T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

SERVİS YÖNELİMLİ MİMARİLERDE GÜVENLİK ÇÖZÜMLERİ İÇİN WEB SERVİS TEMELLİ BİR ALT YAPI MODELİ

YÜKSEK LİSANS TEZİ

Bil.Müh. Deniz DURAL

Enstitü Anabilim Dalı : BİLGİSAYAR VE BİLİŞİM MÜHENDİSLİĞİ

Tez Danışmanı : Prof. Dr. Ümit KOCABIÇAK

Mayıs 2012

T.C.

SAKARYA ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

SERVİS YÖNELİMLİ MİMARİLERDE GÜVENLİK ÇÖZÜMLERİ İÇİN WEB SERVİS TEMELLİ BİR ALT YAPI MODELİ

YÜKSEK LİSANS TEZİ

Bil.Müh. Deniz DURAL

Enstitü Anabilim Dalı : BİLGİSAYAR VE BİLİŞİM MÜHENDİSLİĞİ

Bu tez 23 / 05 /2012 tarihinde aşağıdaki jüri tarafından Oybirliği ile kabul edilmiştir.

Prof.Dr. Ümit KOCABIÇAK	Doç.Dr. A. Turan ÖZCERİT	Doç.Dr. Cemil ÖZ
.....	.....	.....
Jüri Başkanı	Üye	Üye

TEŞEKKÜR

Projenin gerçekleştirilme aşamasında kaynak sıkıntısı çekilmemiştir. Çok sayıda doküman, kaynak materyal, örnek proje ve programlardan yararlanılmıştır. Bu konu üzerinde ileriki zamanlarda, çalışma ve araştırma yapmak isteyecek arkadaşlara büyük yardımı olacağına inandığım bu projeyi hazırlarken bana her konuda yardım eden, desteğini hiç esirgemeyen, değerli fikirleriyle yol gösteren değerli hocam Prof. Dr. Ümit KOCABIÇAK'a ve bana sabırla katlanan tüm mesai arkadaşlarıma ve canım aileme teşekkürü bir borç bilirim.

İÇİNDEKİLER

SİMGELER VE KISALTMALAR LİSTESİ

AES	:Advanced Encryption Standard
ASP	: Active Server Pages
A2A	: Application to Application
API	: Application Programming Internet
BEEP	: Blocks Extensible Exchange Protocol
BPEL	: Business Process Execution Language
CGI	: Common Gateway Interface
DES	: Data Encryption Standard
FTP	: File Transfer Protocol
GoF	: Gang of Four
H2A	: Human to Application
HTML	: Hypertext Markup Language
HTTP	: HyperText Transfer Protocol
IETF	: Internet Engineering Task Force
ISAPI	: Internet Server Application Program Interface
IPSec	: Internet Protocol Security
JSP	: Java Server Pages
OASIS	: Advancing Open Standards for the Information Society
PKI	: Public Key Infrastructure
RC2	: Rivest Cipher Two
SAML	: Security Assertion Markup Language
SMTP	: Simple Mail Transfer Protocol
SOA	: Service Oriented Architecture
SOAP	: Simple Object Access Protocol
SSL	: Secure Sockets Layer
TCP/IP	: Transmission Control Protocol / Internet Protocol
TLS	: Transport Layer Security
TripleDES	: Triple Data Encryption Standard

UDDI	: Universal Description, Discovery and Integration
VPN	: Virtual Private Network
WSA	: Web Service Architecture
WSDL	: Web Service Description Language
WSGA	: Web Servis tabanlı Güvenlik Altyapısı
WSS	: Web Service Security
W3C	: World Wide Web Consortium
WWW	: World Wide Web
XACML	: Xml Access Control Markup Language
XKMS	: Xml Key Management Specification
XML	: Extensible Markup Language

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

ÖZET

Anahtar kelimeler: Servis Yönelimli Mimari(SOA), Dağıtık Mimari, Web Servisleri, Güvenlik

Günümüzde servis yönelimli mimariler, oluşturulan standartlar ve web servislerinin yaygın olarak kullanılmaya başlanmasıyla birlikte yazılım dünyasında daha fazla yer tutmaya başlamıştır. Yeni nesil dağıtık uygulamaların önemli bölümünü oluşturan web servisleri, servis yönelimli mimariye geçişte en çok tercih edilen yöntemdir. Servis yönelimli mimarinin doğası gereği birçok heterojen sistemin birlikte çalışması sırasında ortaya çıkan güvenlik sorunları, gerek akademik, gerekse ticari çalışmalarda üzerinde durulan önemli konular haline gelmiştir.

Her yapılan çalışma, kendi talepleri doğrultusunda farklı güvenlik çözümleri içermektedir. Ancak servis yönelimli mimariye geçişte, farklı çalışma alanları ile iletişim ihtiyacı ortaya çıkmaktadır. Bu tez çalışmasında, bu tarz durumlara imkân sağlamak amacıyla var olan güvenlik yaklaşımlarının hepsinin üzerinde kullanılabildiği bir güvenlik alt yapı modeli önerilmiştir. XML web servislerinin güçlü özelliklerinden yararlanılan bu modelin temel amacı, bağımsız platformların birbirleri ile iletişimleri noktasında güvenliği tek bir yere toplamak ve sistemin yönetimini merkezileştirmektir.

AN INFRASTRUCTURE MODEL BASED ON WEB SERVICES FOR SECURITY SOLUTIONS IN SERVICE ORIENTED ARCHITECTURE

SUMMARY

Key Words: Service Oriented Architecture (SOA), Distributed Architecture, Web Services, Security

Today, service oriented architectures which its popularity has increased nowadays, have started to become more important in software world because of some generated standards and starting to use of web services extensively. Web services which form an imported part of new generation distributed applications have become most preferred method while transition to the service oriented architecture. Due to the nature of service oriented architecture, many safety problems have become important issue that have been emphasized by both academic and commercial studies, resulting from heterogeneous systems work together.

Many studies contain different security solutions according to their demands. But, while transition to the service oriented architecture, there is a need to communicate with different frameworks. In this thesis study, in order to provide to these situations, a security infrastructure model that all security approaches can be used in, is offered. Main objective of model that is utilized from powerful features of XML web services is that collecting security one point during communication independent platforms with each other and centralizing management of the system.

BÖLÜM 1. GİRİŞ

Günümüzde web ortamının basit ve yaygın olarak kullanılmasından dolayı, bütün çalışma alanlarında yeri ve önemi çok büyümüştür. Web üzerindeki bu gelişmeler, HTTP protokolü ile HTML dilinde biçimlendirilmiş statik belgelerin kullanıcılara sunulması web sayfaları olarak karşımıza çıkarken, işletmelerin müşterilerine web üzerinden bazı iş süreçlerini yaptırma gereksinimi sonucunda ortaya çıkan, sunucu tarafında çalışan programlar web uygulamaları olarak tanımlanmıştır. İşletmelerin diğer işletmeler ile olan iş süreçlerini bütünleştirme gereksinimi sonucu ortaya çıkan ve gelişmekte olan yapı ise web servisleri olarak anılmaktadır.

Web servisleri, web ortamında yayınlanabilen, aranıp bulunabilen ve çağrılarak erişilebilen modüler uygulama fonksiyonlarıdır. Birçok yazılım firması tarafından yoğun bir destek bulan bu modelin uygulama bütünleştirmesi konusunda ortaya çıkacak ortama hâkim olacağı yönünde görüşler oldukça fazladır. Web servisleri farklı sistemlerde, farklı dillerle yazılmış uygulamalarda entegrasyonu ve dolayısıyla platformdan bağımsız iletişimi sağlar. Orta katman bileşenlerinin web servisleri biçiminde hazırlanıp direk kullanıcı ara yüzünden çağrılması birçok güvenlik sorununun çözümüdür. Kapsülleme özelliği sayesinde ise karmaşıklık web servislerinin içinde kalmaktadır.

Son zamanlarda servis yönelimli mimari denildiğinde akla web servisleri gelmektedir ki bu da servis yönelimli mimarinin (SOA) web servis teknolojisinin avantajlarından yararlanıyor olması anlamına gelir. Servis yönelimli mimari, özellikle tek merkezli olmayan yazılımların (örneğin kurumsal yazılımlar) tasarım mimarisi olarak anılır.
Birçok kurum ve firmanın, SOA ve web servis teknolojilerinden yararlanıyor olması kaçınılmaz bir hal almıştır. Ancak kurumları servis yönelimli mimariye geçiş aşamasında düşündüren en önemli konu güvenlik konusudur. Çünkü birden fazla, birbirinden tamamen bağımsız ve farklı birçok servisin yer aldığı bir mimaride

güvenliğin sağlanması zor bir durumdur. Her kurumun kendine ait bir güvenlik altyapısı ve kriterleri bulunur. Kendi içlerinde çok güvenli bir ortam sağlanıyor olsa dahi, birden çok platformun bulunduğu ortamda güvenlik açıkları ve kullanım zorlukları yaşanabilmektedir. Bu güvenlik problemleri birçok akademik çalışmaya da konu olmuştur. Aşağıda bu çalışmalarla alakalı bilgiler verilmiştir.

[1] numaralı akademik çalışmada, akademik enstitülerde, öğrenciler, çalışanlar ve yöneticiler için öğrenci kayıt servisi, harç ödeme servisi, ders içerikleri servisi ve kişisel servis gibi web servislerine çoklu güvenlik sağlayan çok katmanlı korunmuş bir mimari (multi-level secured Architecture-MLSA) önermişlerdir. Sistemde kullanılan, farklı coğrafi bölgelerde bulunan çeşitli akademik enstitülerle ve kullanıcılar arasında bir aracı gibi davranan IWMS (Integrated Web Services Manager) sayesinde, güvenli ve bütünleşmiş akademik bir ortam sağlanmıştır. Oluşturulan sistemde, güvenlik konusunda önemli sorunlardan kimlik belirleme, güvenilirlik ve bütünlük sağlanmıştır.

[2] numaralı akademik çalışmada, öncelikle kullanılan uygulamaları içine alan bir web servis kümesi oluşturulmuştur. Bu servis kümesinin erişim kontrolü ve güvenilir iletişim için web servis güvenlik çalışma alanı (WSSF) adı verilen bir çözüm sunulmuştur. Mesaj bütünlüğünü ve güvenilirliğini sağlamak için SOAP mesajına bir güvenlik jetonu eklenerek web servislerine güvenilir bir iletişim sağlandığından, nitelik tabanlı rol atamaları için dinamik izin tanımlamaları ile yetkilendirme yapılabildiğinden bahsedilmiştir.

[3] numaralı akademik çalışmada, web servis güvenliği ve bağlantılı güvenlik standartların karşılaştığı güvenlik sorunlarını analiz etmiştir. Güvenli web servisleri için kimlik belirlemesi yetkilendirme, bütünlük ve güvenilirlik mekanizmalarının kullanımını temel alan bütünleşmiş bir güvenlik çalışması (Integrated Security Framework) tasarlanmış ve yapılan saldırılara karşı duran web servisleri oluşturmak için bu güvenlik mekanizmaları nasıl birleştirilip, tanımlanabildiğinden bahsetmiştir.

[4] numaralı akademik çalışmada, kullanıcıların kimliğinin ve yetkilerinin belirlenmesi için, SSO (Single-Sign On) kavramını kullanarak, web servislerinin

nasıl kullanıldığını göstermiştir. Kullanıcıların tek bir giriş ile bütün servislere erişebilmeleri sağlanmıştır. Farklı ve dağıtık web uygulamalarının birleştirilmesi için SSO kavramının nasıl bir kolaylık sağladığından ve onun güvenliğinden bahsetmiş, farklı platformlarda servisler geliştirilerek, SSO güvenliğine sahip, birlikte çalışabilen bir mimari oluşturmuştur.

Bu tez çalışmasında, yapılan çalışmalardan farklı olarak, standart web servis güvenlik fonksiyonları ayrı web servisleri şeklinde tasarlanmış uçtan uca güvenlik amaçlanan bir altyapı modeli oluşturulmuştur. Bu model sayesinde birden fazla farklı sisteme erişim sırasında güvenlik sağlanmaya çalışılmıştır. Bu tezde, yapılan çalışmalardan farklı olarak, modelin tasarımı sırasında GoF tasarım desenlerinden Facade tasarım deseni mantığından yararlanılmıştır, sistem içerisinde yer alan ve güvenlik için kullanılan servislerin birbirleri ile olan bağlantıları bu desenin çalışma şekline dayandırılmıştır. Ayrıca farklı olarak, model üzerinde web servislerinde kullanılan her türlü güvelik yöntemi uygulanabilmektedir. Böylece, web servislerinin kullanılabildiği her türlü uygulamaya adapte olabilecek şekilde tasarlanan bu alt yapı sayesinde, her kurumun kendi güvenlik yöntemlerini değiştirmeden model üzerinde kullanabilmesi amaçlanmıştır.

Bu tez çalışması 7 bölümden oluşmaktadır. 2. bölümde web servisi kavramı ayrıntılı olarak anlatılmıştır. Bir web servisinin ne olduğu, temel özellikleri, kullandığı teknolojiler ve çalışma yapısı hakkında detaylı bir bilgi sunulmaktadır. Ayrıca konuyla ilgili verilen bilgiler, şekiller ve tablolar ile desteklenmiştir. Tezin 3. bölümünde web servis güvenliğinden bahsedilmiştir. XML ile web servis birlikteliği anlatılmış, XML güvenliğinden ve web servis güvenlik standartları ve kullanımları hakkında bilgi verilmiştir.

Tezin 4. bölümünde web servis güvenlik altyapı modelinin (WSGA) özelliklerinden, modelde bulunan kavramlardan ve modelin güvenlik yığını içerisindeki yerinden bahsedilmiştir. Ayrıca tasarlanan mimarinin bölümleri ve çalışma şekilleri detaylı olarak anlatılmıştır. 5. bölümde modelde kullanılan teknolojilerden ve modelin uygulamasından bahsedilmiştir. Modelde kullanılan servis projeleri, sınıfları ve diyagramları gösterilmiştir.

Tezin 6. bölümünde test ve performans ölçümleri yapılmış, piyasadakilerle karşılaştırılmıştır. 7. bölüm ise tezin sonuç kısmını kapsamaktadır. Bu kısımda uygulamanın bize neler kattığı ve uygulamadan çıkarılan sonuçların neler olduğu anlatılmıştır.

BÖLÜM 2. WEB SERVİSLERİ

2.1. Web Servislerine Giriş

Akademik dünyada çok öncelerden kullanılmaya başlansa da, internetin başarı hikâyesi 90'lı yılların başında başlamıştır. İnternetin başarısının asıl nedeni, standart internet protokollerini ve çeşitli platformlarda tanımlanan tarayıcıların sağladığı basit bir veri erişim protokolünü kullanarak herhangi bir yerden bilgiye erişimde kolaylık sağlayan temek yenilik olan World Wide Web dir. WWW'nin yayılmasıyla birlikte, internet ve ilgili teknolojiler dünyanın her yerindeki bilgisayarlara bağlanmak için fiili bir standart haline gelmişlerdir.

İnternetin yaygınlaşmasıyla birlikte, internet tarafından ortaya konulan altyapının bir tarayıcı kullanarak yalnızca bilgi almak için (human-to-application, H2A olarak adlandırılmaktadır.) kullanılamayacağı açıkça belli olmuştur. Aksine, var olan teknolojileri kullanarak uygulamadan uygulamaya (application-to-application, A2A) iletişime talep artmıştır. Ve mevcut protokollerin bu amaç için kullanılması umut edilmiştir. Ancak kısa sure içerisinde bunun böyle olmadığı anlaşılmıştır. Çünkü HTTP, hypertext ile birbirine bağlanmış olan belgelere dayanan basit bir veri erişim yolunu izleyerek bilgiye erişmek olarak tasarlanmıştır. Sunucular ve son kullanıcılar arasında bilgilerin nasıl aktarılacağına dair kurallar ve yöntemleri düzenleyen sistemdir. Bu protokol, A2A senaryolarından ortaya çıkan karmaşık işlemler için uygun değildir. Ve mevcut protokollerin birçoğu bunun için kullanılamaz, çünkü web dünyası için uygun değillerdir ya da çok sınırlayıcılardır.

1999' ların sonunda, Microsoft XML tabanlı,A2A senaryolarında kullanılabilen, SOAP adı verilen bir protokol yayınlamıştır. Birçok protokol önerisinde bulunmuş olsa da, 2000'lerin başında IBM de SOAP'u desteklemeye başlamıştır. Bu noktada, SOAP, karmaşık A2A senaryolarını gerçekleyen bir protokol haline gelmiştir.

Ancak, popülerliğinin gitgide artmasıyla, SOAP kullanarak uygulanan servislerin daha iyi tanımlanması ve bulunması için bir gereklilik ortaya çıkmıştır. Web servisleri terimi, IBM, Microsoft ve Ariba'nın ortaklaşa web servis tanımlama dilini (WSDL) yayınlamalarından birkaç ay sonra icat edilmiştir. Son olarak UDDI'ın ortaya çıkmasıyla birlikte, web servislerin temelini oluşturan standartlar ve protokoller tamamlanmıştır. İlerleyen yıllarda, bu teknolojinin nasıl geliştirilebileceği hakkında birçok görüş ortaya atılmıştır [5].

İnternet üzerindeki sistemler düşünüldüğünde çok sayıda donanım, bu donanımlar üzerinde kurulu çok çeşitli platformlar ve bu platformlar üzerinde çalışan farklı amaçlı yazılımlar bulunmaktadır. Bu çeşitlilik göz önüne alındığında, sistemler arasındaki iletişim esnasında çok sayıda problem baş göstermektedir. Bu tarz problemleri çözmenin en tabii yolu belirli standartlar oluşturmak ve sistemlerin birbirleriyle iletişimini bu standartlar üzerinden sağlamaktır. Bu noktada web servisleri farklı uygulamalar arasında iletişim kurmak için yeni bir yöntem sağlamaktadır. Web servisi dendiğinde farklı birçok kurumun ve çeşitli kitapların ortaya koyduğu pek çok tanım bulunmaktadır.

Öncelikle W3C (World Wide Web Consortium) tarafından yapılan resmi tanımıyla web servisleri, bilgisayarlar arasında ağ üzerinden etkileşimi ve uyumluluğu sağlayacak yazılım sistemleridir. Günümüzde birbiriyle haberleşecek sistemleri gerçeklemek için en çok tercih edilen yöntem web servisleridir [6].

Bir web servisi, standartlaşmış XML mesajlaşma sistemini kullanan ve internet üzerinde ulaşılabilir olan yazılım parçalarıdır. Bir web servisine her türlü iletişim için, XML kodlama kullanılır. Örneğin, bir istemci web servisine XML bir mesaj göndererek istekte bulunur ve ardından ilgili XML cevap için bekler. Çünkü bütün iletişim XML üzerinedir, web servisleri hiçbir işletim sistemine ya da programlama diline bağlı değildir. Yani Java ile Perl ya da Windows uygulamasıyla Unix uygulaması konuşabilmektedir.

Web servisleri, internet üzerinde tanımlanan, yayınlanan, bulunan ya da çağırılan, modüler, dağıtık, dinamik uygulamalardır. Bu uygulamalar lokal, dağıtık ya da web

tabanlı olabilirler. Web servisleri TCP/IP, HTTP, Java, HTML ve XML gibi açık standartlar üstüne kuruludur [7].

Bir web servisi, uygulamalar ve sistemler arasında veri alışverişi için kullanılan açık standartlar ve protokoller topluluğudur denilebilir. Çeşitli programlama dillerinde yazılan ve çeşitli platformlar üzerinde çalıştırılan yazılım uygulamaları, internet gibi büyük bilgisayar ağları üzerinde veri alışverişi yapmak için web servislerini kullanabilirler. Asıl fikir, servisleri internet üzerinde dağıtmak ve onları istemciler için ulaşılabilir hale getirmektir. Yani haberleşecek sistemlerin birbirlerinden haberdar olması veya platformlarının uyumlu olması gerekmez. XML Web Servislerine farklı bir bilgisayar ve farklı bir platformdan istemci olunabilir. Bu birlikte çalışabilirlik özelliği (Java ve Python ya da Windows ve Linux uygulamaları gibi) açık standartların kullanımı sayesindedir [8].

2.2. Web Servis Modeli

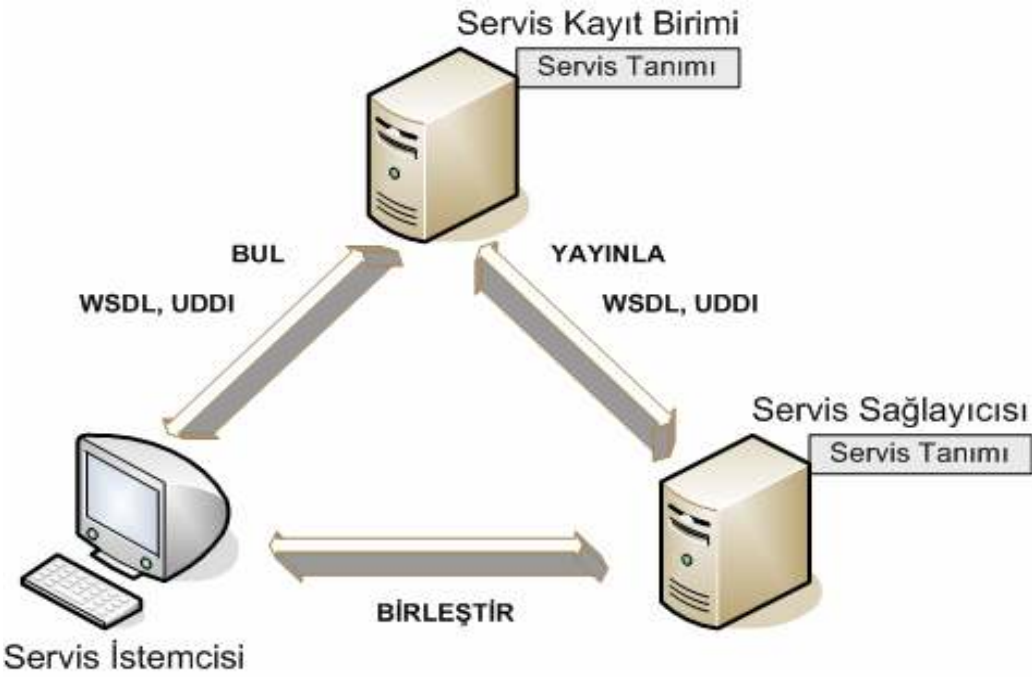
Web servisleri modeli üç temel birimin, servis sağlayıcı, servis kayıt birimi ve servis istemcisi, arasındaki etkileşimler üzerine kuruludur. Etkileşimler yayınla, bul ve bağlan işlemlerini içerir.

Web servisleri modeli üç temel birimin etkileşimine dayanır. Şekil 2.1'de gösterilen bu birimler şunlardır:

- Servis Sağlayıcısı (Service Provider): Servis sağlayıcı istemcilerin sağlayıcıda bulunan servislere erişimini sağlar. Servis sağlayıcı kendi sitesinde bulunan web servisleri tanımını servis kayıt birimine (service registry) kaydederek bu servisinin nasıl çağrılacağını belirtir.

- Servis İstemcisi (Service Requester): Servis sağlayıcısında bulunan web servislerini çağırarak kullanan istemci uygulamalarıdır. Web servisinin nasıl çağrılacağını belirler ve ilgili parametreleri servis kayıt biriminden arayarak bulur ve çağırır.

- Servis Kayıt Birimi: Servis sağlayıcılarının yayınladıkları web servisi tanımlarını saklar ve aranıp bulunmasını sağlar. Servis sağlayıcıları servis kayıt birimini tarayarak istediği servisler hakkında bilgi alabilir. Servis kayıt birimi her servisin nasıl çağırılacağı konusunda tanım bilgileri içerir [1].

