge message M, and two associated challenge topics T_0 and T_1., and receives an encryption of M associated with one of the Topics T_b. He succeeds if he manages to guess the bit b corresponding to the targeted topic with non-negligible advantage. (We also let the adversary request trapdoors for topics different from T_0, T_1). In the original ABKS-UR, this would mean our construction is semantically secure against chosen keyword attack under a selective ciphertext policy model (IND-sCP-CKA).

IV. FRAMEWORK DETAILS

The goal of our security framework for topic-based pub/sub systems is to allow:

- Publishers to encrypt the payload of a publication with CP-ABE.
- Publishers to encrypt the topics of publications and subscribers to ensure the privacy of their subscription topics with ABKS-UR.
- Publishers to authenticate publications with ABS.

Our framework is divided into seven parts: System Setup, New Publisher Enrollment, New Subscriber Enrollment, Publication, Subscription, Matching and User Revocation. Since we use our modified version of the Sun *et al.*-ABKS-UR to make it suitable for use in a distributed environment without decreasing the level of security, our modifications of the scheme are detailed in Section V. For the other schemes, the algorithms are used as in the original schemes.

A. SYSTEM SETUP

When the system is set up for the first time, the TA runs the Setup algorithm for each scheme (CP-ABE, ABKS-UR and ABS) and generates PK_{ABE}, MK_{ABE}, $PK_{ABKS-UR}$, $MK_{ABKS-UR}$, PK_{ABS}, MK_{ABS}. It makes the public keys available and keeps the master secret keys confidential.

B. NEW PUBLISHER ENROLLMENT

When a publisher wants to join the system, he asks the TA to generate his ABS secret key (SK_{ABS}) associated with a set of attributes α by calling the ABS Extract algorithm.

C. NEW SUBSCRIBER ENROLLMENT

When a subscriber wants to join the system, he asks the TA to generate his secret keys (SK_{ABE} and $SK_{ABKS-UR}$) associated with a set of attributes $\mathcal{A}$ by calling the ABE and ABKS-UR KeyGen algorithms.

A short summary of the cryptographic keys provided by the TA is presented in Figure 2.

D. PUBLICATION

When a publisher wants to send a publication, he first uses the EncIndex algorithm to encrypt the topic with ABKS-UR to produce an encrypted index. Then, he uses the Encrypt algorithm of ABE to encrypt the payload. Both schemes use the same access structure. The publication (D, c) is then sent to the broker with an optional signature σ generated with the Sign algorithm of ABS on both the encrypted message and the associated topic. As both topics and data are encompassed in σ, this protects against ciphertext replacement attacks even if the ABE and the ABKS-UR schemes are not firmly bound.

E. SUBSCRIPTION

When a subscriber wants to subscribe to a topic, he has to generate a trapdoor using his ABKS-UR secret key and the GenTrapdoor algorithm. He expresses his interests using a set of keywords W to define his subscription topic and generates his trapdoor using his secret key $SK_{ABKS-UR}$. The trapdoor is then sent to the broker as it represents a subscription. When the matching process succeeds at the broker level, the subscriber receives a ciphered publication. He can decrypt it with the Decrypt algorithm using his SK_{ABE} and verify the signature (if there is one) with the Verify algorithm.

F. MATCHING

When the broker receives a publication or a subscription, the ABKS-UR Search algorithm running on the matching engine compares encrypted indexes and trapdoors. If a match is found, the broker routes the related publication to the corresponding subscriber. An overview of the matching process is presented in Figure 3 and detailed in section IV-H.

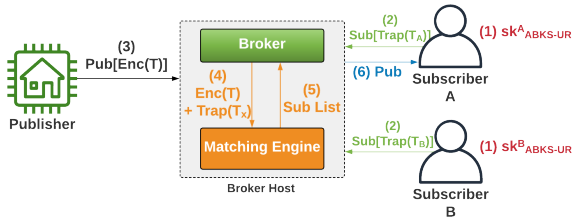


FIGURE 3. Broker and matching process with ABKS-UR.

G. SUBSCRIBER REVOCATION

To revoke a subscriber from the system, the TA uses the ReKeyGen, ReEncIndex and ReKey algorithms to perform a global update of keys for every entity involved in the system. A new master secret key and master public key are generated, and the topics are re-encrypted accordingly. After that, the TA modifies every legitimate secret key concerned (keys that contain the same attributes as those that were owned by the revoked subscriber). The TA also makes the new master public key available to every user.

H. DISCUSSION

To sum up, our security framework for a topic-based pub/sub system uses CP-ABE on the payload and an encrypted matching mechanism of ABKS-UR that allows subscriber revocation in real time, without using a user revocation list or a proxy.

The main steps for a search of encrypted topics are illustrated in Figure 3.

Each *Subscriber* X (A or B) has his own secret key $SK^X_{ABKS-UR}$ (1) that is used to generate a trapdoor $Trap(T_X)$ to subscribe to his topics of interests on the *Broker* $Sub[Trap(T_X)]$(2). After that, subscribers wait for the *Broker* to provide them publications that match their own interests. When a *Publisher* publishes an encrypted topic $Pub[Enc(T)]$ on *Broker* (3), the *Broker* forwards it to the *Matching Engine* (4). The *Matching Engine* runs the Search algorithm on the incoming encrypted topic (i.e. index) of publication and all the subscription trapdoors it has received before. Then, it warns the *Broker* of every subscription that matches (5).

