Web Services Security in Pervasive Environment

Author

Muhammad Shoaib

MS-07 (Software Engineering)

Supervisor

Dr.Ghalib AsadUllah Shah

Assistant Professor

DEPARTMENT OF COMPUTER ENGINEERING

COLLEGE OF ELECTRICAL & MECHANICAL ENGINEERING

NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY

ISLAMABAD.

August, 2010

Web Services Security in Pervasive Environment

Author

Muhammad Shoaib

MS-07 (Computer Software Engineering)

A thesis submitted in partial fulfillment of the requirements for the degree of

MS (Computer Software Engineering)

Thesis Supervisor:

Dr. Ghalib AsadUllah Shah

Assistant Professor,

Department of Computer Engineering.

Thesis Supervisor Signature: _____

DEPARTMENT OF COMPUTER ENGINEERING

COLLEGE OF ELECTRICAL & MECHANICAL ENGINEERING

AUGUST, 2010

ABSTRACT

Several security solutions including open source protocols and specifications have been proposed in Pervasive Environment. Most of these schemes rely on intermediate servers for authentication and security provision for end to end secure communications between devices. This might be suitable for certain scenarios but broad range of users are not comfortable with it. End to end secure communication is among the most important requirement in pervasive environment.

Due to heterogeneous nature of the pervasive operating environment and ubiquity of communication devices, service adaptation is required at run time by inferring environment state. Dealing with security issues in such diverse conditions becomes a real challenge. In the pervasive environment, the security framework needs to be context-sensitive and services being provided in the pervasive environment also needs secure mechanisms for access control. So every single service need secure channel or some mechanism to provide scalable and efficient environment to operate in.

There are four security features that should be provided by any system called, Confidentiality, Integrity, Authentication, and Non repudiation. In the scheme we have chosen available secret key algorithms for confidentiality, PKC for authentication and non repudiation, and one way encryption schemes (Hash Functions) for integrity provision among communicating entities.

In this work, we have aim to provide security mechanism for end to end users without intervention of intermediate servers using secret key and public key cryptography with REST services. We will provide a comparison study of security algorithms already being used in industry in this environment too.

UNDERTAKING

I certify that research work titled **"Web Services Security in Pervasive Environment"** is my own work. The work has not been presented elsewhere for assessment. Where material has been used from other sources it has been properly acknowledged / referred.

Muhammad Shoaib,

REG NO: 2007-NUST-MS PhD-CSE (E)-08

ACKNOWLEDGEMENTS

First of all I am grateful to all mighty Allah for his blessings and continuous inspirations without those I am nothing.

I admit my supervisor Dr. A. Ghalib patience and continuous effort right from start. I salute you sir and really appreciate your coordination and guidance throughout my research work.

Few people you meet very few times but you never forget them in rest of your life, Mr. Nazir Malik is that kind of person. He gave me lot of confidence and of course ideas how to research and seek knowledge.

DEDICATION

To my parents for their support and trust

TABLE OF CONTENT

LIST OF FIGURES

Chapter 1. **Introduction**

1.1 Overview

The development of information technology has great impact on human social and habitual aspects of life. Now people can communicate, share information and even trade irrespective of their locality. Technology Analysts predicts that pervasive computing is the next stage in the development of information technology arena. Vendors around the globe are investing to build tools to assist pervasive computing. PDAs, mobiles and pagers are far outnumbering desktops and laptops computers and trend is still going on. It is true for other equipments like vehicles, home appliances and surveillance systems or almost anything that was dump once, but not anymore.

Today applications are being develop in services oriented way, because this approach provides platform independency and lot more features those were dreams. When talk about pervasive environment it consist of different kind of devices so services oriented architecture is suitable for such platforms. Pervasive/Ubiquitous (for being everywhere) is among the hot research area in computing nowadays.

Diversity of platforms demands a framework of communication, computing and more importantly security. Security is one of the most important factor determine the success of any system. Such a framework will provide security via authenticating the users / services, define authorization rules and policies for available resources, confidentiality and message integrity.

1.2 **Pervasive/Ubiquitous Computing**

With the advancement of electronics particularly wireless and internet technology, pervasive computing is a trend toward increasing ubiquitous connected computer devices in environment. Pervasive devices are not personal computers but very tiny, even invisible devices either mobiles or embedded in any object imaginable like including home appliances, cars, clothing and various consumer goods. According to Dan Russell, director of the User Sciences and Experience Group at IBM's Almaden Research Center, by 2010 computing will have become so naturalized within the environment that people

will not even realize that they are using computers [16]. These devices will no more be dump and will keep track of their location, environment context and user information. User will not keep track of them; they will find and update users.

Several pervasive computing frameworks like [1] have been proposed so far but still lot more research is required in this area to choose the best among all. For any secure distributed system there are at least four security features needs to be provided including Authentication, Confidentiality, Integrity, and non-repudiation. These all will be discussed in details in later chapters.

1.3 Service Oriented Architecture

Nowadays applications are built by using concept of Service Oriented Architecture (SOA) for several reasons [17], it reduces development and maintenance cost exponentially, provides cross platform neutrality, better scalability and security features. Large systems may consist of number of services provided by different vendors from different operating environment. It is common to authenticate service or user accessing some other resource or service in SOA environment like other distributed systems. Likewise in pervasive environment authentication is required to know the user of service. One enhancement has made in this case is that user identities may be transferred across domains by using special central authorizes to keep the user information. Security Assertion Markup Language (SAML) is used for exchanging securities tokens among different service domains.

Simple Object Access Protocol (SOAP) is an xml based protocol was introduced in late 90s for developing distributed services. It uses xml messages for transfer data between intended entities. Before exchanging data entities need to agree on format of messages, mode of communication and encoding etc. SOAP specification does not provide mechanism to secure communication. End users use some other security schemes for that purpose. SOAP revolutionized the world of distributed computing. But it also suffers with some drawbacks those make it unsuitable for some environment like pervasive environment. For remedy of SOAP problem new architectural approach is introduced called Representational State Transfer (REST) for building web services. It is more like a mind set to build distributed services using world most widely used protocol

i.e. http. REST is based on the concept of resource centric approach to build services rather traditional API libraries based programming approach.

1.4 Problem Statement

Pervasive environment is post desktop model of computing. It has several distinguishing features and constraints as compare to corresponding desktop computing paradigm. In pervasive environment devices have numerous limited hardware factors like memory, power, processor and bandwidth etc. due to the these basic constrains software solutions proposed in desktop environment can be simply adopted in ubiquitous systems and devices.

Another very common issue is that these small devices outnumber corresponding desktop computing devices along with diversity of different vendors. These diversities require adopting some standard to follow to make compatible and interoperable components.

So there are two main research aspects those are still need lot research focus including how to deal diversities of pervasive devices and corresponding security infrastructure.

1.5 Objectives and Goals

As stated above to deal with the diversity of devices and building web services over these devices need common framework. Several such frameworks are proposed but we are concentrating on one that was proposed by N. A. Malik [1].

For secure communication over unsecure channel symmetric and asymmetric schemes are used together with hashing.

Most of the systems use hybrid approach to provide security to their users. In this work one such scheme is used. PKC is basically used for Authentication and non repudiation, secret key cryptography for confidentiality, hashing for integrity.

1.6 Thesis Organization

This thesis is organized in following chapters;

- Chapter 1 introduces the basic idea of pervasive devices and computing along with its future trends.

- Chapter 2 highlights encryption techniques, hashing schemes in context of REST based services. Hybrid approach is described that how to secure our services with these available schemes while communicating over unsecure channel.

- Chapter 3 includes some history of distributing computing and modern approaches as well. SOAP is described briefly in context of WS-Security that how security is provided in desktop environment. at the end Pervasive environment framework is discussed shortly.

- Chapter 4 provides the details of our approach and details building of test cases and Application.

- Chapter 5 highlights performance evaluation of our application. At the end conclusion are made and futures research areas are mentioned.

Chapter 2. Science of Cryptography in Context of REST

2.1 Overview

Writing in secret codes is a science being used since 2000 BC. Secure communication via message disguise is very common nowadays. Security is one of the prime factors that determine success of any system available in the world. Cryptography defined in advanced Oxford dictionary as;

"A secret manner of writing, either by arbitrary characters, by using letters or characters in other than their ordinary sense, or by other methods intelligible only to those possessing the key; also anything written in this way. Generally, the art of writing or solving ciphers."

Within the context of application to application communication there are certain security requirements listed as follow;

- Authentication determines one's identity.
- Privacy/Confidentiality ensures no one can read message except intended party.
- Integrity ensures message received at other end is unaltered.
- Non-Repudiation is a mechanism that sender actually sent the message.

There are basically three types of cryptography schemes used to accomplish these goals; *Symmetric (secret key), public key (asymmetric)* and *hashing*. As shown in figure 3-1, protocol is in fact application using these schemes to ensure secure communication e.g. TLS/SSL.

In all subsequent section there will two common communication parties will be involved called Alice and Bob. If there are third and fourth parties are involved then those will be Carol and Dave. Mallory is malicious user, Eve is eavesdropper and Trent is trusted party.

2.2 Cryptography Types

There are three types of cryptographic schemes to provide essential security requirements including authentication, confidentiality, integrity, and non-repudiation. In the following sections all of these are described briefly with some examples.

- Secret Key uses a single key for both encryption and decryption
- Public uses one key for encryption and another for decryption
- Hashing uses a mathematical transformation to irreversibly "encrypt" information

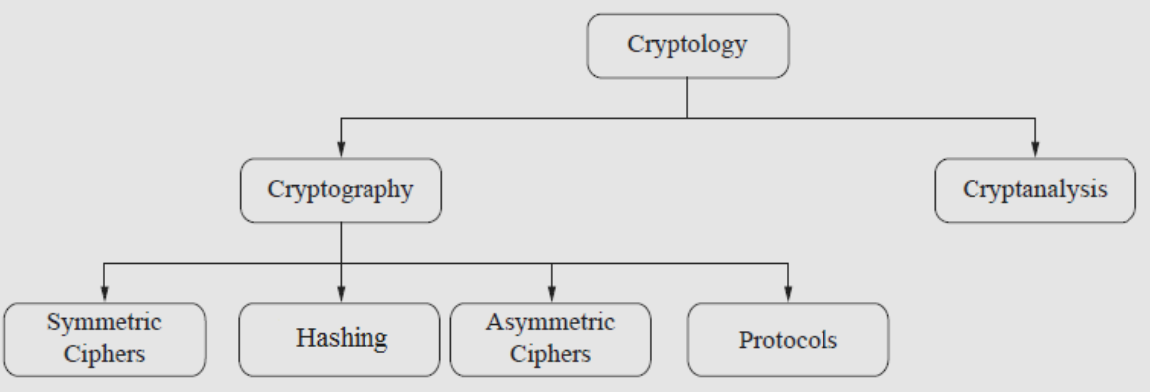


Figure 2-1: Field of Cryptology

2.2.1 Secret Key Cryptography

Single key is used for encryption and decryption purposes. Plain text is encrypted with some well known key to both communicating parties. Because single key is shared between interested parties so this scheme is known as Symmetric Encryption Scheme.

