Secure Distributed Publish/Subscribe (P/S) pattern for IoT

EDUARDO B. FERNANDEZ, Florida Atlantic University
NOBUKAZU YOSHIOKA, National Institute of Informatics
HIRONORI WASHIZAKI, Waseda University

We present here the Secure Distributed Publish/Subscribe pattern, one of the most used patterns in IoT software design. Its intent is: In an IoT system, decouple the publishers of events from those interested in the events (subscribers). Subscription and publication are performed securely. This pattern is derived from an earlier abstract version where the effects of the new environment have been reflected in the context, forces, and solution. In other words, the context, forces, and solution are new, although the pattern preserves the core functions of its abstract counterpart.

Categories and Subject Descriptors: D.2.11 **[Software Engineering]** Software Architectures–Patterns;

General Terms: Design

Additional Key Words and Phrases: Security patterns, secure software, software architecture, IoT, network segmentation

ACM Reference Format:

E.B.Fernandez, N. Yoshioka, H. Washizaki, "Secure Distributed Publish/Subscribe (P/S) for IoT, 2020. Procs. Asian PLoP'20, March 4-6, Taipei, Taiwan. 9 pages.

1. INTRODUCTION

The P/S pattern is one of the most used patterns in the design of software systems. It first appeared in [Bus96] but without much detail. We expanded it using UML models and a more complete description, as well as specializing it for distributed systems and adding security defenses [Fer 11, Fer13]. Later, Uzunov wrote a whole article about it indicating its many possible uses and varieties [Uzu16]. Our description used the concept of Abstract Security Pattern (ASP) [Fer14], where only the core aspects are included with no implementation aspects. Starting from that description we specialize it here for use in IoT systems. A concrete pattern describes a security solution within a technological context, e.g., distributed systems, XML web services, etc., which means that the pattern for IoT P/S is a concrete security pattern. No patterns for IoT P/S have been published but there exist a few papers discussing its implementation [Gra18, Rou02].

We present the Secure P/S for IoT in Section 2; its intent is:
"In an IoT system, decouple the publishers of events from those interested in the events (subscribers). Subscription and publication are performed securely." With respect to our earlier (abstract) Secure P/S pattern [Fer11], we have modified the context to fit distributed systems where IoT devices are normally used, we have added forces to reflect new requirements such as heterogeneity and identity, we have added a client which can assume the role of publisher or subscriber with appropriate access control, and defined the structure of the secure channel. We also added mutual authentication between publishers and subscribers.

Figure 1 shows its relationship to other middleware patterns. All these patterns can be found in [Fer13]. Our audience includes software architects, software developers, and system administrators. We describe our patterns using a slightly modified POSA template [Bus96].

Authors' addresses: Eduardo B. Fernandez (corresponding author), Dept. of Computer and Electrical Eng. and Computer Science, Florida Atlantic University, 777 Glades Rd., Boca Raton, FL33431, USA; email: fernande@fau.edu;.Nobukazu Yoshioka, GRACE Center, National Institute of Informatics, Tokyo, Japan,; email: nobukazu@nii.ac.jp. Hironori Washizaki, Waseda University, Tokyo, Japan; email: washizaki@waseda.jp.

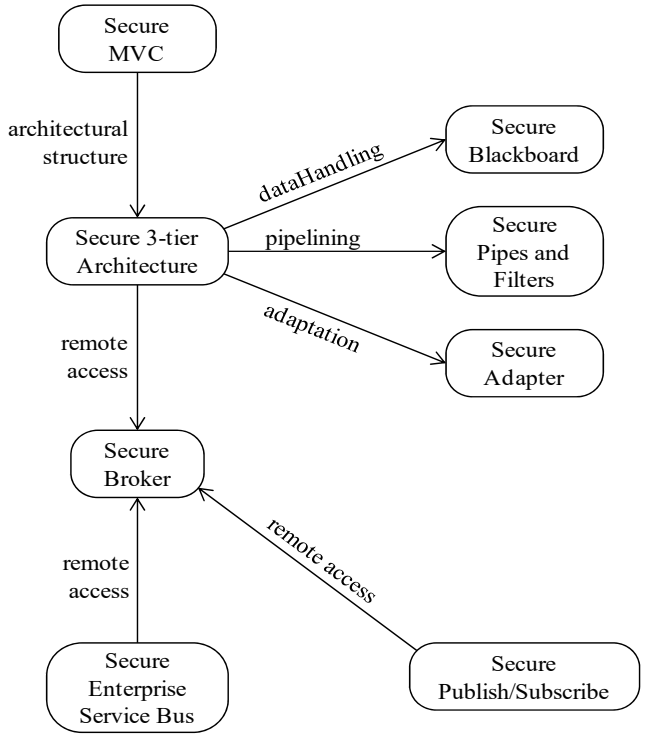

Figure 1. Pattern diagram relating middleware patterns (from [Fer12]).