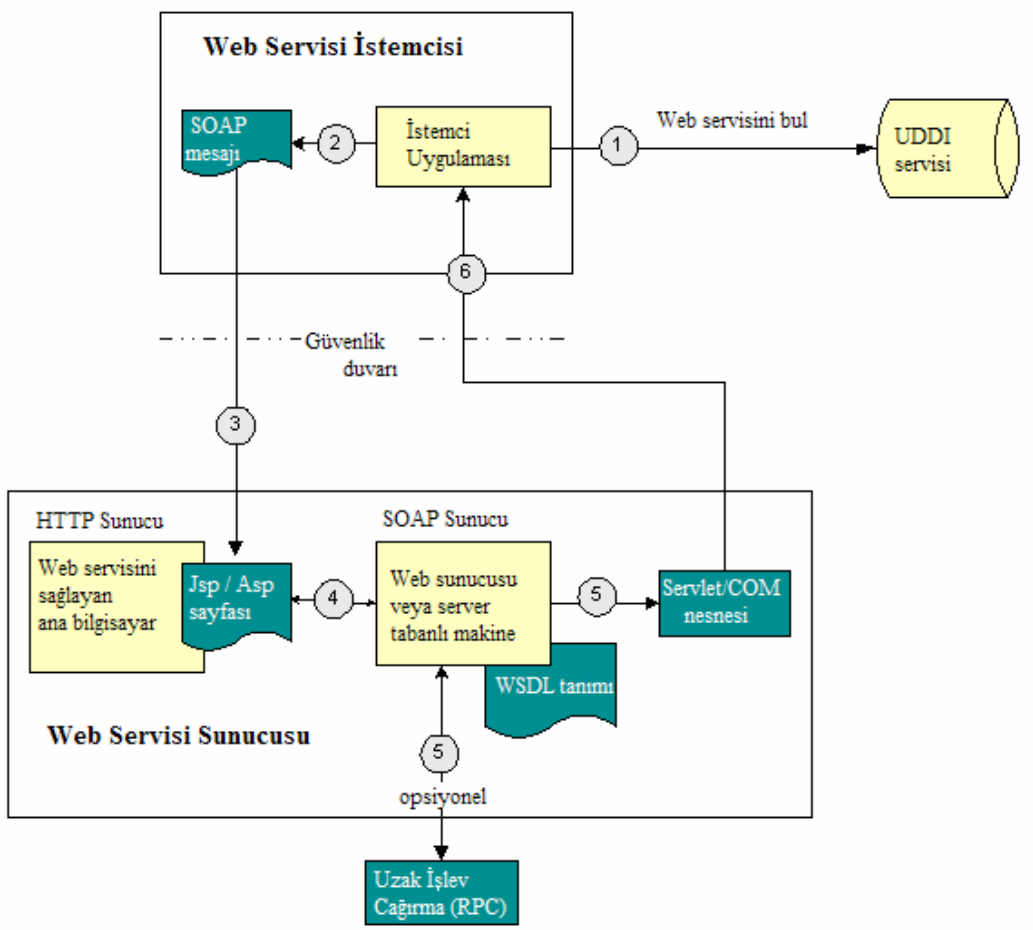


Şekil 2.1. Web Servis Modeli

Şekil 2.2' de gösterilen, bir web servisi istemcisinin, servis sağlayıcıdan herhangi bir servisi çağırma aşamasındaki temel adımları şunlardır:

1. Web servisi istemcisi (SOAP Client) servis kayıt biriminden (UDDI) web servisini bulur.
2. İstemci bir SOAP mesajı hazırlar. SOAP mesajı bir XML belgesidir.
3. İstemci SOAP mesajını web sunucusu veya uygulama sunucusunda çalışan SOAP istek dinleyicisine gönderir. İstek dinleyici gelen isteklere cevap veren sunucu programlarıdır. Bu programlar bir Java SunucuSayfaları (JSP: Java Server Pages), Aktif Sunucu Sayfaları (ASP: Active Server Pages), Ortak Geçit Ara yüzü (CGI: Common Gateway Interface) veya Internet Sunucusu Uygulama Programı Ara yüzü (ISAPI: Internet Server Application Program Interface) programıdır.
4. SOAP sunucusu gelen SOAP mesajını çözümler ve gerekli parametreleri göndererek istenen nesnenin istenen yöntemini çağırır.
5. Çağrılan nesnedeki yöntem çalışır ve sonuçları SOAP sunucusuna gönderir. SOAP sunucusu gelen sonucu SOAP mesajı formatında biçimlendirerek istemciye gönderir.

6. İstemci gelen SOAP mesajının içindeki bilgileri alarak istekte bulunan programa gönderir [9].

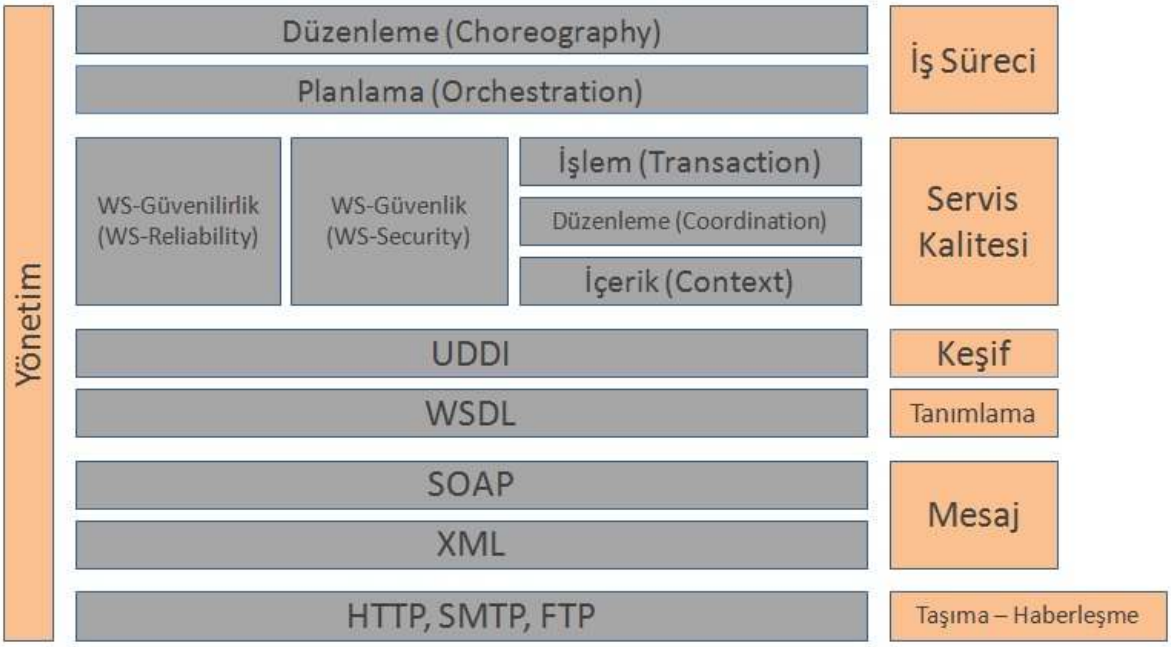


Şekil 2.2. Web servisi istemci ve sağlayıcısı arasındaki temel işlemler [10]

İlk olarak servis sağlayıcı web servislerinin tanımını yapar (WSDL) ve servis kayıt biriminde ya da servis istemcisinde yayınlar. Ardından web servis istemcisi talep ettiği servisi almak için kendi üzerinde yani yerel olarak ya da kayıt birimi üzerinde bu servisi arar, bulur. Son olarak web servis istemcisi elde ettiği bilgileri kullanarak bu servise istekte bulunur. İstemci servis sağlayıcıya istekte bulunurken SOAP istek mesajını kullanır ve geriye dönen mesaj ise SOAP cevap mesajıdır. Veriler ve mesajlar HTTP üzerinden XML olarak aktarılmaktadır.

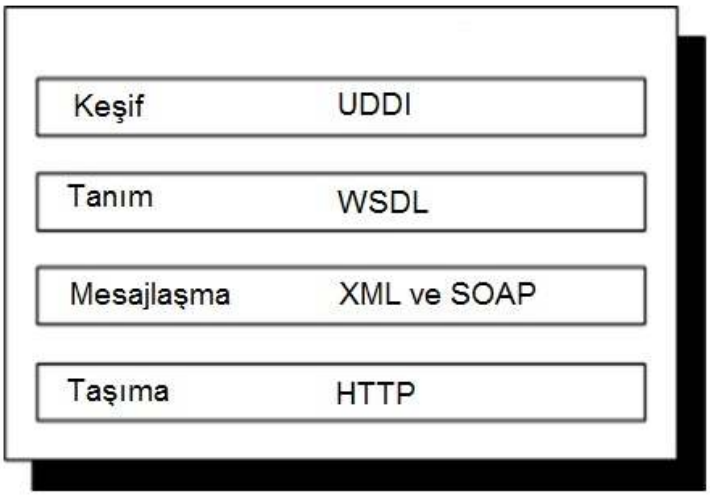
2.3. Web Servis Protokol Yığını ve Web Servis Standartları

Bir web servis yığını mimarisi kurumlara göre çeşitlilik göstermektedir. Yani yığındaki katmanların sayısı ve karmaşıklığı kurumlara bağlıdır. Web servisleri çeşitli teknolojiler kullanarak oluşturulur. Şekil 2.3'te de web servislerinin temelini oluşturan belirli ve tamamlayıcı standartlar yığını gösterilmektedir [7].



Şekil 2.3. Web Servis Teknoloji Yığını

Web servisleri açık internet standartlarına dayanır. Şekil 2.4 web servisi mimarisindeki temel katmanları göstermektedir. Bu katmanlarda belirtilen güvenlik, iş akışı, servis kalitesi ve yönetim gibi konulardaki web servisi standartları henüz araştırma aşamasındadır. Bunların yanında bir takım temel çekirdek standartlar oluşmuştur.



Şekil 2.4. Web Servis Protokol Yığını

Web servislerine bakıldığında çok karmaşık gibi görünse de, web servislerinin çekirdeği olarak tanımlanan 3 standart vardır. Bu standartlar web servis protokolü SOAP, web servis tanımlama dili olan WSDL, web servisleri yayınlamak ve keşfetmek için kullanılan UDDI'dır. Bu çekirdek standartlar XML ailesi özellikleri (XML ve XML Schema) ve IETF HTTP(S) standartları üzerine inşa edilmiştir. XML ve onunla ilgili standartları anlamak web servis çekirdek standartlarını daha iyi anlamak için temel oluşturur.

- SOAP (Simple object access arotocol) : İnternet üzerinde web servislerini çalıştırmak için kullanılan protokol
- WSDL (Web services description language) : Web servislerini tanımlama dili
- UDDI (Universal description, discovery and integration) : Web servislerinin indekslenip bulunduğu kayıt servisi [11]

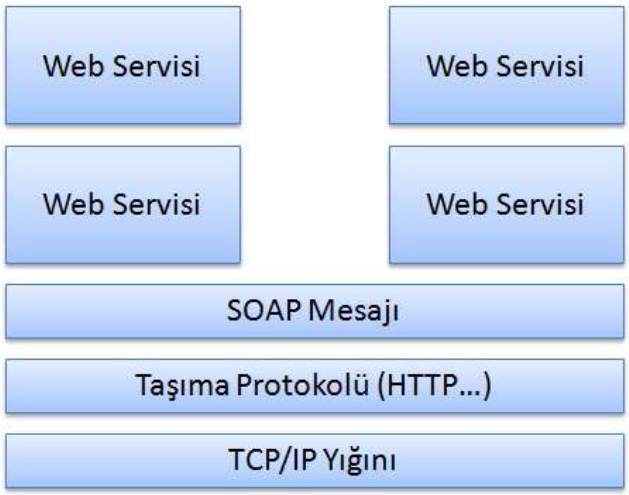
HTTP,SMTP,FTP ve BEEP gibi birçok taşıma katmanı varken, HTTP en çok kullanılan protokoldür. WSDL bileşeni taşıma katmanı olarak HTTP protokolünü kullanır. Bu teknolojiler arasındaki iletişim hakkında basit bir örnek vermek gerekirse, HTTP telefon kablosu (uygulamalar arasındaki taşıma) ve UDDI bir telefon defteri (kayıtlı servisleri bulmak için UDDI kayıtlarına bakma) olarak düşünülebilir. SOAP insanların telefonda konuşmaları (bilgi alışverişi) ve XML ise konuştukları dil olarak düşünülebilir. WSDL ise, belirli bir web servisini çağırmak

için kullanılan telefon numarası olabilir (WSDL tabi ki bir telefon numarasından daha fazlasıdır, veri tipleri ve metotlar gibi bilgileri içerir.) [12].

2.3.1. SOAP (Simple Object Access Protocol)

SOAP, web servisler tarafından kullanılan mesajlaşma protokolüdür. Dağıtılmış platformların birlikte çalışabilmesini sağlamak için tasarlanmıştır. Bu amaç diğer başarılı web protokolleri ile basitlik, esneklik, güvenlik duvarı kullanım kolaylığı, XML tabanlı mesajlaşma ile sağlanan platform bağımsızlığı gibi aynı prensipleri izleyerek gerçekleştirilir. Yeni bir teknolojik yenilik sunmak yerine, SOAP, web üzerindeki dağıtık iletişimleri standartlaştırmak amacıyla, yalnızca var olan internet teknolojilerini kullanımının bir şekilde derlenmesini göstermektedir.

SOAP mevcut internet altyapısında olan router, firewall ve proxy sunucularda herhangi bir değişiklik yapmadan kolayca çalışmaktadır. SOAP genellikle web kaynaklarına ulaşmak için web tarayıcılar tarafından kullanılan HTTP üzerinden değiştirilir. HTTP SOAP mesajlarının gönderilip alınması için etkili bir yol oluşturmaktadır.



Şekil 2.5. Web Servislerin İletişim ve Mesajlaşma Ağı

Şekil 2.5 SOAP mesajlarının taşınması için HTTP den farklı protokollerin de kullanılabileceğini göstermektedir. SOAP'nin rolü mesajın nasıl biçimlendiğiyle ilgilidir, nasıl gönderildiğiyle ilgili değildir. HTTP iletişim protokolleri arasında en çok kullanılan protokoldür. Ancak SMTP ya da FTP protokolleri gibi diğer protokoller de kullanılabilir.

