



MONASH University

Information Technology

FIT5183: Mobile and Distributed Computing Systems (MDCS)

Lecture 4A - WS Standards

WSDL, UDDI, BPEL, RM and Security

Outline

- ❑ WS* Technology Stack
- ❑ SOAP, WSDL and UDDI (Review only)
- ❑ WS-BPEL (WS Composition)
- ❑ WS-ReliableMessaging
- ❑ WS-Security

WS* Standards Overview

WS Standards and Specifications

Transport	HTTP, IIOP, SMTP, JMS		
Messaging	XML, SOAP		WS-Addressing
Description	XML Schema, WSDL		WS-Policy, SSDL
Discovery	UDDI		WS-MetadataExchange
Choreography	WSCL	WSCI	WS-Coordination
Business Processes	WS-BPEL	BPML	WSCDL
Stateful Resources	WS-Resource Framework		
Transactions	WS-CAF	WS-Transactions WS-Business Activities	
Reliable Messaging	WS-Reliability		WS-ReliableMessaging
Security	WS-Security SAML, XACML		WS-Trust, WS-Privacy WS-SecureConversation
Event Notification	WS-Notification		WS-Eventing
Management	WSDM		WS-Management
Data Access	OGSA-DAI		SDO

WS Standards and Specifications

Transport	HTTP, IIOP, SMTP, JMS		
Messaging	XML, SOAP		WS-Addressing
Description	XML Schema, WSDL		WS-Policy, SSDL
Discovery	UDDI		WS-MetadataExchange
Choreography	WSCL	WSCI	WS-Coordination
Business Processes	WS-BPEL	BPML	WSCDL
Stateful Resources	WS-Resource Framework		
Transactions	WS-CAF	WS-Transactions WS-Business Activities	
Reliable Messaging	WS-Reliability		WS-ReliableMessaging
Security	WS-Security SAML, XACML		WS-Trust, WS-Privacy WS-SecureConversation
Event Notification	WS-Notification		WS-Eventing
Management	WSDM		WS-Management
Data Access	OGSA-DAI		SDO

Simple Object Access Protocol (SOAP)

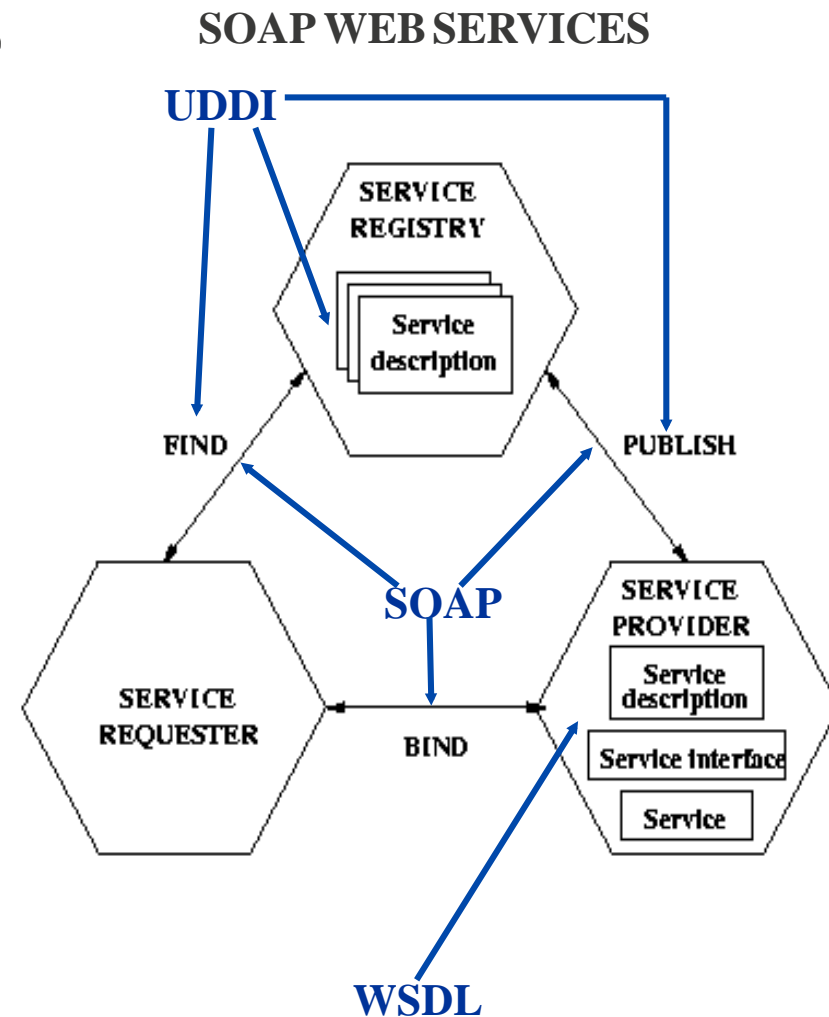
IBM's *Web service architecture*

❑ SOAP Web Service

Definition by W3C:

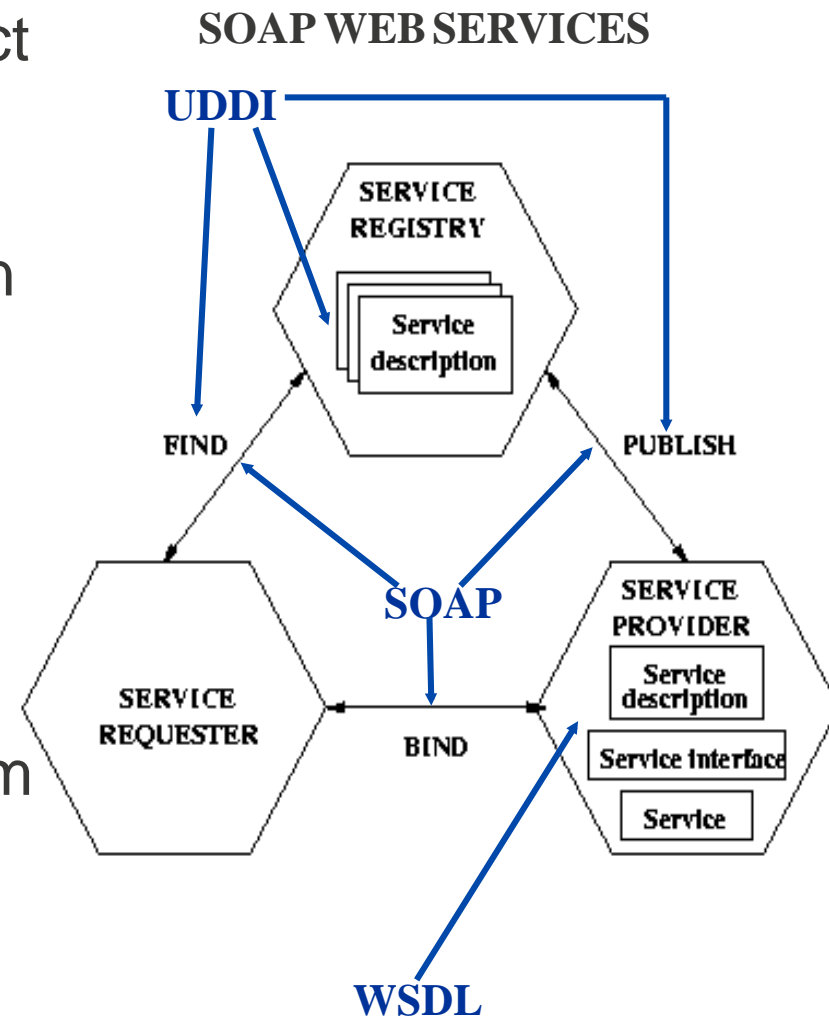
“a software application **identified by a URI**, whose **interfaces and bindings** are capable of being defined, described, and discovered as **XML artifacts**.”

❑ SOAP = XML + HTTP (in practice, at least)



Web Services Description Language (WSDL)

- ❑ Once it is possible to interact with any Service using the SOAP protocol we still need to **Describe the Services**, in particular their interfaces.
- ❑ **WSDL** is used to automatically generate the **client stubs** and **server skeleton** code, shielding details of the distribution from the developer.