2. SECURE P/S FOR IOT

2.1 Intent.

In an IoT system, decouple the publishers of events from those interested in the events (subscribers). Subscription and publication are performed securely.

2.2 Context

Typically, IoT devices send data to an edge/fog system and or to a cloud. Distributed systems applications using IoT devices can be of two types [Gra18]: the set of devices includes only a few devices that periodically send their data up to the cloud or fog. This is typical in consumer-oriented applications such as smart thermostats or sprinkler systems. The other type of applications has several devices or sensors sending concurrently large amounts of data. This is the case of industrial applications (IIoT). In general, there can be a large variety of different types of devices collaborating in complex applications, where each device has often minimal security controls.

2.3 Problem

Subscribers register and receive messages of their interest sent by a publisher. How do we organize publishers and subscribers such that their interactions are secure?

The solution is constrained by the following forces:

- Loose coupling: Publishers should be able to work without any knowledge of their subscribers and vice versa.
- Location transparency: Neither subscribers nor publishers need to know each other's locations.

- Security degree. Clients, publishers and subscribers may contain data with different levels of sensitivity, which must be protected from attacks.

- Attack surface. The attack surface of our computational entities should be as small as possible.

- Heterogeneity. Typical networks include devices from many origins, which may be necessary for our applications but may bring threats.

- Identity. We need to keep track of the devices which are under our control or participating in our network. Knowing who is very important for security.

- Compliance. Some of the information in our system may have to follow regulations or industrial standards. These regulations may restrict access to only some types of users; for example, medical events may be accessed only by health care personnel.

- Overhead. Event distribution should not result in a large overhead.

- Power consumption. IoT devices are usually power constrained. Our solution must be economic on the use of power.

Threats: We can relate threats to use cases as goals of the attacker [Fer13]:

Subscription:
S1: An impostor impersonates a subscriber and subscribes to receive information that will be billed to somebody else or which will give her access to sensitive information.
S2: The publisher is an impostor and collects information (and maybe money) from potential subscribers. For example, subscribers may need to provide personal information and may need to pay for services from the publisher. For example, I subscribed my PC to receive malware defenses.
S3: The subscription messages are intercepted and read or modified by an attacker. The attacker may obtain in this way credit or other personal information from the subscriber.

Unsubscription:
U1 An impostor acting on the publisher removes a subscriber without its consent.

Publishing:
P1: An impostor (impersonating a subscriber) receives information illegally from the publisher.
P2: A publisher publishes erroneous information. This action can inject data or commands to a device thus disturbing its operation; for example, publish erroneous sensor data for a robot [Die17].
P3: An attacker reads or modifies information intercepted in the publishing channel.
P4. An attacker floods subscribers with fake messages thus stopping the subscribers from doing any useful work; this is a Denial of Service attack.

2.4 Solution

In addition of the standard P/S functions it is possible to use secure channels for protected communications, access control for restricting the actions of publishers and subscribers, security logging, and digital signatures. Depending on their required level of security, not all of these security controls may be used in specific applications,

2.4.1 *Structure*

Figure 2 shows the class diagram of this pattern. A Publisher receives and stores Data that may represent events or conditions from other units that should be distributed to different units in a distributed system. Interested units can become Subscribers and start Subscriptions to specific events. Events are sent to Subscribers through a Secure Channel that can be encrypted by the publisher using an Encryptor. Specific Events may be digitally signed using a

Signer. On its part, the subscribers can decrypt the events using a Decryptor and can verify signatures using a Signature Verifier. Clients may take up roles as publisher or subscriber and their functions can be controlled using Role-Based Access Control with predefined Rights. Before they can use these roles, clients must be authenticated using an Authenticator. In addition, the Authenticator can mutually authenticate subscribers to publishers. A Security Logger/Auditor can record the sent events (not shown in Figure 2). The subscription channel can also be a secure channel (not shown in Fig. 2).

