2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom) | 978-1-6654-1658-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/TRUSTCOM53373.2021.00037

MagikCube: Securing Cross-Domain Publish/Subscribe Systems with Enclave

Shuran Wang, Dahan Pan, Runhan Feng and Yuanyuan Zhang*
Shanghai Jiao Tong University
Shanghai, China
Email: {wshuran, dhpan98, fengrunhan, yyjess}@sjtu.edu.cn

***Abstract*—The publish/subscribe(pub/sub) is an asynchronous messaging service or content distribution framework. For the idempotency it provides, pub/sub diagram is an efficient solution for large-scale content distributing systems, thus it is widely used in stock exchange systems or e-Health content sharing systems. Some wide-area applications require cross-domain pub/sub service, making it a natural choice to deploy on the public cloud. However, it would bring about security and privacy issues. Recent research proposes security enhancements to prevent thefts, such as searchable data encryption and attribute-based encryption, which allow the matching process to perform encrypted matching without learning the content of the publications and subscriptions. Besides the considerable performance loss, they could not resist the *collusion attacks*. If the malicious brokers collude with a malicious publisher or subscriber in a cross-domain environment, they can still infer the subscriptions of benign subscribers.**

We propose the MagikCube framework that provides confidentiality and integrity of the contents and also protects the privacy of the publishers and subscribers in cross-domain scenarios. Moreover, MagikCube can also resist the collusion attacks from malicious brokers in a cross-domain environment. It achieves these security goals by dynamically selecting and placing the sensitive data and some necessary components in enclaves protected by trusted hardware such as Intel SGX. Our experiment result shows that, compared with the baseline model, MagikCube does not introduce much overhead loss when providing better security for all the participants in the pub/sub system.

***Index Terms*—Pub/Sub, Collusion Attack, Intel SGX**

1. Introduction

The publish/subscribe (pub/sub) paradigm is attractive for scalable asynchronous communication among large groups of participants [1]. Compared to traditional point-to-point communications, the pub/sub paradigm enables disseminating data from publishers to interested subscribers without establishing direct contacts between publishers and subscribers. In a typical content-based pub/sub domain, the publishers produce data to various *topics* in the form of publications, and the subscribers register their interesting topics to the broker that performs the matching and routing procedures. The broker maintains the subscriptions registered by subscribers and forward incoming matching publications to interested subscribers. Once the publication matches the subscription, it will be forwarded to the corresponding subscribers.

Some small-scaled facilities implement the single domain pub/sub system, such as a stock exchange or an e-Health system [2]. For example, in the e-Health system, the medical entities (hospitals, clinics, physicians, and pharmacists) would subscribe to the patients' records by their interesting topics. Worth mentioning, those records contain credential information such as patient's name, age, home address, and some others, which are not allowed to leak to unauthorized entities. Meanwhile, some complex applications require a wide-area subscription service so that the publications flow between and through various domains of administrative control [3]. The brokers work under different administrations to cooperate in providing inter-domain communications.

Deployment on the public cloud brings security and privacy challenges to many systems. As the pivot of a pub/sub system, the broker is channeling all the publishers and subscribers. No one can afford a compromised or malicious broker in a critical system. An ideal broker would guarantee the security of the content and the privacy of any participants in its system. Meanwhile, it has to be tamper-resistant to any external attacker by exploiting any software or system vulnerabilities. There have been several prior works to provide information with confidence, consisting of mainly two types of approaches, 1) cryptographic approaches and 2) hardware-based solutions [4].

Cryptography approaches allow brokers to match encrypted subscriptions against encrypted publications without learning their contents [5]. Although these methods can provide confidentiality guarantees for data, they introduce additional overhead in performance or operations. Moreover, these methods cannot resist colluding attack.

In this paper, we present MagikCube, a novel pub/sub framework that protects the privacy of publishers and subscribers in cross-domain environments. MagikCube guarantees the confidentiality and integrity of publications and subscriptions in the untrusted cloud environment with the help of Intel SGX. It protects sensitive messages from the

* *is corresponding author.*

2324-9013/21/$31.00 ©2021 IEEE
DOI 10.1109/TrustCom53373.2021.00037

cloud by executing sensitive processes such as the matching process within Intel SGX enclaves. Furthermore, even if the broker colludes with publishers or subscribers, they cannot learn about innocent subscriber's subscriptions.

Our contributions are summarized as follows:

- We propose MagikCube, a secure pub/sub framework that leverages Intel SGX to secure transmission and processing of publications and subscriptions on untrusted brokers. The broker cannot learn about sensitive publications and subscriptions.
- On one hand, MagikCube prevents the untrusted broker from stealing and leaking sensitive publications and subscriptions. On the other hand, it prevents collusion attacks from malicious brokers and publishers/subscribers in a cross-domain environment.
- MagikCube follows the loosely-coupled property of the pub/sub model. Publishers and subscribers do not communicate directly.

