Lecture 8

Design of component-based software Architecture

Design of Secure Connectors for secure CBSA

1

Designing Component-based Software Architecture
Reference: H. Gomaa, Chapters 17, 23 - Software Modeling and Design, Cambridge University Press, 2011

Design of Secure Connectors
Michael E. Shin, H. Gomaa, D. Pathirage, C. Baker, and B. Malhotra, "Design of Secure Software Architectures with Secure Connectors," International Journal of Software Engineering and Knowledge Engineering, Volume 26, No. 4, 2016, pp. 769-805.

Michael E. Shin, Hassan Gomaa, and Don Pathirage, "Model-based Design of Reusable Secure Connectors," 4th International Workshop on Interplay of Model-Driven and Component-Based Software Engineering (ModComp2017), September 17, Austin/Texas, USA, 2017.

2

Design of Component-Based Software Architecture (CBSA)

- Goal of CBSA
  - Provide a concurrent message-based design that is highly configurable
    - Same software architecture is deployed to many different distributed configurations
    - How components will be mapped to physical nodes
      - Not made at design time, but at system deployment time
    - All communication between components must be restricted to message communication

3

Component-Based Software Architecture (CBSA)

- Designing configurable components
  - Deployed to execute on distributed nodes
- Design of component interface
  - Component ports
  - Component provided and required interfaces
  - Connectors for joining compatible ports
- Architectural communication patterns
  - Synchronous, Asynchronous, Broker, Group communication patterns
- Depicted with UML composite structure diagrams

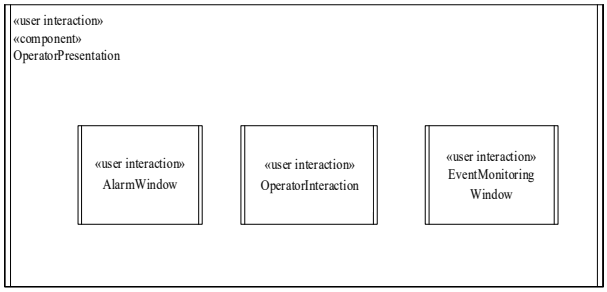
4

Composite Component

- Composite Component
  - Encapsulate internal component (objects)
  - Configurable unit
    - Objects that are part of a composite component must reside at the same location
  - Fig. 17.1
  - Pass-through mechanism
    - Incoming messages to a composite component are passed through to internal component
    - Outgoing message from inter component are passed through to external components

5

Figure 17.1 Composite Component



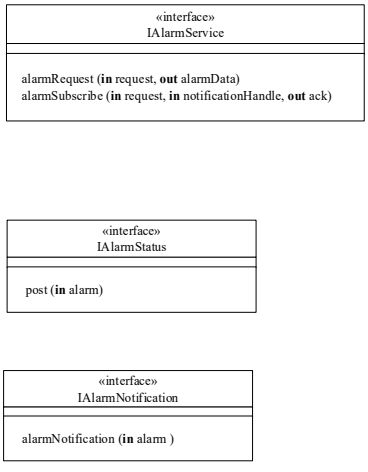
6

Design of Component Interfaces

- Interface
  - Specifies externally visible operations of a component
  - A component can provide more than one interface
  - Fig. 17.2: Alarm Service component with three interfaces
    - IAlarmService
    - IAlarmStatus
    - IAlarmNotification