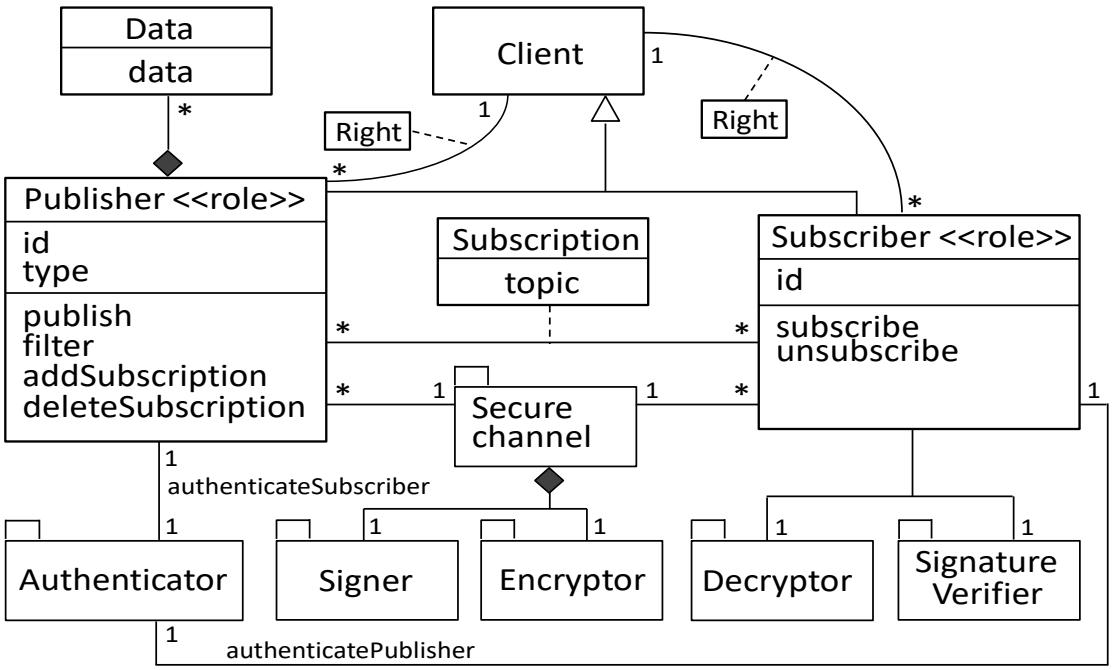

Figure 2. Class diagram of the IoT P/S pattern

2.4.2 *Dynamics*

Figure 3 shows a sequence diagram for the use case "Publish an event". It treats the secure channel as a whole system able to do encryption and signing. Its description is:

1. When an event is received the publisher first filters it according to the restrictions of specific subscriptions.
2. The event is then published and sent to the secure channel specifying the signature and encryption algorithms to be used by the secure channel (if needed).
3. When the subscriber receives the signed and encrypted event it first decrypts it to recover the message and its signature.
4. It then verifies its signature to prove that it comes from a legitimate origin.
5. The received event is sent to its corresponding client unit.