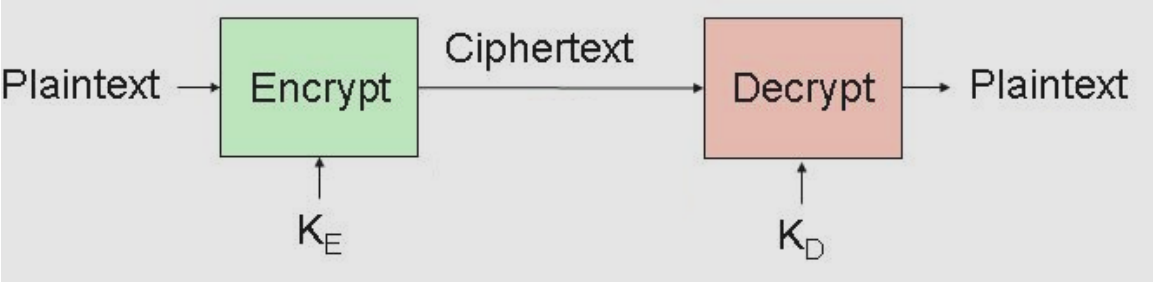


Figure 2-2: Symmetric Encryption Scheme

With this scheme it is obvious that key must be shared between two parties in order to encrypt and decrypt messages successfully. Key sharing is one of biggest problem in this scheme. Secret key cryptography is further classified into *Stream ciphers* and *Block ciphers*.

Stream ciphers will operate on single bit or word at a time with some feedback mechanism. There are several stream ciphers available but among all are *Self Synchronizing Stream ciphers*. It actually considers previous n bits while calculating current bit.

One the other hand Block cipher works on fixed sized blocks of data. In general keeping key and plain text same resultant cipher text will be always same in the case of Block ciphers.

2.2.2 Symmetric Cryptographic Algorithms

There are number of commercially available cryptographic algorithms including DES, AES, Blowfish, Rivest Cipher (RC2, RC4, RC5 and RC6), Twofish, Camellia and many more. In following section some of these are briefly described.

i. Data Encryption Standard (DES)

It is block cipher selected by National Bureau of Standards as an official standard for Federal Information Processing Standard (FIPS) in 1976 in United States. It uses 56bit key and 64bit encryption block. It is now considered as a weak crypto system because of its smaller key size. In 1999 distributed.net and Electronic Frontier Foundation publicly break DES in 22 hours and 15 minutes.

ii. Triple DES

The weakness of DES was its smaller key size. In order to secure it a new variant was proposed called Triple DES. In this scheme each data block is undergoes DES scheme three times. But now Triple DES is superseded by Advanced Encryption Scheme (AES).

iii. Advanced Encryption Standard (AES)

AES is another symmetric key encryption standard that is adopted by US government as standard encryption scheme. It has 128 bits block size with variable key sizes are supported like 128, 192, and 256 bits.

It is based on a different design pattern from DES and use *Substitution Permutation Network* rather Feistel network. It is faster in both software and hardware implementation compare to DES and Triple DES.

2.2.3 Public Key Cryptography (PKC)

Public key cryptography was proposed by a professor Martin Hellman and graduate student Whitfield Diffie in 1976 of Stanford University. In this scheme two parties engage each other over an unsecure channel without using a shared secret key. PKC is basically based on a mathematical function that is easy to computers but its inverse is difficult to compute. There are several schemes to elaborate this concept like factorization and exponentiation etc. one important property of this scheme is that message encrypted with one key can only be decrypted with other one and vice versa.

One key is known as public key and other is called as private. As names suggests that public key is published in intended audience whereas private is kept secret always. It is really important to note that knowing one key does not reveal information to determine second one. If two parties Alice and Bob want to communicate with each other then Alice will send message to bob encrypted with his own private key. It serves three purposes;

- Aauthentication of Alice (she is only person who has private key to encrypt the message)
- It also provide confidentiality (information is not disclosed except to users having corresponding public key) and
- Non repudiation (Alice cannot deny that she did not send that message because she is the only person keeping secret key)

PKC systems being used for key exchange and digital signatures are listed and briefly described in following sections;

i. RSA

It was first proposed and implemented by three MIT mathematicians Ronald Rivest, Adi Shamir, and Leonard Adleman. RSA is being used in number of software and hardware applications to provide secure communication mechanism. It is used for *key exchange*, *digital signature* and *small messages encryption* (as it is computation hungry compare to symmetric encryption schemes). Key pair is derived from a very large number, n (product of two prime number chosen by special rules). Basically there two prime numbers are very large number consists of more than hundred digits most of the time. Public key include one of prime factor and number n and it is difficult enough to compute corresponding factor from this information. Difficulty lies in computing private key because of larger key space. RSA security lies into this whole idea. As RSA support variable sized encryption block and key size so users can easily increase number n to make system more secure and difficult to break.

ii. Diffie – Hellman

After RSA Diffie and Hellman came up with new scheme. It is used only for secret key exchange, but not for digital signatures and authentication.

iii. Digital Signature Algorithm (DSA)

It is specified in NIST Digital Signature Standard (DSS) and used for message authentication via digital signatures.

2.2.4 Hash Function

It is also called message digest or one way encryption. It is basically based on one way encryption function with fixed length. So it makes impossible to perform reverse process for actual plain text. Hash functions are used to provide digital fingerprint for a file in order to make sure that file is not altered or corrupted. Number of operating systems use hash functions to secure passwords. In order words hash functions measure the file integrity. Commonly used hash functions are briefly described below;

i. Message Digest (MD)

It is a 128 bit hash values producing scheme and has number of variations like MD2, MD4, and MD5.

- MD2 is used for only limited memory systems like smart cards.
- MD4 was developed by Rivest and it is designed to work fast in software.
- MD5 removes the weakness of MD4 and slower than the MD4 because it performs more manipulation on data. It was designed by Rivest.

ii. Secure Hash Algorithm (SHA)

It is National Institute of Standards and Technology (NIST) standard hash scheme and produces 160 bits value. SHA has five variations including SHA1, SHA224, SHA256, SHA384, and SHA512 with 160, 224, 256, 384, and 512bits hash values.

Two files may have same hash value called collision. In case of MD scheme there is 128bit hash length. But there may be lot more than 2 128. In real worlds there may be way more than that values. But it is really very difficult to create a file with same hash values. Hashing is extensively being used to provide message integrity. In most of situations people use multiple hashing schemes at the same time like SHA1 and MD5.

2.3 Representational State Transfer (REST)

Distributed systems can be consider as a system that consist of resources each referenced via some standard way (URI), use standard way to access (HTTP). Every resource may have some representation and metadata modeling (XML), reference to other related documents (XPointers, XLink), and description how it can be accessed (WSDL). Distributed systems may employ common authentication and authorization mechanism.

Representational State Transfer (REST) based services got lot of attentions due to their simplicity in last few years. It isn't a standard rather an architectural style to build services. Due to its resource centric approach it suits the environment where data manipulation is common [12]. Service-oriented computing promotes the idea of assembling application components into a network of services that can be loosely coupled

to create flexible, dynamic business processes and agile applications that span organizations and computing platforms [6].

REST stands for Representational State Transfer, and it was first time coined by Roy Fielding in his doctoral dissertation. REST encompasses a simple philosophy for modeling problem domains: “give a URI to everything that can be manipulated by a limited set of operations and let the client software determine the usage metaphor” [15]. Major concern in development of REST type architecture was how to use existing infra structure without destroying or adversely impact widely deployed setup.

2.4 **REST Elements**

In traditional distributed systems key processing aspects of systems are encapsulated into processing components. REST style is an abstraction of architectural elements in distributed system and there are six such elements are identified including;

- Resource is a key abstraction of information. Any information that can have a name is a resource. Every resource has identity, state and behavior.
- Resource Identifier is used to identifies a resource through some standard way. URI are used for that purpose.
- Resource metadata describes the resource.
- Representation is something (web page or document) that is generated by resource. Some resources may be static their representation does not change.
- Representation metadata describes the representation.
- Control data defines the purpose of messages between components.

Every resource has some identified via some URI as follow;

http://www.goolge.com/searchengin/help.php

When some resource is accessed, its representation is returned. A resource may have multiple representations in different context of usage like it may represent in HTML

format or some other markup language like XML. Each representation place client application in some state when some link in that representation is traversed different representation is returned. In result client application undergoes number of state transfer. It is reason why the term Representation State Transfer is used.

2.5 REST Principles

There are certainly some principles that REST architectural styles follow [15].

1. A resource is anything that has identity.
2. Every resource has a URI.
3. A URI is “opaque,” exposes no details of its implementation.
4. GET operations are “idempotent,” free of side effects.
5. Any request that doesn’t have side effects should use GET.
6. All interactions are stateless.
7. Data and metadata formats are documented.
8. Data is available in multiple flavors.
9. Representations include links to other resources.
10. Document and advertise your service API.
11. Use available standards and technology.
12. Refine and extend architecture, standards and tools.

2.6 URI and Resource Modeling

URI are used to identify resources uniquely. There are generally two schemes of URIs; *hierarchical* and *non hierarchical*. Hierarchical schemes can represent both absolute and relatives URIs (Http :); whereas a non hierarchical scheme only represents absolute URIs (mailto:).

Absolute scheme is consists of three elements; scheme, scheme specific part, and fragment. Scheme specific part is further divided into three elements; authority, path, and query as shown below (first two are for absolute addresses and third is for relative);

<scheme>:<scheme-specific-part>#<fragment>

<scheme>://<authority><path>?<query>#<fragment>

<path>?<query>#<fragment>

Real strength and flexibility of REST comes from pervasive usage of URIs. Resources are exposed via URI instead of some messaging interface like in traditional web services implementation. Exposed resources support well defined actions to transfer representation from server to client. Modeling problem domain into a set of resources is called *Resource Modeling*. Putting any kind of action, method name or process in URI is considered to be bad practice. URI actually represent some noun or resource on that some action is performed.

Resources can be modeled via some hierarchical way easily. For example there are two entities Class and Student. These entities can be modeled in several ways. In following example URI are represented relative.

/Class=6B/Student=Shoaib

/6B/Shoaib

/Classes/6B/Students/Shoaib

/School?Class=6B;Student=Shoaib

There may be different variations of resource modeling as mention above. But one that is seems more convenient is;

/Classes/6B/Students/Shoaib

It can easily be extended to solve some other resource collection mystery. Following URI represents Class collection in a given context of Scheme and authority;

/Classes/

This part represents a particular class in all classes

/Classes/6B/

Following part represents collection of students in particular class

/Classes/6B/Students

Particular student in any particular class can be access via;

/Classes/6B/Students/Shoaib

Further particular resource can also be extracted via using fragment of the URI;

/Classes/6B/Students/Shoaib/?age=20

It is strongly recommended that to put only required information that identifies the resource.

2.7 REST Methods

There are number of methods available in HTTP specification but most common among all are; *GET* is used to retrieve resource, *POST* is used to update and insert and extends existing resource and may affect state of other resources, *PUT* is for creating new or replace resource, *DELETE* to delete resource, and *HEAD* is used to retrieve resource representation and metadata. All of the methods are briefly described in the context of REST.

2.7.1 GET

It is an idempotent method to access resource representation without affecting resource state. Great advantage comes when results can be cached for performance boosting.

2.7.2 POST

It is used to modify the resource represented in corresponding URI. Usually it is misused and even deletes resources.

2.7.3 DELETE

It is used to delete a resource given at valid URL. It is suppose to delete one resource at a time. If more than one resource is involved then POST method is used.

2.7.4 PUT

To change the resource representation client uses PUT method. There is a major difference between PUT and POST. PUT and DELTE methods operate on resource representation as a whole. It may also include CONTENT-RANGE headers to modify only a portion of the entity.

2.8 Conclusion

In most of the cases Secret key, Public key cryptography, and hashing is used all together to proved all essential elements of any secure system i.e. Authentication, Confidentiality, Integrity, and non repudiation.

Alice use PKC to for hash security and Secret key scheme to encrypt message for confidentiality. PKC can also use for encryption but it is rarely used for that because it is 1000 times slower then secret key cryptography.

Alice use a randomly generated session key in secret key scheme and share that session key via PKC using Bob public key. On receiving end Bob will get will session key by using his private key and decrypt message then compute hash values of that message. Computed hash will be matched with decrypted hash values from Alice sent digital signature using her public key. If two hash values matches the message it came altered otherwise corrupted.