7

Figure 17.2 Example of Component Interfaces



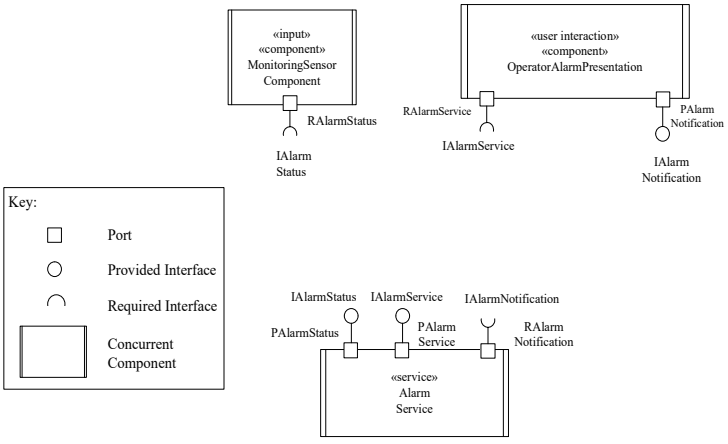
8

Design of Component Interfaces

- Provided Interfaces
  - Specifies the operations that a component must fulfill
- Required Interfaces
  - Describes the operations that other components provide for this component
- Port
  - A component has one or more ports
  - Through which a component interacts with other components
  - Provided port supports provided interface
  - Required port supports required interface
  - Complex port supports both provided and required interfaces
- Fig. 17.3

9

Figure 17.3 Example of component ports, provided and required interfaces



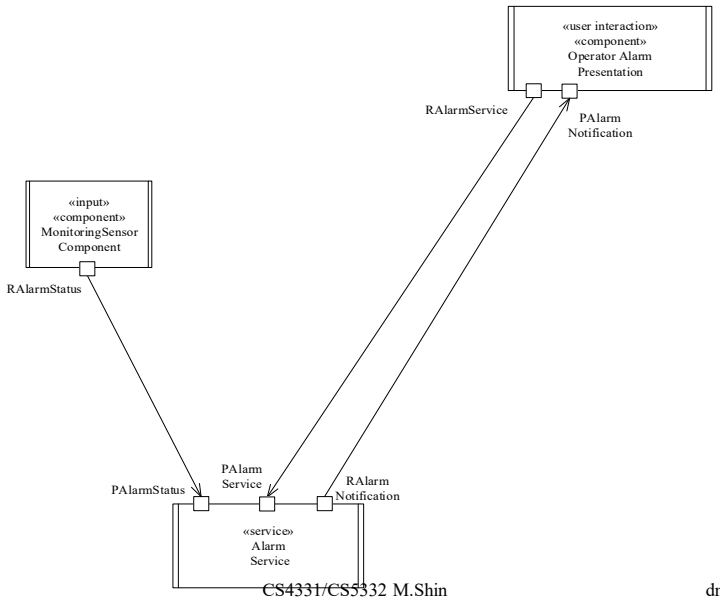
10

Design of Component Interfaces

- Connector
  - Joins the required port of one component to the provided port of another
  - The connected ports must be compatible with each other
  - The operations required in one component's required interface
    - same as the operations provided in the other component's provided interface
- Unidirectional/bidirectional connector
  - Fig. 17.4

11

Figure 17.4 Example of components, ports and connectors in software architecture



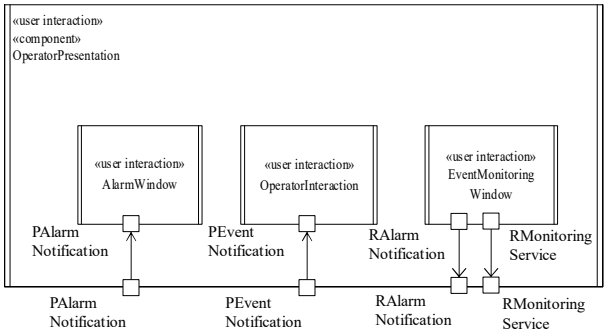
12

Design of Component Interfaces

- Designing Composite Components
  - Simple component and composite component
  - Delegation connector
    - Fig. 23.15
  - Pass-through mechanism
    - Incoming messages to a composite component are passed through to internal component
    - Outgoing message from inter component are passed through to external components

13

Fig. 23.15 User Interaction Components



14

Design of Component Interfaces

- Only composite component is named as a component
  - Internal object (concurrent object) is not a component
    - They are instances of the component
  - Only composite component can be configurable
    - The internal objects cannot be configurable separately