The organization of this article is as follows. We give a brief technical background in Section 2. Next, we present the threat model and design considerations of secure pub/sub systems in Section 3. In Section 4, we present the design of the MagikCube in detail. In Section 5, we provide a security analysis of MagikCube. In Section 6, we report the performance analysis of the prototype of MagikCube. Finally, we conclude in Section 7.

2. Background

2.1. Cross-domain Pub/Sub Architecture

We use the term *domain* to describe an independent unit where a domain manager has responsibility for policy specification [6]. A *domain*, such as a hospital, naturally maps to an independently administered organization that manages its infrastructure and formulating its policies. In a typical pub/sub system, a domain is composed of a *broker* infrastructure that provides routing service and two sets of clients, one of which called *publishers* submit publications to the system and the other of which called *subscribers* receive the publications that match their interest.

We also employ the term *cross-domain* to describe the pub/sub systems in which the publications traverse multiple domains with different trust assumptions [4]. Each domain that is responsible for the information it handles should share information with other domains appropriately. The access policy related to the specific publications is formulated by the owner of the publications, and then the pivot of the domain stores and executes the access policy related to the publications.

2.2. Collusion Attacks

The concept of collusion attacks in pub/sub system was first introduced in [7]. In a collusion attack, compromised subscribers collude together with untrusted brokers against honest subscribers to learn their interests. For instance, a group of users subscribes to e-health information containing encrypted medical records. Suppose that one of these users is compromised and colludes with an untrusted broker. Due to the one-to-many nature of the pub/sub paradigm, the broker correlates publications with a set of matched subscriptions. Among the matched subscriptions, some are defined by compromised subscribers and some by honest subscribers. Since the untrusted broker learns the match result when the compromised subscribers receive and decrypt matched encrypted publications, attackers can infer honest subscriptions which are matched with the same publications, and therefore the honest subscribers' interest is exposed to the attackers. As a result, traditional encryption techniques cannot defend against the collusion attack launched by untrusted brokers and compromised subscribers.

However, in the real world, Internet-scale pub/sub systems are likely to span multiple administrative domain [8] and collusion attacks may be launched by two parties of different domains. Considering the actual situation, we extend the collusion attack scenario into single-domain scenarios and cross-domain scenarios in this paper.

In a single-domain scenario, publications are shared in an independent administrative unit. For example, a hospital can be regarded as a domain, and medical records are shared among entities of the hospital, such as doctors, nurses, and accountants. If one publisher or one subscriber is compromised by an adversary, he can collude with the untrusted broker to infer the honest subscriber's interest. More specifically, as shown in Figure 1, the compromised publishers keep publishing different publications to the broker. Then the broker matches the incoming specific publications with a stored subscription. Although the publications and subscriptions are encrypted, the untrusted broker knows the match result, that is, knows whether the specific publications meet the constraints of the subscription. In this way, the adversary can infer the subscription of the innocent subscriber according to the specific publications and the match result. Similarly, when a compromised subscriber colludes with the untrusted broker, he can inject fake subscriptions and infer the encrypted interests of innocent subscribers by observing whether they match the same publications as the injected ones. Thus, the subscription privacy of the honest subscriber is leaked to the adversary even if the encryption technique is used in the pub/sub system in a single-domain scenario.

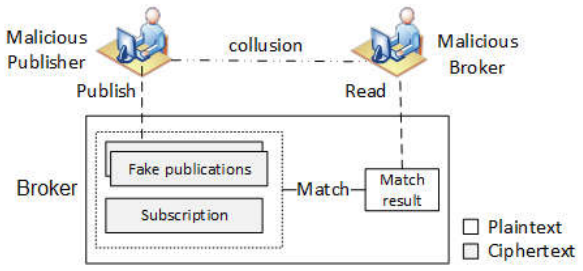


Figure 1. A collusion attack example using encryption matching protection methods in a single-domain scenario.

In a cross-domain scenario, publications are shared over several independent administrative units, and trust assumptions over each domain are different. The publisher often publishes access control policies with the publications to restrict access to the publications by entities in other domains. For example, the name information in the medical records published by the hospital should not be disclosed to the research institute in a cross-domain pub/sub system. Thus, the publisher formulates an access control policy for medical records to restrict specific entities' access to the name attribute, and the policy is executed by the broker. For the convenience of description, we use domain *A* and domain *B* to represent two different domains, respectively. If one publisher in domain *A* is compromised by an adversary, he can collude with the untrusted broker in domain *B* to infer the honest subscriber's interest. More specifically, As shown in Figure 2, the compromised publisher first publishes fake encrypted publications and encrypted access policies to the local broker. Then the local broker transmits the publications and policies to domain *B*. Next, the broker in domain *B* matches incoming publications/policies with stored subscriptions/identity attributes of an innocent subscriber. Even though all the sensitive information (i.e., subscriptions and identity attributes) of subscribers stored on the broker is encrypted, the adversary can learn the matching result from the untrusted broker. Since the adversary knows the plaintext of the fake publications/policies and the match result, he can infer the sensitive subscription/identity attributes of the innocent subscriber. Thus, the innocent subscriber's privacy can be leaked to the adversary.

