# A Software Product Line Approach for Feature Modeling and Design of Secure Connectors

Michael Shin[1], Hassan Gomaa[2] and Don Pathirage[1]
*[1]Department of Computer Science, Texas Tech University, Lubbock, TX, U.S.A.*
*[2]Department of Computer Science, George Mason University, Fairfax, VA, U.S.A.*

Keywords: Software Product Line, Feature Model, Secure Connector, Secure Software Architecture, Component-based Software Architecture, Secure Software Design, Message Communication Patterns, Security Patterns, Model-based Design, UML.

Abstract: This paper describes a software product line approach to modeling the variability of secure software connectors by means of a feature model, which consists of security pattern and communication pattern features used in the design of secure component-based software architectures for concurrent and distributed software applications. Applying separation of concerns, these features are designed as security and communication pattern components. Each secure connector is designed as a composite component that encapsulates both security pattern and communication pattern components. Integration of these components within a secure connector is enabled by a security coordinator. This paper describes the feature model, design of secure connectors, how applications are built using secure connectors, and the validation of the approach.

## 1 INTRODUCTION

Secure component-based software architectures for concurrent and distributed software applications are composed of components and connectors in which connectors encapsulate the details of communication between components. Although connectors are typically used in software architecture (Medvidovic and Taylor, 2010) to encapsulate communication mechanisms between components, security concerns can also be encapsulated in software connectors, which are referred to as secure connectors (Shin et al., 2012, 2016a, 2016b, 2017) separately from application components that contain application logic. However, to facilitate reusing secure connectors in different applications, it is necessary to design secure connectors that are both modular and reusable.

Each secure connector is designed as a composite component using component concepts by reusing security pattern components and communication pattern components, which are designed separately from each other. Each security pattern component encapsulates a security pattern, such as symmetric encryption or digital signature. Each communication pattern component encapsulates the communication pattern between application components, such as synchronous or asynchronous message communication. A secure connector is then constructed by composing security pattern components and communication pattern components. Integration of security patterns and communication patterns within a secure connector is provided by a security coordinator. Once a secure connector is constructed, it can then be reused in different applications.

This paper describes modeling secure connectors by means of a software product line (SPL) approach. Our previous papers (Shin et al., 2007, 2012, 2016a, 2016b, 2017) have focused on designing single reusable secure connectors in an ad hoc way. This paper investigates how applying SPL concepts can lead to a more systematic approach that addresses the inherent variability in the design of secure connectors that separately encapsulate both the security and communication concerns.

This paper is organized as follows. Section 2 describes existing approaches to modeling and designing security concerns in software applications. Section 3 describes a software product line approach for secure connectors, followed by the feature model

Shin, M., Gomaa, H. and Pathirage, D.
A Software Product Line Approach for Feature Modeling and Design of Secure Connectors.
DOI: 10.5220/0006904805060517
In *Proceedings of the 13th International Conference on Software Technologies (ICSOFT 2018)*, pages 506-517
ISBN: 978-989-758-320-9

for secure connectors in section 4. Section 5 describes communication and security features components. Section 6 describes security coordinator components. Section 7 describes a secure connector derived from a software product line for secure connectors. Section 8 describes the validation of reusable secure connectors. Section 9 describes conclusion of this paper with future research.

## 2 RELATED WORK

Related work focuses on approaches to designing software architectures for secure applications, patterns for distributed communication and component-based software product lines. The authors in (Lodderstedt et al., 2002) proposed SecureUML, which is a new modeling language based on UML for the model-driven development of secure systems. Work has also been proposed to provide an extension of UML called UMLsec (Jürjens, 2002) that helps with the expression of security-relevant information within design diagrams. In model-driven security (Basin et al., 2011), a system is modeled with its security requirements and security infrastructures are generated using the models.

Using connectors as the central construct, a distributed CBSA in (Gomaa et al., 2001, 2011) is composed of a set of components and a set of connectors that can be used to connect the components. In (Ren et al., 2005), a connector centric approach is used to model, capture, and enforce security. The security characteristics of a CBSA are described and enforced using software connectors. Methods in (Al-Azzani and Bahsoon, 2012) propose SecArch to evaluate architectures with significant security concerns.

Security patterns in (Fernandez-Buglioni, 2013; Schumacher et al., 2006) address the broad range of security issues that should be taken into account in the stages of software development lifecycle. The authors describe the problem, context, solution, and implementation of security patterns in a structured way with a template so that the presentations are consistent. The security patterns can help developers to construct secure systems, even though the developers may not have security expertise.

A software product line (SPL) (Gomaa, 2005) is a family of software systems that have some common functionality and some variable functionality. The functionality of a SPL can be modeled by means of features, which are designed with kernel, optional, and variant components. Each component has ports with provided and required interfaces. The authors in (Gomaa and Shin, 2004, 2007, 2008, 2010) addressed multiple-view modeling and meta-modeling of software product lines. A co-author in (Abu--Matar and Gomaa, 2011; Fant et al., 2015; Tzeremes and Gomaa, 2018) investigated the design of the software product line architecture for service-oriented systems, space flight systems, and smart spaces.

In recent work by the authors (Shin et al., 2012) described secure asynchronous and synchronous connectors for modeling the software architectures for distributed applications and the design of reusable secure connectors that are structured into reusable security components and communication components. The authors in (Shin et al., 2016a, 2016b, 2017) address the design of secure connectors in terms of maintainability and evolution, which are used in the design of secure software architectures. A co-author in (Gomaa et al., 2010; Albassam et al., 2016) has also investigated designing dynamically adaptable and recoverable connectors.

## 3 SPL FOR SECURE CONNECTORS

A security service (Farahmandian and Hoang, 2017; Taha et al., 2017) is software functionality for realizing a security goal, such as authentication, authorization, confidentiality, integrity, availability or non-repudiation, which can be implemented by means of different security techniques. A security service can be realized by means of different security patterns (Fernandez-Buglioni, 2013; Schumacher et al., 2006), each of which addresses a specific security technique that realizes a security service. For instance, a confidentiality security service can be realized by means of a symmetric encryption security pattern (Fernandez-Buglioni, 2013) or an asymmetric encryption security pattern (Fernandez-Buglioni, 2013).

Although there are other types of communications between distributed components, typical message communication patterns between the components are synchronous message communication with reply, synchronous message communication without reply, asynchronous message communication, and bidirectional asynchronous message communication (Gomaa, 2011). Communication patterns (Schneider, 2005) are frequently used protocols by which distributed components communicate with each other. Each communication pattern is designed with a sender communication pattern component (CPC) and a receiver communication pattern component (CPC), which are encapsulated in a secure sender connector and a secure receiver connector respectively.