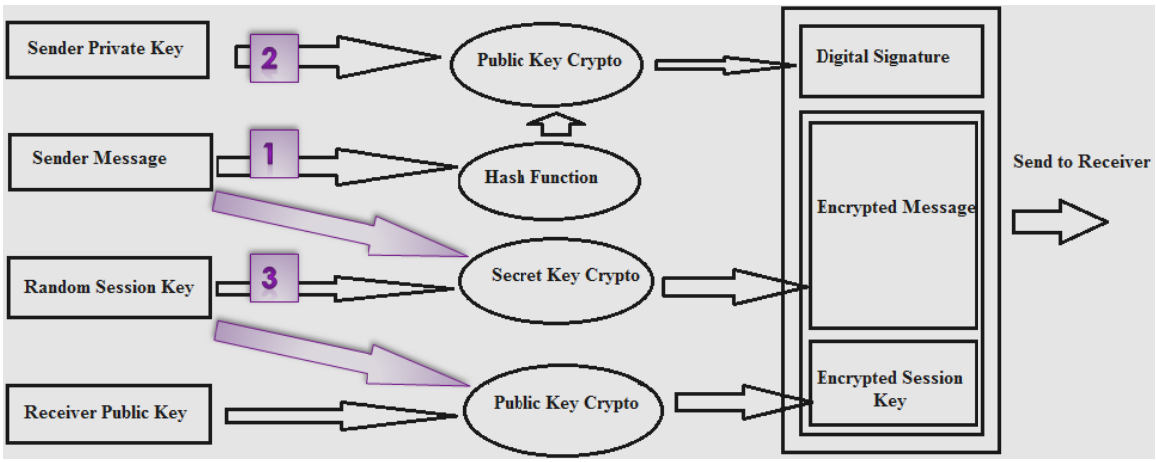


Figure 2-3: Hybrid Security System

Chapter 3. **Web Services in Pervasive Environment**

3.1 **Overview**

Pervasive computing is an open research area. several aspects needs more attention like security provision between end-end devices without intervention of third party servers, optimized bandwidth utilization, platform neutral communication framework, context aware computing and services adaptation etc. In this case only the first issue is highlighted. Lot of published work and material is reviewed as categorized below;

- Distributed computing
- Web Services and SOAP
- Web Services Security (WSS)
- Security Assertion Markup Language (SAML)
- Pervasive Security Frameworks

Let's have a look on each one by one along with pros and cons.

3.2 **Distributed Computing**

Systems in which hardware or software components are distributed on different machines over networks and they communicate and coordinate their actions via messages [3]. Most prominent distributed object technologies or middleware are DCOM of Microsoft, CORBA of OMG, and RMI of JavaSoft. These are extension of traditional object oriented systems in a way that they allow objects to be distributed in heterogeneous environments. Distributed objects may reside in separate address space independent of application or computer and still can be accessed in application on other computers over network or same machine.

Let's take a look on these distributes object technologies or middleware one by one.

i. Common Object Request Broker Architecture (CORBA)

It is an open distributed object computing infrastructure and standardized by Object Management Group (OMG). OMG was founded in 1989 to promote the adaptation of object technology and reusable software components. CORBA is widely used in non-window platforms. It has own object extension model CORBA Component Model (CCM).

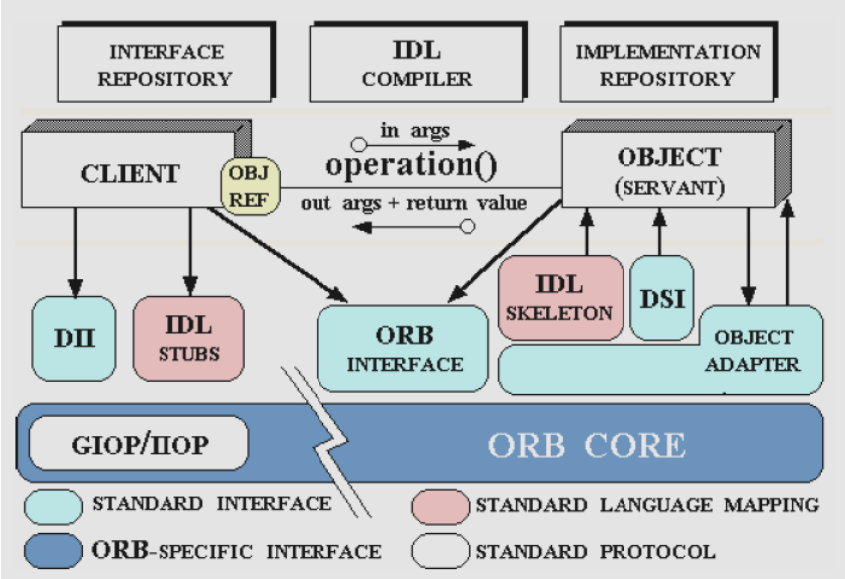


Figure 3-1: CORBA ORB Architecture [3]

Object Request Broker (ORB) is a core component of CORBA architecture. It hides complexities from programmers; CORBA objects interact with ORB whether requesting object is located locally or remotely. It provides basic messaging, communication, directory, security and location transparency and insulates applications from details about hardware, networking and system.

Internet Inter-ORB Protocol is a specialization of General Inter-ORB Protocol and works over TCP/IP. It was developed according to CORBA 2.0 specifications and for communication of ORBs from different vendors. CORBA relies on IIOP or other specialization of GIOP for remoting objects. The CORBA object is a programming entity that consists of an identity, an interface, and an implementation, which is known as a Servant. The servant is an implementation programming language entity that defines the

operations that support a CORBA IDL interface. OMG has defined an interface definition language (IDL) that declares the interfaces and methods of a CORBA server object. Every CORBA object must be declared in IDL. The IDL compiler creates stubs and skeletons that serve as the “glue" between the client and server applications respectively, and the ORB.

ii. Distributed Component Object Model (DCOM)

It is an extension of Component Object Model and standard of Microsoft for distributed computing. Most of desktop computer using window are based on COM. Software components communicate with each other locally or remotely using DCOM approach. DCOM use Microsoft Transaction Server (COM+/MTS) as extension model.

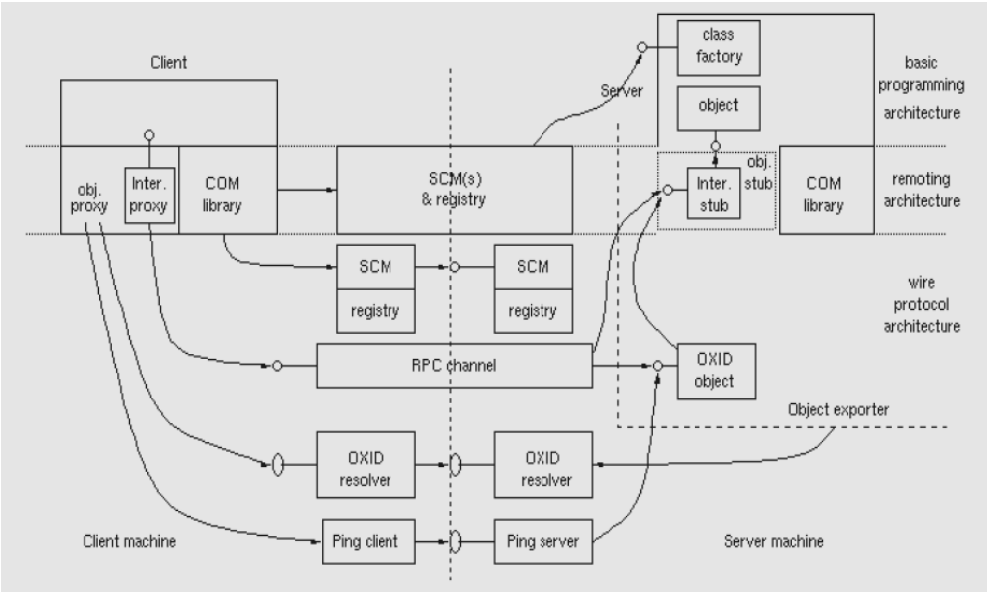


Figure 3-2: Overall Architecture of DCOM [3]

DCOM runs over protocol Object Remote Procedure Call (ORPC). ORPC is built over DCE/RPC and interact with runtime COM services. Dispatch tables are maintained to keep the function pointers each represent some interface. Client use these function pointer of the sever interface and start calling exposed methods as that method resides in client own address space.

Microsoft Interface Description Language (MIDM) is used to define interface for DOM/RPC objects. As every COM object has IUnknown interface implemented which has three essential methods QueryInterface(), AddRef(), and release(). There is another interface IDispatch that is extension of COM IUnknown and it acts as a gateway for many more interfaces. Both these interfaces are supported by MIDL for interface descriptions.

In order to call objects implementing IDispatch interface, MIDL generates type libraries to store the object type information. Unique Universally Identifier (UUID) identifies each class and interface in COM.

iii. Java/Remote Method Invocation (Java/RMI)

RMI standard is developed by JavaSoft. Java is promoted from merely a programming language to three compatible platforms like J2SE (Java to Standard Edition), J2EE (Java to Enterprise Edition), and J2ME (Java to Micro Edition). J2SE is standard toolkit for programming; J2EE is for enterprise and internet applications development whereas J2ME is for embedded, mobiles devices and mobiles etc. JavaBean and Enterprise JavaBean are being used for Java/RMI extension model.

RMI supports remote objects by running on a protocol called the Java Remote Method Protocol (JRMP). Object serialisation is heavily used to marshal and unmarshal objects as streams. Both client and server have to be written in Java to be able to use the object serialisation. The Java server object defines interfaces that can be used to access the object outside the current Java virtual machine (JVM) from another JVM on for instance a different machine. A RMI registry on the server machine holds information of the available server objects and provides a naming service for RMI. A client acquires a server object reference through the RMI registry on the server and invokes methods on the server object as if the object resided in the client address space. The server objects are named using URLs and the client acquires the server object reference by specifying the URL.

iv. Problems with Traditional Distributed Computing

All three middleware technologies use client/server approach for communication and use transport levels protocol like TCP for communication. To abstract the networking calling conventions and details proxies or stubs are used so that programmer can only concentrate business logic implementations. All the heavy lifting like Marshalling of parameters is done by these underlying proxies. Marshaling / Unmarshaling actions convert local data into network format and remote system format. It also includes the byte ordering and number representation on different machines.

These technologies opened a new ways to distributed services but suffer number of problems like lack of interoperability between different distributed technologies, firewall restriction, deployment and debugging problems. E.g. in the case DCOM components dynamic ports opening is required to access the services and it is true for both client and server. It may be possible when applying this methodology in Intranet environment because firewalls may configure in that case easily to allow those ports but in Internet environments corporate firewalls does not allow that facility. Allowing dynamic port opening may be a security problem for enterprises. Likewise due to heterogeneous nature of platforms different parameter and result marshaling is required in order to make communication possible.

3.3 Web Services and SOAP

XML (eXtensible Mark-up Language) and SOAP have together with application servers shown a great promise for interoperability between the different object models. Microsoft and OMG as well as many other vendors have adopted XML as standard messaging format. XML is a true platform independent standard and can be used by any type of application or object model on any hardware platform. XML-files are ASCII-files and can be transported by the HTTP-protocol avoiding problems with for instance firewalls. A disadvantage with XML is to define agreed meta-models between suppliers and consumers of XML messages. In addition is XML quite space-intensive with possible very large overhead of descriptive information. It is however possible to use style-sheet translators as XSLT, to translate a given XML-file from an XML-model to another XML-model. SOAP satisfies the need to exchange structured data by the Web independently of

the underlying platforms. SOAP does not, in contrast to CORBA, offer any services or management tools. The firewall argument in favour of SOAP is maybe dubious by making SOAP perform RPC calls through firewalls. This subverts the security regime because the firewall thinks that it is just harmless Web traffic that passes through it. Microsoft's .NET is promised to implement both XML and SOAP giving the Windows applications opportunity easily to interact with non-Windows applications.