The *Broker* adds them to its routing table and sends each publication to the corresponding subscriber (6). For the sake of our example, in Figure 3, only the interests (i.e. trapdoor) of *Subscriber A* matches the publication topic (i.e. encrypted index) and thus the *SubList* returned by the *Matching Engine* only contains matching between one publication and *Subscriber A*.

When a subscriber tries to access to a topic where his attributes do not match the access structure, the *Broker* acts as if the topic does not exist, which improves confidentiality at the subscription level.

Both CP-ABE and ABKS-UR define users with the same set of attributes to enforce the homogeneity of our proposal.

To provide data authentication, we have added the ABS signature scheme, which also maintains loose coupling between publishers and subscribers. It allows publishers to sign with attributes without revealing their identity.

It can also be noted that ABKS-UR is natively protected against the keyword guessing attack as it provides an access control through attributes. As only users possessing credentials for a given attribute are able to send meaningful queries, the random guessing is transformed into an authentication problem.

V. MODIFICATIONS OF ABKS-UR

In our security framework, ABKS-UR is used to encrypt published topics and to construct subscriptions as trapdoors. It also enables the TA to revoke subscribers in real time. However, integrating the original ABKS-UR by Sun *et al.* [24] in our topic-based pub/sub system would result in a strong coupling between publishers and subscribers because publishers would have to manage users' lists corresponding to their topics (i.e. one list of subscribers per topic). To maintain the loose coupling between publishers and subscribers, we have modified the original scheme to have users' lists managed by the broker. In this section the construction of the original ABKS-UR is detailed, and then our modifications are described. Since we do not modify the part related to user revocation, it is not detailed in this section.

A. ORIGINAL CONSTRUCTION

The original construction scheme is implemented to perform encrypted matching between data owners and data consumers via a cloud server. Each data owner retains a list of consumers who have access to their data. This scheme is organised in five steps: **System setup**, **New user enrollment**, **Secure index generation**, **Trapdoor generation** and **Search**.

System setup: When the server is online, a TA sets the system up with a universe of n attributes $\mathcal{N} = \{1, \ldots, n\}$ and performs the following operations:

- Select a set $\{t_1, \ldots, t_{3n}\}$ of $3n$ random elements $\in \mathbb{Z}_p$.
- Define a *collision resistant keyed hash function* H with an associated secret key.
- Set $T_k = g^{t_k}$ for $k \in \{1, \ldots, 3n\}$ with $g \in \mathbb{G}$.
- Compute $Y = e(g, g)^y$ with $y \xleftarrow{\$} \mathbb{Z}_p^*$.

The TA publishes the master public key $PK = (ver, g, Y, T_1, \ldots, T_{3n})$ and retains the master secret key $MK = (ver, y, t_1, \ldots, t_{3n})$, where ver is the key version (initially set to 1).

New user enrollment: When a consumer identified by ID_f wants to join the system, he asks the TA to generate a secret key (SK) associated with a set of attributes $\mathcal{A}$. The TA proceeds as follows:

- Select a random $x_f \in \mathbb{Z}_p$ as a new MK component, and computes the PK component $X_f = Y^{x_f}$.
- Select random r_i for $i \in \mathcal{N}$ and compute $r = \sum_{i=1}^{n} r_i$.
- Set $\widehat{K} = g^{y-r}$ and if $i \in \mathcal{A}$ then set $K_i = g^{\frac{r_i}{t_i}}$, else set $K_i = g^{\frac{r_i}{t_{n+i}}}$.
- Set $F_i = g^{\frac{r_i}{t_{2n+i}}}$.

The secret key $SK = (ver, x_f, \widehat{K}, \{K_i, F_i\}_{i \in \mathcal{N}})$ is sent to the consumer by the TA.

The data owner computes $D_f = X_f{}^{-s}$ with $s \xleftarrow{\$} \mathbb{Z}_p^*$ and adds the tuple (ID_f, D_f) to his users list.

Secure index generation: When a data owner wants to publish his data, defined by a keyword set W, he has to:

- Define an access structure GT that contains positive and negative attributes separated with *AND* clauses.
- Set $\widehat{D} = g^s$, $\widetilde{D} = Y^s$.
- For $i \in GT$, set $D_i = T_i^s$ for positive attributes and $D_i = T_{n+i}^s$ for negative ones.
- Let $D_i = T_{2n+i}^s$ for $i \notin GT$.
- Assume a positive attribute i' and set $D_{i'}$ as $D_{i'} = T_{i'}^{\frac{s}{\prod_{w_j \in W} H(w_j)}}$, where H is a collision-resistant hash function $\{0, 1\}^* \to \mathbb{Z}_p$.