As described above, the traditional encryption technique cannot defend against collusion attacks both in single-domain scenarios and cross-domain scenarios.

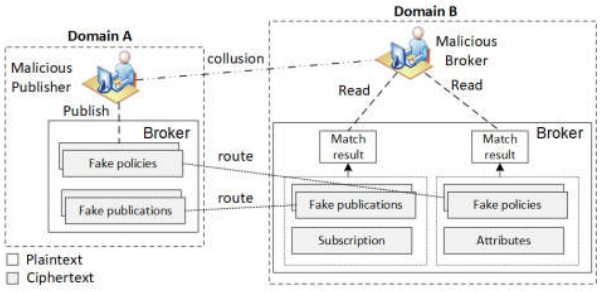


Figure 2. A collusion attack example using encryption matching protection methods in a cross-domain scenario.

2.3. Intel Software Guard Extensions (SGX)

Intel SGX aims to provide confidentiality and integrity guarantees for applications running on a potentially malicious host [9]. Specifically, SGX provides applications with a trusted execution mechanism in the form of secure enclaves. Enclave code and data are stored in a data structure called the Enclave Page Cache (EPC), which resides in a special memory region called the Processor Reserved Memory (PRM). The CPU protects the PRM from all non-enclave memory accesses, including privileged software such as the kernel, hypervisor accesses, and even DMA accesses from peripherals. Only application code executing inside an enclave is permitted to access the PRM memory that belongs to the enclave. Moreover, the enclave code can also access non-PRM memory.

In addition to isolation, another important feature of hardware enclaves is remote attestation. Remote attestation proves to a remote client that she is communicating with specific software loaded into a secure enclave hosted by the trusted hardware. When the client requests remote attestation, the enclave code generates a quote signed by the processor, which contains enclave attributes, including a measurement of the enclave's initial state, and then sends the quote to the challenger. The client can then verify the quote's authenticity and integrity using Intel's attestation verification service. Once the trust is settled, both the client and the enclave derive the session keys that would be used to transmit future messages between the client and the enclave during the session.

3. Design Considerations

In this section, we first describe the threat model in our design. Then, we illustrate the security objectives in our model. Finally, we present the related work showing that no existing solution fulfills these objectives.

3.1. Threat Model

Unlike the passive attack threat model in most pub/sub studies [4], we present an active attack threat model in this paper. In a passive attack model, the entities, including the brokers, are considered *honest-but-curious*. However, in some practical scenarios, malicious entities might actively attack the pub/sub system. In our threat model, we assume that an adversary may control several entities to launch collusion attacks. We assume that an adversary controlling brokers deployed on the cloud tries to gain as much sensitive information as possible. The adversary can control the broker's entire software stack, which includes privileged software such as kernel and hypervisor. As a result, the adversary can monitor and tamper with data in memory or on disk; observe all network communications, and monitor communication between untrusted and trusted software parts.

We made a conservative assumption about the enclave that the attacker cannot observe any information about the protected code and data in the enclave. We assume that the remote attestation procedure establishes a secure channel between the correct parties. We also assume that the code placed inside the enclave is correct and does not leak secrets intentionally. We assume that the Intel processor is trusted. All attacks that violate the enclave assumption above are out of scope for MagikCube. For example, research has shown that the Intel SGX's implementation suffers from various side-channel attacks such as cache-based attacks [10] or page fault side channel [11]. Although these are important

issues with SGX, MagikCube treats these attacks as out of scope since solutions have been proposed for mitigating the attacks [12]. And these solutions can be regarded as a complement to our contributions in this paper.

Besides, we assume that each broker has a processor with Intel SGX allowing running a TEE enclave.

3.2. Security Objectives

In the real world, publications and subscriptions may reveal sensitive information about relevant subjects. For example, in an e-health system, the publications can be medical records. The medical record is privacy-sensitive since it contains personal information about the patient such as his name, address, age as well as type of injury [13]. Besides, the subscription is also privacy-sensitive since it can reveal which patient is treated for which type of disease. Thus, a privacy-preserving pub/sub system should protect the confidentiality of both the publications and subscriptions.