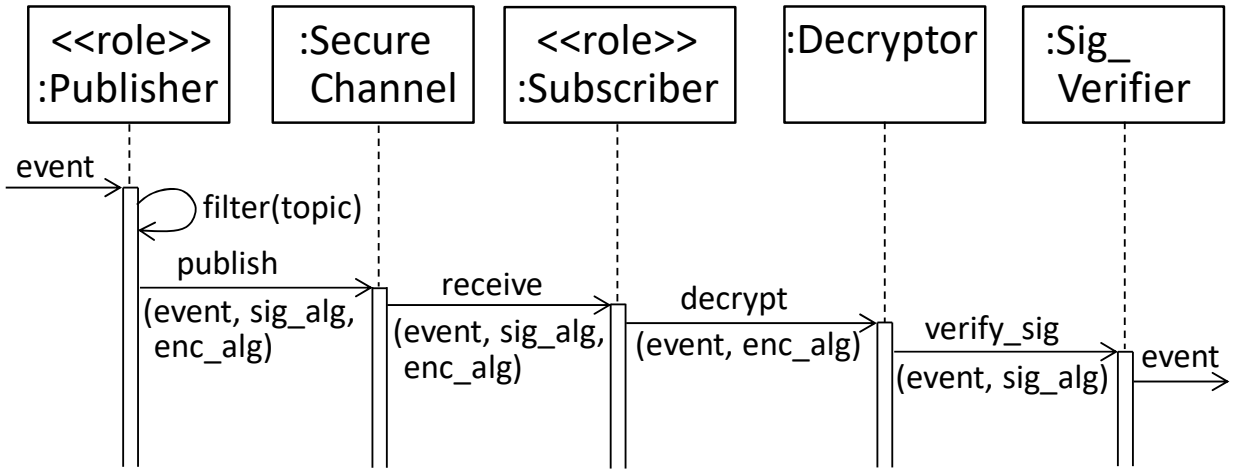

Figure 3 Sequence diagram for the use case "Publish an event or data".

Other use cases are: "Subscribe"and "Unsubscribe", which are very simple and not shown.

2.5 Implementation

The idea of security patterns is to help software developers who don't know much about security to build secure systems so we do not expect the typical developer to build the security services; in fact, all of the standard security mechanisms are available from one or more vendors. To make things even simpler, some of the standard protocols such as MQTT and DDS include security mechanisms as part of their code. If for any reason a specialized security mechanism is not available the developers may need to resort to security specialists. Refs. [Gra18] and [INT] provide details of the implementation of the functional aspects of a P/S, including content filtering and reliability. A 7-layer reference architecture showing implementation details is given in [Uzu16]. [Nei14] shows an extension of the MQTT protocol to enforce authorization rules in the message exchanges between publishers and subscribers. [Lag10], although is not about IoT shows architectural and implementation details that should be useful to builders of P/S systems. [Esp15] is a comprehensive survey of security in P/S and include many implementation details.

Several implementations for the secure channel have been used:

- Advanced Message Queuing Protocol (AMQP), which defines an efficient, binary, peer-to-peer protocol for transporting messages between two network processes (usually a client and a broker)

- Constrained Application Protocol (CoAP) is a software protocol that was designed to support the connectivity of simple low power electronic devices (e.g. wireless sensors) with Internet based systems

Section 2.6 lists other protocols that include the secure P/S pattern as part of their specifications. Each one of them has different ways to implement and manage their security services.

The Authenticator on the publisher is needed only for registration or when the subscribers pull their events from the publisher (see Variants). Authenticators on the subscribers are needed when the events are sensitive and must be received correctly; for example, [Die17] shows an example where a robot receives events from sensors that detect obstacles or humans in its path which if tampered with by an attacker could produce catastrophic situations.

2.6 Known uses

- A framework for data-centric P/S for IoT applications is shown in [Gra18]. This framework considers reliability, filtering, and real-time deadlines. The framework also has a 3-level hierarchy to define degrees of security.

- Distributed Publish & Subscribe for the IoT (DPS) is a new protocol that implements a P/S pattern [INT]. It uses a dynamic multiply-connected mesh where each node is in effect a message router.

- Message Queue Telemetry Transport (MQTT), an open source protocol which acts like a broker between publishers and subscribers [Mal19]. This protocol provides Authentication, for clients (publishers or subscribers); Authorization (to restrict a client for publishing and/or subscribing to specific topics; TLS to provide a secure channel; and Payload encryption. An extended version enforcing authorization rules is shown in [Nei14].

- The OMG Data-Distribution Service for Real-Time Systems (DDS) is the first open international middleware standard directly addressing publish-subscribe communications for real-time and embedded systems [OMG]. DDS introduces a virtual Global Data Space where applications can share information by reading and writing data-objects addressed by means of an application-defined name (Topic) and a key. DDS features fine and extensive control of QoS parameters, including reliability, bandwidth, delivery deadlines, and resource limits. It is heavily used in IoT applications, including smart cars, energy, flight control [Gra18]. The OMG DDS Security Specification defines a comprehensive Security Model and Service Plugin Interface (SPI) architecture for compliant DDS implementations. DDS provides standardized authentication, encryption, access control and logging capabilities to enable secure end-to-end data connectivity in an IoT system.