SOAP mesaj yapısı oldukça basittir. Mesajda Header (başlık) ve Body (gövde) olarak adlandırılan iki bölüm bulunmaktadır. Bu iki kısım Envelope (zarf) adı verilen eleman içerisinde bulunmaktadır. Aşağıda basit bir SOAP mesaj yapısı yer almaktadır.

```
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Header>
    ....
    </SOAP-ENV:Header>
    <SOAP-ENV:Body>
    ....
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

İstemci SOAP uygulaması bir SOAP istek mesajı oluşturarak bu isteği SOAP sunucusunda tanımlanmış servis uç noktalarından (end point) birisi tarafından çalıştırılması için gönderir. SOAP sunucu ilgili servisi çalıştırdıktan sonra SOAP yanıt mesajı hazırlar. Hazırlanan SOAP yanıt mesajı istemciye iletilir. SOAP mesajı HTTP POST metodu veri paketinin içinde gönderilir. SOAP gövdesi çağırılacak metot ve metodun içerdiği parametreleri içerir. SOAP gövdesi içinde kodlanarak gönderilen bu mesaj, web servisi tarafından çözülür, gerekli parametreler ve metot çağırım bilgileri eşliğinde işlemlerini gerçekleştirdikten sonra, istemciye döndüreceği cevap bilgileri için, yine SOAP protokolüne uygun XML mesajlarını oluşturur. Bu mesajlar HTTP üzerinden istemci uygulamaya ulaşır, burada çözülür ve değerlendirilir. HTTP protokolünü desteklediğinden, SOAP sayesinde web servisine, platformdan bağımsız çağırımlar yapılabilmektedir. Örneğin, web servislerinin farklı

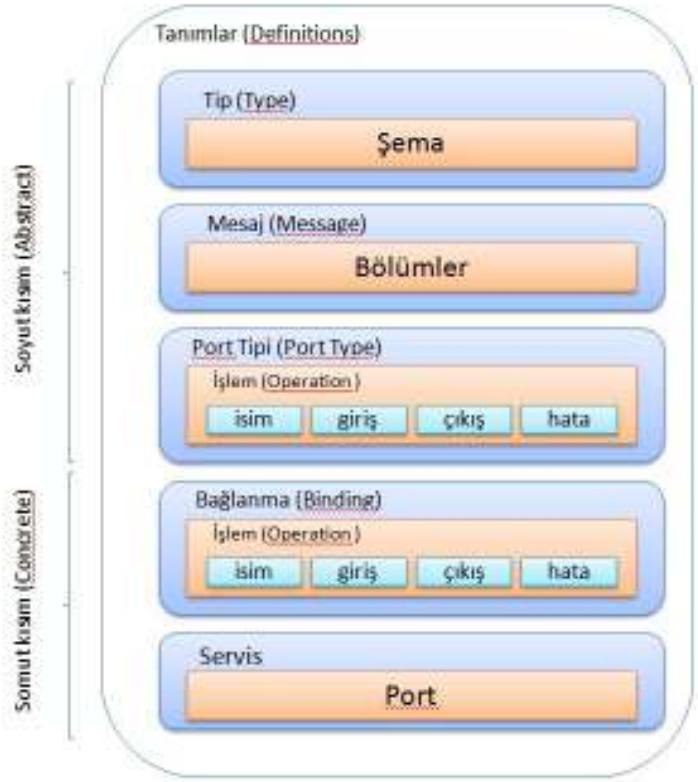
bileşenler üzerinde çalışacağı, .NET (Microsoft) veya J2EE (Sun) teknolojilerinden hangisinin kullanılacağı kararı, tamamen web servislerinin üzerinde geliştirildiği uygulama platformuyla ilgilidir [12 , 13].

2.3.2. WSDL (Web services description language)

Bir uygulamanın bir web servisini kullanması için web servisinin nasıl çağırılacağının, ara yüzünün, hangi protokollerin ve kodlama standartlarının kullanıldığının belirtilmesi gerekir. WSDL web servisini tanımlayan bir XML belgesidir. Web servisi tanımı işlemler, giren ve çıkan mesaj formatları, ağ ve bağlantı noktası adresleri gibi bilgileri tanımlar. WSDL dokümanı bir servis sağlayıcı için uç nokta (endpoint) tanımlar. WSDL servis ara yüzünün genel tanımlarını içermektedir, böylece servis sağlayıcıyı çağırmak isteyen istemciler mesajların nasıl oluşturulduğunu bilirler. İstemci bu dosyayı kendi bilgisayarına indirdikten sonra, o web servisinin metotlarını kullanabilir. Ayrıca WSDL servisin fiziksel konumunu içermektedir [14].

Şekil 2.6'da da gösterildiği gibi bir web servisi tanım belgesi aşağıdaki temel elemanları içerir: [15]

- Veri Tipi (Types): Mesajlarda kullanılacak veri tiplerini belirtir.
- Mesaj (Message): İletişimde kullanılacak mesajları tanımlar.
- Port Tipi (PortType): Web servisinin içerdiği metotları ve ilgili mesajları tanımlar.
- Bağlanma (Binding) : İşlem ve mesajlarda kullanılacak veri formatlarını tanımlar.
- Port: Binding ve web adresinden oluşan servis noktasını tanımlar. Web adresi servisin çalıştırılacağı URL'dir.
- Servis: Kullanılan bağlantı noktalarının kümesidir.



Şekil 2.6. WSDL Dökümanının Yapısı [7]

2.3.3. UDDI (Universal description discovery and integration)

Bir web servisini kullanmak için kullanıcının web servisi sağlayan kurumları ve bu kurumların verdikleri web servislerinin neler olduğunu bilmesi gerekir. UDDI kısaltmasında geçen Evrensel, Tanım, Buluş ve Bütünleştirme kelimelerinin ifade ettiği gibi UDDI, platformdan bağımsız, XML tabanlı kayıtları tutar ve internet üzerinde yayınlar. İlaveten OASIS (Yapılandırılmış Bilgi Standartları Örgütü) tarafından desteklenen bir girişim olan UDDI, kurumların sağladıkları servisleri yayınlamasını, bu bilgilerin daha sonra diğer kurumlarca taranıp bulunmasını, servislerin ve yazılım uygulamalarının internet üzerinde nasıl iletişimde olduğunu tanımlamasını sağlayan bir standarttır.

UDDI Kurum Kayıt Servisi (UDDI Business Registry) kurum ve web servisleri bilgilerini saklayan sunuculardır. Bu sunucular servis sağlayıcılarından gelen

bilgilerini kendi veritabanlarına kayıt ederek diğer kurumların erişimine açar. Şu anda aktif olarak çalışan kurum kayıt sunucuları uddi.microsoft.com ve uddi.ibm.com ‘dur. Bu sunucular kendilerine kayıt edilen bilgileri diğer sunuculara da kopyalayarak kolayca hızlı bir şekilde erişilmesini sağlarlar. UDDI sunucuları kurum ve servis kayıt, güncelleme ve tarama işlemlerini web servisleri (SOAP mesajları) ile gerçekleştirir [16].

2.3.4. XML (Extensible markup language)

XML web üzerinde veri değişimi için standart haline gelen metin temelli bir işaretleme dilidir. HTML’den farklı olarak XML etiketleri verinin nasıl gösterileceğini değil, verinin nasıl tanımlanacağını gösterir. İnternet üzerinde platformdan bağımsız veri alış-verişinde XML kullanılmaktadır. XML‘de veriler etiket ve sonlandırıcı etiket arasında paketlenerek gönderilir. XML’de veriler iç içe etiketlerden oluştuğu için XML’in hiyerarşik bir yapısı vardır. XML, Internet Komitesi (W3C) tarafından ortaya çıkarılmıştır. XML farklı uygulamalarda, verilerin bütünleştirilmesinde araç olarak kullanılır. Yazılım firmaları XML’e büyük destek vermektedir. XML, Firmadan Firmaya (B2B: Business-to-Business) olan uygulamalarda standart bir dil ve belge türüdür.

XML metin halinde olup ikili (binary) formatında olmadığı için standart metin editörlerinden görsel (visual) geliştirme ortamına kadar herhangi bir araçla oluşturulup, geliştirilebilir. Diğer taraftan bir veritabanı büyük miktarda XML verisi depolayabilir. XML, veri tanımlaması sayesinde, ne çeşit verinin (metin, sayı, vb.) olduğunu gösterir. İşaretleme etiketleri bilgiyi belirttiği ve veriyi kısımlara ayırdığı için bir e-posta programı onu işleyebilir, bir arama programı belirli kişilere yollanacak mesajları arayabilir, bir tarayıcı ile de XML çözümlenebilir. Kısaca, bilginin farklı bölümleri tanımlandığından bunlar farklı uygulamalarda farklı şekillerde kullanılabilir. XML uygulama, dil, işletim sistemi gibi kısıtlamalara bağlı değildir. Şu anda hemen her işletim sistemi ve uygulama XML belgelerini okumak ve yazmak için yerleşik özelliklere sahiptir. Bütün XML belgeleri unicode tabanlıdır. İşletim sistemleri farklı kod sayfalarını (code page) kullandıkları için farklı dillerdeki

belgeleri görüntüleyemezler. Ancak, unicode kullanımı farklı dillerdeki belgelerin görüntülenmesine olanak tanır [5].

2.4. Web Servislerin Özellikleri

- Web servisleri bağımsız uygulamalardır.

İstemci için, XML ve HTTP destekleyen bir programlama dili yeterlidir. Sunucu kısmında yalnızca bir web sunucu ya da SOAP sunucu gerekir. Web servisler var olan uygulamaları tek satır kod yazmadan kullanabilmeyi sağlar.

- Web servisleri dilden bağımsız ve birlikte çalışabilir özelliktedir.

İstemci ve sunucu farklı platformlarda tanımlanmış olabilirler. Bu durumda web servislerine erişebilmek için kod kısmında bir değişiklik yapmaya gerek yoktur. Herhangi bir web servisi diğer web servisleriyle etkileşimde bulunabilir. Bu, XML-tabanlı bir ara yüz tanımlama dili ve iletişim protokolü ile elde edilir. Birlikte çalışabilirlik için gereksinimleri sınırlayarak, etkileşimde bulunan web servisleri gerçekten platform ve dilden bağımsız olabilirler. Bu, geliştiricilerin web servisi üretmesi ya da tüketmesi için geliştirme ortamlarını değiştirmeleri gerekmediği anlamına gelir. Ayrıca, eskiden kalma uygulamaların web servisleri olarak sunulmasına olanak veren bir web servisi mimarisi, web servisleri ile eskiden kalma uygulamalar arasında birlikte çalışabilirliği kolaylıkla sağlamış olur.

- Web servisleri doğası gereği açık ve standartlara dayalıdır, kolay ve hızlı dağıtılır.

XML ve HTTP web servisleri için temel oluşturur. Web servis teknolojisinin büyük bir kısmı açık kaynak kodludur. Web servisleri modelini kullanan şirketler, yeni servis ve ürünlerini gecikme ve fazla harcama olmadan sunarlar. Ayrıca, geliştirmek yerine kendilerine en uygun bileşen servislerini kullanabilirler.

- Web servisleri düşük bağımlıdır (loose coupling).

Genel olarak, uygulama tasarımlarında, geliştirme ve ürün arasında sıkı bir bağ vardır. Web servisleri ise, söz konusu servislerin entegrasyonu için daha esnek bir yeniden yapılandırma gerektirir.

- Web servisleri kendi kendini tanımlar (self-describing).

WSDL, bir web servisinin istemci ve sunucu tarafında gereken bütün bilgileri bulundurur. Bir servis sağlayıcı, UDDI kayıtlarını ve WSDL dokümanlarını kullanarak geliştiricinin servisi oluşturmak ve dağıtmak için anlaması gereken bütün bilgileri yayınlar.

- Çalışma zamanında entegrasyon sağlarlar.

Geleneksel sistem mimarilerinde bileşenler statik olarak entegre oldukları için sıkı sıkıya bağlıdırlar ve küçük bir değişiklik işbirliğini bozar. Web servisleri mimarilerinde ise yeni servisler ve uygulamalar o anda elde edilebilir olan servislerin keşfinden sonra dinamik olarak çalışma zamanında entegre olurlar.

- Etkin uygulama geliştirme sağlarlar.

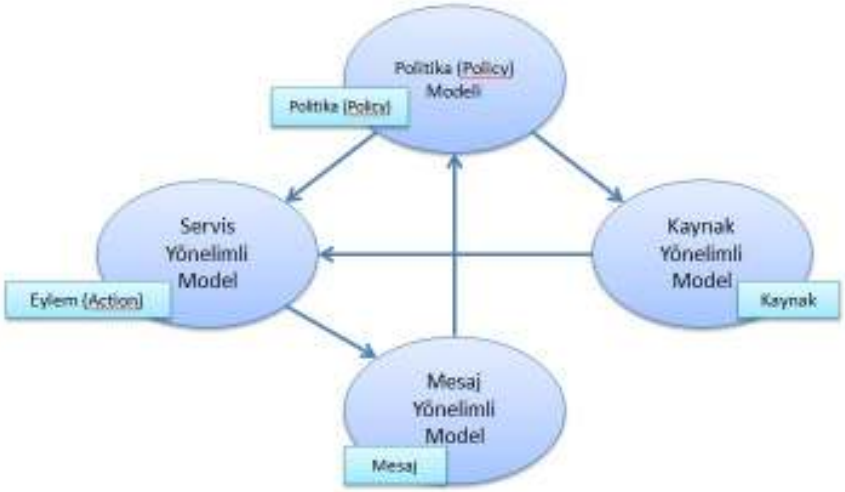
Uygulama geliştirme daha etkindir. Çünkü varolan web servisleri tekrar kullanılabilirler ve yeni web servisleri oluşturulabilir. Bir web servisi, üst-seviye özellikler sunmak için başka web servislerini birleştirebilir.

- Kapsülleme (Encapsulation) özellikleri vardır.

Bütün bileşenler servislerdir. Önemli olan bir servisin sunduğu davranış tipidir, nasıl gerçekleştirildiği değildir. Bu sistemin karmaşıklığını azaltır. Çünkü uygulama tasarımcıları, çağırdıkları servislerin gerçekleştirim detaylarıyla ilgilenmezler [5,17].

2.5. Web Servis Mimarileri

Web servis mimarilerinin şekil 2.7'de de gösterildiği gibi 4 modeli bulunmaktadır [18].



Şekil 2.7. Web Servis Mimarileri Modelleri

2.5.1. SOA (Servis yönelimli mimari)

Servis yönelimli mimari (SOA), birçok uygulamanın kendi modüllerini veya fonksiyonlarını (bunlara servis deniyor) başka uygulamalar tarafından kullanılacak şekilde tasarladığı bir mimari kavramdır. SOA, bağımsız servisler olarak iş mantığının sağlanmasına olanak tanır [19]. İş modellerinin çok hızlı bir şekilde değişmesinden dolayı birçok kuruluş servis yönelimli mimari ile gelişmektedir. SOA'da servislerin keşfedilebilir olması, kendi tanımlanabilmesi, birlikte çalışabilirliği, düşük bağımlılığı ve birleştirilebilir özelliklerde olması, işyerlerinde bir tasarım yapılırken bu teknolojinin daha kolay kullanılabilmesine olanak tanır ve radikal değişimler için bir potansiyel oluşturur [20].

Pratikte, SOA uygulamalarının sayısındaki artışla birlikte, geleneksel bilgi sistemi güvenlik durumlarını beraberinde getirmiştir. Son zamanlarda, SOA'nın web servis teknolojisini kullanarak geliştiriliyor olması, web servis teknolojisinin birçok avantaj ve dezavatajlarını devralıyor olması demektir. Uygulama düzeyindeki SOA güvenliği

çoğunlukla web servis güvenlik çözümleri tabanlıdır (SOAP, WS-Security, WS-SecureConversation, WS-Policy, WS-Trust vb.) [21].

SOA genellikle farklı güvenlik yönetim sistemleri altında ve farklı koşullarda çalışan, farklı organizasyon ve farklı platformlarda geliştirilen ve üretilen birden fazla web servisinden oluşmaktadır. Bütün bu bileşenler, kullanıcılarına klasik bilgi sistemindeki güvenlikle aynı düzeyde bir güvelik sağlamak zorunda olan bir bilgi sistemine bağlanır [22].

Service yönelimli mimari, dağıtık uygulamalar için ideal bir mimaridir. Bugün pek çok web servis uygulaması aşırı bağımlı alt sistemlerden oluşmaktadır. Bu durum, alt sistemlerden birinde yapılacak bir değişikliğin uygulamanın tümünü olumsuz etkilemesine neden olmaktadır. Bu kırılgan yapı, bakım maliyetlerinin yüksek olmasındaki birincil sebeptir. Ayrıca bu durum, değişen iş gereksinimlerinin karşılanmasını ve ileride yapılması muhtemel modifikasyonları zorlaştırmaktadır [19].

2.5.2. Mesaj yönelimli model (Message oriented model)

Mesaj yönelimli model, mesajların içeriğiyle ya da nedenleriyle ilgilenmeden yalnızca mesajlarla, mesaj yapısıyla ve mesajların iletimiyle ilgilenir. Özellikle, bu modelde, bir mesajın içeriğinin anlamsal önemiyle ya da diğer mesajlarla ilişkisiyle ilgilenilmez. Ancak, mesaj yönelimli model mesajın yapısıyla, mesajın göndereni ve alıcısı arasındaki ilişkiye ve mesajın nasıl iletildiği üzerine yoğunlaşılır [18].

2.5.3. Kaynak yönelimli model (Resource oriented model)

Bu model mevcut kaynaklarla ve bu kaynakların sahibiyle ilgilenir. Kaynaklar web servislerinin ve web in altında yatan temel kavramlardır. Örneğin, bir web servisi bu model için önemli bir kaynağın belirli bir türüdür [18].

2.5.4. Politika modeli (The policy model)

Politika modeli servislerin ve ajanların davranış kısıtlamaları üzerine yoğunlaşır. Politikalar aktif kaynakların yanı sıra dokümanlara (servis tanımlamaları gibi) da uygulanabilir [18].

BÖLÜM 3. WEB SERVİS GÜVENLİĞİ

Web servislerinde güvenlik son günlerde en çok üzerinde durulması gereken konu haline gelmiştir. Veriye erişimdeki kolaylık, uygulamadan uygulamaya dinamik bağlantılar, insan müdahalesi gerektirmeden kendi kendine çalışmak gibi özellikleri içeren ve böylece Web servislerini cazip hale getiren birçok özellik geleneksel güvenlik modelleriyle ve kontrolleriyle uyuşmamaktadır. Web servisleri uygulamaların API'lerine ve hedef uygulamalara erişim sağladığından birçok güvenlik açığı bulunmaktadır. Web servislerinin dağıtık ve uçtan uca yapısı tehdit ve güvenlik açıklarının bir uygulamadan başka uygulamalara atlamalarına neden olabilmektedirler. Ayrıca internet ortamında çok fazla sayıda kullanıcı ve çok daha fazla sayıda belge ve servislere yapılacak ataklar düşünüldüğünde aktarılan verilerin güvenilirliği ve bütünlüğünün sağlanmış olması büyük önem taşımaktadır.

SOA'nın web servis teknolojisini kullanarak geliştiriliyor olması, web servis teknolojisinin birçok avantaj ve dezavantajını devralıyor olması demektir. Uygulama düzeyindeki SOA güvenliği çoğunlukla web servis güvenlik çözümleri tabanlıdır, ağ güvenliği için kullanılan çözümler SOA'nın güvenliğini sağlamak için yetersiz kalmaktadır [19].

Web servis modeli, SOAP mesajları ve XML dokümanlarının, istemci, sağlayıcı ve ara servisler arasındaki iletiminin güvenilir olmasına ihtiyaç duyar. ISO güvenlik standartları tarafından tanımlanan ve genel güvenlik çerçevesi olarak ele alınması gereken 7 tane güvenlik gereksinimi vardır.

- Tanımlama (Identification) : Sistemde kişilerin kaynağa erişmek için kimliklerini belirtmesidir.

- Kimlik Belirleme (Authentication) : Kimlik belirleme kullanıcının doğrulanması işlemidir. Bir istemci, bir son kullanıcı, bir makine ya da bir uygulama olabilir.
- Yetkilendirme (Authorization) : Kimliği belirlenen kullanıcının ulaşılmak istenilen kaynaklara izinli olup olmadığının kontrol edilmesidir.
- Bütünlük (Integrity) : Bilginin yetkisizce ya da yanlışlıkla değiştirilmediğinden, kaybolmadığından ya da bozulmadığından emin olmaktır.
- Gizlilik (Confidentially) : Bilgilerin yalnız yetkili tarafından kullanılmasını, diğer bir değişle yetkisiz kullanıcıların erişiminin engellenmesi demektir. Bilginin isteyerek ya da istemeyerek ortaya çıkarılmasını engeller.
- İzleme (Auditing) : Bütün işlemler kaydedilir, böylece oluşacak problemler sonradan analiz edilebilir.
- İnkâr edememe (Non-repudiation) **:** Bu prensip verinin iletildiği gönderici ve alıcı arasında ortaya çıkabilecek iletişim sorunları ve anlaşmazlıkları en aza indirmeyi amaçlar. İki sistem arasında bir bilgi aktarımı yapılmışsa ne gönderen veriyi gönderdiğini nede alıcı veriyi aldığını inkâr edememelidir.

Web servis güvenliğinin anahtar konularından olan kimlik belirleme, yetkilendirme, güvenilirlik ve bütünlük konuları derinlemesine ele alınmaya ve bunlarla alakalı birçok fikir ortaya konmaya başlamıştır. Son zamanlarda IBM, OASIS, Microsoft gibi standart gruplar web servis güvenliği konusunda görüşlerini beyan etmişlerdir [23].

3.1. XML ve Web Servisleri

Heterojen sistemleri bir araya getirmek için kullanılan web servislerinin uyumluluğu, XML in geniş kullanım alanı sayesinde çok kolaylaşmıştır. Web servislerinin XML tabanlı karakteristiğe sahip olması, arka plan sistemleri ne kadar farklı olursa olsun onları servis sağlayıcılar ve servis istemciler arasında kullanılabilir kılmaktadır. Bu özellik web servislerini XML tabanlı teknoloji yapmaktadır. Yani diğer bir değişle web servis teknolojisinin SOAP, WSDL, BPEL gibi temel bileşenlerinin tamamı

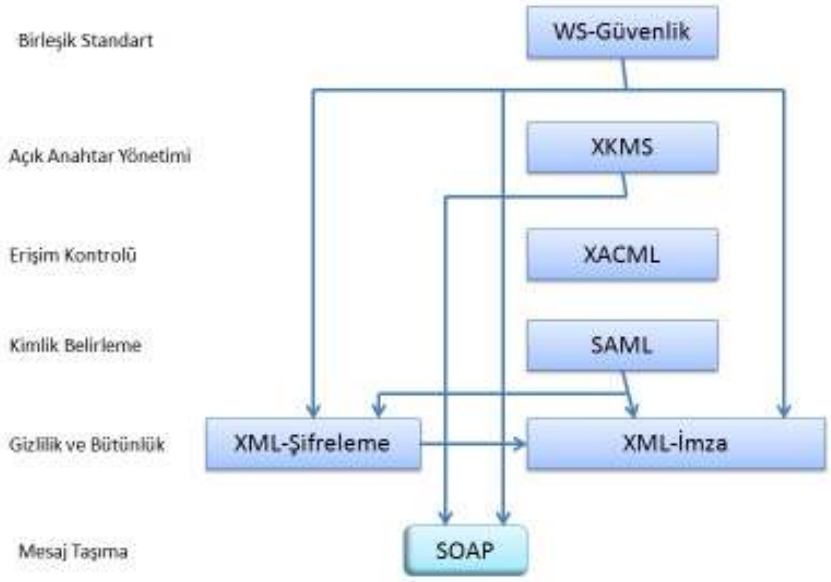
XML tabanlıdır. Dolayısıyla web servislerinde güvenlik denildiğinde XML güvenliğiyle ilgilenilmektedir. XML-İmza (XML-Signature) ve XML-Şifreleme (XML-Encryption), W3C (World Wide Web Consortium) tarafından 2002 yılında ortaya konmuş XML güvenlik standartlarının ana parçalarıdır. Birçok dil de aynı amaç için geliştirilmiştir. Bunlardan biri olan WS-Security, SOAP'un genişletilmiş hali olup, mesaj bütünlüğü ve güvenilirliği için kullanılmaktadır. WS- Security, XML digital signature ve XML-Encryption elemanlarının SOAP içerisinde kullanılabilmesini sağlar [24,25].

XML'in genişletilmiş hali olan diğer diller;

- XML- İmza (Signature) : Düz XML dökümanlarına kriptografik koruma kazandırır. Bu yapı ile bütünlük, kimlik doğrulama ve inkar edilememe sağlanmış olur [26].
- XML- Şifreleme (Encryption) : XML dökümanlarının şifrelenmesiyle ve aynı güvenilirlikte gönderilmesi hakkında destek sağlar [27].
- XKMS: XML-İmza ve XML-Şifreleme ile birlikte kullanılan genel anahtarların (public keys) yönetimi için kullanılır [28].
- XACML: (Extensible Access Control Markup Language) Bir takım kurallar belirterek, karmaşık yetkilendirme kararlarının alınmasını sağlayan standart olarak tanımlanmaktadır [29].

3.2. XML Güvenliği

XML çok iyi bilinen ve veri yapılandırılması için yaygın olarak kullanılan bir teknolojidir ve bu nedenle, XML tabanlı güvenlik, bilgi sistemleri arasındaki güvenlik için karmaşık ihtiyaçları ele alırken XML in birçok avantajını kullanır. W3C ve OASIS tarafından geliştirilen XML güvenlik standartlarının amacı XML dokümanları korumak için bazı tanımlamalar yapmaktır. Bunun için XML kelimeler ve kurallar tanımlanmıştır. Şekil 3.1 XML ve Web servis güvenliği için kulanılan en önemli tanımlamaları göstermektedir.



Şekil 3.1. XML Güvenlik Standartları

Bu XML tabanlı uygulamalar için standart bir çalışma alanı sağlar. SOAP mesajların iletiminde kullanılan protokoldür. XML- imza (XML Signature), XML- şifreleme (XML Encryption), mesajın gizliliğini ve bütünlüğünü sağlar. XACML yetkilendirme sırasında erişim kontrollerini sağlarken, SAML kimlik belirleme de onaylama için kullanılır. XKMS ise şifreleme ve imzalama sırasında anahtar yönetimini sağlamaktadır. Bu kavramlar ilerleyen bölümlerde ayrıntılı olarak anlatılacaktır [30].

3.2.1. XML – Şifreleme (Encryption)

XML Şifreleme, güvenli veri alışverişi gerektiren uygulamalarda uçtan uca güvenliği sağlar. XML in kendisi veri yapılandırması için kullanılan en popüler teknolojidir. Bu nedenle XML tabanlı şifreleme veri değişimi yapan uygulamalarda ihtiyaç duyulan güvenlik için kullanılması en tabi yoldur.

XML şifreleme bir XML dokümanının seçilen bölümünü şifreleyerek gizlilik sağlar. bilginin iletimi sırasında ve hatta depolanması sırasında dahi gizliliğini sağlama amacına hizmet eder. Gizliliği sağlamak için kullanılan SSL (güvenli soket katmanı),

TLS (taşıma katmanı güvenliği) ya da VPN (sanal özel ağ) gibi diğer teknolojiler bilgi depolanırken değil yalnızca bilgi taşınırken gizlilik sağlarlar.

XML şifreleme belirli söz dizimleri ve algoritmalar kullanarak XML içeriklerini şifreleme ve şifreyi çözme (decryption) işlemidir. Şifrelemenin (cryptography) temel mantığı hemen hemen aynı kalır ancak tek farkı şifrelenmiş XML verinin değişimi ve gösterimi için standart bir formatı olmasıdır. Bu standart format XML in şifrelenmiş içeriğinin gösterimi için standart bir söz dizimi, aynı zamanda alıcı tarafta şifrenin çözülmesi için gereken bilgiyi içerir. Tablo 3.1'de orijinal XML verisi, Tablo 3.2'de ise şifrelenmiş XML verisi örnekleri gösterilmektedir.

Tablo 3.1. Orijinal XML verisi

```
<?xml version='1.0'?>
 <PaymentInfo xmlns='http://example.org/paymentv2'>
  <Name>John Smith</Name>
  <CreditCard Limit='5,000' Currency='USD'>
   <Number>4019 2445 0277 5567</Number>
   <Issuer>Example Bank</Issuer>
   <Expiration>04/02</Expiration>
  </CreditCard>
 </PaymentInfo>
```

Tablo 3.2. Şifrelenmiş XML verisi

```
<?xml version='1.0'?>
 <PaymentInfo xmlns='http://example.org/paymentv2'>
  <Name>John Smith</Name>
  <EncryptedData Type='http://www.w3.org/2001/04/xmlenc#Element'
   xmlns='http://www.w3.org/2001/04/xmlenc#'>
   <CipherData>
    <CipherValue>A23B45C56</CipherValue>
   </CipherData>
  </EncryptedData>
 </PaymentInfo>
```

W3C XML şifreleme çalışma grubu – W3C (World Wide Web Consortium) ve IETF (Internet Engineering Task Force) arasında yapılan etkili bir işbirliği- XML şifreleme standartlarını ve tanımlamalarını içerir (W3C ve IETF, XML/Web servis

teknolojileri için birlikte çalışabilen standartları tanımlarken başı çeken iki kuruluştur.) [27,31].

3.2.2. XML – İmza (Signature)

XML İmzaları, XML tabanlı işlemlerde kullanılmak üzere tasarlanmış dijital imzalardır. W3C ile IETF ortak çabasıyla tanımlanmışlardır. XML İmzaları, imzalama için kullanılan algoritma ve imzalamada kullanılan anahtar hakkındaki bilgiler de dâhil olmak üzere, XML belgesini güvenli bir biçimde imzalamak için gereken çeşitli bileşenleri tanımlayan bir dizi XML öğesi ve özniteliğini, imzalanacak XML belgesinin konumu hakkında bilgileri içerir.

XML İmzalarının önemli bir özelliği, tek bir XML belgesinin içinde birden çok dijital imzanın bulunabilmesidir. Bu birden çok kullanıcının aynı XML belgesinde çalışmasına ve her kullanıcının diğer kullanıcılar tarafından eklenen imzaların üzerinde yeni bir dijital imza sağlamasına olanak verir.

XML belgesine dijital imza eklemek, belgedeki verilerin daha güvenli olmasını, yetkili kişi ve yazılımlar dışındakilerin okuyamamasını, değiştirememesini ve XML belgesini imzalayan kişinin kimliğinin doğrulanabilmesini sağlar. Tablo 3.3, XML imzanın yapısını ve anahtar kelimelerini göstermektedir.

Tablo 3.3. XML imza yapısı

```
<Signature ID>
<SignedInfo>
<CanonicalizationMethod/>
<SignatureMethod/>
<SignatureValue>
(<Reference URI>
<DigestMethod>
<DigestValue>
</Reference>)
</SignedInfo>
(<KeyInfo>)
</Signature>
```

İmzanın anahtar öğeleri şunlardır:

- SignedInfo — İmzalı verilere başvuruları ve imzalama sırasında hesaplanan özet değerleri içerir. İmzalı veriler, XML imzasıyla aynı belgede yer alabilir veya harici olabilir.
- SignatureValue — İmza sahibinin özel anahtarıyla şifrelenmiş SignedInfo öğesinin özetini içerir.
- KeyInfo — İmzalayıcı sertifikasının yanı sıra, güven zincirinin oluşturulması için gerekli ek sertifikaları içerir.

Üç genel XML imza türü mevcuttur:

- Zarflı — imza, imzalamakta olduğu XML verilerinin içine eklenir.
- Zarflanan — imzalı XML verileri, Signature öğesinin içinde yer alan bir nesne (object) öğesinde bulunur.
- Ayrılan — imzalanan veriler, XML imzasının dışında yer alır. İmzalanan veriler, harici bir dosyada bulunabilir. Alternatif olarak imzayla aynı XML belgesinde olabilir, ancak Signature öğesinin üst veya alt öğesi olamaz [26].

3.2.3. XML – Anahtar yönetim sistemi (XKMS)

Açık anahtar teknolojisi (public key technology-PKI) ; XML imza, XML şifreleme ve diğer güvenlik uygulamalarının zorunlu bir bölümünü oluşturur. XML anahtar yönetim tanımlaması (XKMS), açık anahtar altyapısının (PKI) gerçekleştirilmesi için XKMS istemcisi ve sunucusu arasındaki protokolleri tanımlar. Açık anahtar kaydı, doğrulaması, keşfi ve iptalini içeren açık anahtar yönetimi için istek ve cevaplarına XML mesaj formatları tanımlar. Bundan dolayı, XKMS XML imza ve XML şifreleme ile birlikte kullanılmak üzere tasarlanmıştır ve imza doğrulaması ve şifrelemeyi sağlayan açık anahtar yönetimine yardım eder. PKI web servislerinde ve e-ticaret sitelerinde önemli bir rol oynamaktadır. PKI işlemlerinin küçük cihazlar için çok masraflı olurken, XKMS kullanımı işlem yükünü azaltmıştır. Diğer bir yandan, PKI işlemleri birçok uygulama için çok karmaşıktır, ancak XKMS kullanımı, PKI işlemlerinin karmaşıklığını bir XKMS sunucusuna taşıyarak azaltabilir.

Aşağıda gösterilen örnekte XKMS bir doküman imzasıyla çalışmaktadır. Bir istemci imzalanmış bir XML doküman alır, <KeyName> ve <KeyValue> değerlerini talep etmek üzere <ds:Keyinfo> bilgisini servise gönderir. <ds:Keyinfo> öğesi içinde genel anahtar içeren bir X.509 sertifikasını bulunduran <ds:RetrievalMethod> bulunur. Servis sertifikayı elde etmek için <ds:RetrievalMethod> öğesini çözümler. İstemciye genel anahtar değerini geri göndermek için sertifika çözümlenir ve sertifikadan elde edilen <KeyName> geri döndürülür [28,30].

Tablo 3.4. XKMS doğrulaması için istek ve cevap [30]

Request:

```
<Locate>
   <Query>
      <ds:KeyInfo>
         <ds:RetrievalMethod
         URI= “http://www.PKeyDir.test/
         Certificates/01293122”
         Type= “http://www.w3.org/2000/09/
         xmldsig#X509Data”/>
     </ds:KeyInfo>
  </Query>
  <Respond>
  <string>KeyName</string>
  <string>KeyValue</string>
  </Respond>
</Locate>
```

Response:

```
<LocateResult>
     <Result>Success</Result>
       <Answer>
          <ds:KeyInfo>
             <ds:KeyName>
               O=XMLTrustCernter.orgOU= “Crypto”
               CN= “Alice”
            </ds:KeyName>
           <ds:KeyValue>...</ds:KeyValue>
         </ds:KeyInfo>
     </Answer>
</LocateResult>
```

3.2.4. Genişletilebilir erişim kontrol işaretleme dili (XACML)

XACML (Extensible access control mark-up language), erişim denetim politikalarını göstermek için kullanılan XML tabanlı bir dildir. Erişim kontrolü için kuralları, istek ve cevapları tanımlar. Dağıtık sistem ortamlarında erişim kontrol kararı vermek için, farklı yetki alanındaki kuralların bir kural kümesi içinde birleşmesine izin verir ve böylece erişim kurallarının gösteriminde esneklik ve mekanizma bağımsızlığı sağlar.

Bir XACML politikası hedef, etki ve koşul içeren kurallar kümesinden oluşur. Hedef, kuralın uygulanacağı kaynaklar, özneler ve eylemler kümesini tanımlamaktadır. Kuralın etkisi izin ya da yok saymak olarak olacaktır. Koşul, kuralın uygulanabilirliğini belirten bir boole tanımı gösterir. Bir istek, istek ile ilgili öznenin, istekte yer alan kaynağın, yerine getirilen eylemin ve çevrenin ilişkili olduğu öznitelikleri içerir. Yanıt ise dört karardan birini içerir: izin (permit), yok saymak (deny), uygulanamaz (not applicable), belirsiz (indeterminate). Uygulanamaz kararı, uygulanabilecek politikaların ya da kuralların bulunamadığı durumu; belirsiz kararı ise erişim denetim işlemi sırasında bazı hataların meydana geldiğini belirtir. Bir istek, bir politika ve ilgili yanıt XACML bağlamını (XACML context) oluşturur. Kuralların uygulandığı özneler ve kaynaklar önceden tanımlanmış işlevler (örneğin; eşitlik, küme karşılaştırma, aritmetik) ve veri türleri (örneğin; tamsayı, boole, karakter dizisi) ile tanımlanmaktadırlar [32]. Tablo 3.5'te gösterilen örneğe göre; hastanın medikal kayıtlarının yalnızca kendi doktorları tarafından okunabilmesini gösteren bir örnek verilmiştir.

Tablo 3.5. XACML kullanılan bir örnek [30]

```
<content>
    <entry>
        <name> Alice</name>
        <record> mental problem </record>
   </entry>
</content>
<policy>
   <xacl>
       <object href= "/contents"/>
       <rule>
```

