M14 L1
Encryption and Decryption Algorithms

Ira A. Fulton Schools of Engineering
Arizona State University

Lecture Outline

- **Encryption / Decryption Algorithms**
  - DES: Data Encryption Standard
  - Pretty Good Privacy (PGP)
- **Ethics**
- **Encryption / Decryption Algorithms in .Net**
  - Library Functions
  - Service Implementation

Need for Encryption and Decryption

- We use SSL to address the problem during data transmission
- In the situations where we need to store data, we need to encrypt and decrypt data of our own.
  For example,
  - Date stored: Confidential data to be stored in the Web.config file, XML file, text file, or database;
  - Data transmitted over the internet without SSL: We cannot use remote service in this case: Data sent to the remote service is not secure;
  - Hashing: One way encryption, e.g., used for session ID generation and password saving
- Saving sensitive data in clear text is not acceptable for an organization that cares about protecting its users, and it may be liable for damage caused.

Encryption and Decryption Algorithms

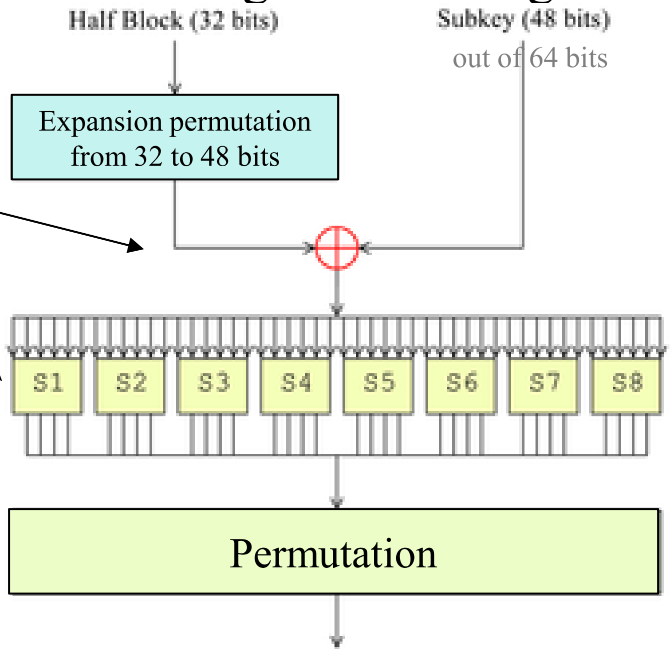
- There are many Encryption and Decryption Algorithms using the secret and public key systems: e.g., DES, SHA-1, MD-5, RSA, PGP
- Data Encryption Standard (DES), developed by IBM in 1970, was selected by the National Bureau of Standards as an official Federal Information Processing Standard (FIPS) for the U.S. in 1976, used for unclassified data.
  - It is based on a 56-bit key encryption. (Note: MD-5 uses 128 bits.)
  - As the key length 56 is short in today's standard, and DES is used for low-level security purpose only.
- RSA (by Rivest, Shamir and Adleman) is an open key algorithm developed in 1973, publicly known in 1977, and declassified in 1997.
  - It is based on the product of two prime numbers and uses a factor as the public and private keys. It relies on very large numbers and their factors.
  - A 768-bit key was broken in two-year computation. NIST recommends 2048-bit key. Many organization uses 4094-bit key.

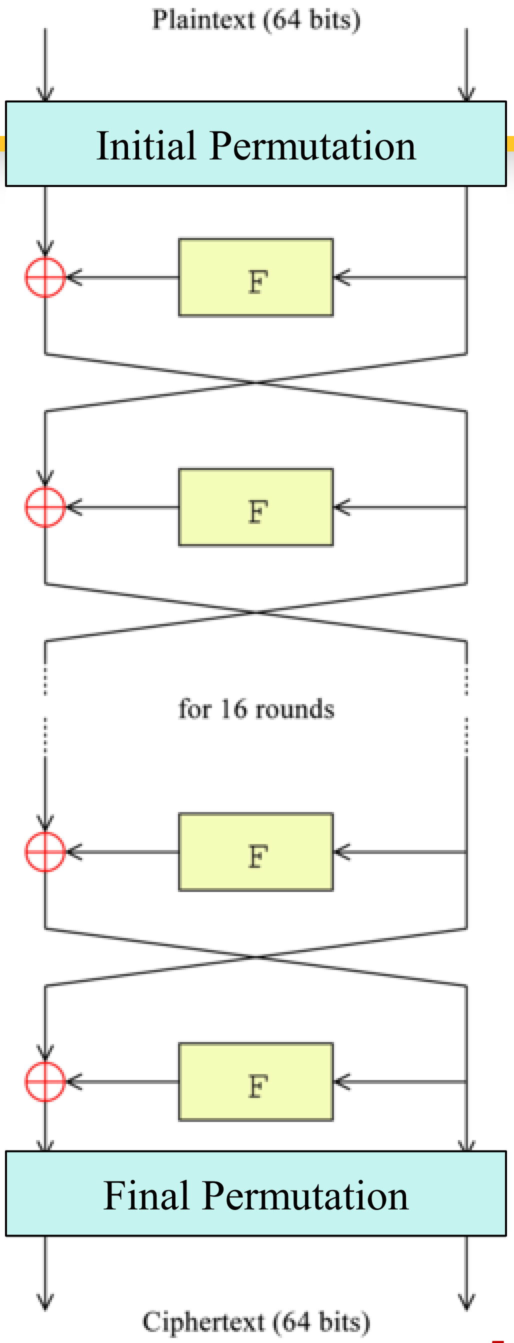
DES: Data Encryption Standard

http://en.wikipedia.org/wiki/Data_Encryption_Standard

- Uses a secret key of 8 bytes, with each byte containing a check bit, and thus 56 function bits + 8 check bits = 64
- Uses block cypher encryption
- Uses multiple rounds of encryptions
- It is safe if the secret key does not need to be transferred to the other sites.
- You can use local encryption/decryption functions to encrypt/decrypt data before saving/after reading

1. Take a half block
2. XOR with subkey
3. Cut into 6x8 groups
4. Swap the order of groups
5. Take the other half
6. Go back to step 2 and repeat the process 16 rounds

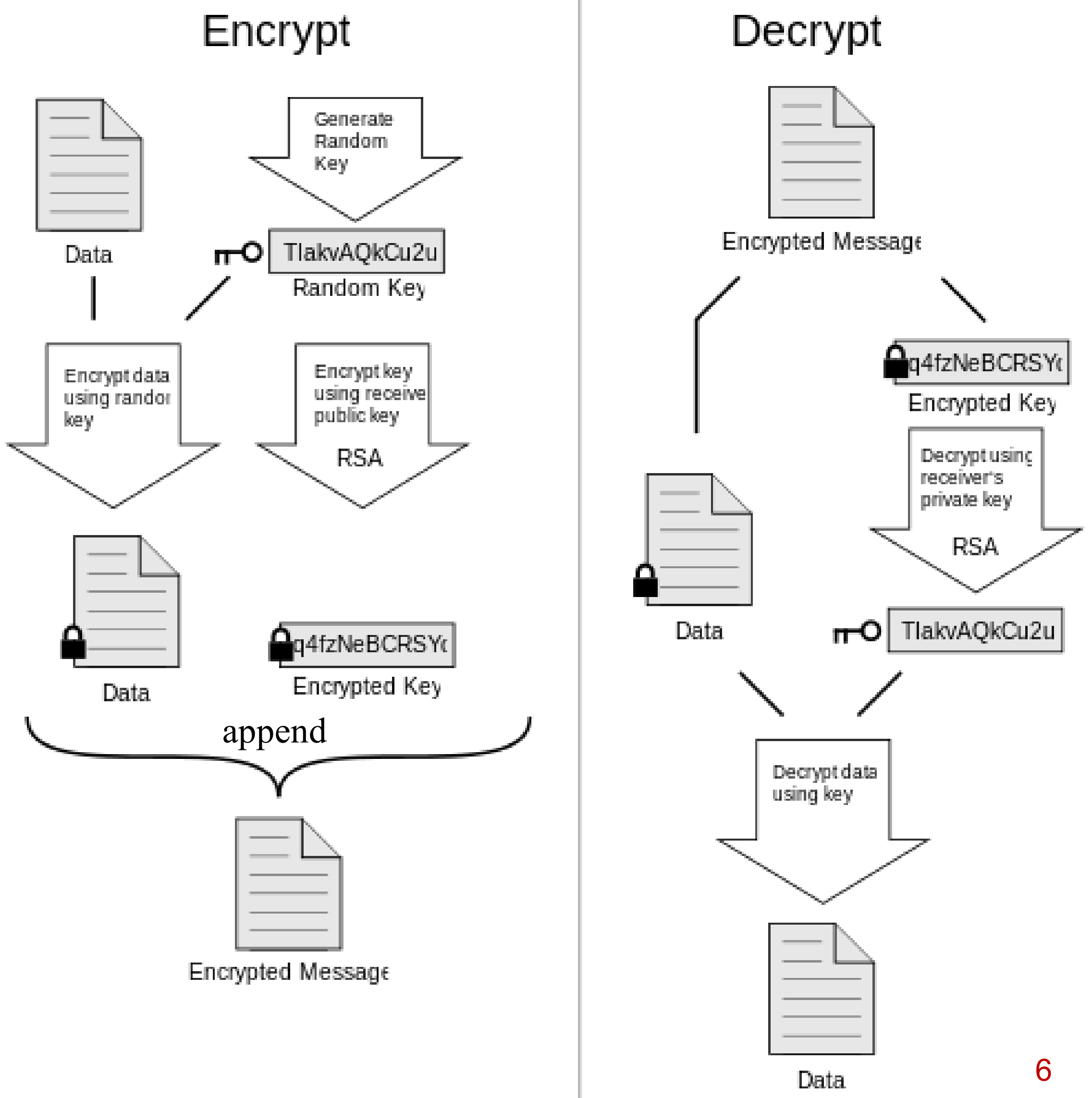




Pretty Good Privacy (PGP) Algorithm

http://en.wikipedia.org/wiki/Pretty_Good_Privacy

- PGP is an open key encryption/decryption algorithm developed by Phil Zimmermann in 1991 and published in 1993 in a book.
- It uses multi-steps of hashing, data compression, symmetric-key cryptography, and finally public-key cryptography.

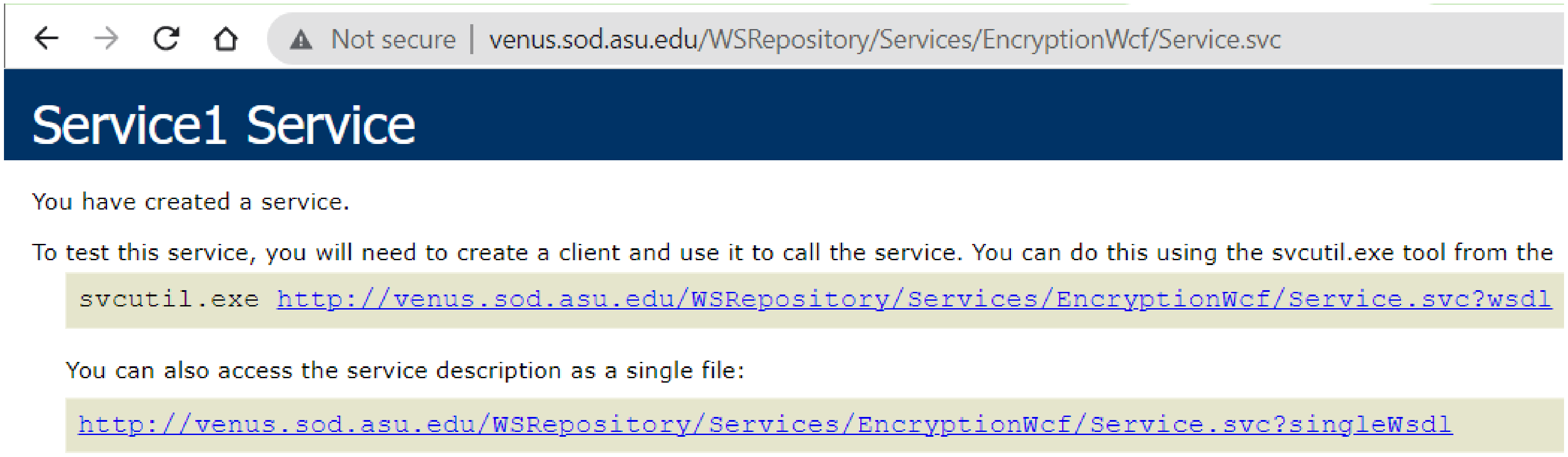


Legal and Ethics Cases in Data Privacy and Encryption and Decryption Algorithms

- Many encryption and decryption algorithms are of matters of national security and are classified by the U.S. government.
- Encryption technologies have been used for protecting secret military and government communications.
- Encryption use is entirely legal inside the U.S. borders.
- Encryption algorithms are subject to strict export control to other countries, because they can be used in military applications.
- Phil Zimmermann published the entire code of PGP in 1993.
- He became the formal target of a criminal investigation by the US Government for "munitions export without a license", and the trial lasted for three years.

Encryption and Decryption Algorithms in .Net

- .Net Framework Class Library implemented a namespace System.Security.Cryptography
- It includes a number of security-related classes [http://msdn.microsoft.com/en-us/library/system.security.cryptography.aspx]
- The example presented here uses the DES SymmetricAlgorithm class to create a WCF Web service with two operations:
  - string Encrypt(string)
  - string Decrypt(string)


Not secure | venus.sod.asu.edu/WSRepository/Services/EncryptionWcf/Service.svc

Service1 Service

You have created a service.

To test this service, you will need to create a client and use it to call the service. You can do this using the svcutil.exe tool from the

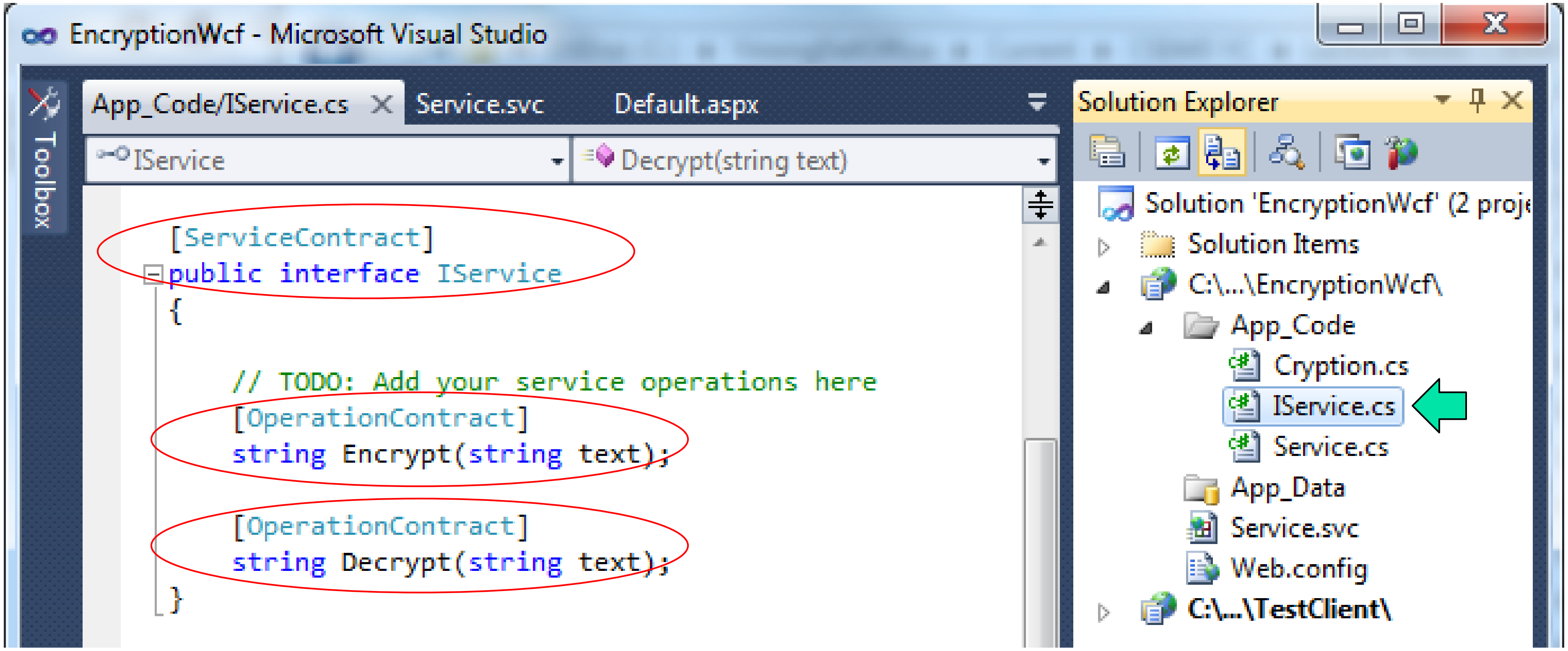
```
svcutil.exe http://venus.sod.asu.edu/WSRepository/Services/EncryptionWcf/Service.svc?wsdl
```

You can also access the service description as a single file:

```
http://venus.sod.asu.edu/WSRepository/Services/EncryptionWcf/Service.svc?singleWsdl
```

Interface Definition: Contract

File → New Web Site → WCF Service
ServiceContract and OperationContract Template are created
Enter the “Encrypt” and “Decrpyt” interface as the [OperationContract]

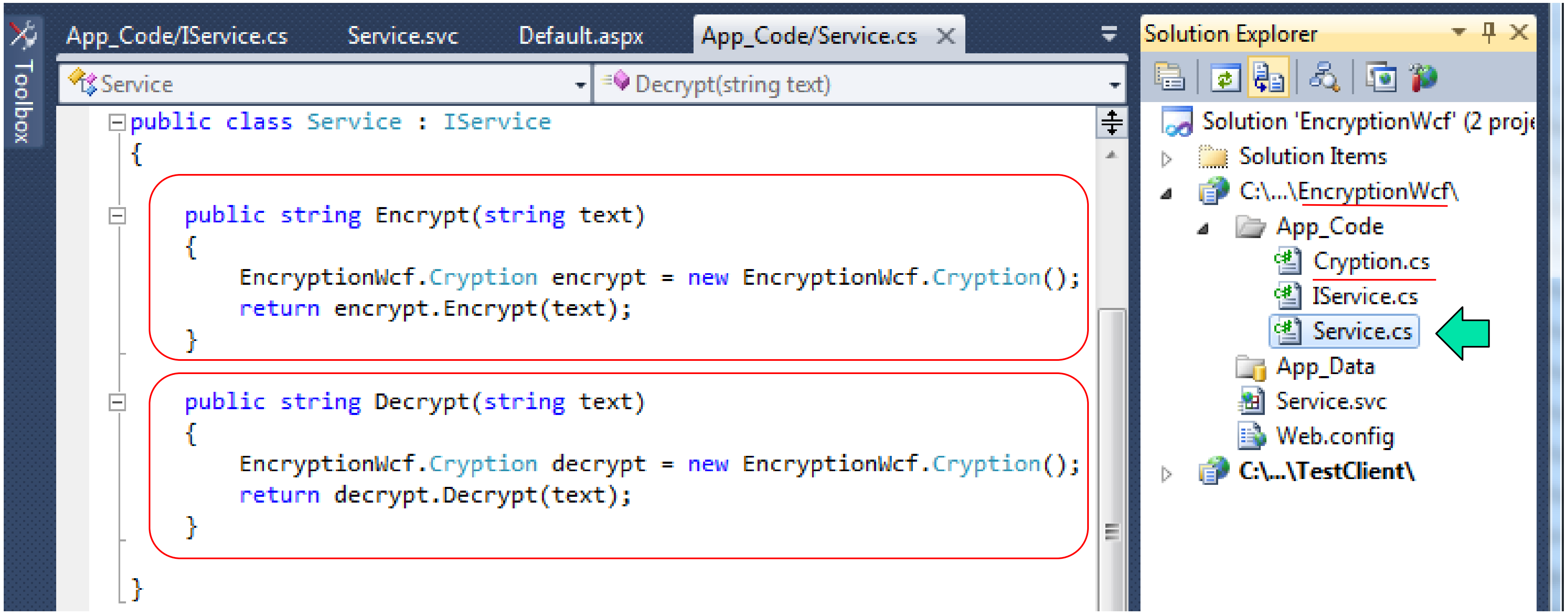


```
[ServiceContract]
public interface IService
{

    // TODO: Add your service operations here
    [OperationContract]
    string Encrypt(string text);

    [OperationContract]
    string Decrypt(string text);
}
```

Service Implementing OperationContract



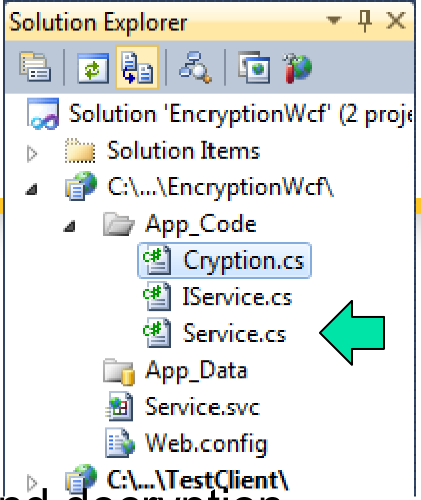
```
public class Service : IService
{

    public string Encrypt(string text)
    {
        EncryptionWcf.Cryption encrypt = new EncryptionWcf.Cryption();
        return encrypt.Encrypt(text);
    }

    public string Decrypt(string text)
    {
        EncryptionWcf.Cryption decrypt = new EncryptionWcf.Cryption();
        return decrypt.Decrypt(text);
    }

}
```