We now have the following encrypted index D:

$$D := (ver, GT, \widehat{D}, \widetilde{D}, \{D_i\}_{i \in \mathcal{N}}). \tag{1}$$

Trapdoor generation: When a data consumer wants to make a request to the server, he has to generate a trapdoor Q with his secret key SK and a keyword set W. Then, he proceeds as follows:

- Select a random $u \in \mathbb{Z}_p$, compute $\widehat{Q} = \widehat{K}^u$ and $\widetilde{Q} = u + x_f$.
- For $i \in \mathcal{N}$, set $Q_i = K_i^u$ and $Qf_i = F_i^u$.
- At the index i', compute $Q_{i'} = (K_{i'}^{\prod_{w_j \in W} H(w_j)})^u$ and $Qf_{i'} = (F_{i'}^{\prod_{w_j \in \mathcal{W}} H(w_j)})^u$.

The trapdoor is then $Q = (ver, \widehat{Q}, \widetilde{Q}, \{Q_i, Qf_i\}_{i \in \mathcal{N}})$

Search: When the server receives a trapdoor, it verifies the matching with existing indexes. To achieve that, it compares an encrypted index D with a trapdoor Q. Specifically, it verifies that the ID_f is in the users list. If not, the consumer is not authorized to access this data. Those checks are done to avoid unnecessary computation if the user is not in the system.

Then, the server checks the trapdoor version:

- If the version of Q is less than the version of D, it outputs $\perp$.

- If the version of Q is greater than the version of D, the lazy re-encryption mechanism is applied.
- If the version of Q is equal to the version of D, the matching process can proceed.

For each attribute i, if $i \in GT$, the matching engine computes $e(D_i, Q_i)$; otherwise it computes $e(D_i, Qf_i)$. Then, it verifies:

$$\widetilde{D}^{\widetilde{Q}} \cdot D_f \stackrel{?}{=} e(\widehat{D}, \widehat{Q}) \cdot \prod_{i=1}^{n} e(D_i, Q_i^*) \tag{2}$$

where $Q_i^* = Q_i$ if $i \in GT$ and $Q_i^* = Qf_i$ otherwise.

If the equation holds, the trapdoor matches the associated index and the consumer can receive the data.

B. MODIFICATIONS FOR DISTRIBUTED ENVIRONMENTS

To make ABKS-UR work in a distributed environment like a pub/sub architecture, we have modified four phases: **System setup**, **Secure index generation**, **New user enrollment** and **Search**.

System setup: We change the fourth step so that the pairing is removed in the system setup. We now compute Y so that $Y = g^y$.

Secure index generation: We introduce a pairing-based cryptographic identifier called DT so that $DT = e(g, \widetilde{D})$. DT is used in the **Search** step of the *Matching Engine*. Equation 1 is modified so that D now also contains DT: $D := (ver, GT, \widehat{D}, \widetilde{D}, DT, \{D_i\}_{i \in \mathcal{N}})$.

New user enrollment: The first step is modified and the PK component is now computed as follows: $X_f = g^{-x_f}$. At the end of the enrollment phase, we change the computation of D_f such that: $D_f = e(\widetilde{D}, X_f)$. The tuple (ID_f, D_f) is sent to the broker and added to its users list.

Search: The step is changed so that Equation 2 is now expressed as $DT^{\widetilde{Q}} \cdot D_f \stackrel{?}{=} e(\widehat{D}, \widehat{Q}) \cdot \prod_{i=1}^{n} e(D_i, Q_i^*)$

C. CORRECTNESS OF THE MODIFIED SCHEME

Provided that subscriber has the attributes that match the access structure and the keywords match, then:

- The subscriber possesses: $x_f, \widehat{K} = g^{y-r}, K_i = g^{\frac{r_i}{t_i}}, F_i = g^{\frac{r_i}{t_{2n+i}}}$
- For a specific subscriber, the TA generates $D_f = e(g, g)^{yx_f}$, which is sent to the broker
- the encrypted index generated by the publisher contains: $\widehat{D} = g^s, \widetilde{D} = Y^s, D_i = T_i^s, D_{i'} = T_{i'}^{\frac{s}{\prod_{w_j \in W} H(w_j)}}$

The *Matching Engine* will then test for matching:

- first, using the trapdoor $\widetilde{D}^{\widetilde{Q}}$ and D_f:

$$\begin{aligned} DT^{\widetilde{Q}} \cdot D_f &= e(g, \widetilde{D})^{u+x_f} \cdot e(\widetilde{D}, X_f) \\ &= e(g, g)^{y \cdot s \cdot (u+x_f)} \cdot e(g, g)^{-y \cdot s \cdot x_f} \\ &= e(g, g)^{s \cdot u \cdot y} \end{aligned}$$

- then, using the trapdoors $\widehat{Q}, Q_i^*$ generated by the subscriber

$$\begin{aligned} e(\widehat{D}, \widehat{Q}) \cdot \prod_{i=1}^{n} e(D_i, Q_i^*) &= e(g^s, g^{u \cdot y - u \cdot r}) \cdot \prod_{i=1}^{n} e(g, g)^{s \cdot u \cdot r_i} \\ &= e(g, g)^{s \cdot u \cdot y - s \cdot u \cdot r} \cdot e(g, g)^{s \cdot u \cdot r} \\ &= e(g, g)^{s \cdot u \cdot y} \\ &= DT^{\widetilde{Q}} \cdot D_f \end{aligned}$$

A trapdoor generated with the relevant keywords and the attributes that match the access structure for an encrypted index will find a match, which proves that our modifications of ABKS-UR are correct.