Based on the previous survey [4], we define the following security objectives for a privacy-preserving pub/sub system under the considerations of our attack model:

1) When performing matching operations on publications and subscriptions, the broker should not know the specific content of the publications and subscriptions since the content can reveal sensitive information.
2) The pub/sub system can resist the collusion attacks in both single-domain scenarios and cross-domain scenarios. More specifically, if an adversary controls a publisher/subscriber and a broker, he cannot infer the subscriptions of an innocent subscriber.
3) The pub/sub system could force access control mechanisms over publications even if the broker was controlled by an attacker. Furthermore, even if the adversary controls the broker and the subscribers, he cannot break through the restrictions of access control over publications that are not authorized to him.
4) The pub/sub system should ensure loosely-coupled property, and the communication between publishers and subscribers should be decoupled in time and space. More specifically, the subscribers should not communicate with the publishers directly. The publishers do not need to know the location or identity of the interested subscribers, nor do they need to synchronize with subscribers. Similarly, the subscribers do not need to know the location or identity of the publishers.

3.3. Related Work

To effectively protect the privacy of subscribers, a pub/sub system should satisfy the security requirements mentioned above. A secure pub/sub system should resist collusion attacks in both single-domain and cross-domain scenarios. Unfortunately, there are few studies on the defense or mitigation methods against collusion attacks in the pub/sub system [4]. Besides, some of these approaches destroy the loosely-coupled property of the pub/sub model while others cannot resist collusion attacks in cross-domain scenarios. In the following, we describe those mechanisms and point out their drawbacks.

Rao *et al.* addressed the collusion attack issue in [7] and [14]. To protect the privacy of honest subscribers, they introduce a trusted third-party engine to cloak subscriptions before sending them to the broker. As a result, the subscribers will receive more publications than needed. Thus, the attacker cannot precisely identify the real interests of honest subscribers in a collusion attack scenario. Moreover, to prevent the untrusted broker from learning sensitive publications and subscriptions, they follow the idea of [15] to employ the encrypted filtering method.

However, the encrypted filtering method needs secret sharing among publishers and subscribers reducing the loosely-coupled property of the traditional pub/sub scheme. Besides, their solution does not provide access control mechanisms to restrict access to sensitive publications.

Cui *et al.* [16] present a privacy-preserving pub/sub system that resists collusion attacks using multiple types of brokers to match and route publications to the intended subscribers. They divide the match operations into different phases, and each phase is executed by a different type of broker. If a compromised subscriber colludes with one untrusted broker, they cannot infer the contents of subscriptions since each type of broker only processes partial information. However, although their scheme can resist collusion attacks in single-domain scenarios, it cannot resist collusion attacks in cross-domain scenarios. Specifically, if three types of brokers managed by three different domains collude with a malicious publisher or subscriber, they could learn the subscriptions of innocent subscribers.

Pires *et al.* [17] present a secure pub/sub routing engine that exploits the SGX technology to execute a routing engine in a secure enclave. In their scheme, the matching process between publications and subscriptions is performed in the SGX enclave. Thus, the untrusted broker cannot access the publications and subscriptions due to the protection of the enclave. This scheme can also resist single-domain collusion attacks since the attacker cannot access the results of the match operations performing in the enclave. However, their scheme requires the subscribers to first send the subscriptions to the publishers during the subscription process, violating the loosely-coupled property of pub/sub systems. PubSub-SGX [18] also leverages the feature of SGX enclave to guarantee confidentiality and privacy of publishers and subscribers. However, it does not consider cross-domain scenarios.

4. MagikCube Design

In this section, we introduce the overall architecture of MagikCube. MagikCube fulfills all the security objectives described in section 3.2. We start by presenting the leading

roles in the MagikCube framework, and then we describe in detail the design of the MagikCube.

4.1. Main Roles

MagikCube consists of four main roles: the publishers, the subscribers, the brokers, and the certification service. The publishers and subscribers are clients and the brokers are the server in pub/sub systems. Moreover, we introduce a trusted third party called the Certification Service to verify the identities of these entities.

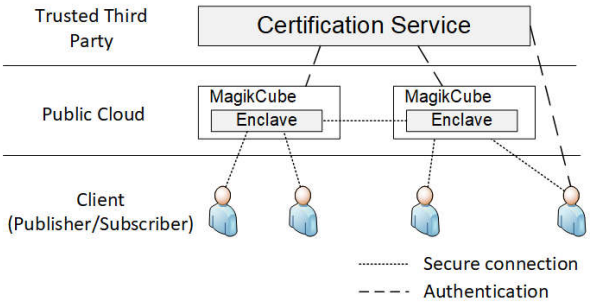


Figure 3. MagikCube main roles interaction

Publishers produce data in the form of publications which consists of payload and attributes. Subscribers express their interests by defining subscriptions composed of constraints to receive a subset of publications.