In order to reduce maintenance cost of distributed applications XML data representation language was introduced. Introduction of XML with Simple Object Access Protocol (SOAP) gives a birth to new web application paradigm. This technique was far more superior because it solved all the major problems like platform dependency and firewall restriction. SOAP envelop enclosed with XML messages deliver over HTTP as transport protocol. SOAP flexibility of adding custom security headers, binding on any transport protocol including SMTP and TCP/IP along with its platform and language neutrality are very important features.

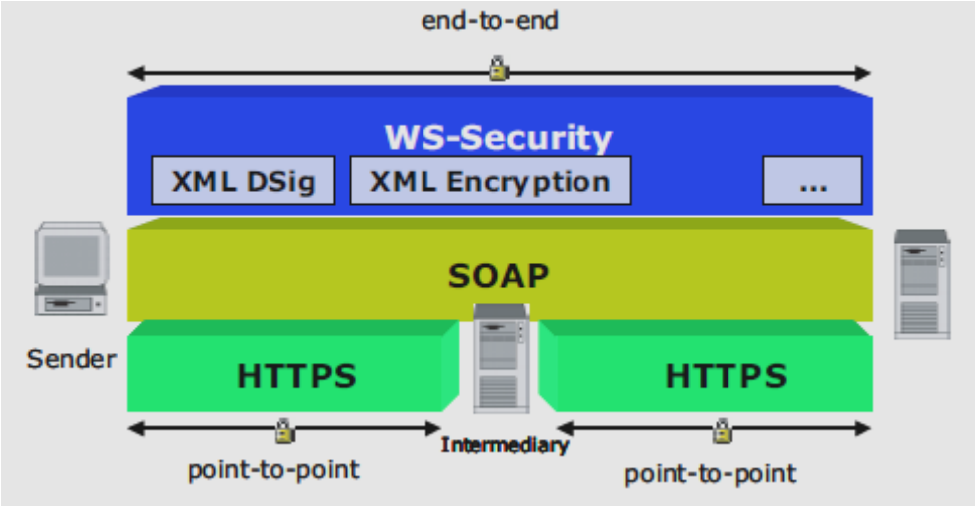


Figure 3-3: Protocol Layer and SOAP Security [4]

SOAP does not provide any message level security mechanism rather we need to some other ways to secure our communication. We can use other security models like WS-Security and its specifications [4]. Firewalls are bypassed usually when use SOAP over HTTP. The second problem is that connection based security like SSL/TLS only provide point to point security and fails when there are number of intermediate nodes.

WS-Security proposed message based security mechanism for SOAP. It defines security specifications how different security standards are like digital signature, encryption and SAML [8] for protecting whole or part of message and even insert security tokens can be used for all SOAP messages. Several application level security mechanisms are proposed. In order to provide better and maintainable security model we need better one like web services security proxy [4]. Its main advantage is that it will integrate with existing web services as well.

3.4 Web Services Security (WS-Security)

There is still no security testing methodology adopted specifically for web services implementing applications. WS-Security must be independent of underlying communication protocol e.g. authentication of both service provider and service invoker, end to end message content security not only transport level security. These are among the most important requirements in WS-Security models.

WSS was developed as an extension of the SOAP standard, describing the mechanism for using XML Encryption and XML signature to secure SOAP messages. Each messaging option has its own strengths and weaknesses. Its describe how to attach digital signature and encryption in soap headers. WSS also specify different security tokens like KERBOROS, X509 and SAML for security information exchange.

WSS is used for construction of variety of security models like PKI, Kerberos and SSL. It also supports multiple security tokens, multiple trust domains, signature formats, and encryption techniques as well as provide mechanism to secure a single envelop.

The core security mechanisms like XKMS, SAML, WSS (XML Encryption and Signing) are directly integrated with XML, thus provide fine grain integrity, data origin authentication and selective field confidentiality to all applications which use XML for data storage and exchange.

WSS is as Microsoft's strategy for dealing WS-Security. It’s a comprehensive security model that support and integrates and unifies several popular security models, mechanisms and technologies (both symmetric and public and private techs) makes WS interposable and platform neutral. WS security challenges;

- End to end message content security not just transport level security
- Authorization is more difficult to write when environments are more loosely coupled
- Methodologies for secure WS
- Clients and services do not have a way to negotiate their mutual constraints and capabilities before interacting
- Securing WS infrastructure needs XML's granularity
- Multiple security tokens

WSS is message level standard that is based on securing SOAP messages via XML digital signature, confidentiality via XML encryption, and credential propagation via security tokens.

In fall 2002 Microsoft, IBM, and VeriSign moved their web service security standard to OASIS in order to provide the baseline security for web services. This standard includes encryption and digital standard for message confidentiality and integrity respectively along with authentication. For authentication different security tokens were introduced like user Name token, Kerboros token, X.509 certificate token, and SAML token [5]. For each token there is token profile specification that describes how to actually implement it.

WSS defines set of SOAP headers to implement security measures in web services. WS-Security also defines the standard how to passing around security tokens.

The ultimate goal of WS-Security is to enable the secure SOAP messages exchange. The deriving WSS specifications includes multiple security tokens, multiple trust domains, multiple encryption and digital signatures, and end to end message content security instead of transport level security. At this point SOAP and SAML are working together. WSS defines how to insert security information in SOAP and SAML define what that information is. Overall thrust of WSS is to build the protocol stack similar to web services like SOAP, WSDL, and UDDI etc.

WSS support wide variety of security models like PKI, Kerboros and SSL etc. It also support multiple security tokens, trust domains, signature formats and encryption technologies.

3.4.1 Security Assertion Markup Language

SAML [8] is a specification that allows passing security tokens defining authentication and authorization rights in XML format. WSS use SOAP header to send security tokens and it support multiple security tokens. We can use different security tokens e.g. X.509 Certificates, Kerboros and SAML token [7] to authenticate users and services. In order to make communication possible SAML profiles requires agreement between system entities regarding entities identifiers, binding support, certificates information, and end points [9]. There are several SAML profiles define the usage of SAML assertions and request-response in communication protocols and frameworks [10].

SAML is an open standard that encodes security assertions and corresponding protocols messages in xml messages. Currently it defines authentication, authorization and attributes assertions. It also encodes security assertions and corresponding protocols messages in xml messages. Furthermore it specifies a request response protocol which can be used by the services provider to request assertions from identity provider. Binding defines how to send SAML protocol messages using SOAP over HTTP. Profiles determine how SAML can be used in standard web browsers. It also provides extension mechanism e.g. XACML (Extensible Access Control Markup Language) for fine grain access control being used in grid computing.

SAML standard includes descriptions of the use of SAML assertions in communication protocols and frameworks called profiles contains protocol flows and security constraints for applications of SAML. There are three types of SAML statements are supported;

- Authentication Assertion
- Authorization Assertion
- Attribute Assertion

WSS is a messaging language and SAML is security language. It is expected that both these provide the solid foundation for stable and flexible web services architecture.

3.4.2 XML Key Management Specification (XKMS) [13]

It is a joint venture of Microsoft, VeriSign and webmethods as open standard to simplify the securing of XML based internet transactions using PKI and digital certificates. XKMS describes the protocols for distributing and registering public keys, suitable for use in conjunction with the standards for xml signature and xml encryption. It overcomes the WS PKI complexity by treating the WS as client of key management services. Key objective of XKMS design is to minimize the complexities from client application implementations by shielding them from the complexity and syntax of underlying PKI used to establish trust relationship. It has two parts;

- XML KEY INFO SERVICES SPEC AND
- XML KEY REGISTRATION SERVICE SPEC

XKISS is responsible for maintaining public part of public private key combination and XKRSS is for private part.

3.4.3 XML Access Control Markup Language (XACML)

It is standard language that specifies schemas for authorization policies and authorization decision request response. It also specifies how to evaluate policies against requests to compute response. XACML has three top level elements <Rule>, <Policy>, and <PolicySet>. Using XACML enterprise can define platform independent rules for how its resources can used by those inside and outside the enterprise. Enterprises can communicate without aligning their computing platform, just has to align access policies.

3.5 Pervasive Computing Framework

Generally we use same protocols for pervasive environment as being used for corresponding desktop environment with little precautions due to certain hardware constraints.

As proposed in paper [1], general framework for secure service architecture in pervasive environment. But if we further elaborate it we can divide it into following categories.

Access Control Module (ACM) is responsible for access of requested resource by the corresponding user.

- Access Control Module
  - Authentication
    - Trusted Third Party
  - Authorization
    - Context Awareness
  - Service Provider
- Security Policy Enforcer
  - User Policies
  - Enterprise Policies

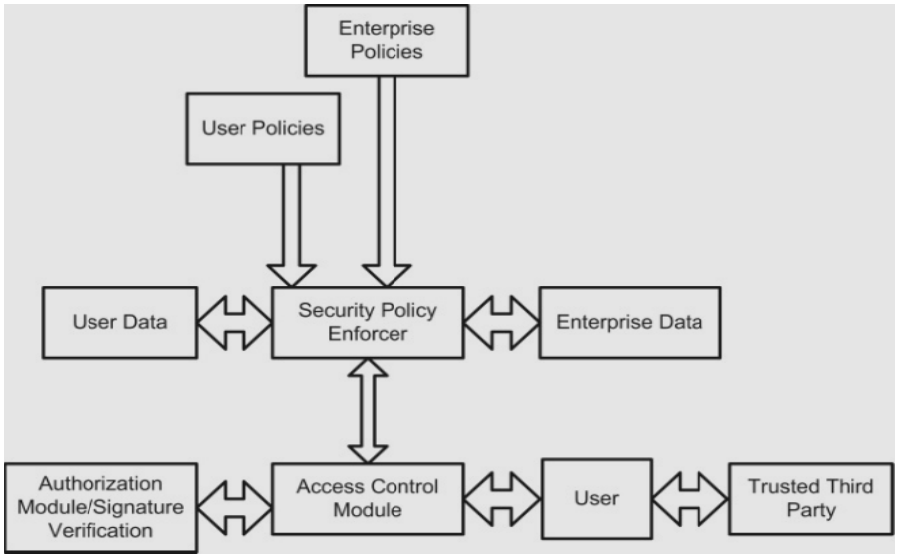


Figure 3-4: Secure Pervasive Architecture

3.5.1 Access Control Module

Access Control Module (ACM) is basically have two important parts may be three sometime if we consider Service as part as well.

i. Authentication

Authentication is done through SAML Response. It is for identifying user, whether legitimate or not. It includes the participation of Trusted Third Party for registration and Assertions. TTL will be a reliable server who will keep the user profiles and authentication information. There may be several TTL servers for single organization.

ii. Authorization

Authorization is for the purpose of granting access to requested resource. It is more complex then authentication, it depends upon number of factors in pervasive environment most importantly context and security levels in different conditions.

iii. Services

It may be a separate part but here we are considering it as part of ACM. It keeps the resources in hand and provides different operations to authorized users.

3.5.2 Security Policy Enforcer

There may be two types of policies in pervasive environment generally. All other may be categories in these if necessary.

i. User Policies

Define user policies for services access and manipulation.

ii. Enterprise Policies

Define enterprise policies can override user policies.