The Class Performing the Job

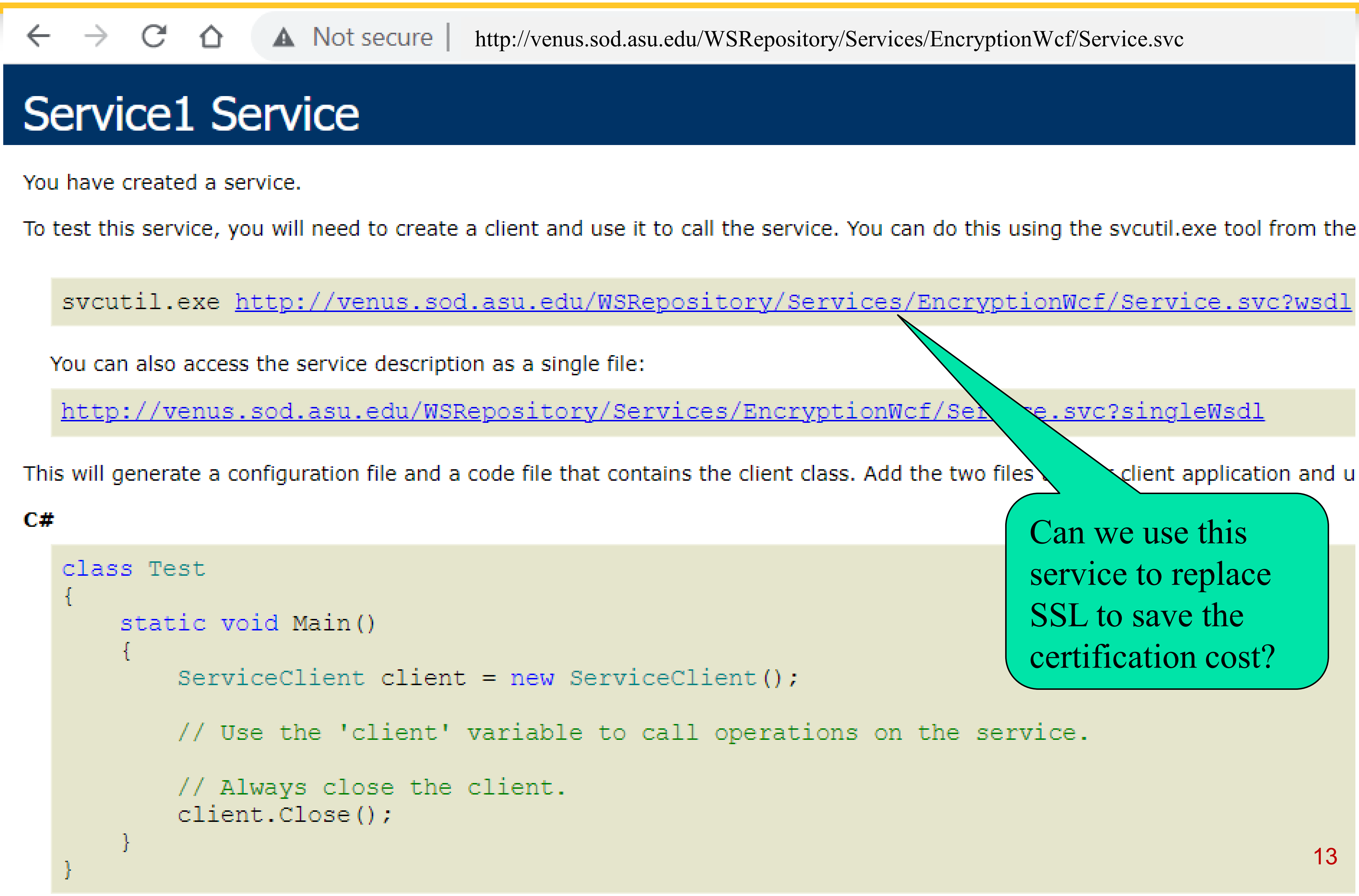

Solution Explorer
Solution 'EncryptionWcf' (2 proj
Solution Items
C:\...\EncryptionWcf\
App_Code
Cryption.cs
IService.cs
Service.cs
App_Data
Service.svc
Web.config
C:\...\TestClient\

```
using System; using System.IO; using System.Text;
using System.Security.Cryptography;
namespace EncryptionWcf {
   // Cryption class uses .Net Cryptography classes to perform en- and decryption
   public sealed class Cryption {
      byte[ ] seed = ASCIIEncoding.ASCII.GetBytes("cse44598"); // A seed from a
binary array for encryption. We could encrypt the seed to make it even more secure.
      public string Encrypt(string plainString) { // encryption using DES
         if (String.IsNullOrEmpty(plainString)) {
            throw new ArgumentNullException("The input cannot be empty or null!");
         }
         SymmetricAlgorithm saProvider = DES.Create();
         MemoryStream mStream = new MemoryStream();
         CryptoStream cStream = new CryptoStream(mStream,
            saProvider.CreateEncryptor(seed, seed), CryptoStreamMode.Write);
         StreamWriter sWriter = new StreamWriter(cStream);
// Continued next page
```

The Class Performing the Job (Contd.)

```
        sWriter.Write(plainString);
        sWriter.Flush(); // Buffer flush is necessary when switching writing modes
        cStream.FlushFinalBlock();
        return Convert.ToBase64String(mStream.GetBuffer(), 0, (int)mStream.Length);
    }
    public string Decrypt(string encryptedString) { // decryption using DES
        if (String.IsNullOrEmpty(encryptedString)) {
            throw new ArgumentNullException("The string cannot be empty or null!");
        }
        SymmetricAlgorithm saProvider = DES.Create();
        MemoryStream memStream = new MemoryStream
                (Convert.FromBase64String(encryptedString));
        CryptoStream cStream = new CryptoStream(memStream,
            saProvider.CreateDecryptor(seed, seed), CryptoStreamMode.Read);
        StreamReader reader = new StreamReader(cStream);
        return reader.ReadLine();
} } }
```

Encryption and Decryption Service Deployed



Not secure | http://venus.sod.asu.edu/WSRepository/Services/EncryptionWcf/Service.svc

Service1 Service

You have created a service.

To test this service, you will need to create a client and use it to call the service. You can do this using the svcutil.exe tool from the

```
svcutil.exe http://venus.sod.asu.edu/WSRepository/Services/EncryptionWcf/Service.svc?wsdl
```

You can also access the service description as a single file:

```
http://venus.sod.asu.edu/WSRepository/Services/EncryptionWcf/Se     e.svc?singleWsdl
```

This will generate a configuration file and a code file that contains the client class. Add the two files   client application and u

C#

```
class Test
{
    static void Main()
    {
        ServiceClient client = new ServiceClient();

        // Use the 'client' variable to call operations on the service.

        // Always close the client.
        client.Close();
    }
}
```

Can we use this service to replace SSL to save the certification cost?

ASU® Ira A. Fulton Schools of Engineering

Arizona State University

M14 L2 En/Decryption Service Client and Hashing

Ira A. Fulton Schools of Engineering
Arizona State University

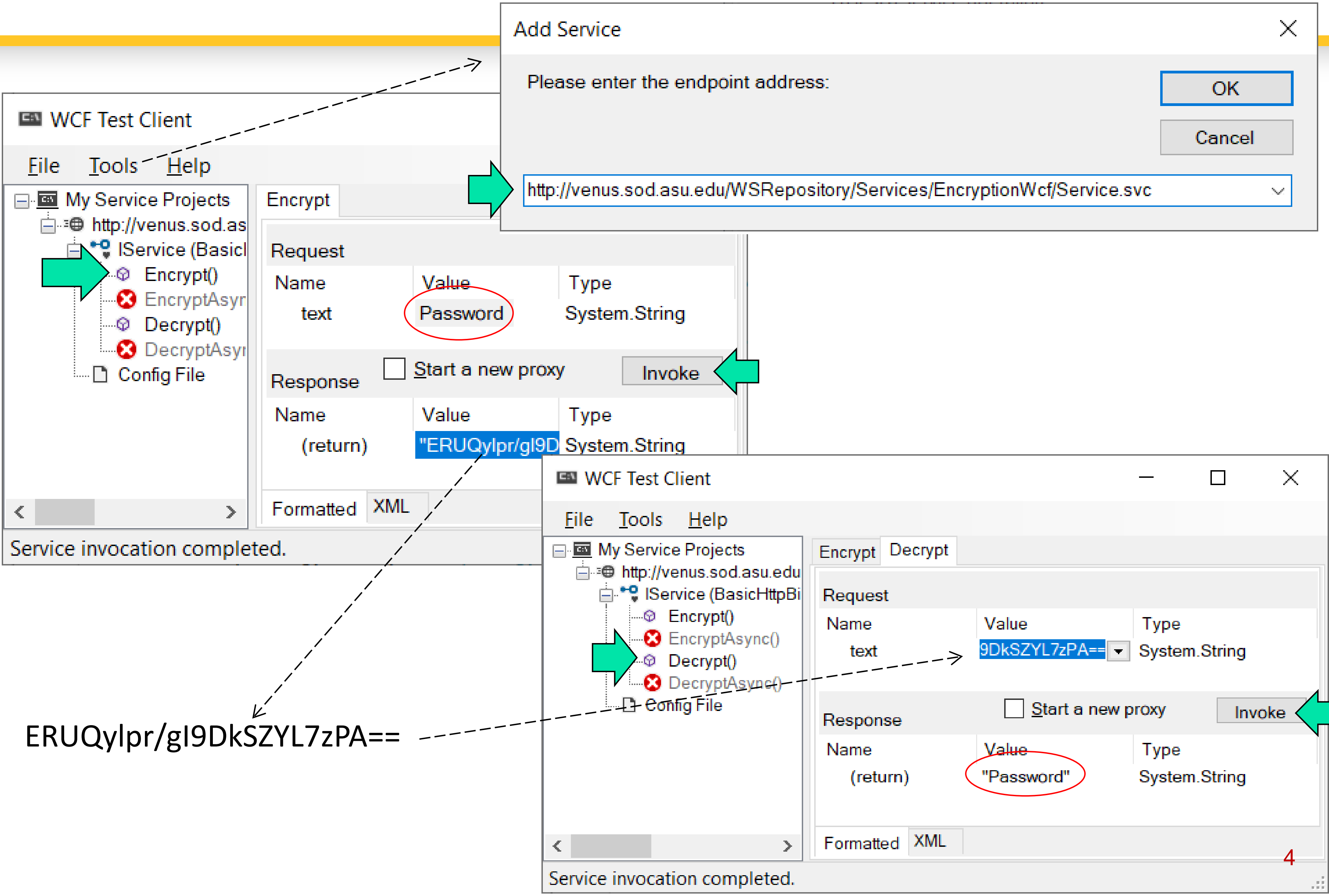
Lecture Outline

- Encryption/Decryption Service Client
  - Accessing Service in Client
  - Binding Protocols
- Hashing

WCF Service Clients

- WCF is used for creating services only;
- WCF services can be invoked by any service clients: Console application, ASP .Net application, Workflow application, Silverlight application, etc.
- Clients connect to the WCF services through proxy:
  - Proxy connects to the endpoints
  - Proxy is generated from service contracts
  - Client needs to specify what endpoint to invoke for its operations
  - Use "Add Service Reference" to add references to WCF services.

Test the Encryption and Decryption Service


Add Service

Please enter the endpoint address:

http://venus.sod.asu.edu/WSRepository/Services/EncryptionWcf/Service.svc

WCF Test Client

Request

Name	Value	Type
text	Password	System.String

Response

Name	Value	Type
(return)	"ERUQylpr/gl9D	System.String

Service invocation completed.

WCF Test Client

Request

Name	Value	Type
text	9DkSZYL7zPA==	System.String

Response

Name	Value	Type
(return)	"Password"	System.String

Service invocation completed.

ERUQylpr/gI9DkSZYL7zPA==

Endpoints: A-B-C Created in the Client

```
<configuration>
<system.serviceModel>
<bindings>
   <basichttpsBinding>
    <binding name="BasichttpsBinding_IService“ >
      <security mode="None">
      <transport clientCredentialType="None" proxyCredentialType="None“ realm="" />
      <message clientCredentialType="UserName" algorithmSuite="Default" />
     </security>
    </binding>
   </basichttpsBinding>
</bindings>
<client>
     <endpoint
         address="https://venus.sod.asu.edu/WSRepository/Services/EncryptionWcf/Service.svc"
         binding="basichttpsBinding" bindingConfiguration="BasichttpsBinding_IService"
         contract="EncryptService.IService" name="BasichttpsBinding_IService"
     </endpoint>
</client>
</system.serviceModel>
</configuration>
```

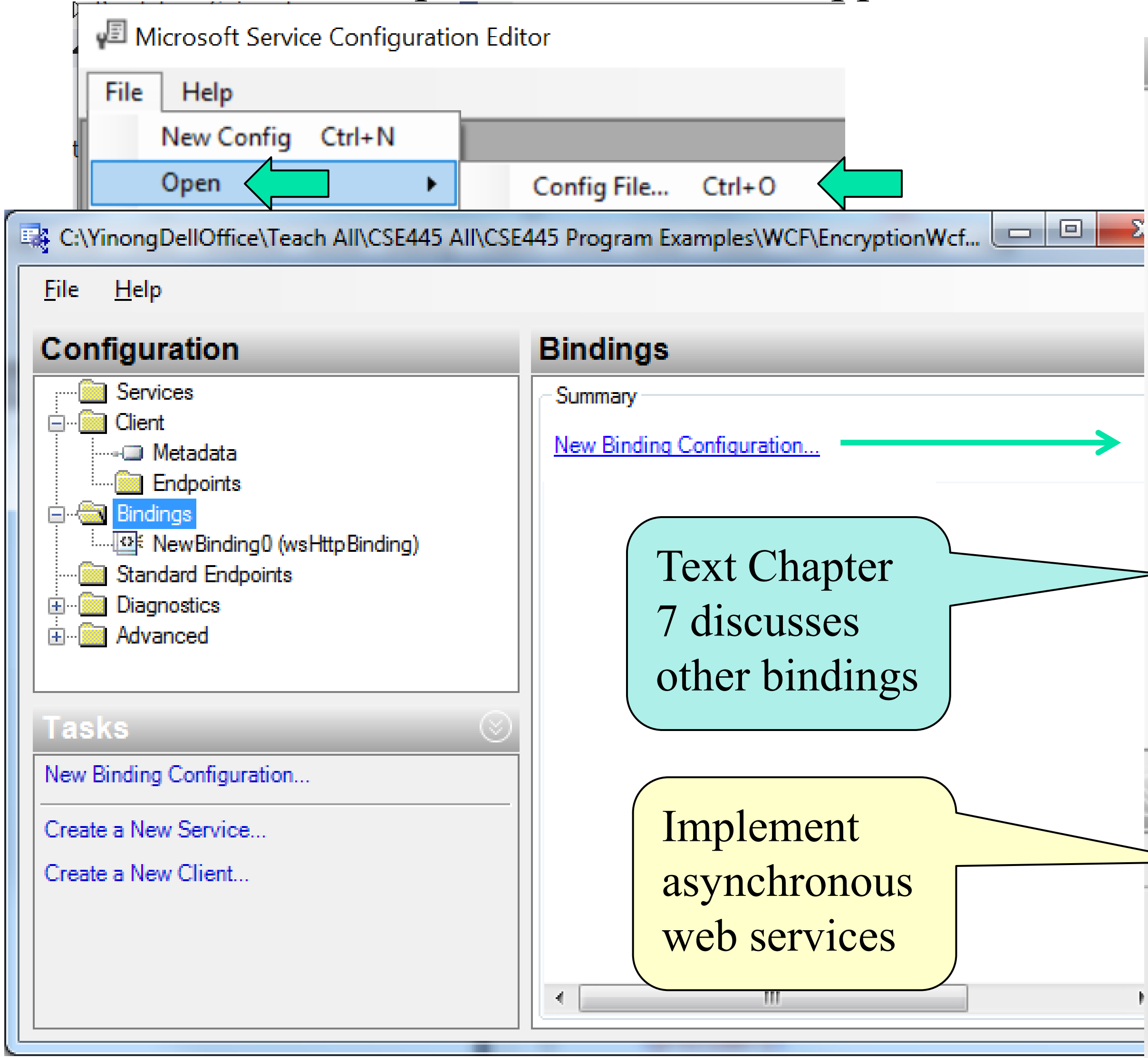
Endpoint: A-B-C

What other bindings available?

Adding a Different Binding Protocol

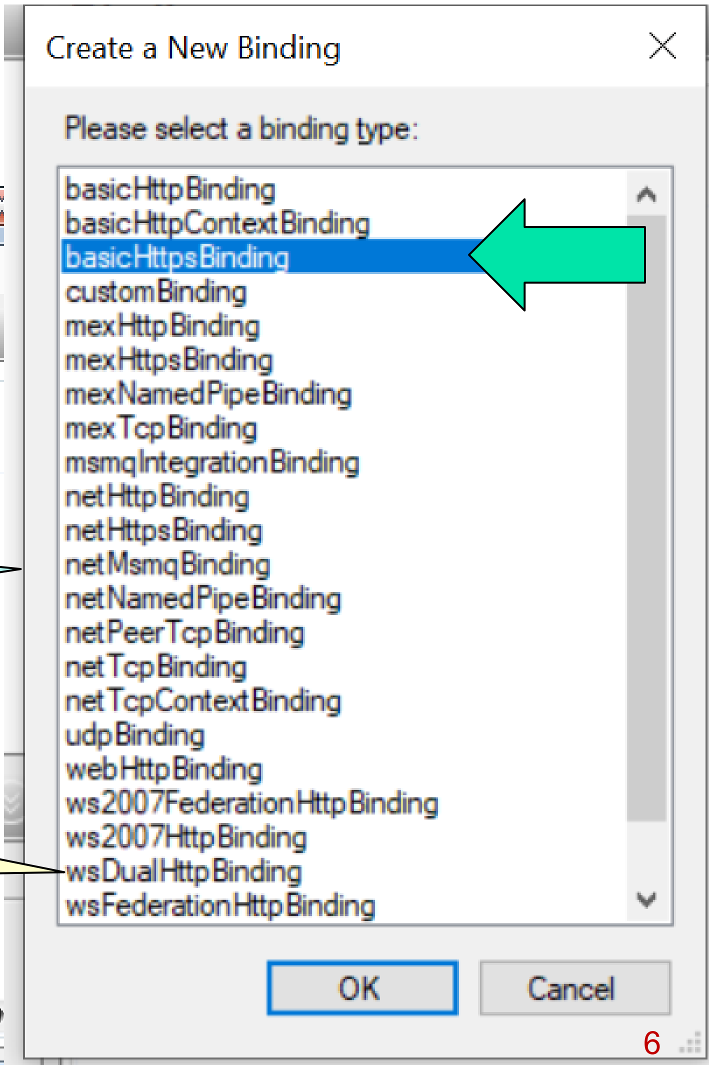
In VS Menu: Tools → WCF Service Configuration Editor.

Choose File → Open: Browse to the application and select the Web.config file

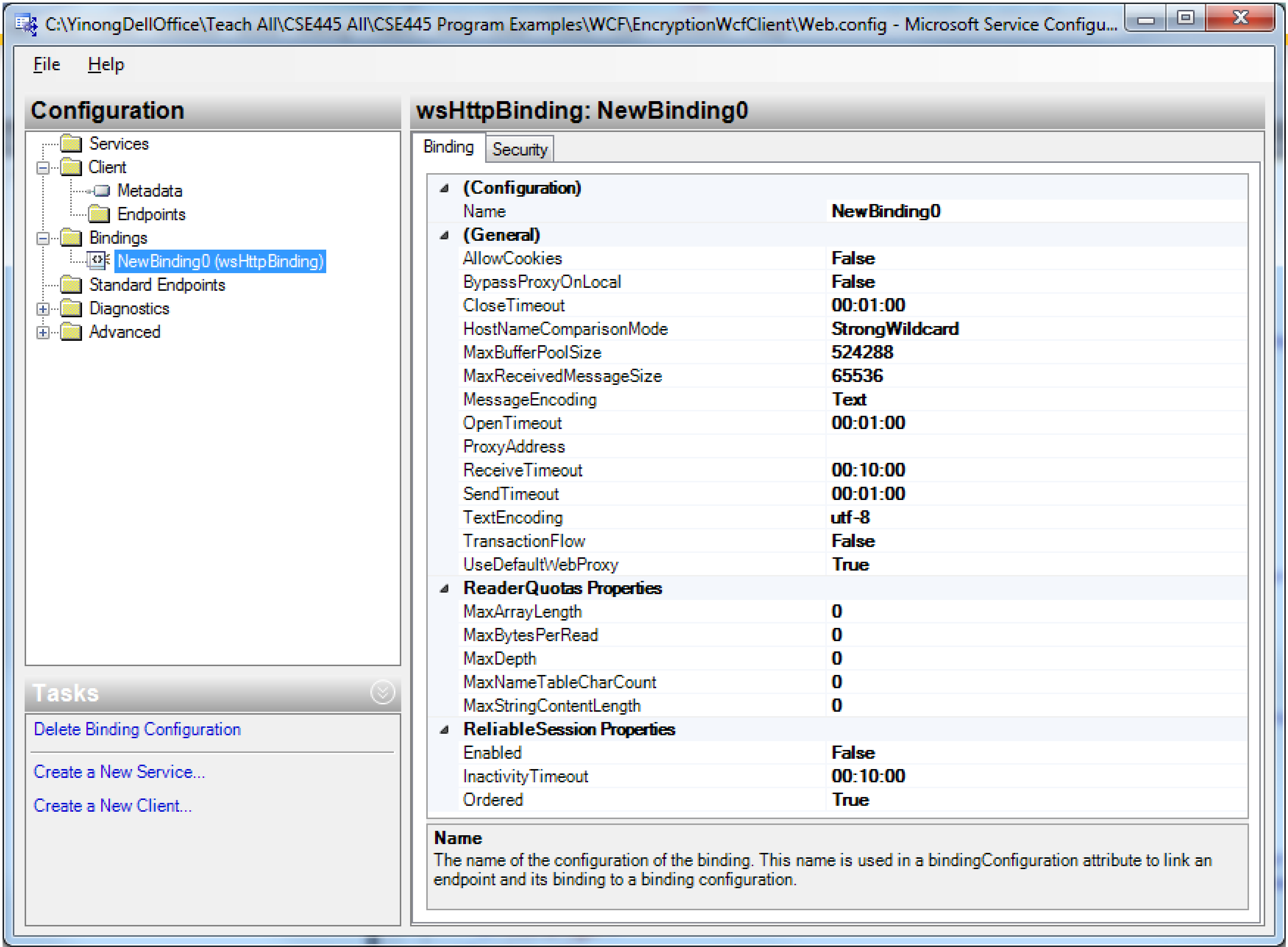


Text Chapter 7 discusses other bindings

Implement asynchronous web services



Editing the wsHttpBinding in Configuration File


C:\YinongDellOffice\Teach All\CSE445 All\CSE445 Program Examples\WCF\EncryptionWcfClient\Web.config - Microsoft Service Configu...

wsHttpBinding: NewBinding0

(Configuration)	
Name	NewBinding0
(General)	
AllowCookies	False
BypassProxyOnLocal	False
CloseTimeout	00:01:00
HostNameComparisonMode	StrongWildcard
MaxBufferPoolSize	524288
MaxReceivedMessageSize	65536
MessageEncoding	Text
OpenTimeout	00:01:00
ProxyAddress	
ReceiveTimeout	00:10:00
SendTimeout	00:01:00
TextEncoding	utf-8
TransactionFlow	False
UseDefaultWebProxy	True
ReaderQuotas Properties	
MaxArrayLength	0
MaxBytesPerRead	0
MaxDepth	0
MaxNameTableCharCount	0
MaxStringContentLength	0
ReliableSession Properties	
Enabled	False
InactivityTimeout	00:10:00
Ordered	True

Name
The name of the configuration of the binding. This name is used in a bindingConfiguration attribute to link an endpoint and its binding to a binding configuration.