15

Design of secure connector for secure CBSA

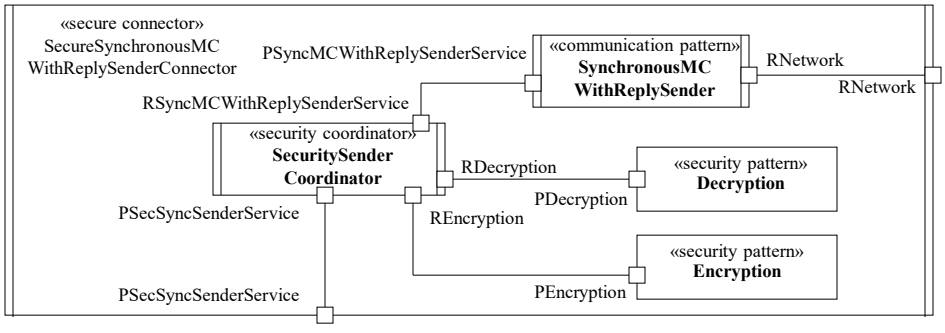
- Component-based software architecture (CBSA)
  - Composed of components and connectors
  - Connectors encapsulate communication mechanisms between components
- Designing secure CBSA
  - Separate security concerns from application concerns
  - Provide security services for secure applications
- Design of secure connectors
  - Security services are encapsulated in connectors
  - Separately from communication patterns

16

Secure Connector

- Secure connector consists of
  - Secure sender connector on sender node
  - Secure receiver connector on receiver node
- Secure connector designed as a composite component
  - One or more security pattern components
    - Encapsulates security service
      - Required by application components
  - Message communication pattern component
    - Encapsulates inter-component message communication protocol
  - Security coordinator
    - Integrates security and communication patterns

Example of Secure Connector (UML 2 composite structure diagram)



Design of Security Pattern Components

- A security service realized by security techniques
  - E.g., confidentiality security service
    - Protect against information being revealed to unauthorized entities
    - Symmetric encryption, asymmetric encryption
  - E.g., non-repudiation security service
    - Prevent one party from falsely denying that a business transaction occurred
    - Digital signature
- A security pattern is a specific security technique that realizes a security service
  - E.g., Symmetric encryption security pattern
  - E.g., Digital signature security pattern

19

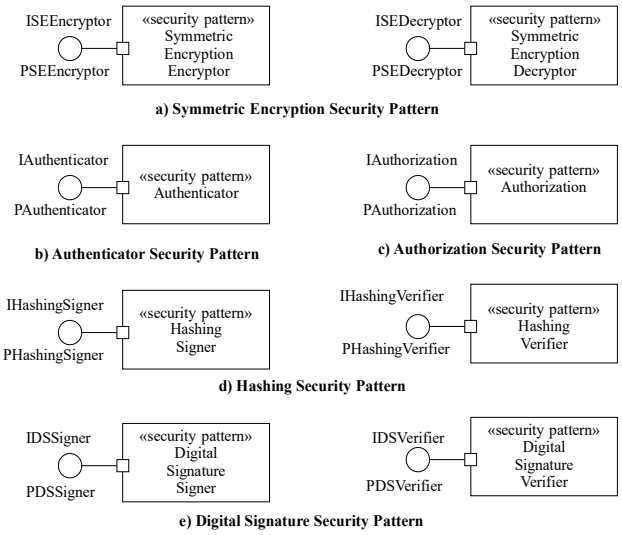
19

Design of Security Pattern Components (SPC)

- Security pattern designed using security pattern components (Fig 1)
  - Symmetric encryption security pattern
    - Symmetric Encryption Encryptor SPC
    - Symmetric Encryption Decryptor SPC
  - Hashing security pattern
    - Hashing Signer SPC
    - Hashing Verifier SPC
  - Digital signature security pattern
    - Digital Signature Signer SPC
    - Digital Signature Verifier SPC
  - Authenticator security pattern
    - Authenticator SPC
  - Authorization security pattern
    - Authorization SPC

20

Fig. 1 Security Pattern Components



Design of Security Pattern Component (SPC)

- SPC interfaces accessed via ports
- Each security pattern component has
  - Provided interface
    - Specifies externally visible security operations
- E.g., Symmetric encryption security pattern
  - Symmetric Encryption Encryptor SPC
    - » Provided PSEEncryptor port
    - » Provided ISEEncryptor interface
- Fig. 2 Interfaces of Security Pattern Components