VI. PROOF OF THE KEYWORD SEMANTIC SECURITY (KSS)

The simulator receives a TDH challenge, $A = g^a, B = g^b, C = g^c, Z \in \mathbb{G}$.

The adversary sends to the simulator the challenge access structure $\mathrm{GT} = \bigwedge_{i \in I} i'$, and an attribute set $\mathcal{A}$ from the adversary.

A. SETUP

The simulator sets $PK = e(A, B)$ which implicitly sets $MK = ab$.

The simulator picks a random $x \in \mathbb{Z}_p$, and sets $Y' = PK^x$.

For each $i \in \mathcal{N}$, he picks random $\alpha_i, \beta_i, \gamma_i \in \mathbb{Z}_p$ and

For $i \in I$: If $i' = i, T_i = g^{\alpha_i}, T_{n+i} = B^{\beta_i}, T_{2n+i} = B^{\gamma_i}$; $T_i = B^{\alpha_i}, T_{n+i} = g^{\beta_i}, T_{2n+i} = B^{\gamma_i}$ otherwise.

For $i \notin I, T_i = B^{\alpha_i}, T_{n+i} = B^{\beta_i}, T_{2n+i} = g^{\gamma_i}$;

For each attribute set Φ^j in $\mathcal{A}$ given by the adversary, the simulator generates $rk^{(j)}$ honestly, and sets it to 1 otherwise.

B. PHASE 1

If an adversary sends a query for a keyword w, and a set of attributes S that does not comply with GT, then under the collision-resistance property of the hash function there exists an index k such that $k \in S \Delta \mathcal{A}$ (where Δ is the symmetric difference). Similarly to [24], we can leverage this index to introduce a perfectly random value in the part Qf_k of the trapdoor.

C. CHALLENGE

Upon receiving the challenge keywords w_0, w_1 from the adversary, the simulator picks a random bit b, and encrypts w_b following the challenge access structure GT. The simulator can then output $\widehat{D} = C, \widetilde{D} = Z$ and $D_f = e(Z^{-x}, g)$.

This is the crucial difference from the original proof, since at this step their Z lived in $\mathbb{G}_1$ while ours is in $\mathbb{G}$.

D. PHASE 2

The simulator proceeds to answer the post-challenge queries exactly as before (aborting if queried on the challenge access structure GT).

E. WRAPPING UP

Now, the adversary outputs its guess for the value b.

If Z is honestly generated, then the adversary sees the real game.

If Z is random, then the adversary's view is indistinguishable from random.

Hence the advantage of an adversary against the keyword semantic security is lower than the advantage of the simulator against TDH ($\mathsf{Adv}^{kss} \leq \mathsf{Adv}^{\mathsf{TDH}}$).

VII. PERFORMANCES EVALUATION

In this section, both theoretical complexity analysis and the experimental evaluation based on real implementation of the proposed security framework for a topic-based pub/sub system are provided for the sake of performance assessment. Since the main originality of the framework relies on the ABKS-UR construction for a topic-based pub/sub system, a large part of the evaluation focuses on this specific component.

A. COMPLEXITY ANALYSIS

In Table 1, we provide a comparison of the theoretical complexity of the original ABKS-UR scheme [24] with our modified version, assuming n attributes. We note the pairing operations P, the group exponentiation E in $\mathbb{G}$ and E_1 in $\mathbb{G}_1$, the group multiplications M in $\mathbb{G}$ and M_1 in $\mathbb{G}_1$. We add a pairing in the **New user enrollment** and in the **Secure index generation** steps, to suppress one in the **System setup** step. Since those operations are done a limited number of times, the overall complexity does not change significantly.

B. EXPERIMENTAL RESULTS

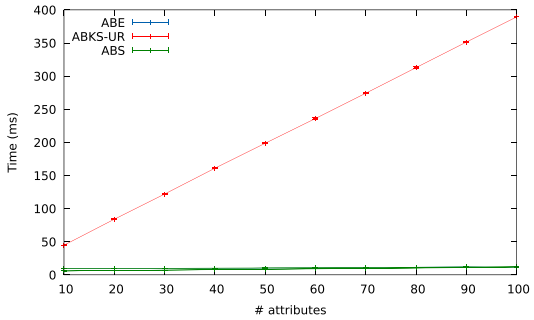
The aim of this section is to highlight the practicability of the proposed security framework in an IoT context in terms of computation time. For this purpose, subscribers, MQTT broker and TA were deployed on virtual machines (VM). Each VM was configured with 4 GB of dedicated RAM running on a host computer with a 2.6 GHz Intel Core i7 processor. The publisher was implemented on a Raspberry Pi 3 model B. It simulated, as a gateway, the data collection from multiple IoT devices and published them on a regular basis. Since it is assumed that the gateway is properly powered, the energy consumption is not evaluated in this paper. The MQTT broker was implemented with the open source broker HBMQTT [29] and was modified at the matching level so that the proposed distributed ABKS-UR construction was used. For both publishers and subscribers, we used the Eclipse Paho MQTT Python library [30] to communicate (i.e. to perform publication and subscription) with the broker. Finally, the cryptographic operations were implemented with the Charm-crypto python library [31]. More specifically, we used the Charm Waters'09 [27] implementation for CP-ABE. Both the new ABKS-UR construction and the ABS scheme were implemented specifically in Python 3 using the Charm curve *SS512*, which represents a symmetric curve with a 512-bit base field [32]. For performance purposes, in all ABE schemes, ABE is only used to cipher an AES key, which is then used to protect data (i.e. in our case the publication payload) and thus we proceeded similarly in our real implementation of the framework. Standard deviation is presented on all figures related to time measurements.