When Should You Not Use Encryption/Decryption Web Service?

- If you do not have SSL (Secure Socket Layer) protocol for Secure HTTP (HTTPS), the data sent to / from En-/Decryption Service is in clear text
- You may want to create a local DLL library function in your application:
  - You call local encryption function before sending the data to remote server/client;
  - You call the decryption function after you have retrieved the encrypted data;
  - If the receiver site is different from sender, both sides must install the save encryption and decryption functions.
  - Do not call Encryption/Decryption Service in your DLL functions.

Security Vulnerability Discussion

- Open Key Systems generate a public key and a secrete key.
- If the secrete key is intruded, the security is broken.
- A vulnerability is a weakness in the system for intrusion. Without a weakness, it would take 100 years to breake in the system using systematic exploiting, such as dictionary attack.
- For example, Microsoft Security Advisory (http://technet.microsoft.com/en-us/security/advisory/2416728) in 2010 acknowledged that a vulnerability was found in ASP .Net through padding oracle attacks.
- Padding oracle attacks:
    - Generate a large number of incorrectly encrypted messages of different types and send requests for decrypting these messages;
    - The system (e.g. ASP .Net) sends different error messages back;
    - Using the error messages types to figure out the secrete key.
- If you need ultrahigh security, do not use ASP .Net crypto library. They are not design for ultrahigh security purposes. Recall that ultrahigh encryption and decryption algorithm are munitions like ABC weapons!

Data Hashing

When to Use Encryption and Decryption?

When to use Data Hashing?

Data Hashing

- A hash value is a numeric value with fixed length that uniquely identifies a data, but without revealing the data's value;
- One-way feature: from data to hash value, but cannot recreate the data from the hash value;
- Different from the digital signature method in Open Key system, where there are two keys, and data and signature are typically mixed, and data hashing has one key to create a separate signature.
- Applications:
  - Proof of identity: Session ID;
  - Separate digital signature: For a large piece of data, e.g., 100 pages of a *contract*, we can generate a (short) hash value1 from the *contract* and send the hash value1 with the *contract*. The receiver re-generates hash value2 from the received contract. If value1 == value2, the contract is not modified during the transmission.

The Best Way to Secure User’s Password

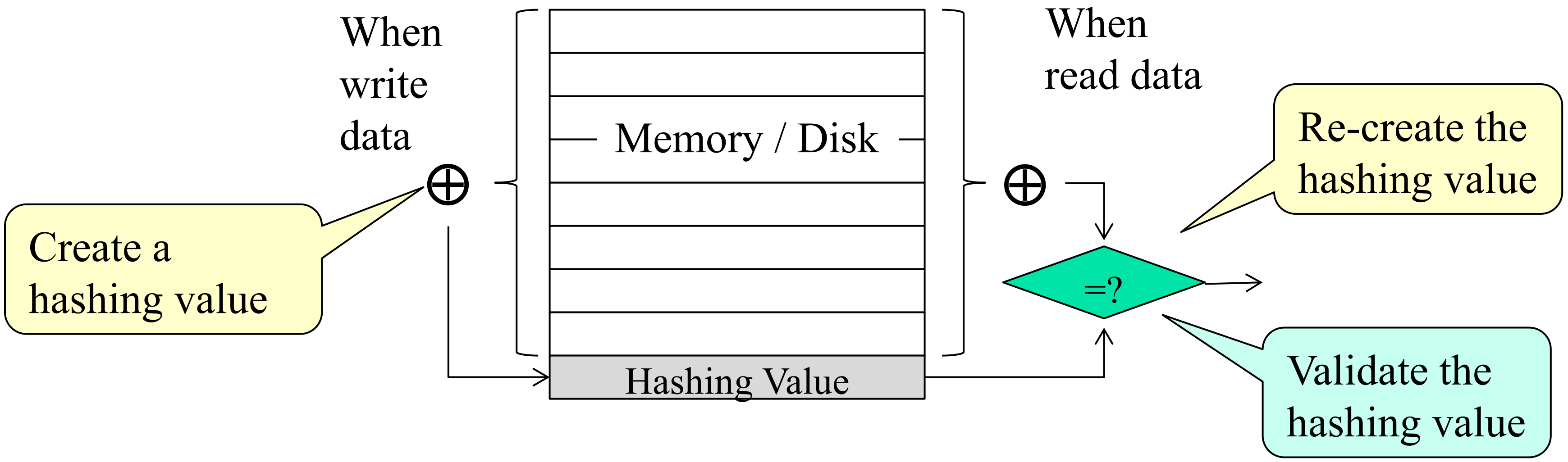
1. Save users’ passwords in a database (or a file). Secure the database using access control.
2. Save users’ passwords in a database. Make database writable, searchable, but not readable. When validating a user, search (user-name + password) in the database and it returns true or false.
3. Encrypt the passwords before saving into a database. Decrypt the password when validating a user.

In all solutions above, someone can see the passwords. Is there a way that no one (no hackers) can see the stored passwords?

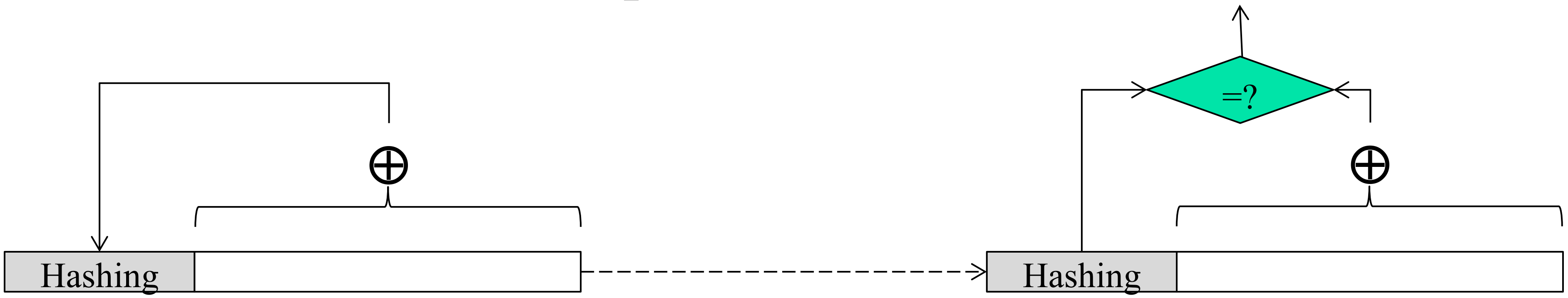
4. Hash the passwords and save the hashed values. When validating a user, hash the user entered password and compare it with the stored hashed value.

Hashing for Dependable Data Storage and Communication

- Used for Dependable Data Storage.



- Used for Dependable Communication.



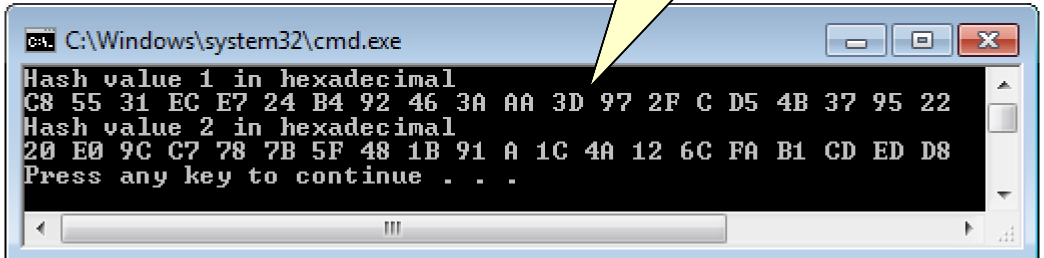
Hashing Example in Application

```
static void Main(string[] args)  {
    byte[] HashValue1;  byte[] HashValue2;
    string MsgShort = "You won the lottery";
    string MsgLong = "This is an official announcement from the government and the
message comes with a digital signature";
    UnicodeEncoding Uce= new UnicodeEncoding(); // UnicodeEncoding object
    byte[] BytesShort = Uce.GetBytes(MsgShort); // convert to byte array
    byte[] BytesLong = Uce.GetBytes(MsgLong);
    SHA1Managed SHhash = new SHA1Managed(); //Create a SHA1 object
    HashValue1 = SHhash.ComputeHash(BytesShort); // Hashing
    HashValue2 = SHhash.ComputeHash(BytesLong);
    Console.WriteLine("Hash value 1 in hexadecimal");
    foreach (byte b in HashValue1) Console.Write("{0:X} ", b);
    Console.WriteLine("\nHash value 1 in hexadecimal");
    foreach (byte b in HashValue2)
            Console.Write("{0:X} ", b);
    Console.WriteLine();
}
```

Short and long messages

hex

The length of 2 msgs are the same



```
Hash value 1 in hexadecimal
C8 55 31 EC E7 24 B4 92 46 3A AA 3D 97 2F C D5 4B 37 95 22
Hash value 2 in hexadecimal
20 E0 9C C7 78 7B 5F 48 1B 91 A 1C 4A 12 6C FA B1 CD ED D8
Press any key to continue . . .
```

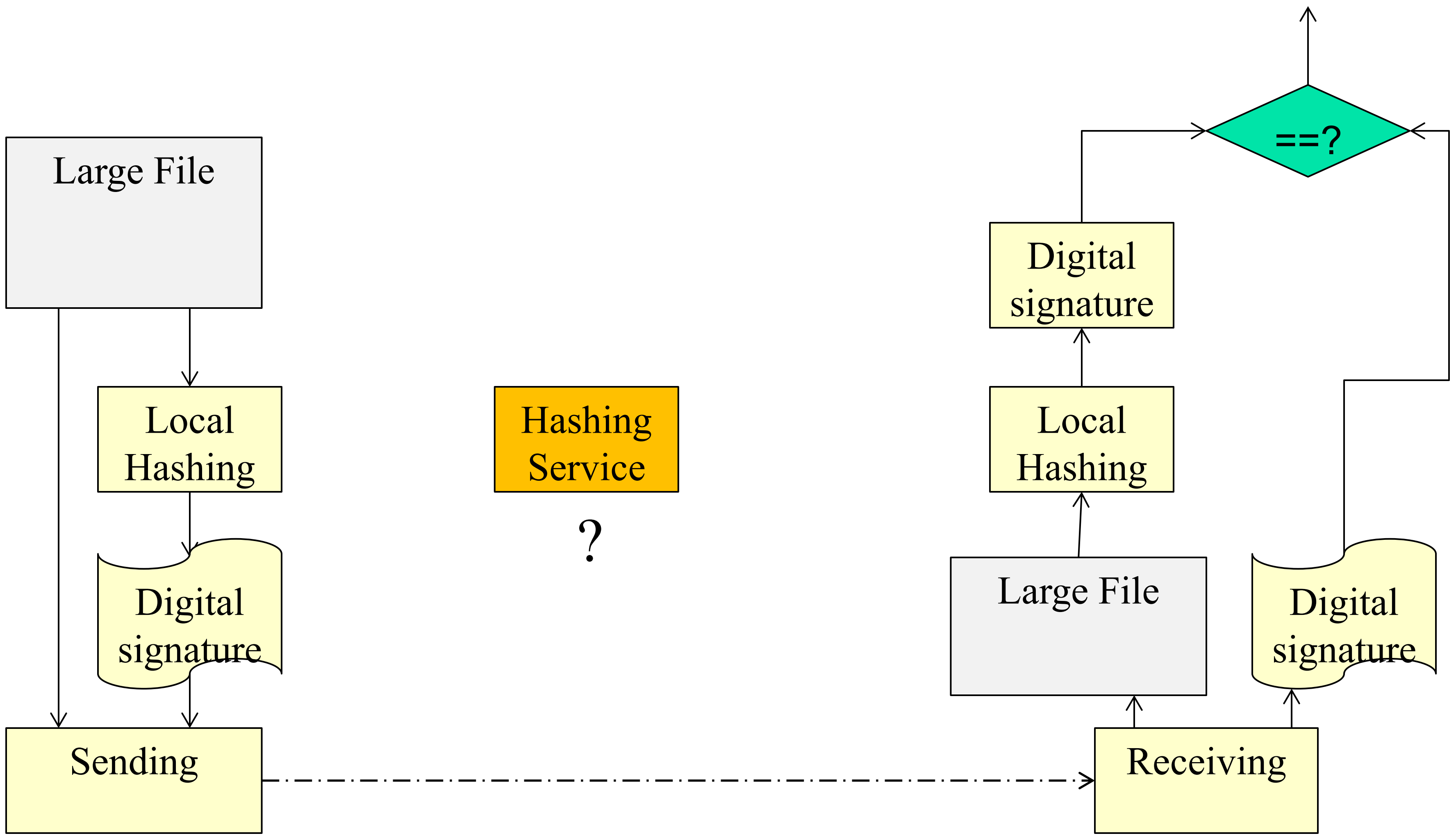
Creating a Data Hashing Service

```
[ServiceContract]
public interface IService {
    [OperationContract]
    string Hash(string value, string salt);
}
```

```
using System;
using System.Text;
using System.Security.Cryptography;
using System.Text;public class Service : IService {
    //Hash a value with a salt
public string Hash(string value, string salt) {
        using (var sha = new SHA512CryptoServiceProvider()) {
            var hashedString
 = sha.ComputeHash(Encoding.Default.GetBytes(value + salt));
            return Convert.ToBase64String(hashedString);
        }
    }
}
```

Test: http://venus.sod.asu.edu/WSRepository/Services/HashSha512/Service.svc

Example: How can you create an efficient Digital Signature System for a Large File?



Ira A. Fulton Schools of Engineering

Arizona State University

M14 L3
Reliable Messaging

Ira A. Fulton Schools of Engineering
Arizona State University

Lecture Outline

- **Reliable Messaging**
  - At-Least-Once delivery semantics
  - At-Most-Once delivery semantics
  - Exactly-Once delivery semantics
  - Guaranteed message ordering
- **Transaction**

Web Service Reliability Concerns

"Web services are not yet widely used because of security concerns. But there's an even bigger roadblock waiting just down the road -- it's called trust. The big issue is "Will the service work correctly every time when I need it?" As yet few are thinking about the issues of testing and certification, we suggest that testing and certification of Web services is not business as usual and that new solutions are needed to provide assurance that services can really be trusted."

-- *By CBDi Forum*, at

https://searchmicroservices.techtarget.com/tip/Is-Trust-the-Achilles-Heel-of-Web-services

11 Jul 2002.

Web Service Testing for Reliability

- Software testing can take more effort than software development, and thus there are huge demands on software testing professionals;
- Software black-box testing vs. white-box testing
- How do we do Web service testing?
  - Source code not available: Black-box Testing;
  - WSDL file is available: WSDL analysis to obtain URL, operations, input and output types;
  - Generate test cases: based on the input types;
  - Dynamically invoke the Web service;
  - Analyze the return values

WS-RM: ReliableMessaging and More

- WS-ReliableMessaging (WS-RM) deals with faults at the message level, including
  - Lost messages
  - Duplicated messages
  - Messages received out of order
  - Suitable in the scenarios where both parties are online
- Message Queuing, e.g., MSMQ:
  - Ensure reliable communication between the sender and the receiver.
  - Suitable in the scenarios where the receiver may be offline
- Transactions for enterprise application

WS-ReliableMessaging (WS-RM)

WS-RM Adoption Among Leading Vendors (Data 2007)

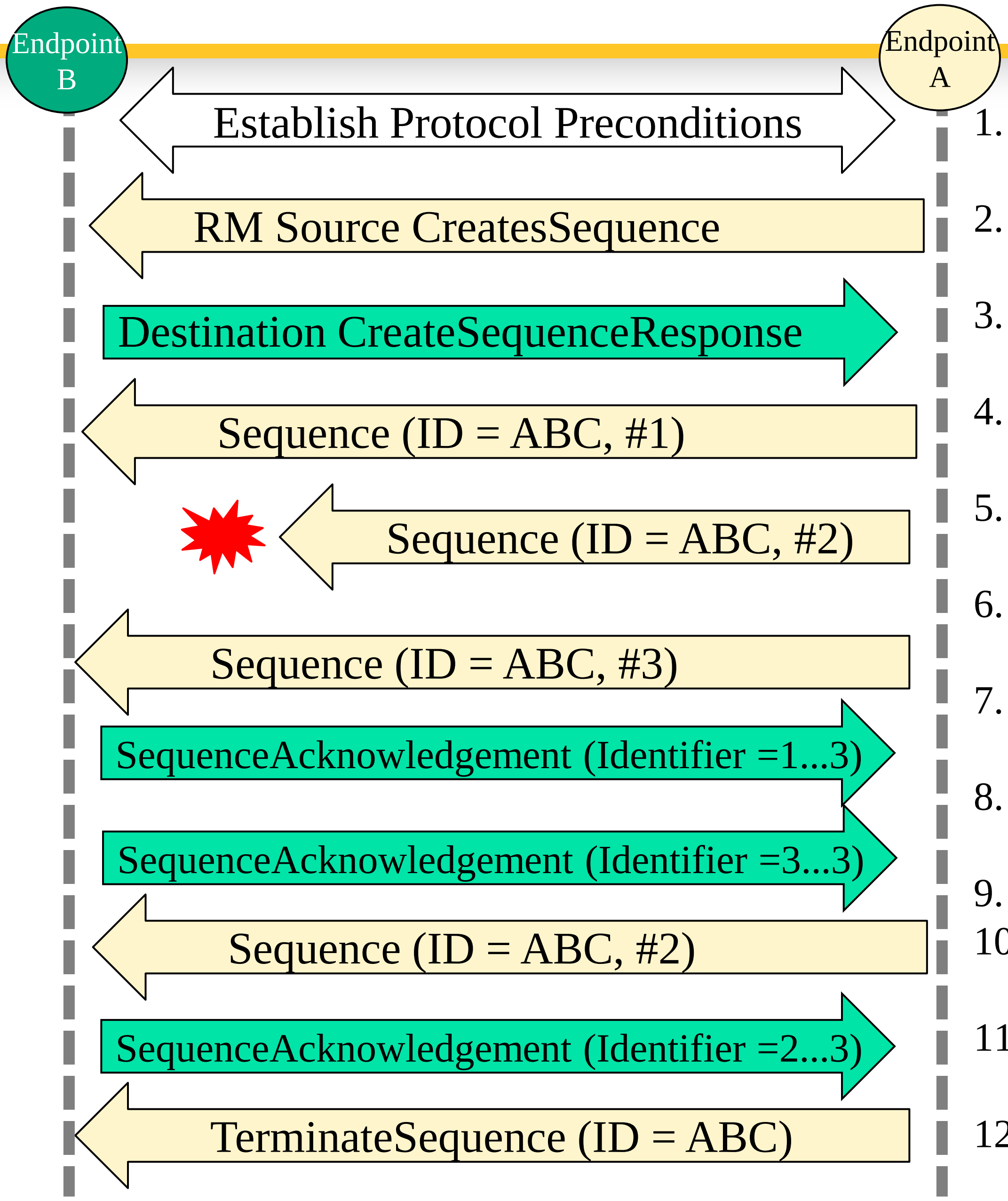
Company/Platform	**WS-RM Support**
Microsoft WCF	Yes
IBM, ETTK—AlphaWorks	Yes
IBM WebSphere Message Q	Yes
Oracle SOA Suite	Yes
BEA, WebLogic 9.0	Yes
Cape Clear	Yes
Systinet	Yes
Blue Titan	Yes
Apache Axis 1.3 Sandesha 05	Yes
SAP NetWeaver Process 07	Yes
Sonic	In Development
Tibco	In Development

WS-ReliableMessaging Features

WS-RM specification defines the following reliability features

- Guaranteed message delivery, or At-Least-Once delivery semantics
- Guaranteed message duplicate elimination, or At-Most-Once delivery semantics
- Guaranteed message delivery and duplicate elimination, or Exactly-Once delivery semantics
- Guaranteed message ordering for delivery within a group of messages

WS-ReliableMessaging Protocol



Endpoint B
Endpoint A
Establish Protocol Preconditions
RM Source CreatesSequence
Destination CreateSequenceResponse
Sequence (ID = ABC, #1)
Sequence (ID = ABC, #2)
Sequence (ID = ABC, #3)
SequenceAcknowledgement (Identifier =1…3)
SequenceAcknowledgement (Identifier =3…3)
Sequence (ID = ABC, #2)
SequenceAcknowledgement (Identifier =2…3)
TerminateSequence (ID = ABC)

1. It includes policy exchange, endpoint resolution, and establishing trust;
2. RM Source requests creation of a new Sequence;
3. RM Destination creates a new Sequence and returns its unique Identifier (ID = ABC);
4. RM Source transmits messages in the Sequence with MessageNumber 1.
5. RM Source transmits the 2nd message with MessageNumber 2, and is lost in transit;
6. RM Source transmits the 3rd message with MessageNumber 3,
7. The 1st message is acknowledged by destination;
8. The 3rd message is acknowledged by destination;
9. No acknowledgement in given time;
10. RM Source retransmits the 2nd message with MessageNumber 2;
11. The 2nd message is acknowledged by destination;
12. RM Source terminates the sequence ABC.

