**T.C.**
**YILDIZ TEKNİK ÜNİVERSİTESİ**
**FEN BİLİMLERİ ENSTİTÜSÜ**

# SERVİS ODAKLI MİMARİDE KULLANILAN ŞİFRELEME YÖNTEMLERİNİN DEĞERLENDİRİLMESİ

**MİRSAT YEŞİLTEPE**

**YÜKSEK LİSANS TEZİ**
**BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI**
**BİLGİSAYAR MÜHENDİSLİĞİ PROGRAMI**

**DANIŞMAN**
**ÖĞR. GRV. DR. ÖMER ÖZGÜR BOZKURT**

**İSTANBUL, 2015**

**T.C.**
**YILDIZ TEKNİK ÜNİVERSİTESİ**
**FEN BİLİMLERİ ENSTİTÜSÜ**

# SERVİS ODAKLI MİMARİDE KULLANILAN ŞİFRELEME YÖNTEMLERİNİN DEĞERLENDİRİLMESİ

Mirsat Yeşiltepe tarafından hazırlanan tez çalışması 3.6.2015 tarihinde aşağıdaki jüri tarafından Yıldız Teknik Üniversitesi Fen Bilimleri Bilgisayar Mühendisliği Anabilim Dalı'nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

**Tez Danışmanı**
Öğr. Grv. Dr. Ö. Özgür Bozkurt
Yıldız Teknik Üniversitesi

**Jüri Üyeleri**
Öğr. Grv. Dr. Ö. Özgür BOZKURT
Yıldız Teknik Üniversitesi _____

Yrd. Doç. Dr. Ahmet Tevfik İNAN
Yıldız Teknik Üniversitesi _____

Yrd. Doç. Dr. Tolga ENSARİ
İstanbul Üniversitesi _____

# ÖNSÖZ

Büyük sistemlerde çok fazla servisin olduğu bilinmektedir. Örneğin Amazon'da 150.000 ile 450.000 arası sunucu istemcilerin bağlanması için genel buluta bağlanmaktadır [1]. Google ve Microsoft'un da yaklaşık bir milyon sunucusunun altyapılarında olduğu ve bunun Gmail, Google Arama veya Bing Arama işlerinde kullanıldığı bilinmektedir [2]. Bu yüzden bulut servisler çok geniş ölçüden inşa edilebilir ve Amazon, Google ve Microsoft gibi bulutu kullanması zorunlu uygulamalarda gerçekten büyüleyici işler yapılmasına olanak sağlamaktır. Burada amaç bu büyük ölçeklerin nasıl inşa edildikleri ve bulut servislerin mobil cihazlar ile nasıl iletişim kurduklarının ortaya konmasıdır.

Günümüzde web bâzlı servislerin nasıl inşa edildiklerine dair çok fazla kaynak bulunmaktadır. Buluttaki servisler ve bunların Hiper Metin Transfer Protokolünü (HTTP) kullanarak mobil cihazlarla nasıl etkileşim içinde olduğu bahsedilecek esas kısımdır. Java Spring Çatısı'nın HTTP üzerinden istekleri nasıl kabul ettiği ve bunları nasıl işlediği, bulutta özel bir veri depolayıcıda aynı kuyruk kapasitesinin olmadığında nasıl saklandığı, bulut mimarisindeki değişiklikler üzerinde durulan diğer konulardır.

Esas amaç tekil(bireysel) veya çoklu mobil cihazların buluta veri göndermesi, bunların işlenmesi, kombine edilmesi ve bireysel cihazla kullanılmayacak şekilde saklanmasıdır.

Mobil cihazlar çok gelişmesine rağmen, tekil mobil cihazların bazen sayıları yüz bini aşan sunuculara bağlanırken zorlanmaları doğaldır. Bu sayı sadece on sunucu olsa bile bu servisler normal tekil cihazlardan daha çok güçlüdür. Bir sorun ise bu cihazların buluta bağlanmaları için ayrık uygulamaların nasıl inşa edildiği bunun güvenlik endişesi gibi ölçülere uygun olmasıdır.

Mayıs, 2015

Mirsat YEŞİLTEPE

# İÇİNDEKİLER

Sayfa

## KISALTMA LİSTESİ

| | |
|---|---|
| ACL | Erişim Kontrol Listesi |
| ACS | Azure Erişim Kontrol Servisi |
| AON | Uygulama Oda Ağı |
| ASCII | Bilgi Değişimi İçin Amerikan Standart Kodlama Sistemi |
| CA | Sertifika Yetkilisi |
| CPU(MIB) | Merkezi İşlem Birimi |
| CRM | Müşteri İlişkileri Yönetimi |
| CRUD | Programlamada Oluşturma, Okuma, Güncelleme ve Silme |
| DAO | Veri Erişim Nesnesi |
| DoD | Savunma Bakanlığı |
| DRM | Dijital Haklar Yönetimi |
| EDI | Elektronik Veri Geçidi |
| EL | İfade Dili |
| E-Tag | Varlık Etiketi |
| FTP | Dosya Aktarım İletişim Kuralı |
| GCM | Google Bulut Mesajlaşma |
| HATEOAS | Uygulama Durum Altyapısı Gibi Hiperortam |
| HTML | Zengin Metin İşaretleme Dili |
| HTML5 | HTML İşaretleme Standardının Beşinci Sürümü |
| HTTP | Hiper Metin Transfer Protokolü |
| HTTPS | Güvenli Hiper Metin Transfer İletişim Kuralı |
| IaaS | Altyapı Olarak Servis |
| ID | Kimlik Kartı |
| IETF | İnternet Mühendisliği Görev Gücü |
| IIS | İnternet Bilgi Servisleri |
| IM | Anında Mesajlaşma |
| IP | İnternet Protokolü |
| ISP | İnternet Servis Sağlayıcı |
| JDBC | Java Veri Bağlantısı |
| JDBC | Java Veri Tabanı Bağlanabilirliği |
| JGSS | Java Jenerik Güvenlik Servisi |
| JNI | Java Asıl Arabirimi |
| JPA | Java Kalıcı API |
| JPA | Java Kullanıcı Ara yüzü |

| | |
|---|---|
| JPEG | Birleşik Fotoğraf Uzmanları Kurumu |
| JPQL | Java Kalıcı Sorgulama Dili |
| JSON | JavaScript Nesne Gösterimi |
| JSR | Java Toplu İşleme |
| KB | Kilo Bayt |
| LDAP | Basit Dizin Erişim Protokolü |
| M4V | Video Ipod'un Standart Video Formatı |
| MIM | Ortam Adam Saldırısı |
| MIME | Çok amaçlı İnternet Posta Eklentileri |
| MPNS | Microsoft İtme Notifikasyonu Servisi |
| MVC | Model – Görünüm – Denetleyicisi |
| MVVM | Model Görünüm Model-Görünüm |
| NoSQL | SQL Ara yüzü Olmayan Hafif Bir Açık Kaynak İlişkisel Veri tabanı |
| OAAM | Adapte Erişim Yöneticisi Oracle |
| OAuth | Açık Kimlik Doğrulama |
| OData | Genel Veri Protokolü |
| ORM | Nesne İlişkisel Eşleme |
| OSI | Açık Sistemler Bağlantısı |
| PaaS | Platform Olarak Servis |
| PKI | Açık Anahtar Altyapısı |
| REST | Temsili Durum Transferi |
| RMI | Uzaktan Yöntem Çağırma |
| SaaS | Yazılım Olarak Servis |
| SAML | Güvenlikli İddiası İşaretleme Dili |
| SAP | Veri İşlemede Sistemler, Uygulamalar ve Ürünler |
| SHA | Güvenli Karışık Şifreleme |
| SLA | Servis Düzeyi Anlaşma |
| SMTP | Basit Posta Aktarım Protokolü |
| SQL | Yapılandırılmış Sorgu Dili |
| SSL | Güvenli Giriş Katmanı |
| SSO | Tek Oturum Açma |
| SWT | Basit İnternet Belirteci |
| TCP | Aktarım Kontrol Protokolü |
| TLS | Transfer Seviye Güvenlik |
| UDP | Kullanıcı Veri Bloğu İletişim Kuralları |
| UI | Kullanıcı Ara yüzü |
| URI | Tek Düzen Kaynak Tanımlayıcısı |
| URL | Tek Düzen Kaynak Bulucu |
| URN | Tek Düzen Kaynak İsmi |
| VM | Sanal Makine |
| WAR | Web Uygulama Arşivi |
| WAS | Windows Etkileştirme Hizmeti |
| WCF | Windows İletişim Temelleri |
| WebDAV | Web Dağıtımlı Yayın Ve Sürümlenme |
| WSDL | Web Servis Tanımlama Dili |
| XAML | Genişletilmiş Uygulama Geliştirme Dili |
| XHTML | Genişletilebilir Büyütülmüş Metin İşaretleme Dili |