As illustrated Figure 4a, during the System Setup part of the framework, which is run by the TA (VM), the time to generate ABKS-UR master keys increases linearly with the number of attributes in the system to reach 390 *ms* for 100 attributes, which is suitable for a real-world scenario, where the number of attributes will likely be lower. It can also be seen that master key generation for both CP-ABE and ABS is achieved in 10 *ms* each, which is negligible.

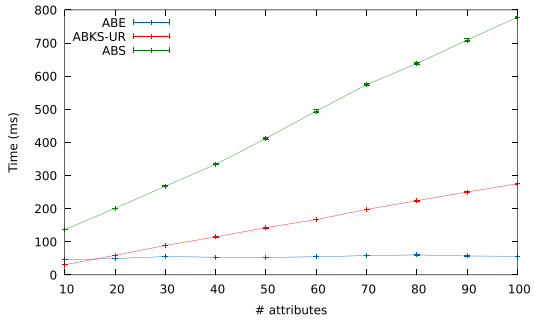
As illustrated in Figure 4b, during the New Subscriber Enrollment part of the framework (executed by the TA), the generation of 100 attributes of the ABKS-UR secret key takes 275 *ms* while the CP-ABE secret key takes 55 *ms*. During the New Publisher Enrollment part of the framework, only the ABS secret key has to be computed, which takes 780 *ms*. These operations are done only once per user.

In the Publication part of the framework (performed by the Raspberry Pi), we evaluate the computation time for: (1) payload encryption with CP-ABE, (2) topic encryption with ABKS-UR and (3) signature with ABS. As illustrated in Figure 4c, (1) has a constant computation time that is around 1100 *ms* (only attributes in the access structure are involved in the encryption, and not the whole universe). It can also be seen that (2) and (3) increase linearly with respect to the number of attributes in the universe. For instance, for 100 attributes in the universe (2) takes around 1900 *ms* and (3) around 12000 *ms*, with the signature being optional. As illustrated in Figure 4d, the total size of the publication is around 37 *kB* (4 *kB* for the ABE ciphertext, 15 *kB* for the encrypted topic with ABKS-UR and 18 *kB* for the ABS signature) for 100 attributes in the universe, with the ABKS-UR encrypted topic having a constant size regardless of the number of attributes in the universe.

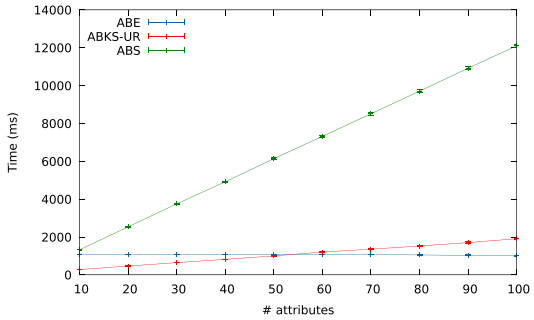
We also evaluate the computation time and the size of the publication with 100 attributes in the universe and a variable number of attributes in the access structure. As illustrated in Figure 4e, it takes 9360 *ms* to perform the ABE encryption, 1950 *ms* for the ABKS-UR topic encryption and 12000 *ms* for the ABS signature with an access structure containing 100 attributes. These values are considerable but it should be noted that they are for an access structure of 100 attributes, which is not realistic for most real use cases. With a more credible access structure containing 20 attributes, the computation time values are respectively 2060 *ms*, 1860 *ms* and 12000 *ms*, with the signature being optional. As illustrated in Figure 4f, the total size of the publication is around 60 *kB* (30 *kB* for the ABE ciphertext, 15 *kB* for the encrypted topic with ABKS-UR and 15 *kB* for the ABS signature). The ABKS-UR encrypted index has a constant size regardless of the number of attributes, whereas the signature size decreases as the number of attributes in the policy increases. This



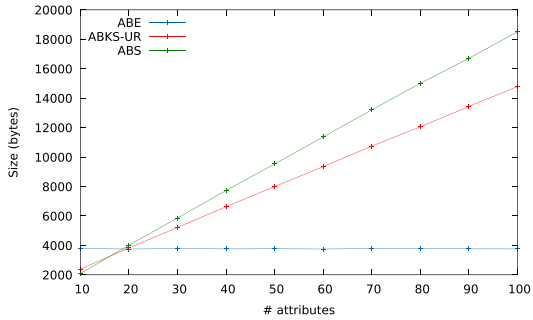
(a) System Setup



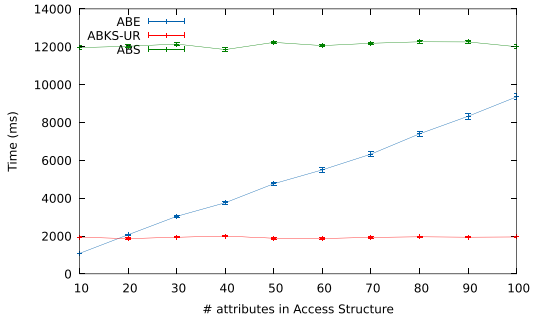
(b) New Subscriber and Publisher Enrollment