A secure connector is designed by separately considering the message communication pattern and the security patterns required by application components. A secure connector is a distributed connector, which consists of a secure sender connector and a secure receiver connector that communicate with each other. A secure sender or receiver connector consists of a security coordinator, one or more security objects, and a communication object (Shin et al., 2016b). A secure connector can be reused for different applications once it is constructed.

In this paper, the reusability of secure connectors is enabled by applying SPL concepts to model the variability of secure connectors and to design reusable secure connectors in a systematic way. The SPL approach provides a capability of modeling the variability of secure connectors in terms of security patterns and communication patterns. Applying SPL concepts enables us to create variants of secure connectors based on the variability of security and communication patterns.

In the SPL for secure connectors, security and communication patterns are modeled as features in a feature model. The security pattern and communication patterns features for secure connectors are modeled in the feature model by means of the optional attribute of software product line. Also, the relationships between the security and communication pattern features are modeled by means of the dependencies between the features. In addition, a feature/component table is used to determine which communication and security components are needed to realize each feature. To derive a given secure connector from the feature model, the appropriate features are selected, the components that realize those features are then selected and integrated as described next.

## 4 FEATURE MODEL FOR SECURE CONNECTORS

The feature model (Fig. 1) for secure connectors is developed to describe the variability of secure connectors in terms of communication pattern (CP) and security pattern (SP) features and the dependency relationships between the features. The feature model consists of two at-least-one-of-feature groups, *Communication Patterns* and *Security Patterns*, which means that one or more features need to be selected from each group. The *Communication Patterns* at-least-one-of-feature group is composed of three further feature groups: the *Unidirectional* feature group, which consists of three optional one-way message communication pattern features (CPFs), the *SMC* (synchronous message communication) *without Reply, Broadcast* and *AMC* (asynchronous message communication) features; the *Bidirectional* feature group, which consists of two two-way optional message communication pattern features, *Bidirectional AMC* and *Subscription/Notification* features; and *Message with Single Reply* feature group, which consists of two optional message communication features, *AMC with Callback* and *SMC with Reply* features. The optional communication pattern features (Fig.1) are:

- In *SMC with Reply* feature, a sender component sends a message to a receiver component and waits for a response from the receiver. When a response arrives from the receiver, the sender can continue to work and send the next message to the receiver (Gomaa, 2011; Schneider, 2005).
- In *SMC without Reply* feature, a receiver component acknowledges a sender component when it receives a message from the sender. As the sender is acknowledged by the receiver, it can continue to work and send the next message to the receiver (Gomaa, 2011; Schneider, 2005).
- In *AMC* feature, an asynchronous message is sent from a sender component to a receiver component and is stored in a queue if the receiver is busy. The sender component can continue to send the next message to the receiver component as long as the queue is not full (Gomaa, 2011; Schneider, 2005).
- *Bidirectional AMC* feature uses asynchronous message communication pattern feature in both directions between the sender and receiver components, with the receiver component sending responses to the sender component asynchronously. Responses are sent to a queue from which the sender component retrieves each response (Gomaa, 2011; Schneider, 2005).
- In *AMC with Callback* feature, a sender component sends a service request message to a server component, which includes the client operation (callback) handle. The client com-ponent does not wait for reply. After a

receiver component services the client request, it uses the handle to call the client operation remotely (the callback) (Gomaa, 2011; Schneider, 2005).

- *Subscription/Notification* feature uses *SMC with Reply* communication pattern and asynchronous message communication pattern. Client components subscribe to receive messages of a given type from a server component via *SMC with Reply* communication pattern. When a server component receives message of this type, it notifies all client components that have subscribed to it through AMC pattern (Gomaa, 2011; Schneider, 2005).
- In *Broadcast* feature, a server component sends a message to all client components, regardless of whether clients want the message or not. The client component decides whether it wants to process the message or just discard the message (Gomaa, 2011; Schneider, 2005).

The security pattern features (SPFs) that constitute the *Security Patterns* at-least-one-of-feature group (Fig. 1) are *Authenticator*, *Authorization*, *Hashing*, *Digital Signature* and *Symmetric Encryption*, as follows:

- Symmetric Encryption feature prevents secret information from being disclosed to any unauthorized party. A message sent by a sender to a receiver is encrypted using a secret key, and the encrypted message is decrypted by the receiver (Fernandez-Buglioni, 2013).
- Hashing feature protects against unauthorized changes to secret information. A hash value for a message is generated by a sender and the message with the hash value is sent to a receiver, which verifies the integrity of the message using the hash value (Fernandez-Buglioni, 2013).
- Digital Signature feature protects against one party to a transaction later falsely denying that the transaction occurred. A message is signed by a sender using the sender's private key and the signed message is verified by a receiver to check if the sender has signed the message (Fernandez-Buglioni, 2013).
- Authenticator feature allows an entity (a user or system) to identify itself positively to another entity. This can be achieved using a password, personal-identification number or challenge response (Fernandez-Buglioni, 2013).
- Authorization feature protects against unauthorized access to valuable resources. Authorization may be implemented using mandatory access control, discretionary access control, role-based access control or attribute-based access control (Fernandez-Buglioni, 2013; Ren et al., 2005).