# ŞEKİL LİSTESİ

Sayfa

# ÇİZELGE LİSTESİ

Sayfa

# ÖZET

## MOBİL CİHAZLARIN BULUTLA ETKİLEŞİMİ VE BU ETKİLEŞİMİN GÜVENLİK ÖZELLİKLERİ

Mirsat YEŞİLTEPE

Bilgisayar Mühendisliği Anabilim Dalı
Yüksek Lisans Tezi

Tez Danışmanı: Öğr. Grv. Dr. Ömer Özgür BOZKURT

Bu tez çalışmasında günümüzde popüler olan iki mobil işletim sisteminin yine günümüzde neredeyse internet kelimesinin yerine geçmeye başlayan bulut kavramına olan ilişkileri, farklılıkları, kendileri için bulut kavramı ve bu ortamda kullandıkları güvenlik mekanizmalarına değinilmiştir.

İkinci kısımda ise WCF'de güvenlik tiplerinden biri olan mesaj seviye güvenlik tipinin iletişim mesaj boyutu ve sayısı arttırılarak bu güvenlik tipindeki algoritmalarıyla birlikte çeşitli protokollere karşı çeşitli parametreler ile test edilmiş. Protokollerin bu güvenlik tipine olan uyumu araştırılmıştır.

**Anahtar Kelimeler:** HTTP, mesaj seviye güvenlik, TLS, Rest

## ABSTRACT

### INTERACTION BETWEEN MOBILE DEVICES AND CLOUD IN SECURITY FEATURES

Mirsat YEŞİLTEPE

Department of Computer Engineering
MSc. Thesis

Thesis Advisor: Öğr. Grv. Dr. Ömer Özgür BOZKURT

One type of security in WCF second part of the message-level security by increasing the size and number of types of communication messages with various parameters tested against various protocols with algoritmalarıl in this type of security. Protocols were investigated compliance with this security type.

One type of security is increased in the second part WCF message-level security type, size and number of communication messages with various parameters tested against a variety of protocols, algorithms with this security type. Protocols were investigated compliance with this security type.



**Keywords:** HTTP, mesage level security, TLS, Rest

# BÖLÜM 1

# GİRİŞ

Bu bölümde genel olarak çalışma ile ilgili literatür özeti, tezin amacı ve hipotezine değinilmiştir. Her bir platformun buluta bakış açısı açıklanmıştır.

## 1.1 Literatür Özeti

Bulut web servislerinin mobil cihazlarla platforma bağlı farklılıkları ve benzerliklerinden bahsedilmeden önce platformların bulut tanımlamalarına değinilmiştir.

### 1.1.1 Windows Phone

Bulut kavramına Windows Phone, internette OneDrive (Microsoft taraflı veri depolama) gibi disk alanı kullanımı olarak bakmaktadır. Saklanabilecek şey resim, müzik, video gibi aslında dosya olarak saklanabilen her şey olup bu bilgilere telefon, TV(televizyon) gibi aslında internet bağlantısı olan her cihazdan erişim sağlanabilir [1].

Şekil 1. 1 Windows Phone ile bulut etkileşimi[1]

### 1.1.2 Android

Android, yazılım özgürlüğü politikasına bağlı olarak oluşturulmuş bir işletim sistemidir. Android için özel olarak bir bulut tanımı bulunmadığından, Google Bulut platformu kavramı kullanılmaktadır. Bu kavram [2]'de:

Google Bulut Platformu kullanıcıların basitten karmaşık sitelere kadar birçok çeşit site oluşturabileceği bulut bazlı servisler bütünüdür[2]. Google Bulut Platformu ile mobil cihazlar için bir arka uç rahatça oluşturulabilir. Yeni mobil deneyimleri amatör bile olsanız geniş kitlelere bu platform sayesinde ulaşabilirsiniz. Artan isteğe ölçümsüdeyseniz yeni arka uçlar geliştirebilirsiniz. Bunu yaparken de yük denesi ve makine yönetimi gibi sorunları düşünülmez[2].

Şekil 1. 2 Google Bulut Platformu şeması (Sadece Android ve IOS mobil desteği vardır.)[2]

İki platformun buluta bakışlarında ki farklılıklar ve benzerlikler şu şekildedir:

Farklılıklar:

- İkisinin yardımcı programları farklıdır. Örn: her ikisinde veri depolarken bunu kendi programlarıyla yapmaktadırlar.
- Bulut platformunu sadece kendi ortamları gibi düşünmek isterler. Başka platformla olan uyumlarından bahsetmek istemezler.

Benzerlikler:

- Her iki platformda bulutu bir veri depolama alanı gibi düşünmektedir.
- Veriler halka açık, belli bir kesime açık veya gizli olarak saklanabilir.
- Her ikisinde internet bağlantısı olan cihazlarla uyumludur.

- Kullanıcılar sadece yapmak istedikleri iş ile ilgilenirler. Bulutta olan bitenden haberleri yoktur.
- Platformdaki istemci ve sunucuların çevrimiçi olmasını isterler.
- Her ikisi aslında kendisinde barındırdıkları bulut bazlı servisler topluluğudur.

Platformların bulut kavramına yükledikleri anlamı toparlarsak bulutun tanımı kavramı ortaya çıkar.

Bulut internet üzerinden bulut servislerinin dağıtıcısıdır. Bulut hizmetleri bireylerin ve işletmelerin uzak yerlerde üçüncü şahıslar tarafından yönetilen yazılım ve donanımları kullanmaları içindir. Bulut hizmetlerine örnek olarak çevrimiçi dosya depolama, sosyal ağ siteleri, web posta verilebilir. Bulut hizmetleri ağ bağlantısı uygun olduğunda bilgiye ve bilgisayarlar kaynaklarına erişim için kullanılır.

Bu bölümde son olarak hedef kullanıcılar açısından bulut kavramına değinilecektir.
**Özel Bulut:** Bulut altyapısı, belirli bir organizasyon için sadece işletilmektedir ve kuruluş veya üçüncü bir şahıs tarafından yönetilmektedir[3]. Kullanıcının mail bilgilerine istediği yerden sadece kendisi tarafından ulaşılması örnek verilebilir.

**Topluluk Bulutu:** Hizmet çeşitli kuruluşlar tarafından paylaşılan ve sadece bu gruplara uygun hale getirilir. Altyapı sahibi ve kuruluşlar tarafından veya bir bulut servis sağlayıcı tarafından işletilebilir. Kullanıcının bulutta yer alan bilgilerini herkesin görebilmesine olanak tanıması örnektir.

**Hibrit Bulut:** Kaynak havuzundaki metotların kombinasyonudur. Kullanıcının bazı bilgilerinin sadece kendisi tarafından bazı bilgilerinin ise açık olarak paylaşıldığı bulut türüdür.

### 1.2 Tezin Amacı

Bulut servislerindeki ilerlemelere bakıldığında birkaç yıl içinde cihazların depolama araçlarına gerek duymadan veri iletişimi yapacağı düşünülmektedir. Tezin amacı da farklı platformların bu bulut servislerini nasıl desteklediğini, diğer servislerden farklılıklarını, güvenlik mekanizmalarını, içinde kullanılan mimarileri platformdan bağımsız ve platforma bağlı olarak inceleyerek yerel konularda çeşitli kriterlere göre en iyisini belirlemeye çalışmaktır. Her platformun kendine göre üstünlükleri ve

dezavantajları olacağı bellidir. Burada amaç üstünlükleri ön plana çıkarıp dezavantajlardan da bahsederek en iyiye ulaşmaktır.