Broker is responsible for filtering and forwarding publications to interested subscribers. We deploy the MagikCube server components on the broker. More specifically, the MagikCube server components store subscriptions in an enclave, routing publications based on the constraints of subscriptions in an enclave. Moreover, the MagikCube server components achieve fine-grained control of publication's attributes by transforming publication based on the related access policies before forwarding the publication to the destination.

Certificate Service is a trusted third party, and there is only one Certificate Service in the MagikCube framework. Certificate Service validates clients' identities and attests MagikCube trusted server components periodically to provide clients with a guarantee that the MagikCube server components are indeed running on a valid SGX platform.

4.2. MagikCube Framework

MagikCube utilizes the isolation feature of the SGX enclave to ensure that important security-sensitive operations are performed in a secure region where the adversary cannot learn information. Furthermore, to provide clients with a guarantee that the MagikCube server components are indeed running on a valid SGX platform, the certification service to attest the server components periodically. More specifically, MagikCube leverages the SGX enclave to perform matching operations as well as store sensitive subscriptions. Thus, the untrusted broker cannot learn sensitive information about the publications and subscriptions. Besides, enforcement of the access control policy is also executed in the enclave. Thus, MagikCube can restrict access to sensitive publications. Moreover, the match results are produced in the enclave, and therefore the attacker cannot infer the subscriptions of innocent subscribers in both single-domain and cross-domain scenarios. Last but not least, MagikCube does not require the subscribers to communicate with publishers directly and therefore ensures the loosely-coupling property of the pub/sub paradigm.

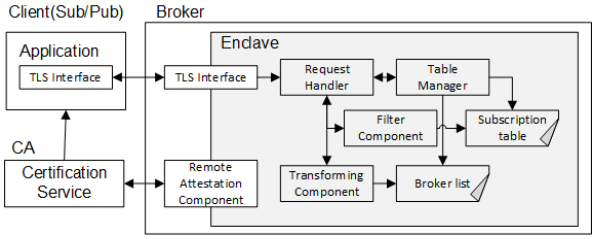


Figure 4. MagikCube architecture

Certification Service (CA). The CA is responsible for issuing a certificate for clients and brokers. For each client, the CA validates the client's identity and provides a client certificate to him. For each broker, the CA performs remote attestation of the enclave on the broker periodically. The period can be configured by the CA's administrator. If the broker passes the attestation, the CA issues a certificate with an expiration date to the enclave. The expiration date is lower than the attestation period. Before the two entities establish a secure communication channel, they will verify each other's certificates first.

The Broker's Trusted Component. The other trusted components within the enclave are the request handler component, the filter component, the table manager, and the transforming component. The request handler component parses each incoming request and sends it to the corresponding components. More specifically, if the request is a subscription request, the handler forwards it to the table manager. The table manager maintains a subscription table and adds, deletes, or modifies the item of the table according to the request. If the request is from a broker, the request is also forwarded to the table manager. The table manager maintains another broker list that contains the identity and the address of the connected broker. If the request is a publication request, the handler forwards it to both the filter component and the transforming component. The filter component matches each item of the subscription table against the incoming publication. On success, it transfers the publication to the transforming component. The transforming component transforms the incoming publication according to the corresponding access policy as well as the identity attributes of the subscriber. Then it encapsulates the transformed publication and the subscriber's address into a data packet. Next, it sends the packets to the request handler, and the request handler sends packets through the secure

TLS connection to the destination.

4.3. Setup Phase

The setup phase of MagikCube establishes mutual trust between the two related entities. This procedure is performed only once, and the trust established is the foundation for the end-to-end security of sensitive events.

Software that uses enclave needs to provide the following assurances to connected entities: 1) this software is indeed running inside a genuine SGX hardware rather than SGX simulator, which may be controlled by an adversary, and 2) the connection between them is secure.

The attestation procedure can achieve these goals. In MagikCube, the CA performs remote attestation of the enclave periodically, and the whole period can be configured. If the enclave passes the attestation, it will receive a certificate that contains an expiration date lower than the period. The certificate contains the enclave's public key, which is sent from the enclave to the CA during the attestation procedure and the enclave's identity. Once the enclave receives the certificate, it checks the certificate's validity using the CA's public key. On success, it persists the certificate in untrusted memory. Every time an entity connects to the enclave, it will request the enclave to send the certificate. Since the CA is trusted and it has performed remote attestation to check the validity of the enclave, the entity only needs to use the CA's public key to verify the certificate to ensure that it is indeed communicating with a trusted entity.

In MagikCube, only authenticated publishers and subscribers can publish events and subscribe to messages. To establish broker's and proxy's trust in clients, each client should provide its identity to the CA for validation. Then the client will receive a certificate issued by the CA. The certificate contains the client's identity information, such as the user ID, the name, and the mail address. The client can use the CA's public key to verify the certificate's validity. During the handshake, the client sends his client certificate to the enclave, and then the enclave verifies the certificate using the CA's public key. Once the verification is successful, the enclave knows that it is communicating with a legitimate client.