```
        <subject>
            <uid primary care doctor />
        </subject>
        <action >
            name= “read” permission= “grant”
        </action>
    </rule>
</xacl>
</police>
```

3.2.5. Güvenlik onayı işaretleme dili (SAML)

SAML bir güvenlik protokolü olup asıl amacı birbirinden farklı yetkilendirme ve kimlik doğrulama bileşenleri arasındaki bilgilerin kesin ve güvenli bir şekilde taşınmasını sağlamaktır. Örneğin kurumsal yapılardaki şirketlerde kullanıcılar pek çok farklı sisteme giriş yapmak zorunda kalabilirler. Bundan dolayı da aslında hatırlamaları gereken pek çok kullanıcı adı ve şifre bulunmaktadır. Oysaki merkezi bir kimlik doğrulama ve yetkilendirme yazılımları SAML protokolü sayesinde diğer sistemler ile birlikte çalışarak kendi üzerinden vereceği bir kullanıcı adı şifre ile diğer sistemlere erişim sağlayabilir. Yani SAML, güvenlik bilgilerinin tek bir defa girilerek farklı platform ve sistemlerde bu güvenlik bilgilerini paylaşmak için kullanılır. SAML, bilgilerin taşınırken gizliliği, bütünlüğü ve inkâr edilememesiyle ilgilenmez.

SAML protokolü, açık kodlu bir protokol olup XML tabanlıdır. Bu protokolü kullanan sistemlere Kerberos, Shibboleth, SimpleSAMLphp, JBOSS SSO, Google Apps SSO, Athens Service, OpenID örnek verilebilir. SAML, HTTP,SMTP, FTP gibi birçok protokolle çalışır ve SOAP, BizTalk ve ebXML’ i destekler. Tablo 3.6’ da tek bir defa tanımlayacağı kimlik bilgileriyle alakalı bir örnek gösterilmiştir [33].

Tablo 3.6. SAML Örneği [30]

```
<Assertion>
        <AuthenticationStatement
        AuthenticationMethod= “password”
        AuthenticationInstant= “2003-12-05T10:00:00Z”>
        <subject>
```