WSDL Document Elements (Review)

ABSTRACT PART:

<types> - The data types used by the web service

<message> - The messages used by the web service

<portType> - The operations performed by the web service

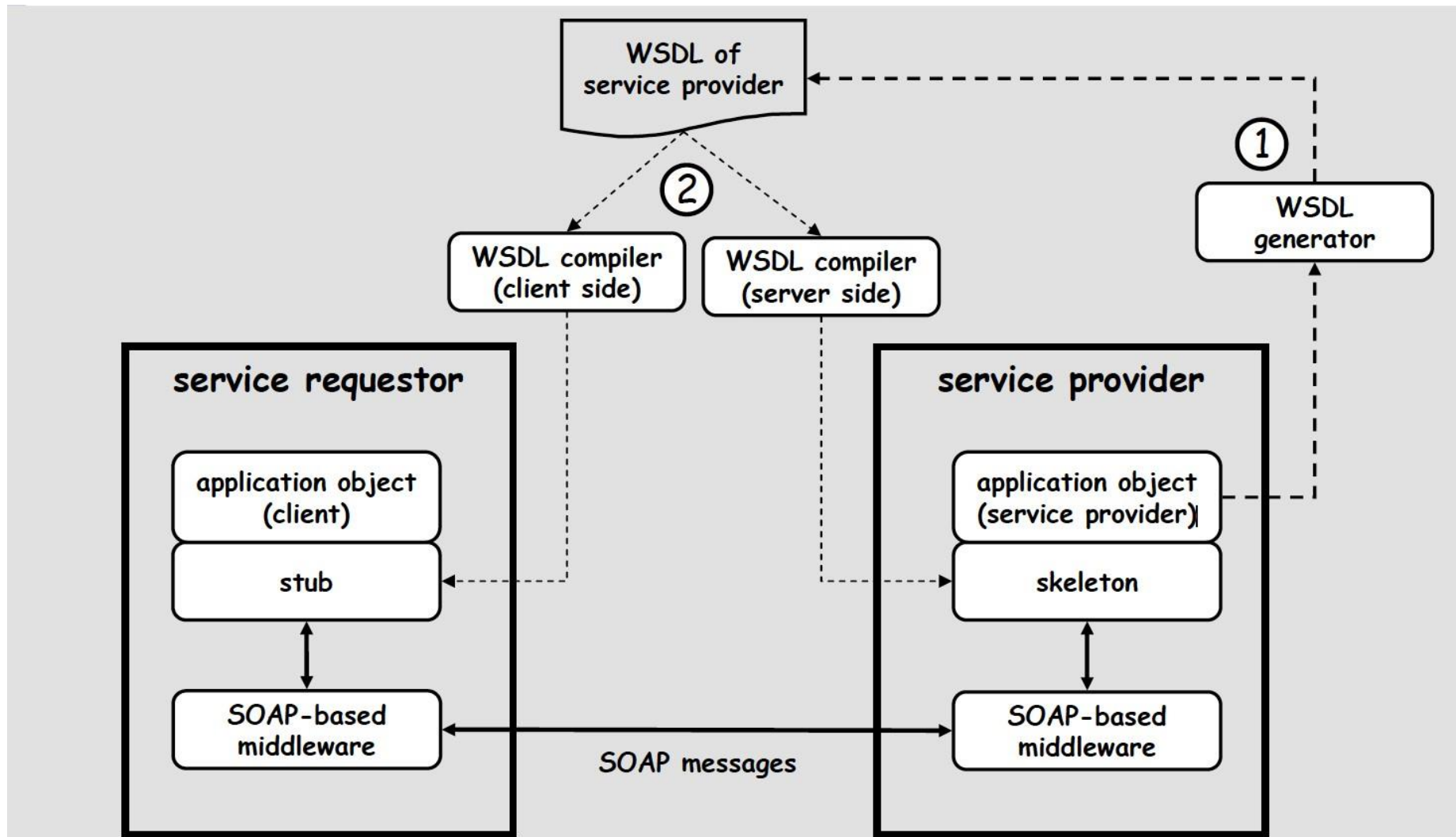
CONCRETE PART:

<binding> - The communication protocols used by the web service

<service> - Which binding to use and Location of the service

```
<definitions>
<types>
    definition of types...
</types>
<message>
    definition of a message...
</message>
<portType>
    definition of a port...
</portType>
<binding>
    definition of a binding...
</binding>
<service>
    definition of the
    service....
</service>
</definitions>
```