- A highly secure P/S protocol was implemented in a project at the Brno University of Technology [Mal19]. It provides three security levels to accommodate functions that need different degrees of security, as well as mutual authentication between publishers and subscribers.

- A secure P/S protocol for robotic operating systems is described in [Dei17].

2.7 Consequences

The pattern presents the following advantages:

- Loose coupling: Publishers can work without knowledge of their subscriber details and vice versa. This fact protects the subscribers in case the publisher is compromised.

- Location transparency: Neither subscribers nor publishers need to know each other's locations, a lookup service can find their locations. This aspect protects both publishers and subscribers.

- Security degree. Data with different levels of sensitivity can be protected by including some or all of the defenses of Figure 2.

- Attack surface. The attack surface of our computational entities can be reduced by applying the need-to-know principle: publishers communicate only with legitimate subscribers and clients can be restricted to use only some operations in a publisher or subscriber.

- Heterogeneity. Devices from different origins or levels of trust can be separated using IoT Security Segmentation [Fer19].

- Identity. Identity patterns [Fer13] can be used to define identity federations and unique identities.

- Compliance. Compliance with regulations can be obtained including security patterns and compliance patterns.

- Overhead – There is some overhead in the event structure, i.e., a tight coupling of subscribers to their publishers would have better performance at the cost of flexibility and security. A distributed mesh [INT], instead of a broker, can reduce event distribution overhead. This selection is an implementation aspect, not dependent on the software architecture of the patterns.

- Power consumption.—Signature and encryption algoritms should be lightweight due to the limited computational power of most IoT devices. The protocol used may also have an effect; for example MQTT requires devices to be always awake to keep their TCP connection. Again, this is an implementation problem.

Threats: if events are sensitive we can encrypt the event channel. We can also use digital signatures for authenticity. We can require clients using this pattern to authenticate themselves and then control their interactions using Role-Based Authorization. Mutual authentication avoids impostors. We can also use a secure channel for subscription. A security logger/auditor can detect insider attacks.
Possible liabilities include:

- Excessive interoperability – Because of its decoupling effect, this pattern allows the interaction of any type of publishers and subscribers, and hence is liable to attackers gaining easier access.

- A distributed system may also suffer denial of service attacks which cannot be controlled at this pattern's level.

- When a secure broker is used this broker becomes a single point of failure. Intel's DPS solves this problem by using a more distributed structure, a kind of mesh [INT].

2.8 Related patterns

- Secure P/S [Fer11, Fer13]. Decouples the publishers of events from those interested in the events (subscribers). Subscription and publishing are performed securely. This pattern is an abstract pattern, it does not include any effect of the technology of the layer where the pattern is used.

- Broker. A Broker can be used as the distribution channel. It typically includes a look up service and can distribute events to subscribers in a transparent way. A broker may include security services and thus become a Secure Broker [Fer13].

- The Secure Channel pattern [Bra98], supports the encryption/decryption of data.

- In P/S middleware communication may be based on group broadcast, which would require secure group communication protocols. There are currently no security patterns for this purpose

- Enterprise Service Bus (ESB). An ESB includes all the services needed for the P/S functions and uses the P/S functions for its own functions. An ESB may include its own security services [Fer13].

- Authenticator [Fer13]. After identification the authenticator grants access to the system if the requester is a registered subject.

- Authorizer. Control of who can access a network entity and in what way [Fer13]. An important variety is Role-Based Access Control.

- Digital Signature with Hashing. A principal can prove that a message was originated from it. It also provides message integrity by indicating whether a message was altered during transmission.

- IoT Segmentation [Fer19]. Security segmentation for IoT partitions a network of IoT devices and supporting entities into subnetworks in order to isolate groups of IoT devices and entities with different security requirements.

- Identity [Fer13]. The Identity Provider centralizes the administration of a security domain's users creating and managing identities for their credentials. The Circle of Trust represents a federation of service providers that

share trust relationships. The Identity Federation pattern allows the federation of multiple identities across multiple organizations under a common identity.

- Security Logger/Auditor[Fer13]. How can we keep track of user's actions in order to determine who did what and when? Log all security-sensitive actions performed by users and provide controlled access to records for Audit purposes.