```
            <NameIdentifier
            SecurityDomain= “Johns.com”
            Name= “Smith”/>
            <ConfirmationMethod>
            http://...core-25/sender-vouches
            </ConfirmationMethod>
        </subject>
        </AuthenticationStatement>
</Assertion>
```

3.3. Web Servis Güvenliği

Web servis güvenliği (Web service security-WSS) tanımlamaları SOAP kullanarak güvenli web servisleri oluşturmak için bir çalışma alanı sağlamaktadır. Birçok ilave profil ve çekirdek bir tanımlama içerir. Çekirdek tanımlama olarak bahsedilen web servis güvenliğidir. SOAP mesaj güvenlik tanımlaması, SOAP mesaj içerisinde kullanılmak için bir güvenlik başlığı ve bu başlığın gizlilik ve bütünlüğü sağlamak için nasıl kullanıldığını tanımlar. Mesajın bütünlüğü XML imza ile sağlanırken, gizlilik XML şifreleme ile sağlanır. Bu mekanizmalar kullanılarak, SOAP mesajının gövdesi, seçilen başlıkları ya da bunların birleşimi imzalanabilir ve ya şifrelenebilir. Genellikle farklı alıcılar ve aracılardaki farklı SOAP rolleri için farklı imzalar ve şifreler kullanılır.

Web servis güvenliği (WS-Security) dediğimizde aklımıza ilk gelen koruma kalitesi, mesaj güvenirliliği, gizliliği ve tek mesaj doğrulamasıdır. Bu mekanizmalar geniş çeşitlilik gösteren güvenlik modellerini ve şifreleme teknolojilerini kullanırlar.

WS- güvenlik; esnek ve PKI, Kerberos ve SSL ‘i içeren güvenlik modelleri yapısını temel alarak tasarlanmıştır. HTTP üzerinden doğrulama, mesajı imzalama ve mesajın içeriğine şifreleme yapılabilir. Bu tip durumlarda mesaj güvence altındadır. Şöyle ki mesajı gönderen bilinir, mesajın alıcısı doğruluğu kanıtlar, mesaj taşınma esnasında değişmez. Diğer kişiler ele geçirse bile doğrulama yapamazlar, veri değişik bir durumda olduğundan işlerine yaramaz durumdadır. Bu açıdan baktığımızda SOAP mesajlaşma sistemi, HTTP-temelli güvenlik yetersiz olmasına rağmen büyük problemleri çözmektedir. HTTP mekanizmaları sadece noktadan noktaya (point-to-

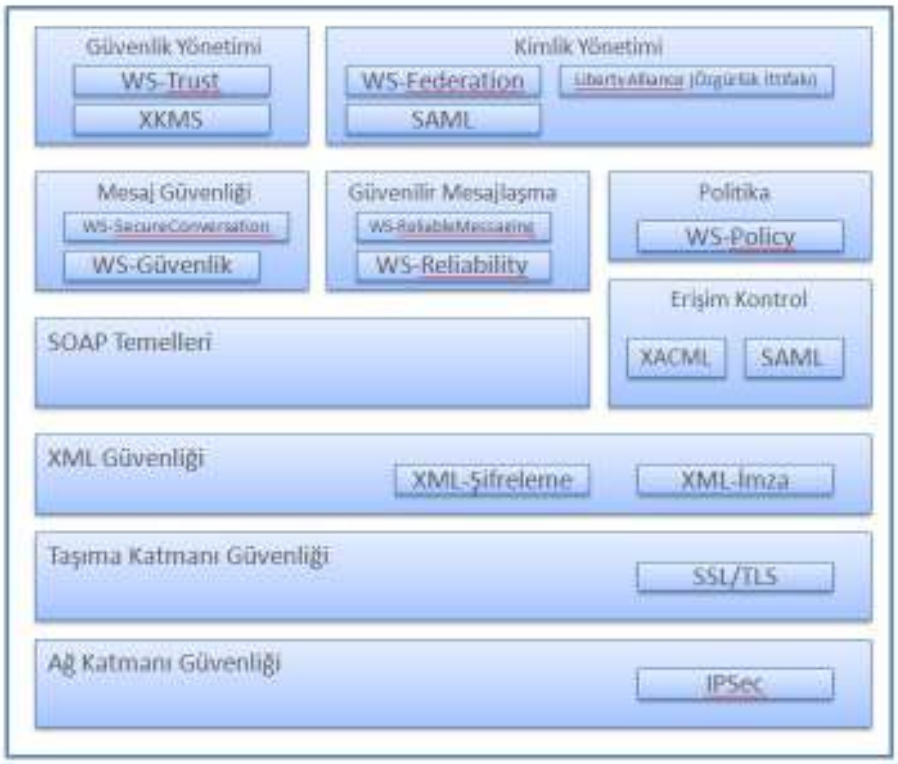
point) güvenliğine hitap eder. Daha karmaşık projeler (çözümler) uçtan uca (end-to-end) güvenliğe ihtiyaç duyar.

WS-güvenlik güvenlikle ilgili olan veriyi taşımak için bir SOAP başlık tanımlar. Eğer XML İmzalama kullanıldıysa, bu başlık mesajın nasıl imzalandığını, anahtarın kullanıldığını, imza sonuç değerini içerir. Ayrıca eğer mesajdaki bir element şifrelendiyse, şifreleme bilgisi WS-güvenlik başlığı içinde yer alabilir. WS-güvenlik tam olarak imza ve şifreleme formatını belirtmez. Onun yerine bir SOAP mesajına diğer otoriteler tarafından nasıl bir güvenlik bilgisinin gömüldüğünü tanımlar.

WS-güvenlik tanımlamaları Microsoft, IBM ve Verisign tarafından geliştirilmiştir ve daha sonra Web Servisleri Mimarisi (Web Services Architecture, WSA) olarak adlandırılmıştır. WS-güvenlik tanımlamaları bu alandaki diğer tanımlamalar için temel oluşturmuş, mesaj-tabanlı güvenlik alışverişi için altyapıyı hazırlamıştır. Birlikte çalışabilir web servisleri oluşturmadaki öneminden dolayı, OASIS' e verilmiş ve gerekli komite işlemlerinden sonra, resmi olarak kabul edilmiş bir standart olmuştur.

3.3.1. Güvenli web servis standartları yığını

Web servislerini oluşturan açık standart toplulukları web servisleri için bir dizi güvenlik standartları geliştirmiştir. Şekil 3.2'de, Web servis güvenlik standartları için kurumsal bir referans modeli gösterilmektedir. Bu referans modeli tipik bir web servisin farklı fonksiyonel katmanları için farklı standartları eşleştirmiştir. Bu katmanlar OSI referans modelinden sonra modellenmiştir ancak hiyerarşik olarak tasarlanmamıştır.



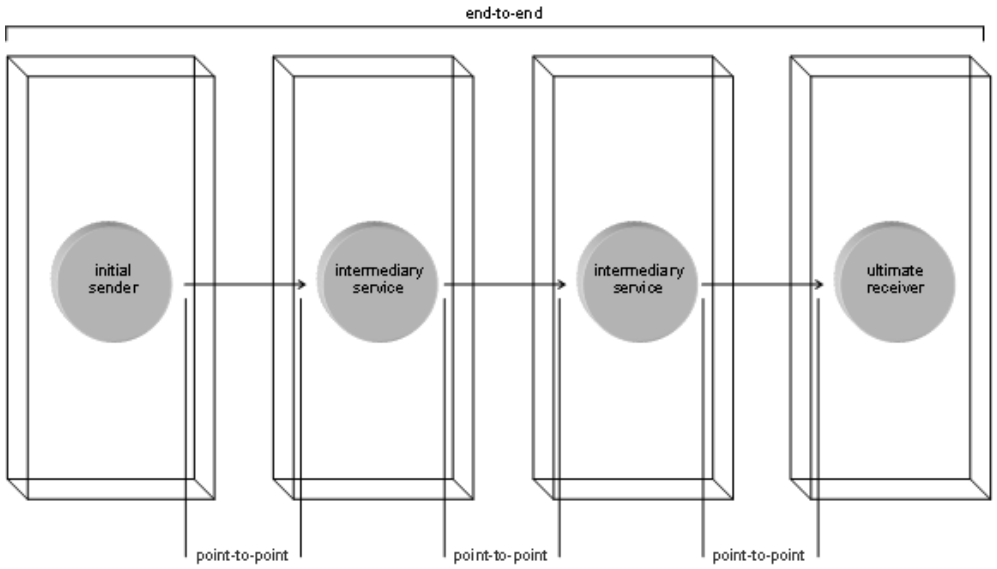
Şekil 3.2. Web servisleri güvenlik standartları yığını

XML güvenlik, taşıma ve ağ katmanındaki standartlar, mesajların ağ üzerinde taşınırken güvenliğini sağlamak için kullanılır. IPsec, SSL/TLS (Secure Sockets Layer/Transport Layer Security), XML şifreleme ve XML imza gibi güvenlik standartları SOAP mesajın farklı seviyelerinde işlem yapmaktadırlar. XML güvenlik katmanının üzerinde SOAP ve bağımsız standartlar üzerinde inşa edilmiş 2 tip standart bulunmaktadır. Mesaj güvenlik standartları WS-Security (WS-güvenlik) ve WS-SecureConversation (WS-güvenli konuşma), XML imzanın, XML şifrelemenin ve kimlik bilgilerinin SOAP güvenliği için nasıl kullanılacağını tanımlar.

Erişim kontrol standartları yalnızca web servisleri için kullanılmaz. XACML herhangi bir sistem için erişim kuralları tanımlayabilirken, SAML herhangi bir ortamda kaynağa erişim için onayları tanımlamaktadır. Kural katmanı WS-Policy (WS-Politikası), bir web servisinin iletişim kurarken ihtiyaç duyduğu kurallar için bir dilbilgisi tanımlar. Güvenlik yönetim özellikleri, SOA içerisindeki PKI sertifikaları gibi kimlik bilgilerini yönetmek için başka web servisleri tanımlar. Kimlik yönetim standartları, SOA içinde kullanıcı kimliklerini ve bilgilerini dağıtmak ve yönetmek için SOAP standartları, erişim kontrol ve kural standartlarından yararlanır [34].

3.3.2. Web servis güvenlik standartları

Web servis güvenlik dili olarak bilinen WS-Güvenlik (WS-Security), ayrı güvenlik modelleri arasında bir köprü oluşturmak için kullanılır. Ancak, SOAP mesajları için uçtan uca (end-to-end) güvenlik modeli sağlamak için geleneksel taşıma katmanı güvenliğinin ötesine geçer. Çünkü servis yönelimli sistemler SOAP mesajın içeriğinin yol boyunca korunmasını sağlayan uçtan uca bir güvenlik modeline ihtiyaç duyarlar. Bu, taşıma düzeyinde güvenliğin genellikle yeterli olduğu geleneksel noktadan noktaya (point-to-point) modelden farklıdır. Şekil 3.3'te, aracı (intermediary) servisleri içeren noktadan noktaya bağlantı dizisi gibi bir mesaj yolu görülmektedir [35] .



Şekil 3.3. Mesaj yolu

Uçtan uca bir mesaj yolunu güvenli yapmak için, WS-Güvenlik mesajla taşınan SOAP başlık bloklarını kullanarak güvenlik önlemleri uygular. Aşağıda WS-güvenlik çalışma alanının bazı standartlarından bahsedilmiştir.

- WS- Policy (WS-Politika) : Bu standart WS-güvenlik çalışma alanı için anahtar kısımdır. Mevcut kurumsal güvenlik politikaları, servis gruplarına uygulanabilen politika kuralları sayesinde ifade edilebilir. WS-Policy, web servislerinin kendi politikalarını (güvenlik, servis kalitesi gibi) tanıtması için ve web servis istemcilerinin kendilerine gereken politikaları belirlemesi için XML'i kullanmalarına izin verir. aracılar ve uç noktalar (örneğin gereken güvenlik anahtarları, şifreleme algoritmaları ve gizlilik kuralları) üzerinde güvenlik politikalarının yeteneklerini ve kısıtlamalarını tanımlar, servisler ve uç noktalarla politikaları nasıl birleştireceğini tanımlayan bir dizi özellik sunar.

WS-Policy dokümanı karmaşık politika gereksinimlerini ve kalitelerini tanımlamak üzere tasarlanmıştır. Yayınlanan politika ile SOAP uç noktaları kendi güvenlik gereksinimlerini yayabilir ve istemcileri de taleplerini hazırlarken gereken uygun mesaj koruma önlemlerini alabilirler. Genel WS-Policy dokümanında (aslında 3 ayrı dokümandan oluşur) ayrıca özel politika türleri için de eklentiler vardır, örneğin güvenlik için WS-SecurityPolicy [36].

- WS-Trust (WS-Güven): Bu özellik, mevcut olan modelleri birleştirmek için kullanılan standart bir güven modeli oluşturur, böylece değiştirilen güvenlik anahtarlarının geçerliliği kontrol edilebilir. WS-Trust bu kontrole yardımcı olmak için istekte bulunan üçüncü kişi için bir iletişim süreci sağlar. Eğer istemci gereken parçacıklara sahip değilse, bu parçacıkları güven mekanizmaları yoluyla alabilirler. WS-trust servisleri, birçok güvenli parçacık değişimi talebinde çağrılır.

WS-Trust, güvenilir mesaj değişiminde katılımcılar arasında güven ilişkisinin oluşmasına aracılık eder, güvenlik anahtarlarının kontrol edilmesiyle, yenilenmesiyle ve düzenlenmesiyle ilgilenir. WS-Trust ta tanımlanan uzantılar kullanılarak, web servis çerçevesinde çalışmak üzere tasarlanmış uygulamalar güvenli iletişim halinde bulunabilir [37].

- WS-SecureConversation (WS-Güvenli konuşma): WS-SecureConversation güvenlik içeriklerinin paylaşılması ve oluşturulması için tasarlanmıştır. Çeşitli güvenlik modelleri web servisleri arasında güvenlik bilgilerinin değişimi sırasında standart bir mekanizma kuran WS-SecureConversation ile desteklenir. İlgili oturum anahtarların oluşturulması ve güvenlik içeriklerinin değişimi için resmi tanımlama sağlar [38].

- WS-Federation (WS-Anlaşma): WS-Security, WS-Policy ve WS-Trust standartları kullanıldığında, farklı güven alanlarını birleştirmek için birçok yol bulunur. WS-Federation, anlaşma oluşturmak için bir dizi standart ve güvenlik modeli sağlar. Amacı etki alanları arası güvenli ilişkiler kuran mekanizmayı sağlamaktır. Böylece kullanıcı kendi etki alanında tüm gerekli izinleri aldığında, başka bir etki alanı altında bulunan servislere ve uygulamalara aynı izinler ile bağlanabilmektedir. Bu olay tek bir sign-on işlemi ile uygulamalar ve etki alanları arasında tekrar eden kullanıcı hesapları durumunu ortadan kaldırır ve uygulamayı genişletirken maliyet yükünü düşürür [39].

- WS-Reliability (WS-Güvenilirlik): Web servislerinin bazı uygulamaları için güvenilir mesajlaşma ihtiyaçlarını karşılar. HTTP üzerinden SOAP yeterli değildir. Bir uygulama katmanı mesajlaşma protokolü güvenilirlik ve güvenliğin bazı katmanlarını güvence altına almalıdır. Bu özellik, geçerli web servis standartlarının içeriklerine güvenilirlik tanımlar. Bu özellik diğer tamamlayıcı protokoller ile birlikte kullanım için tasarlanmıştır [40].

- WS-Adressing (WS-Adresleme) : Web servis uç noktalarını belirtmek ve mesajların uçtan uca güvenliğini sağlamak amacıyla XML öğeler tanımlar. Mesajların taşınması sırasında doğal bir iletim ortamı oluşturur.

BÖLÜM 4. WEB SERVİS TEMELLİ GÜVENLİK ALTYAPI MODELİ - WSGA

Önceki bölümlerde bahsedilen güvenlik yöntemlerinden hiçbiri dağıtık web uygulamalarında güvenilir web servisleri sağlamada tek başına yeterli olamamaktadır. Bu durum göz önünde bulundurularak içerisinde bu güvenlik fonksiyonlarının hepsinin kullanıldığı güvenilir bir mimari önerilmiştir. Tasarlanan modele WSGA-Web Servis tabanlı Güvenlik Altyapı Modeli adı verilmiştir.

4.1. WSGA Modelinin Özellikleri

WSGA modelinin mimari özellikleri şu şekildedir:

a) Model web servis mimarisini ve standartlarını temel almaktadır.
b) Web servis güvenlik standartları kullanılmaktadır.
c) Web servisleri kullanılarak Servis Yönelimli Mimari (SOA) oluşturulması amaçlanmıştır.
d) Birbirlerine olan bağımlılıkları az, ayarlamaların kolay yapılabildiği servis yönelimli bir alt yapı hedeflenmiştir.
e) Modelde bulunan servisler, veri tabanları ve uygulamalar aynı lokasyonda bulabilecekleri gibi dağıtılabilen bir yapıya da sahiptirler.
f) Web servis güvenlik fonksiyonları web servisleri şeklinde tasarlanmış, web servislerinin gücünden yararlanarak uçtan uca ve uygulama katmanında güvenliği sağlamaktadır.
g) Modelde, GoF tasarım desenlerinden biri olan Facade deseni mantığından yararlanılmıştır.
h) Model içerisindeki bütün iletişim web servis tabanlıdır.
i) Modelde bulunan kullanıcıların ulaşmak istedikleri servis ve uygulamaların güvenliği, yine model içerisinde bulunan güvenlik alt yapısı sayesinde sağlanmaktadır.

j) Kurumsal uygulamalar için merkezileştirilmiş güvenlik servisleri ve veri depoları (güvenlik ayarları, kullanıcı bilgileri, sertifikalar, raporlar) sağlamaktadır.
k) Modelin güvenlik servislerinin tek noktadan yönetebilme özelliği sayesinde her türlü uygulamaya kolayca entegre edilebileceği düşünülmektedir.
l) Her türlü güvenlik yönteminin sistemde kullanılabilmesi sayesinde, servis yönelimli mimariye geçmek isteyen kurumların alt yapılarına kolayca uyum sağlayabilmektedir.

4.2. WSGA Modelinde Bulunan Kavramlar

4.2.1. Facade tasarım deseni

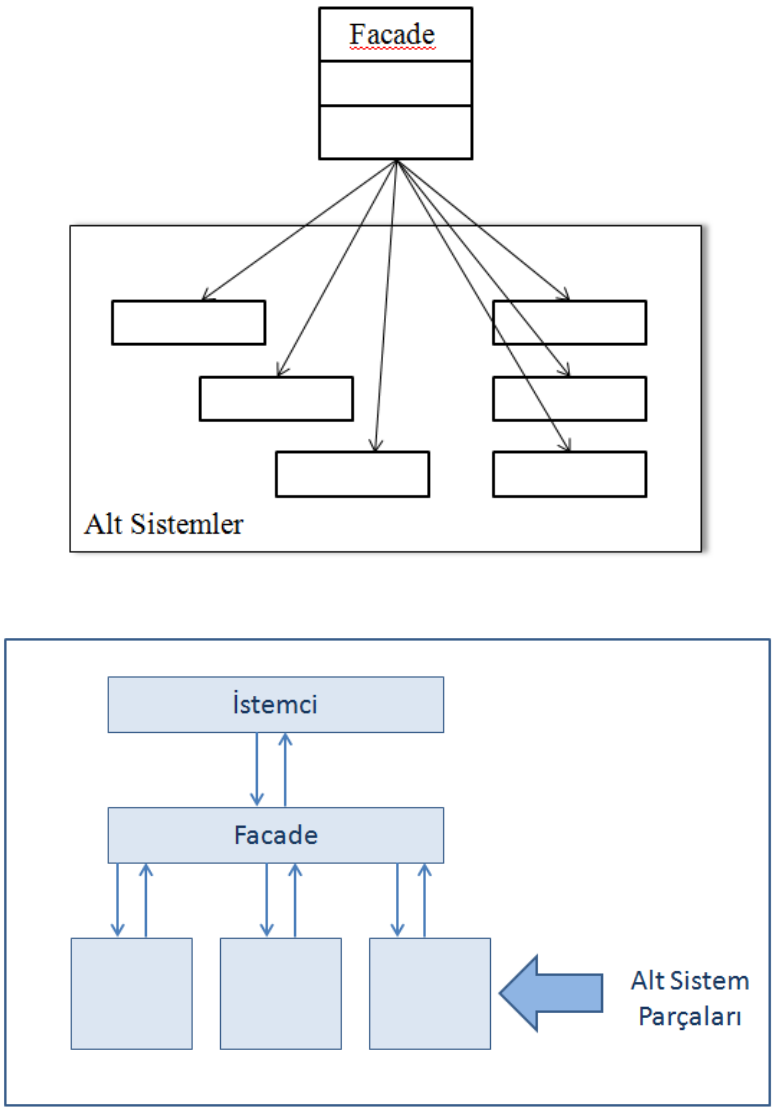
Web servislerini güvenli hale getirmeyi amaçlayan tasarım, güvenlik fonksiyonlarını web servisleri olarak kullanıp kurumsal bir alt yapı teknoloji modeli sağlamayı hedeflemiştir. Mimari tasarlanırken GoF (Gang of Four) tasarım desenlerinden Facade tasarım deseninin mantığından yararlanılmıştır.

Facade deseni alt sistemlerin üst sistemler tarafından uygulanması sırasında yapılacak işlemlerin daha basit bir kullanımla ifade edilerek tekrar yazılmasıdır. Yani alt sistemlerin karmaşıklığı içerisinde kaybolmadan, sadece gerekli parametreleri temel alarak diğer sistemler için ideal bir kullanım ortamı sağlamak bu desenin temel işlevidir.

Desenin kullanım amacı, istemcilerin sistemimizin alt parçaları olan sınıflar, servisler veya metotlarla direkt olarak iletişime geçmelerini engellemektir. Sistemimizin alt parçaları, istemcilerin doğrudan kullanımına kapalıdır. İstemci ve alt sistemlerin ilişkisi başka bir katman (*facade*) tarafından yönetilmektedir. Böylelikle sistem güvenliği artırılmakla beraber, kullanım kolaylığı da artmaktadır. Sistem alt parçaları, tamamen birbirinden bağımsız olduğundan dolayı, her bir parça bu sistemden alınıp, başka sistemlere kolayca uygulanabilmektedir. Hatta bir alt sistem öğesi, birden fazla facade tarafından da kullanılabilme imkânına sahiptir.

Facade katmanı sistem için yararlı olmasına rağmen zorunlu değildir. Çünkü facade, sistemde asıl işi yapan katman değil, asıl işi yapan katman ile kullanıcı arasında çalışan bir ara katmandır. Her bir alt sistem, kendi içerisinde başarıyla çalışmaktadır. Bu nedenle, facade katmanı sistemden çıkarıldığında, alt sistemler çalışmaya devam edeceklerdir. İlaveten, facade katmanı, alt sistemleri referans almasına rağmen, bunun tersi kesinlikle yanlıştır. Eğer, alt sistemler, facade katmanını referans alırlarsa, bu katmanının sistemdeki zorunluluğu artacaktır. Bu ise Facade'ın kullanım mantığına aykırıdır [41].

Facade desenin UML diyagramı ve bileşenleri Şekil 4.1'de gösterildiği gibidir.



Şekil 4.1. Facade Deseni UML Diyagramı ve bileşen parça gösterimi

4.2.2. Simetrik şifreleme algoritmaları

Simetrik şifrelemede genelde şifreleme ve şifre çözme işlemleri bir anahtar (key) sayesinde olur. Bu anahtar iki taraf tarafından daha önceden bilinen bir bilgidir ve bir tarafın anahtar ile şifrelediği bilgi diğer taraftaki anahtar ile açılabilir. Gönderilecek gizli metinle beraber üstünde anlaşılmış olan gizli anahtar da alıcıya gönderilir ve şifre çözme işlemi gerçekleştirilir.

Simetrik şifrelemenin en önemli avantajlarından birisi oldukça hızlı olmasıdır. Asimetrik şifrelemeye göre hız konusunda daha başarılıdırlar. Ayrıca simetrik algoritmalarda kullanılan anahtarın boyu ve dolayısıyla bit sayısı çok daha küçüktür. RijndaelManaged, RC2, DES ve TripleDES simetrik algoritmalara örnek olarak verilebilir

Simetrik algoritmalar blok şifreleme ve dizi şifreleme algoritmaları olarak ikiye ayrılmaktadır.

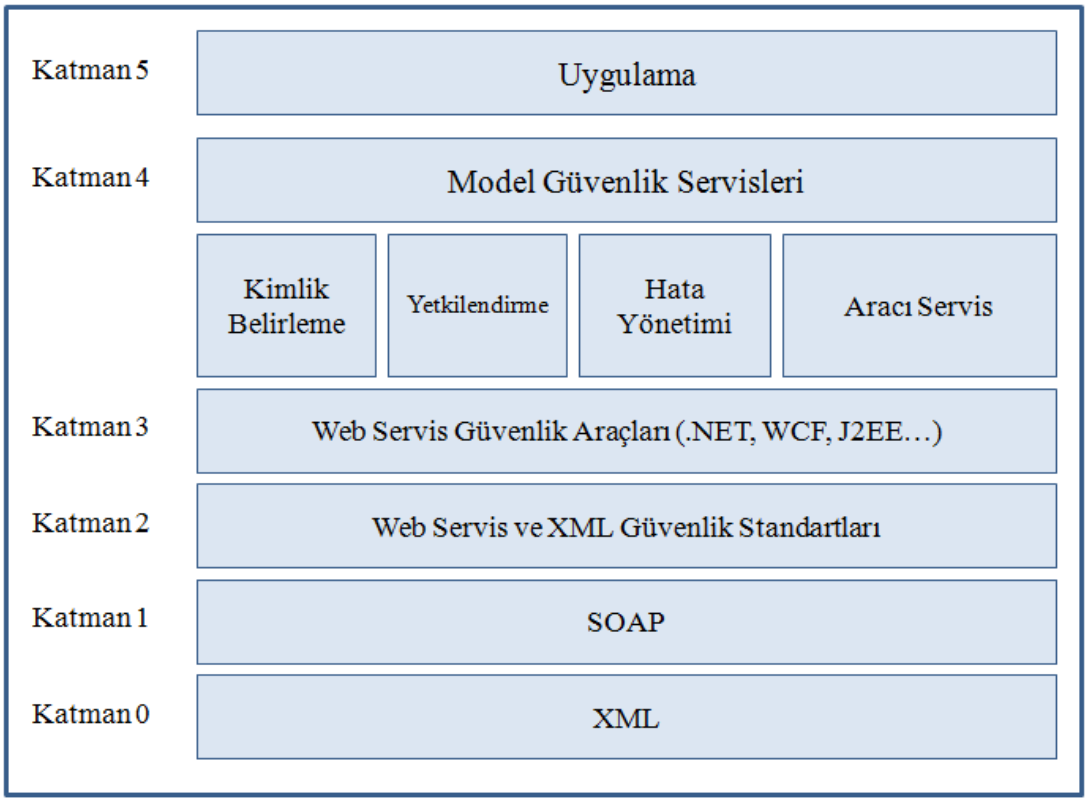
- Blok Şifreleme Algoritmaları veriyi bloklar halinde işlemektedir. Bazen bağımsız bazen birbirine bağlı olarak şifrelemektedir. Bu algoritmalarda iç hafıza yoktur, bu yüzden hafızasız şifreleme adını da almıştır. Bütünlük kontrolü gerektiren uygulamalarda genellikle blok şifreleme algoritmaları tercih edilir.

- Dizi şifreleme algoritmaları ise veriyi bir bit dizisi olarak almaktadır. Bir üreteç aracılığı ve anahtar yardımıyla istenilen uzunlukta kayan anahtar adı verilen bir dizi üretilir. Kayan anahtar üretimi zamana bağlıdır ve bu yüzden bu algoritmalara aynı zamanda hafızalı şifreleme denir. Telsiz haberleşmesi gibi gürültülü ortamlarda ses iletimini sağlamak için genellikle dizi şifreleme algoritmaları kullanılır.

WSGA modelinde Kimlik Belirleme, Yetkilendirme ve Hata Yönetimi Servisleri'nde verileri şifrelemede simetrik algoritmalardan Rijndael Algoritması (AES - Advanced Encrytion Standard - Gelişmiş Şifreleme Standardı**)** kullanılmıştır.

Rijndael algoritması, Belçikalı Vincent Rijmen ve Joan Daemen tarafından bulunmuş, DES'in ve zayıf yönlerini tamamen düzelterek, matematikle oluşturulmuş bir blok şifreleme algoritmasıdır. 128 bit, 192 bit ve 256 bit olmak üzere üç farklı anahtar uzunluğuna sahip olabilir. Çok iyi, çok hızlı ve çok performanslı çalışan ileri boyutta düzgün matematiksel alt yapıya sahip bir algoritma düzeni vardır. Rijndael algoritması, donanımı uygun kullanma açısından da çok iyi bir algoritmadır. Ancak Rijndael algoritmasının iki adet dezavantajı vardır. Bunlardan biri uzun anahtar boyutlarını uzunca bir zamanda şifrelemesidir. İkincisi ise 256 bit anahtar kullanımlarında döngüsel artım olduğu için hızının yüksek oranda düşmesidir. Ancak bunlara rağmen projelerde en çok tercih edilen şifreleme algoritması Rijndael algoritmasıdır [42].

4.3. WSGA Modelinin Web Servis Güvenlik Yığını İçerisindeki Yeri



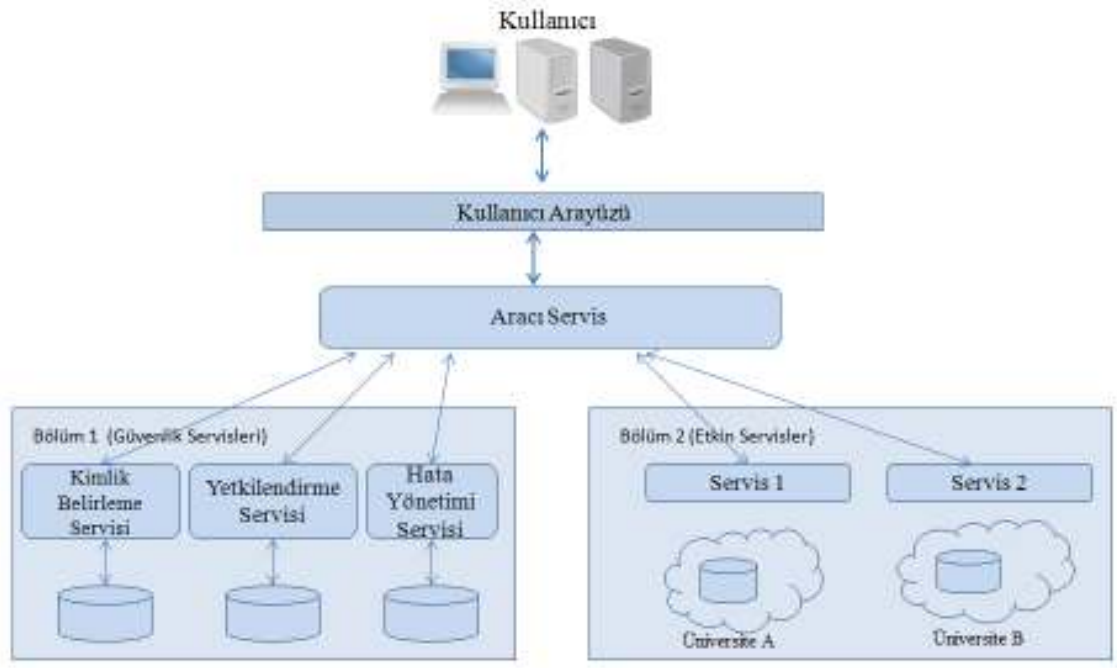
Şekil 4.2. WSGA modelinin web servis güvenlik yığını içerisindeki yeri

4.4. WSGA Modelinin Bölümleri

WSGA modeli üç farklı bölümden oluşmaktadır.

1. Modelin kullanıcıları,
2. Modelin güvenlik servislerinin bulunduğu, Bölüm 1 olarak adlandırılan kısım,
3. Modelde kullanıcıların erişmek istedikleri etkin servislerin bulunduğu, Bölüm 2 olarak adlandırılan alandır.

Şekil 4.3, bu üç bölümün birbirleriyle olan etkileşimini göstermektedir.



Şekil 4.3. WSGA Modeli

WSGA modelinin Bölüm 1 olarak adlandırılan kısmında; web servislerinde güvenlik için kullanılan fonksiyonlar ayrı birer web servisi şeklinde tasarlanmış ve Kimlik Belirleme, Yetkilendirme ve Hata Yönetimi adı verilen 3 adet web servisi oluşturularak sisteme dâhil edilmiştir. Bu servisler birbirlerinden tamamen bağımsız durumdadırlar ve iletişimlerini Aracı Servis adı verilen servis üzerinden yapmaktadırlar. Servislerin bu şekilde kullanılmasının amacı, kullanıcıların belirli uygulamalara erişirken aşması gereken güvenlik fonksiyonlarını aşamalı duruma

getirmektir. Böylelikle daha karmaşık hale gelen sistem, dinleme ya da diğer saldırı durumlarına karşı daha dirençli hale gelebilecektir.

Bölüm 2 olarak adlandırılan kısımda; kullanıcıların direk erişmek istedikleri uygulamalar, veritabanları, servisler vb. bulunabilir. Bu tasarımda temsilen Servis1 ve Servis2 olarak adlandırılan iki adet web servisi ve bunların bağlı oldukları veritabanları bulunmaktadır. Kullanıcının asıl kullanmak, erişmek istediği hizmetler bunlardır. Ancak öncelikle Bölüm 1 de bulunan güvenlik aşamalarını atlamak zorundadırlar.

WSGA Modeli'nde bulunan servisler ve kullanılış amaçları alt bölümlerde açıklanacaktır.

1. Aracı Servis: Aracı Servis önceki bölümde de bahsedilen Facade tasarım deseninin kullanım mantığından esinlenerek oluşturulmuş bir servistir. Bu servis desendeki Facade katmanıyla eş görevdedir. Yani güvenlik servisleri ve etkin servislerin bulunduğu bölüm ile kullanıcı arasındaki doğrudan iletişimi kesmek amacındadır. Ayrıca bu servis sayesinde, Bölüm 1 deki güvenlik servisleri arasındaki ilişki dolaylı hale getirilerek bu servislerin kendi aralarında da güvenliğini arttırmak ve birbirlerine olan bağımlılığı düşürmek hedeflenmiştir. Desendeki Facade katmanı gibi Aracı Servis de bütün servislerden referans almaktadır. Böylece bütün servislerle iletişim halinde olacaktır. Ancak bu durumun tersi mümkün değildir. Burada amaçlanan sistemde bulunan servislerin Aracı Servis'e bağımlı olmasını engellemektir. Yani Aracı Servis sistemden çıkarıldığında sistemin çalışmasında bir aksaklık olmayacaktır. Ancak Aracı Servis'in sistemde var olması, diğer servislerin bu servis üzerinden haberleşmesi ve kullanıcı ile servisler arasına girerek direk iletişimi engellemesi sistemin güvenliğine önemli ölçüde katkı sağlamaktadır.

2. WSGA Modelinin Güvenlik Servisleri: Kullanıcılar, Bölüm 2'de bulunan servislere erişim için istekte bulunduklarında, kimlik ve yetki belirleme işlemleri için öncelikle Bölüm 1'deki servislere yönlendirilirler.

- Kimlik Belirleme Servisi: Kullanıcılar Bölüm2'de bulunan servislere istek gönderdiklerinde, öncelikle kimlik belirleme servisine yönlendirilirler. Kullanıcıların istek mesajları kullanıcı adı ve şifre bilgileri ile birlikte Rijndael algoritmasına göre XML-Şifreleme işlemine tabi tutulur. Gelen şifreli istek paketi burada çözülür. Kimlik Belirleme Servisi'ne ait veritabanından kullanıcı kimlik bilgilerinin kontrolü yapılır. Kullanıcı bilgilerinin sonuna elde edilen olumlu olumsuz cevap da eklenerek mesaj tekrar şifrelenir, Yetkilendirme Servisi'ne iletilmek üzere Aracı Servis'e geri gönderilir.

- Yetkilendirme Servisi: Kimlik Belirleme Servisi'nden gelen şifreli paket burada deşifre işlemine tabi tutulur. Mesajda bulunan Kimlik Belirleme Servisi'nden dönen olumlu/olumsuz cevaba göre bu servise ait veritabanından o kullanıcının grup ya da rol tanımlamasına göre erişimi olup olmadığı kaynaklar belirlenir. Burada oluşturulan cevap paketi içerisine, kullanıcı ve rol bilgileri eklenerek Rijndael algoritması ile paket şifrelenir, ardından Hata Yönetimi Servisi' ne iletilmek üzere Aracı Servis'e gönderilir.

- Hata Yönetimi Servisi: Bu servis, güvenlik aşamalarının son adımıdır. Kullanıcının Bölüm 2'de ki uygulama ya da servislere yapmak erişim için yaptığı istek buradan dönecek cevaba göre belirlenecektir. Bu servis öncelikle, Yetkilendirme Servisi'nden gelen cevabı veritabanına kaydeder, daha sonra eğer gelen mesajda 2 servisten de olumlu cevap bulunuyorsa, kullanıcının isteğine karşılık erişebilir bilgisi Aracı Servis'e geri gönderilir.

3. WSGA Modeli'nde Kullanılan Etkin Servisler: Modelde Servis 1 ve Servis 2 olarak adlandırılan bu servisler kullanıcıların direk iletişim kurmak istedikleri servislerdir. Bunlar farklı iki üniversitenin e-öğrenme platformu şeklinde düşünülmektedir. Her bir servisin veri alışverişi yapabildiği, bu iki üniversitenin uzaktan eğitim derslerinin bulunduğu kendine ait veri tabanları bulunmaktadır. Bu servisler temsili olarak oluşturulmuştur. Daha farklı konularda, kurumların her türlü ihtiyaçlarına göre değiştirilebilirler.

4. WSGA Modelinin Kullanıcıları: Modelde 2 tip kullanıcı bulunmaktadır. Bunlardan ilki, sistem üzerindeki servislerde her türlü yapılandırmayı yapabilen yönetici rolü, diğeri ise etkin servis kullanıcılarıdır. Kullanıcılar Bölüm 2'de bulunan servislere erişmek isterler. Oluşturulan senaryoda, birden fazla uzaktan eğitim platformu birleştirilerek kullanıcılara daha geniş bir yelpazede daha fazla bilgi sunulması amaçlanmıştır. Kullanıcılar sisteme dâhil olabilmek için öncelikle Bölüm 1'de oluşturulmuş güvenlik kriterlerini geçmek zorundadırlar. Ancak bu durum kullanıcıların Bölüm 2'de bulunan servisleri kullanırken sisteme bir kez dâhil olmalarını sağlayarak, son zamanlarda çok problem olmaya başlayan birden fazla kullanıcı adı ve şifre ezberleme olayına alternatif bir çözüm sunmaktadır.

BÖLÜM 5. UYGULAMA

5.1. Kullanılan Teknolojiler

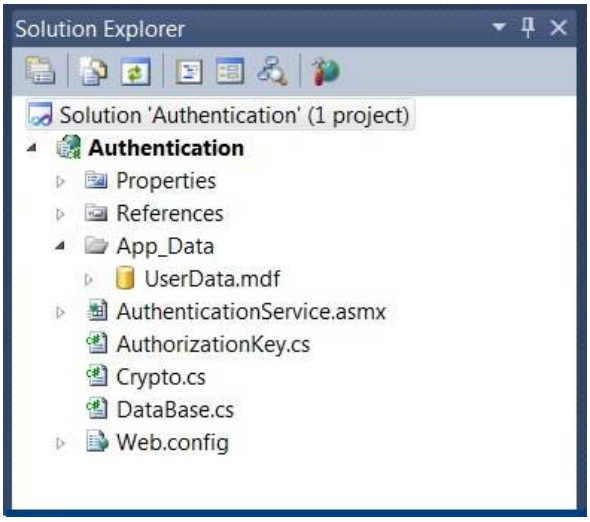
Modelin web servisleri C# programlama dili (Microsoft C#) kullanılarak, Visual Studio 2010 (Microsoft VS, 2010) ortamında ve .NET platformunda geliştirilmiştir. Sistem için gerekli kullanıcı kimlik ve rol bilgileri, bunların yanında sistemin işleyişi için tutulan kayıtlar, Microsoft SQL Server 2008 veritabanı kullanılarak saklanmıştır. Ayrıca modelin kullanıcıları için geliştirilen platformlar ise ASP.NET (Microsoft ASP.NET) ile oluşturulmuştur.

5.2. Uygulama

Tasarlanan mimarinin uygulaması için oluşturulan projeler ve çözüm dosyaları aşağıdaki gibidir:

- Güvenlik Servisleri**:** WSGA Modeli'nde geliştirilen XML güvenlik servislerinin hepsi (Kimlik Belirleme, Yetkilendirme, Hata Yönetimi) kullanıcılar ile haberleşme esnasında XML-Şifreleme (XML-Encryption) yöntemini kullanırlar. Şifreleme, simetrik şifrelemede önemli algoritmalardan biri olan Rijndael Algoritması kullanılarak yapılmaktadır.

1- Kimlik Belirleme Servisi:



Şekil 5.1. Kimlik Belirleme Servisi çözüm mimarisi

Kimlik Belirleme Servis'inde; kullanıcı tarafından girilen kullanıcı adı ve şifrenin kontrolü yapılarak, kullanıcıların sistemde kayıtlı olup olmadıkları anlaşılır. Kullanıcı adı ve şifre paketi servise şifrelenmiş olarak gelir. Burada şifre çözülür ve veritabanından kontrolü yapılır. Elde edilen cevap tekrar şifrelenerek ilgili servise geri gönderilir. Projede kullanılan sınıflar ve özellikleri şu şekildedir:

Crypto.cs: Bu sınıf içerisinde bu servisten dönen mesajların şifrelendiği ve şifreli gelen mesajların şifrelerinin çözüldüğü metotlar bulunmaktadır.

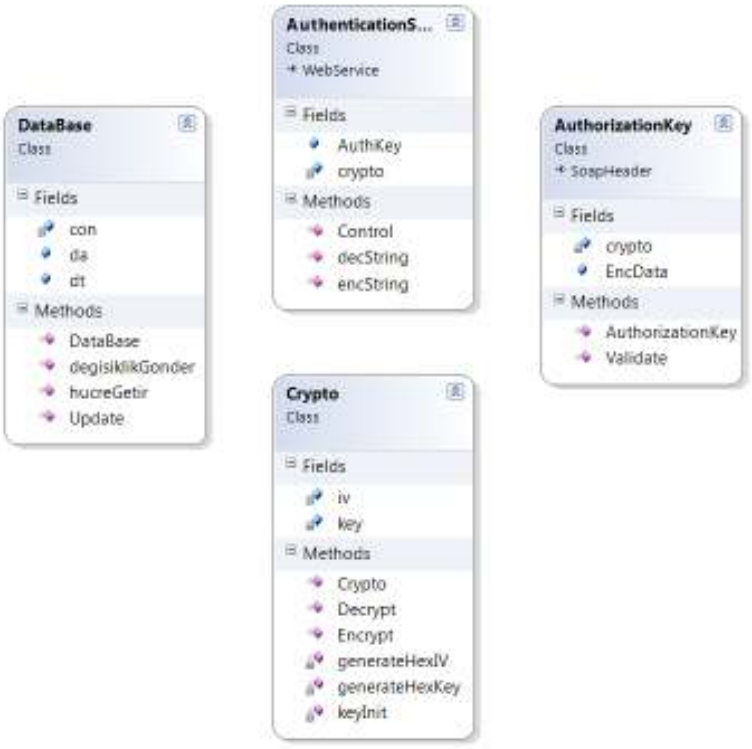
AuthorizationKey.cs: Şifreleme yapılırken dışarıdan gelebilecek dinlemeler ve araya sızmaların engellenebilmesi için mesajların zaman damgası içermesi gerektiği düşünülmüştür. Bu nedenle geliştirilen bu sınıfta öncelikle mesajın geçerli bir tarih ve saat içerip içermediği, ardından ise yapılan isteğin yalnızca 5 dakika geçerli olmasını sağlamak amacıyla oluşturulmuş, geçerlilik metodu bulunmaktadır.

DataBase.cs: Bu sınıfta veri tabanı bağlantılarının yapılması için gereken bağlantı ayarları ve veritabanından veri çekmek, veri eklemek ve güncelleme yapmak için geliştirilmiş metotlar bulunmaktadır.

UserData.mdf : Kullanıcı bilgilerinin tutulduğu veritabanıdır.

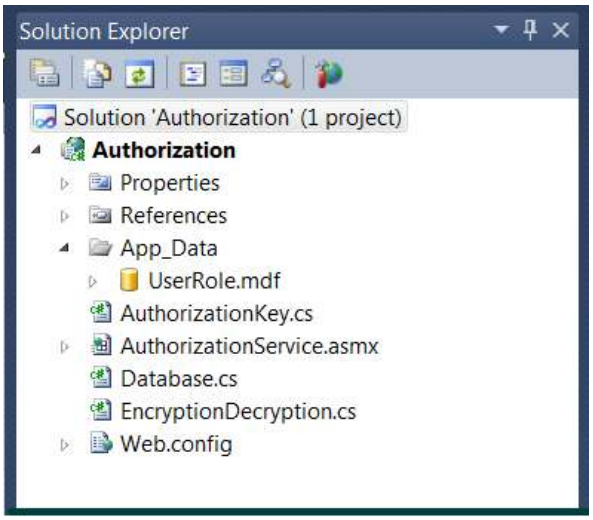
Authentication.asmx.cs: Bu servis sınıfı içerisinde yukarıda bahsedilen sınıflar ve metotları kullanılarak gelen şifreli mesajın çözüldüğü, veri tabanından kontrolün yapıldığı, ve ardından zaman damgası eklenerek sonucun tekrar şifrelendiği bir kontrol metodu bulunmaktadır.

Kimlik Belirleme Servisi projesinin sınıf diyagramı Şekil 5.2'de gösterildiği gibidir. Ayrıca Kimlik Belirleme Servisi WSDL Belgesi tezin EK-B kısmında bulunmaktadır.



Şekil 5.2. Kimlik Belirleme Servisi Sınıf Diyagramı

2- Yetkilendirme Servisi:



Şekil 5.3. Yetkilendirme Servisi Çözüm Dosyaları

Kimlik Belirleme Servisi'nden gelen, kullanıcı onay/hata bilgisinin eklendiği şifreli cevap Aracı Servis üzerinden Yetkilendirme Servisi'ne gelir. Burada, bu servise ait veritabanından kullanıcıların rol bilgilerine ve erişim izinlerine ulaşılır. Servise gelen istek mesajları şifrelenerek alınır, cevap mesajları da şifrelenerek yollanır. Yetkilendirme Servisi projesinde kullanılan sınıflar ve özellikleri şu şekildedir:

EncryptionDecryption.cs: Bu sınıf içerisinde bu servisten dönen cevapların şifrelendiği ve şifreli gelen mesajların şifrelerinin çözüldüğü metotlar bulunmaktadır.

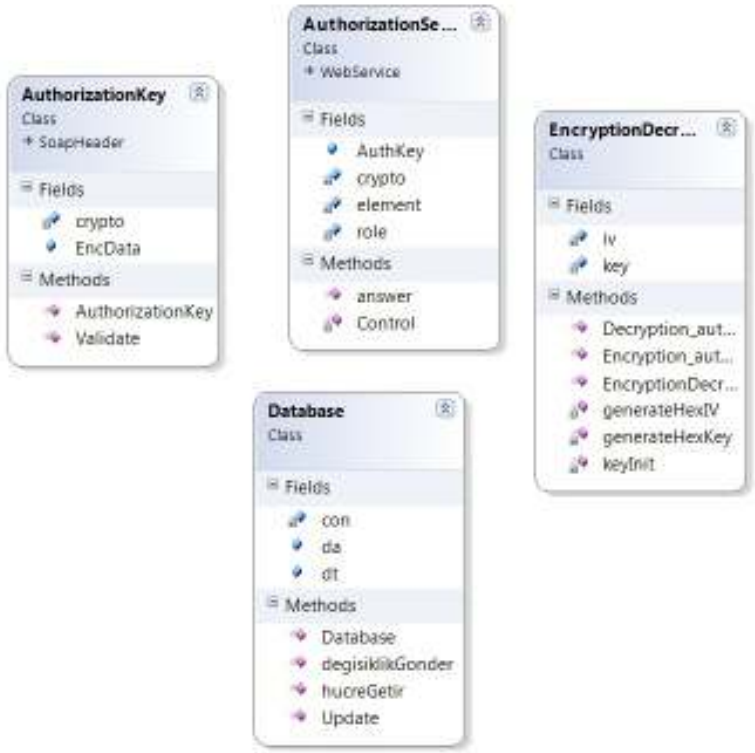
AuthorizationKey.cs: Şifreleme yapılırken dışarıdan gelebilecek dinlemeler ve araya sızmaların engellenebilmesi için mesajların zaman damgası içermesi gerektiği düşünülmüştür. Bu nedenle geliştirilen bu sınıfta öncelikle mesajın geçerli bir tarih ve saat içerip içermediği, ardından ise yapılan isteğin yalnızca 5 dakika geçerli olmasını sağlamak amacıyla oluşturulmuş, geçerlilik metodu bulunmaktadır.

Database.cs: Bu sınıfta veri tabanı bağlantılarının yapılması için gereken bağlantı ayarları ve veritabanından veri çekmek, veri eklemek ve güncelleme yapmak için geliştirilmiş metotlar bulunmaktadır.

UserRole.mdf : Kullanıcı rol bilgilerinin tutulduğu veritabanıdır.

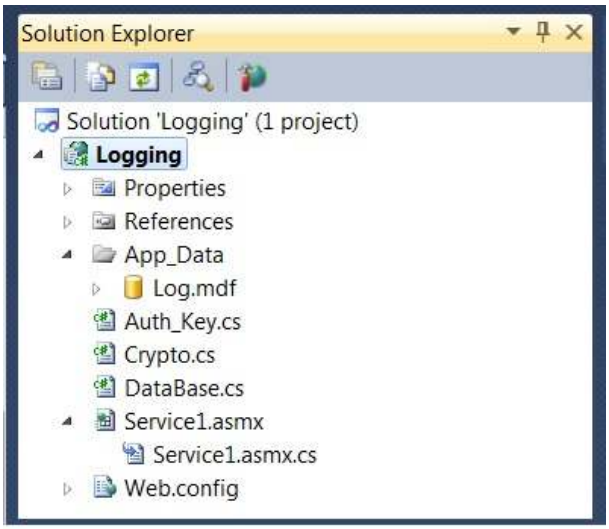
Authorization.asmx.cs: Bu servis sınıfı içerisinde yukarıda bahsedilen sınıflar ve metotları kullanılarak gelen şifreli mesajın çözüldüğü, veri tabanından rol kontrollerinin yapıldığı, ve ardından zaman damgası eklenerek sonucun tekrar şifrelendiği kontrol ve cevap metotları bulunmaktadır.

Yetkilendirme Servisi projesinin sınıf diyagramı Şekil 5.4'de gösterildiği gibidir. Ayrıca Yetkilendirme Servisi WSDL Belgesi tezin EK-C kısmında bulunmaktadır.



Şekil 5.4. Yetkilendirme Servisi Sınıf Diyagramı

3- Hata Yönetimi Servisi:



Şekil 5.5. Hata Yönetimi Servisi Çözüm Dosyaları

Hata Yönetimi Servisi, güvenlik servisleri içerisindeki son aşamadır. Kimlik Belirleme ve Yetkilendirme Servisi'nden gelen cevaplara göre kullanıcının istekte bulunduğu uygulama ve servislere erişimleri buradan dönecek cevaba göre belirlenir. Çünkü Kimlik Belirleme Servisi'nden dönen olumlu/olumsuz cevap, Yetkilendirme Servisi'nden rol ve erişim izinleri burada veritabanına kaydedilir. Daha sonra kullanıcının isteğine karşılık oluşturulan ulaşabilir/ulaşamaz cevabı, kullanıcıya bildirilmek üzere Aracı Servis'e gönderilir. Kısaca kullanıcının isteğine buradan gelecek cevap ile karşılık verilir. Hata Yönetimi Servisi projesinde kullanılan sınıflar ve özellikleri şu şekildedir:

Crypto.cs: Bu sınıf içerisinde bu servisten dönen cevapların şifrelendiği ve şifreli gelen mesajların şifrelerinin çözüldüğü metotlar bulunmaktadır.