Web.config Element for Reliable Messaging

To use reliable messaging in your Web application/Web service is simple. Simply Add the following binding element into your Web.config file, and the system will create the SOAP communications that implement Reliable Messaging.
For more detail: Text Section 6.4.2.

```
<customBinding>
      <binding
            configurationName="customReliableHttpBinding">
                  <reliableSession ordered="true" />
                  <textMessageEncoding messageVersion="Default"
                        encoding="utf-8" />
                  <httpTransport />
      </binding>
</customBinding>
```

SOAP Message for Creating a Sequence

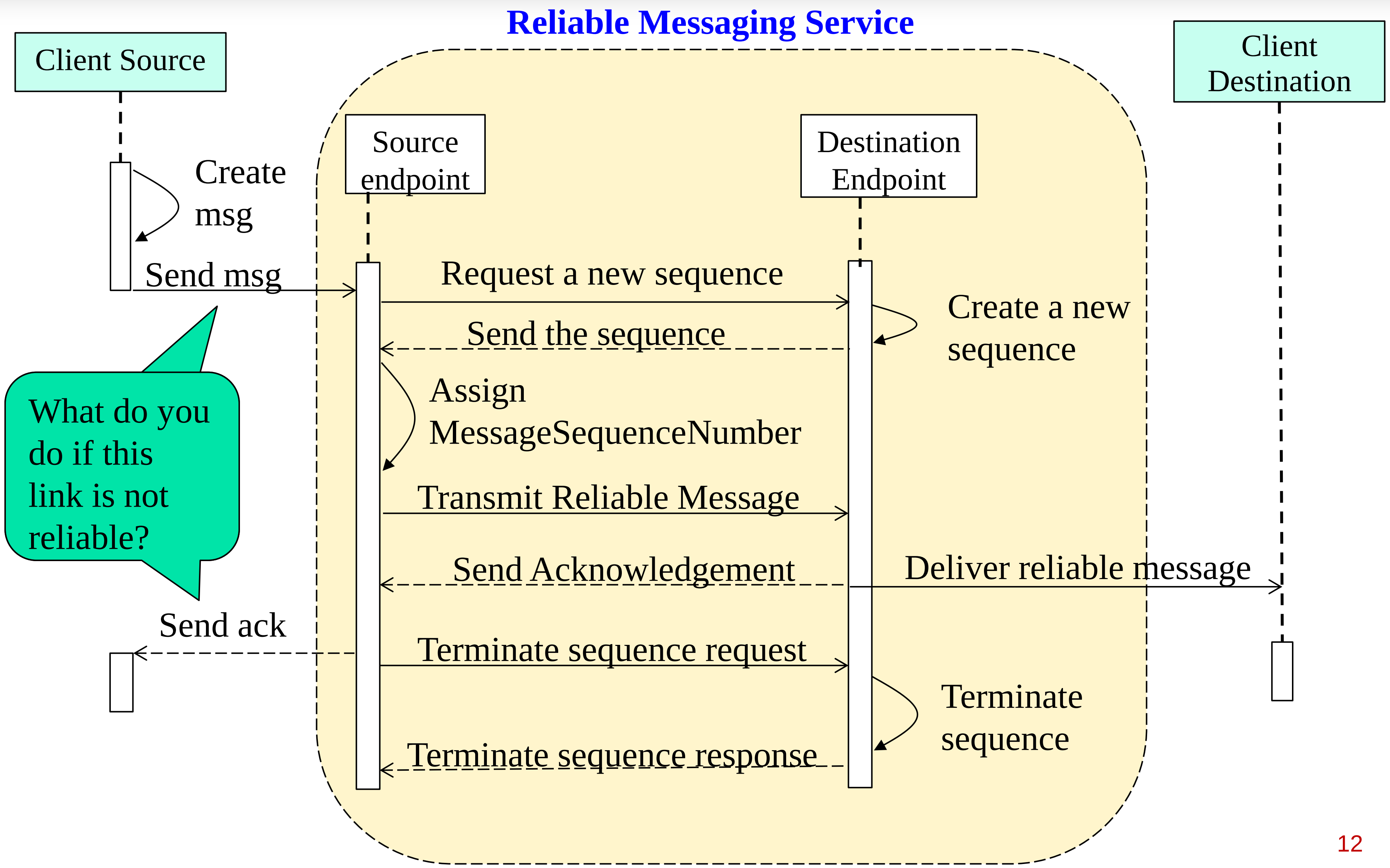
```
<soap:body>
    <wsrm:CreateSequence ...>
        <wsrm:AcksTo> wsa:EndpointReferenceType </wsrm:AcksTo>
        <wsrm:Expires ...> xs:duration </wsrm:Expires> ?
        <wsrm:Offer ...>
            <wsrm:Identifier ...> xs: http://cse445.com/ </wsrm:Identifier>
            <wsrm:Endpoint> wsa:EndpointReferenceType </wsrm:Endpoint>
            <wsrm:Expires ...> xs:duration </wsrm:Expires> ?
            <wsrm:IncompleteSequenceBehavior>
                wsrm:IncompleteSequenceBehaviorType
            </wsrm:IncompleteSequenceBehavior> ?
        </wsrm:Offer> ?
</wsrm:CreateSequence>
<soap:body>
```

For Message Number and Acknowledge

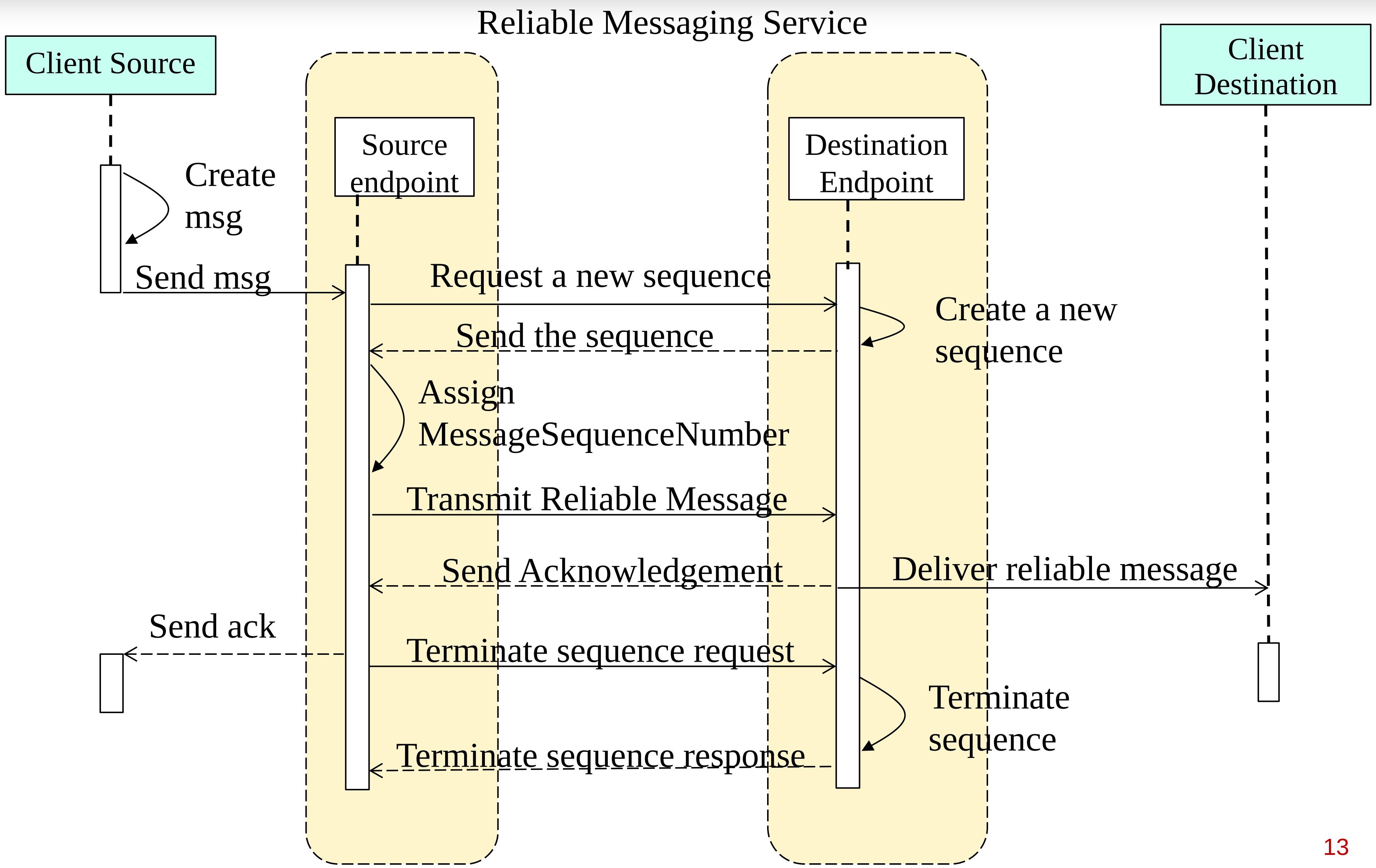
```
<soap:header>
  <wsrm:sequence>
    <wsrm:identifier>http://cse445.com/RM/ABC</wsrm:identifier>
    <wsrm:messagenumber>1</wsrm:messagenumber>
  </wsrm:sequence>
</soap:header>
```

```
<soap:header>
  <wsrm:sequenceacknowledgement>
    <wsrm:identifier>http://cse445.com/RM/ABC</wsrm:identifier>
    <wsrm:acknowledgementrange lower="1" upper="1" />
    <wsrm:acknowledgementrange lower="3" upper="3" />
  </wsrm:sequenceacknowledgement>
</soap:header>
```

Reliable Messaging as a Service, e.g. ESB



Create a Local Reliable Service



WCF Transaction

Ensuring Data Integrity

in multiple services (or database) calls,

where some services may fail.

Client Initiated Transactions

- WCF transactions are built on top of System.Transactions + WS-*
- Initiate a transaction in client code
- Transaction is flowed to all services called that support transactions: The service must support transaction.

```
using (TransactionScope scope = new TransactionScope())
{
        // Call Service1 – transaction flowed to service1
        // Call Service2 – transaction flowed to service2
        scope.Complete();
} // All commit or all rollback
```

Code with More Detail

```
using (TransactionScope transScope = new TransactionScope()) {
    // Create an connection channel
    using (SqlConnection connection1 = new SqlConnection(connectString1)) {
        connection1.Open(); // Opening connection1 to prepare withdrawal
        connection1.BeginTransaction(IsolationLevel.Serializable);
        // The connection will be listed in the TransactionScope
        // Create another connection channel
        using (SqlConnection connection2 = new SqlConnection(connectString2)) {
            connection2.Open(); // Opening connection2 to prepare deposit
            connection2.BeginTransaction(IsolationLevel.Serializable);
            // The connection will be listed in the TransactionScope
        }
    }
    //  The actual method that commits the transaction.
    transScope.Complete();
}
```

What happen if this operation fails in the way one action is completed but not the other?

Enabling Transactions

- Transactions do not flow by default
- Requirements:
  - Enclose service calls in transaction scope on client
  - Attribute transactionFlow = true on binding
  - [TransactionFlow] attribute on service contract methods
    - Optional – specify transaction scoping options
  - OperationBehavior attribute in service methods
    - TransactionFlowRequired = true
    - TransactionAutoComplete = true
  - Enable WS-Atomic Transactions on DTC (Distributed Transaction Coordinator):
    - xws_reg –wsat + at command line

Other Dependable Computing Techniques: Redundancy

- N-Version Programming
- 2-Version Programming with recovery block
- Error Control Code and Redundancy
- Shannon's Theory in 1948:
  Any given reliability target can be achieved, as long as sufficient redundancy is provided.
- Error Control Code …

Ira A. Fulton Schools of Engineering

Arizona State University

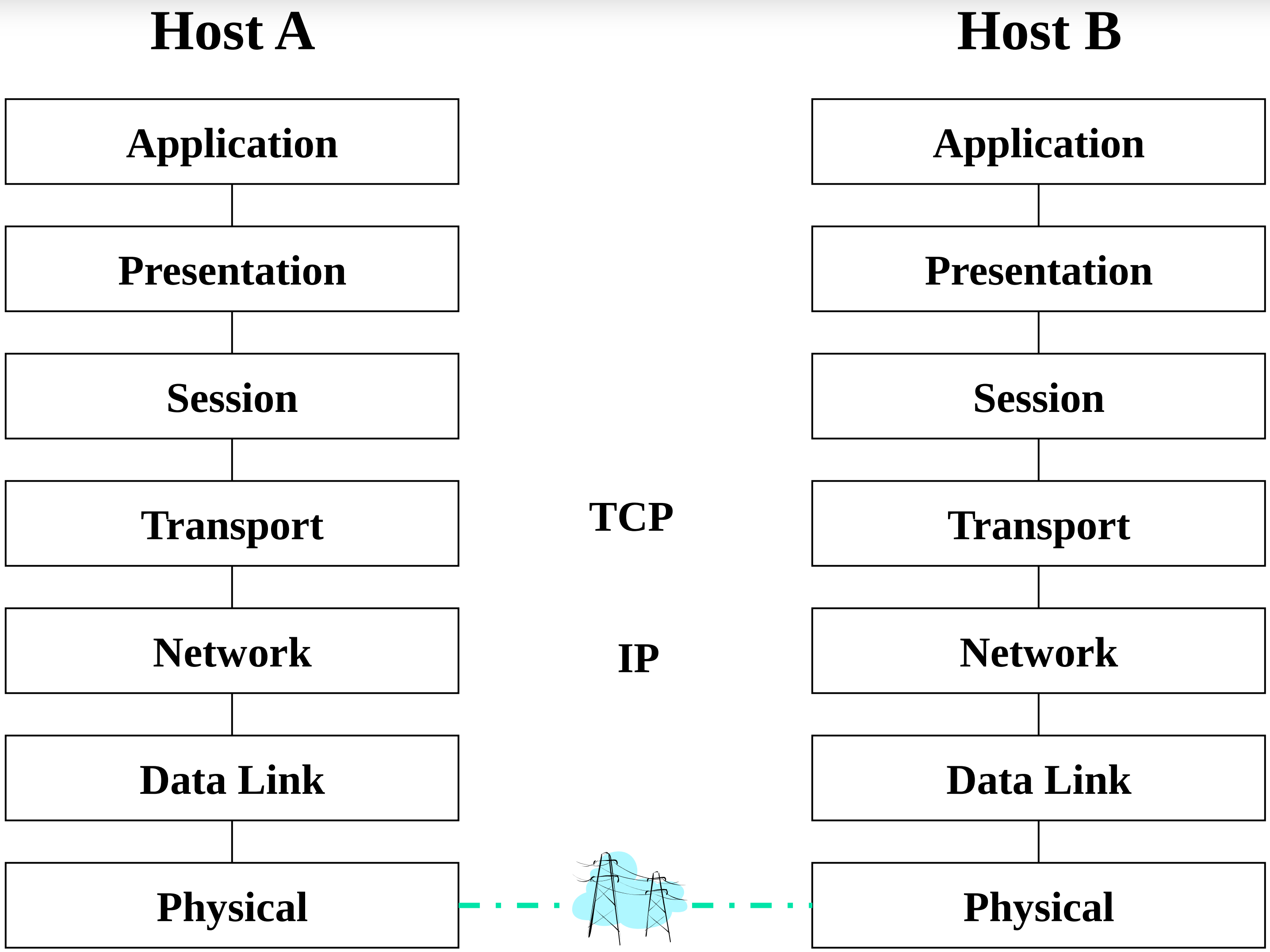
Error Control Codes

Ira A. Fulton Schools of Engineering
Arizona State University

Lecture Outline

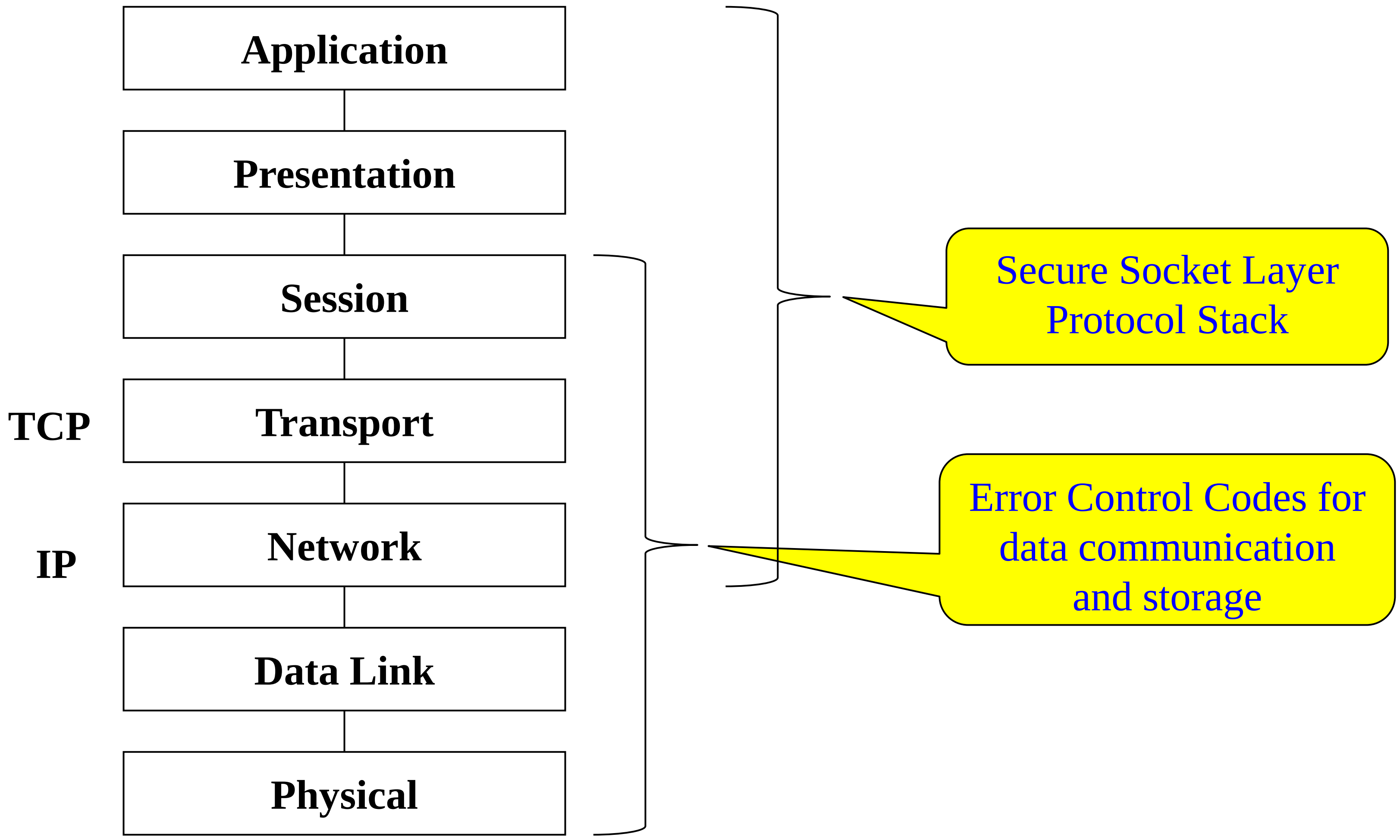
- **Reliable and Secure Communication**
- **Shannon's Theorem**
- **Error Control Codes**
  - Redundancy and Code Distance
  - Parity Code
  - Checksum
  - Arithmetic Codes
  - m-of-n Codes
  - 5G Error Control Codes

Communication and Protocols



Open System Interconnection (OSI) 7-layer reference model

Reliable and Secure Communication


Application
Presentation
Session
Transport
Network
Data Link
Physical
TCP
IP
Secure Socket Layer Protocol Stack
Error Control Codes for data communication and storage

1956 Dartmouth Conference: The Founding Fathers of AI

Father of information theory

John MacCarthy

Marvin Minsky

Claude Shannon

Ray Solomonoff

Alan Newell

Herbert Simon

Arthur Samuel

Oliver Selfridge

Nathaniel Rochester

Trenchard More

Founding fathers of AI. Courtesy of scienceabc.com

Shannon's Theorem [1948]:

Even in a noisy channel, errors in data transmission can be reduced to any desired level, if a certain minimum percentage of redundancy is maintained by means of proper encoding and decoding of the data.

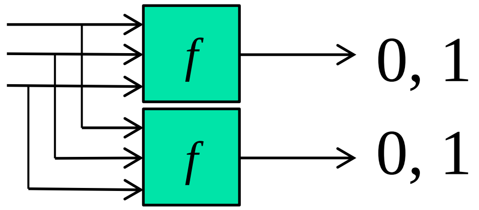
- This theorem founded the theories of error control codes as well fault tolerant computing, including maximum capacity, efficiency, and reliability.
- Shannon' theorem did not suggest any procedure for constructing such codes.
- Golay [1949], Hamming [1950], and many other pioneers in the area developed Shannon' theorem into Coding Theory.
- Most fault tolerance techniques can be viewed as implementations of error control codes.

Basic Definitions

- **Coding** is the representation of information (signals, numbers, messages, etc.) by **code symbols**, or **words** of code symbols
- Let *W* be the **word space**: set of all possible words in a code. In order to control errors, *W* is divided into two subsets *C*, and *W* - *C*. Only the words in *C* are used for representing information, while those in *W* - *C* are redundant, indicating incorrect words.
- *C* is the **code** space, the words in *C* are called **codewords**.