- [Has17] proposes a blockchain-based system where clients of the blockchain can use subscribers to receive updates from a blockchain used for access control in IoT or autonomous cars. This idea is not described as a pattern but could be the basis of a pattern.

- Several security and misuse IoT patterns are discussed in [Fer19], while several design IoT patterns are discussed in [Was19].

Although clearly different, this pattern is sometimes confused with the Observer [Gam94].

2.9 Variants

There are Push and Pull varieties of this pattern. The example of Figure 3 shows a Push P/S. In a Pull pattern, the publisher notifies the subscribers which then ask for the new event.

ACKNOWLEDGMENTS

This pattern was started during the visit of the first author in March of 2019 by invitation of the National Institute of Informatics of Japan. Our shepherd, Pin-Ying Tu, provided insightful comments that have significantly improved this paper. .

REFERENCES

[Bus96] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal, Pattern-Oriented Software Architecture: A System of Patterns, Volume 1, John Wiley & Sons, West Sussex, England, 1996.

[Die17] B. Dieber, B. Breiling, S. Taurer, S. Kacianka, S. Rass, P. Schartner, "Security for the robot operating system", Robotics and Autonomous Systs. , 98, 2017, 192-203.

[Esp15] C. Esposito, M. Ciampi, "On security in Publish/Subscribe services: A survey". IEEE Comm. Surveys and Tutorials, vol. 17 No 2, Second-Quarter 2015, 966-997.

[Fer11] E.B.Fernandez, Nobukazu Yoshioka, and Hironori Washizaki, "Two patterns for distributed systems: Enterprise Service Bus (ESB) and Distributed Publish/Subscribe", 18th Conference on Pattern Languages of Programs (PLoP 2011)

[Fer12] E.B. Fernandez and A.V. Uzunov, "Secure middleware patterns", 4th International Symposium on Cyberspace Safety and Security (CSS 2012), Melbourne, Australia, Dec. 12-13, 2012.

[Fer13] E.B.Fernandez, "Security patterns in practice: Building secure architectures using software patterns", Wiley Series on Software Design Patterns, 2013.

[Fer19] E.B. Fernandez, N. Yoshioka, H. Washizaki, "Abstract and IoT security patterns for network segmentation", Proceedings of the 8th Asian Conference on Pattern Languages of Programs., March 2019.

[Gam94] E. Gamma, R. Helm,R. Johnson, and J. Vlissides, Design patterns –Elements of reusable object-oriented software, Addison-Wesley 1994.

[Gra18] Sara Granados Cabeza, "Use a data-centric publish/subscribe framework for IoT applications", Electronic Design, March 30, 2018.

[Has17] S.H.Hashemi, F.Faghri, R.H.Campbell, "Decentralized user-centric access control using PubSub over blockchain", arXiv: 1710.00110

[Int] Intel Corp., Distributed Publish & Subscribe for IoT. https://intel.github.io/dps-for-iot

[Lag10] D. Lagutin, K. Visala, A. Zahemszky, T. Burbridge, G.F. Marias, "Roles and security in a Publish/Subscribe network architecture", IEEE Symp. on Comps. and Comms., June 2010, DOI: 10.1109/ISCC.2010.5546746

[Mal19] L. Malina et al., "A secure publish/subscribe protocol for Internet of Things", Procs. of ARES '19, Aug. 2019, Canterbury, UK.

[Nei14] R. Neisse, G. Steri, G. Baldini, "Enforcement of security policy rules for the Internet of Things", 2014 Third Int. Workshop on IoT Comms. And Technologies, 165-172.

[OMG] Data Distribution Service (DDS), https://www.omg.org/omg-dds-portal/

[Rou02] P. Rousselle, "Implementing the JMS Publish/Subscribe API", Dr. Dobbs Journal, April 2002, 28-32.

[Uzu16] A.V. Uzunov, "A survey of security solutions for distributed publish/subscribe systems". Computers & Security 61: 94-129 (2016)

[Was19] H. Washizaki, N. Yoshioka, A. Hazeyama, T. Kato, H. Kaiya, S. Ogata, T. Okubo and E.B. Fernandez, "Landscape of IoT Patterns," 2019 IEEE/ACM 1st International Workshop on Software Engineering Research & Practices for the Internet of Things (SERP4IoT 2019), Montréal, QC, Canada, May 2019.