Chapter 4. **Proposed Methodology**

4.1 **Overview**

Several encryption and hashing schemes are available for message confidentiality and integrity respectively. REST is an architectural style of building services using well known protocol like HTTP. This chapter is classified in following sections;

- SAML Response Generation
- Hashing
- Encryption
- REST Services

4.2 **SAML and Serialization**

There are three available versions (v1.0, v1.1, and v2.0) of SAML and version 2 is latest and used in current case. SAML is built upon number of existing protocols and standards including;

- XML Schema
- XML Encryption
- XML Signature
- Hyper Text Transfer Protocol (HTTP)
- Simple Object Access Protocol (SOAP)

W3C has defined and standardized each one of above mentions building blocks in separate documents. In order to generate and consume SAML messages, it is essential to convert schemas to classes. But at the end we have to serialize the whole or part of the content in order to communicate across the boundary of application domain. .Net

platform is being used for implementation of all steps and running simulation but can be ported to any platform independent of language.

4.2.1 Classes Generation

The very first step of implementation is to generate classes from corresponding schema files. In .Net the procedure is quite simplified and can be done using utility XML Schema Generation (xsd.exe).

XSD utility has several switches used for configuring the behavior of the utility. XSD utility can be accessed via going Programs Files -> Visual Studio 2008 -> Visual Studio Tools -> Visual Studio Command Prompt and typing xsd.

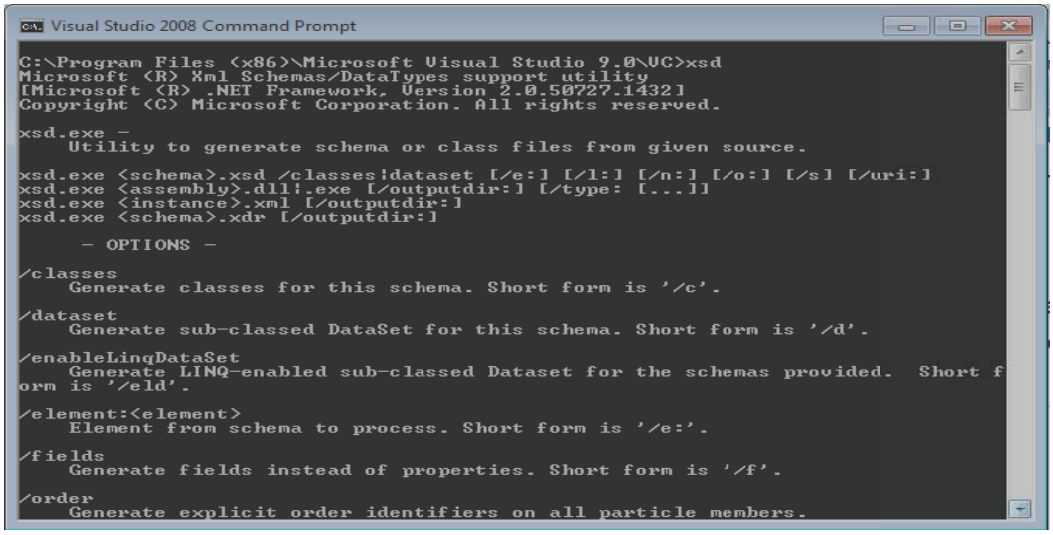


Figure 4-1: XSD Switches

```
xsd /c /language:CS XSDSchemaFile.xsd
```

When the above command run it will create corresponding class (XSDSchemaFile.cs) in C# language in schema directory. It is also possible to specify multiple .xsd files as well. In current case following command created corresponding types. As shown in figure below.

```
xsd xmldsig-core-schema.xsd xenc-core-schema.xsd saml-schema-assertion-2.0.xsd saml-schema-protocol-2.0.xsd /classes
/namespace:saxxxon.common.Security.Saml20 /outputdir:..
```

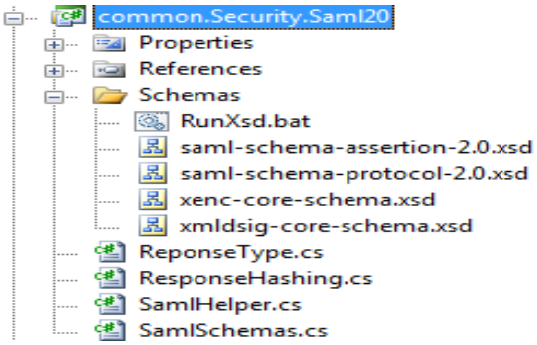


Figure 4-2: SAML Classes Generated through XSD Utility

4.2.2 SAML Response Generation

SAML is used primarily for security tokens delivery across application domains and it define the way how security relevant information is packaged. But how SAML response will be sent over network is described in other protocols like SOAP and usually details are part of SAML binding. Its overall structure is shown in figure 3.1;

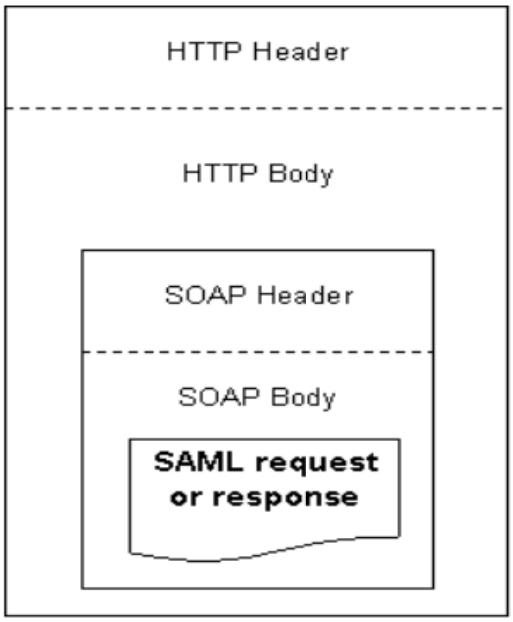


Figure 4-3: Binding over SOAP

SAML request and response has very similar format and structure as shown in figure 3.2.

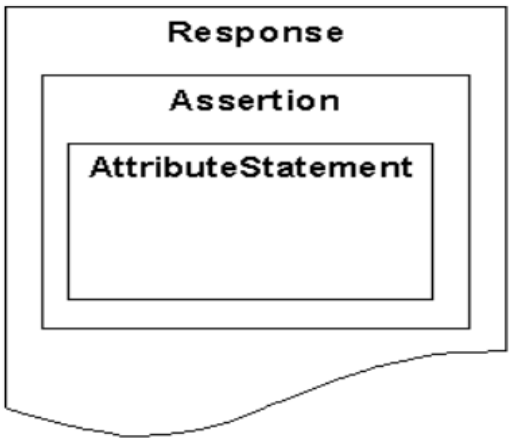


Figure 4-4: Abstract SAML Request/Response Structure

Loosely speaking, a relying party interprets an assertion as follows:

Assertion *A* was issued at time *t* by issuer *R* regarding subject *S* provided conditions *C* are valid.

A SAML assertion contains the security information and enclosed in following way

```
<saml:Assertion ...>
  ...
</saml:Assertion>.
```

There are several attributes of an assertion including information about issuer, recipient, target, domain, subject and custom key values paired attributes. Additionally it also contains other conditional statements as well like expiration, audience and dependencies on other statements. As shown in following figure, our application contains a simple interface for SAML response generation. Following information is required to generate SAML response.

4.2.3 SAML Attributes

A SAML assertion is a package of information including issuer of SAML response and subject, conditions and advice, and/or attribute statements, and/or authentication statements and/or other statements. Statements are optional. The SAML assertion "container" itself contains the following information:

- Issuer is the Trusted Third Party (TTP) who is response for SAML response generation
- Recipient is end point where this response is entertained
- Target specify the targeted resource for being assertion is made
- Mention domain of SAML response
- Subject is the user/service for which assertion is made

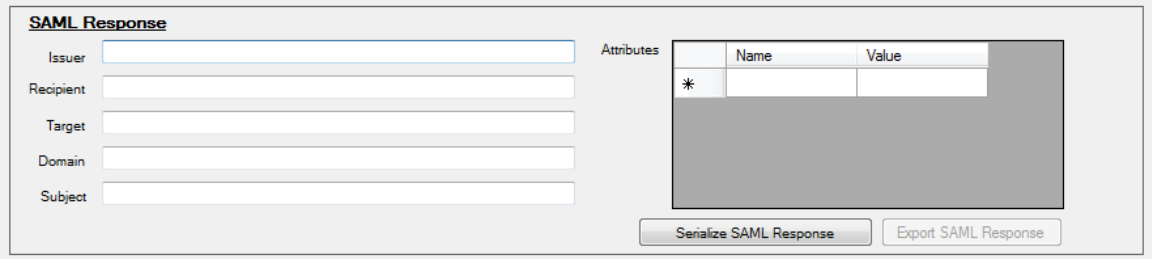


Figure 4-5: SAML Response Generation

There are three types of SAML statements

- Authentication statement assert about some subject on behalf of some third trusted party
- Authorization statement contains user privileges and resource access information
- Attribute statement contain custom information in the form of key value pair format

In this case when SAML response is sent by this application will be an authentication statement (Asserting about some subject). And when authentication statement is processed at the end point, it will generate an Authorization statement specifying whether requested resource is accessible or not. Entities can exchange custom information via attributes statement but it is optional.

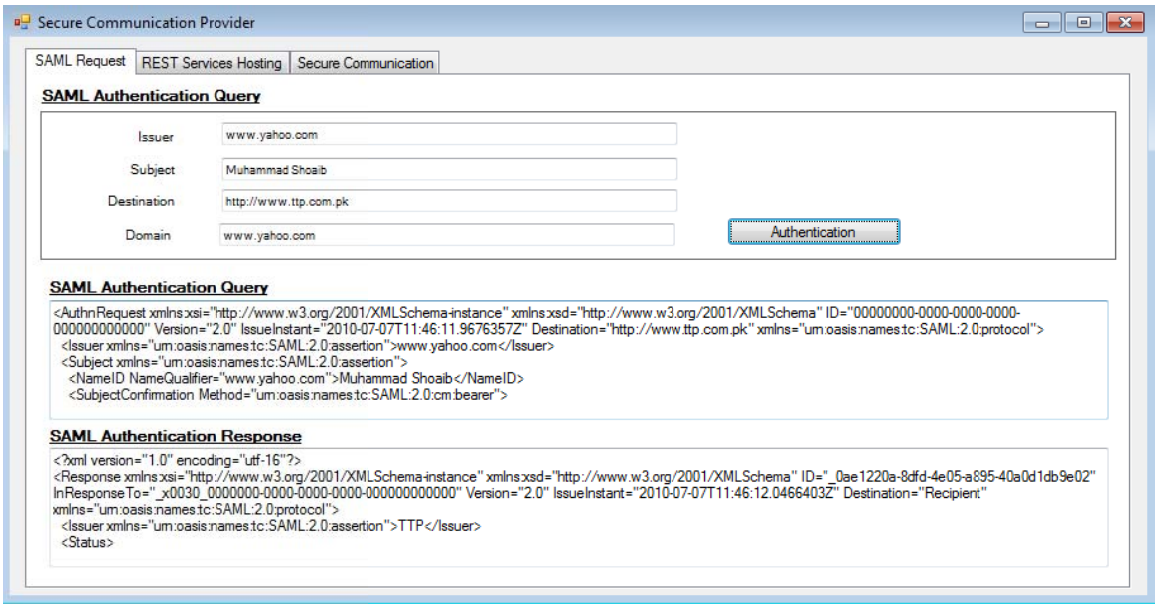


Figure 4-6: SAML Utility

In above SAML utility when user pass mentioned parameters then it will issue an authentication request to TTP. In response of that request TTP will send back response consists of an assertion if request is valid. Corresponding request and response are;