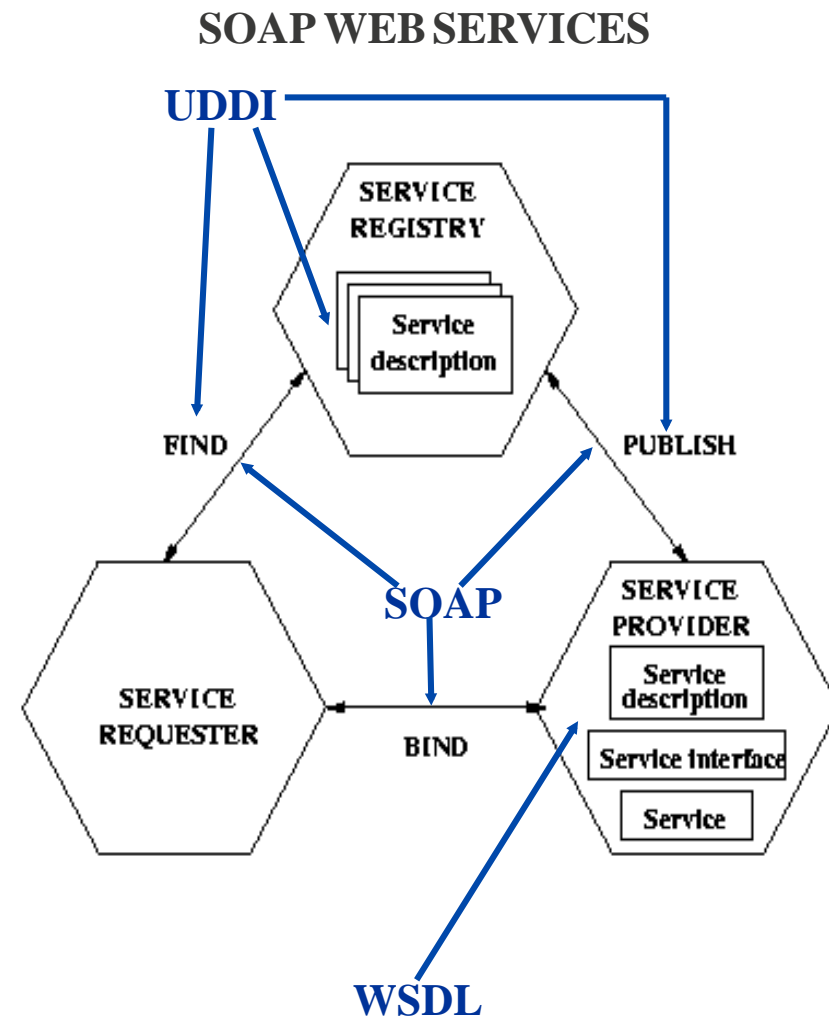

WSDL in Action



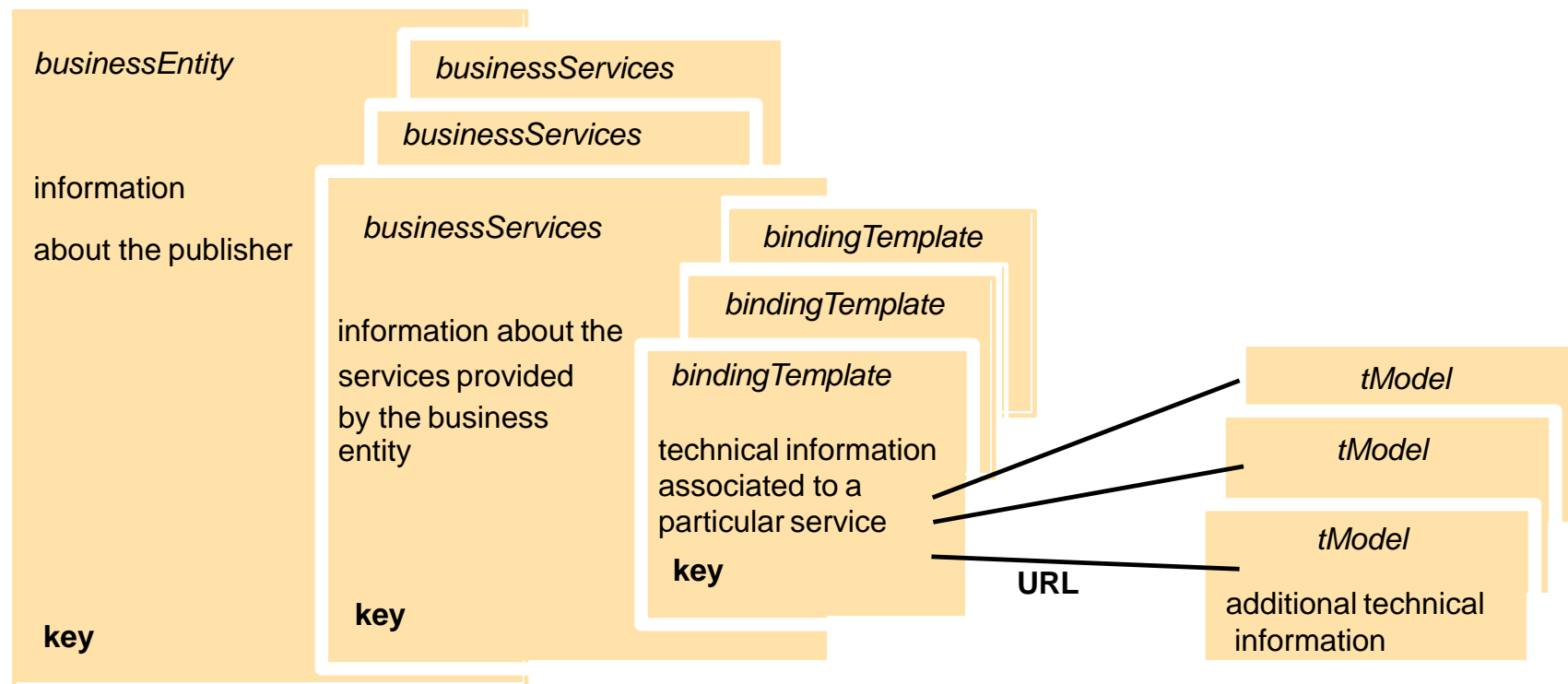
Copyright Springer Verlag Berlin Heidelberg 2004

Universal Description, Discovery and Integration (UDDI)

- ❑ **Service requester:** The potential user of a service
- ❑ **Service provider:** The entity that implements the service and offers to carry it out on behalf of the requester
- ❑ **Service registry:** A place where available services are listed and which allows providers to advertise their services and requesters to query for services




The Main UDDI Data Structures



Adapted from (Instructor's Guide for Coulouris, Dollimore and Kindberg Distributed Systems: Concepts and Design Edn. 4)

WS Standards and Specifications

Transport	HTTP, IIOP, SMTP, JMS		
Messaging	XML, SOAP	WS-Addressing	
Description	XML Schema, WSDL	WS-Policy, SSDL	
Discovery	UDDI	WS-MetadataExchange	
Choreography	WSCL	WSCI	WS-Coordination
Business Processes	WS-BPEL	BPML	WSCDL
Stateful Resources	WS-Resource Framework		
Transactions	WS-CAF	WS-Transactions WS-Business Activities	
Reliable Messaging	WS-Reliability		WS-ReliableMessaging
Security	WS-Security SAML, XACML		WS-Trust, WS-Privacy WS-SecureConversation
Event Notification	WS-Notification		WS-Eventing
Management	WSDM		WS-Management
Data Access	OGSA-DAI		SDO



Choreography, in a [Web services](#) context, refers to specifications for how messages should flow among diverse, interconnected components and applications to ensure optimum interoperability.

Orchestration vs. Choreography

❑ Orchestration

- Used in private business processes
- A central process (may be a WS) takes control of the involved Web services and coordinates the execution of different operations involved.
- Web services do not "know" (and do not need to know) they are involved in a composition process and are taking part in a higher-level business process.
- Only the central coordinator is aware of this goal

❑ Choreography

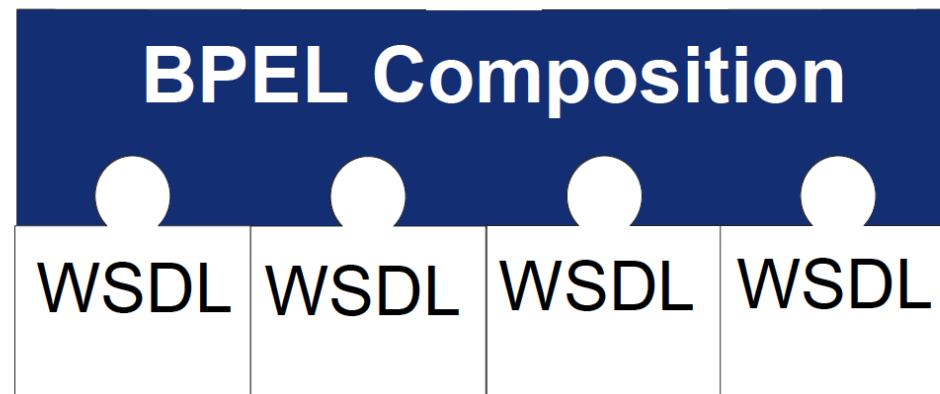
- Does not rely on a central coordinator
- Each Web service involved in the choreography knows exactly when to execute its operations and with whom to interact.
- Is a collaborative effort focusing on the exchange of messages in public business processes.
- All participants in the choreography need to be aware of the business process, operations to execute, messages to exchange, and the timing of message exchanges

Composition in the Context of Web Services

Composing Executable Processes

- ❑ Executable Processes describe the composition (or orchestration) of different Web service interfaces (WSDL Port Types)
- ❑ The result of a composition using **BPEL** (Business Process Execution Language) is meant to be recursively published as a Web service.