Redundancy and Distance

- Example: We can use two identical modules/lines to represent the same information. The entire output space is
  - W = {00, 01, 10, 11}
  - The code space $C = \{00, 11\}$



- Distance between two words *d(x, y)*: Bit positions in which the words x and y differ.
  - *For example:* $d(00, 01) = 1$ and $d(01, 10) = 2$

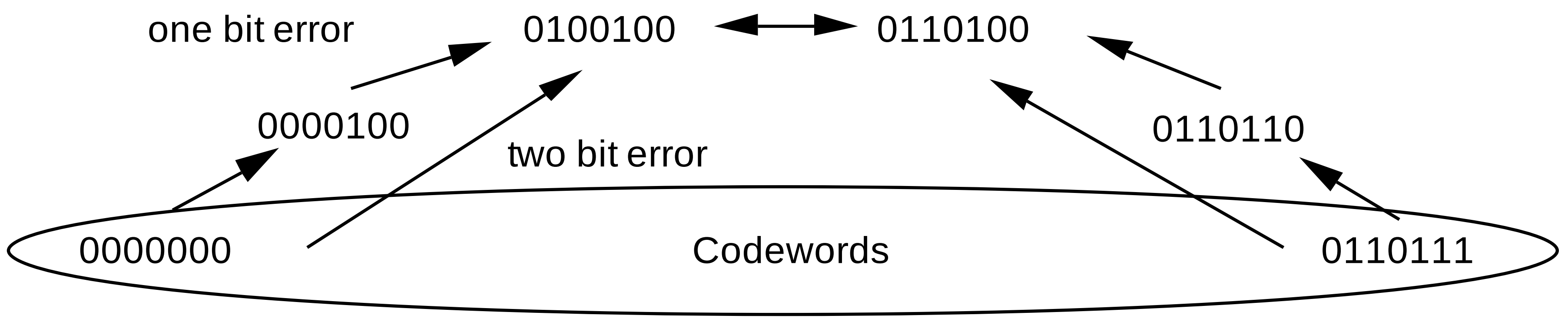
- Hamming distance $d_m(C)$ of a code space C:

  $d_m(C) = \min(d(x, y))$ for all pairs

  - *For* $C = \{00, 11\}$, $d_m(C) = 2$

Error Detection and Correction Capability

- Hamming distance determines the error detection and correction capability:
  - If $d_m(C) = p + 1$, the code C can detect all errors, in which up to p bits are erroneous.
  - If $d_m(C) = 2l + 1$, the code C can correct all errors, in which up to l bits are erroneous. For example, if $d_m(C) = 5$, C can detect up to 4 bit errors and correct 2 bits errors. What if $d_m(C) = 3$?



Example: Parity Code

- Definition

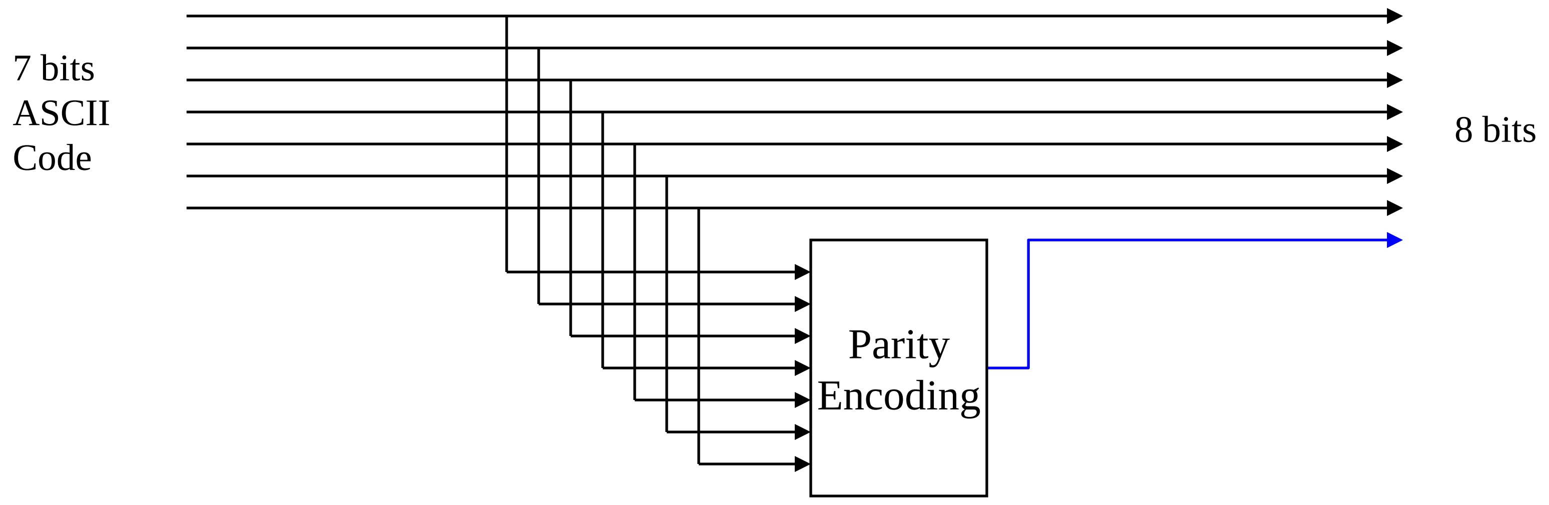
  $w = (b_0, b_1, \ldots, b_{n-1})$ is a word

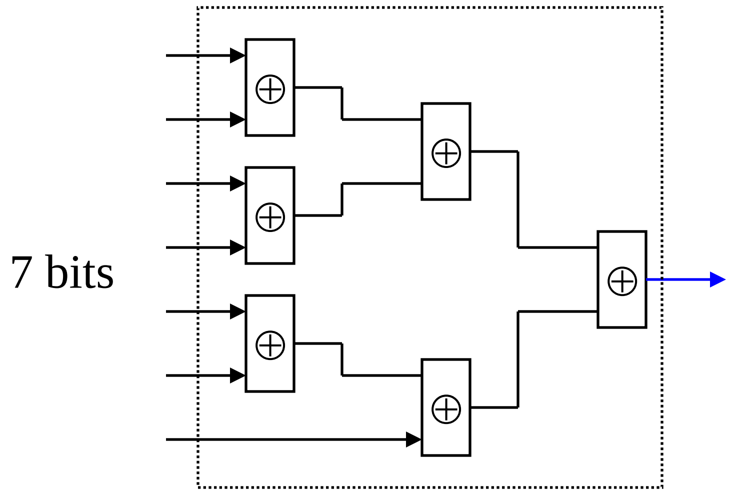
  $c = (b_0, b_1, \ldots, b_{n-1}, b_n)$ is a codeword, where

  $b_n = b_0 \oplus b_1 \oplus \ldots \oplus b_{n-1}$ (Even-parity code)

  $b_n = b_0 \oplus b_1 \oplus \ldots \oplus b_{n-1} \oplus 1$ (Odd-parity code)
- Merits:
- Hamming distance: 2 -- single error detecting code.
- Error coverage: The larger the n, the lower the detection coverage.
- Simplicity: Linear separable code
- Cost: one extra bit for any n, and log(n) level of encoding and decoding delay.

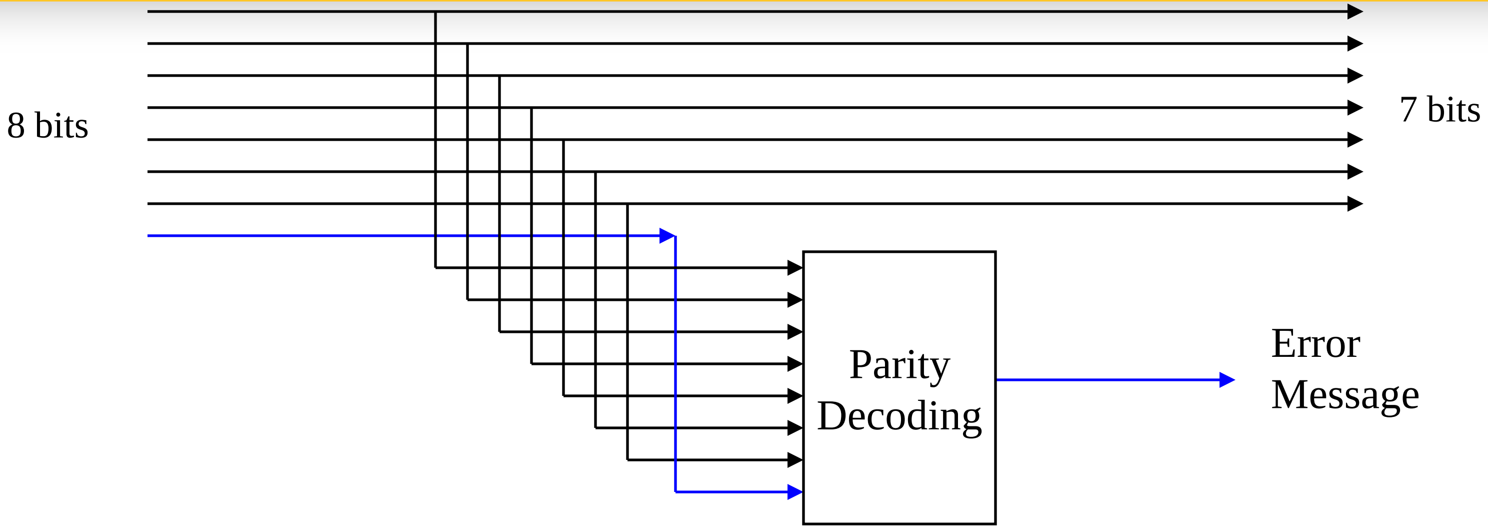
Parity Generation (Encoding)

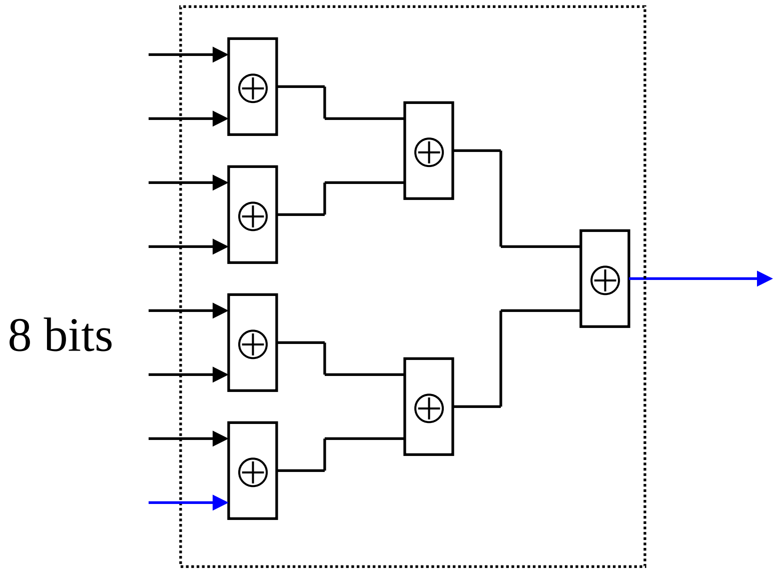




Encoder using XOR gates

Parity Checking (Decoding)





Decoder using XOR gates

Example: Checksum

Hashing is a variation of such scheme

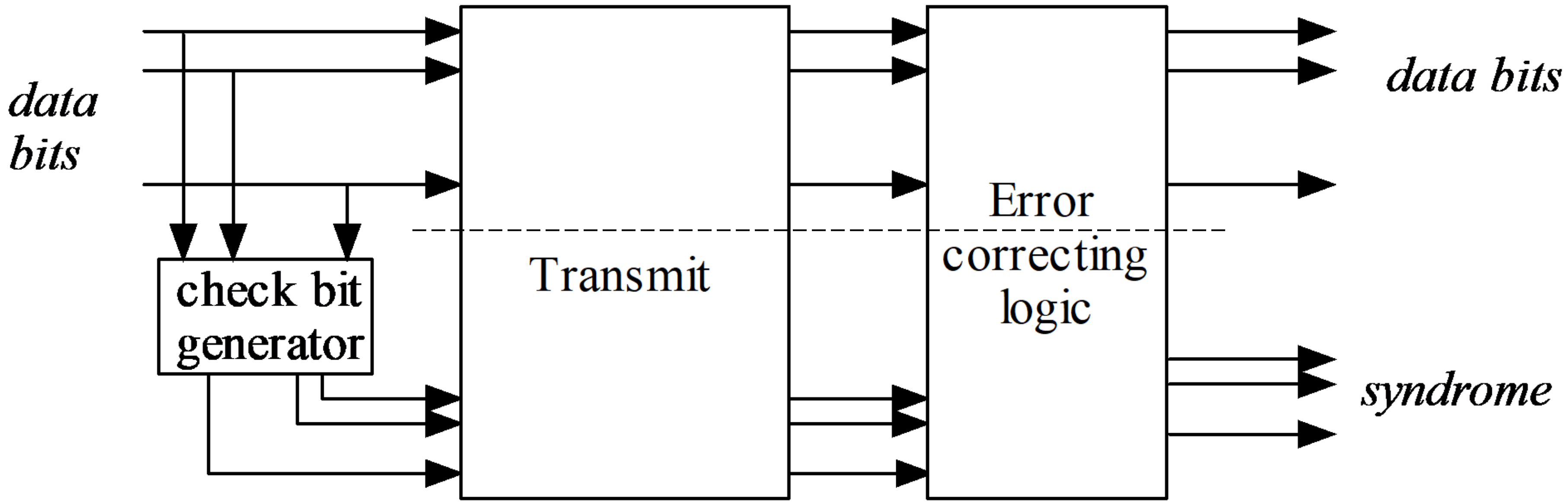
- Checksum is often used on a set of words.
- Let $(x_0, x_1, x_2, \ldots, x_{m-1}, x_m)$ be a vector of words of length m with an additional check word, where

$$x_m = \sum_{i=0}^{m-1} \bmod (2^n) .$$

- The last word x_m is the checksum for words $x_0, x_1, \ldots, x_{m-1}$.
- Sender first calculates the checksum x_m, and then transmits m + 1 codewords $(x_0, x_1, x_2, \ldots, x_{m-1}, x_m)$.
- Receiver recalculate the sum and checks whether the calculated sum is equal to the received checksum.
- Can be applied to blocks of consecutive words in memory or communication.
- Each write operation leads to recalculation of the checksum.
- Low cost and low diagnostic resolution.
- The upper bits (overflow) are ignored, and it may result in no detection of single error, depending on the word structure and the vector length (m).

Implementation of Separable Codes

- For Parity Check, logic operation XORs are used
- For Checksum, arithmetic operations additions are used
- For Hashing, hashing algorithm is used, which is much more complex.
- Separable codes are easy for encoding, decoding, and error detection.



Example: Arithmetic Codes

- Let A be a function. A defines an arithmetic code in word space W, if

$$(\forall a)(\forall b)(a \in W \wedge b \in W \Rightarrow A(a \otimes b) = A(a) \otimes A(b)))$$

where, $\otimes$ is one of arithmetic operations given, e.g., $+$, $-$, $*$.

- AN Code with A*N is an arithmetic code, where, $\otimes$ is *

The condition is met:

$$A*(a + b) = A*a + A*b \text{ and } A*(a - b) = A*a - A*b$$

- Characteristics:
  - Non-separable code
  - 2 bits redundancy for $A = 3$
  - Single fault detection for $A = 3$

Example: m-of-n Codes (m/n Code)

- An m/n codeword consists of n bits, in which exactly m bits must be 1's.
- Example: 2/3 codewords: (011, 101, 110); 1/n: (001, 010, 001)
- There are $\binom{n}{m}$ codewords in 2^n words. The usage is $\binom{n}{m}/2^n$.
- It contains a large amount of redundancy, and the cost is high
- For example, if n = 16 and m = 3, then

  $\binom{16}{3} = 3360$ and $2^{16} = 65536$

  About 5% words are used as codewords.
- m/n Code is a non-separable code
- Error coverage: It detects all single and all unidirectional errors: $0 \rightarrow 1$ or $1 \rightarrow 0$ errors, but not both types.

One-Hot Encoding for coding non-numerical items in machine learning

The 5th-Generation Wireless Systems: 5G

- 5G uses additional spectrum in the existing 4G frequency range.
- 5G can be up to 20 times faster than 4G, with a peak speed of 20 Gb/s. The fastest WiFi today is over 100 Gb/s.
- 5G is further taking the shares of landlines, not only for wireless devices, but also for internet.
- Challenges: Wireless has a higher error rates than landlines.
- 5G must control more errors while keeping data transmission efficiency!

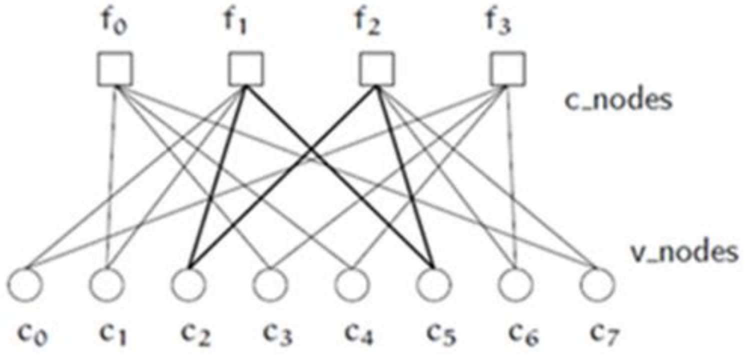
Wireless Channel Codes

- Turbo Codes: developed by Claude Berrou in 1991. The first public paper on turbo codes: "*Near Shannon Limit Error-correcting Coding and Decoding: Turbo-codes*." The codes offer high-performance forward error correction capacity and are widely used in 3G and 4G cell phone network.
- LDPC codes: invented by Gallager in his Ph.D. thesis at MIT in 1960. They were ignored for several decades as they were thought to be impractical. They emerge in 4G network, with similar performance to Turbo codes, but are simpler in implementation. Become 5G data channel uplink and downlink code standard.
- Polar Codes: introduced by Arikan in 2008. The latest major breakthrough in coding theory. Use linear block error correcting codes, which are constructed based on a multiple recursive concatenations of a short kernel codes. They are optimized Codes for Bitwise Multistage Decoding. 5G control channel code standard.

Example of LDPC and Polar Codes

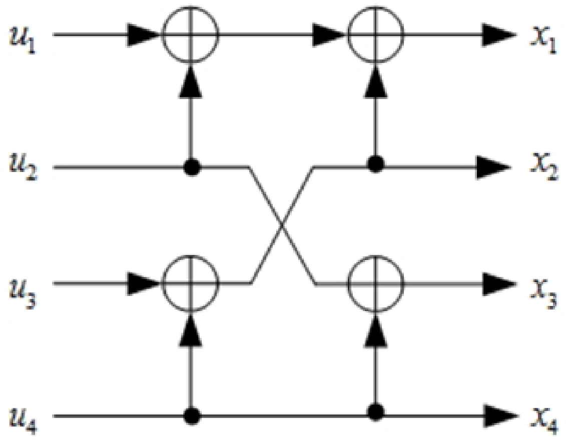
LDPC Codes: Non-separable

$\downarrow$ **Encoding** $\downarrow$

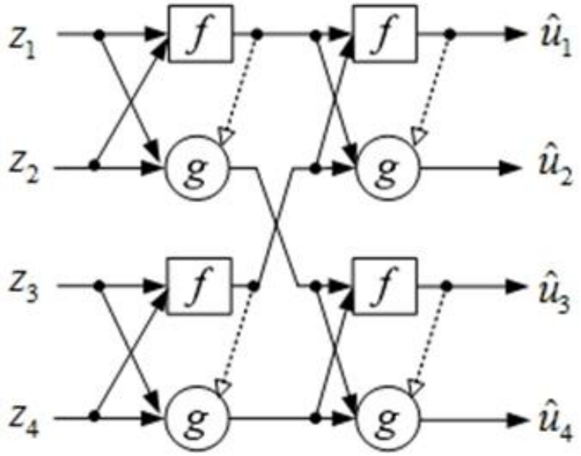


Polar Codes: Separable

$\longrightarrow$ **Encoding** $\longrightarrow$



$\longrightarrow$ **Decoding** $\longrightarrow$



Ira A. Fulton Schools of Engineering

Arizona State University

M14 L5
Secure Sockets Layer and HTTPS

Ira A. Fulton Schools of Engineering
Arizona State University

Lecture Outline

- SSL Protocol Stack
- Secure Sockets Layer in Web
- Create Your Own Certificate for Testing Purpose

Secure Sockets Layer in Web Application

- SSL over HTTP is the most common application of SSL: It is called HTTPS.
- HTTPS encrypts the data before sending to network.
- It decrypts data upon receiving data.
- To apply/purchase HTTPS for your Web server,
  1. Request/Purchase the security certificate from a certification authority; Analogy: Driver License.
  2. Install the certificate on the server, which consists of an open key and a secret key.
  3. The open key will be sent to the browser when connecting.
  4. A browser uses the open key to encrypt data and send encrypted data to the server. Server decrypts it using the secret key

SSL Protocol Stack

The SSL Protocol Stack is composed of two layers.

1. The first layer is the higher layer (above HTTP), used in the management of SSL exchanges. It consists of
   1) SSL Handshake Protocol,
   2) SSL Change Cipher Spec Protocol,
   3) SSL Alert Protocol, and
   4) HTTP.
2. The second layer is the lower layer (Below HTTP), consisting of
   1) SSL Record Protocol,
   2) TCP, and
   3) IP.

SSL Session and Connection States

- An SSL session is a connection between a client and server. Multiple sessions are possible between a client and server.
- An SSL connection is a transport that provides a type of service. Connections are peer-to-peer relationship.

Session State Used in SSL

The session state includes the following elements:

- **session identifier -** A byte sequence chosen by the server to identify an active or resumable session state.
- **peer certificate -** X509.v3[X509] certificate of the peer. This element of the state may be null.
- **compression method -** the algorithm used to compress data prior to encryption.

Session State Used in SSL (cont'd)

- **cipher spec** - Specifies the bulk data encryption algorithm (such as DES, etc.) and a MAC (Message Authentication Code) algorithm. It also defines cryptographic attributes such as the hash_size.
- **master secret -** 48-byte secret shared between the client and server.
- **is resumable -** A flag indicating whether the session can be used to initiate new connections.