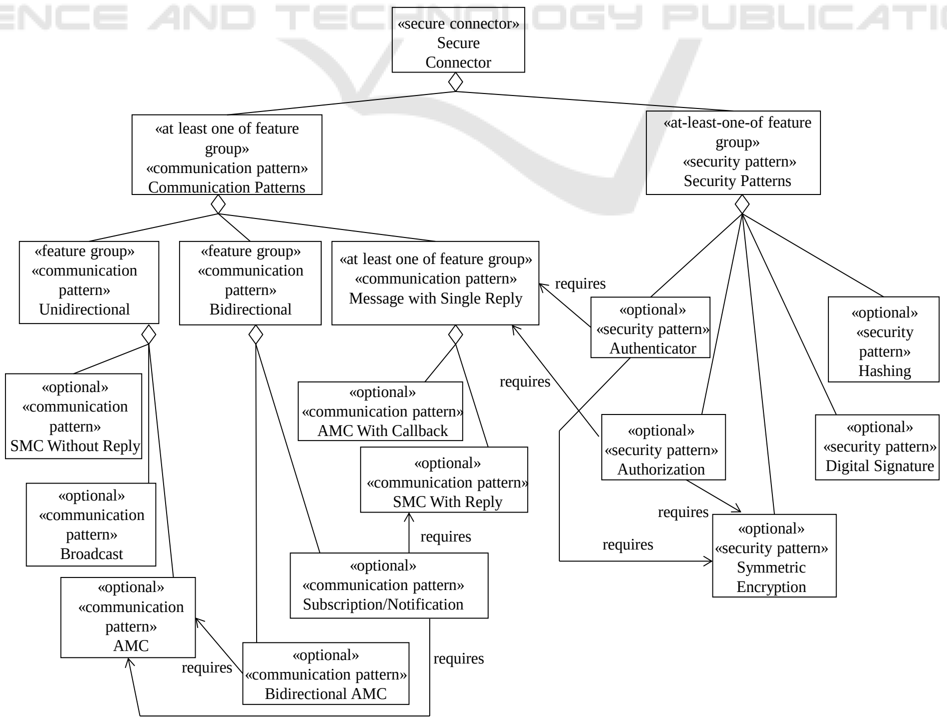
«secure connector» Secure Connector
«at least one of feature group» «communication pattern» Communication Patterns
«at-least-one-of feature group» «security pattern» Security Patterns
«feature group» «communication pattern» Unidirectional
«feature group» «communication pattern» Bidirectional
«at least one of feature group» «communication pattern» Message with Single Reply
requires
«optional» «security pattern» Authenticator
«optional» «security pattern» Hashing
«optional» «communication pattern» SMC Without Reply
«optional» «communication pattern» AMC With Callback
requires
«optional» «security pattern» Authorization
«optional» «security pattern» Digital Signature
«optional» «communication pattern» SMC With Reply
«optional» «communication pattern» Broadcast
requires
requires
requires
«optional» «security pattern» Symmetric Encryption
«optional» «communication pattern» Subscription/Notification
«optional» «communication pattern» AMC
requires
«optional» «communication pattern» Bidirectional AMC
requires

Figure 1: Feature Model for Secure Connectors.

Authenticator feature and Authorization feature each require Symmetric Encryption feature because a sender's credentials for authentication and sender's parameter for requesting permission should be encrypted to prevent unauthorized access. Both Authenticator feature and Authorization feature also require *Message with Single Reply* feature due to their need for a response to the original request. In Authenticator feature, a sender requests authentication from a server and the server needs to reply the authentication result. Similarly, in Authorization feature, a sender requests permission from a receiver and the receiver needs to send back the approval or denial to the sender.

## 5 COMMUNICATION AND SECURITY COMPONENTS

Each feature in the feature model for secure connectors is designed using components, which can be encapsulated into a secure connector that is designed as a composite component. A security pattern feature is designed as security pattern components, whereas a communication pattern feature is designed as communication pattern components. Integration of security pattern and communication pattern components within a secure connector is enabled by means of a security coordinator.

Table 1 shows each feature in the feature model and their components. The *SMC with Reply* feature is designed as a *SMC with Reply Sender* component for sending messages to the receiver, a *SMC with Reply Receiver* component for receiving messages from the sender, and *SMC with Reply Security Sender Coordinator and Receiver Coordinator* Components for sequencing the interactions with one or more security components and with *SMC with Reply* components. Similarly, the *Broadcast*, *AMC*, *Bidirectional AMC* and *SMC without Reply* features are designed as sender and receiver components respectively. Because the *Bidirectional AMC* feature depends on the *AMC* feature, it is designed to use components from the *AMC* feature. Also, *Subscription/Notification* feature is designed to use components from both *AMC* and *SMC with Reply* features due to its dependency on these features. The *Authenticator* feature is designed as *Authenticator* component whereas *Authorization* feature is designed with the *Authorization* component. Each of the *Symmetric Encryption*, *Digital Signature*, and *Hashing* features is designed as two components for sender and receiver components.

Table 1: Feature and their Components.

| Feature | Components | Reuse Stereotype |
|---|---|---|
| SMC Without Reply Feature | SMC Without Reply Sender Component | Optional |
| | SMC Without Reply Receiver Component | Optional |
| | SMC Without Reply Security Sender Coordinator Component | Optional (Variant) |
| | SMC Without Reply Security Receiver Coordinator Component | Optional (Variant) |
| Broadcast Feature | Broadcast Sender Component | Optional |
| | Broadcast Receiver Component | Optional |
| | Broadcast Security Sender Coordinator Component | Optional (Variant) |
| | Broadcast Security Receiver Coordinator Component | Optional (Variant) |
| AMC Feature | AMC Sender Component | Optional |
| | AMC Receiver Component | Optional |
| | AMC Security Sender Coordinator Component | Optional (Variant) |
| | AMC Security Receiver Coordinator Component | Optional (Variant) |
| Bidirectional AMC Feature | Components from AMC Feature | |
| Subscription/ Notification Feature | Components from AMC Feature | |
| | Components from SMC Feature | |
| SMC With Reply Feature | SMC With Reply Sender Component | Optional |
| | SMC With Reply Receiver Component | Optional |
| | SMC With Reply Security Sender Coordinator Component | Optional (Variant) |
| | SMC With Reply Security Receiver Coordinator Component | Optional (Variant) |
| AMC With Callback Feature | AMC With Callback Sender Component | Optional |
| | AMC With Callback Receiver Component | Optional |
| | AMC With Callback Security Sender Coordinator Component | Optional (Variant) |
| | AMC With Callback Security Receiver Coordinator Component | Optional (Variant) |
| Authenticator Feature | Authenticator Component | Optional |
| Authorization Feature | Authorization Component | Optional |
| Symmetric Encryption Feature | Symmetric Encryption Encryptor Component | Optional |
| | Symmetric Encryption Decryptor Component | Optional |
| Digital Signature Feature | Digital Signature Signer Component | Optional |
| | Digital Signature Verifier Component | Optional |
| Hashing Feature | Hashing Signer Component | Optional |
| | Hashing Verifier Component | Optional |
| Public Key Infrastructure Feature | Public Key Repository Component | Optional |

A security pattern feature (SPF) is designed using one or two security pattern components (SPCs), as depicted in Fig. 2. The *Symmetric Encryption* SPF (Fig. 2a) is composed of the symmetric encryption encryptor and decryptor SPCs (Fernandez-Buglioni, 2013). The *Digital Signature* SPF (Fig. 2b) is designed with the digital signature signer SPC (Fig. 2b) and digital signature verifier SPC (Fig. 2b) (Fernandez-Buglioni, 2013). Each port of a component is defined in terms of provided and/or required interfaces (Gomaa, 2011). Each security pattern component (Fig. 2) has a provided port through which the component provides security services to other components. Fig. 3 depicts the interfaces provided by the ports of the SPCs in Fig. 2.