When client send this request to TTP, it will be validated for tempering if request is not tempered and user credential are matched then a valid response with success status will be generated. Otherwise failure message or blank message will be send back.

```
<AuthnRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    ID="00000000-0000-0000-0000-000000000000" Version="2.0" IssueInstant="2010-07-05T06:01:35.5830406Z"
    Destination="www.server1.com.pk" xmlns="urn:oasis:names:tc:SAML:2.0:protocol">
  <Issuer xmlns="urn:oasis:names:tc:SAML:2.0:assertion">www.server1.com.pk</Issuer>
  <Subject xmlns="urn:oasis:names:tc:SAML:2.0:assertion">
    <NameID NameQualifier="www.server2.com.pk">shoaib</NameID>
    <SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
      <SubjectConfirmationData NotOnOrAfter="2010-07-05T06:06:35.5840406Z" Recipient="www.server1.com.pk" />
    </SubjectConfirmation>
  </Subject>
</AuthnRequest>
```

Figure 4-7: Authentication Request

Corresponding response is;

```
<?xml version="1.0" encoding="utf-16"?>
<Response xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    ID="_a8bab301-1cec-4c41-a1e7-75a14bf9cfcc" InResponseTo="_x0030_0000000-0000-0000-0000-000000000000" Version="2.0"
    IssueInstant="2010-07-05T06:01:36.0670682Z" Destination="Recipient" xmlns="urn:oasis:names:tc:SAML:2.0:protocol">
  <Issuer xmlns="urn:oasis:names:tc:SAML:2.0:assertion">http://www.ttp.com.pk</Issuer>
  <Status>
    <StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success" />
  </Status>
  <Assertion Version="2.0" ID="_9c86da83-d93b-446c-ada8-c7edbae1faaf" IssueInstant="2010-07-05T06:01:36.0670682Z"
    xmlns="urn:oasis:names:tc:SAML:2.0:assertion">
    <Issuer>http://www.ttp.com.pk</Issuer>
    <Subject>
      <NameID NameQualifier="www.server2.com.pk">shoaib</NameID>
      <SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
        <SubjectConfirmationData NotOnOrAfter="2010-07-05T06:06:36.0670682Z" Recipient="Recipient" />
      </SubjectConfirmation>
    </Subject>
    <Conditions NotBefore="2010-07-05T06:01:36.0670682Z" NotOnOrAfter="2010-07-05T06:06:36.0670682Z">
      <AudienceRestriction>
        <Audience>www.server2.com.pk</Audience>
      </AudienceRestriction>
    </Conditions>
    <AuthnStatement AuthnInstant="2010-07-05T06:01:36.0670682Z">
      <AuthnContext>
        <AuthnContextClassRef>AuthnContextClassRef</AuthnContextClassRef>
      </AuthnContext>
    </AuthnStatement>
    <AttributeStatement>
      <Attribute Name="email" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
        <AttributeValue xsi:type="xsd:string">ttp@www.server2.com.pk</AttributeValue>
      </Attribute>
    </AttributeStatement>
  </Assertion>
</Response>
```

Figure 4-8: Authentication Statement in Response Body

4.3 Message Integrity

For message integrity there are number of pre existing one way encryption or hash functions are available. Hash functions generate fixed length value for a given input; minor change in file can have drastic change in hash values. Hash functions are used along with Public Key Cryptography (PKC) to secure communication.

There are several hash functions available for signing the documents. E.g. Message Digest (MD) and Secure Hash Algorithm (SHA) are commonly used. MD has 128bit hash length whereas SHA has five variation including SHA1, SHA224, SHA256, SHA384, and SHA512 with 160, 224, 256, 384 and 512bits hash length respectively. Greater length of hash value takes more time to compute. SHA1 is computed in less time compare to SHA512. SHA512 is more secure then SHA1.

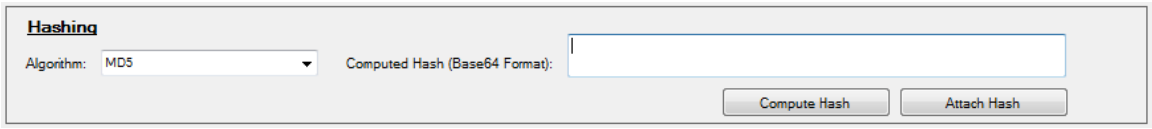


Figure 4-9: Hashing Interface

Message can be hashed via MD5, SHA1, SHA256, or SHA512. Then hashed value in bit format is encoded in Base64 in order to display it in printable characters and send across wire.

Generally different application uses multiple hash function to ensure integrity of messages. e.g. SHA1 and MD5 are common pair of use.

As shown in above Figure 6-6. When Attach button is clicked it will attach corresponding hash value with already generated SAML response.

4.4 Confidentiality

PCK system is commonly used for authentication and small messages encryption. It is 1000 slower then corresponding Secret Key Crypto system. It is the reason Secret key algorithms are used for encryption to provide confidentiality.

Figure 4-10: Secret Key Encryption System

4.5 REST Services

REST services are built using Microsoft WCF technique. This part can be further divided into two parts i.e. Service Hosting and Service Description.

4.5.1 Service Hosting

WCF services can be hosted in separate application, window service or in Internet Information Server (IIS) provided .NET framework 3.0 or later is installed. In this case Service is hosted in separate window service.

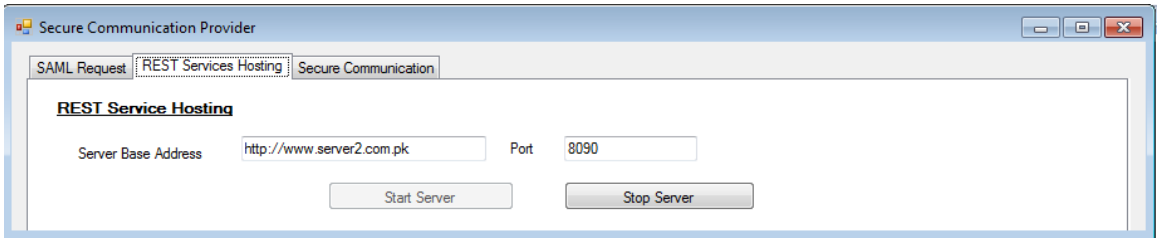


Figure 4-11: REST Services Hosting

When service is running at that specified address and port then its services can be accessed through web browser for testing easily. But whole testing is performed on local machine and all addressed are mapped to 127.0.0.1 so port being assigned should not collide. For testing whether service is working type following setting; for Server base Address is http://www.server2.com.pk and port is 8080.

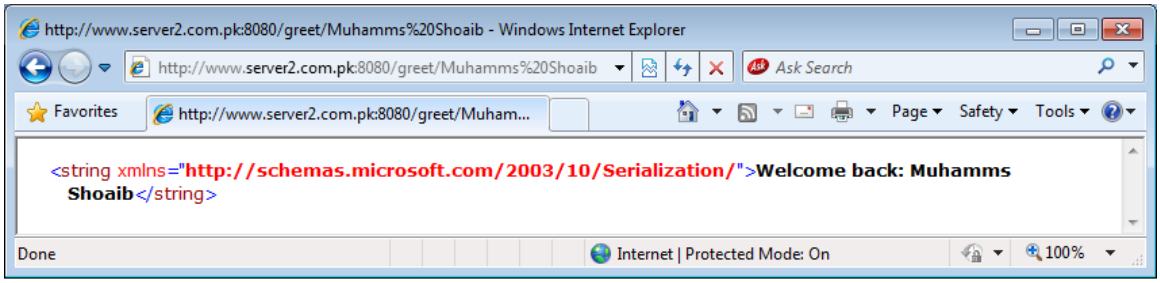


Figure 4-12: REST Service Greeting Message

Service status can also check in Services tab of Window Task Manager.

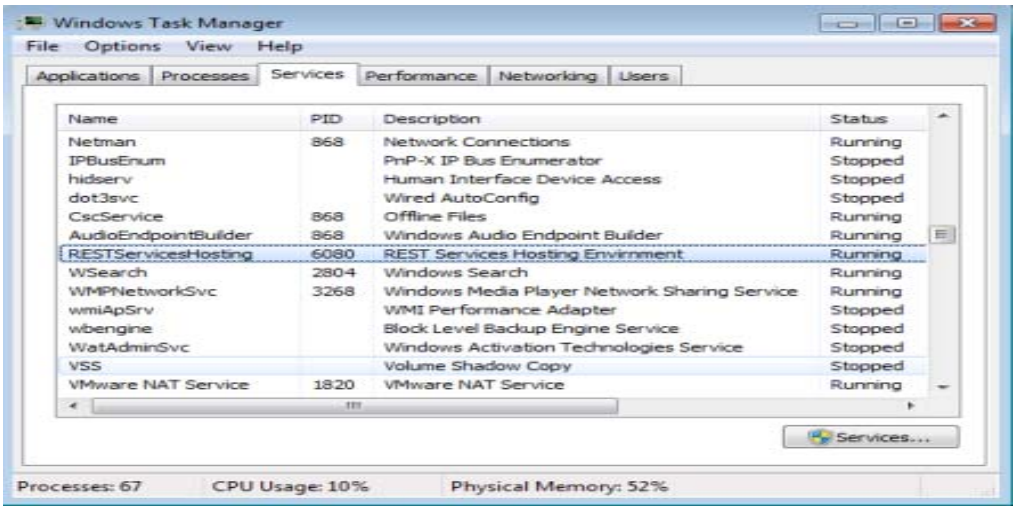


Figure 4-13: REST Service Status in Window Task Manager

4.5.2 Service Description

Current version of REST service implemented here only support GET operation and only few sample resources are modeled e.g. Student. Only relative URL are mentioned here with short description;

/Students	It will return all students collection stored.
/Students/Muhammad Shoaib	It will return single specified resource.
/	Another way of getting All students.
/Muhammad Shoaib	Another way of getting single specified resource.
/Students/Shoaib/?Age=20	All Student with name Shoaib and Age 20

Resources can be modeled in variety of ways as described in REST section. Beside GET operation other methods like POST, PUT, and DELETE can also be implemented similarly.

4.6 Hybrid Security Approach

In this scheme different available security standards are used. In order to main trust between enterprises it digital certificates (x509) are being used. Sender sign messages with his private key for authentication and integrity of message being sent out. It also use random session key for encryption and session key is encrypted using receiver's public key. Receiver uses his private key to decrypt session key and decrypt message using that session key. Message hash is validated against sender signature, if both match then message is received unaltered.

4.6.1 Trusted Third Party (TTP) Hosting

TTP is hosted in ISS for access remotely. When user send Assertion request this service will return a SAML assertion on valid request as shown in figure 5-6. Web server for hosting is IIS 7 as shown in figure 5-14.

4.6.2 REST Service Hosting

REST service is used to access modeled resources using standard HTTP methods. This service is hosted in separate window service that can be started and stop from user interface provided as shown in figure 5-11.