After entities' identities are verified, secure channels are established between two nodes to protect the confidentiality and integrity of the events. As a result, events are sent from authenticated nodes, and the receiving nodes never learn the content of the encrypted data.

Furthermore, brokers located in different domains need to establish a secure connection. If both brokers are untrusted, the secure channel will be established between enclaves on the two different brokers. As we mentioned above, trusted brokers are considered not to leak sensitive information. Therefore, trusted brokers do not need the assistance of SGX, and the normal SSL connection can be established between the two trusted brokers. The connection between a trusted broker and an untrusted broker is set up between normal software on the trusted broker and the enclave on the untrusted broker. As a result, when events are transmitted across different domains, an adversary cannot infer sensitive information about the events.

4.4. Subscription and Publication Process

The confidentiality and integrity of publications and subscriptions should be protected in the publication process and subscription process. In this section, we focus on describing the publication and subscription process in an untrusted environment. We describe the publication and subscription process in both single-domain and cross-domain scenarios in detail.

4.4.1. single-domain. Only authorized subscribers and publishers can publish or receive sensitive publications. Besides, the subscriptions and publications should be protected with confidentiality and integrity during the subscription and publication process.

Since a secure connection has been established between the entities during the setup phase, the subscriber can also send subscriptions to the enclave on the local broker safely. The untrusted broker receives encrypted subscriptions through the secure channel and performs decryption operations in the enclave. The enclave stores subscriptions in plaintext. Furthermore, one authorized subscriber is allowed to have multiple subscriptions, and he can request the dedicated broker to add, delete or modify his subscriptions.

Likewise, the publisher can directly send the encrypted publications to the enclave on the local broker through the secure communication channel. As some properties of the publications are inaccessible to entities outside the domain, the publisher will also attach an access policy to the specific sensitive publications. As the connection is secure, the adversary cannot eavesdrop or modify sensitive information on the communication channel. When receiving encrypted publications, the enclave first decrypts the ciphertext and then matches subscriptions against publications in plaintext. Then the enclave re-encrypts the matching events and sends them to the corresponding subscribers through the secure channel. Thus, the subscriber can receive publications of interest without disclosing the subscriptions to untrusted entities.

4.4.2. cross-domain. Subscribers in one domain can also subscribe to events of other domains, but their access needs to be restricted. The secure channel between two brokers located in different domains has been established in the setup phase. Then the broker maintains a list of connected broker's addresses and identities extracted from the certificate. After the enclave receives the publication and related access policies, it transfers them to the enclave of the connected broker. The connected enclave decrypts them and matches the publications with the list of the subscriptions in plaintext. On success, the enclave matches the policies with the identity attributes of the subscriber and transforms the publication according to policies. For instance, if the access policy is (id1, {nurse}, {name, injury}), the enclave will extract the name and injury attributes as well as the

corresponding value of the publication whose id is id1. Then the enclave will send the transformed publication to the subscriber if his identity attributes contain nurse. If the publisher uploads his publications without access policy, the enclave on the local broker won't send the publication to another domain. In this way, it can control other domains' access to the publications of the local domain.

5. Security Guarantees

In this section, we describe the security guarantees of MagikCube under the aforementioned threat model. We first discuss the confidentiality of publications and subscriptions under the MagikCube framework. Second, we analyze the effectiveness of MagikCube against collusion attacks. Then, we discuss MagikCube's TCB size. Finally, we analyze the effectiveness of MagikCube against attacks over a network channel.

Publications and subscriptions confidentiality. MagikCube prototype protects the confidentiality of the publications and subscriptions with the help of TEE. TEE is attested with the help of the hardware attestation method provided by TEE hardware vendors, which is the Intel Attestation Server in our model. In addition to TEE, MagikCube provides confidentiality and integrity guarantees applying various security primitives such as TLS to protect network connection. Furthermore, MagikCube provides fine-grained access control for sensitive publications. More specifically, MagikCube employs the CA to authorize subscribers' identities and use transformation technology to process publications before forwarding them to other authorized brokers. Then subscribers would only receive and read specific parts of publications that they are permitted to access.

Against collusion attacks. MagikCube can resist not only collusion attacks between malicious brokers and publishers/subscribers in a single domain environment but also collusion attacks between malicious publishers and cross-domain malicious brokers. More specifically, subscriptions are stored in the trusted SGX enclave, and the matching operation between publications and subscriptions is performed within the SGX enclave, as shown in Fig.5. In this case, if the brokers collude with malicious publishers or subscribers in the same domain, they cannot observe whether subscriptions of innocent subscribers match the injected publications or the same publications as the injected subscriptions since the malicious attackers cannot access the results of the match operations within the enclave.