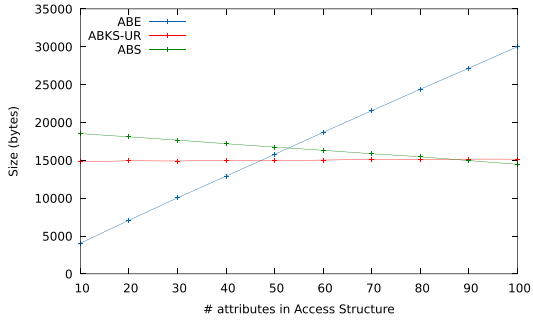
(c) Publication time with variable number of attributes in the universe and an access structure of 10 attributes



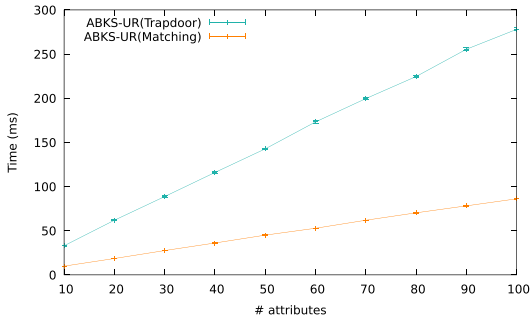
(d) Publication size with variable number of attributes in the universes and an access structure of 10 attributes



(e) Publication time with variable number of attributes in the access structure and a universe of 100 attributes



(f) Publication size with variable number of attributes in the access structure and a universe of 100 attributes



(g) ABKS-UR Trapdoor generation and Matching

FIGURE 4. Framework performance evaluation.

decreasing signature size can be explained because a part of the signature represents every attribute in the universe that is not used to sign the message. Therefore, the more attributes are used, the smaller the number of attributes to embed in the signature.

For the Subscription part of the framework (run by a VM), as illustrated in Figure 4g, the trapdoor generation time increases linearly with the number of attributes in the universe and takes 280 *ms* for 100 attributes as presented in Figure 4g. This operation has to be done only once per topic.

TABLE 1. Comparison of theoretical complexity between ABKS-UR and our modified version.

Step	ABKS-UR [24]	Our modified version
System setup	$3nE + E_1 + P$	$(3n+1)E$
New user enrollment	$(2n+1)E + 2E_1$	$(2n+2)E + P$
Secure index generation	$(n+1)E + E_1$	$(n+1)E + E_1 + P$
Trapdoor generation	$(2n+1)E$	$(2n+1)E$
Search	$(n+1)P + (n+2)M_1 + E_1$	$(n+1)P + (n+2)M_1 + E_1$

Finally at the broker level, as illustrated in Figure 4g, the time for the Matching part of the framework (run by a VM) also increases linearly with the number of attributes in the universe as presented in Figure 4g. The overall performance is satisfactory as it only takes 85 *ms* with 100 attributes, which makes it suitable for real-world applications.

VIII. CONCLUSION

In this paper, we propose an attribute-based security framework to enhance security and privacy for topic-based pub/sub systems that satisfies the constraints of real IoT devices and deployment scenarios. The framework combines, in a consistent manner, several existing attribute-based cryptographic solutions (CP-ABE and ABS) and the ABKS-UR scheme that we modified to make it suitable for distributed environments while keeping the necessary loose coupling between publishers and subscribers. All schemes thus use the same universe of attributes, which is satisfactory and homogeneous both from a cryptographic standpoint and a real deployment perspective.

Our framework ensures subscription confidentiality, publication confidentiality and payload confidentiality even when facing a realistic honest-but-curious model for the broker. It also allows the authentication of publications and revocation of malicious subscribers with perfect forward secrecy.

One of our main contributions is the modifications done on the ABKS-UR scheme to make the private matching algorithm suitable for pub/sub systems (and other distributed systems), we have demonstrated the correctness of the modified scheme. We have also provided a formal security proof that our modifications do not decrease the security of the original scheme.

To illustrate that our framework is suitable for real-world IoT deployments, we undertook a complexity analysis and conducted comprehensive experiments to evaluate the performance of the framework implementation.

However, a limitation to the current framework lies in the expression of subscriptions that rely on strict equality. As future work, our goal is to allow subscribers to express their interests with inequalities while still relying on an encrypted matching system that allows user revocation in real time. Another improvement will be to add a pre-filtering algorithm to avoid unneeded operations before the encrypted matching operation. Moreover, we plan to evaluate the framework performances (e.g. in terms of memory consumption, bandwidth, computational complexity in distributed environment, power consumption) in a real environment with a high number of devices.

REFERENCES

[1] A. Banks and R. Gupta, "MQTT version 3.1. 1," *OASIS Standard*, vol. 29, p. 89, Oct. 2014.