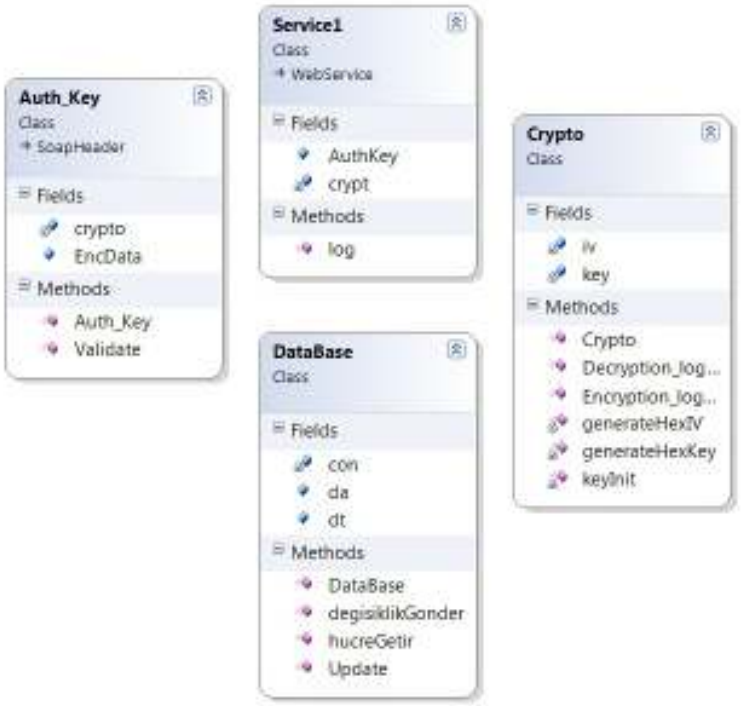
Auth_Key.cs: Şifreleme yapılırken dışarıdan gelebilecek dinlemeler ve araya sızmaların engellenebilmesi için mesajların zaman damgası içermesi gerektiği düşünülmüştür. Bu nedenle geliştirilen bu sınıfta öncelikle mesajın geçerli bir tarih ve saat içerip içermediği, ardından ise yapılan isteğin yalnızca 5 dakika geçerli olmasını sağlamak amacıyla oluşturulmuş, geçerlilik metodu bulunmaktadır.

DataBase.cs: Bu sınıfta veri tabanı bağlantılarının yapılması için gereken bağlantı ayarları ve veritabanına veri eklemek için geliştirilmiş metotlar bulunmaktadır.

Log.mdf : Kimlik Belirleme ve Yetkilendirme Servisi'nden dönen olumlu/olumsuz mesajların tutulduğu veritabanıdır. Hatalı girişlerin tutulduğu hata tablosuyla, onaylanmış girişlerin ve rollerin tutulduğu onay tabloları bulunmaktadır.

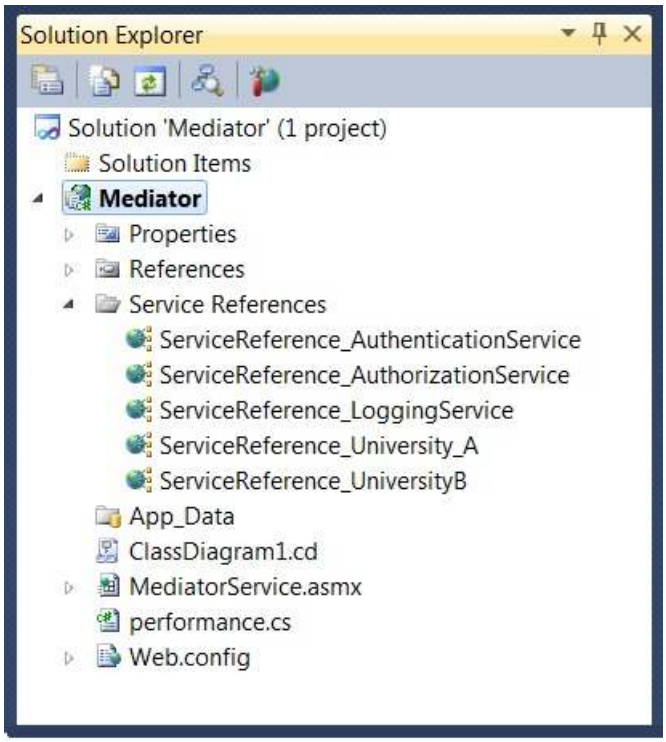
Service1.asmx.cs: Bu servis sınıfı içerisinde yukarıda bahsedilen sınıflar ve metotları kullanılarak gelen şifreli mesajın çözüldüğü, veri tabanına kayıtların yapıldığı, ve ardından zaman damgası eklenerek sonucun tekrar şifrelendiği kayıt metodu bulunmaktadır.

Hata Yönetimi Servisi projesinin sınıf diyagramı Şekil 5.6'da gösterildiği gibidir. Ayrıca Hata Yönetimi Servisi WSDL Belgesi tezin EK-D kısmında bulunmaktadır.



Şekil 5.6. Hata Yönetimi Servisi Sınıf Diyagramı

4- Aracı Servis:



Şekil 5.7. Aracı Servis Çözüm Dosyaları

Aracı servis, şekilde de görüldüğü gibi güvenlik için kullanılan servislerin hepsinden referans almaktadır. Bu şekilde kullanılmasının iki türlü yararı bulunmaktadır. Bunlardan ilki referans aldığı üç servisin birbirleri ile iletişimini kesmiş olmasıdır. Böylece Aracı Servis diğer servislere ait metotlardan yalnızca gerekli olanlara ulaşmakta, servislerin birbirlerine ait özellikleri ve metotları görmelerinin önüne geçmektedir. Diğer bir yararı ise kullanıcılar ile güvenlik servisleri arasındaki bağlantıyı kendi üzerinden yaparak üç güvenlik servisini kullanıcıdan soyutlamasıdır. Bu iki türlü soyutlama sayesinde servislerin kullanımını kolaylaşmakta ve sistem güvenliği artmaktadır.

Aracı Servis bütün servislerden referans almakla birlikte bu durumun tersi mümkün değildir. Çünkü sistem üzerinde Aracı Servis bulunması zorunlu bir servis değil, yalnızca sistemin çalışmasını kolaylaştıran ara bir katmandır. Bu servis sistemden çıkarılsa dahi sistemin çalışmasında bir aksaklık olmamaktadır. Daha önceki bölümlerde de bahsedildiği gibi, Aracı Servis kullanımı bu yönüyle Facade Tasarım Deseni'ne benzemiştir.

Facade deseninde, istemciler ile sistemin alt parçaları arasındaki iletişim facade katmanı sayesinde engellenmektedir. Sistemin alt parçaları, istemcilerin doğrudan kullanımına kapalıdır. Sistem alt parçaları, tamamen birbirinden bağımsız olduğundan dolayı, her bir parça bu sistemden alınıp, başka sistemlere kolayca uygulanabilmektedir. Hatta bir alt sistem öğesi, birden fazla facade tarafından da kullanılabilme imkânına sahiptir.

Aracı Servis üzerinde bulunan sınıflar ve özellikleri şu şekildedir:

Performance.cs: Bu sınıf sistemin çalışma performansını ölçmek için geliştirilmiştir. İçerisinde servislerin yapılan isteklere karşılık verdikleri cevap sürelerini ölçüp bir dosyaya yazdıran performans metodu bulunmaktadır.

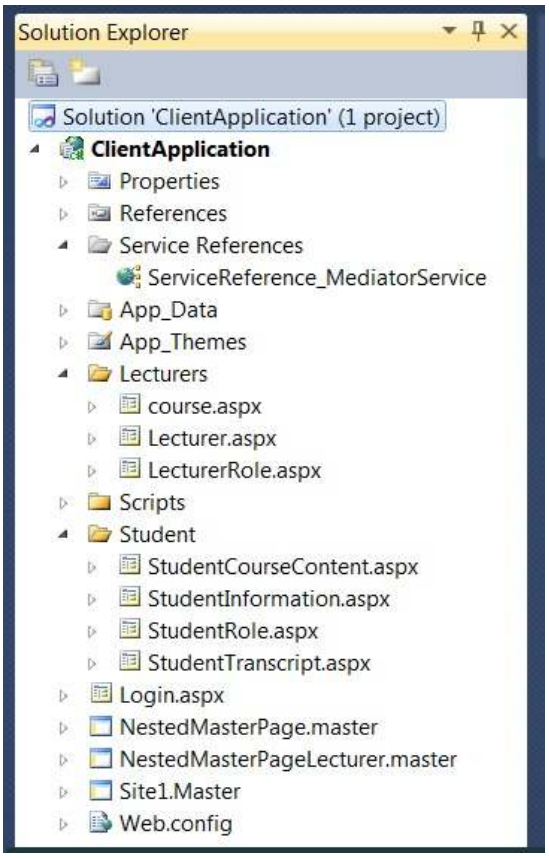
MediatorService.asmx.cs: Bu sınıf içerisinde Kimlik Belirleme Servisi'nden gelen cevabı Yetkilendirme Servisi'ne aktaran, Yetkilendirme Servisi'nden gelen cevabı Hata Yönetimi Servisi'ne aktaran, en son olarak da Hata Yönetimi Servisi'nden dönen cevaba göre kullanıcıyı yönlendirdiği iç adet metot bulunmaktadır.

Aracı Servisi projesinin sınıf diyagramı Şekil 5.8'de gösterildiği gibidir. Ayrıca Aracı Servis WSDL belgesi tezin EK-A kısmında bulunmaktadır.

Şekil 5.8. Aracı Servis Sınıf Diyagramı

- Etkin Servisler: Bu bölümde kullanıcıların direk erişmek istedikleri, modelde de Servis1 ve Servis2 olarak gösterilmiş olan 2 adet web servisi bulunmaktadır. Bu servisler gerçeklenirken Servis 1 olarak adlandırılan servis Üniversite A, Servis 2 olarak belirtilen servis ise Üniversite B olarak tasarlanmıştır. İki üniversitenin de kendine ait, öğrenci ve ders bilgilerinin tutulduğu veritabanları bulunmaktadır.

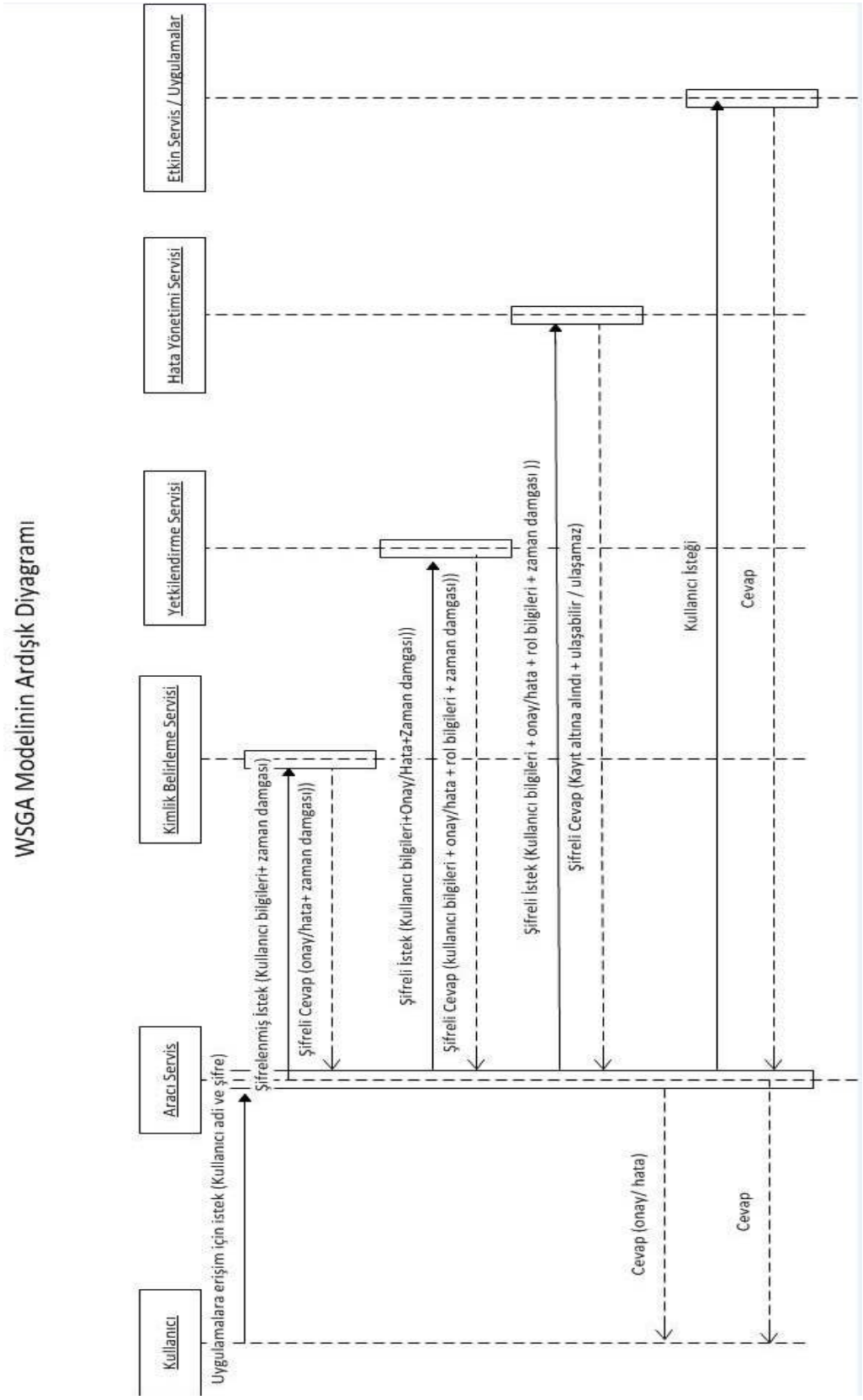
- Kullanıcı Arayüzü: Kullanıcı arayüzü güvenlik servisleri ve etkin servisler ile direk iletişim halinde değildir. Kullanıcılar yalnızca Aracı Servis metotlarını görebilmektedir. Şekil 5.9'da kullanıcı uygulamasının çözüm dosyaları gösterilmiştir.



Şekil 5.9. Kullanıcı uygulaması çözüm dosyaları

Şekil 5.9'da görüldüğü gibi kullanıcı uygulaması yalnızca Aracı Servis'ten referans almaktadır. Böylece asıl kullanılacak servisler ile direk iletişimi kesilmiştir. Kullanıcı uygulamasının ekran görüntüleri tezin EK-E bölümünde bulunmaktadır.

5.3. WSGA Modelinin Ardışık Diyagramı



Şekil 5.10. WSGA modelinin ardışık diyagramı

BÖLÜM 6. MODELİN GÜVENLİK VE PERFORMANS ANALİZİ

6.1. Güvenlik Değerlendirmesi

Bir sistemin güvenliğinin sağlanması için gereken önlemlerin neler olacağına, genel olarak sistemin karşı karşıya kaldığı tehditlere göre karar verilir. Web servislerinin güvenliği için kullanılan etkili ve sağlıklı olarak nitelendirilen birçok önlem olmasına rağmen, birden çok bağımsız çalışma ortamının bulunduğu servis yönelimli mimarilerde bu önlemler pek etkili olamayabilmektedir. Bu sebeple kurumların hangi tehditlerle karşı karşıya olduğu iyi analiz edilmeli ve alınacak güvenlik önlemleri ona göre belirlenmelidir.

Web servislerinin karşılaştığı en önemli tehditler şunlardır:

- Mesajın değiştirilmesi: Saldırgan, alıcıyı aldatmak için mesaj üzerinde değişiklik yapabilir, bilgi ekleyip çıkarabilir.
- Gizlilik kaybı: Mesajın içeriğinin yetkisiz kişiler tarafından ifşa edilmesidir.
- Sahte mesajlar: Saldırganın hayali bir mesaj oluşturarak, alıcıyı mesajın geçerli bir kullanıcı tarafından yollandığına inandırmaya çalışmasıdır.
- Ortadaki adam: Saldırgan alıcı ve gönderici arasına üçünü bir kişi olarak yerleşir. Mesajları değiştirerek, alıcı ve göndericiden habersiz mesaj iletimine devam eder. Böylece onları güvenilir bir iletim ortamı olduğuna inandırır.
- Sızma: Saldırgan mesajı oluşturur ve belirli kimlik bilgileri ile onu gönderir. Böylece mesaj yetkili ve farklı bir kişi tarafından gönderilmiş görünür.
- Sahte istekler: Saldırgan alıcı için geçerli görünen ancak yanlış kimlik bilgileri ile bir mesaj oluşturur.
- Mesaj tekrarı: Saldırgan daha önce gönderilen mesajları tekrar gönderir.

- Mesaj bölümleri tekrarı: Saldırgan daha önce gönderilen mesajların belirli bölümlerini kullanarak yeni bir mesaj oluşturur.

- Servis durdurma: Saldırganın sisteme kaynakların tükenmesine yol açacak kadar istek yollamasıdır. Böylece servisi durdurarak atak yapabilecektir [43].

Web servislerin karşılaştığı bu tehditlere karşı, göndericinin ya da alıcı tarafında güvenliğin sağlanması, daha önceki bölümlerde de anlatılan web servis güvenlik standartları sayesinde yapılmaktadır. Ancak tabi ki her biri bütün tehditlere etkili yanıt veremezler. Dolayısıyla bu standartların sistemin bütününe ve ihtiyaçlarına uygun olarak birleşik halde kullanılması uygun olmaktadır. Tablo 6.1'de web servis güvenliği standartlarının hangi saldırılar üzerinde etkili olduğu gösterilmiştir.

Tablo 6.1. Olabilecek tehditlere karşı web servis güvenlik standartlarının etkileri [43, 3]

	Mesajın değiştirilmesi	Gizlilik kaybı	Sahte mesajlar	Ortadaki adam	Sızma	Sahte istekler	Mesaj tekrarı	Mesaj bölümleri tekrarı	Servis durdurma
XML-Şifreleme		✓		✓	✓	✓		✓	
XML-İmza	✓		✓		✓	✓	✓	✓	
WS-Güvenlik	✓	✓		✓					
WS-Adresleme(WS-Addressing)								✓	
SSL/TLS	✓	✓	✓	✓	✓	✓		✓	
SSL/TLS kullanıcı kimlik belirleme ile	✓	✓	✓	✓	✓	✓		✓	
HTTP Kimlik Belirleme			✓		✓	✓			

Servis yönelimli mimarilerde birden fazla sistemin bir arada çalışması güvenlik açısından birçok zayıflık ortaya çıkarmaktadır. Dolayısıyla güvenlik yöntemlerinin bir arada kullanılması servis yönelimli uygulamalar üzerinde daha etkili sonuçlar yaratmaktadır. Bu çalışmada oluşturulan mimari üzerinde her bir güvenlik standardının kolayca uygulanabilmesi amaçlanmıştır. Yani Tablo 6.1'de gösterilen

güvenlik konusundaki her bir çalışma alanı mimariye rahatlıkla uygulanabilir. Dolayısıyla yalnızca tek bir güvenlik standardı yerine her bir açık için birden çok güvenlik önlemi kullanılabilir. Öncelikle sistemin güvenlik ihtiyacı analiz edilir ve ardından web servis güvenliğinde kullanılan her bir standart, bu istekler doğrultusunda mimari üzerinde uygulanabilir. Mimariyi birçok çalışmadan ayıran en önemli yönü de bu esnek yapıya sahip olmasıdır. Bu durum da daha güvenilir bir ortam oluşmasını sağlamaktadır.

[3, 44, 45] gibi SOA güvenliği ile ilgili yapılan çalışmalarda da, sistemler üzerinde güvenliği sağlamak için birden fazla güvenlik yöntemi bir arada kullanılmıştır. Öncelikle WSGA modelindeki gibi sistemler üzerindeki tehditler analiz edilmekte ve ihtiyaçlar doğrultusunda güvenlik yöntemleri ile bir karşı önlem çalışma alanı oluşturulmaktadır. Bu şekilde sistemin diğerlerine kıyasla daha güvenilir olduğu gösterilmiştir.

Bunların yanı sıra WSGA modelinde olduğu gibi, [1, 46] gibi çalışmalarda da dağıtık ve bütünleşmiş web servislerinin bir arada çalışması esnasında güvenliğin arttırılması için, sistem katmanlı hale getirilmiştir. Böylece kullanıcılar erişmek istedikleri servislere ulaşana kadar farklı katmanlardaki birden çok güvenlik önlemini geçmek zorunda kalmaktadırlar. Katmanların kendi içindeki güvenliği ve birbirleri ile iletişimleri sırasındaki güvenlikleri sağlanarak sistem daha güvenilir hale gelmiştir. Bu çalışmada da sistem belirli bölümlere ayrılmış her bir bölüm üzerinde ihtiyaçlar doğrultusunda farklı güvenlik yöntemlerinin uygulanabildiği, kendi içinde ve birbirleri ile olan etkileşimleri sırasında güvenliğin sağlanmaya çalışıldığı bir mimari ortaya konulmuştur.

6.2. Performans Değerlendirmesi

WSGA modelinde sunucu ya da parametre sayılarının sabit olmasından dolayı performans değerlendirmesi, model üzerine eklenen güvenlik önlemlerinin etkisiyle değişecek olan istek-cevap sürelerine göre yapılmıştır. Öcelikle, Kimlik Belirleme Servisi, Yetkilendirme Servisi ve Hata Yönetimi Servisi'nden dönen cevap süreleri ölçülmüştür. Ardından yine aynı mimari üzerinde ancak bu sefer hiçbir güvenlik önleminin (şifreleme algoritması) kullanılmayarak, servislerden dönen cevap süreleri hesaplanmıştır. Elde edilen sonuçlar katmanlı yapıya sahip bir web servis uygulamasının performansı ile kıyaslanmıştır [47].

Oluşturulan WSGA modelinin çalışma performansını istek-cevap süresi açısından değerlendirmek için farklı ortamlarda bir dizi testler yapılmıştır. 3 farklı ortam oluşturulmuştur:

- Güvenlik için kullanılan bütün servislerin ve kullanıcının aynı makine üzerinde olduğu yerel bir ortam
- Güvenlik servislerinin ve kullanıcının farklı birer makine üzerinde bulunduğu bir ağ ortamı
- Güvenlik servislerinin ve kullanıcının farklı makineler üzerinde bulunduğu sanal ağ ortamı

İlk olarak yapılan testler güvenlik servislerinin ve kullanıcıların aynı makinede bulunduğu yerel ortam üzerinde yapılan testlerdir.

Bu test için kullanılan bilgisayarın özellikleri şu şekildedir:

- Intel Core 2 Duo CPU 2.40 GHz
- 4.00 GB RAM
- Windows 7 Professional İşletim Sistemi
- .Net Framework 3.5

Yapılan birçok deneme sonucunda, kullanıcının istekte bulunmasının ardından Kimlik Belirleme, Yetkilendirme ve Hata Yönetimi servislerinden kullanıcıya dönen cevap sürelerinden bazıları Tablo 6.2' de gösterilmiştir.

Tablo 6.2. Test 1- Sistem Performansı Ölçüm Sonuçları-1

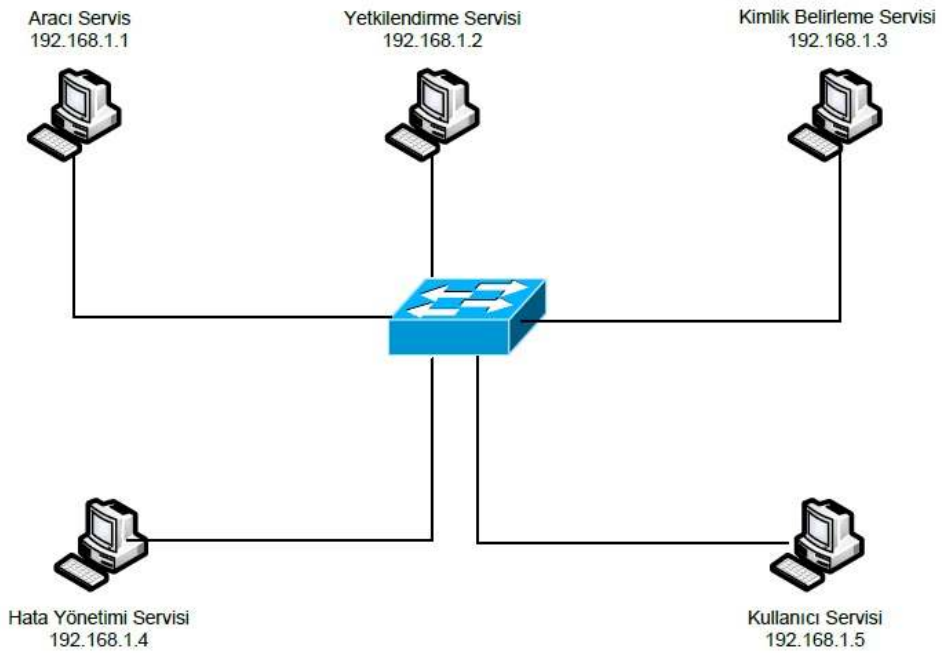
	Kimlik Belirleme Servisi (sn)	Yetkilendirme Servisi (sn)	Hata Yönetimi Servisi (sn)
Deneme 1	1.192	1.053	1.121
Deneme 2	1.105	1.096	1.130
Deneme 3	1.091	1.091	1.138
Deneme 4	1.009	0.991	1.078
Deneme 5	1.142	1.025	1.186

Yapılan denemeler sonucunda ortalama servislerden kullanıcıya dönen cevap süreleri Kimlik Belirleme Servisi'nden 0,987 sn., Yetkilendirme Servisi'nden 0,921 sn., Hata Yönetimi'nden ise 1,024 sn. olarak ölçülmüştür. Aynı denemelerin yine aynı mimari üzerinde hiçbir güvenlik standardı kullanılmadan elde edilen istek-cevap süreleri Tablo 6.3'de gösterildiği gibidir:

Tablo 6.3. Test 1- Sistem Performansı Ölçüm Sonuçları-2

	Kimlik Belirleme Servisi (sn)	Yetkilendirme Servisi (sn)	Hata Yönetimi Servisi (sn)
Deneme 1	0.078	0.053	0.097
Deneme 2	0.065	0.061	0.083
Deneme 3	0.024	0.021	0.052
Deneme 4	0.086	0.073	0.092
Deneme 5	0.046	0.042	0.074

Testlerin yapıldığı ikinci ortam, güvenlik servislerinin ve kullanıcının farklı birer makine üzerinde bulunduğu bir yerel alan ağı (LAN) ortamıdır. Bu çalışma için Bilgisayar Mühendisliği, Bilgisayar Ağları Laboratuarı'nda 5 makine içeren bir cluster (küme) oluşturulmuştur. Oluşturulan yerel alan ağı şekil 6.1 de gösterildiği gibidir. 3 makineye güvenlik için kullanılan servisler, Kimlik Belirleme, Yetkilendirme, Hata Yönetimi servisleri, 1 makineye de Aracı Servis kurulmuştur. Ardından IIS ayarları yapılarak, servisler bulunabilir olmaları için ağ üzerinde yayınlanmıştır. Son makine de kullanıcı olarak ayarlanmıştır.



Şekil 6.1. Oluşturulan Yerel Alan Ağı

Bu test için kullanılan bilgisayarların özellikleri şu şekildedir:

- Intel i5 İşlemci, CPU 2.40 GHz
- 4.00 GB RAM
- Windows 7 Professional İşletim Sistemi
- .Net Framework 3.5

Yapılan birçok deneme sonucunda, Kimlik Belirleme, Yetkilendirme ve Hata Yönetimi servislerinden kullanıcıya dönen cevap süreleri Tablo 6.4' de gösterilmiştir.

Tablo 6.4. Test 2- Sistem Performansı Ölçüm Sonuçları-1

	Kimlik Belirleme Servisi (sn)	Yetkilendirme Servisi (sn)	Hata Yönetimi Servisi (sn)
Deneme 1	3.022	3.011	3.087
Deneme 2	2.117	2.010	2.167
Deneme 3	2.720	2.703	2.751
Deneme 4	3.021	2.956	3.068
Deneme 5	2.745	2.704	2.774

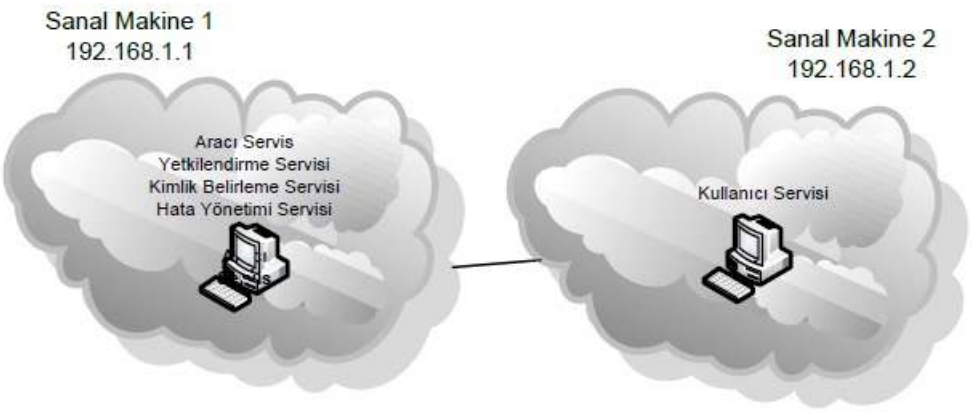
Yapılan denemeler sonucunda ortalama kullanıcıya dönen cevap süreleri Kimlik Belirleme Servisi'nden 2.865 sn., Yetkilendirme Servisi'nden 2.712 sn., Hata Yönetimi'nden ise 2.972 sn. olarak ölçülmüştür. Aynı denemelerin yine aynı mimari üzerinde hiçbir güvenlik standardı kullanılmadan elde edilen istek-cevap süreleri Tablo 6.5'de gösterildiği gibidir:

Tablo 6.5. Test 2- Sistem Performansı Ölçüm Sonuçları-2

	Kimlik Belirleme Servisi (sn)	Yetkilendirme Servisi (sn)	Hata Yönetimi Servisi (sn)
Deneme 1	2.011	1.987	2.097
Deneme 2	1.965	1.934	2.001
Deneme 3	2.020	2.001	2081
Deneme 4	2.013	1.834	2.018
Deneme 5	2.101	2.088	2.112

Testlerin yapıldığı son ortam ise, güvenlik servislerinin ve kullanıcının farklı makineler üzerinde bulunduğu sanal ağ ortamıdır. Sanal ağ ortamının seçilmesinin nedeni şöyle açıklanabilir. Sanallaştırma sayesinde mevcut donanım daha verimli kullanılmaktadır, dolayısıyla yazılım ve donanım bağımlılıklarını ortadan kaldırdığı ve yeni ürün ve servis geliştirme sürecinde maliyetleri çok aza indirdiği için son zamanlarda birçok kurum tarafından tercih edilmektedir [48].

Bu test için tek bir bilgisayarda VMWARE Workstation sanallaştırma programı üzerinde 2 farklı sanal makine kurulmuştur. Kurulan ağ şekil 6.2' de gösterilmiştir. Bir makinede güvenlik için kullanılan servisler bulunmakta, diğerinde ise kullanıcı bulunmaktadır.



Şekil 6.2. Oluşturulan Sanal Ortam

Bu test aşamaları için kullanılan sanal bilgisayarların özellikleri şu şekildedir:

Güvenlik servisleri için;

- Intel i5 CPU 2.40 GHz
- 2.00 GB RAM
- Windows 7 Professional İşletim Sistemi
- .Net Framework 3.5

Kullanıcı uygulaması için;

- Intel i5 CPU 2.40 GHz
- 1.00 GB RAM
- Windows 7 Professional İşletim Sistemi
- .Net Framework 3.5

Yapılan birçok deneme sonucunda, Kimlik Belirleme, Yetkilendirme ve Hata Yönetimi servislerinden kullanıcıya dönen cevap süreleri Tablo 6.6' da gösterilmiştir.

Tablo 6.6. Test 3- Sistem Performansı Ölçüm Sonuçları-1

	Kimlik Belirleme Servisi (sn)	Yetkilendirme Servisi (sn)	Hata Yönetimi Servisi (sn)
Deneme 1	2.017	2.003	2.048
Deneme 2	1.650	1.636	1.679
Deneme 3	1.405	1.397	1.442
Deneme 4	2.106	1.095	2.143
Deneme 5	1.230	1.219	1.275

Yapılan denemeler sonucunda kullanıcıya dönen ortalama cevap süreleri Kimlik Belirleme Servisi'nden 1.711 sn., Yetkilendirme Servisi'nden 1.612 sn., Hata Yönetimi'nden ise 1.927 sn. olarak ölçülmüştür. Aynı denemelerin yine aynı mimari üzerinde hiçbir güvenlik standardı kullanılmadan elde edilen istek-cevap süreleri Tablo 6.7'de gösterildiği gibidir:

Tablo 6.7. Test 3- Sistem Performansı Ölçüm Sonuçları-2

	Kimlik Belirleme Servisi (sn)	Yetkilendirme Servisi (sn)	Hata Yönetimi Servisi (sn)
Deneme 1	1.522	1.498	1.581
Deneme 2	1.501	1.481	1.534
Deneme 3	1.426	1.389	1.442
Deneme 4	1.619	1.601	1.713
Deneme 5	1.592	1.572	1.612

Yapılan testler sonunda oluşturulan modelin kullanıcıya dönen cevap sürelerinin, güvenlik standartları kullanılmadan aynı mimariden dönen cevap sürelerine göre daha uzun olduğu görülmüştür. Ayrıca [48] numaralı çalışmada ise sıralı sunucu sistemleri temelinde oluşturmuş bir web uygulamasında elde edilen cevap sürelerinin yine bu çalışmaya kıyasla daha kısa sürdüğü görülmektedir. İstek-cevap süresini etkileyen faktörlerin model üzerinde kullanılan şifreleme algoritması ve ağ üzerindeki iletim gecikmeleri olduğu gözlenmiştir. Model üzerinde kullanılacak her algoritma, dijital imza, sertifika işlemleri ve erişim kuralları servislerden dönecek cevap sürelerini etkilemektedir. Ancak bu tez çalışmasındaki amaç sürelerin kısalığından çok, güvenilir bir altyapı oluşturmaktır.

BÖLÜM 7. SONUÇLAR VE ÖNERİLER

Bu tez çalışmasıyla öncelikle web servisleri, web servis güvenliği, tasarlanan modelin yapısı ve çalışma mantığından bahsedilmiş, ardından uygulama anlatılmış, son olarak da yapılan test ve performans sonuçları gösterilmiştir.

Bu çalışmada, web servis güvenliğinde temel olan fonksiyonları kullanarak, güvenilir bir alt yapı oluşturulmuştur. Bu model oluşturulurken Facade tasarım deseninin çalışma mantığından yararlanılmıştır. Desen mantığındaki gibi, kullanıcı ve uygulama katmanları arasındaki ilişki tamamen kesilmiştir. Ayrıca uygulama katmanı 2 kısma bölünerek kullanıcı erişimi 2 aşamalı hale getirilmiştir. İlk kısım güvenlik fonksiyonları, 2. Kısım ise kullanıcıların erişmek istedikleri uygulama ve servislerin bulunduğu kısımdır. Güvenlik bölümünde bulunan fonksiyonlar, web servis güvenliğinde temel fonksiyonlardan bazıları olan kimlik belirleme, yeki belirleme ve hata yönetimidir. Bu fonksiyonların her biri ayrı birer web servisi şeklinde tasarlanmıştır ve Facade desenindeki gibi birbirleriyle olan iletişimleri kesilmiştir. İletişimlerini Aracı Servis adı verilen servis üzerinden sağlamaktadırlar. Aracı Servis desendeki Facade katmanıyla eş göreve sahip bir servis olarak düşünülmüştür. Aracı Servis bütün servislerden referans alarak çalışmaktadır ancak bu servisin sistemden çıkarılması sistemin bütününde herhangi bir aksaklığa sebep olmayacaktır. Çünkü diğer servislerin buna bir bağımlılığı bulunmamaktadır. Bu şekilde sistemde güvenliğin merkezileştirilmesi hedeflenmiştir.

Modelin tasarımı sayesinde üzerinde yapılacak ayarlamalar ile modelin yönetimi kolaylaşmış, güvenlik tek bir yere toplanmış, bağımsız ve birbirinden habersiz servisler oluşturarak güvenlik arttırılmaya çalışılmıştır. Bu mantık sayesinde sistemin geliştirilmesi ve uygulamalara entegresi kolaylaşmıştır. Farklı uygulamaların, platformların sisteme dahil edilmesi istendiğinde ya da mevcut servisler üzerinde güncelleme yapıldığında Aracı Servis üzerinde ufak bir ayarlama yeterli olacak, sistemin bütünü etkilenmeyecektir. Sistemin birbirine bağımlılığının az olması

sayesinde, model üzerindeki servislere tercihe göre istenilen güvenlik yöntemi uygulanabilir. Hatta her bir servis üzerine farklı güvenlik yöntemleri eklenebilir. Bu sistemin bütününü bozmayacaktır.

Sonuç olarak; servislerin birbiriyle iletişiminin kesilmesi, her bir servisin güvenlik ayarlarının kendi içinde yapılması, farklı platformların birlikte çalışabilmeleri ve yeni servislerin sisteme eklenmesindeki kolaylık sayesinde oluşturulan sistem daha güvenilir, genişletilebilir, esnek, kolay ayarlama yapılabilir, dağıtılabilir ve servis yönelimli bir yapıya sahip olmuştur. Ayrıca oluşturulan modelin, servis yönelimli mimariye geçiş aşamasında birçok kurumu düşündüren güvenlik sorunlarına bir alt yapı oluşturması hedeflenmiştir. İlaveten bu çalışmada, oluşturulan model üzerinde XML-Şifreleme algoritmalarından Rijndael simetrik algoritması kullanılarak servisler arasındaki mesajlaşma sırasında mesaj bütünlüğü ve güvenilirliği sağlanmıştır. Ancak bundan farklı olarak, her bir servis üzerine farklı güvenlik yöntemleri (XML-İmza, sertifika, asimetrik şifreleme) ve algoritmalar uygulanabilir. Sağlanan yazılımsal güvenliğin yanında donanımsal güvenlik elemanları da sisteme eklenerek daha güçlü bir altyapı oluşturulabilir.

KAYNAKLAR

[1] SATHIASEELAN, J.G.R., RABARA, S.A., MARTIN, J.R., Multi-Level Secure Architecture for Distributed Integrated Web Services, Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference, 9-11 July 2010 , pp. 180 – 184.

[2] PENG, Y., WU, Q., Secure Communication and Access Control for Web Services Container, Proceedings of the Fifth International Conference on Grid and Cooperative Computing (GCC'06), 2006.

[3] ZHANG, W., Integrated Security Framework for Secure Web Services , Third International Symposium on Intelligent Information Technology and Security Informatics, IEEE, 2010., pp. 178-183.

[4] CHALLAGULLA, S., Design and Development of an Authorization and Authentication Web Service, Master of Science in Technology, Division of Computing Studies, Arizona State University.2004.

[5] WAHLI, U., BURROUGHS, O., CLINE,O., GO, A., TUNGWS, L., Handbook for Websphere Application Server Version 6.1, SG24-7257-00, September 2006.

[6] KREGER, H., Web Services Conceptual Architecture (WSCA) 1.0, IBM Software Group, May 2001.

[7] SHNE, A., FUHRER, P., PASQUIER, J. S., Web Services Technologies: State of the Art Definitions, Standards, Case Study, Working Paper September 2009.

[8] Point, Tutorials. What are Web services. http://www.tutorialspoint.com/webservices/what_are_web_services.htm (Erişim Tarihi: 06.03.2012).

[9] RYMAN,A., http://www.ibm.com/developerworks/websphere/library/techarticles/0307_ryman/ryman.html, (Erişim Tarihi: 16.11.2011).

[10] ÇELİK, H., Web Servisleri İle Elektronik Ticaret Uygulaması Yüksek Lisans Tezi, Haliç Üniversitesi ,Ocak, 2008.

[11] KREGER, H., IBM Software Group, Web Services Conceptual Architecture (WSCA 1.0), May 2001,

[12] PAPAZOGLOU, M., Web Services: Principles and Technology, s.l.: Prentic Hall, 2008.

[13] MITRA, N., LAFON, Y., SOAP Version 1.2, 2007 , http://www.w3.org/TR/soap12-part0/, (Erişim Tarihi: 11.07.2011).

[14] THOMAS, E., Service-Oriented Architecture, Concepts, Technology, and Design, Prentice Hall Indiana, 2006. 0-13-185858-0.

[15] CHRISTENSEN, E., CURBERA,F, MEREDITH, G., WEERAWARANA, S.,Web Services Description Language (WSDL) 1.1,15 March 2001, http://www.w3.org/TR/wsdl, (Erişim Tarihi: 11.07.2011).

[16] CLEMENT, L., HATELY, A., RIEGEN, C., ROGERS, T., UDDI Universal Description, Discovery and Integration, 2004, http://uddi.org/pubs/uddi_v3.htm#_Toc85907968, (Erişim Tarihi: 11.07.2011).

[17] SWITHINBANK, P.,GIGANTE, F., PROCTOR, H., RATHORE, M., WIDJAJA, W., WebSphere and .Net Interoperability Using Web Services, 2005, ibm.com/redbooks SG24-6395-00.

[18] BOOTH, D., HAAS, H., MCCABE, F., NEWCOMER, E.,CHAMPION, M., FERRIS, C.,ORCHARD, D., Web Services Architecture, 2004, http://www.w3.org/TR/ws-arch/ , (Erişim Tarihi: 17.12.2011).

[19] WARSCHOFSKY,R., MENZEL, M., MEINEL, C., Transformation and Aggregation of Web Service Security Requirements, 2010 Eighth IEEE European Conference on Web Services, 1-3 Dec. 2010, pp. 43 – 50.

[20] ONO, K., NAKAMURA, Y., SATOH, F., TATEISHI, T., Verifying the Consistency of Security Policies by Abstracting into Security Types, 2007 IEEE International Conference on Web Services, 9-13 July 2007, pp. 497 – 504.

[21] GERIC, S., Security of Web Services based Service-oriented Architectures, MIPRO 2010, Opatija, 2010., pp.1250-1255.

[22] GERIC, S., HUTINSKI, Z., Service Oriented Security, MIPRO 30th International Convention, Proceedings of Information System Security, Opatija, 2007., pp. 125 - 132.

[23] OASIS, http://docs.oasis-open.org

[24] NORDBOTTEN, N.A., XML and Web Service Security Standards, IEEE Communication Survey&Tutorials, vol. 11, pp. 4-21, 2009.

[25] MIRTALEBI, A., KHAYYAMBASHI, M.R., Enhancing Security of Web Service Against WSDL Threats, Emergency Management and Management Sciences (ICEMMS), 2011 2nd IEEE International Conference on, 8-10 Aug. 2011, pp. 920 - 923 .

[26] BARTEL, M., BOYER, J., FOX, B., LAMACCHIA, B., SIMON, E. (2008) XML Signature Syntax and Processing. http://www.w3.org/TR/xmldsig-core/, (Erişim Tarihi: 17.12.2011).

[27] IMAMURA, T., DILLAWAY, B., SIMON, E., XML Encryption Syntax and Processing, 2002, http://www.w3.org/TR/xmlenc-core/, (Erişim Tarihi: 10.02.2012).

[28] HALLAM-BAKER, P., MYSORE, S.H., XML Key Management Specification (XKMS 2.0), 2005, http://www.w3.org/TR/xkms2/, (Erişim Tarihi: 12.12.2011).

[29] PARDUCCI, B., LOCKHART, H., LEVINSON, R., OASIS eXtensible Access Control Markup Language (XACML), 2011, http://www.oasis-open.org/committees/xacml/, (Erişim Tarihi: 17.02.2012).

[30] SUN, L., LI, Y., XML and Web Services Security, Computer Supported Cooperative Work in Design, 2008. CSCWD 2008. 12th International Conference, 16-18 April 2008, pp. 765-770.

[31] XML ENCRPYTION, http://www.xyzws.com/scdjws/SGS41/3, (Erişim Tarihi: 07.02.2012).

[32] HU, V.C., MARTIN, E., HWANG, J.H., XIE, T., Conformance Checking of Access Control Policies Specified in XACML, COMPSAC '07 Proceedings of the 31st Annual International Computer Software and Applications Conference ,2007 , ISBN:0-7695-2870-8.

[33] SECURITY ASSERTİON MARKUP LANGUAGE (SAML) V2.0 OASIS Standard, 2005, http://docs.oasis-open.org/security/saml/v2.0/ (Erişim Tarihi: 20.02.2012).

[34] SINGHAL, A., WINOGRAD, T., SCARFONE, K., Guide to Secure Web Services Recommendations of the National Institute of Standards and Technology , 2007.

[35] ERL, T., An Overview of the WS-Security Framework, http://www.whatissoa.com/soaspecs/ws-security.php, (Erişim Tarihi: 10.02.2012).

[36] WS-POLICY, http://en.wikipedia.org/wiki/WS-Policy, (Erişim Tarihi: 20.01.2012).

[37] WS-TRUST, http://en.wikipedia.org/wiki/WS-Trust, (Erişim Tarihi: 20.01.2012).

[38] WS-SECURECONVERSATION, http://en.wikipedia.org/wiki/WS-SecureConversation, (Erişim Tarihi: 20.01.2012).

[39] WS-FEDERATION , http://en.wikipedia.org/wiki/WS-Federation, (Erişim Tarihi: 20.01.2012).

[40] WS-RELIABILITY, http://en.wikipedia.org/wiki/WS-Reliability, (Erişim Tarihi: 20.01.2012).

[41] GAMMA, E., HELM, R., JOHNSON,R., VLISSIDES, J., Design Patterns: Elements of Reusable Object Oriented Software. Addison-Wesley Professional, 1994.

[42] DAEMEN,J., RIJMEN, V., The Design of Rijndael: AES - The Advanced Encryption Standard, Springer, 2002. ISBN 3-540-42580-2.

[43] SCHWARZ, J., HARTMAN, B., NADALIN, A., Security Challenges, Threats and Countermeasures Version 1.0, The Web Services-Interoperability Organization, 2005.

[44] SIDHARTH, N., LIU, J., IAPF: A Framework for Enhancing Web Services Security, International Computer Software and Application Conference, COMPSAC, IEEE, 2007.

[45] NARGES, S., MOHSENZADEH, M., SEYYEDI, M. A., QORANI, S. H., A New Security Framework Against Web Services' XML Attacks in SOA, International Conference on Next Generation Web Services Practices, IEEE, 2011.

[46] ZHAO, F., PENG, X., ZHAO,W., Multi Tier Security Feature Modeling For Service-Oriented Application Integration, International Conference On Computer And Information Science, IEEE, 2009.

[47] ANAGÜN, Y., Üç-Sıralı İstemci-Sunucu Mimarisinde Web Servisleri Kullanımı Ve Uygulaması, Yüksek Lisans Tezi, Dumlupınar Üniversitesi, Şubat - 2007

[48] BALTA, M., Sanal Ortam Üzerinde Oluşturulan Örnek Bir Kurumsal Ağ Topolojisinin Snmpv3 İle Topoloji Keşfi Uygulaması, Yüksek Lisans Tezi, Sakarya Üniversitesi Fen Bilimleri Enstitüsü, Ocak 2012.

EKLER

EK -A

Aracı Servis Uygulaması WSDL Belgesi (MediatorService.asmx?WSDL)