## 1.3 Hipotez

Mobil cihazlar oran itibariyle internet kullanımında en büyük paya sahiptir. Bu oran, mobil cihazların kullandığı REST mimarisinin web servisleri erişiminde kullanılan diğer mimarilerle karşılaştırılmasıyla da ortaya çıkmaktadır. İleride REST mimarisinin SOA mimarisi ile arasında her açıdan (sadece söz dizim açısından değil) eşitleneceği, belki de bilgisayar dünyasının kendini mobil dünyaya bırakacağı açıktır. Fakat bir sorun her zaman var olmaya devam edecektir. Güvenlik dün olduğu gibi bugün de sorun olmaya devam etmektedir. Bulut servislerindeki gelişmeler yeni güvenlik mekanizmalarının oluşturulmasına imkân tanırken, kötü niyetli kullanıcılarında güvenlik açıklıklarını daha hızlı bulmasının yolunu açmaktadır. Bu yarış ileride de devam edecektir.

# BÖLÜM 2

# PLATFORM BAĞIMSIZ MOBİL İLE BULUT UYUMUNUN ALTYAPISI

Bu bölümde iletişim protokollerinin tanımı, HTTP protokolüne genel bakış, bulut servislerinin mantığı, HTTP istek metotlarının çeşitleri ile çerezler konusuna değinilmiştir.

## 2.1 İletişim Protokolleri

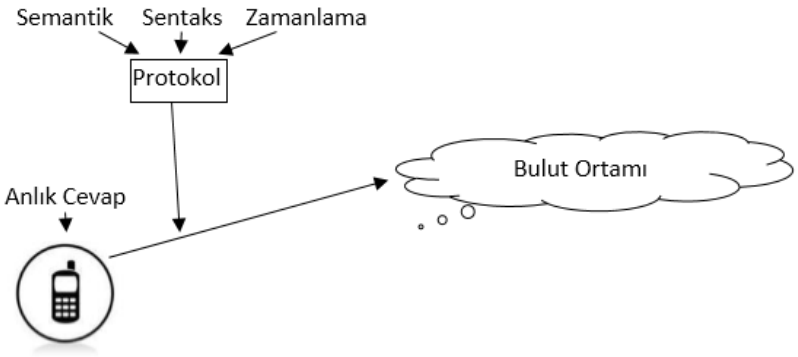
Bu bölümde mobil cihaz ile farklı ortamda bulunan sunucunun iletişimi nasıl gerçekleştirdiği konusu işlenecektir. Çoğu zaman mobil cihazlar bir ortamda olurken kendi aralarında ve bazende bulut kaynaklarıyla iletişim halinde olurlar. Bulut ile olan iletişim bu bölümün konusudur.

Bulut ile konuşmak için ilk başta bir iletişim protokolünün seçimi iletişim protokolünün desteklenmesi gerekir.

İletişim protokolü, mesaj değişimlerinde iyi tanımlanmış formatları kullanır. Her mesaj söz konusu durum için önceden belirlenmiş muhtemel bir karşılığa, bir yanıta yani ortaya çıkartılması amaçlanan tam bir anlama sahiptir. Bu yüzden bir protokolde esas olarak bir protokol sözdizimi, anlambilim ve iletişim senkronizasyonunun tanımlanması gerekir. Belirtilen davranışı nasıl uygulayacağı konusunda uygulanmaya tipik olarak uygulamadan uygulamadan bağımsızdır.

Bir protokol sadece yazılım veya donanım için uygulanabileceği gibi her ikisi için de uygulanabilir. Haberleşme protokollerinde tarafların mutabakata varması gerekir[6]. Anlaşmaya varmak için protokol, teknik bir standart halinde olmalıdır.

Bir programlama dili hesaplamalar için aynı durumları (semantik, sentaks ve zamanlama) kullanır. Bu nedenle protokoller ve programlama dilleri arasında yakın bir benzerlik vardır. Her ikiside iletişim için kullanılır [5].




Şekil 2. 1 Mobil ortamda protokol amaçları şeması

Eskiden beri bir yere gidildiğinde ve sinyal kesildiğinde görüşmelerin bitmesi sorunu bulunuyordu. Bulut ortamına geçildiğinde zaman aşımının belirlenmesi önemli bir sorundur. Zaman aşımından doğan problemlere bir web sitesine bağlanıldığında yeniden kimlik doğrulamanın ne zaman isteneceği örnek verilebilir. Zaman aşımı problemi protokollerin uzaklarla güvenlikli, güvenilebilir ve hızlı bir biçimde iletişimi için faydalarıdır.

İletişimin kabul edilen makul süresi bulunulan ortama göre değişir. Örneğin uzay ile iletişime geçilmek istenildiğinde iletişimin bir kaç dakika sürmesi kabul edilebilir bir durumken, bu durum mobil cihazlar için kabul edilmez.

Protokol, başka varlıklar ile nasıl iletişim içinde olunulacağının sentaksını tanımlayıp, başka varlıklara gönderilecek mesajların temel olarak formatlarını belirler. Yani bir mesaj gönderildiğinde protokolün özellikleri ve sentaksına sahip olmalıdır. Semantiği kurallarını belirleyerek mesajların anlamlarını tanımlar. Bu sayede bir istemci bir mesaj gönderdiğinde karşı taraf semantiği bilirse bu mesajı doğru yorumlayıp anlamlandırabilir. Yoksa karşı taraf için bu mesaj anlamsız olur.

Öte yandan mesaja yanıt gelmezse ne kadar bekleneceği bu durumda farklı bir mesaj mı yoksa hata mesajı mı gönderileceği hala sorundur. İyi tanımlanmış ve doğru zamanda kullanılan bir protokol mobil cihazların bulut ile iletişimde sorun çıkarmayacaktır.

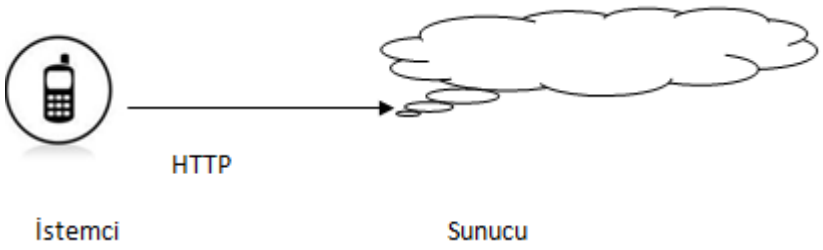
## 2.2 HTTP'ye Giriş

Bu bölümün amacı çok sayıdaki sunucu ile nasıl iletişime geçileceği ve küçük mobil cihazların dış ortamdaki ağa ve esas olarak binlerce sunucunun yer aldığı buluta nasıl bağlanacağının açıklanmasıdır.

Hiper Metin Aktarım Protokolü (HTTP) dağıtılmış, işbirlikçi, hiper bilgi sistemleri için bir uygulama protokolüdür[6].

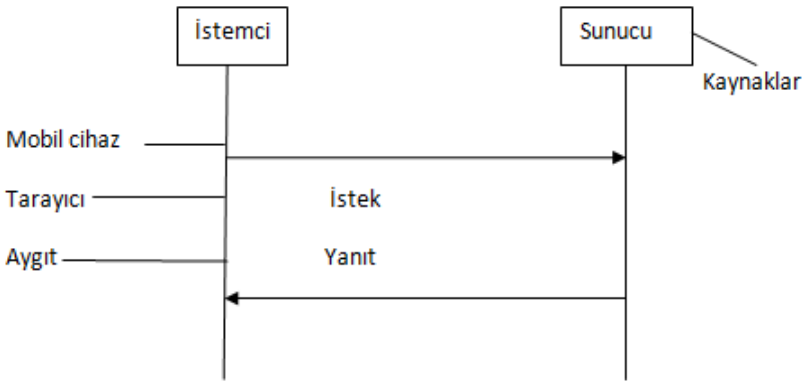
Mobil cihazların bulut ile iletişiminde HTTP kullanılır. HTTP temel bir diyalog veya dil olarak düşünülebilir. Ağdaki cihazların iletişimde kullanılabilir fakat bu çalışmada esas mobil cihazların bulut ile olan etkileşiminde kullanılabilmesi önemlidir. Burada (bulut ile iletişimde) iletişim için HTTP protokolü kullanılacaktır.