WSDL



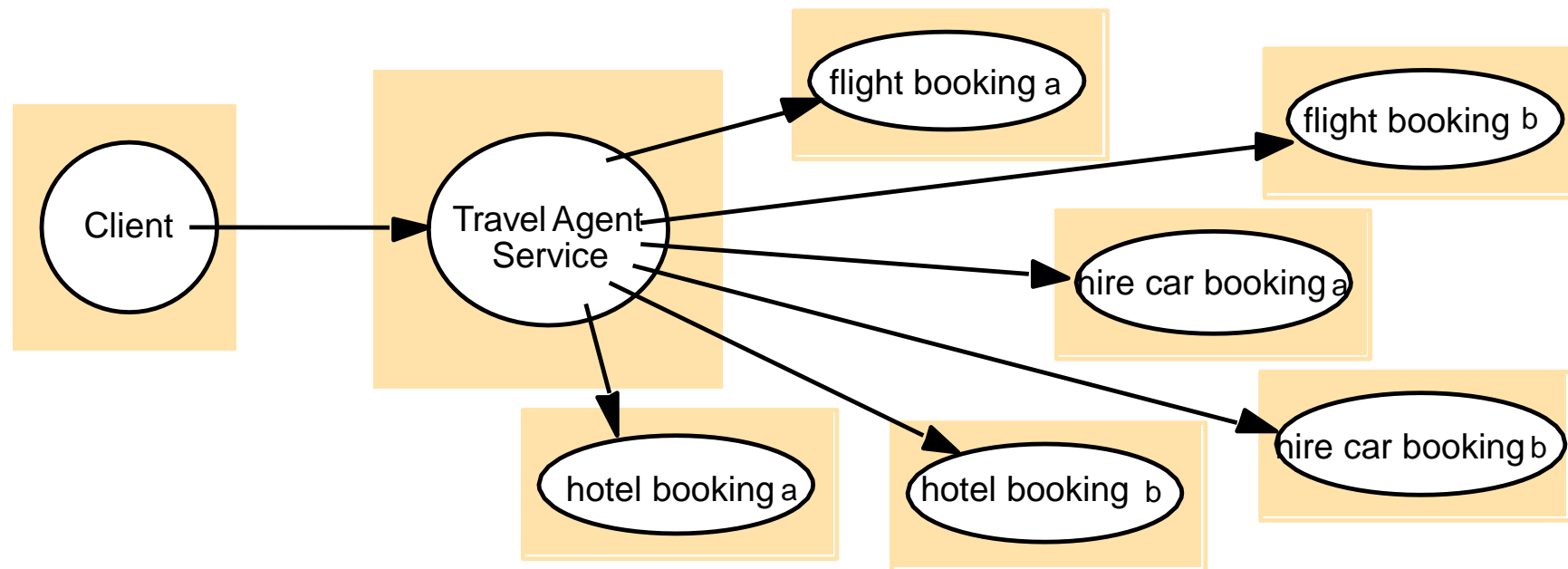
Web Service Interfaces

Value Added Services

- Once basic services begin to be published on the Web, the natural thing to do is to find and combine them into more complex, value added services.
- Assuming that the necessary basic services are available, it is possible to define the business logic of an application out of their composition. Different applications can be built by composing the same services in different ways.
- Note: composite services are still services, i.e., services that can be composed: **composition is recursive**
- Web service composition middleware provides abstractions for high level modeling of compositions and the infrastructure for executing them efficiently
- Examples:
 - Look for the cheapest price for a book/movie/song on all Internet e-stores and buy it, ensuring that it will be shipped before your birthday
 - Search from different Web search engines, merge all results by removing duplicates
 - Plan for a vacation: order plane tickets, reserve hotels and cars for a set of destinations
 - Convert stock quote prices to a different currency
 - Get weather information and snow & avalanches reports for the nearest ski resorts
 - Gather bids from suppliers, select the winner after a deadline and notify all participants of the outcome

©Gustavo Alonso, D-INFK. ETH Zürich.

The 'travel agent service' combines other web services



Instructor's Guide for Coulouris, Dollimore and Kindberg
Distributed Systems: Concepts and Design Edn. 4
© Addison-Wesley Publishers 2005

Web Service Interactions

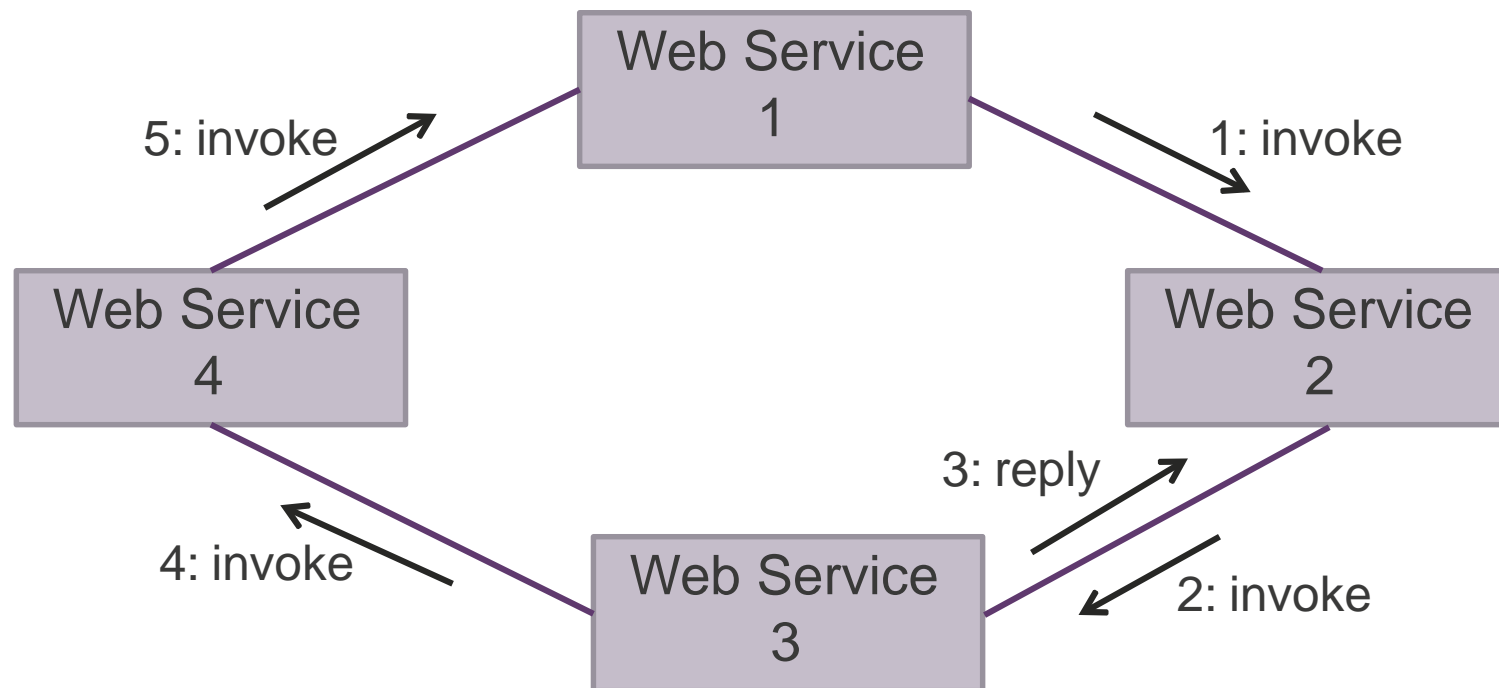
Web services usually expose operations of certain applications or information systems. Consequently, combining several Web services actually involves the integration of the underlying applications and their functionalities.

- ❑ **Compositions** - implementation of a Web service whose internal logic involves the invocation of operations offered by other services
- ❑ **Conversations** - dialog among two or more Web services participating in an interaction
- ❑ **Coordinations** - protocol describes a set of accepted conversations (the external observable behavior of involved Web services)

Composition of Web services with Orchestration



Composition of Web services with Choreography



Web Services Business Process Execution Language (WS-BPEL)

- ❑ WS-BPEL is a standard proposal for specifying business process behavior based exclusively on Web services
- ❑ WS-BPEL is a language based on the XML syntax.
- ❑ It does not directly deal with implementation of the language but only with the semantics of the primitives it provides.
- ❑ The goal is to define coordination and composition of Web services in a portable way
- ❑ The language is used to define:
 - **Executable processes**, with the actual interactions between different services. These process can implement the internal business logic of a Web service (Composition)
 - **Abstract processes**, modeling the messages exchanged by the parties involved in a business protocol without revealing details about the internal implementation (Coordination)