```
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:tns="http://tempuri.org/" xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="http://tempuri.org/">
<wsdl:types>
<s:schema elementFormDefault="qualified" targetNamespace="http://tempuri.org/">
<s:element name="GetMessages">
<s:complexType>
<s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="values" type="s:string"/>
</s:sequence>
</s:complexType>
</s:element>
<s:element name="GetMessagesResponse">
<s:complexType>
<s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="GetMessagesResult"
type="s:string"/>
</s:sequence>
</s:complexType>
</s:element>
<s:element name="GetRole">
<s:complexType>
<s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="deger" type="s:string"/>
</s:sequence>
</s:complexType>
</s:element>
<s:element name="GetRoleResponse">
<s:complexType>
```

```
<s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="GetRoleResult"
type="tns:ArrayOfString"/>
</s:sequence>
</s:complexType>
</s:element>
<s:complexType name="ArrayOfString">
<s:sequence>
<s:element minOccurs="0" maxOccurs="unbounded" name="string" nillable="true"
type="s:string"/>
</s:sequence>
</s:complexType>
<s:element name="Log">
<s:complexType>
<s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="value" type="s:string"/>
</s:sequence>
</s:complexType>
</s:element>
<s:element name="LogResponse">
<s:complexType>
<s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="LogResult" type="s:string"/>
</s:sequence>
</s:complexType>
</s:element>
<s:element name="decryption">
<s:complexType>
<s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="value" type="s:string"/>
</s:sequence>
</s:complexType>
</s:element>
<s:element name="decryptionResponse">
<s:complexType>
<s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="decryptionResult"
type="s:string"/>
</s:sequence>
</s:complexType>
</s:element>
</s:schema>
</wsdl:types>
<wsdl:message name="GetMessagesSoapIn">
<wsdl:part name="parameters" element="tns:GetMessages"/>
</wsdl:message>
<wsdl:message name="GetMessagesSoapOut">
<wsdl:part name="parameters" element="tns:GetMessagesResponse"/>
</wsdl:message>
```