HTTP protokolü tarayıcının web sayfası ile iletişiminde kullanılan bir protokoldür. Web sayfası gibi web sunucusundan kaynak kabul etmek için kullanılır. Örneğin Facebook gibi büyük sitelere bağlanılmak istenildiğinde tarayıcımın bir ya da birden fazla Facebook sunucusuna HTTP istekleri serisi gönderir. Sonra sunucu veya sunucular bu istek veya istekleri alır. Verileri tarayıcıya gönderir.




Şekil 2. 2 HTTP protokolü ana şeması

HTTP istemci taraflı bir protokol olarak adlandırılır. Yani etkileşimler istemcinin isteklerine göre başlatılır ve sunucuya ulaşır. Sunucu istemcinin talep ettiği kaynaklara sahip olduğundan istemci kaynaklar için isteklerde bulunur. Bu isteklere yanıtlar dönülür. Bu yanıtlar isteklerle ilgili kaynakları içerebileceği gibi mesajla ilgili istemcinin gönderdiği isteğe sunucunun neden işleyemediğini durum olabilir. Sonuç olarak HTTP istemci taraflı uygulama protokolü olup istemcinin iletişimi başlattığı bir protokoldür.




Şekil 2. 3 İstemci ve sunucu ile HTTP protokolü ilişkisi

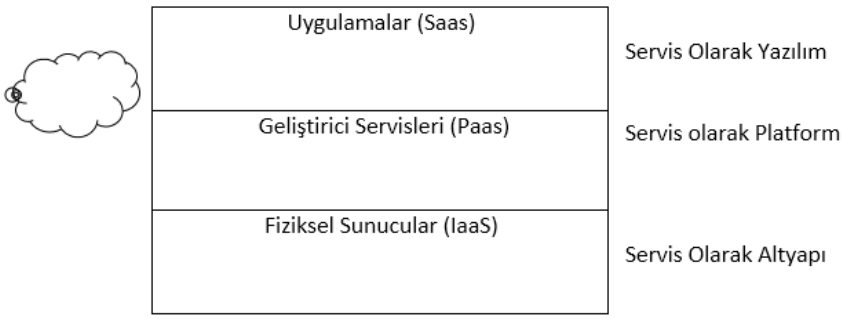
İstemci istek göndererek iletişim başlatır. Sunucu etkileşimi kabul eder. İşlemler yürütülür. Yanıtlar istemciye döndürülür. Bu HTTP'nin çalışma mantığıdır. İstemci mobil cihaz olabileceği gibi tarayıcı veya içinde interneti olan veya içindeki parçalarla HTTP protokolü üzerinde haberleşen aygıt olabilir.

HTTP istemci ile sunucu arasındaki etkileşimde kullanıla bilinecek genel amaçlı bir protokoldür. HTTP'nin popüler olmasının bir nedeni de bulut ile mobil cihazların etkileşimini önceden desteklemesi ve kullanılan birçok web sitesinde desteklenmesidir.

Sunucudan istek geri döndürüldüğünde; istemcinin mobil cihaz web sayfası mı yoksa hiç görülmemiş bir aygıt mı olduğu, HTTP açısından önemli değildir. HTTP üzerinden konuşulduğu zaman sunucudaki uygun servis ve kaynaklar tüketilir.

## 2.3 Bulut Servisleri

Bulut hizmetleri; bir kurumun sunucuları üzerinden istemcilerin isteklerine cevap olarak kullanıcılara sunulan kaynakların birleşimi anlamına gelmektedir. Yani çevrimiçi olarak istemcilerin istekte bulunup sunucuların istekler doğrultusunda cevap üreterek istemcilere göndermesi hizmetidir. Bulut hizmetleri uygulamaları, kaynaklara ve hizmetlere kolay, ölçeklenebilir erişim sağlamak için tasarlanmıştır ve bir bulut hizmeti sağlayıcısı tarafından yönetilmektedir[7].




Şekil 2. 4 Bulut servisi altyapısı [8]

Bulut bilişim, düzenli ve ölçeklenebilir bilişim teknolojilerinin sağladığı olanakların dağıtıldığı ve gerçek bir zamanda internet teknolojilerini kullanarak bir servis olarak tüketildiği bir çeşit programlamadır. Bulut servisleri, sadece internete ya da ağı kullanmaktan çok, birisi internet ya da ağdaki servis veya kaynağın dağıtımı için sorumluluk aldığında var olur[11].

## 2.4 HTTP İstek Metotları

Kullanıcı tarafından bir adres görüntülenmek istendiğinde, ilgili adrese bir talep gönderilir. Bu istek, sunucu tarafında değerlendirilerek bir yanıt oluşturulur ve istemciye geri gönderilir. Burada tarayıcı tarafından oluşturulan istek nesnesinde kullanıcı tarafından girilen bazı bilgiler de gönderilir[11]. Bu işlemler yapılırken çeşitli HTTP istek metotları kullanılır. Bunlar aşağıdaki gibidir.

**GET:** Belirtilen kaynağın bir temsilini ister. İsteklerde GET'in kullanımının veri almaktan başka bir etkisi bulunmamalıdır[6].

**HEAD:** GET metoduna benzer fakat sadece durum satırı ve başlık bölümü aktarılır.

**POST:** URI tarafından tanımlanmış web kaynaklarının yeni bir alt örneğinde varlık kapsüllemesi yapılmış isteği sunucunun kabul etmesi için kullanılır. İletilmek istenen veri; var olan kaynaklar için açıklama, haber, e-posta listesi, açıklama parçacığı için bir mesaj veya veri tabanına eklenecek bir veri olabilir[12].

**PUT:** Verilen URI'de saklanmış varlık kapsüllemesindeki isteklerde kullanılır. URI var olan bir kaynağa başvuruyorsa, bu URI değiştirilir, URI var olan bir kaynağı işaret etmiyorsa, o zaman sunucu bu URI ile kaynak oluşturulur[13].

**DELETE:** Belirlenmiş kaynağı siler.

**CONNECT:** Genellikle şifrelenmemiş bir HTTP Proxy üzerinden SSL şifreli iletişimi (HTTPS) kolaylaştırmak için bağlantıyı TCP/IP isteğine dönüştüren metottur[14].

**PATCH:** Bir kaynakta kısmi değişiklikleri uygulamak için kullanılır[15].

## 2.5 İstek Gövdesi Çözümleme

İnternette bir form doldurulduğunda bu gövde tiplerinden ve gövde tipinden URL çözümlemesi kullanılmış olur. Bu şekilde URL çözümlemesi gövde MIME tipidir ve sunucuya kuyruk parametreleriyle yapılması gerekeni gösterir. Sahip olunan anahtar değer çifti de URL'le ilişkilendirilir.

Parametrelerin kuyruk parametrelerine konulmasının yerine bu anahtar değer çiftinin gövdeye konulması daha çok tercih edilir. Bu sayede veriden gömülmüş URL elde edilir veya değer anahtar çifti sunucuya gönderilir.


Şekil 2. 5 İstek vücut çözümleme çeşitleri özellikleri

## 2.6 Çerezler

Bu bölümde yalnızca istemci tabanlı durumlardan olan çerezler (cookies) konusu üzerinde durulacaktır. Android tarafının desteklenmeyip WindowsPhone ortamı tarafından desteklenen viewstate (görünüm durumu) ve sunucu taraflı oturum konularından olan oturum yönetimi (session state) ve uygulama durum yönetimi (application state) konularına Windows Phone bölümünde değinilecektir.

Çerezler oturum tanımlanmasında kullanılır ve sunucu üzerinde tutulan bilgilere benzer bir yapının metin dosyası olarak istemci tarafında tutulmasını sağlar.

Çerezler iki çeşittir. Bunlar,

**Oturum Çerezleri:** Oturum çerezleri hafızada saklanır ve kullanıcı web uygulamasını kullandığı sürece erişilebilir.

**Kalıcı Çerezler:** Kalıcı çerezler, kullanıcı tercihleri ve kullanıcı kimlik bilgileri gibi uzun vadeli bilgileri depolamak için kullanılır[40]. Kalıcı çerezler kalıcı bir şekilde saklanır ve kullanıcı uygulamadan çıktığında kaybolmaz. Süresi dolduğunda kalıcı çerezler kaybolur.