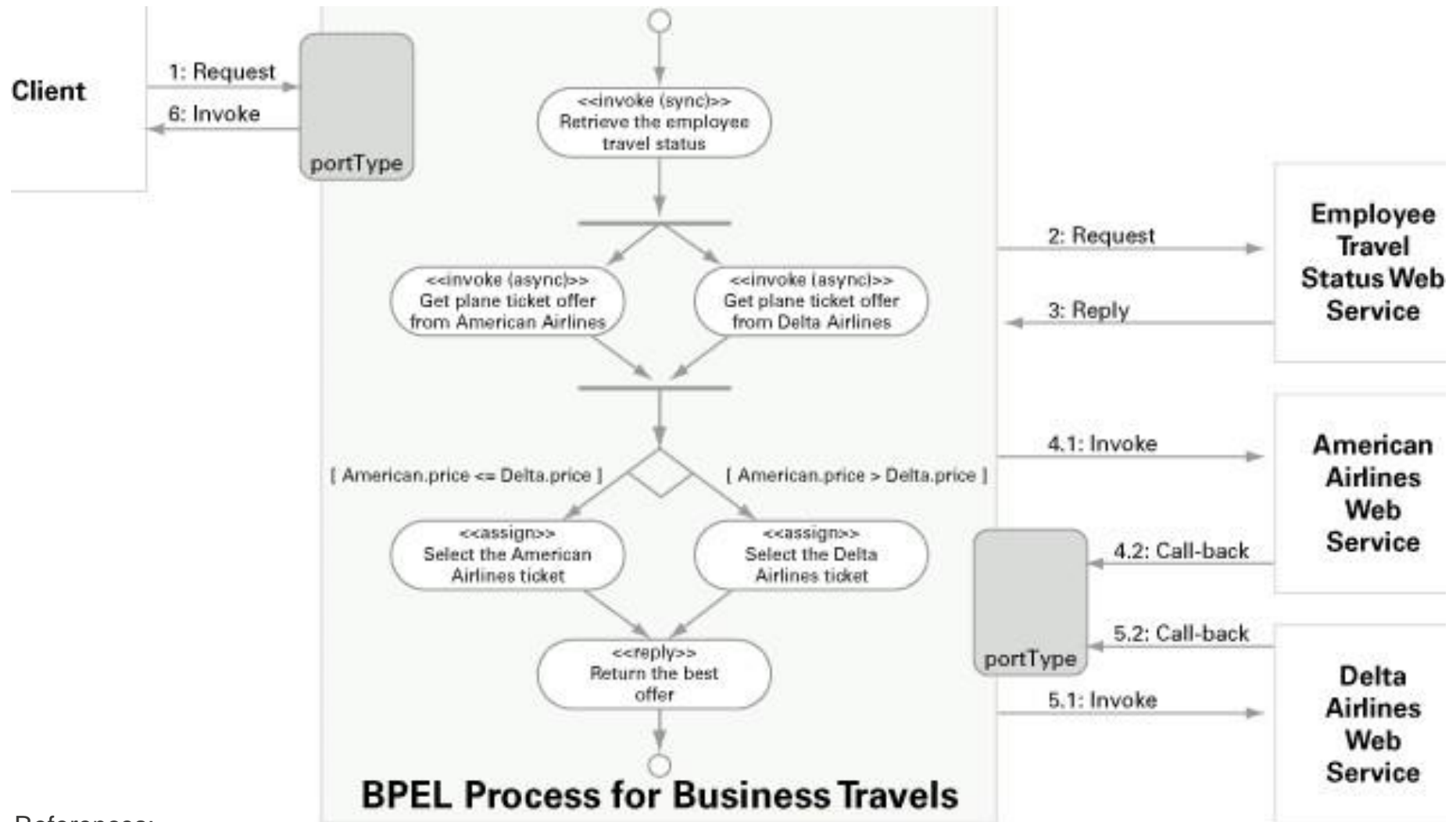
WS-BPEL

- ❑ BPEL supports two different ways of describing business processes that support **orchestration** and **choreography**:
- ❑ Executable processes
 - specify exact details of business processes
 - follow the **orchestration paradigm**
 - can be executed by an orchestration engine
- ❑ Abstract business protocols
 - specify public message exchange between parties only
 - Do not include the internal details of process flows
 - are not executable

Building a WS-BPEL Business Process

- ❑ A BPEL process specifies the exact order in which participating Web services should be invoked
 - Sequentially or in parallel
- ❑ Can express conditional behaviors
 - For example, an invocation of a Web service can depend on the value of a previous invocation.
- ❑ Can also construct loops, declare variables, copy and assign values, define fault handlers, and so on.
- ❑ By combining all these constructs, you can define complex business processes in an algorithmic manner.

Schematic of Working Scenario



References:

<http://www.oracle.com/technetwork/articles/matjaz-bpel1-090575.html>

Reliable Messaging

WS-ReliableMessaging Overview

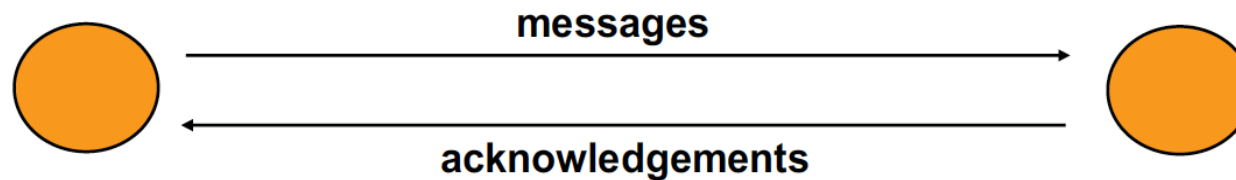
WS-ReliableMessaging defines a set of SOAP Header extension elements that enable:

- ☐ Guaranteed Message Delivery
- ☐ Message Ordering Preservation
- ☐ Duplicate Detection

Basic Model:

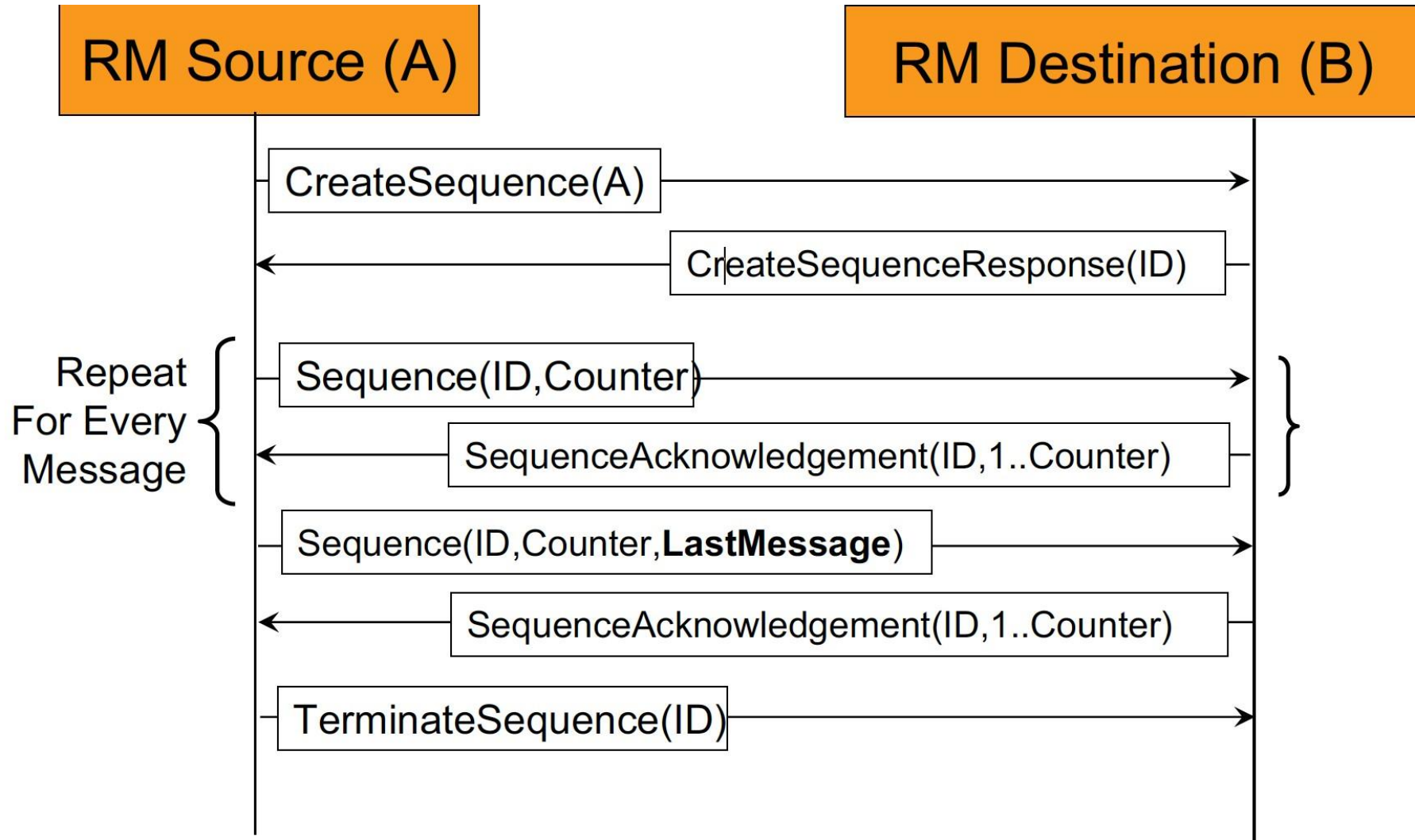
- ☐ Based on a sender and a receiver
- ☐ Communication happens one way (from the sender to the receiver)
- ☐ Reliable two way communication requires to duplicate the set up in the other direction