Similarly, if malicious publishers collude with the brokers of a different domain, they cannot observe whether innocent subscribers meet the access policy and whether innocent subscriptions match the publications since access check and match operations are performed by the enclave, as shown in Fig.6. In this way, the colluding entities cannot infer sensitive information of innocent subscribers in a cross-domain environment.

TCB size. The MagikCube prototype provides the smallest software and hardware TCB for deploying a pub/sub

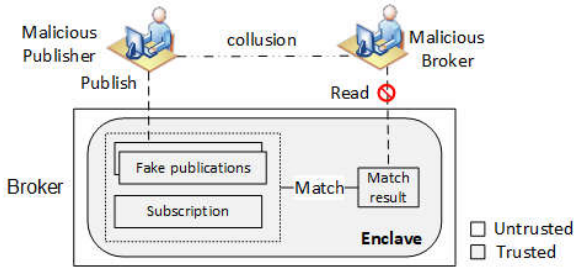


Figure 5. A collusion attack example using SGX protection methods in a single-domain scenario.

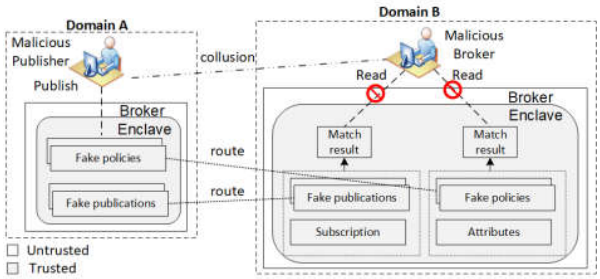


Figure 6. A collusion attack example using SGX protection methods in a cross-domain scenario.

architecture in the cloud. As for a broker node using SGX in the public cloud, the software components of TCB are necessary. The hardware of TCB is the CPU package of SGX nodes. The MagikCube framework excludes all high-privileged cloud system software such as hypervisor from the TCB.

Against attacks over a network channel. We consider that an adversary eavesdrops on the communication among different nodes. The adversary can learn insensitive information such as the source and the destination of the packets. However, the adversary cannot learn anything about the messages in the payload of the packet because of our network protection mechanisms. The SSL/TLS protects communication.

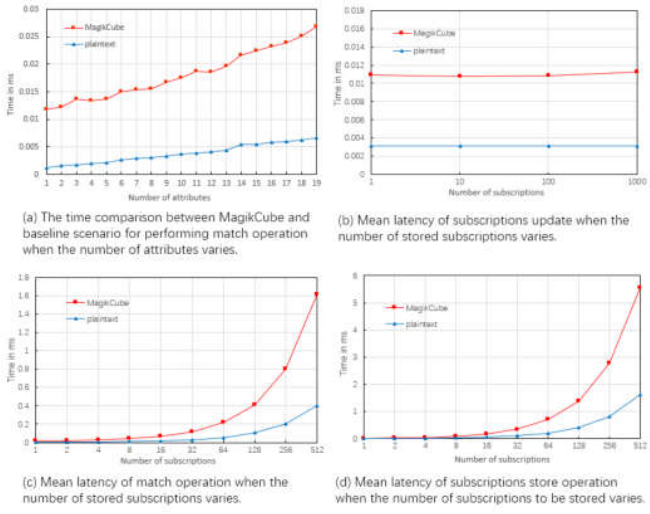
6. Performance Evaluation

We have implemented a prototype of our system in C/C++ and tested its performance. We are mainly interested in the delay at the broker level introduced by SGX. Since the communication between publishers and subscribers is asynchronous, the time it takes to encrypt and decrypt publications or subscriptions at their side is not critical. We consider the latency introduced at the broker side to be the most important measure for the performance of our scheme.

In our experiments, a broker node is a desktop machine with a 3.00GHz Intel i7-9700 CPU which has 16GB RAM and supports SGX. The OS of the broker node is Ubuntu 18.04.3 TLS 64bit with Linux kernel 4.15.0. We run Intel SGX SDK Linux 2.9 Release. All the results presented in the following section are averaged over 1000 runs.

As experiments in other pub/sub systems [14] [16], we consider the number of attributes in the publications and subscriptions varies from 1 to 20, which is due to practical considerations derived from an industry-standard benchmark used for measuring the performance of pub/sub system [5]. We compare the times needed for matching publications against subscriptions when the number of attributes varies from 1 to 20. In addition, we also focus on the influence of workloads and compare our solution with the original plaintext solution.

We conduct experiments to compare the performance loss between MagikCube and baseline model from four aspects, 1) time comparison when the number of attributes varies, 2) the mean latency of subscriptions update, 3) the mean latency of match operation, and 4) mean latency of subscriptions store operation. As depicted in Figure 7, MagikCube, like most enclave-based security solutions, causes some performance loss and latency to baseline pub/sub while it's introducing security features into the pub/sub paradigm. However, compared with the latest research solutions [16], the MagikCube solution is more efficient.