Bir çerez dosyası en fazla 4 KB'lık alan kaplamaktadır. İstemci tarafında tutulduğu için sunucunun performansını etkilemez. Tek dezavantajı, tarayıcının oluşturduğu dosyalar temizlendiğinde tüm çerez dosyaları silineceğinden sayfa yüklenirken çerez dosyalarının kontrolünün yapılması gerekir[41].

### 2.6.1 Çerezlerin Güvenirliği

Her ne kadar çerezlerin var oluş sebepleri iyi niyetli olsa ve çerezlerin kendileri virüs veya kötü amaçlı yazılımlar taşımasa da (en fazla 4 KB'lık veri taşındığı için) güvenlik açığı oluşturma şansları vardır. Sonuçta çerez, istemcinin tarayıcıda açtığı uygulama ne kaydetmek isterse onu kaydedecektir[42]. Bu bir forma girilen bir kişisel bilgi olabilir. Esas amaç kötü niyetli uygulamaların özel bilgilere ait çerezlere ulaşamamasıdır.

# BÖLÜM 3

# BULUT SERVİS İNŞASI

Bu bölümde bulut servislerinin inşasında dikkat edilmesi gereken özellikler, protokol katmanlama, SOA ile REST farklılıkları ve HTTP havuzlama konularına değinilmiştir.

## 3.1 Bulut Servis İnşası

Günümüzde özellikle kurumsal firmalar yeni iş olanakları, piyasayla olan uyumlarının kolaylaştırması daha iyi müşteri servisleri ve değişen teknolojiye uyum sağlamak için kendi bulut servislerini oluşturmak isterler. Bunu yaparak istemci listelerini geliştirirler. Bu listenin içinde partnerler, müşteriler hatta ücretsiz olarak bu servise ulaşan kullanıcılar olabilir.

Bulut servisini oluşturmadaki amaç ne olursa olsun şu üç ana ilke üzerinde durulması gerekir[21].

- **Bulut servisi oluşturmada ki temel kullanım amaçlarını belirlenmelidir.**

Bu sağduyu gibi görünebilir, ancak birçok işletme iyi bir plan ya da temel tasarım olmadan ileriye taşımaktadır. Bu yapılmazsa benzer projelerin aynı aşamaları bir daha geçmesi söz konusu olduğunda ileride savurganlığa neden olur.

- **Hangi bilgilerin ne zaman açıklanacağına karar verilmelidir.**

Bu ilke yönetim sorunları ve güvenlik için çok önemlidir. Verilerin fiziksel konumunu, meta ve bulut barındırma hizmeti olanlar için kaynak sistemlerden uygun uyum biçimini anlamak bu ilkenin amacıdır.

- **API/Servis yönetim stratejisinin oluşturulmalıdır.**

Bulut servislerini oluşturmak için daha çok yol almak gerektiği açıktır. Fakat bu ilkelere bağlı kalınmadan yapılacak ilerlemenin de verimsiz olacağı da açıktır.

## 3.2 Protokol Katmanlama

Katmanlama, ağ sistemini mümkün olduğu kadar büyük parçalara ayırmaktır. Bu parçalar önceki parçadan sağlanan servise tek başına cevap verebilecek donanıma sahiptir[20].

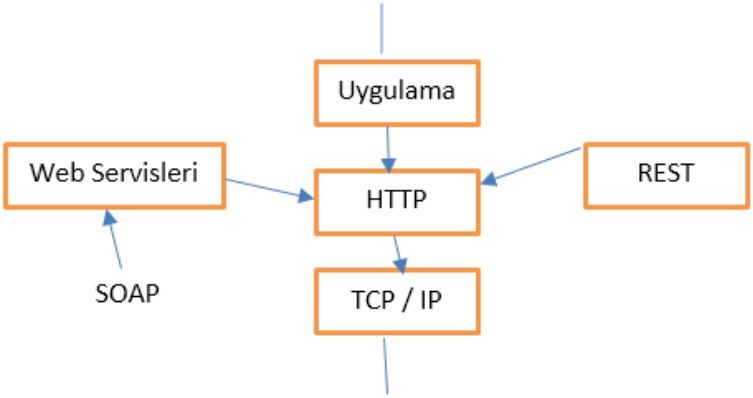
Katmanlama, her katmana özel donanımlar yapılmasına olanak sağlamıştır: fiziksel katmanda paket yönlendirmesi seçiciler, IP katmanında paket yönlendirmesi yönlendirici, taşıma katmanında paket yönlendirmesi ise NAT'lar tarafından yapılır. Bu sayede, basit donanımlarla yüksek TCP/IP performansları elde edilebilmektedir[23].

Katmanlamada her katmana özel donanım yapılabilmesi en önemli avantaj gibi görülse de bir katmandan diğer katmana gönderilecek verinin doğruluğundan emin olma gibi fazladan bilgi eklenmesi ise en önemli dezavantajıdır.

Mobil dünyada iletişim için HTTP protokolü kullanılır. Protokolde TCP/IP'den katmanlama olur. Daha altta farklı protokoller olabilir. Fakat üst yapı genelde aynıdır. HTTP'yi kullanan mobil uygulamaya özgü başka protokollerde bu katmanlama da üst sıralarda yer alabilir. Bu protokollerin amacı HTTP'ye bağlı kalarak yeni iletişim kuralları türetebilmektir. Esas olarak katmanlamada diğer protokoller HTTP'yi örnek alır. Yani HTTP mobil iletişimde bir nevi referans protokoldür.

Mobil iletişimde protokol katmanlamanın kullanılmasının esas nedeni istemci ve sunucu arasındaki iletişimlerde verilerin formatlarının değiştirilebilmesi ve buluta bağlanırken sorun olmamasını sağlamaktır. Yani esasen format değişse de referans protokol olan HTTP sayesinde buluta bağlanmada sorun olmayacaktır.

Mobil dünyada HTTP tasarım metodolojileri temel de iki çeşit olarak karşımıza çıkmaktadır. Bunlar SOAP ve REST olup detaylarına ilerdeki bölümlerde değinilmiştir. Fakat burada şunun bilinilmesi yeterlidir. Her iki yöntem de istemci ve sunucu arasında bulut ortamında iletişim için kullanılabilir. SOAP daha eski bir yöntemdir. Her ikisinin de kullanımı artmasına karşın, son yıllarda REST'in bütün bu tür metodolojilerde ki artış hızı o kadar yüksektir ki SOAP'ın genel yüzde içinde azalmasına neden olmuştur[12].




Şekil 3. 1 Mobil protokol katmanlama

### 3.3 Servis Odaklı Mimari

Bu bölümde servis odaklı mimari yapısı detaylı olarak açıklanacaktır. Mimarinin SOAP mesaj iletişim düzeni üzerinde durulmayıp, ileride REST iletişim düzeninden bahsedilecektir. Bulut bilişimde ilgili alanlarda iki farklı yapı kullanılmaktadır. SOA hakkında daha geniş bilgiye EK-1'de erişilebilir.

SOA(Servis Odaklı Mimari) = SOAP(kurumsal) + REST(mobil)

### 3.4 REST

REST esas olarak SOAP'ın dezavantajları düşünülerek oluşturulmuştur. Bu yüzden de SOA'nın avantajları REST'in dezavantajı, REST'in dezavantajı SOAP'ın avantajıdır. Aslında ikisi de istemci ve sunucu arasındaki iletişimde kullanılır fakat farklı ortamlarda farklı avantajlardan faydalanılmak istenildiği için farklı iletişim mesaj kuralları oluşturulmuştur.

SOAP'ı bir lüks otomobile REST'i ise motorsiklete benzetebiliriz. Eğer amaç otobanda(kurumsal) gitmek ise SOAP ama eğer amaç ara sokaklarda (mobil) gezmek ise REST daha iyidir. Mobil dünyada da amaç düşük hafıza ile etkin işlemler yapmak olduğundan REST'in kullanılması daha iyidir. Aşağıdaki şekillere bakılınca aynı amaçlı SOAP'ın mesaj kabaramı (iletişim mesajının orijinal mesajdan fazla olma miktarı) REST'ten çok fazladır.