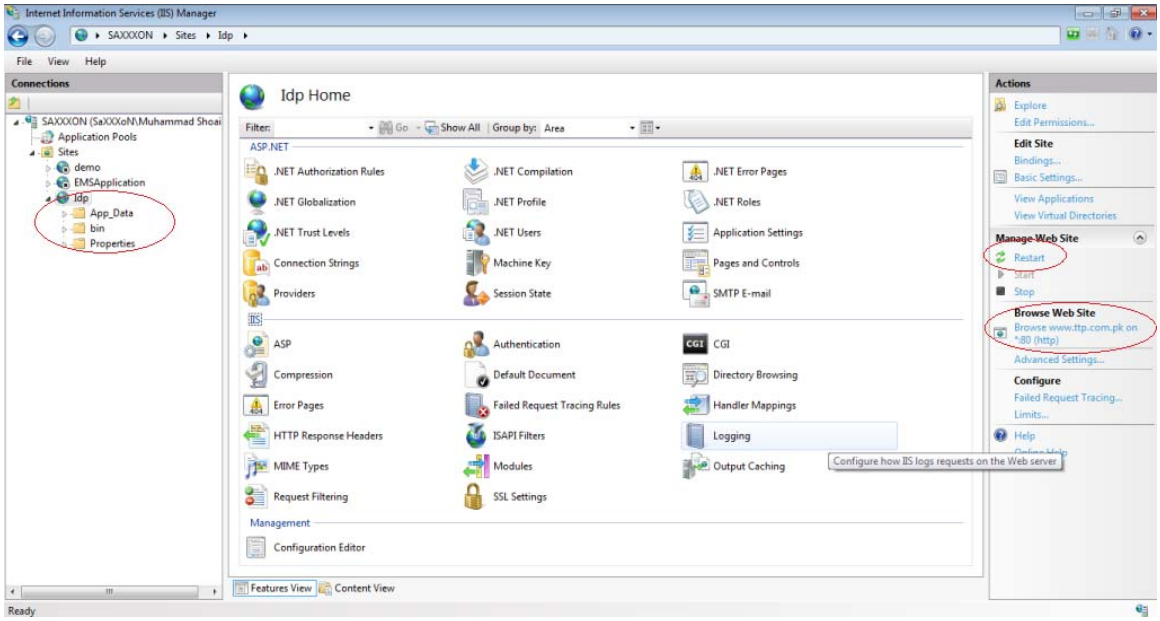


Figure 4-14: TTP Service Hosted in IIS7

4.6.3 Secure Communication

A hybrid approach is used for secure messaging over insecure channel. AES is used as secret key encryption scheme because it more secure and faster in both software and hardware implementations as compare to corresponding block ciphers. And SHA1 (hash length with 160 bits) is used for digital signatures. Digital certificates are used for maintain trust across corporate.

REST response is generated in specific format. On client side format is tear off to separate different parts of information. Traffic travelled over in secure channel i.e. http protocol, security is provided via end to end process without intervention of third party server involvement. REST Service sends back response in format shown below. Content is encoded in Base64 format to transmit across wire.

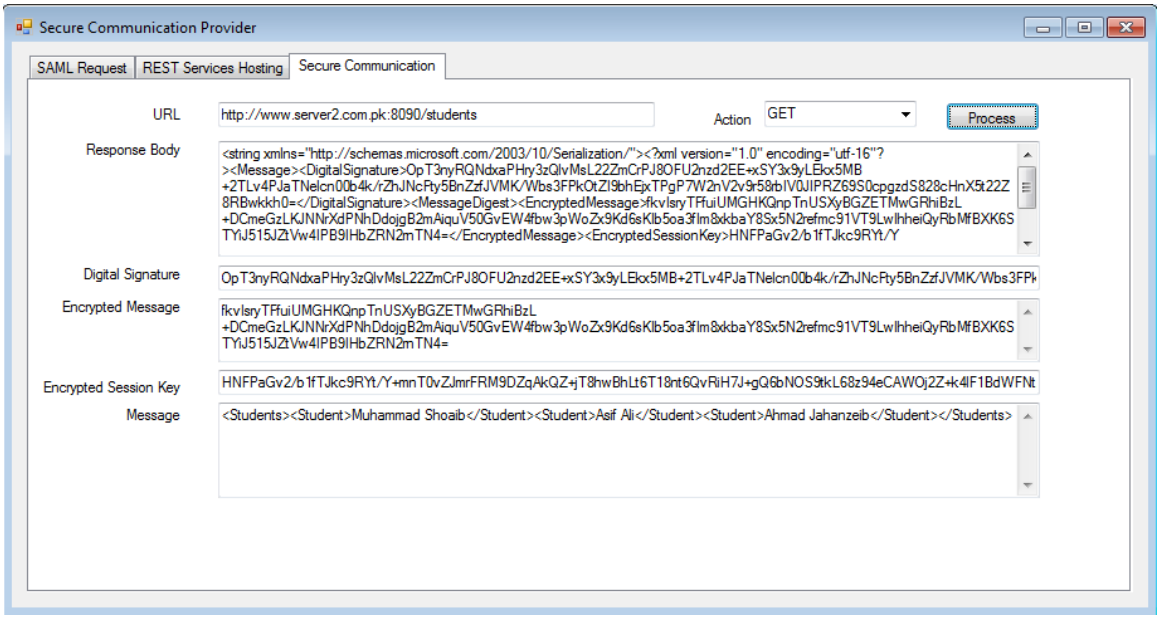


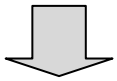
Figure 4-15: Hybrid Scheme for Secure Communication

```
<Message>

<DigitalSignature></DigitalSignature>

<MessageDigest>
        <EncryptedMessage></EncryptedMessage>
        <EncryptedSession></EncryptedSession>
</MessageDigest>

</Message>
```




```
<Message><DigitalSignature>OpT3nyRQNdxaPHry3zQlvMsL22ZmCrPJ8OFU2nzd2EE+xSY3x9yLEkx5MB+2TLv4PJaTNelcn00b4k/rZhJNcFty5BnZzfJVMK/Wbs3FPkOtZI9bhEjxTPgP7W2nV2v9r58rbIV0JIPRZ69S0cpgzdS828cHnX5t22Z8RBwkkh0=</DigitalSignature><MessageDigest><EncryptedMessage>fkvIsryTFfuiUMGHKQnpTnUSXyBGZETMwGRhiBzL+DCmeGzLKJNNrXdPNhDdojgB2mAiquV50GvEW4fbw3pWoZx9Kd6sKIb5oa3fIm8xkbaY8Sx5N2refmc91VT9LwIhheiQyRbMfBXK6STYiJ515JZtVw4IPB9IHbZRN2rnTN4=</EncryptedMessage><EncryptedSessionKey>HNFPaGv2/b1fTJkc9RYt/Y+mnT0vZJmrFRM9DZqAkQZ+jT8hwBhLt6T18nt6QvRiH7J+gQ6bNOS9tkL68z94eCAWOj2Z+k4lF1BdWFNt/KW3M1Swakfy GafLO/TAuZNxIy3jBIx+3mK4kyFpU61Tv1u0GuOoNAlfkieQAxFslf8=</EncryptedSessionKey></MessageDigest></Message>
```

Chapter 5. **Performance Analysis**

5.1 **Overview**

First of all performance of some well known symmetric schemes and hashing are discussed. After that overall performance of our methodology is briefly described.

5.2 **Symmetric Schemes and Hashing**

All the communication is made over unsecure channel that is HTTP. It is obvious to make sure message security we need some mechanism. In our hybrid scheme symmetric, public key and hashing all being applied. Different simulations are run with same data size and average of those values is then taken.

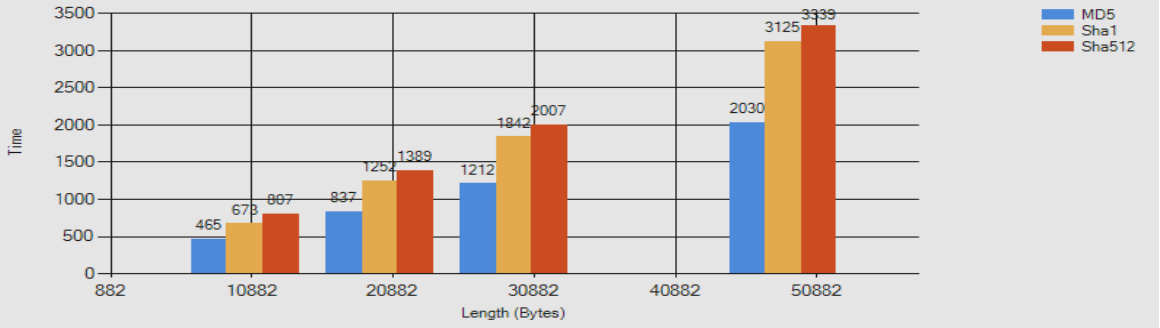


Figure 5-1: Hashing Schemes Comparison

MD5 has digest size of 128 bits smaller then corresponding Sha scheme. Its performance is better as compare to two variations of Sha with 160 and 512 bits digest size. As size of digest increase computation requires to compute the result also increase. But size of digest determines the effort required to find collision. Smaller the size more chances are there for collision to occur. So MD5 is more unsecure then corresponding Sha512 because of digest space.

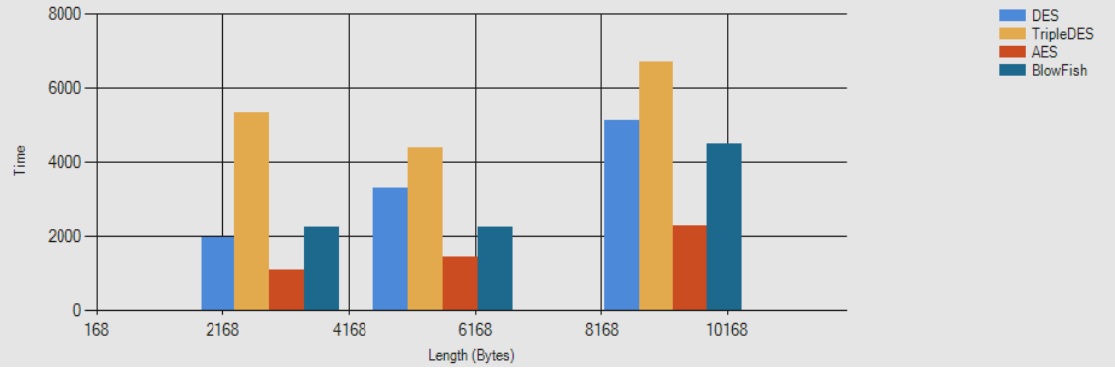


Figure 5-2: Secret Key Encryption Schemes Comparison

AES is out performing in almost all cases. It is based on special finite fields to compute the encrypted text. It is designed to work in both software and hardware in the same way. DES has as data size increase, performance difference between DES and Triple DES is obvious as compare to AES. Triple DES is most computation hungry in all cases.

5.3 Performance of REST VS SOAP

Two services are built one over REST and other over SOAP. Services are hosted in separate window services to avoid the need of installation of IIS or any web server dependency.

We used fiddler as HTTP debugging utility in order to intercept and evaluate requests and response sizes and others features. Fiddler is shown in [figure 5-3]. By using Fiddler we intercept all HTTP requests and responses sent by any application. We can change response and request and replay them later as well. Fiddler is useful utility for monitoring other features like content type, response time, compression techniques and security information like certificates etc.

We have modeled some resources like Persons and Examination for testing purposes. When we send some resource using REST service then response is sent back over HTTP as shown in [Figure 5-4].

When same resource is requested via SOAP service the bandwidth consumption is more than corresponding REST service [Figure 5-5]. Performance table is given below for number of requests.