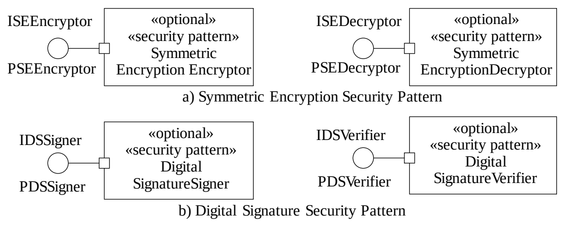
ISEEncryptor
PSEEncryptor
«optional» «security pattern» Symmetric Encryption Encryptor
ISEDecryptor
PSEDecryptor
«optional» «security pattern» Symmetric EncryptionDecryptor
a) Symmetric Encryption Security Pattern
IDSSigner
PDSSigner
«optional» «security pattern» Digital SignatureSigner
IDSVerifier
PDSVerifier
«optional» «security pattern» Digital SignatureVerifier
b) Digital Signature Security Pattern

Figure 2: Security Pattern Components.

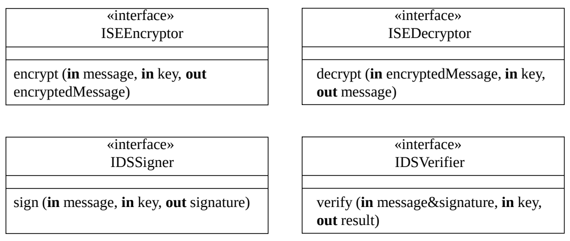
«interface» ISEEncryptor
encrypt (in message, in key, out encryptedMessage)
«interface» ISEDecryptor
decrypt (in encryptedMessage, in key, out message)
«interface» IDSSigner
sign (in message, in key, out signature)
«interface» IDSVerifier
verify (in message&signature, in key, out result)

Figure 3: Interfaces of Security Pattern Components.

Each communication pattern is designed with a sender communication pattern component (CPC) and a receiver communication pattern component (CPC), which are encapsulated in a secure sender connector and a secure receiver connector respectively. Fig. 4a depicts the AMC Sender CPC and AMC Receiver CPC for the secure AMC connector. The AMC Sender CPC (Fig. 4a) has the provided PAsyncMCSenderService port through which the Security Sender Coordinator component (Fig. 7a) sends to the AMC Sender CPC a message being sent to the receiver component, whereas it requests a service from the AMC Receiver CPC via the required RNetwork port. Similarly, the AMC Receiver CPC (Fig. 4a) has the required RSecurityService port and provided PNetwork port. Fig. 4b depicts the interfaces provided by each port of the AMC Sender and Receiver CPCs.

## 6 SECURITY COORDINATOR COMPONENTS

A security sender coordinator component receives messages from a sender component, and a security receiver coordinator component delivers messages to a receiver component. The security sender and receiver coordinator components are variant optional components (Table 1), optional because they are needed for each optional communication pattern, and variant because the design of each coordinator component needs to be customized for each secure connector based on one or more selected security features. Templates for the high-level security sender and receiver coordinator components are designed for each communication pattern. A communication pattern needs one template for the high-level security sender coordinator component (see Fig. 5) and another template for the receiver coordinator component (Fig, 6). The templates are customized for each secure connector based on the security features selected.

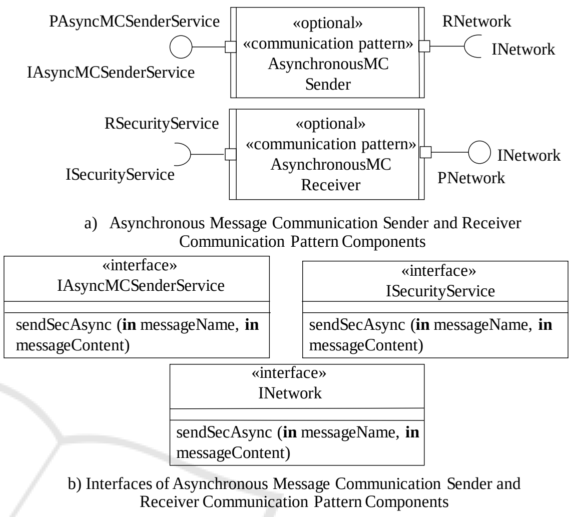
PAsyncMCSenderService
IAsyncMCSenderService
«optional» «communication pattern» AsynchronousMC Sender
RNetwork
INetwork
RSecurityService
ISecurityService
«optional» «communication pattern» AsynchronousMC Receiver
INetwork
PNetwork
a) Asynchronous Message Communication Sender and Receiver Communication Pattern Components
«interface» IAsyncMCSenderService
sendSecAsync (in messageName, in messageContent)
«interface» ISecurityService
sendSecAsync (in messageName, in messageContent)
«interface» INetwork
sendSecAsync (in messageName, in messageContent)
b) Interfaces of Asynchronous Message Communication Sender and Receiver Communication Pattern Components

Figure 4: Asynchronous Message Communication Sender and Receiver Communication Pattern Components and their Interfaces.

```
loop
-- Wait for message from sender component;
receive (SenderComponentMessageQ, message);
Extract MessageName, MessageContent and
SenderSecurityPatternAttribute from message;

-- Apply security patterns to message content;
 while SecurityPatternsRequiredByMessageContent do
        Apply security pattern to message content;
end while;

-- Send message to AMC Sender CPC;
AsynchronousMCSender.sendSecAsync (in MessageName, in
MessageContent);

end loop;
```

Figure 5: Pseudocode template for Security Sender Coordinator in Secure AMC Connector.

The pseudocode template for the security sender coordinator is depicted in Fig. 5 in which the security related code (in italics) is replaced by the pseudocode for the security patterns selected for a secure AMC connector, as described in Section 7 depicted in Fig. 7a. Similarly, the pseudocode template for the Security Receiver Coordinator is specified in Fig. 6 in which the interfaces of security receiver coordinator component are depicted in Fig.