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
 <soap:body pb="http://www.acme.com/phonebook">
  <pb:GetUserDetails>
   <pb:UserID>12345</pb:UserID>
  </pb:GetUserDetails>
 </soap:Body>
</soap:Envelope>
```

Şekil 3. 2 SOAP mesajı[37]

```
http://www.acme.com/phonebook/UserDetails/12345
```

Şekil 3. 3 REST mesajı[37]

Temsili durum transferi (REST) World Wide Web (WWW) mimarisinin bir soyutlamasıdır; daha doğrusu, REST bileşenleri, bağlayıcıları ve veri elemanları uygulanan mimari kısıtlamaları koordineli bir dizi oluşan bir mimari tarzı, dağıtılmış bir hiper sisteminde gerçekleşir. REST bileşenlerin rolleri odaklanmak amacıyla komponent (bileşen) uygulama ve protokol sözdizimi ayrıntıları göz ardı ederek, diğer bileşenler ile etkileşim ve önemli veri elemanlarının kendi yorumlanması üzerine kısıtlamalardır[50].

REST mimari kısıtlamaları tarafından uyarılan mimari özellikler şunlardır[51]:

- Performans, bileşen etkileşimleri kullanıcı tarafından algılanan performans ve ağ verimliliği baskın faktör olabilir.
- Ölçeklenebilirlik, bileşenleri arasında büyük parçaların numaraları ve etkileşimlerini destekleyecek bir iletişim kuralları bütünüdür.
- Ara yüzlerindeki basitlik.
- Değiştirile bilirlik, değişen ihtiyaçları karşılamak için (uygulama çalışınca bile) yapılan değişiklerin yeni ihtiyaca uyma zorluğu.
- Veri program kodu hareketli bileşenlerin değiştirile bilirliği.
- Güvenilirlik; bileşenleri, bağlantıları, ya da veri içindeki hataları varlığında sistem düzeyinde başarısızlık direnci olarak çeşitlendirilebilir[52].

REST stilinde sunucu/istemci, duyumsuzluk, düzgün ara yüz, katmanlı sistem, ön belleğe alınabilirlik, istek üzerinde kod konularına ileride değinilecektir.

### 3.5 HTTP Havuzlama

Bu bölümün konusu sunucuda kaynak ile ilişkilendirilmiş veri kullanılmak istendiğinde oluşabilecek bekleme süresidir. Yani herhangi bir durumda veri güncellemesi yapılmak istenildiğinde cevabın gelmesi için ne kadar süre bekleneceğidir. Bu süre, sunucudan gelen cevaba göre yeni veri döndürmesi için istek şekilleneceğinden önemlidir.

Sunucu ile çok sayıda istemcinin iletişimde olduğunu düşünelim. Sunucuya istemcinin istek göndermek istiyor. Bu mesaj önemli bilgiler içerdiği durumda sunucuya gönderilen mesajın istemci tarafından bilinilmesi istenir. Bazen de istemciler verilerini sunucundan gelecek cevaba göre güncellemek ister. İstemci ile sunucu arasındaki iletişimin istemci isteğine göre başlaması ve verinin güncellemesi de karşılaşılabilecek bir durumdur.

Bu bölümün esas sorunu sunucular ile istemcilerin haberleşmesinde istemcinin istek gönderip sunucudan gelecek yanıta göre yeniden istek göndermesi durumunda ve ilgili cevabın istemciye ulaşmaması durumunda istemcinin ne kadar süre beklemesi gerektiği, bu süre aşımında uygulayacağı stratejiler ile sunucuların çok istemciye cevap vermesi durumunda sunucu kaynaklarının böyle bir durumda en uygun kullanılarak israf edilmesinin önüne geçmede ki stratejiler incelenmiştir.

Sorunun bir çözümü sunucudan verinin ne zaman gelmesi gerektiğine istemcinin (kullanıcının) karar vermesidir. Verilerin yer aldığı liste görüntüleri ile belirlenmiş kullanıcı ara yüzlerinin bulunduğu ve ne zaman güncelleneceği konusunda bu yöntem yaygındır. Burada en önemli sorun, istemcinin sunucudan beklediği cevaba göre bir sonraki isteği ne zaman göndereceğinin belirlenmesidir. Bu sorunda en zor olanı istemcinin sunucudan beklediği cevaba göre yeni istek göndereceği zamandır. Bu yol sunucudan veri getirilmesi durumunda kullanışlıdır. İstemci güncelleme zamanını kendi seçer. Bu yöntemin başka bir özelliği de güncellemenin otomatik olarak yapıldığı durumda ya da uygulama çalıştığında verilerin sunucun üzerinde senkronize edilmesi istenildiğinde kullanılabilmesidir. Bunlar istemcinin sunucu ile olan etkileşiminde olay yönetim modelinin kullanılmasının amaçları ve faydalarıdır. Bu yöntemde sunucu ile iletişimi başlatan istemci sunucudan cevap gelmediği halde ne kadar sürece iletişimi koparmayacağını kendi seçer.

Sunucu bilgileri ile istemciyi güncel tutmak için kullanabilecek bir başka model havuzlama modelidir. Buradaki düşünce sunucu hazır olduğunda istemcinin iletişime başlamasıdır. Uygulama çalıştığı veya uygulamadaki servis hazır olduğu sürece istemci belli zaman aralıkları ile sunucudan verileri ister. İstemci tarafından seçilen aralıklarla sunucuya olan istek yenilenebilir. Bu durum istemcinin istek gönderip geriye veri döndürülmediği ve durumun neden kaynaklandığı bilinmediğinde olur. Burada sadece kör bir biçimde sunucu cevap gelmesi ümidiyle sunucuya bütün isteklerin gönderilmesiyle devam edilerek istek bırakılır. Fakat sunucudaki kaynaklar, sunucuya her zamana istek gönderilip yanıt dönmemesi ve sunucuda kaynakların yeni istemci için kullanılamaması yer ve istek işleme işlemleri yüzünden israf edilir. Bu nedenle istemcinin uygunluğu kontrol edilmelidir. Eğer hiçbir şey yoksa yanıt döndürülmelidir. Kaynak olarak ağ kaynakları, CPU (MIB) ve hafıza vs. düşünülmesi gerekir.

Çoklu istemcinin bulunduğu ortamda istemciler sürekli veri isteklerinde bulunup sunucudaki kaynakların verimli kullanımını etkileyecek kadar trafik yoğunluğu oluşturur. Bu yüzden havuzlama istemcilere çok hızlı cevap verebilmesinin yanında sunucuda çok fazla kaynak harcaması (kullanımı) yapıldığında çok fazla maliyet artışına neden olur.