Connection State

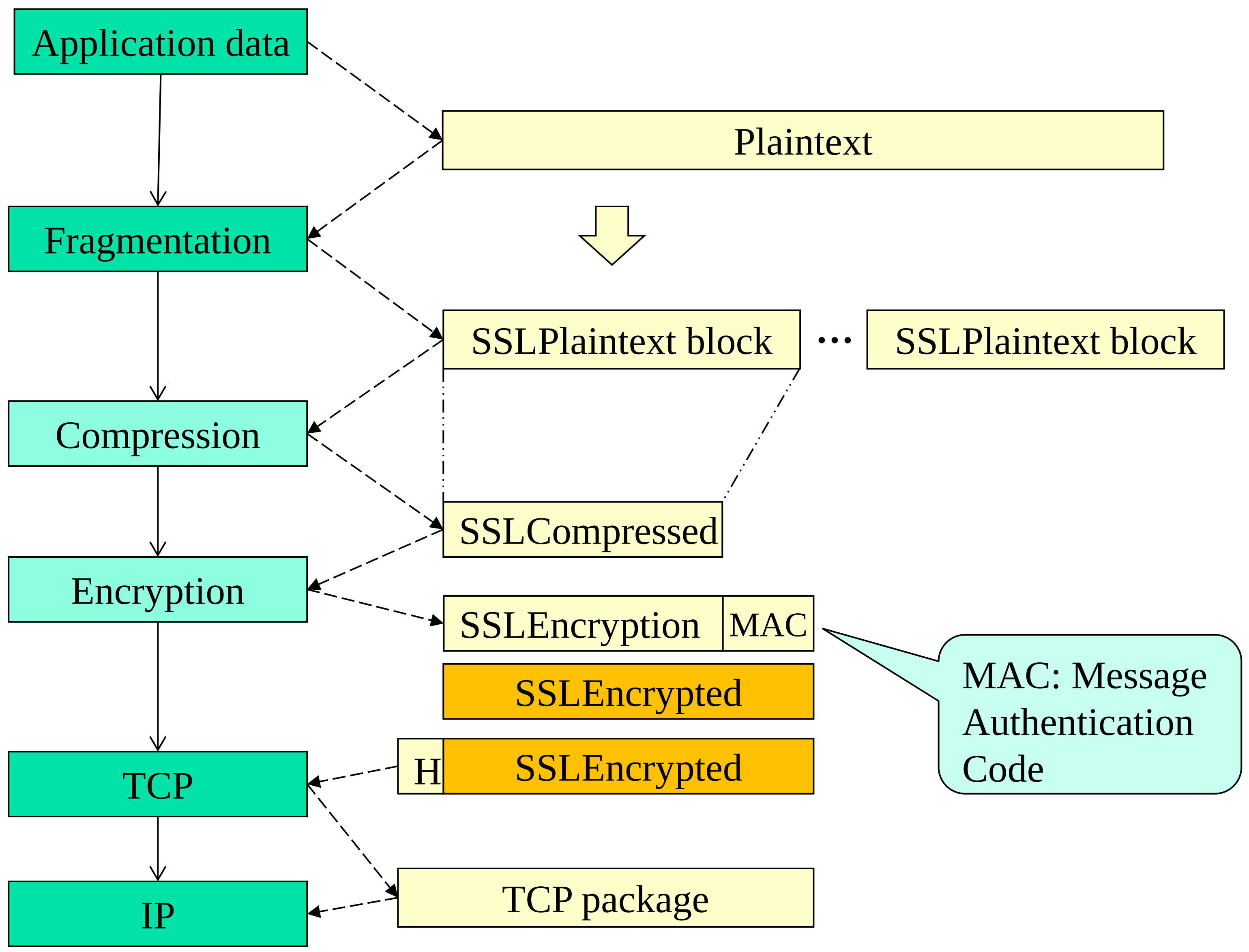
The connection state includes the following elements:

- **server and client randoms -** Byte sequences that are chosen by the server and client for each connection.
- **server write MAC secret -** The secret used in MAC operations on data written by the server.
- **client write MAC secret -**The secret used in MAC operations on data written by the client.
- **server write key -** The bulk cipher key for data encrypted by the server and decrypted by the client.

Connection State (cont'd)

- **client write key -** The bulk cipher key for data encrypted by the client and decrypted by the server.
- **initialization vectors -** When a block cipher in CBC (Cipher Block Chaining) mode is used, an initialization vector (IV) is maintained for each key.
- **sequence numbers** - Each party maintains separate sequence numbers for transmitted and received messages for each connection. When a party sends or receives a change cipher spec message, the appropriate sequence number is set to zero.

SSL Record Protocol below HTTP



SSL Record Protocol Steps

- **Fragmentation:** The record layer fragments information blocks into SSLPlaintext records of 2^{14} bytes or less. Management issue.
- **Compression:** All records are compressed using a compression algorithm. The compression algorithm translates an SSLPlaintext structure into an SSLCompressed structure. Efficiency issue.
- **Compute a MAC** – All records are protected using the encryption and MAC algorithms defined in the current CipherSpec. A shared secret key is used. Security issue.

SSL Record Protocol Steps (cont’d)

- **Null or standard stream cipher -** Stream ciphers convert SSLCompressed fragment structures to and from stream SSL Ciphertext
- **CBC** (Cipher Block Chaining) **block cipher -** For block ciphers (such as DES), the encryption and MAC functions convert SSLCompressed fragment structures to and from block SSLCiphertext fragment structures.
- **Record header**

Secure Sockets Layer in Web Application

- HTTPS is based on Secure Sockets Layer (SSL)
- To apply HTTPS to your Web server,
  1. Request/Purchase the security certificate from a certification authority;
  2. Install the certificate on the server, which consists of an open key and a secret key;
  3. Clients use https connection to access the server. https: SSL over HTTP is the most common application of SSL

Roles of SSL Certificate Authorities

- Issue a certificate showing the server uses the up-to-date encryption method;
- Make sure that the browsers recognize the certificates

A certificate of a server is similar to the driver’s license of a person.

- It certifies that the server uses the up-to-date encryption method and thus has certain level of trustworthiness in providing services (the communication part is secure).
- It does not guarantee the quality of services or the security within the server.
- Analogy: Driver’s license: the person can drive, but it does not tell the person has good driving habit. It does not tell the person has good credit score. Credit bureau or BBB.

SSL Certificate Offering Organizations

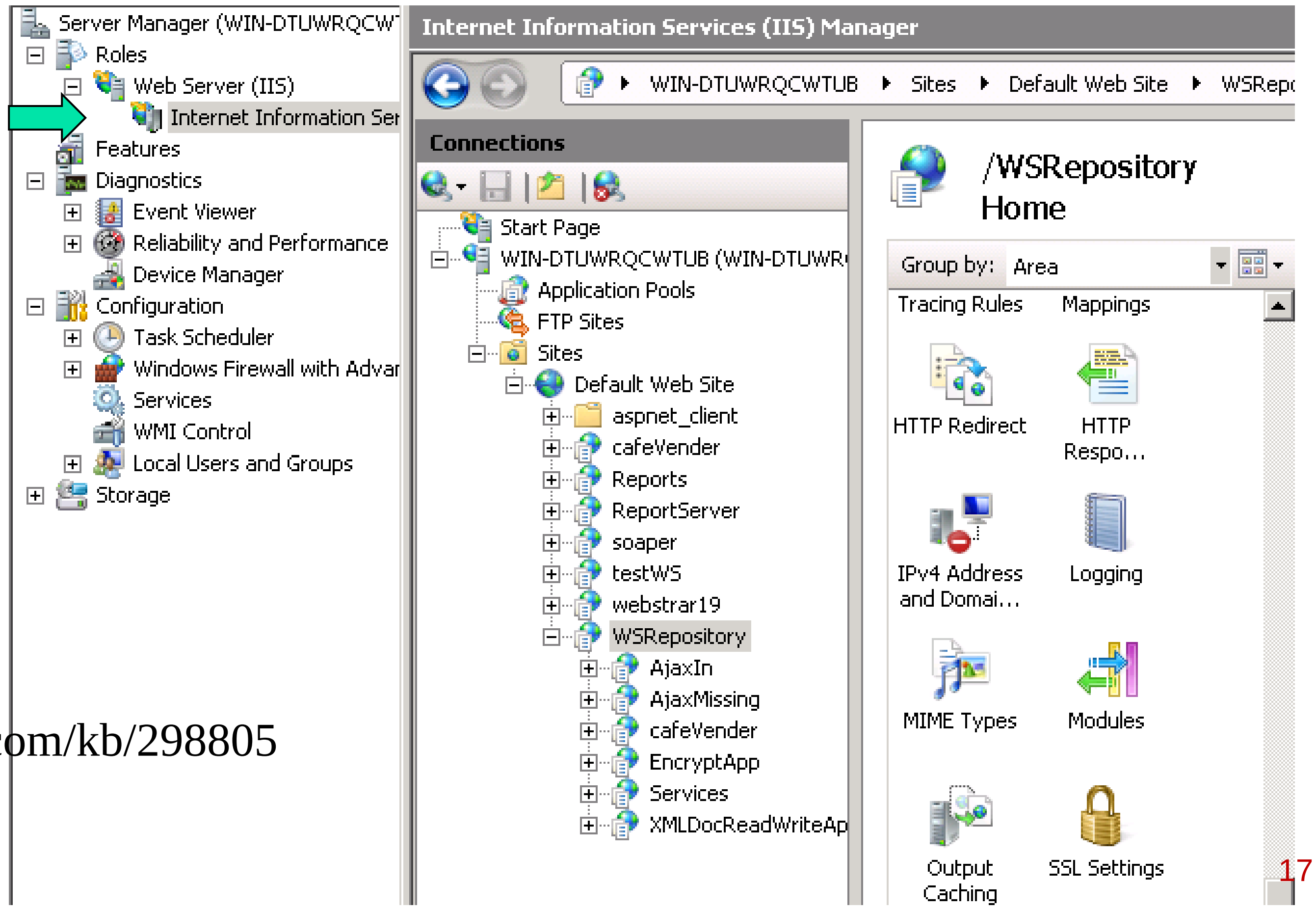
- Certificate is offered by independent organizations, such as
  - GeoTrust: http://www.geotrust.com
  - GlobalSign: http://www.globalsign.com
  - Thawte: http://www.thawte.com
  - VeriSign: http://www.verisign.com
- There are some organizations offer free or low-cost SSL certificates
  - StartCom: https://cert.startcom.org/ (http://en.wikipedia.org/wiki/StartCom)
  - Cacert: http://www.cacert.org/ (http://en.wikipedia.org/wiki/CAcert.org)

Create Your Own Certificate for Testing Purpose

- Before you purchase the SSL certificate, you can generate a certificate of your own for testing purpose;
- After you have completed your development and before you release to customer, you must purchase the certificate from authorized third party;
- There are many encryption methods. Some are not secure. Certification makes sure you use a secure one.
- It is unethical to use https without being certified: When users see https, they assume that you have been certified, and they use your website assuming that their confidential information will not be compromised in the communication.

Generating Certificate Signing Request (CSR) -1

- Access the IIS Microsoft Management Console (MMC): right-click **My Computer** and click **Manage**.
- This opens the Computer Management Console.
- Locate **IIS** and expand the IIS console.



Source: http://support.microsoft.com/kb/298805

Generating Certificate Signing Request (CSR) - 2

- Select the Default Web site in IIS on which you want to install a server certificate. Right-click the site and click Properties.
- Click the Directory Security tab. In the Secure Communications section, click Server Certificate. This starts the Web Server Certificate Wizard. Click Next.
- Select Create a New Certificate and click Next.
- Select Prepare the request now, but send it later and click Next.
- In the Name field, enter a name that you can remember. It will default to the name of the Web site for which you are generating the CSR.
- When you generate the CSR, you need to specify a bit length. The bit length of the encryption key determines the strength of the encrypted certificate. Most third-party CAs prefer a minimum of 1024 bits.

Install Your own Certificate for your Server

1. For testing purpose, you could create a certificate for your machine, run the Setup.bat that is included in each of the samples that use secure communication with IIS.
2. Ensure that the path includes the folder that contains Makecert.exe before you run this batch file. The command used to create the certificate in Setup.bat is:

```
makecert -sr LocalMachine -ss My
-n CN=ServiceModelSamples-HTTPS-Server
-sky exchange -sk ServiceModelSamples-HTTPS-Key
```

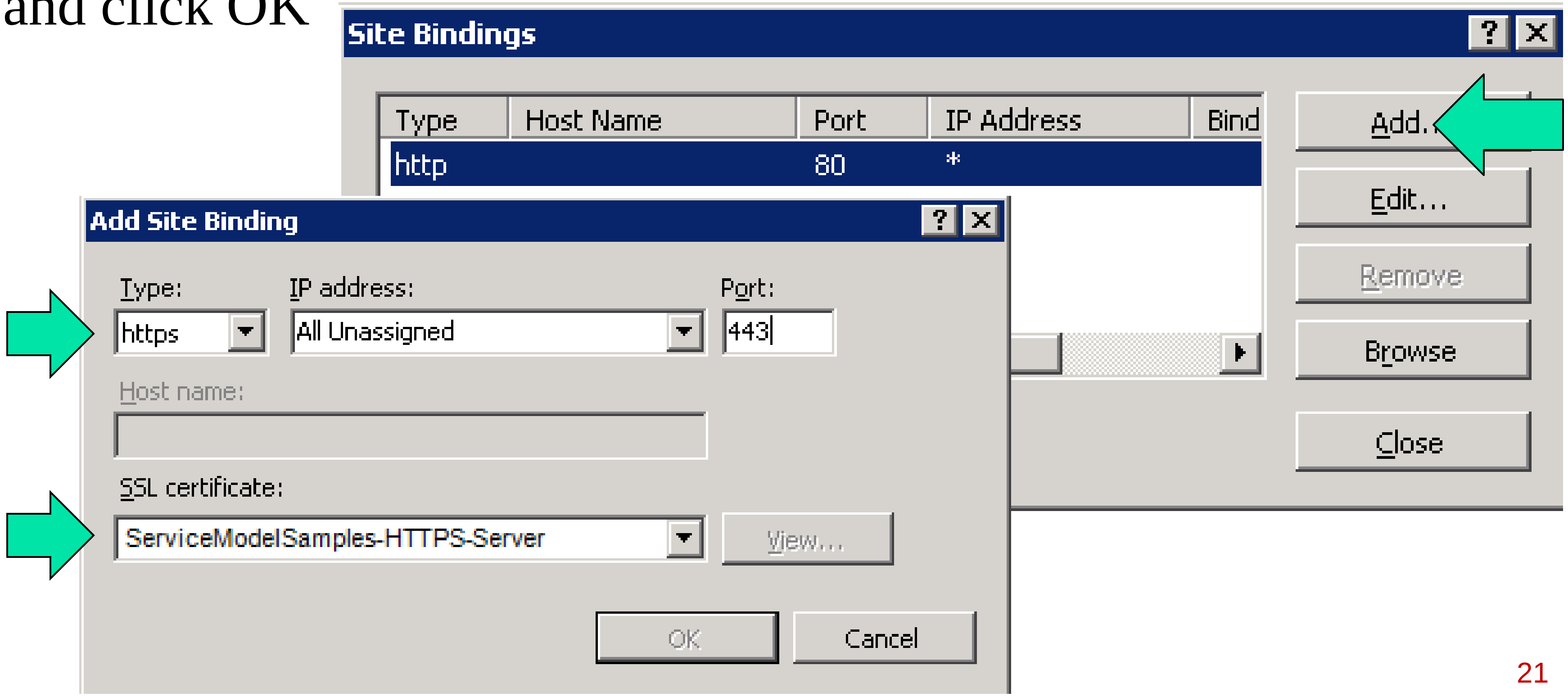
Source: http://msdn.microsoft.com/en-us/library/ms751408(VS.85).aspx

Installing Certificates in Windows Server

- Open the Internet Information Services Manager.
- Right-click the default Web site and select **Properties**.
- Select the **Directory Security** tab.
- Click the **Server Certificate** button. The Web Server Certificate Wizard starts.
- Complete the wizard. Select the option to assign a certificate. Select the ServiceModelSamples-HTTPS-Server certificate from the list of certificates that are displayed.

Installing Certificates in Windows Server (contd.)

- Open the Internet Information Services Manager.
- Right-click the Default Web Site and select Add… or Edit…
- Select HTTPS from the Type drop-down list.
- Select the certificate from the SSL certificate drop-down list and click OK



Site Bindings

Type	Host Name	Port	IP Address	Bind
http		80	*	

Add Site Binding

Type: https
IP address: All Unassigned
Port: 443
Host name:
SSL certificate: ServiceModelSamples-HTTPS-Server

Override Certificate Validation Callback

- WCF checks whether a certificate has been signed by a third-party authority.
- In the test phase using unsigned certificate, you can ignore the warning before every WCF call:

warning

```
//Overriding the certificate validation callback

System.Net.ServicePointManager.ServerCertificateValidationCallback += (se, cert, chain, sslerror) =>
{
                return true;
};
```

Overriding certificate validation callback is for testing purpose only. It is unethical and can be a crime to let user believe that your sever has SSL certificate, but it does not.

Testing Your HTTPS Connection

- Test access to the service in a browser by using the HTTPS address
  https://localhost/secureConnectionServices/service1.svc

Indicating the message from browser to server will be encrypted

Testing Client’s Connection

```
Protected void Page_Load(Object sender, EventArgs e)
{
    if (request.IsSecureConnection)
    {
          lblStatus.Test = “This page is connected through SSL”;
    }
    else
    {
           lblStatus.Test = “Please use https connection”;
    }
}
```

Ira A. Fulton Schools of Engineering

Arizona State University

Summary and Outlook

Ira A. Fulton Schools of Engineering
Arizona State University

Lecture Outline

Course Summary

- From CSE445 (SOC-I) to CSE446 (SOC-II)

CSE446 Outlook

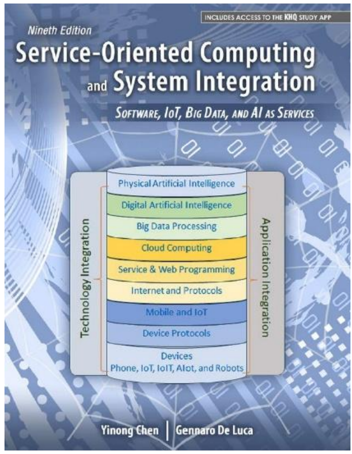
- Self-Hosting Services
- Oracle’ SOA Suite for BPEL Applications
- Workflow Foundation
- Message-Based Integration and ESB
- Database and Language Integrated Query
- Internet of Things and Robot as a Service
- Big Data Processing, AI and Cloud Computing

From CSE445 (SOC-I) to CSE446 (SOC-II)

- **CSE445/598**
  - Language-based SOC software development;
  - Must have strong CS background and object-oriented programming skill;
  - Largely bottom-up approach
- **CSE446/598: Emphasis on software integration**
  - Less emphasis on language-based development
  - Workflow/Visual composition at component level
  - Resource-oriented (REST) and data-driven approaches
  - Cutting edge: Database interfacing, big data, AI, machine learning, ontology and, cloud computing

One Text for Two Courses

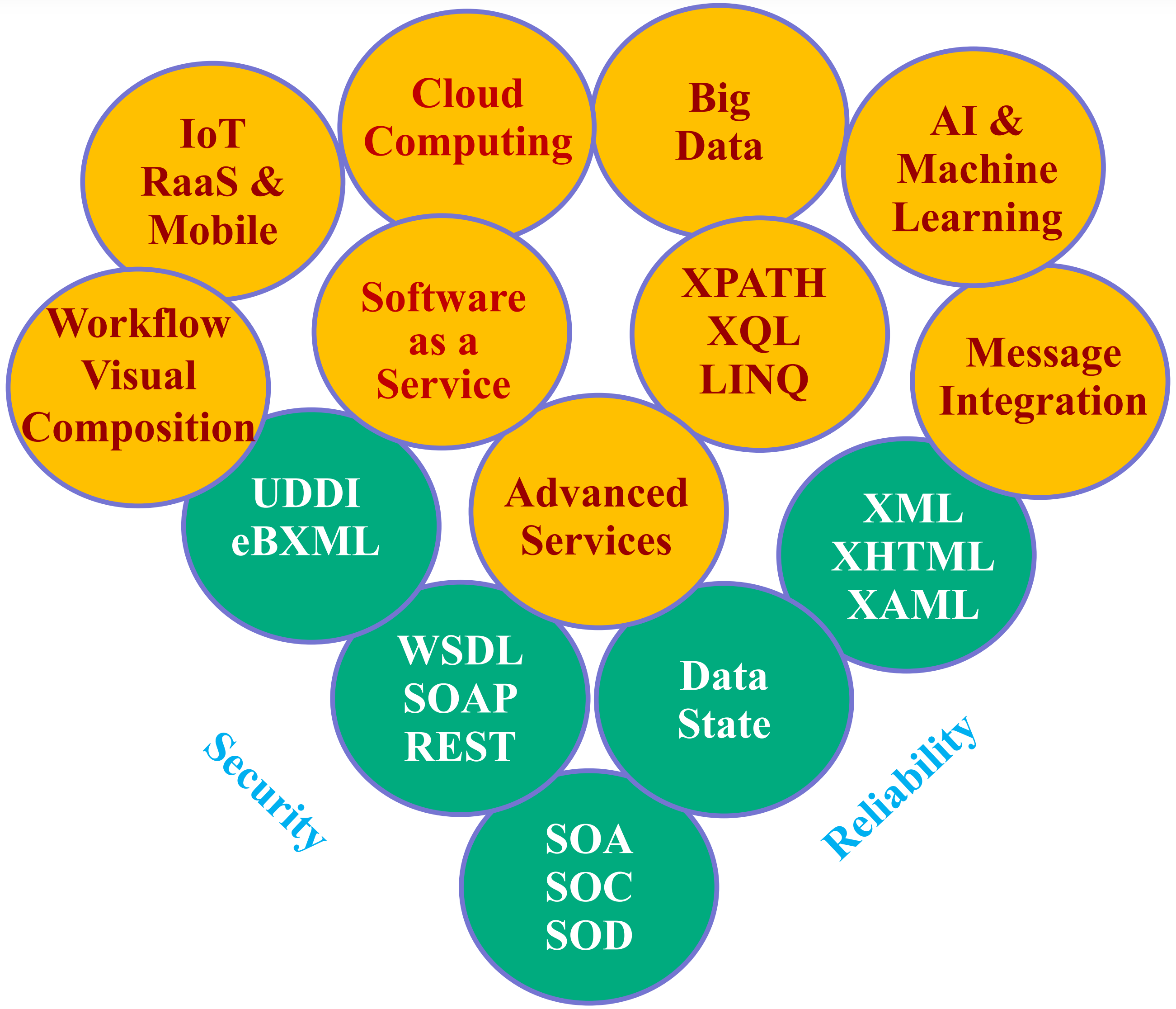
CSE445	Part I	Distributed Service-Oriented Software Development and Web Data Management
	Chapter 1	Introduction to Distributed Service-Oriented Computing
	Chapter 2	Distributed Computing with Multithreading
	Chapter 3	Essentials in Service-Oriented Software Development
	Chapter 4	XML and Web Data Formats
	Chapter 5	Web Application and State Management
	Chapter 6	Dependability of Service-Oriented Software
CSE446	Part II	Advanced Service-Oriented Computing and System Integration
	Chapter 7	Advanced Services and Architecture-Driven Application Development
	Chapter 8	Enterprise Software Development and Integration
	Chapter 9	IoT, Robotics, and Device Integration via Visual Programming
	Chapter 10	Interfacing Service-Oriented Software with Databases
	Chapter 11	Big Data Processing and Cloud Computing
	Chapter 12	Artificial Intelligence and Machine Learning
	Chapter 13	Mobile Computing and Application Development