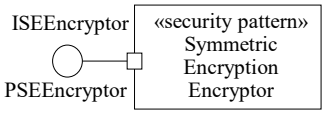
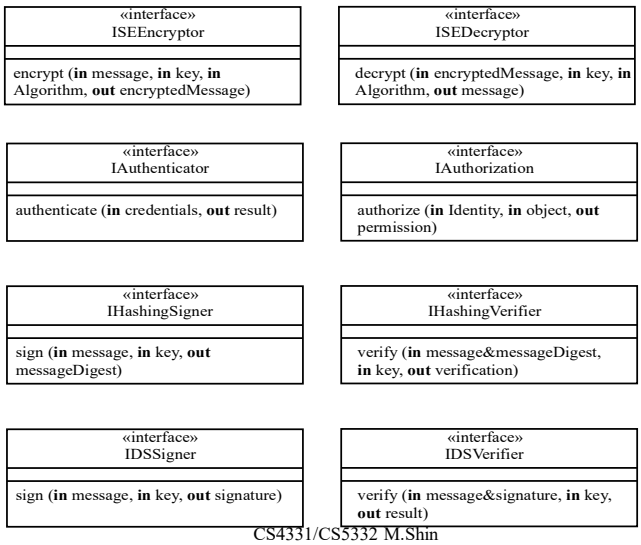


Fig. 2 Interfaces of Security Pattern Components



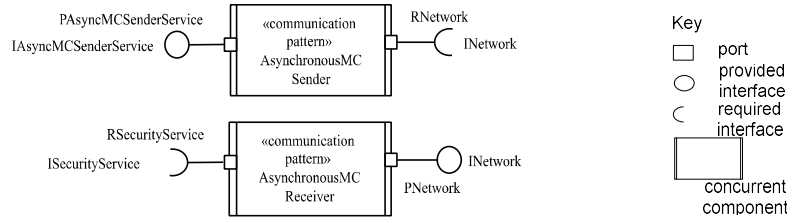
Design of Communication Pattern Components

- Message Communication Patterns
  - Synchronous message communication with reply
  - Synchronous message communication without reply
  - Asynchronous message communication
  - Bidirectional asynchronous message communication
- Each communication pattern designed with
  - Sender communication pattern component
    - Encapsulated in a secure sender connector
  - Receiver communication pattern component
    - Encapsulated in a secure receiver connector

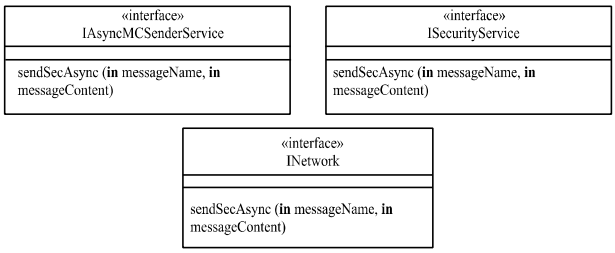
Design of Communication Pattern Components (CPC)

- Asynchronous message communication
  - Message sent from sender component to receiver component
  - Messages buffered in internal queue
- Secure asynchronous message communication connector (Fig. 3)
  - Asynchronous Message Communication Sender CPC
    - Provided PAsyncMCSenderService port
    - Required RNetwork port
  - Asynchronous Message Communication Receiver CPC
    - Required RSecurityService port
    - Provided PNetwork port

Fig. 3 Asynchronous Message Communication Sender and Receiver Communication Pattern Components and their Interfaces



a) Asynchronous Message Communication Sender and Receiver Communication Pattern Components



b) Interfaces of Asynchronous Message Communication Sender and Receiver Communication Pattern Components

Design of Security Coordinators

- Security coordinator
  - Need to be designed for each secure connector
    - Security Sender Coordinator
    - Security Receiver Coordinator
- Integrates communication pattern with selected security patterns
  - Template for high-level security coordinator designed for each communication pattern
  - Customized for each secure connector based on the security pattern(s) selected
- Security Sender Coordinator Interfaces for Secure AMC Connector (Fig. 4)
- Pseudocode template for Security Sender Coordinator in Secure AMC Connector (Fig. 5)