[2] Online. *Kafka*. Accessed: Dec. 9, 2020. [Online]. Available: https://kafka.apache.org/

[3] V. Shnayder, B.-R. Chen, K. Lorincz, T. R. F. F. Jones, and M. Welsh, "Sensor networks for medical care," in *Proc. 3rd Int. Conf. Embedded Netw. Sensor Syst. (SenSys)*. New York, NY, USA: Association for Computing Machinery, 2005, p. 314, doi: 10.1145/1098918.1098979.

[4] A. Geissbühler, J. F. Grande, R. A. Bates, R. A. Miller, and W. W. Stead, "Design of a general clinical notification system based on the publish-subscribe paradigm," in *Proc. AMIA Annu. Fall Symp.* Bethesda, MD, USA: American Medical Informatics Association, 1997, p. 126.

[5] A. Banks, E. Briggs, K. Borgendale, and R. Gupta, *MQTT Version 5.0*, OASIS Standard, Mar. 2019, p. 137. [Online]. Available: https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.docx

[6] A. Belokosztolszki, D. M. Eyers, P. R. Pietzuch, J. Bacon, and K. Moody, "Role-based access control for publish/subscribe middleware architectures," in *Proc. 2nd Int. Workshop Distrib. Event-Based Syst. (DEBS)*, 2003, pp. 1–8.

[7] B. Shand, P. Pietzuch, I. Papagiannis, K. Moody, M. Migliavacca, D. M. Eyers, and J. Bacon, "Security policy and information sharing in distributed event-based systems," in *Reasoning Event-Based Distrib. Syst.* Berlin, Germany: Springer, 2011, pp. 151–172.

[8] L. Fiege, A. Zeidler, A. Buchmann, R. Kilian-Kehr, G. Mühl, and T. Darmstadt, "Security aspects in publish/subscribe systems," in *Proc. 3rd Int. Workshop Distrib. Event-Based Syst. (DEBS)*. Edinburgh, Scotland: IET, Jul. 2004, pp. 44–49.

[9] A. Fongen and F. Mancini, "Identity management and integrity protection in publish-subscribe systems," in *Proc. IFIP Work. Conf. Policies Res. Identity Manage.* Berlin, Germany: Springer, 2013, pp. 68–82.

[10] Z. Miklós, "Towards an access control mechanism for wide-area publish/subscribe systems," in *Proc. 22nd Int. Conf. Distrib. Comput. Syst. Workshops*, 2002, pp. 516–521.

[11] M. Ion, G. Russello, and B. Crispo, "Design and implementation of a confidentiality and access control solution for publish/subscribe systems," *Comput. Netw.*, vol. 56, no. 7, pp. 2014–2037, May 2012.

[12] W. Chen, J. Jiang, and N. Skocik, "On the privacy protection in publish/subscribe systems," in *Proc. IEEE Int. Conf. Wireless Commun., Netw. Inf. Secur.*, Jun. 2010, pp. 597–601.

[13] L. Opyrchal and A. Prakash, "Secure distribution of events in content-based publish subscribe systems," *Ann Arbor*, vol. 1001, pp. 42122–48109, Aug. 2001.

[14] Y. Yan, Y. Huang, G. C. Fox, S. Pallickara, M. E. Pierce, A. Kaplan, and A. E. Topcu, "Implementing a prototype of the security framework for distributed brokering systems," in *Security and Management*. 2003, pp. 212–218.

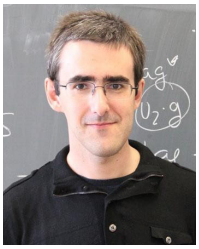
[15] C. Raiciu and D. S. Rosenblum, "Enabling confidentiality in content-based publish/subscribe infrastructures," in *Proc. Securecomm Workshops*, Aug. 2006, pp. 1–11.

[16] G. Padmavathi and S. Annadurai, "A security framework for content-based publish–subscribe system," *Electron. Commerce Res. Appl.*, vol. 5, no. 1, pp. 78–90, 2006.

[17] C. Esposito and M. Ciampi, "On security in publish/subscribe services: A survey," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 2, pp. 966–997, 2nd Quart., 2015.

[18] T. H. Yuen, W. Susilo, and Y. Mu, "Towards a cryptographic treatment of publish/subscribe systems," in *Proc. Int. Conf. Cryptol. Netw. Secur.* Berlin, Germany: Springer, 2010, pp. 201–220.