Basic Model



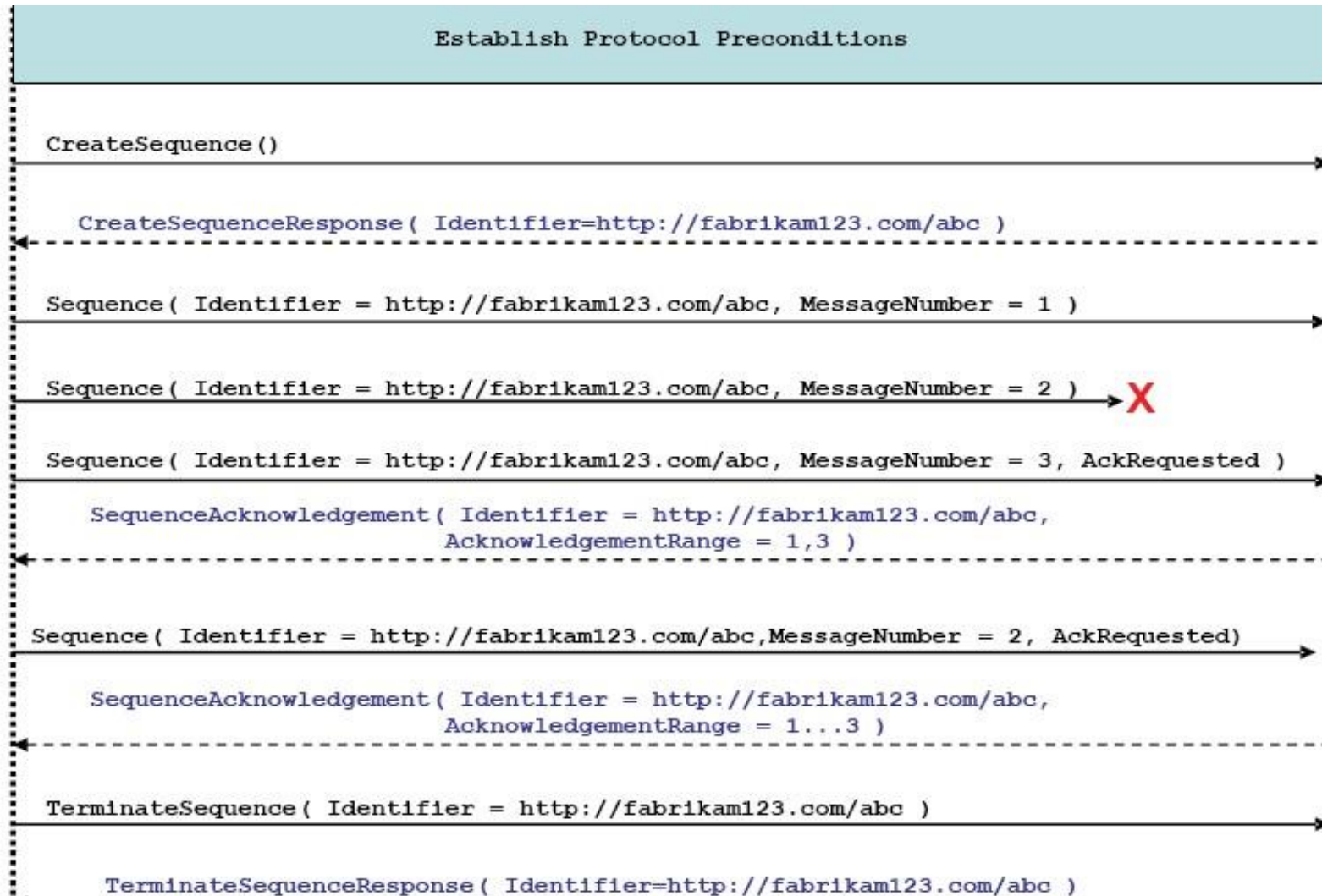
RM Source (RMS)	RM Destination (RMD)
<ul style="list-style-type: none">• Request creation of a sequence of reliable messages (contract)• Requests termination of a sequence of reliable messages• Add proper headers to Messages• Requests acknowledgements• Resend messages if need be	<ul style="list-style-type: none">• Responds to requests to create a sequence of reliable messages• Responds to requests to terminate a sequence of reliable messages• Accepts and acknowledges messages (synchronous or asynchronously)• May drop duplicate messages• May queue messages to deliver in order

Message Sequence Lifecycle



©Gustavo Alonso, D-INFK. ETH Zürich.

Example



©Gustavo Alonso, D-INFK. ETH Zürich.

WS-Security

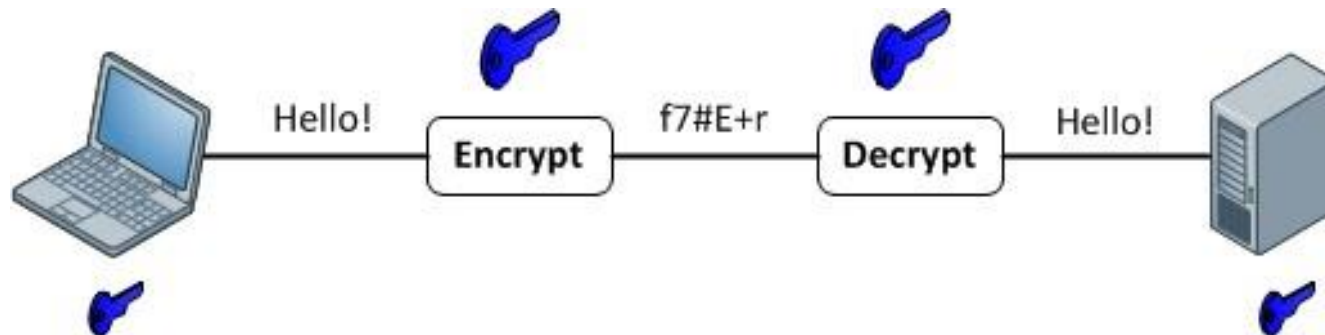
WS-Security Overview

- ❑ Use both XML Encryption and XML Signature to implement secure SOAP message exchange across multiple and independent trust domains
- ❑ Goals: security at the message level (end-to-end)
- ❑ Solution: apply encryption and signatures within a SOAP message independent of the transport. Parts of the message body can be encrypted, signatures are stored in the header.
- ❑ WS-Security features support for:
 - Multiple signature technologies
 - Multiple encryption technologies
 - Multiple security token formats
- ❑ OASIS standard, April 2004

Symmetric and Asymmetric Encryption

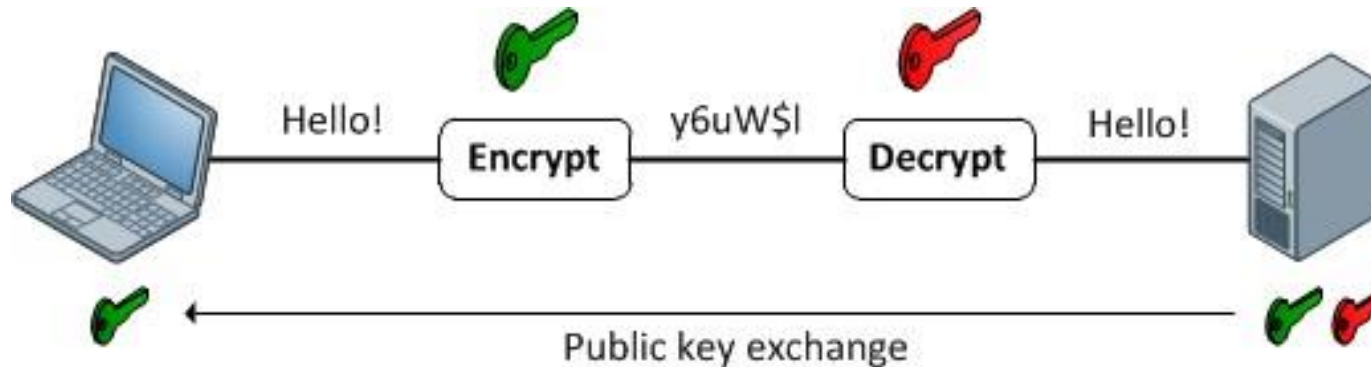
❑ Symmetric encryption

- aka shared key or shared secret encryption
- A single key used both to encrypt and decrypt data



Symmetric and Asymmetric Encryption

- ❑ Asymmetric encryption
 - aka public-key cryptography
 - two keys used (public and private keys)



Two different uses:

- 1 sender doesn't want anyone else to read the data except for the receiver (public key used for encryption and private key for decryption)
- 2 it also can be used when the sender wants the receiver to know the message is from it. In such cases, the private key is used for encryption and public key for decryption (e.g. in xml signature).