Serial #	Type	Resource	Request Size	Response Size
1	SOAP	/Persons/Shoaib	398	686
2	REST	/Persons/Shoaib	61	391
3	SOAP	/Persons/Khan	420	686
4	REST	/Persons/Khan	59	391
5	SOAP	/Persons/Shahid	398	688
6	REST	/Persons/Shahid	61	391
7	SOAP	/Examination	457	783
8	REST	/Examination	59	646

Table 1: REST VS SOAP Performance

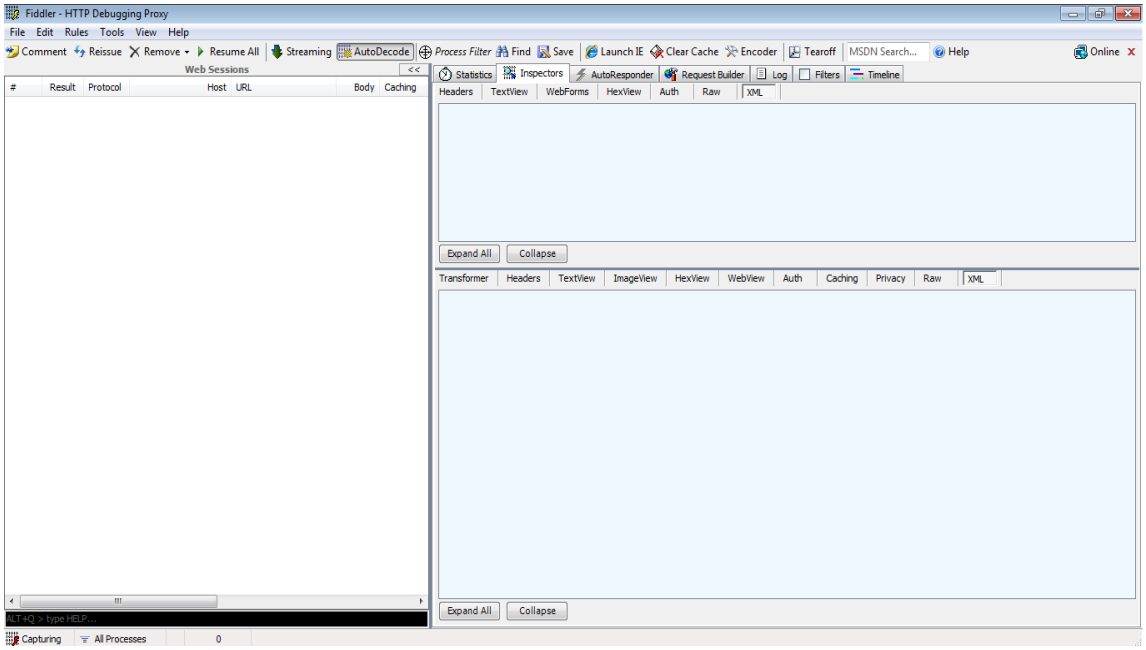


Figure 5-3 : Fiddler Startup Screeen

5.4 Conclusion and Future Research

Secure communication between services and applications in ubiquitous environment is one of the essential requirements. Such environment operates and coordinates with each other without intervention of any third party. Devices and services access each other and authenticate each other seamlessly.

In this work we proposed security mechanism using REST services and some well known secret key and PKC in pervasive environment. Trusted Third Party (TTP) is responsible for user's assertions and provides trust services between different services,

applications or users. Services providers just look for valid assertion and upon authorization of requested resource give access to it.

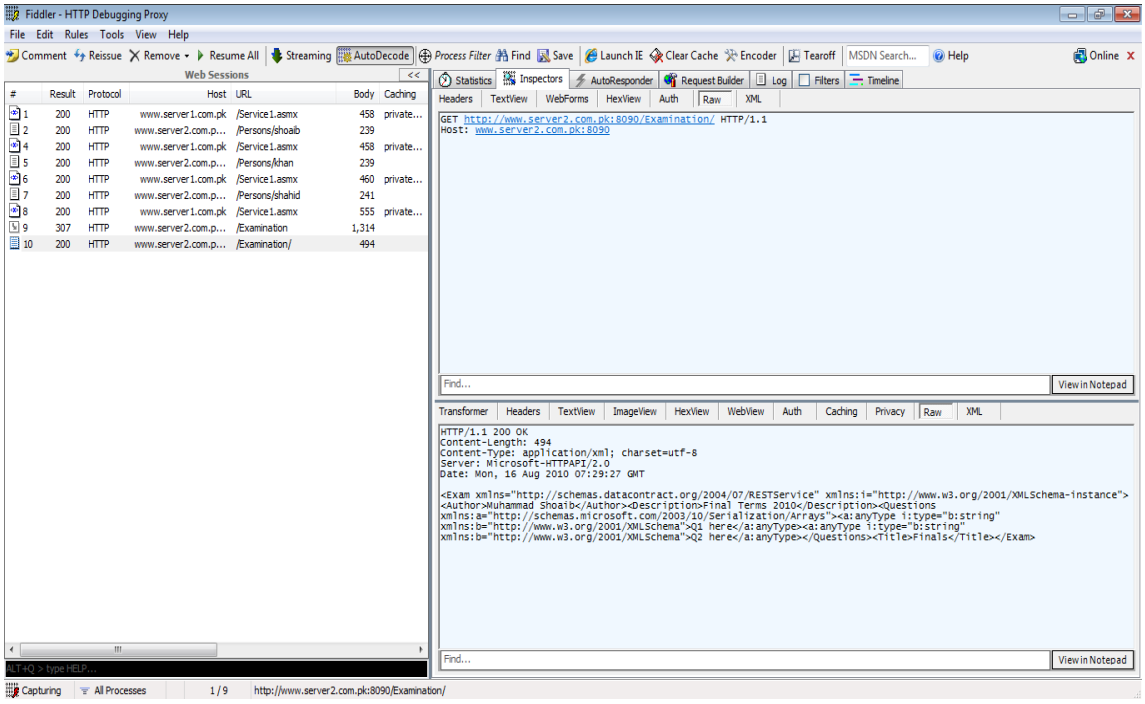


Figure 5-4 REST Service Response in Fiddler

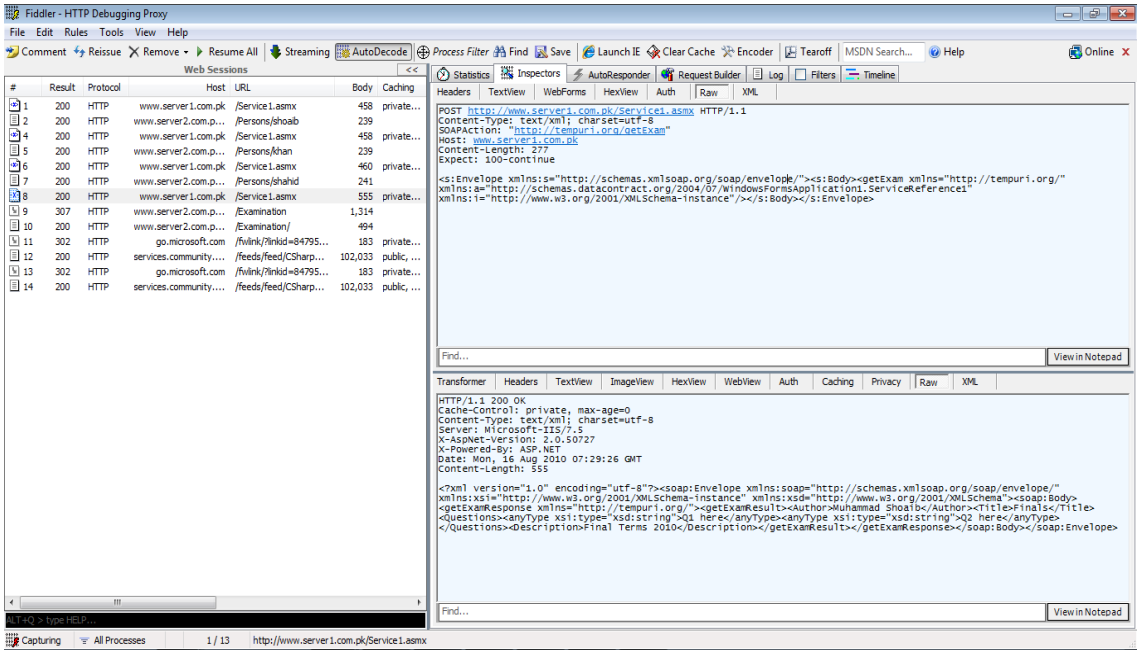


Figure 5-5: SOAP Service Response in Fiddler

One of the major advantages of this approach is that business logic is separate from the access control mechanism. It is less risky now to communicate with remote services and when these services demands identity of user assertion is posted instead of actual user credentials. Application can compose of several vendors' services with one or more Identity Providers. SAML is used for transfer of identities and other user credentials. All the communication takes place over simple HTTP protocol that is most widely used and supported. Resources are modeled and accessed via uniform URLs. As HTTP support number of methods but most important among all are GET, POST, DELETE, PUT. These methods are used to access, manipulate, delete and create new resources.

REST services are preferred over SOAP services because these are light and need no toolkit to deal. In the later case small response is usually ten times of former one and obviously it is not suitable in the case of pervasive computing. We have limited resources like memory, processor and bandwidth etc.

In order to main trust between enterprises it digital certificates (x509) are being used. Sender sign messages with his private key for authentication and integrity of message being sent out. It also use random session key for encryption and session key is encrypted using receiver's public key. Receiver uses his private key to decrypt session key and decrypt message using that session key. Message hash is validated against sender signature, if both match then message is received unaltered.

AES is used as secret key encryption scheme because it more secure and faster in both software and hardware implementations as compare to corresponding block ciphers. And SHA1 (hash length with 160 bits) is used for digital signatures; it is also possible to use multiple hashing algorithms for more security.

References

[1] Nazir, A. M., Tomlinson, A. and Javed, M. Y., "Web-Services Architecture for Pervasive Computing Environment", Pakistan Journal of Sciences, Vol. 61, No 3, September 2009

[2] Nazir, M., Umar, M. and Javed, M.Y., "Future Challenges in Context-Aware Computing", Proceedings of the IADIS International Conference on WWW/Internet 2007, Volume-II, October 5-8, 2007

[3] C. F. Sorencen. A Comparison of Distributed Object Technology, Norwigian university of Science and Technology

[4] G.Brose. Securing Web Services using SOAP Security Proxies

[5] P. H. Baker, C. Kaler, R. Monzillo, A. Nadalin. (2004). Web Services Security: SOAP message security (v1.0), OASIS

[6] M. P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann. Service-Oriented Computing: State of the Art and Research Challenge

[7] P. H. Baker, C. Kaler, R. Monzillo, A. Nadalin. (2004). Web Services Security: SAML Token Profile, OASIS

[8] S. Cantor, J. Kemp, R. Philpott, E. Maler. (2009). Assertions and Protocols for OASIS Security Assertion Markup Language (SAML) V2.0, (Working Draft 06), OASIS

[9] H. Lockhart, T. Wisniewski, S. Cantor, P. Mishra. (2005). Metadata for Security Assertion Markup Language (V2.0), OASIS

[10] J. Hughes, S. Cantor, J. Hodges, F. Hirsch, P. Mishra, R. Philpott, E. Maler. (2009). Profiles for OASIS Security Assertion Markup Language (SAML) V2.0, (Working Draft 06), OASIS

[11] M. S. Mokbel, and J. Jiajin. Integrated Security Architecture for Web Services and its Challenges, Journal of Theoretical and Applied Information Technology

[12] D. Szepielak. REST-based Service Oriented Architecture for Dynamic Integrated Information Systems

[13] W. Ford, P. Hallam-Baker,B. Fox, B. Dillaway, B. LaMacchia, J. Epstein, J. Lapp. (2001). XML Key Management Specifications (XKMS 1.0), W3C Recommendation

[14] M. Bartel, J. Boyer, B. Fox, B. LaMacchia, E. Simon. (2008). XML Signature Syntax and Processing (2nd Edition), W3C Recommendation

[15] R. J. Ray, P. Kulchenko. (2002). Programming Web services with Perl (1st Edition), O'Reilly Media

[16] Pervasive computing, Available: http://searchnetworking.techtarget.com/sDefinition/0,,sid7_gci759337,00.html

[17] Services Oriented Architecture, Available: http://en.wikipedia.org/wiki/Service-oriented_architecture

[18] Block Cipher modes of Operations, Available: http://en.wikipedia.org/wiki/Block_cipher_modes_of_operation