Fig. 4. Security Sender Coordinator Interfaces for Secure Asynchronous Connector

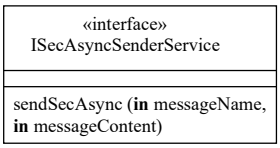


Fig. 5 Pseudocode template for Security Sender Coordinator in Secure Asynchronous Connector

```
loop

-- Wait for message from sender component;
receive (SenderComponentMessageQ, 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 to AMC Sender CPC;
AsynchronousMCSender.sendSecAsync (in MessageName, in
MessageContent);

end loop;
```

Structure of Secure Connector

- Security coordinator
  - Security Sender Coordinator
    - Receives messages from a sender component
  - Security Receiver Coordinator
    - Delivers messages to a receiver component
  - Sequences the interactions with
    - One or more security pattern components
    - Communication pattern component
- Security pattern component
  - Encapsulates specific security technique
  - Might interact with external security service for authentication, authorization or non-repudiation
- Communication pattern component
  - Encapsulates communication mechanism

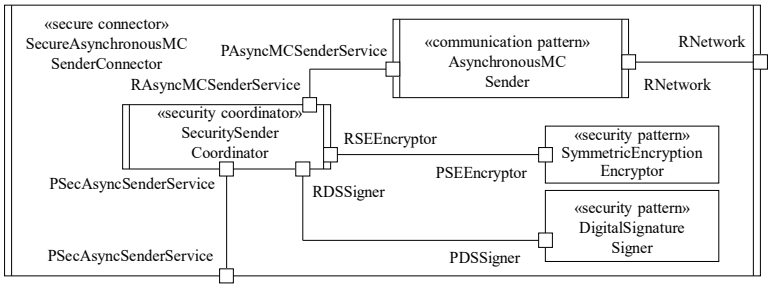
29

Design of Secure Asynchronous Message Communication (AMC) Connector

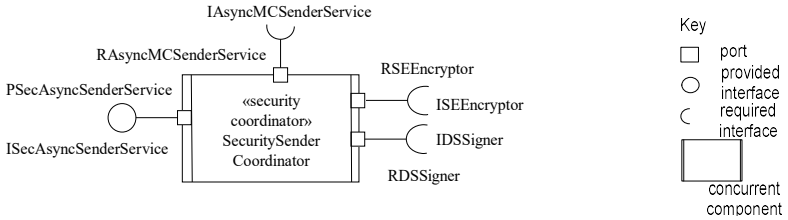
- Secure asynchronous MC connector
  - Contains Symmetric Encryption and Digital Signature security patterns
  - Asynchronous MC communication pattern
- Designed as a distributed composite component
- Secure AMC Sender Connector (Fig. 6a)
  - Security Sender Coordinator
  - Symmetric Encryption Encryptor SPC
  - Digital Signature Signer SPC
  - Asynchronous MC Sender CPC
- Secure AMC Receiver Connector
  - Security Receiver Coordinator
  - Symmetric Encryption Decryptor SPC
  - Digital Signature Verifier SPC
  - Asynchronous MC Receiver CPC
- Security Sender Coordinator component (Fig. 6b)
  - One Provided port, Three Required ports

30

Fig. 6. Security Sender Coordinator and Secure Asynchronous Message Communication Sender Connector



a) Secure Asynchronous Message Communication Sender Connector



b) Security Sender Coordinator and its Interface

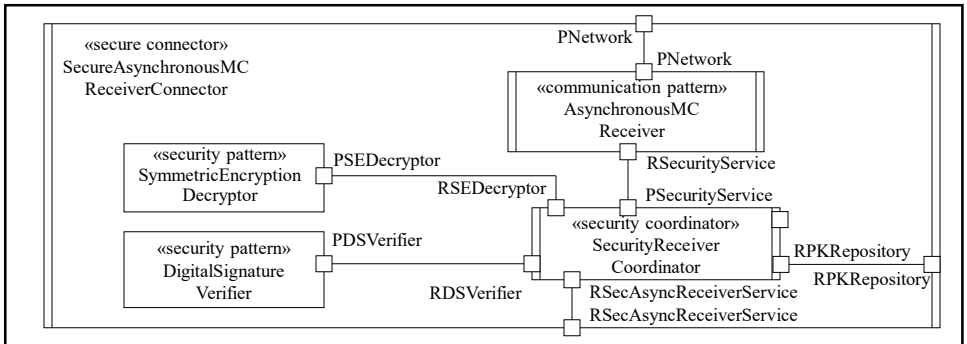
31

Fig. 7. Security Sender Coordinator for Secure AMC Communication Pattern and Symmetric Encryption Encryptor & Digital Signature Security Patterns