Hashing

- ❑ Hash algorithms (aka digest method) condenses a message into a fixed-length value
- ❑ The results called a *Hash* (or *Message Digest*)
- ❑ E.g. SHA-1 (Secure Hash Algorithm 1)

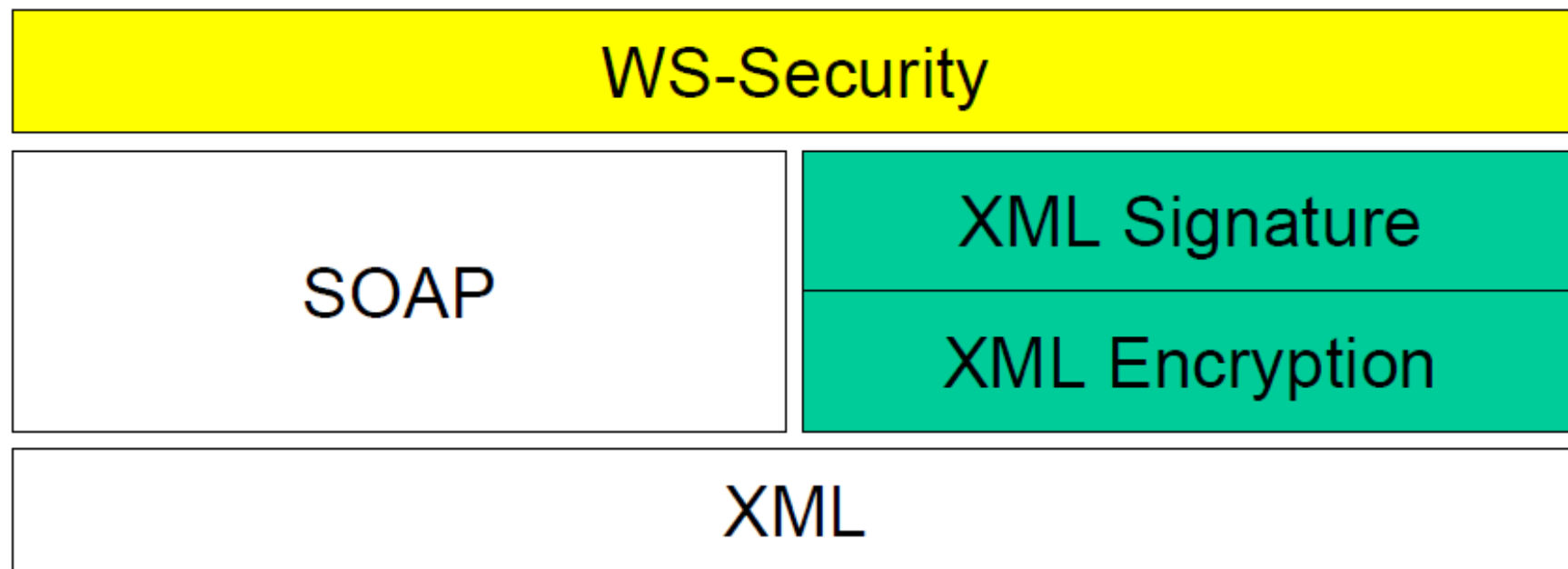


❑ XML Signature

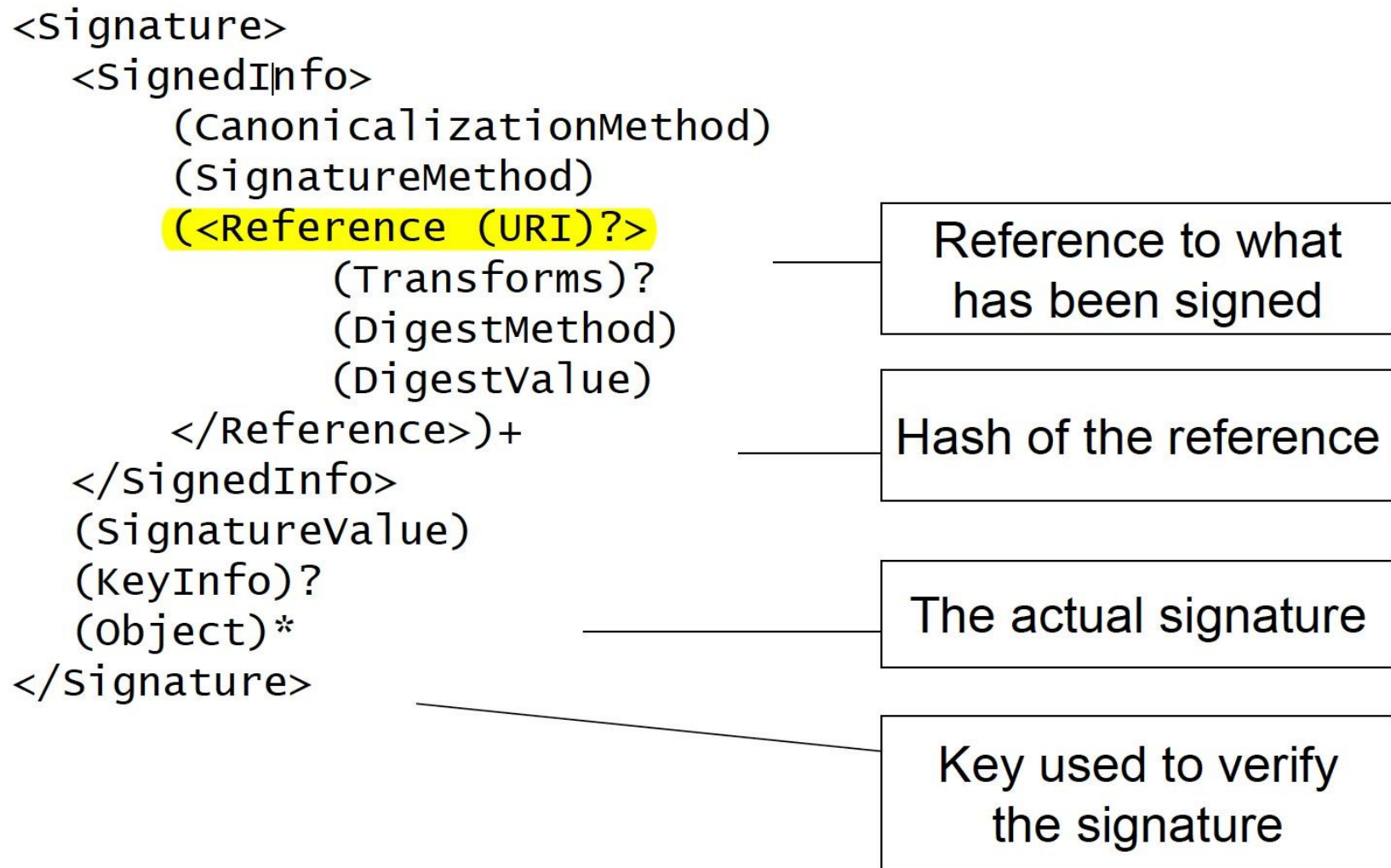
- Integrity of XML messages
- Identity of the source of the document
- Ensure non-repudiation