İsraf edilen kullanımı azaltmak için sunucular istemcilerin veri(yanıt) için bekleme sürelerini bildirmelerini ister. Eğer sunucuda güncelleme hızlı oluyorsa ve bir istemci veya birkaç istemci istek gönderiyorsa cevabın gönderilmesi normaldir. Böyle olduğu durumlarda da sabretme süresinin kısa olması(genellikle birkaç saniye) beklenir. Eğer birkaç istek güncelleme olmadan geri dönerse istemci bekleme süresini arttırarak yeni havuzlama yöntemiyle yoluna devam eder. İstemci bu sefer en fazla bekleme süresini arttırmış olur. Eğer yine cevap gelmez ise bekleme süresi marjinal olarak attırılır

Tavan süresi aşıldığında artık istemci daha fazla beklemeyecektir. Daha hızlı havuzlama oranına geri dönebilir ve bu durumda bazı sonuçlar alır. İstemcinin beklediği bekleme sürenin belirli bir oran altında yeni istekler yanıt döndürmeye başladığında ise bekleme süresi azaltılır. Bu model sunucudaki kaynakların istemci ile iletişimde verimli kullanılmaya çalışılmasını amaçlar.

## 3.6 İtme Hizmetleri

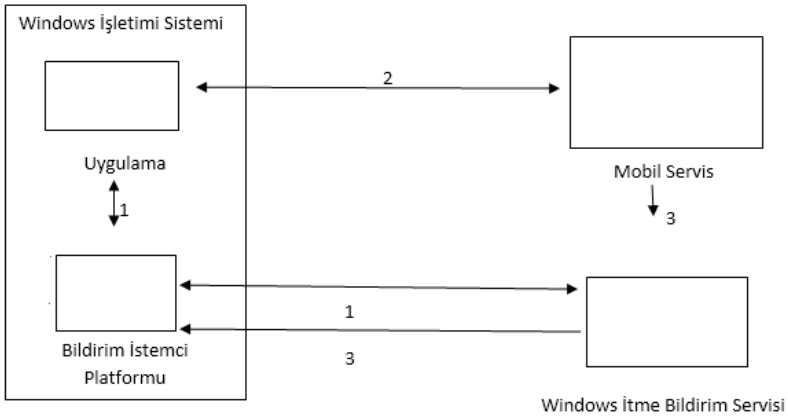
İtme Hizmetleri kullanıcılarını uyarmak için etkin ve güvenilir bir yoldur. Hatta arka plan uygulamaları, gerçek zamanlı olarak kullanıcıları bilgilendirmek için izin verir. İtme Hizmetler yaygın olarak hava durumu güncellemeleri, mesajlaşma hizmetleri, posta bildirimi, kupon hizmetleri gibi mobil uygulamaları içeren çeşitli alanlarda kullanılmaktadır. İtme Hizmetler, isteğe bağlıdır, ancak gerekli hale gelmiştir[47].

1. İtme servisinin kullanımı için izin alınılmalıdır.
2. İtme mesajları POST metotu ile gönderilmelidir.
3. Mesajı istekleri; hata ayıklama hataları, durum kodlarını ve mesajlarını kullanmalıdır[51].

### 3.6.1 Windows Tabanlı İtme Servisleri

Bir zorlama bildirim gönderme işlemi üç temel adımdan oluşur:

**1. Kanal İsteği:** WNS bir Kanal URI istemek için WinRT API yararlanılır. Kanal URI uygulamanıza bildirimler göndermek için kullandığınız benzersiz bir tanımlayıcı olacaktır.




Şekil 3. 4 Windows taraflı itme servisleri kullanımı [49]

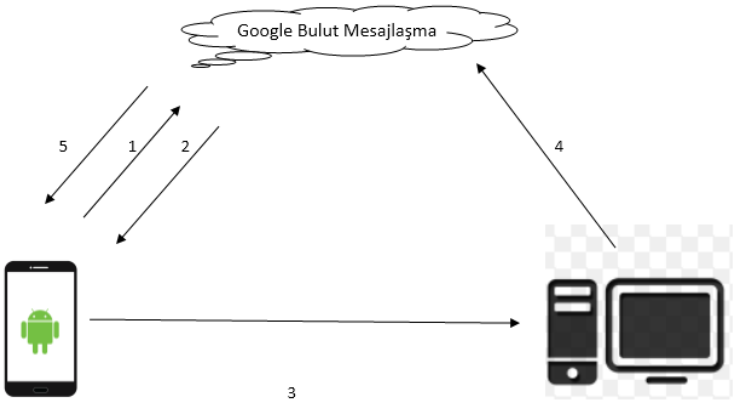
**2. Windows Azure Mobil Servisi İle Kanal Kayıtı:** Eğer kanal var sonra kanalı saklayabilir ve hizmetleri bu verilen kanala bir bildirim göndermek için zamanı olduğuna karar kadar herhangi bir uygulamaya özel veri (örneğin, kullanıcı profilleri ve gibi) ile ilişkilendirilebilir.

**3. WNS İçin Kimlik Doğrulama Ve Bildirim İtimi:** Eğer bunu oluşturmak ve kanal alıcıya bildirim için zorlayabilir ve bir kere öncelikle WNS itmek sonraki her bildirim için

kullanılacak bir simge almak için OAuth2 kullanarak WNS karşı kimlik için gerekli olan kanala URI bildirimleri göndermek için Push. wns kullanılır. Tüm metotların sıfırdan yazılmasına kıyasla gerçekleştirmek için bu görevi son derece hızlı yapılabilir.

### 3.6.2 Android Taraflı İtme Servisleri

Genel olarak Anroid'de itme servislerinin kullanımını sıra ile itme servisine kayıt olma, izinlerin düzenlenmesi, itme ikonunun seçilmesi, ayrıştırma API anahtarını ekleme, itme bildirimlerini etkinleştirme, test itme bildirimi gönderme olarak özetlenebilir[51].




Şekil 3. 5 Anroid taraflı itme servisleri kullanımı

1. ADF mobil uygulaması başlatıldığında, itme bildirim servisi ile bir kayıt isteği (ortalama bağlanma isteği) gönderir. Android cihazları kayıt için GCM sunucusuna gönderici kayıt niteleyicisi gönderir.
2. Başarılı bir kayıttan sonra, GCM sunucusu, ADF mobil uygulamasına ve aygıtına bir kayıt niteleyicisi gönderir.
3. ADF mobil cihazı bu kayıt niteleyicisini aldıktan sonra, uygulama yaşam döngüsü olay dinleyicisindeki onOpen metotu çağrılır.
4. Sağlayıcı uygulama sunucusu, kayıt niteleyicisini daha sonraki kullanımları için veri tabanında saklar.
5. GCM sunucusu kayıt niteleyicisini kullanarak mesaj dağıtılır[50].

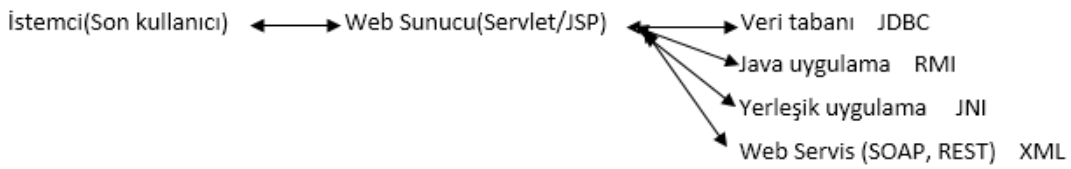
## 3.7 Servlet Nedir?

Servlet bir sunucu yeteneklerini genişletmek için kullanılan bir Java programlama dili sınıftır. Servlet istekleri herhangi bir tür yanıt verebilmesine rağmen, yaygın web sunucuları tarafından barındırılan uygulamaları genişletmek için kullanılır, böylece

sunucular yerine web tarayıcılarında çalışan Java uygulamaları gibi düşünülebilir[52]. Java tabanlıdır, Windows tarafında ASP.NET olarak çevrilir.

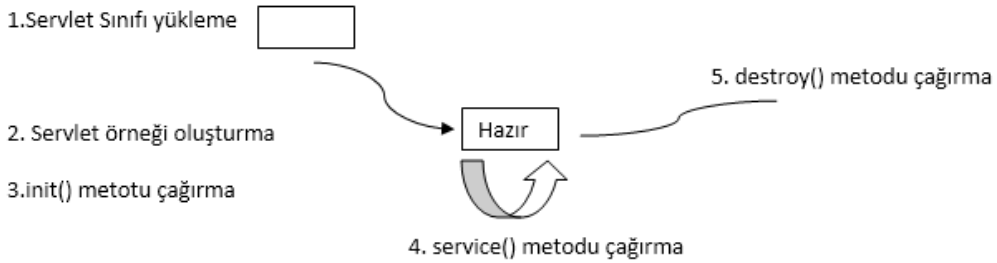
Servet'ler en sık şu nedenler için kullanılmaktadır:

- Sunulan HTTP formatında süreç ve veri depolamak,
- Veri tabanı sorgu sonuçları gibi dinamik içerik sağlamak,
- HTTP duyumsuzluk protokolündeki durumu bilgisini yönetmek için kullanılır[53].




Şekil 3. 6 İstemci servet ilişkisi [54]

### 3.7.1 Servet Yaşam Döngüsü




Şekil 3. 7 Servet yaşam döngüsü

**1.Servlet Sınıfı yükleme:** Web konteyner ile ilk istek istenildiği zaman bu sınıf oluşturularak döndürülür.

**2.Servlet örneği oluşturma:** Servet sınıfı yüklendikten sonra, web konteynerinin örneği oluşturulur. Servet örneği sadece yaşam döngüsü içinde örneklendirilir.

**3.init() metodu çağırma:** init() metodu Web konteyneri tarafından servlet örneğinde servlet oluşturur.

**4.service() metodu çağırma:** Bu konteyner bu servisi servlet için her istek çağrıldığında döndürülür.

**5.destroy() metodu çağırma:** Web konteyneri destroy() metodu çağrılarak uzaktan servlet örneğinden önce çağrılarak temizleme aktivitesinde kullanılır[55].

### 3.7.2 Servlet ile REST Arasındaki Fark

Bu bölümde Android tarafından kullanılan Servlet kavramı ile istemci ile sunucu arasındaki iletişimin kurallar bütününü belirleyen REST'in farkına değinilmiştir.