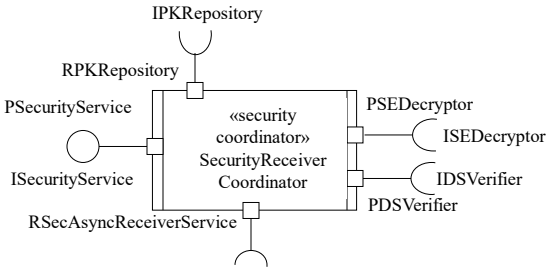
- The pseudocode for the Security Sender Coordinator component
  - Customized from the high-level pseudocode template (Fig. 5)

```
loop

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

-- Apply security patterns to message content;
if MessageContent requires non-repudiation
then
        DigitalSignatureSigner.sign (in MessageContent, in Key, out Signature);
end if;
if MessageContent requires confidentiality
then
        SymmetricEncryptionEncryptor.encrypt (in MessageContent&Signature,
        in Key, in Algorithm, out EncryptedMessageContent&Signature);
end if;

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

end loop;
```


32

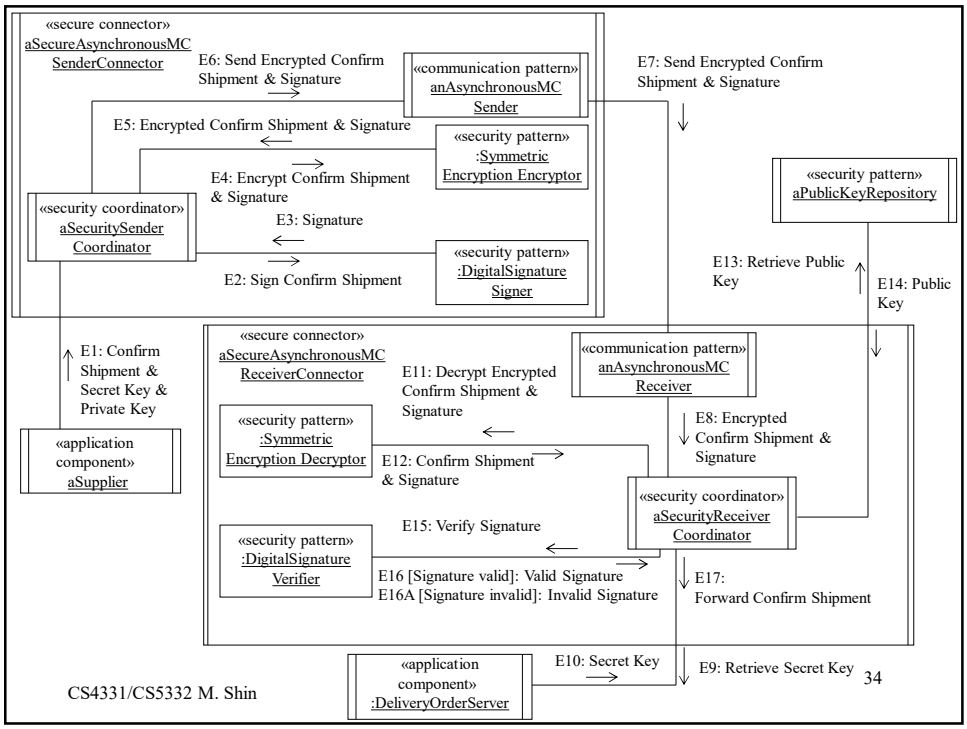


a) Secure Asynchronous Message Communication Receiver Connector



b) Security Receiver Coordinator

33



34