7b. The pseudocode templates for security sender coordinator component (Fig. 5) and security receiver coordinator component (Fig. 6) are customized for a secure AMC connector that encapsulates Symmetric Encryption and Digital Signature SCPs, as described in Section 7 and depicted in Figs. 8 and 9.

```
Loop

 -- Wait for message from AMC Receiver CPC;
receive (AMCReceiverMessageQ, message);
Extract MessageName and MessageContent from message;

-- Apply security patterns to message content;
while SecurityPatternsRequiredByMessageContent do
        Apply security pattern to message content;
end while;

-- Send message name and message content to receiver component;
if MessageContent is secure
then
        ReceiverComponent.sendSecAsync (in MessageName, in
          MessageContent);
end if;

end loop;
```

Figure 6: Pseudocode template for Security Receiver Coordinator in Secure AMC Connector.

## 7 EXAMPLE OF SECURE CONNECTOR

This section describes an example of a secure connector that can be derived from the SPL for secure connectors if an application requires AMC CPF with Symmetric Encryption and Digital Signature SPFs. This needs the selection of the one communication pattern (AMC) and two security patterns, namely Symmetric Encryption and Digital Signature features (Fig. 1). The corresponding components (from Table 1) are the AMC Sender and Receiver components, Symmetric Encryption Encryptor and Decryptor components, and Digital Signature Signer and Verifier components (Table 1). This secure AMC connector is composed of a secure AMC sender connector (Fig. 10) and a secure AMC receiver connector (Fig. 10). The secure AMC sender connector (Fig. 10) is designed as a composite component in which the Security Sender Coordinator component (Fig. 7a) integrates the Symmetric Encryption Encryptor and Digital Signature Signer SPCs (Fig. 2) for the Symmetric Encryption and Digital Signature SPFs with the AMC Sender CPC (Fig. 4) for the AMC CPF.

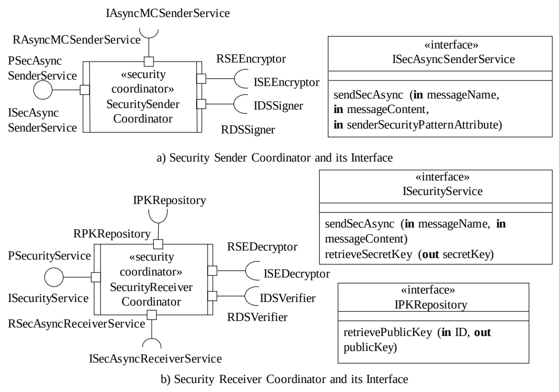
IAsyncMCSenderService
RAsyncMCSenderService
PSecAsync SenderService
ISecAsync SenderService
«security coordinator» SecuritySender Coordinator
RSEEncryptor
ISEEncryptor
IDSSigner
RDSSigner
«interface» ISecAsyncSenderService
sendSecAsync (in messageName, in messageContent, in senderSecurityPatternAttribute)
a) Security Sender Coordinator and its Interface
IPKRepository
RPKRepository
PSecurityService
ISecurityService
RSecAsyncReceiverService
ISecAsyncReceiverService
«security coordinator» SecurityReceiver Coordinator
RSEDecryptor
ISEDecryptor
IDSVerifier
RDSVerifier
«interface» ISecurityService
sendSecAsync (in messageName, in messageContent)
retrieveSecretKey (out secretKey)
«interface» IPKRepository
retrievePublicKey (in ID, out publicKey)
b) Security Receiver Coordinator and its Interface

Figure 7: Security Sender and Receiver Coordinators and their interfaces for Secure AMC Connector with Symmetric Encryption and Digital Signature security pattern features.

Fig. 7a depicts the interface provided by the security sender coordinator for a secure AMC connector. The senderSecurityPatternAttribute parameter in sendSecAsync() specifies the private key or secret key that is needed by security pattern components to apply security services to a message. For integrating the components, the Security Sender Coordinator component (Fig. 7a) has a required RSEEncryptor port to communicate with a provided PSEEncryptor port of the Symmetric Encryption Encryptor SPC, which encrypts messages using the sender's secret key, and it also has a required RDSSigner port to communicate with a provided PDSSigner port of the Digital Signature Signer SPC, which signs a message using the sender's private key. The signed and encrypted messages are sent to the receiver component. The pseudocode for the Secure Sender Coordinator component is depicted in Fig. 8. Similarly, the AMC Receiver Connector (Fig. 10) is designed as a composite component that encapsulates the Security Receiver Coordinator component (Fig. 7b), Symmetric Encryption Decryptor SPC (Fig. 2), Digital Signature Verifier SPC (Fig. 2), and AMC Receiver CPC (Fig. 4). The pseudocode for the Secure Receiver Coordinator component is depicted in Fig. 10.

Fig. 10 depicts the structural view of the secure AMC connector with Symmetric Encryption security pattern and Digital Signature security pattern, which can be applied for confirming a shipment in a business to business (B2B) electronic commerce application. When a Supplier component sends a shipment confirmation to a Delivery Order Server, the shipment confirmation is signed by the Digital Signature Signer SPC in the secure AMC

```
loop

-- Wait for message from sender component;
receive (SenderComponentMessageQ, message);
Extract MessageName, MessageContent, PrivateKey, and
SecretKey from message;

-- Apply security patterns to message content;
if MessageContent requires non-repudiation
then
        DigitalSignatureSigner.sign (in MessageContent,
                in PrivateKey, out SignedMessageContent);
        MessageContent = SignedMessageContent;
end if;
if MessageContent requires confidentiality
then
        SymmetricEncryptionEncryptor.encrypt (
                in MessageContent, in SecretKey,
                out EncryptedMessageContent);
        Message Content = EncryptedMessageContent;
end if;

-- Send message to AMC Sender CPC;
AsynchronousMCSender.sendSecAsync (in MessageName,
        in MessageContent);

end loop;
```

Figure 8: Pseudocode of Security Sender Coordinator for Secure AMC Connector with Symmetric Encryption and Digital Signature Security Pattern features.

```
loop

-- Wait for message from AMC Receiver CPC;
receive (AMCReceiverMessageQ, message);
Extract MessageName and MessageContent from message;

-- Apply security patterns to message content;
if MessageContent requires confidentiality
then
        ReceiverComponent.retrieveSecretKey (out SecretKey);
        SymmetricEncryptionDecryptor.decrypt (in
                EncryptedMessageContent&Signature,
        in SecretKey, out MessageContent&Signature);
end if;
if MessageContent requires non-repudiation
then
        PublicKeyRepository.retrievePublicKey (in SenderID,
                out SenderPublicKey);
        DigitalSignatureVerifier.verify (in MessageContent&Signature,
                in Key, out Result);
end if;

-- Send message name and message content to receiver component;
if Signature is verified
then
        ReceiverComponent.sendSecAsync (in MessageName,
                in MessageContent);
end if;
```