INCLUDES ACCESS TO THE KHQ STUDY APP
Nineth Edition
Service-Oriented Computing and System Integration
Software, IoT, Big Data, and AI as Services
Physical Artificial Intelligence
Digital Artificial Intelligence
Big Data Processing
Cloud Computing
Service & Web Programming
Internet and Protocols
Mobile and IoT
Device Protocols
Devices
Phone, IoT, IoIT, AIoT, and Robots
Technology Integration
Application Integration
Yinong Chen
Gennaro De Luca

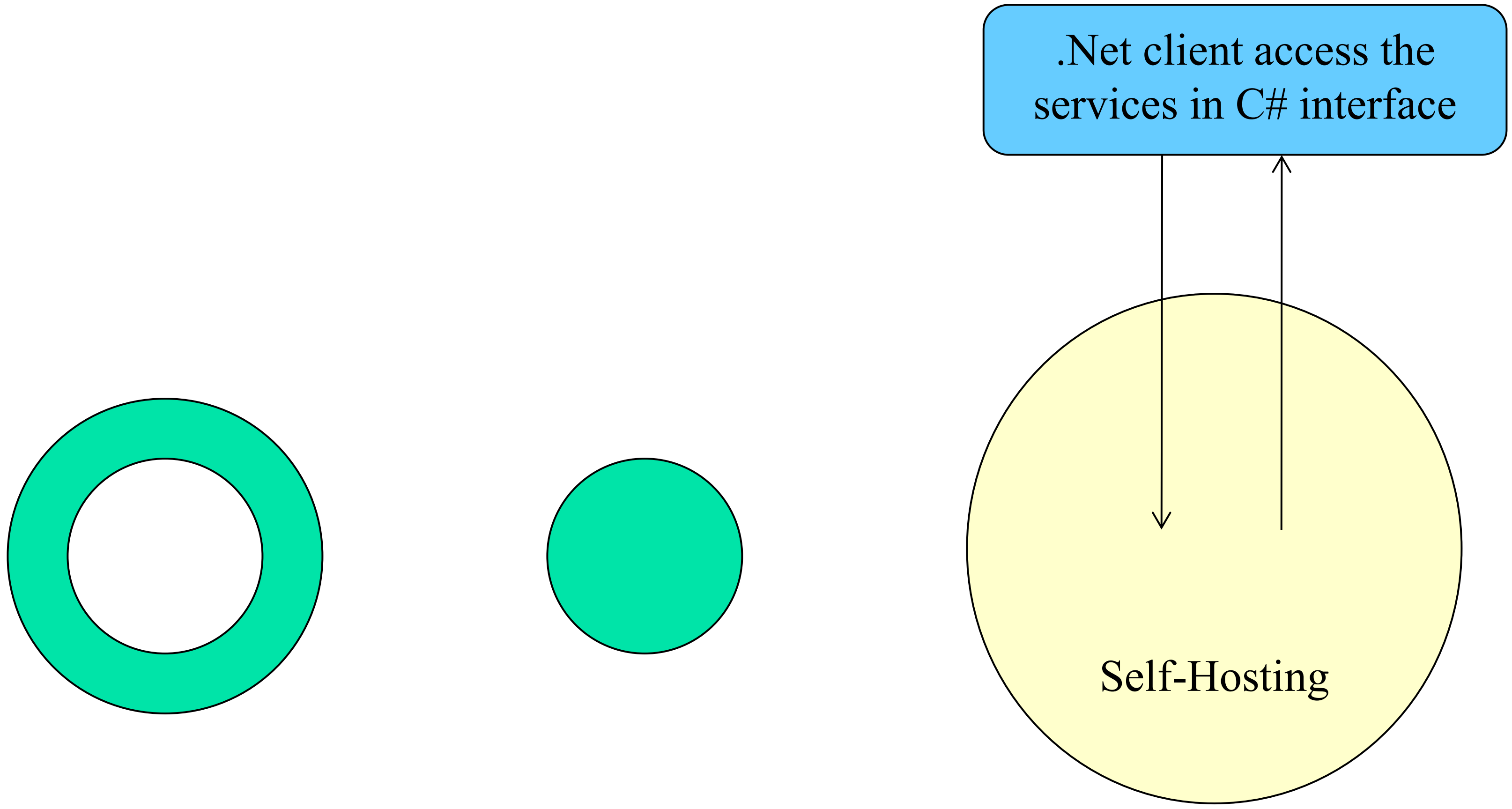
Service-Oriented Computing & Web Software Integration

CSE446 Outlook

CSE445



Creating a Platform-Dependent Service and Self-Hosting Service



Define an interface with method signatures

Define a class as a service with methods

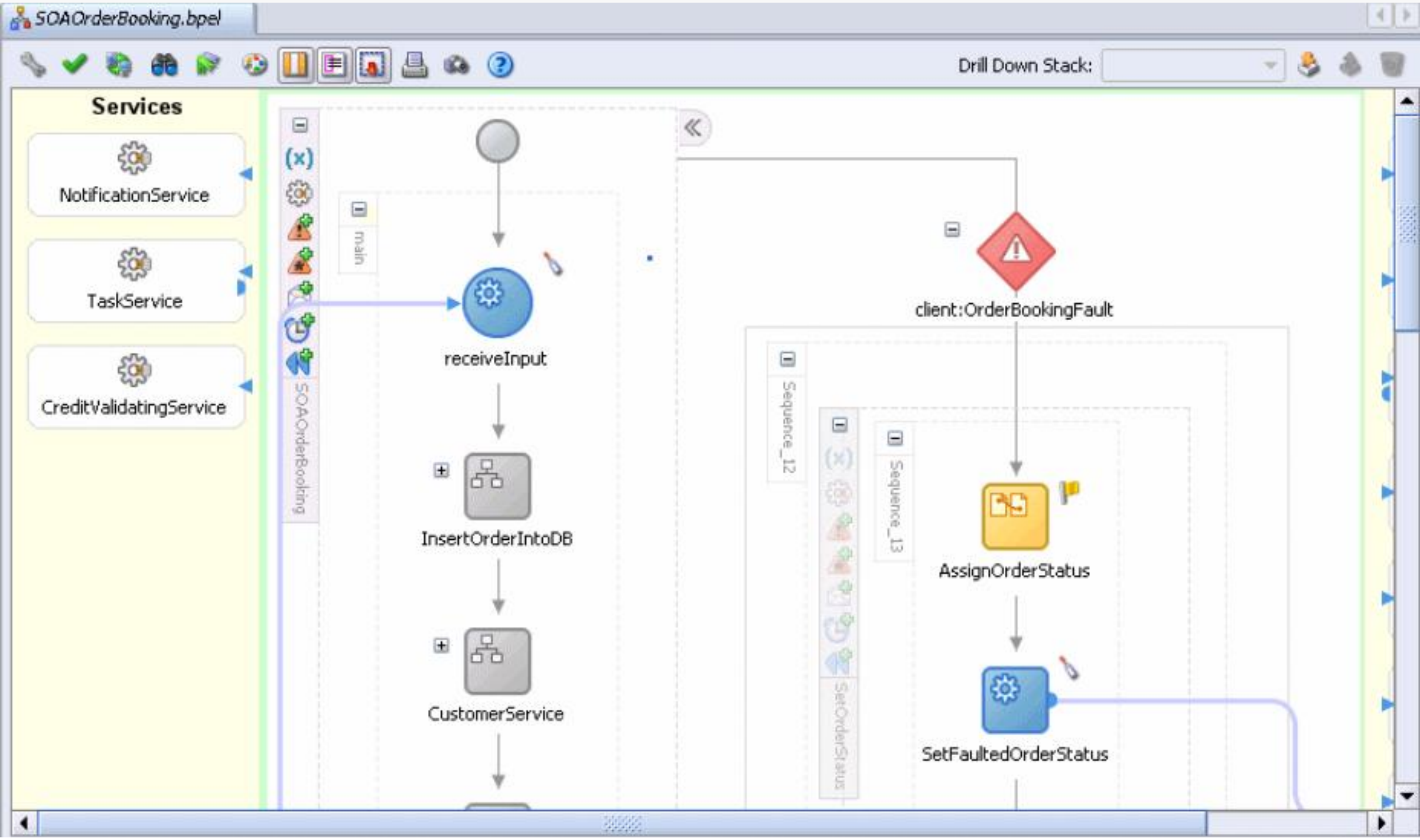
Define a .Net application class to host the service in console application template

eBusiness and Technology Interaction

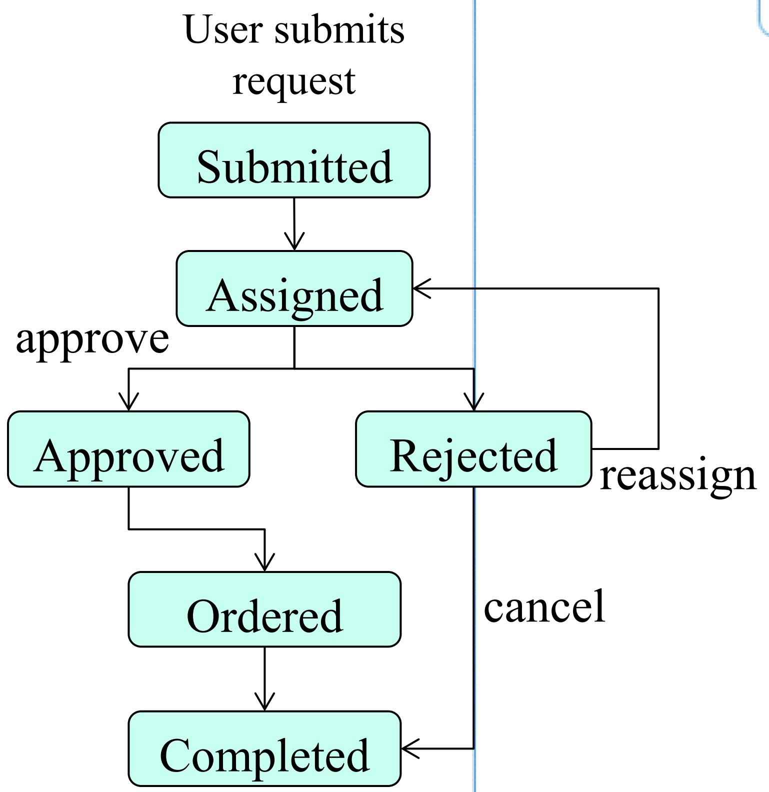
e-business	Computerized establishment of e-business with basic information exchange with clients	Maintain a multi page Web site; Use Web-form and Web-email for data exchange with clients	Core business on Web with database and Web-enabled execution of business process and transaction.	Enterprise level collaboration & integration, Dynamic business composition, EAI, BPM	User-provided contents, Dynamic discovery, Social networking, Social integration	*On Demand e-business with AI-based real-time reconfigur-ation, resilient, & automated evolution*
Technology	Early stage of Internet: Email FTP	Static Web 1.0 with HTML forms and web mail support	Web-based computing model and Web database integration	Web 2.0, WSDL, Web 3.0, SOA, SOC, SOD, ESB, Dynamic application architecture	Web 4.0*, HTML5, REST services, Cloud computing, Big Data, Social computing and networking platforms	*Web 5.0* with proactive artificial intelligent services and applications*
	80s	90s	00-04	05-09	10-19	20-

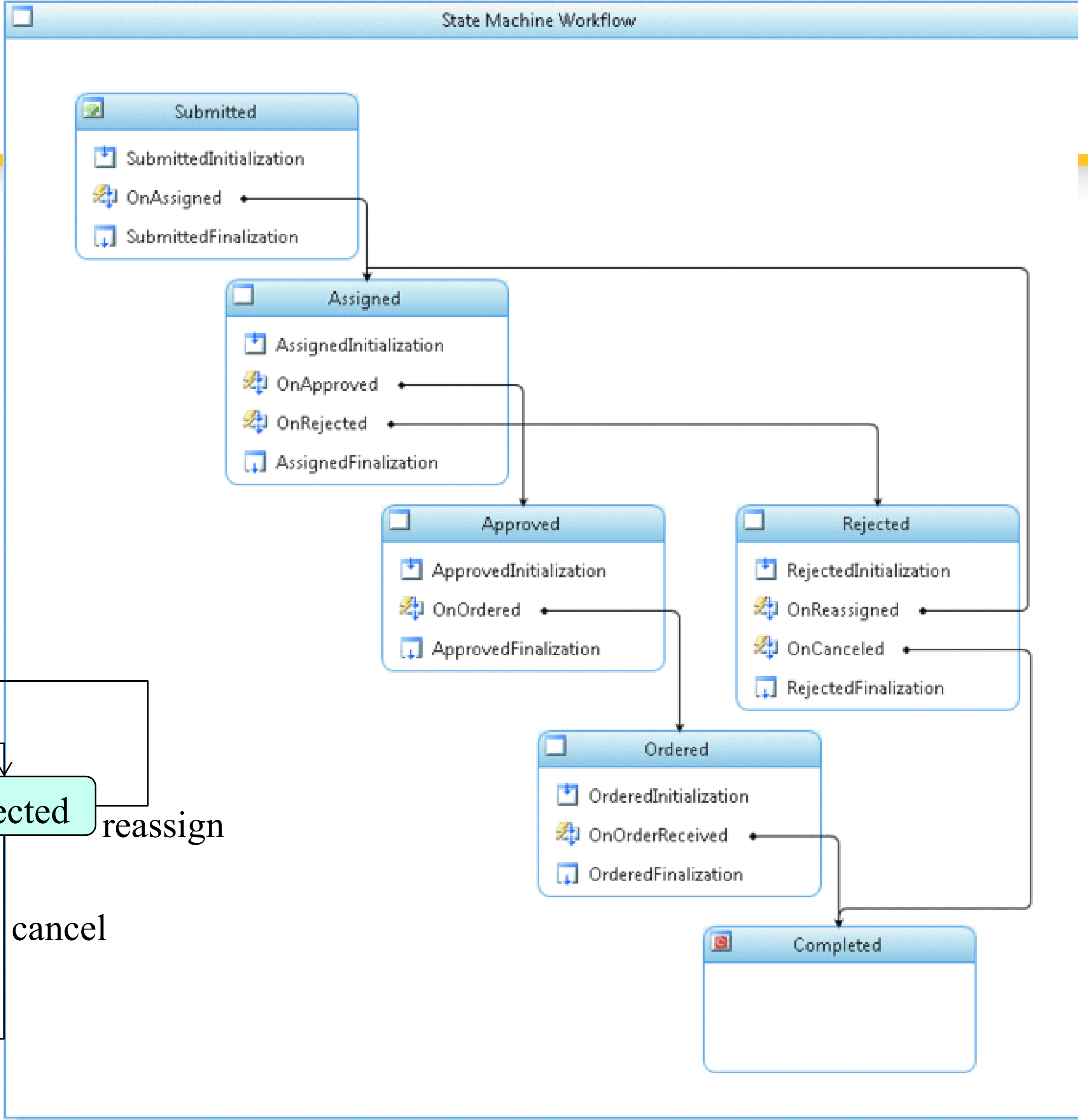
* From technology perspective
From user's perspective: Web 3.0

Architecture-Driven Development: Oracle' SOA Suite for BPEL Applications

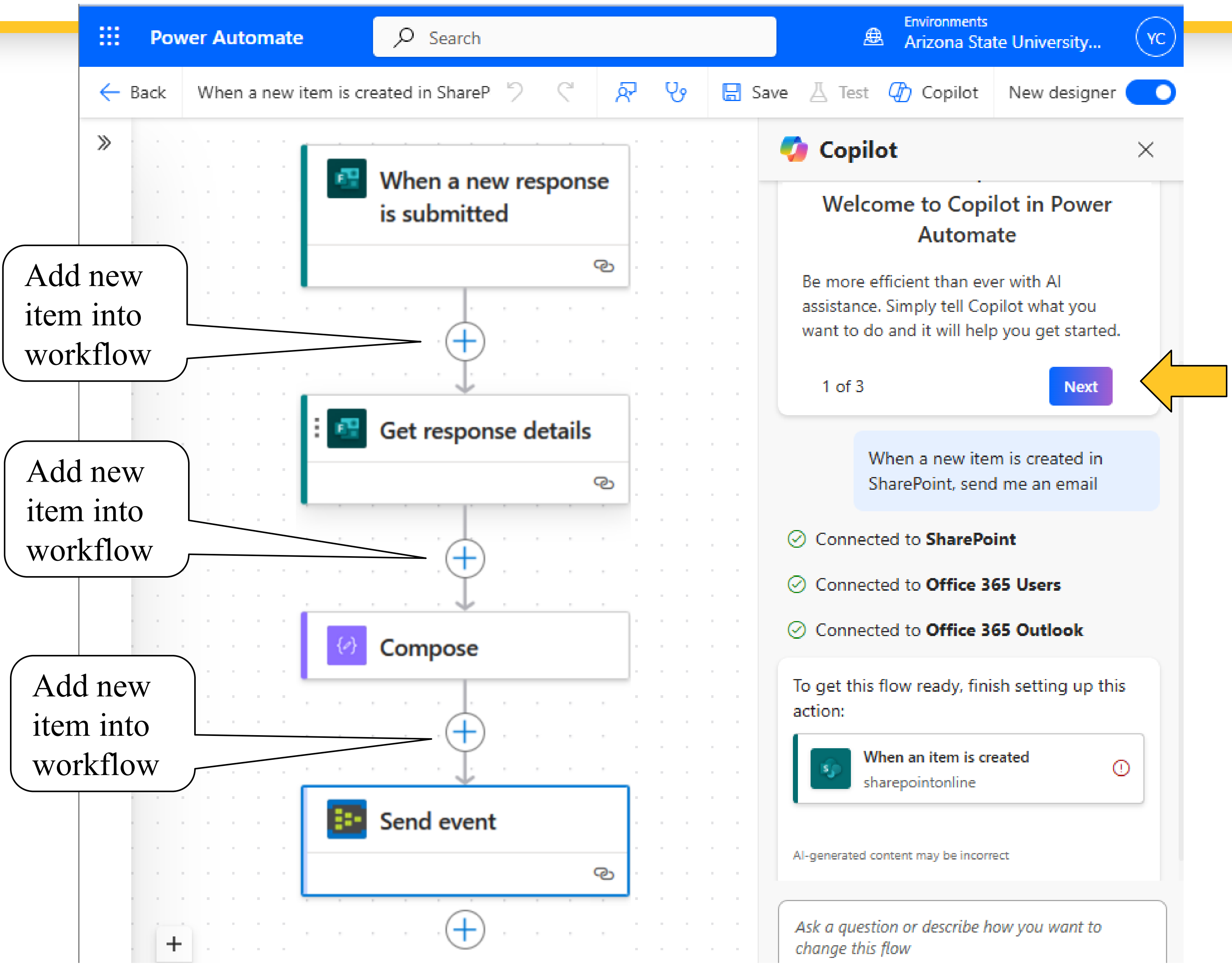


Workflow Foundation Implementing flowchart





Copilot-Based Workflow Generation



Power Automate
Search
Environments
Arizona State University...
YC
Back
When a new item is created in ShareP
Save
Test
Copilot
New designer
When a new response is submitted
Get response details
Compose
Send event
Add new item into workflow
Add new item into workflow
Add new item into workflow
Copilot
Next
1 of 3

Welcome to Copilot in Power Automate

Be more efficient than ever with AI assistance. Simply tell Copilot what you want to do and it will help you get started.