```
<wsdl:message name="GetRoleSoapIn">
<wsdl:part name="parameters" element="tns:GetRole"/>
</wsdl:message>
<wsdl:message name="GetRoleSoapOut">
<wsdl:part name="parameters" element="tns:GetRoleResponse"/>
</wsdl:message>
<wsdl:message name="LogSoapIn">
<wsdl:part name="parameters" element="tns:Log"/>
</wsdl:message>
<wsdl:message name="LogSoapOut">
<wsdl:part name="parameters" element="tns:LogResponse"/>
</wsdl:message>
<wsdl:message name="decryptionSoapIn">
<wsdl:part name="parameters" element="tns:decryption"/>
</wsdl:message>
<wsdl:message name="decryptionSoapOut">
<wsdl:part name="parameters" element="tns:decryptionResponse"/>
</wsdl:message>
<wsdl:portType name="MediatorServiceSoap">
<wsdl:operation name="GetMessages">
<wsdl:input message="tns:GetMessagesSoapIn"/>
<wsdl:output message="tns:GetMessagesSoapOut"/>
</wsdl:operation>
<wsdl:operation name="GetRole">
<wsdl:input message="tns:GetRoleSoapIn"/>
<wsdl:output message="tns:GetRoleSoapOut"/>
</wsdl:operation>
<wsdl:operation name="Log">
<wsdl:input message="tns:LogSoapIn"/>
<wsdl:output message="tns:LogSoapOut"/>
</wsdl:operation>
<wsdl:operation name="decryption">
<wsdl:input message="tns:decryptionSoapIn"/>
<wsdl:output message="tns:decryptionSoapOut"/>
</wsdl:operation>
</wsdl:portType>
<wsdl:binding name="MediatorServiceSoap" type="tns:MediatorServiceSoap">
<soap:binding transport="http://schemas.xmlsoap.org/soap/http"/>
<wsdl:operation name="GetMessages">
<soap:operation soapAction="http://tempuri.org/GetMessages" style="document"/>
<wsdl:input>
<soap:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap:body use="literal"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="GetRole">
<soap:operation soapAction="http://tempuri.org/GetRole" style="document"/>
```

```
<wsdl:input>
<soap:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap:body use="literal"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="Log">
<soap:operation soapAction="http://tempuri.org/Log" style="document"/>
<wsdl:input>
<soap:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap:body use="literal"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="decryption">
<soap:operation soapAction="http://tempuri.org/decryption" style="document"/>
<wsdl:input>
<soap:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap:body use="literal"/>
</wsdl:output>
</wsdl:operation>
</wsdl:binding>
<wsdl:binding name="MediatorServiceSoap12" type="tns:MediatorServiceSoap">
<soap12:binding transport="http://schemas.xmlsoap.org/soap/http"/>
<wsdl:operation name="GetMessages">
<soap12:operation soapAction="http://tempuri.org/GetMessages"
style="document"/>
<wsdl:input>
<soap12:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap12:body use="literal"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="GetRole">
<soap12:operation soapAction="http://tempuri.org/GetRole" style="document"/>
<wsdl:input>
<soap12:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap12:body use="literal"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="Log">
<soap12:operation soapAction="http://tempuri.org/Log" style="document"/>
```

<wsdl:input>
<soap12:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap12:body use="literal"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="decryption">
<soap12:operation soapAction="http://tempuri.org/decryption" style="document"/>
<wsdl:input>
<soap12:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap12:body use="literal"/>
</wsdl:output>
</wsdl:operation>
</wsdl:binding>
<wsdl:service name="MediatorService">
<wsdl:port name="MediatorServiceSoap" binding="tns:MediatorServiceSoap">
<soap:address location="http://localhost:50109/MediatorService.asmx"/>
</wsdl:port>
<wsdl:port name="MediatorServiceSoap12" binding="tns:MediatorServiceSoap12">
<soap12:address location="http://localhost:50109/MediatorService.asmx"/>
</wsdl:port>
</wsdl:service>
</wsdl:definitions>

EK- B

Kimlik Belirleme Servisi WSDL Belgesi (AuthenticationService.asmx?WSDL)


```
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:tns="http://tempuri.org/" xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="http://tempuri.org/">
<wsdl:types>
<s:schema elementFormDefault="qualified" targetNamespace="http://tempuri.org/">
<s:element name="Control">
<s:complexType>
<s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="values" type="s:string"/>
</s:sequence>
</s:complexType>
</s:element>
<s:element name="ControlResponse">
<s:complexType>
<s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="ControlResult" type="s:string"/>
</s:sequence>
</s:complexType>
</s:element>
<s:element name="encString">
<s:complexType>
<s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="input" type="s:string"/>
</s:sequence>
</s:complexType>
</s:element>
<s:element name="encStringResponse">
<s:complexType>
<s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="encStringResult"
type="s:string"/>
</s:sequence>
</s:complexType>
</s:element>
<s:element name="decString">
<s:complexType>
<s:sequence>
```

```
<s:element minOccurs="0" maxOccurs="1" name="input" type="s:string"/>
</s:sequence>
</s:complexType>
</s:element>
<s:element name="decStringResponse">
<s:complexType>
<s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="decStringResult"
type="s:string"/>
</s:sequence>
</s:complexType>
</s:element>
</s:schema>
</wsdl:types>
<wsdl:message name="ControlSoapIn">
<wsdl:part name="parameters" element="tns:Control"/>
</wsdl:message>
<wsdl:message name="ControlSoapOut">
<wsdl:part name="parameters" element="tns:ControlResponse"/>
</wsdl:message>
<wsdl:message name="encStringSoapIn">
<wsdl:part name="parameters" element="tns:encString"/>
</wsdl:message>
<wsdl:message name="encStringSoapOut">
<wsdl:part name="parameters" element="tns:encStringResponse"/>
</wsdl:message>
<wsdl:message name="decStringSoapIn">
<wsdl:part name="parameters" element="tns:decString"/>
</wsdl:message>
<wsdl:message name="decStringSoapOut">
<wsdl:part name="parameters" element="tns:decStringResponse"/>
</wsdl:message>
<wsdl:portType name="AuthenticationServiceSoap">
<wsdl:operation name="Control">
<wsdl:input message="tns:ControlSoapIn"/>
<wsdl:output message="tns:ControlSoapOut"/>
</wsdl:operation>
<wsdl:operation name="encString">
<wsdl:input message="tns:encStringSoapIn"/>
<wsdl:output message="tns:encStringSoapOut"/>
</wsdl:operation>
<wsdl:operation name="decString">
<wsdl:input message="tns:decStringSoapIn"/>
<wsdl:output message="tns:decStringSoapOut"/>
</wsdl:operation>
</wsdl:portType>
<wsdl:binding name="AuthenticationServiceSoap"
type="tns:AuthenticationServiceSoap">
<soap:binding transport="http://schemas.xmlsoap.org/soap/http"/>
```

<wsdl:operation name="Control">
<soap:operation soapAction="http://tempuri.org/Control" style="document"/>
<wsdl:input>
<soap:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap:body use="literal"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="encString">
<soap:operation soapAction="http://tempuri.org/encString" style="document"/>
<wsdl:input>
<soap:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap:body use="literal"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="decString">
<soap:operation soapAction="http://tempuri.org/decString" style="document"/>
<wsdl:input>
<soap:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap:body use="literal"/>
</wsdl:output>
</wsdl:operation>
</wsdl:binding>
<wsdl:binding name="AuthenticationServiceSoap12"
type="tns:AuthenticationServiceSoap">
<soap12:binding transport="http://schemas.xmlsoap.org/soap/http"/>
<wsdl:operation name="Control">
<soap12:operation soapAction="http://tempuri.org/Control" style="document"/>
<wsdl:input>
<soap12:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap12:body use="literal"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="encString">
<soap12:operation soapAction="http://tempuri.org/encString" style="document"/>
<wsdl:input>
<soap12:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap12:body use="literal"/>
</wsdl:output>
</wsdl:operation>

<wsdl:operation name="decString">
<soap12:operation soapAction="http://tempuri.org/decString" style="document"/>
<wsdl:input>
<soap12:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap12:body use="literal"/>
</wsdl:output>
</wsdl:operation>
</wsdl:binding>
<wsdl:service name="AuthenticationService">
<wsdl:port name="AuthenticationServiceSoap"
binding="tns:AuthenticationServiceSoap">
<soap:address location="http://localhost:50037/AuthenticationService.asmx"/>
</wsdl:port>
<wsdl:port name="AuthenticationServiceSoap12"
binding="tns:AuthenticationServiceSoap12">
<soap12:address location="http://localhost:50037/AuthenticationService.asmx"/>
</wsdl:port>
</wsdl:service>
</wsdl:definitions>

EK- C

Yetkilendirme Servisi WSDL Belgesi (AuthorizationService.asmx?WSDL)


```
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:tns="http://tempuri.org/" xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="http://tempuri.org/">
<wsdl:types>
<s:schema elementFormDefault="qualified" targetNamespace="http://tempuri.org/">
<s:element name="answer">
<s:complexType>
<s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="values" type="s:string"/>
</s:sequence>
</s:complexType>
</s:element>
<s:element name="answerResponse">
<s:complexType>
<s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="answerResult"
type="tns:ArrayOfString"/>
</s:sequence>
</s:complexType>
</s:element>
<s:complexType name="ArrayOfString">
<s:sequence>
<s:element minOccurs="0" maxOccurs="unbounded" name="string" nillable="true"
type="s:string"/>
</s:sequence>
</s:complexType>
</s:schema>
</wsdl:types>
<wsdl:message name="answerSoapIn">
<wsdl:part name="parameters" element="tns:answer"/>
</wsdl:message>
<wsdl:message name="answerSoapOut">
<wsdl:part name="parameters" element="tns:answerResponse"/>
</wsdl:message>
<wsdl:portType name="AuthorizationServiceSoap">
<wsdl:operation name="answer">
<wsdl:input message="tns:answerSoapIn"/>
```

<wsdl:output message="tns:answerSoapOut"/>
</wsdl:operation>
</wsdl:portType>
<wsdl:binding name="AuthorizationServiceSoap"
type="tns:AuthorizationServiceSoap">
<soap:binding transport="http://schemas.xmlsoap.org/soap/http"/>
<wsdl:operation name="answer">
<soap:operation soapAction="http://tempuri.org/answer" style="document"/>
<wsdl:input>
<soap:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap:body use="literal"/>
</wsdl:output>
</wsdl:operation>
</wsdl:binding>
<wsdl:binding name="AuthorizationServiceSoap12"
type="tns:AuthorizationServiceSoap">
<soap12:binding transport="http://schemas.xmlsoap.org/soap/http"/>
<wsdl:operation name="answer">
<soap12:operation soapAction="http://tempuri.org/answer" style="document"/>
<wsdl:input>
<soap12:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap12:body use="literal"/>
</wsdl:output>
</wsdl:operation>
</wsdl:binding>
<wsdl:service name="AuthorizationService">
<wsdl:port name="AuthorizationServiceSoap"
binding="tns:AuthorizationServiceSoap">
<soap:address location="http://localhost:50067/AuthorizationService.asmx"/>
</wsdl:port>
<wsdl:port name="AuthorizationServiceSoap12"
binding="tns:AuthorizationServiceSoap12">
<soap12:address location="http://localhost:50067/AuthorizationService.asmx"/>
</wsdl:port>
</wsdl:service>
</wsdl:definitions>

EK-D

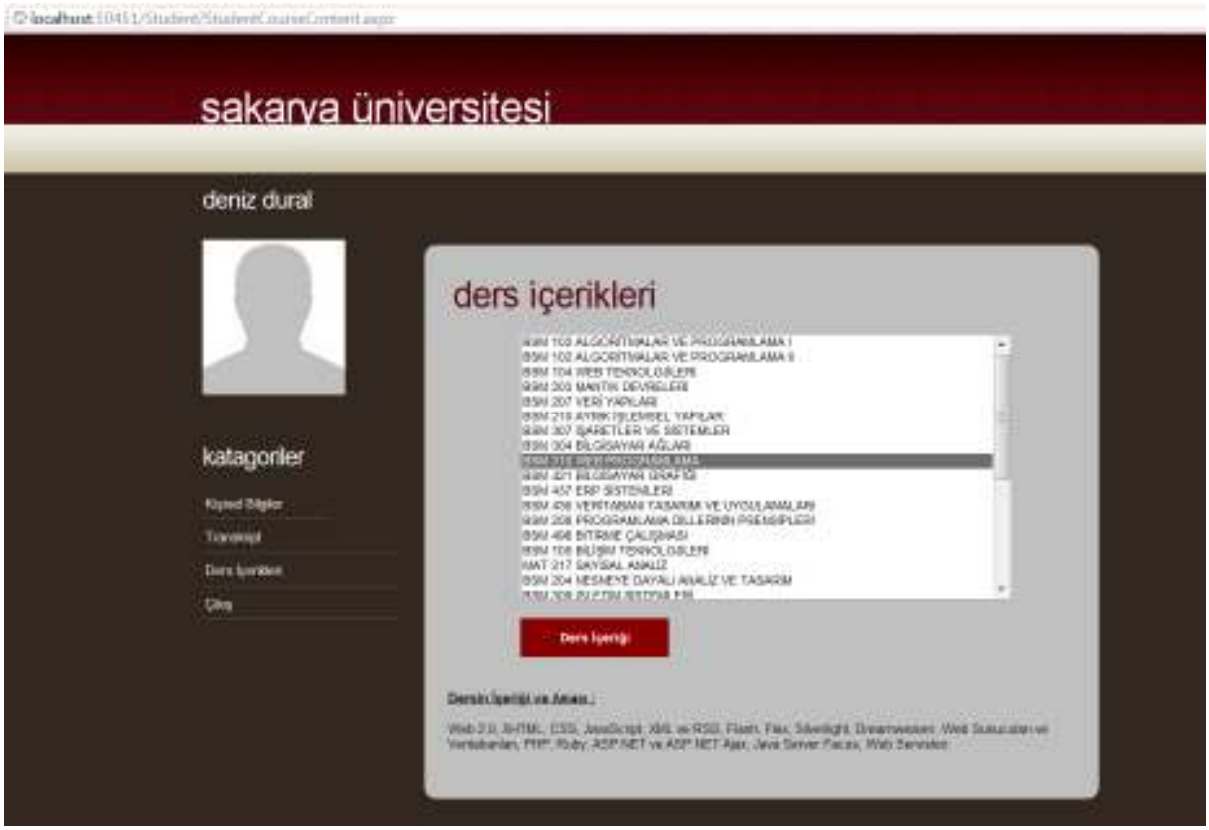
Hata Yönetimi Servisi WSDL Belgesi (LoggingService.asmx?WSDL)

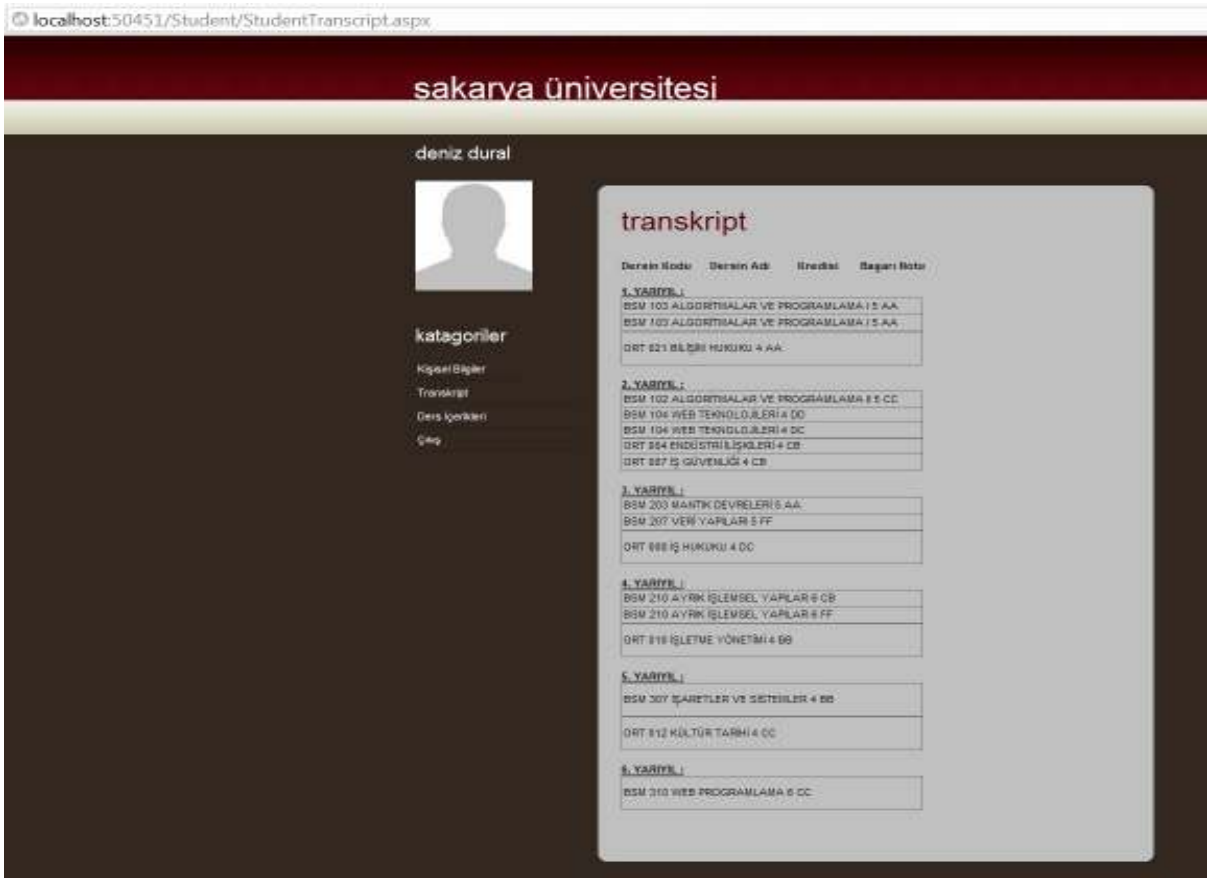

```
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:tns="http://tempuri.org/" xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="http://tempuri.org/">
<wsdl:types>
<s:schema elementFormDefault="qualified" targetNamespace="http://tempuri.org/">
<s:element name="log">
<s:complexType>
<s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="values" type="s:string"/>
</s:sequence>
</s:complexType>
</s:element>
<s:element name="logResponse">
<s:complexType>
<s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="logResult" type="s:string"/>
</s:sequence>
</s:complexType>
</s:element>
</s:schema>
</wsdl:types>
<wsdl:message name="logSoapIn">
<wsdl:part name="parameters" element="tns:log"/>
</wsdl:message>
<wsdl:message name="logSoapOut">
<wsdl:part name="parameters" element="tns:logResponse"/>
</wsdl:message>
<wsdl:portType name="Service1Soap">
<wsdl:operation name="log">
<wsdl:input message="tns:logSoapIn"/>
<wsdl:output message="tns:logSoapOut"/>
</wsdl:operation>
</wsdl:portType>
<wsdl:binding name="Service1Soap" type="tns:Service1Soap">
<soap:binding transport="http://schemas.xmlsoap.org/soap/http"/>
<wsdl:operation name="log">
<soap:operation soapAction="http://tempuri.org/log" style="document"/>
```

<wsdl:input>
<soap:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap:body use="literal"/>
</wsdl:output>
</wsdl:operation>
</wsdl:binding>
<wsdl:binding name="Service1Soap12" type="tns:Service1Soap">
<soap12:binding transport="http://schemas.xmlsoap.org/soap/http"/>
<wsdl:operation name="log">
<soap12:operation soapAction="http://tempuri.org/log" style="document"/>
<wsdl:input>
<soap12:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap12:body use="literal"/>
</wsdl:output>
</wsdl:operation>
</wsdl:binding>
<wsdl:service name="Service1">
<wsdl:port name="Service1Soap" binding="tns:Service1Soap">
<soap:address location="http://localhost:50327/Service1.asmx"/>
</wsdl:port>
<wsdl:port name="Service1Soap12" binding="tns:Service1Soap12">
<soap12:address location="http://localhost:50327/Service1.asmx"/>
</wsdl:port>
</wsdl:service>
</wsdl:definitions>

EK-E

Kullanıcı Uygulaması Arayüzü


sakarya üniversitesi
deniz dural
katagoriler
Kişisel Bilgiler
Transkript
Ders İçerikleri
Çıkış
ders içerikleri
Ders İçeriği


localhost:50451/Student/StudentTranscript.aspx
sakarya üniversitesi
deniz dural
katagoriler
Kişisel Bilgiler
Transkript
Ders İçerikleri
Çıkış
transkript
Dersin Kodu
Dersin Adı
Kredisi
Başarı Notu
1. YARIYIL :
BSM 103 ALGORITMALAR VE PROGRAMLAMA I 5 AA
BSM 103 ALGORITMALAR VE PROGRAMLAMA I 5 AA
ORT 821 BİLİŞİM HUKUKU 4 AA
2. YARIYIL :
BSM 102 ALGORITMALAR VE PROGRAMLAMA II 5 CC
BSM 104 WEB TEKNOLOJİLERİ II 4 DD
BSM 104 WEB TEKNOLOJİLERİ II 4 DC
ORT 864 ENDÜSTRİ İLİŞKİLERİ 4 CB
ORT 887 İŞ GÜVENLİĞİ 4 CB
3. YARIYIL :
BSM 203 MANTIK DEVRELERİ 5 AA
BSM 207 VERİ YAPILARI 5 FF
ORT 888 İŞ HUKUKU 4 DC
4. YARIYIL :
BSM 210 AYRIK İŞLEMSEL YAPILAR II 6 CB
BSM 210 AYRIK İŞLEMSEL YAPILAR II 6 FF
ORT 819 İŞLETME YÖNETİMİ 4 BB
5. YARIYIL :
BSM 307 İŞARETLER VE SİSTEMLER 4 BB
ORT 812 KÜLTÜR TARİHİ 4 CC
6. YARIYIL :
BSM 310 WEB PROGRAMLAMA II 6 CC

ÖZGEÇMİŞ

Deniz DURAL, 14.05.1987'de Sakarya'da doğdu. İlk, orta ve lise eğitimini Adapazarı'nda tamamladı. 2005 yılında Sakarya Anadolu Lisesi'nden mezun oldu. 2005 yılında başladığı Eskişehir Osmangazi Üniversitesi Bilgisayar Mühendisliği bölümünü 2010 yılında bitirdi. 2010 yılında Sakarya Üniversitesi Bilgisayar ve Bilişim Mühendisliği bölümünde yüksek lisansa başladı. Aralık 2010 tarihinden itibaren Sakarya Üniversitesi Bilgisayar ve Bilişim Mühendisliği'nde Araştırma Görevlisi olarak çalışmaktadır.