Figure 9: Pseudocode of Security Receiver Coordinator for Secure AMC Connector with Symmetric Encryption and Digital Signature Security Pattern features.

sender connector assuming the Digital Signature security pattern feature is selected for Supplier component. The shipment confirmation and signature is then encrypted by the Symmetric Encryption Encryptor SPC in the secure AMC sender connector assuming the Symmetric Encryption security pattern feature is also selected for Supplier component. The encrypted shipment confirmation and signature are decrypted by the Symmetric Encryption Decryptor SPC, and then sent to the Delivery Order Server via the secure AMC receiver connector, which requests the sender component's public key from the Public Key Repository SPC (Table 1) that is designed for a certificate authority in the public key infrastructure feature (Table 1). The signature is verified by the secure AMC receiver connector with the sender's public key. The behavioral view of a secure AMC connector can be depicted using UML communication or sequence diagrams. An example is described in (Shin et al., 2016a) for confidentiality and non-repudiation security services.

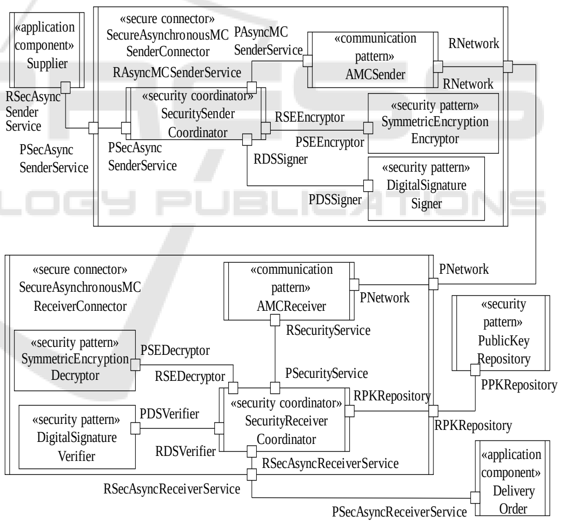
«application component» Supplier
«secure connector» SecureAsynchronousMC SenderConnector
PAsyncMC SenderService
«communication pattern» AMCSender
RNetwork
RAsyncMCSenderService
RNetwork
RSecAsync Sender Service
«security coordinator» SecuritySender Coordinator
RSEEncryptor
«security pattern» SymmetricEncryption Encryptor
PSecAsync SenderService
PSecAsync SenderService
PSEEncryptor
RDSSigner
«security pattern» DigitalSignature Signer
PDSSigner
«secure connector» SecureAsynchronousMC ReceiverConnector
«communication pattern» AMCReceiver
PNetwork
PNetwork
RSecurityService
«security pattern» PublicKey Repository
«security pattern» SymmetricEncryption Decryptor
PSEDecryptor
RSEDecryptor
PSecurityService
PPKRepository
RPKRepository
«security pattern» DigitalSignature Verifier
PDSVerifier
«security coordinator» SecurityReceiver Coordinator
RPKRepository
RDSVerifier
RSecAsyncReceiverService
«application component» Delivery Order
RSecAsyncReceiverService
PSecAsyncReceiverService

Figure 10: Secure AMC Connector with Symmetric Encryption and Digital Signature security pattern features in B2B application.

## 8 VALIDATION

The secure connectors derived from the software product line were validated from the perspectives of implementation and performance analysis of secure connectors.

## 8.1 Implementation of Secure Connectors

The secure connectors designed using the software product line approach were implemented in Java. The implementation environment used is as follows: Eclipse 4.4.2 version on a Windows 7, 64-bit-based computer with 4GB of memory and 2.20 GHz quad core i7 processor. All secure message communications between application components were implemented in a local machine.

The secure asynchronous message communication connector (Fig. 10) was implemented using asynchronous message communication CPF and both symmetric encryption and digital signature CPFs. The connector was implemented using two algorithms for security pattern features, DSA to sign/verify the message in digital signature SPF and DES to encrypt/ decrypt the message in symmetric encryption SPF. The shipment confirmation using the secure asynchronous message communication connector in B2B application (Fig. 10) was implemented with 7 threads for each of supplier component security sender coordinator component, AMC sender CPC, security receiver coordinator component, AMC receiver CPC, public key repository SPC, and delivery order component. Also, a message queue was implemented and placed between threads (for example, a message queue between the supplier application component thread and security sender coordinator component thread, and a message queue between the security sender coordinator component thread and the AMC sender CPC thread).

In addition, another secure connector constructed with secure synchronous message communication with reply CPF and authenticator and symmetric encryption SPFs in B2B application was implemented for the validation. Encryption and decryption SPCs for the symmetric encryption SP were implemented using the data encryption standard (DES) algorithm, which is a block cipher that operates on plain text blocks of a given size (64-bits) and returns cipher text blocks of the same size. The DES works by using the same a 56-bit key to encrypt and decrypt a message. The encrypted Payment Order is sent to the Payment Order application component, the receiver connector of which decrypts the message using the DES algorithm with the same secret key. Authenticator SPC uses MySQL database to store customer's credentials and authenticate them. The size of the MySQL database affects the execution time for the authenticator SPC if the database contains a large amount of customer credentials. The payment order using the secure SMC with reply connector was implemented with 7 threads and 5 separate message buffers. These threads were implemented with customer component, security sender coordinator component, SMC with reply sender CPC, SMC with reply receiver CPC, security receiver coordinator, authenticator SPC, and payment order component. Message buffers were implemented and placed between threads.

## 8.2 Performance Analysis of Secure Connectors

This section describes the performance analysis of secure applications using the secure connectors derived from the software product line and compares them with secure applications executing the same message communication patterns but using other approaches for providing or not providing security. The three approaches compared in this section are the (1) *with secure connector approach*, for secure applications that use the approach described in this paper; (2) *without security service approach*, for applications that do not provide any security services; (3) *without secure connector approach*, for secure applications in which security services are mingled with the application logic. In the *with secure connector approach*, security services are encapsulated in secure connectors separately from application logic. The application functionality implemented in each approach is all the same. However, the underlying difference between *with secure connector approach* and *without secure connector approach* is that the security services in *without secure connector approach* are implemented within application components along with application business logic, whereas *with secure connector approach* separated the security services from application components and implemented them as secure connectors. Also, the difference between *with secure connector approach* and *without security service approach* is that *with secure connector approach* provides security services encapsulated in secure connectors with application components, whereas *without security service approach* implements only business application logic without any security services.