When a new item is created in SharePoint, send me an email

Connected to **SharePoint**

Connected to **Office 365 Users**

Connected to **Office 365 Outlook**

To get this flow ready, finish setting up this action:

When an item is created
sharepointonline

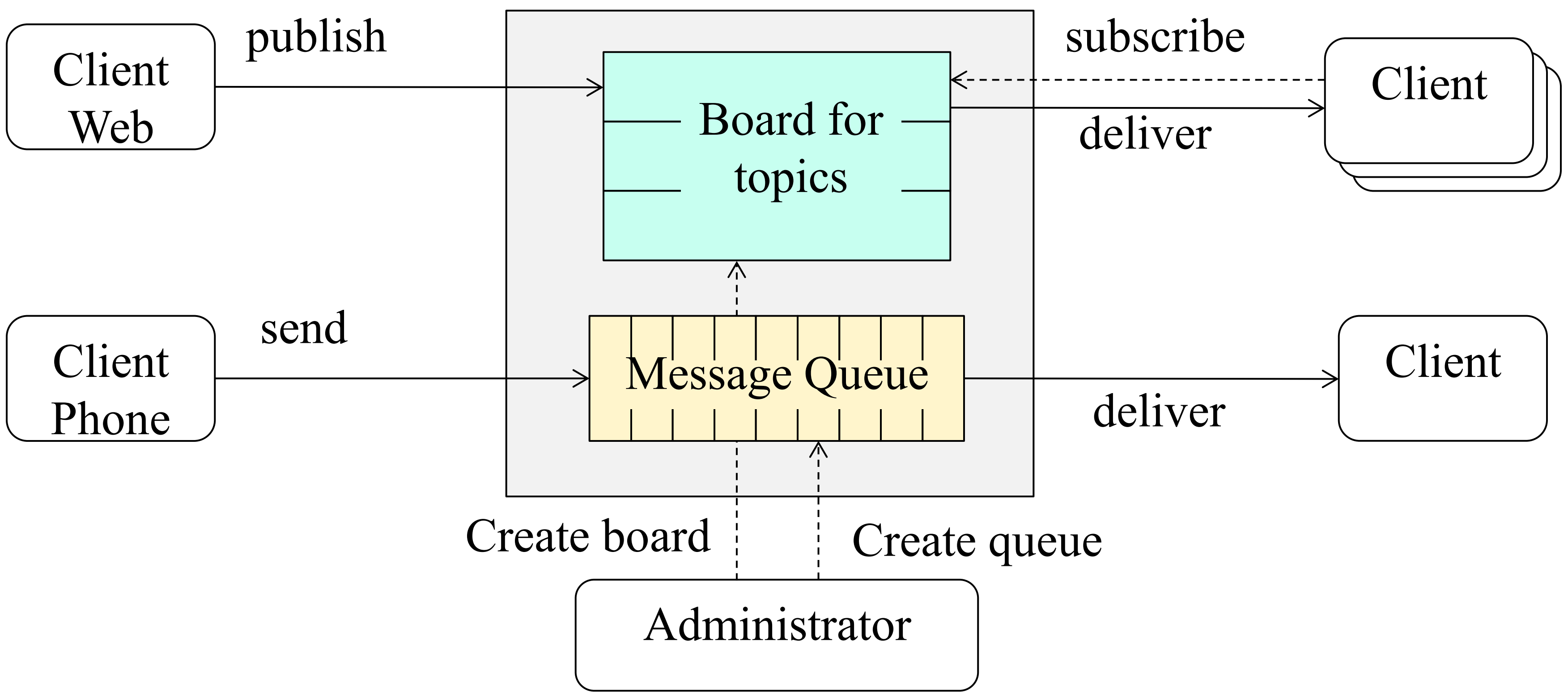
AI-generated content may be incorrect

Ask a question or describe how you want to change this flow

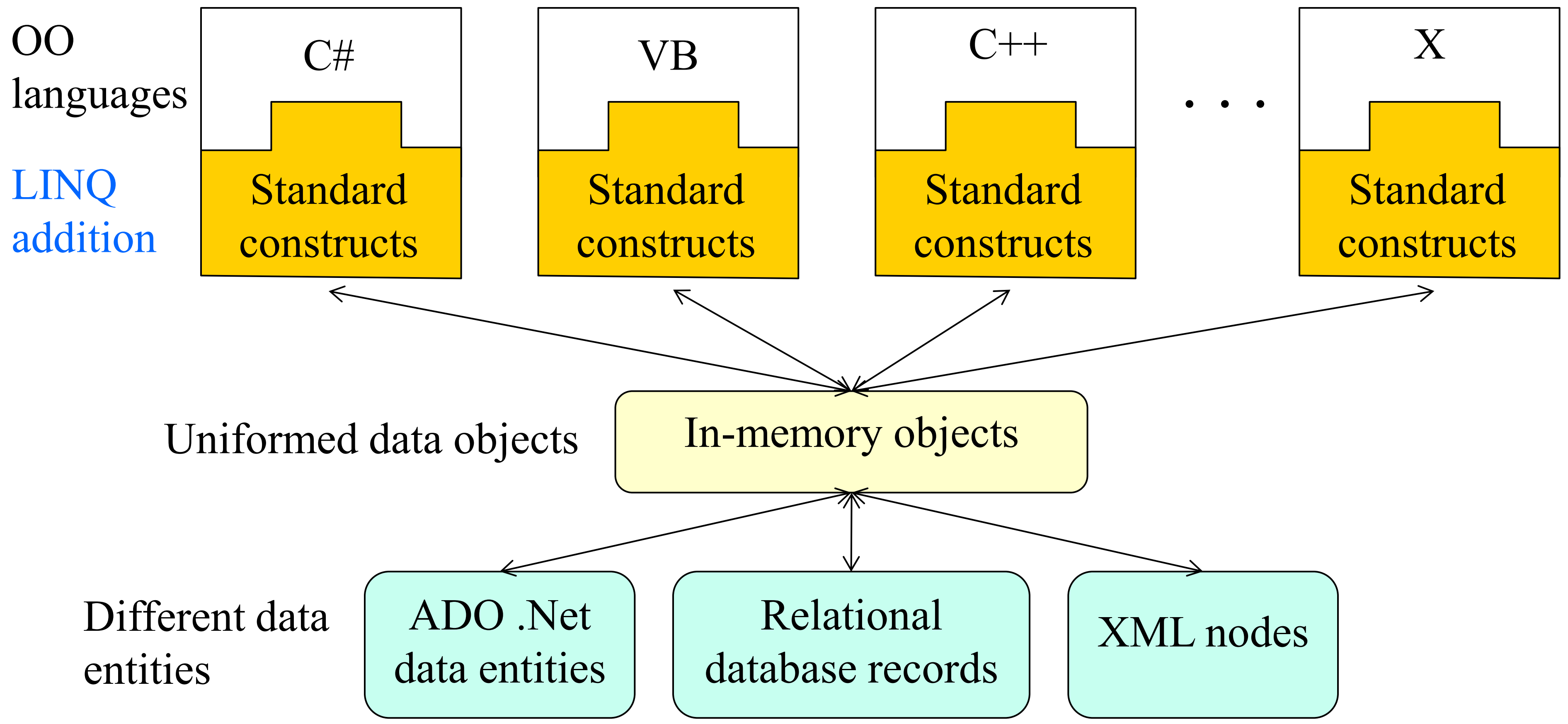
Message-Based Integration and ESB

https://venus.sod.asu.edu/WSRepository/Services/Messenger/Service.svc

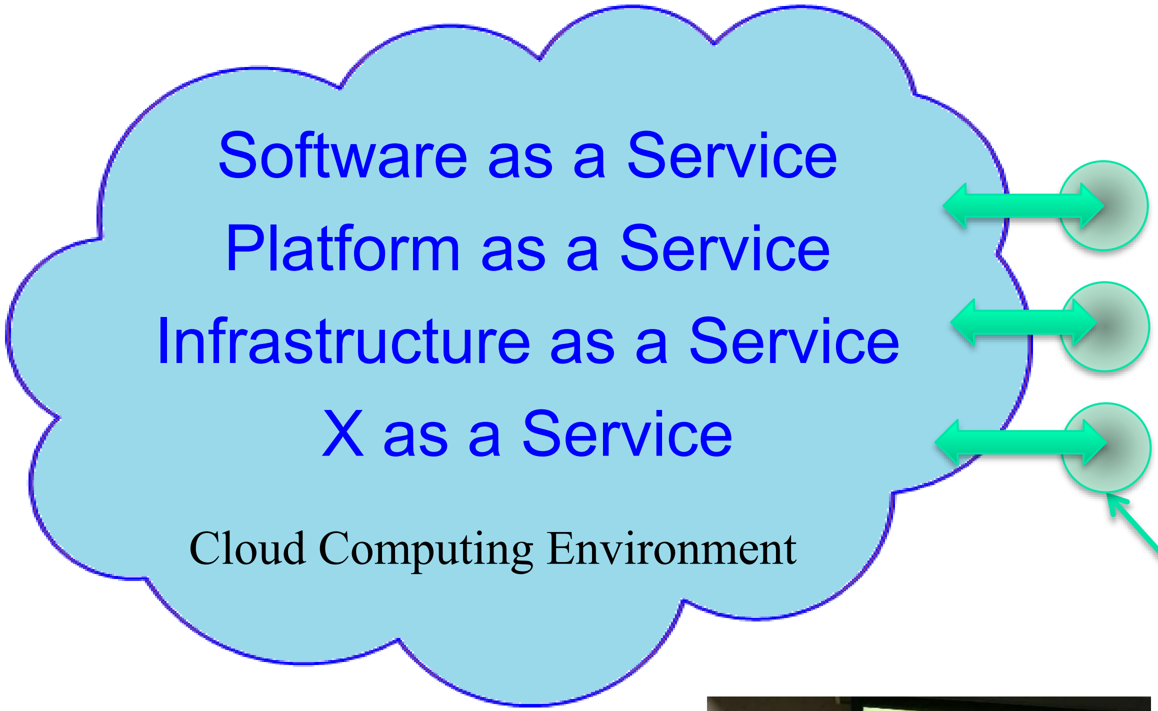
- Message Board: One-to-Many with subscription
- Message Queue: One-to-One with off-line delivery

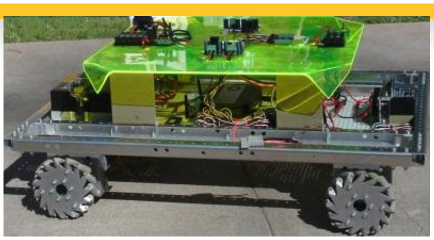

Messaging services

LINQ: Language Integrated Query



Internet of Things and Robot as a Service



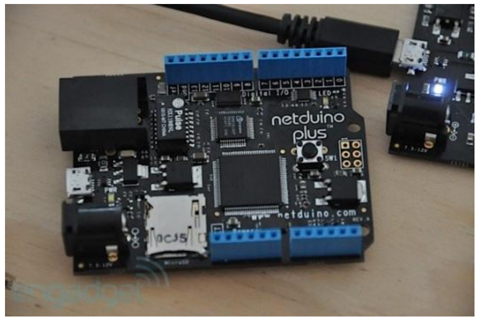


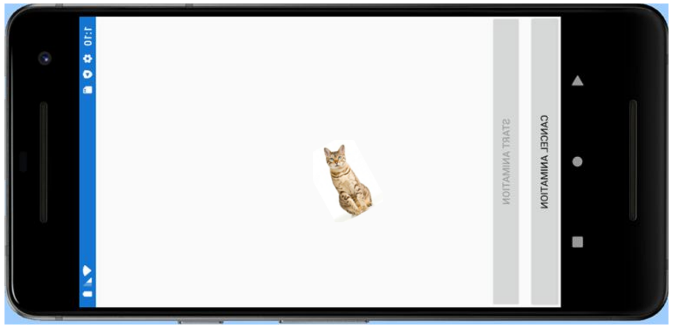
Robot as a Service

Cyber Physical Devices

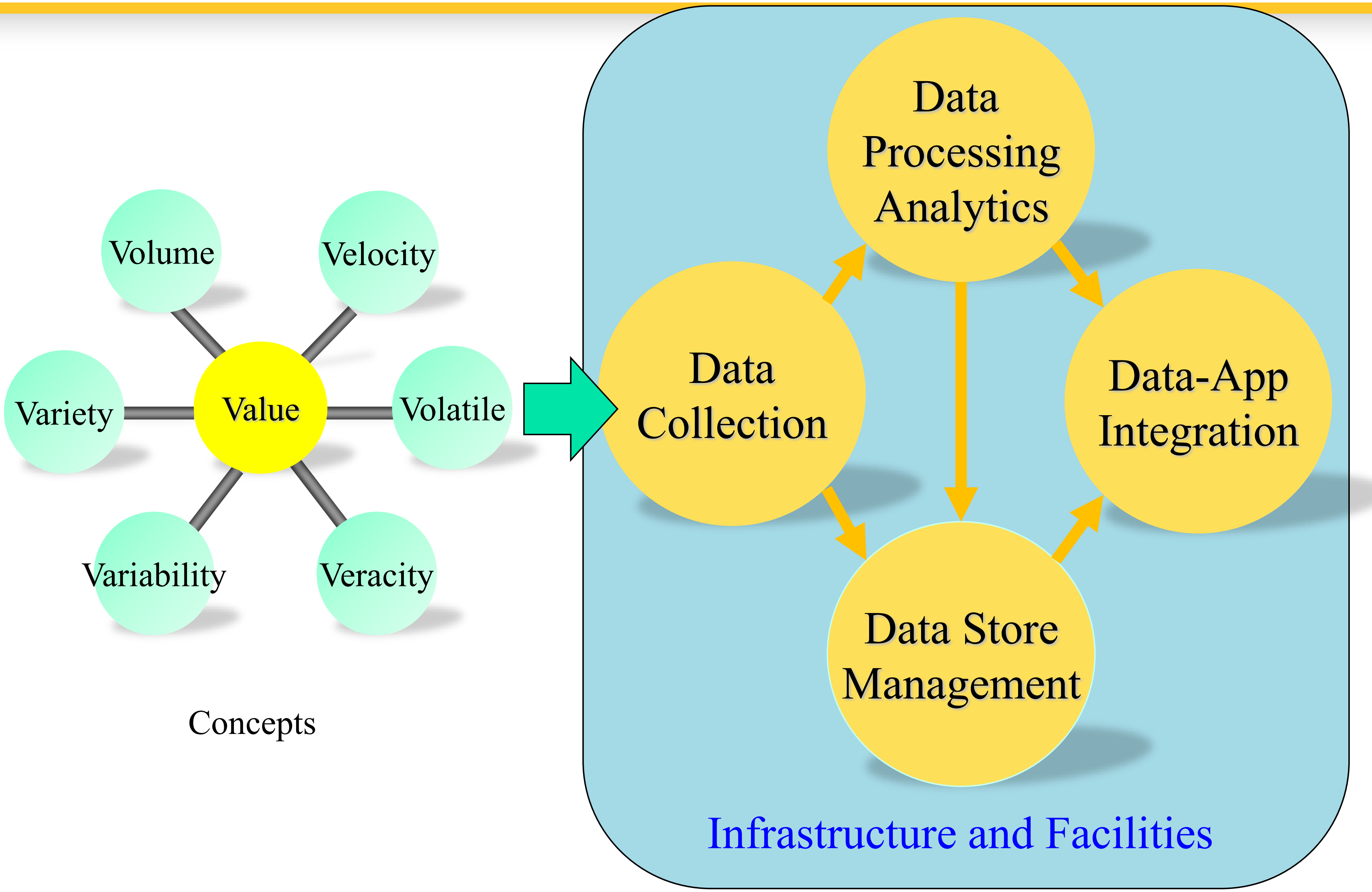
Device as a Service

Service Interface in HTTP, URI, REST. WSDL, SOAP, etc.

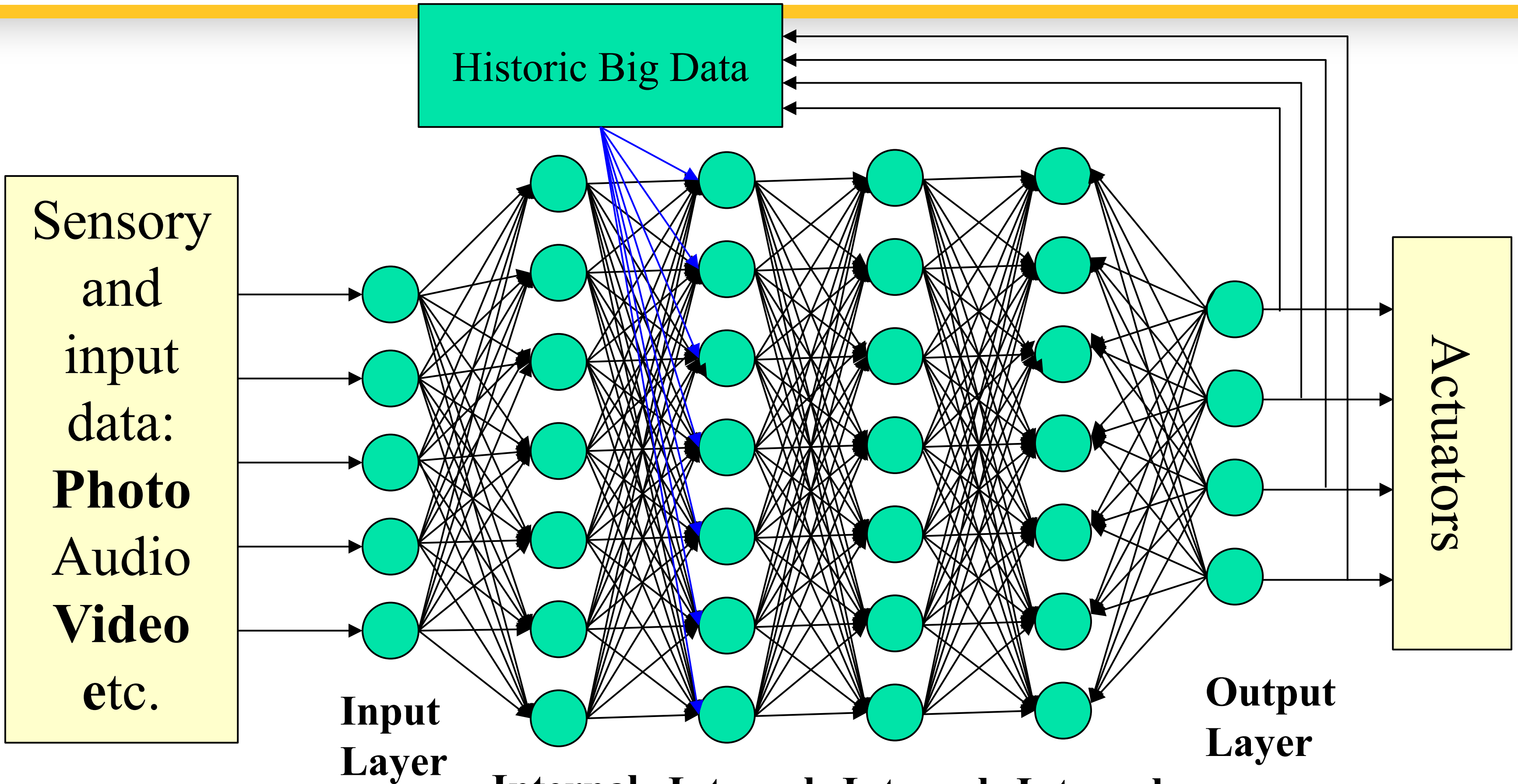




From Big Data Concepts to Domains of Study

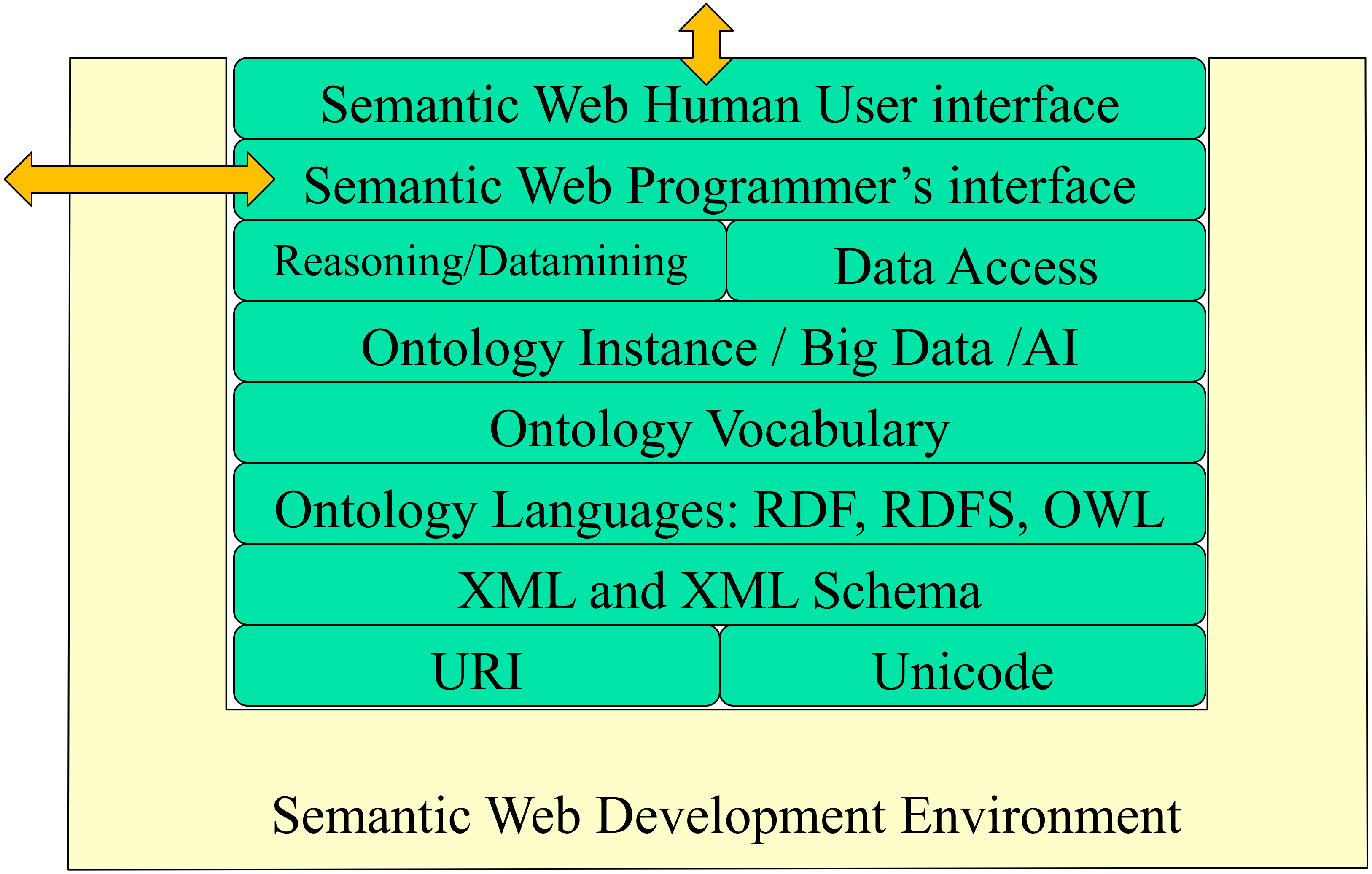


AI: Deep-Learning and Big Data Processing



Input data
Map
Neural cell processing
Reduce, concluding
Multiple layers of individual processing, matching

Semantic Web Language Stack


Semantic Web Human User interface
Semantic Web Programmer's interface
Reasoning/Datamining
Data Access
Ontology Instance / Big Data /AI
Ontology Vocabulary
Ontology Languages: RDF, RDFS, OWL
XML and XML Schema
URI
Unicode
Semantic Web Development Environment

Cloud Environments to be Discussed

- Google Cloud: Free account with limited resources; Java and Python based; Supported by Google APIs and services;
- Microsoft Windows Azure: C# and .Net based; Supported by Microsoft CRM (Customer Relationship Management), ERP (Enterprise Resource Planning), and Web services;
- Oracle Exalogic Elastic Cloud in a Box: From hardware to software in a box. Designed to provide the an environment for enterprise Java applications and Java-based infrastructure; It scales horizontally with no degradation of system performance as the size of the cloud increases.
- Amazon Elastic Compute Cloud: Large number of services in e-commerce, supported by development platform and infrastructure.

ASU®

Ira A. Fulton Schools of Engineering

Arizona State University