(a) The time comparison between MagikCube and baseline scenario for performing match operation when the number of attributes varies.

(b) Mean latency of subscriptions update when the number of stored subscriptions varies.

(c) Mean latency of match operation when the number of stored subscriptions varies.

(d) Mean latency of subscriptions store operation when the number of subscriptions to be stored varies.

Figure 7. Performance results

7. Conclusion

Although existing solutions enable encrypted matching to provide confidentiality in pub/sub system, they cannot protect the privacy of innocent subscribers if malicious publishers (or subscribers) collude with untrusted brokers. To address this issue, we propose a solution called MagikCube that uses SGX to perform matching operations in a secure enclave where attackers cannot learn the matching result and thus they are unable to infer the sensitive data of innocent subscribers. MagikCube can resist collusion attacks in both single-domain and cross-domain scenarios.

References

[1] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "The many faces of publish/subscribe," *ACM computing surveys (CSUR)*, vol. 35, no. 2, pp. 114–131, 2003.

[2] C. Esposito, M. Ciampi, and G. De Pietro, "An event-based notification approach for the delivery of patient medical information," *Information Systems*, vol. 39, pp. 22 – 44, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0306437913000987

[3] J. Bacon, D. Eyers, J. Singh, B. Shand, M. Migliavacca, and P. Pietzuch, "Security in multi-domain event-based systemssicherheit in ereignis-basierten mehrdomänensystemen," *IT-Information Technology*, vol. 51, no. 5, pp. 277–284, 2009.

[4] E. Onica, P. Felber, H. Mercier, and E. Rivière, "Confidentiality-preserving publish/subscribe: A survey," *ACM computing surveys (CSUR)*, vol. 49, no. 2, pp. 1–43, 2016.

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

[6] J. Bacon, D. M. Eyers, J. Singh, and P. R. Pietzuch, "Access control in publish/subscribe systems," in *Proceedings of the second international conference on Distributed event-based systems*, 2008, pp. 23–34.

[7] W. Rao, L. Chen, and S. Tarkoma, "Toward efficient filter privacy-aware content-based pub/sub systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 11, pp. 2644–2657, 2012.

[8] L. I. Pesonen, D. M. Eyers, and J. Bacon, "Encryption-enforced access control in dynamic multi-domain publish/subscribe networks," in *Proceedings of the 2007 inaugural international conference on Distributed event-based systems*, 2007, pp. 104–115.

[9] V. Costan and S. Devadas, "Intel sgx explained." *IACR Cryptol. ePrint Arch.*, vol. 2016, no. 86, pp. 1–118, 2016.

[10] F. Brasser, U. Müller, A. Dmitrienko, K. Kostiainen, S. Capkun, and A.-R. Sadeghi, "Software grand exposure:{SGX} cache attacks are practical," in *11th {USENIX} Workshop on Offensive Technologies ({WOOT} 17)*, 2017.

[11] Y. Xu, W. Cui, and M. Peinado, "Controlled-channel attacks: Deterministic side channels for untrusted operating systems," in *2015 IEEE Symposium on Security and Privacy*. IEEE, 2015, pp. 640–656.

[12] S. Crane, A. Homescu, S. Brunthaler, P. Larsen, and M. Franz, "Thwarting cache side-channel attacks through dynamic software diversity." in *NDSS*, 2015, pp. 8–11.

[13] M. Ion, G. Russello, and B. Crispo, "An implementation of event and filter confidentiality in pub/sub systems and its application to e-health," in *Proceedings of the 17th ACM conference on Computer and communications security*, 2010, pp. 696–698.

[14] W. Rao, L. Chen, M. Yuan, S. Tarkoma, and H. Mei, "Subscription privacy protection in topic-based pub/sub," in *International Conference on Database Systems for Advanced Applications*. Springer, 2013, pp. 361–376.

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

[16] S. Cui, S. Belguith, P. De Alwis, M. R. Asghar, and G. Russello, "Collusion defender: preserving subscribers' privacy in publish and subscribe systems," *IEEE Transactions on Dependable and Secure Computing*, 2019.

[17] R. Pires, M. Pasin, P. Felber, and C. Fetzer, "Secure content-based routing using intel software guard extensions," in *Proceedings of the 17th International Middleware Conference*, 2016, pp. 1–10.

[18] S. Arnautov, A. Brito, P. Felber, C. Fetzer, F. Gregor, R. Krahn, W. Ozga, A. Martin, V. Schiavoni, F. Silva *et al.*, "Pubsub-sgx: Exploiting trusted execution environments for privacy-preserving publish/subscribe systems," in *2018 IEEE 37th Symposium on Reliable Distributed Systems (SRDS)*. IEEE, 2018, pp. 123–132.