❑ XML Encryption

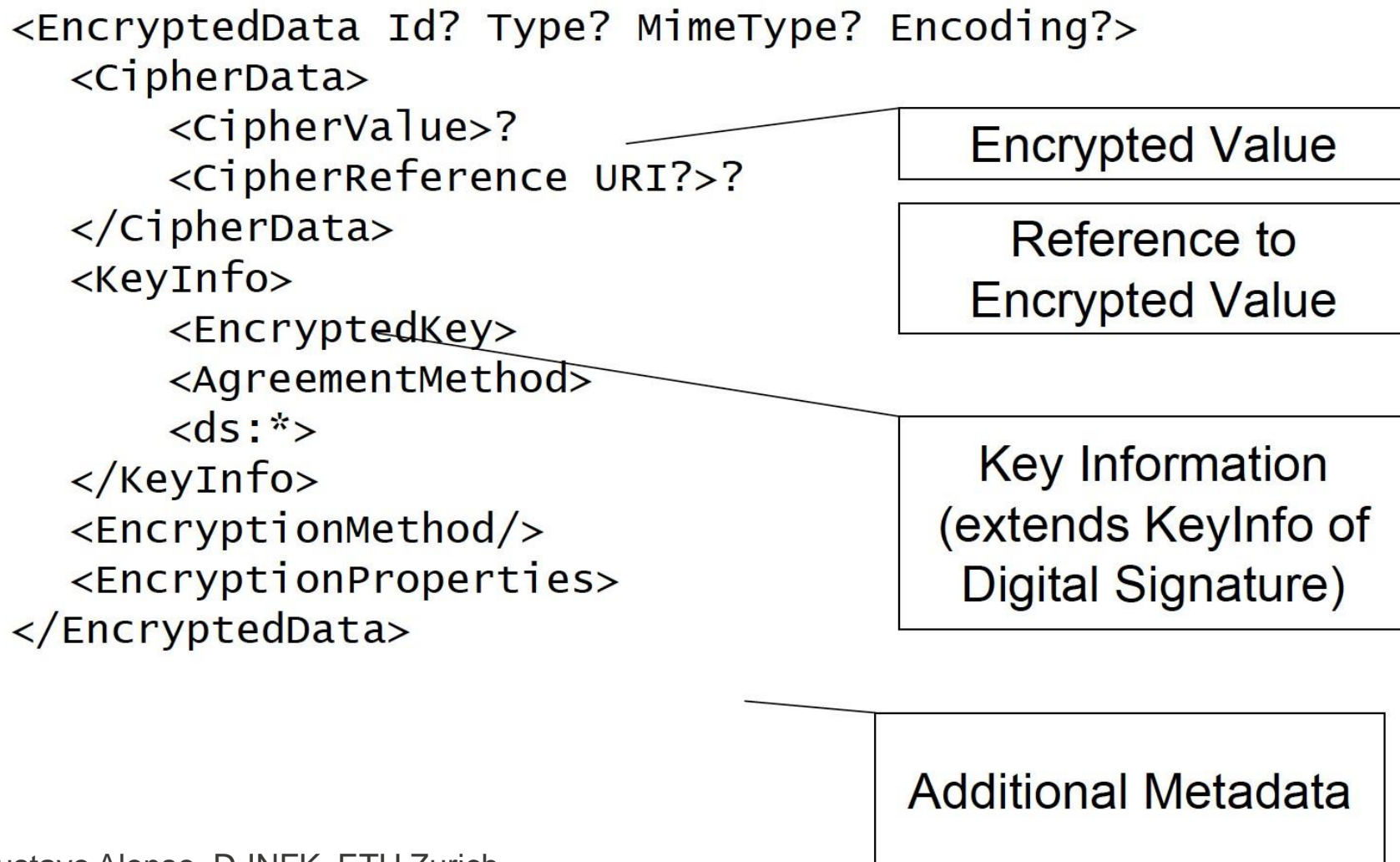
- Ensure confidentiality of XML messages



XML Signature Structure



XML Encryption Structure



XML Encryption Example

```
<Employee>  
  <ID>222-654-456</ID>  
  <Name>Markus Bach</Name>  
  <Salary currency="CHF">100000</Salary>  
</Employee>
```

Original XML Document

```
<Employee>  
  <ID><EncryptedData>...</EncryptedData></ID>  
  <Name>Markus Bach</Name>  
  <EncryptedData>...</EncryptedData>  
</Employee>
```

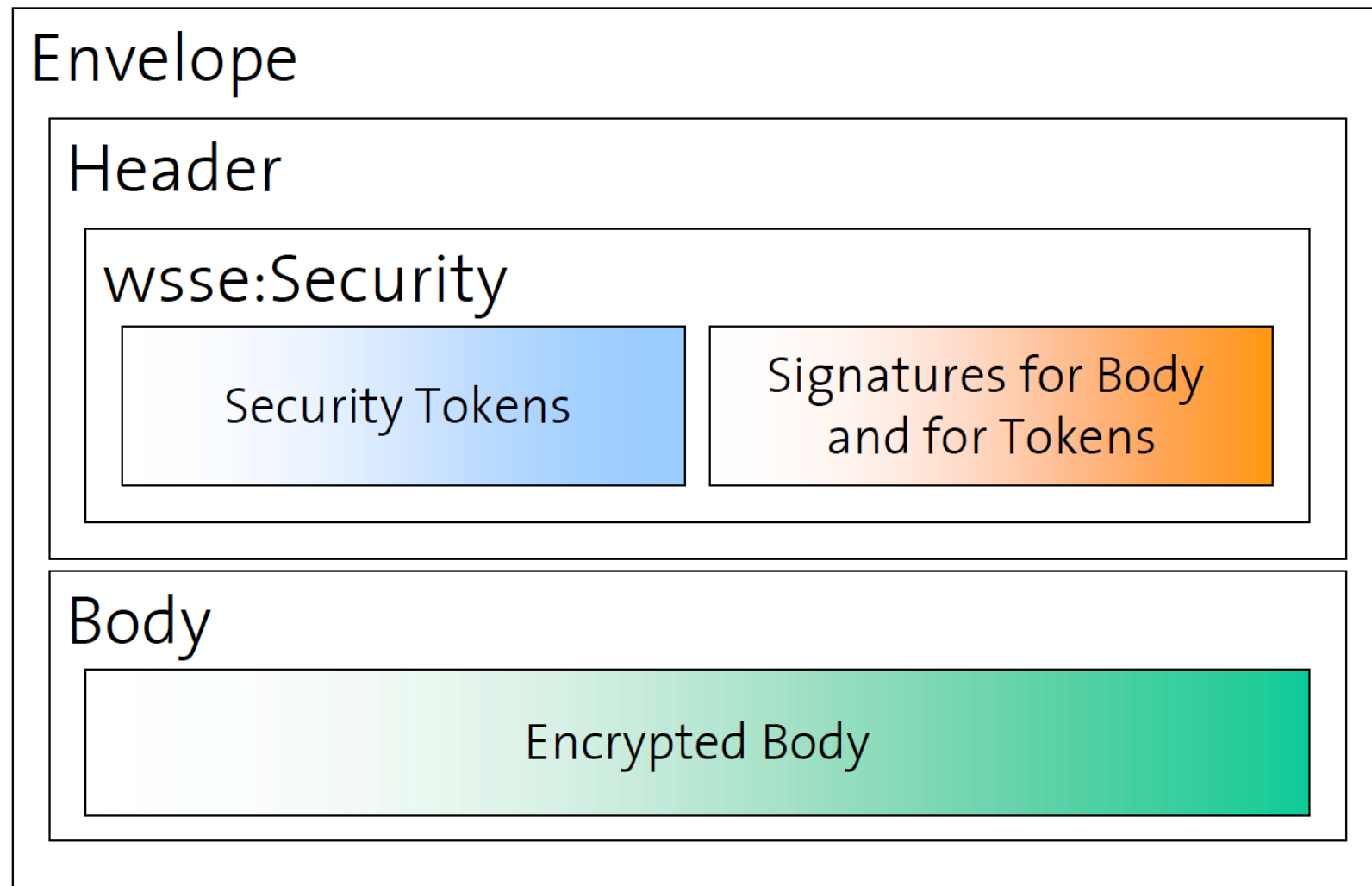
Encrypted XML Document



Protecting SOAP Messages

- ❑ Security Threats to a SOAP message:
 - A message could be read by an attacker
 - A message could be modified by an attacker
 - A message could be sent by an attacker
- ❑ To address these threats, WS-Security applies a combination of:
 1. Encryption (Ensure the confidentiality of the message)
 2. Signatures (Verify the origin and the integrity of a message)
 3. Security Tokens (Authorize the processing of the message based on the credentials associated with the message)
- ❑ Messages with invalid signatures and incorrect or missing tokens are rejected.

A Secure SOAP Message



References

- ❑ Lecture Notes from Gustavo Alonso, ETH
- ❑ Lecture Notes from Aad Van Moorsel
- ❑ <http://www.w3schools.com/wsdl/default.asp>
- ❑ WSDL 2.0
<http://webservices.xml.com/lpt/a/ws/2004/05/19/wsdl2.html>
- ❑ Understanding WSDL
- ❑ <https://msdn.microsoft.com/en-us/library/ms996486.aspx>
- ❑ https://www.w3.org/TR/#tr_XML_Signature
- ❑ https://www.w3.org/TR/#tr_XML_Encryption