For each communication pattern described in this paper, namely (a) synchronous message communication with Authenticator and Symmetric Encryption security patterns and (b) asynchronous message communication with Symmetric

Encryption and Digital Signature security patterns, the performance of the *with secure connector approach* was evaluated by measuring the average time of message communication between sender and receiver components via a secure connector. Each message communication implemented in section 8.1 was run 20 times to calculate the average communication time so that the performance evaluation would not be dependent on a few exceptional communication times. Message communication time (MCT) is measured by observing the overall run time from start to finish for each message communication pattern. For synchronous communication, MCT measures sending a message and receiving a response. For asynchronous communication, MCT measures the time to send a message from sender to receiver.

Table 2 shows the average time of multiple message communication and a comparison of the *with secure connector approach*, *without security service approach*, and *without secure connector approach* for the first pattern. For the *with secure connector approach*, the second column of Table 2 (top section) shows that the average MCT is 44.6 milliseconds (ms) for the sender connector of SMCWR with Authenticator and Symmetric Encryption security patterns, 0.02 milliseconds for the network connection, and 296.8 milliseconds for the receiver connector. Thus the overall MCT for the SMCWR with Authenticator and Symmetric Encryption security pattern is 341.4 milliseconds. For the second pattern the *with secure connector approach* (bottom section of second column) shows the MCT is 44.8 milliseconds for the sender connector of AMC with Symmetric Encryption and Digital Signature security patterns, 0.01 milliseconds for the network connection, 48.9 milliseconds for the receiver connector, giving a total of 93.7 milliseconds for the overall of AMC with Symmetric Encryption and Digital Signature security patterns (Fig. 10).

The *without security service approach* (top section in the third column of Table 2) shows that the average MCT for SMCWR with Authenticator and Symmetric Encryption security patterns is 2.9 ms, 0.02 ms and 8.0 ms for each portion, while the overall average time is 10.9 ms. The bottom section of the *without security service approach* shows that AMC with Symmetric Encryption and Digital Signature security patterns (Fig. 10) has 5.0 ms, 0.01 ms and 4.3 ms for each portion. AMC with Symmetric Encryption and Digital Signature security patterns has a total MCT of 9.3 ms for the *without security service approach.*

The *without secure connector approach* (fourth column of Table 2) shows that the average MCT for SMCWR with Authenticator and Symmetric Encryption security patterns is 41.7 ms, 0.02 ms, 295.9 ms for each portion, giving an overall MCT of 340.0 ms. The bottom section of the same column shows that the AMC with Symmetric Encryption and Digital Signature security patterns (Fig. 10) has MCT of 44.3 ms, 0.01ms and 47.4 ms for each portion. The overall average MCT for AMC with Symmetric Encryption and Digital Signature security patterns is 91.7 ms.

The fifth column of Table 2 indicates that the time difference between the *with secure connector approach* and the *without security service approach* is highly significant. This is because *with secure connector approach* provides application components with security services such as confidentiality and non-repudiation. The security services in the *with secure connector approach* consume processing time for encrypting/decrypting messages, authenticating messages and/or signing/verifying digital signature, whereas the *without security service approach* is much faster due to it providing no security services. It is also important to note that use of authenticator pattern increases the run time of the program due to the time taken to do database retrievals. Thus, the additional processing time taken by the *with secure connector approach* is to make applications secure in comparison to insecure applications developed using the *without security service approach*.

Comparing the performance *without secure connector approach* and *with secure connector approach* shows that there is no significant difference in the runtime performance of the communication patterns. The time difference between the two approaches (sixth column in Table 2) ranges from 1.4 ms to 2.0 ms. Both approaches provide applications with security services; however, the *with secure connector approach* has the advantage of separating security services from application logic, which leads to secure software architectures that are more maintainable and evolvable than the *without secure connector approach*.

Table 2: Average time of message communication and comparison of the with secure connector approach, without security service approach, and without secure connector approach.

| Communication pattern | With secure connector approach | Without security service approach | Without secure connector approach | Time difference between with secure connector approach and without security service approach | Time difference between without secure connector approach and with secure connector approach |
|---|---|---|---|---|---|
| **Secure SMCWR Connector with Authenticator and Symmetric Encryption security patterns** | | | | | |
| ● Secure Synchronous MC with reply Sender | 44.6 ms | 2.9 ms | 41.7 ms | 41.7 ms | 2.9 ms |
| ● Network connection | 0.02 ms | 0.02 ms | 0.02 ms | 0.00 ms | 0.00 ms |
| ● Secure Synchronous MC with reply Receiver | 296.8 ms | 8.0 ms | 295.9 ms | 288.8 ms | 0.9 ms |
| ● Full SMCWR with Symmetric Encryption & Authenticator | 341.4 ms | 10.9 ms | 340.0 ms | 330.5 ms | 1.4 ms |
| **Secure AMC Connector with Symmetric Encryption and Digital Signature security patterns (Fig. 10)** | | | | | |
| ● Secure Asynchronous MC Sender (Fig. 8) | 44.8 ms | 5.0 ms | 44.3 ms | 39.8 ms | 0.5 ms |
| ● Network connection (Fig. 8) | 0.01 ms | 0.01 ms | 0.01 ms | 0.00 ms | 0.00 ms |
| ● Secure Asynchronous MC Receiver (Fig. 8) | 48.9 ms | 4.3 ms | 47.4 ms | 44.6 ms | 1.5 ms |
| ● Full AMC with Symmetric Encryption & Digital Signature (Fig. 8) | 93.7 ms | 9.3 ms | 91.7 ms | 84.4 ms | 2.0 ms |

## 9 CONCLUSIONS

This paper has described a software product line approach to model the variability of secure connectors in terms of security patterns and communication patterns, which makes it possible to design secure software architectures for concurrent and distributed software applications. The feature model for secure connectors captures various security pattern and communication pattern features, and describes the relationships between features. The security and communication pattern features are designed as security and communication pattern components that are encapsulated into secure connectors. Each secure connector is derived from the software product line for secure connectors, which is designed as a composite component that encapsulates both security pattern and communication pattern components. A security coordinator enables security pattern and communication pattern components to be integrated within a secure connector. This paper has also described a secure AMC connector, which is designed with the security pattern and communication pattern features selected for the applications.