[19] M. Suzuki, "Sanitizable signature with secret information," in *Proc. Symp. Cryptogr. Inf. Secur. (SCIS)*, Jan. 2006, pp. 4A1–4A2.
[20] D. Thatmann, S. Zickau, A. Forster, and A. Kupper, "Applying attribute-based encryption on publish subscribe messaging patterns for the Internet of Things," in *Proc. IEEE Int. Conf. Data Sci. Data Intensive Syst.*, Dec. 2015, pp. 556–563.
[21] M. Ion, G. Russello, and B. Crispo, "Providing confidentiality in content-based publish/subscribe systems," in *Proc. Int. Conf Secur. Cryptogr. (SECRYPT)*, 2010, pp. 1–6.
[22] C. Dong, G. Russello, and N. Dulay, "Shared and searchable encrypted data for untrusted servers," *J. Comput. Secur.*, vol. 19, no. 3, pp. 367–397, May 2011.
[23] D. M. Freeman, "Converting pairing-based cryptosystems from composite-order groups to prime-order groups," in *Proc. 29th Annu. Int. Conf. Theory Appl. Cryptograph. Techn., Adv. Cryptol. (EUROCRYPT)*, in Lecture Notes in Computer Science, vol. 6110, H. Gilbert, Ed., Monaco, Europe. Cham, Switzerland: Springer, May/Jun. 2010, pp. 44–61.
[24] W. Sun, S. Yu, W. Lou, Y. T. Hou, and H. Li, "Protecting your right: Attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2014, pp. 226–234.
[25] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Proc. Workshop Theory Appl. Cryptograph. Techn.* Berlin, Germany: Springer, 1984, pp. 47–53.
[26] J. Li, M. H. Au, W. Susilo, D. Xie, and K. Ren, "Attribute-based signature and its applications," in *Proc. 5th ACM Symp. Inf., Comput. Commun. Secur. (ASIACCS)*, 2010, pp. 60–69.
[27] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Proc. Int. Workshop Public Key Cryptogr.* Berlin, Germany: Springer, 2011, pp. 53–70.
[28] A. Arriaga, Q. Tang, and P. Ryan, "Trapdoor privacy in asymmetric searchable encryption schemes," in *Proc. 7th Int. Conf. Cryptol. Africa, Prog. Cryptol. (AFRICACRYPT)*, in Lecture Notes in Computer Science, vol. 8469, D. Pointcheval and D. Vergnaud, Eds., Marrakesh, Morocco. Cham, Switzerland: Springer, May 2014, pp. 31–50, doi: 10.1007/978-3-319-06734-6_3.
[29] Online. (2015). *HBMQTT*. [Online]. Available: https://github.com/beerfactory/hbmqtt
[30] Online. (2018). Paho-MQTT, MQTT-SN Software. The Eclipse Foundation. Accessed: Dec. 9, 2020. [Online]. Available: https://www.eclipse.org/paho/
[31] J. A. Akinyele, C. Garman, I. Miers, M. W. Pagano, M. Rushanan, M. Green, and A. D. Rubin, "Charm: A framework for rapidly prototyping cryptosystems," *J. Cryptograph. Eng.*, vol. 3, no. 2, pp. 111–128, Jun. 2013.
[32] Y. Rouselakis and B. Waters, "Efficient statically-secure large-universe multi-authority attribute-based encryption," in *Financial Cryptography and Data Security*, R. Böhme and T. Okamoto, Eds. Berlin, Germany: Springer, 2015, pp. 315–332.



OLIVIER BLAZY received the Ph.D. degree in proofs of knowledge and their application to blind signatures and authenticated key exchange, in 2012. He is currently an Associate Professor with the University of Limoges, France. Since then, he leads the master level security program in Limoges, and conduct research on the areas of implicit cryptography, and identity-based communications. He is leading the ANR project: IDFix on ID-based communications.

EMMANUEL CONCHON received the M.Sc. and Ph.D. degrees in wireless communication from the "Institut National Polytechnique de Toulouse (INPT)," Toulouse, France, in 2002 and 2006, respectively. In September 2008, he joined the Champollion University as an Associate Professor and IRIT (Institut de Recherche en Informatique de Toulouse) as a Researcher. Since September 2015, he is an Associate Professor with the University of Limoges and a Researcher at XLIM. His research interests include security solutions for wireless networks, context-aware systems, and secure middleware solutions for health applications.

MATHIEU KLINGLER received the master's degree "Cryptis" from the University of Limoges, in September 2018. He is currently pursuing the Ph.D. degree with the University of Limoges. His main research interests include the creation of frameworks addressing the privacy issues in publish/subscribe architecture, attribute-based schemes applied to decentralized architecture, and implementation of diverse cryptosystems for performances evaluations. He started his thesis on October 2018.

DAMIEN SAUVERON received the M.Sc. and Ph.D. degrees in computer science from the University of Bordeaux, France. He has been an Associate Professor with Habilitation with the XLIM Laboratory (UMR CNRS 7252, University of Limoges, France, since 2006. He is the Dean of the Faculty of Science and Technology, University of Limoges. Since 2011, he has been a member of the CNU 27, the National Council of Universities (for France). He has been the Chair of IFIP WG 11.2 Pervasive Systems Security, since 2014, having previously been appointed vice-chair of the working group. His research interests are related to smart card applications and security (at hardware and software level), RFID/NFC applications and security, mobile network applications and security (particularly UAV), sensor network applications and security, Internet of Things (IoT) security, cyber-physical systems security, and security certification processes. In December 2013, the General Assembly of IFIP (International Federation for Information Processing) awarded him the IFIP Silver Core award for his work. He has been involved in more than 100 research events in a range of capacities (including PC Chair, General Chair, Publicity Chair, Editor/Guest Editor, Steering Committee Member, and Program Committee Member). He has served as an external reviewer for several Ph.D. thesis in foreign countries and in France.