This paragraph describes future research for secure connectors. The code of a security coordinator could be generated automatically from a code template of a high-level security coordinator. As security and communication pattern features are selected for an application, the template could be automatically filled with calls to the appropriate methods of the corresponding pattern components. A prototype tool could also be developed to automatically generate the code for security coordinators within secure connectors. In addition, we might need to investigate how multiple communication pattern components could be encapsulated within a secure connector when sender and receiver application components communicate with each other via different types of communication patterns.

## ACKNOWLEDGEMENTS

This work was partially supported by the AFOSR grant FA9550-16-1-0030.

## REFERENCES

Abu-Matar, M. and Gomaa, H., 2011, August. Variability modeling for service oriented product line architectures. In *Software Product Line Conference (SPLC), 2011 15th International* (pp. 110-119). IEEE.

Al-Azzani, S. and Bahsoon, R., 2012, August. SecArch: Architecture-level evaluation and testing for security. In *Software Architecture (WICSA) and European Conference on Software Architecture (ECSA), 2012 Joint Working IEEE/IFIP Conference on* (pp. 51-60).

Albassam, E., Gomaa, H. and Menascé, D.A., 2016, July. Model-based Recovery Connectors for Self-adaptation and Self-healing. In *ICSOFT-EA* (pp. 79-90).

Basin, D., Clavel, M. and Egea, M., 2011, June. A decade of model-driven security. In *Proceedings of the 16th ACM symposium on Access control models and technologies* (pp. 1-10). ACM.

Farahmandian, S. and Hoang, D.B., 2017, October. SDS$_2$: A novel software-defined security service for protecting cloud computing infrastructure. In *Network Computing and Applications (NCA), 2017 IEEE 16th International Symposium on* (pp. 1-8). IEEE.

Fant, J.S., Gomaa, H. and Pettit, R.G., 2015, July. Integrating and applying architectural design patterns in space flight software product lines. In *Software Technologies (ICSOFT), 2015 10th International Joint Conference on* (Vol. 1, pp. 1-11). IEEE.

Fernandez-Buglioni, E., 2013. *Security patterns in practice: designing secure architectures using software patterns*. John Wiley & Sons.

Gomaa, H., Menascé, D.A. & Shin, M.E., 2001. Reusable component interconnection patterns for distributed software architectures. *Proceedings of the 2001 symposium on Software reusability putting software reuse in context - SSR 01*.

Gomaa, H. and Shin, M.E., 2004, July. A multiple-view meta-modeling approach for variability management in software product lines. In *International Conference on Software Reuse* (pp. 274-285). Springer, Berlin, Heidelberg.

Gomaa, H., 2005. Designing software product lines with UML: from use cases to pattern-based software architectures, Boston: Addison-Wesley.

Gomaa, H. and Shin, M.E., 2007, January. Automated software product line engineering and product derivation. In *System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on* (pp. 285a-285a). IEEE.

Gomaa, H. and Shin, M.E., 2008. Multiple-view modelling and meta-modelling of software product lines. *IET software*, 2(2), pp.94-122.

Gomaa, H., Hashimoto, K., Kim, M., Malek, S. and Menascé, D.A., 2010, March. Software adaptation patterns for service-oriented architectures. In *Proceedings of the 2010 ACM Symposium on Applied Computing* (pp. 462-469). ACM.

Gomaa, H. and Shin, M.E., 2010. Variability modeling in model-driven software product line engineering. In *Proceedings of the 2nd International Workshop on Model Driven Product Line Engineering (MDPLE 2010)* (p. 65).

Gomaa, H., 2011. *Software modeling and design: UML, use cases, patterns, and software architectures*. Cambridge University Press.

Jürjens, J., 2002, September. UMLsec: Extending UML for secure systems development. In *International Conference on The Unified Modeling Language* (pp. 412-425). Springer, Berlin, Heidelberg.

Lodderstedt, T., Basin, D. and Doser, J., 2002, September. SecureUML: A UML-based modeling language for model-driven security. In *International Conference on the Unified Modeling Language* (pp. 426-441). Springer, Berlin, Heidelberg.

Medvidovic, N. and Taylor, R.N., 2010, May. Software architecture: foundations, theory, and practice. In *Software Engineering, 2010 ACM/IEEE 32nd International Conference on* (Vol. 2, pp. 471-472). IEEE.

Ren, J., Taylor, R., Dourish, P. and Redmiles, D., 2005, May. Towards an architectural treatment of software security: a connector-centric approach. In *ACM SIGSOFT Software Engineering Notes* (Vol. 30, No. 4, pp. 1-7). ACM.

Schneider, R., 2005. Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures. *Software Quality Professional*, *7*(4), p.46.

Schumacher, M., Fernandez, E.B., Hybertson, D., Buschmann, F. and Sommerlad, P., 2006. Security Patterns, J.

Shin, M.E. and Gomaa, H., 2007. Software modeling of evolution to a secure application: From requirements model to software architecture. *Sci. Comput. Program*, *66*(1), pp.60-70.

Shin, M.E., Malhotra, B., Gomaa, H. and Kang, T., 2012, July. Connectors for Secure Software Architectures. In *SEKE* (pp. 394-399).

Shin, M.E., Gomaa, H., Pathirage, D., Baker, C. and Malhotra, B., 2016. Design of Secure Software Architectures with Secure Connectors. *International Journal of Software Engineering and Knowledge Engineering*, *26*(05), pp.769-805.

Shin, M., Gomaa, H. and Pathirage, D., 2016, June. Reusable Secure Connectors for Secure Software Architecture. In *International Conference on Software Reuse* (pp. 181-196). Springer, Cham.

Shin, M., Gomaa, H. and Pathirage, D., 2017. Model-based Design of Reusable Secure Connectors. In *4st International Workshop on Interplay of Model-Driven and Component-Based Software Engineering (ModComp) 2017 Workshop Pre-proceedings* (p. 6).

Taha, A., Trapero, R., Luna, J. and Suri, N., 2017, June. A Framework for Ranking Cloud Security Services. In *Services Computing (SCC), 2017 IEEE International Conference on* (pp. 322-329). IEEE.

Tzeremes, V. and Gomaa, H., 2018, January. Applying End User Software Product Line Engineering for Smart Spaces. In *Proceedings of the 51st Hawaii International Conference on System Sciences*.