Servlet basit ve yetenekli sunucu tarafı bileşenleri yazmak için API sağlar. REST hizmetleri yazma için daha yüksek düzeyde destek sağlar. Ayrıca REST istemci ve sunucu, düzgün ara yüzü, durumsuz, ön belleğe alma özelliği ve katmanlı mimari gibi diğer olanakları sağlar[56].

Günümüzde web uygulamaları ile temel güvenlik problemlerinin artmasındaki en önemli nedenlerden biri istemci taraflı bileşenlerin ve kullanıcı girdilerinin sunucunun kontrolünde olmamasıdır. İstemci ve istemciden dönen veriler esasen güvenilmezdir[59].

### 3.7.3 İstemci İle Veri İletimi

Çoğu uygulamalar güvenli olmayan verileri istemci yoluyla taşladığından dış ortamdan gelecek saldırılara açıktır. Bu yüzden eğer mümkünse istemci yoluyla veri taşınılmasından kaçınılması gerekir. Sanal olarak akla gelebilen herhangi bir senaryoda veriler sunucu taraflı tutulabilir ve sunucu mantığında gerekli olduğunda referans edilebilir.

Önemli verilerin istemci yoluyla taşınmasından başka alternatifleri yoksa verinin kullanıcının değiştirmesine olanak sağlamamalı için şifreli veya imzalı olması gerekir. Bu yaparken de iki adet gizli tehlike ile karşı karşıya olduğu unutulmamalıdır:

Eğer kullanıcı şifreli verilerin bir kısmının şifresiz değerlerinin biliyor veya kontrol edebiliyorsa bir dizi kriptografik yöntem uygulayarak sunucu tarafından kullanılan şifreleme anahtarına keşfetmeye çalışacaktır. Bu sayede kontrolü istemci ele alarak istenmeyen durumların oluşabilmesine ortam hazırlayacaktır.

### 3.7.4 İstemci Taraflı Üretilen Verinin Sunucuda Doğrulanması

İstemci taraflı HTML'deki form alanları ve JavaScript gibi hafif kontroller kolayca istenilen yapıya dönüştürülebilen biçimdedir ve varsayılan olarak sunucudan işlenerek döndürülen giriş değerleri hakkında hiçbir güvenlik tedbiri almaz.

Ağır siklet istemci komponentleri istenilen biçimlere daha zor döndürülebilirse de saldırganların kısa sürede sisteme saldırmasını önler.

İstemci taraflı üretilen verinin doğrulanmasında güvenli tek yol verinin sunucu taraflı olmasını sağlamaktır. İstemciden veri geldiği her zaman bakıma alınmalıdır. Yani elde edilen veriler istenilen verilere uygunluğu test edilmelidir.

# BÖLÜM 4

# SOA MİMARİSİ KATMANLARI VE KATMANLARLA İLGİLİ TANIMLAMALAR

Bu bölümde SOA mimarisinin katmanları ve ilgili katmanda kullanılan tanımlamalar açıklanacaktır.

## 4.1 SOA Temelleri

SOA mimarisinin ilk katmanı olan SOA temelleri mimarinin oluşturulmaya başlanmasından önceki aşamadır. Amaç alt sistemlerin oluşturulmasından önce ortamın hazırlanmasıdır. Bu katmanda kullanılan tanımlamalar şöyledir.

Çizelge 4. 1 SOA Mimarisi Katmanlarından SOA Temellerinin Kullandığı Araçlar[20]

| ***Kurumsal SOA Güvenliği*** |
|---|
| ***SOA Güvenliğinin Yapıtaşlarının İnşası*** |
| ***SOA Temelleri***<br>•XML, isim uzayları ve şemalar<br>•XPath<br>•SOAP<br>•WSDL<br>•JAXP ve DOM<br>•Apache Axis JAX-RPC |

*XML:* Genişletilebilir İşaretleme Dili (XML) Internet üzerinden yapılandırılmış belge alışverişi için Genelleştirilmiş Standart İşaretleme Dili (SGML) basitleştirilmiş bir versiyonudur[25]. Belge alışverişi olarak veri taşınmasında kullanılmasının yanında istemci ve sunucu arasındaki iletişimin kurallarının belirlenmesinde de kullanılabilir.

*İsim Uzayı:* XML isim uzayı dört özellik kullanılarak tanımlanır. Bunlar dil (eleman üzerinde içeriğin dilini belirlemede kullanılır), kimlik (elemanı benzersiz bir şekilde

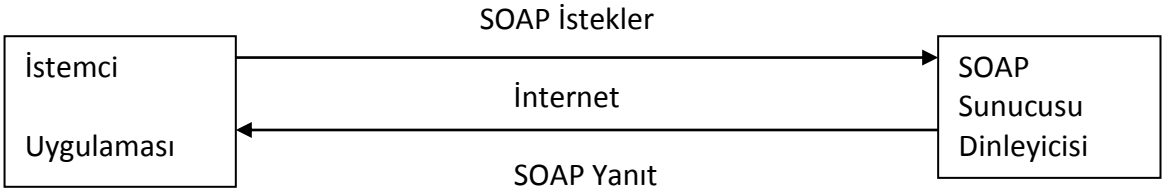
tanımlamak için kullanılır), uzay(eğer işlem sırasında korunur boşluk düğümleri bulunuyor istediğinizi belirtmek için bir eleman bu özelliği kullanılır), temel(temel URL'in tanımlanmasında kullanılır) özelliklerinde oluşur[22].

Şema: İstemci ve sunuculu bir ortamda ki belge alışverişinde her zaman bir şemaya uygun olma zorunluluğu vardır. Bu sayede platform bağımsızlığına yaklaşılmış olur. Şemaların belki de tek dezavantajı platform bağımsızlığı desteklemeye çalıştıklarından yeni versiyonlarının çıkarılmasındaki zorluktur. Örneğin; XML sadece 1. 0 versiyonu vardır.

*XPath*: XPath, bir XML belgesindeki elemanlar ve öznitelikler üzerinde gezinmek için kullanılır[23].

*SOAP:* Uzak nesne çağırmak için çoğunlukla iyi bilinen HTTP protokolü kullanarak istemci / sunucu iletişimini kolaylaştırmak için IBM ve Microsoft tarafından eş zamanlı geliştirilen protokoldür[2].

*WSDL:* WSDL uç noktaları bir dizi gibi ağ hizmetlerini tanımlayan bir XML formatı içeren mesajlar da belge odaklı veya prosedür yönelimli faaliyet bilgidir[28]. WSDL dokümanları çoğu zaman kodlayıcı tarafından değil programatik olarak oluşturulur. Kullanılmasının bir sebebi de sunucunun kendini ortama bu dokümanla açarak istemcilerden çağrı bekler. Platform bağımsızlığını destekler.



Şekil 4. 1 SOAP mesajları

*JAXP ve DOM:* XML verisini okuyup yazabilmek için hazırlanmış program parçacıkları ve API'lerdir [25]. SAX (XML İçin Basit API) XML dokümanlarını baştan sona bir akış şeklinde çeşitli olaylara dayandırarak okuma yapar. DOM (Doküman Nesne Modeli) XML dokümanlarını hafızaya bir ağaç yapısı şeklinde yükler.

*Apache Axis JAX-RPC:* JAX-RPC (XML-RPC Tabanlı için Java API) Web servisleri veya diğer ile uzaktan yordam çağrıları (RPC) dâhil Java geliştiriciler sağlayan Java Web Hizmetleri Geliştirici Paketi (WSDP) bir uygulama programı arabirimi (API) Web-tabanlı

uygulamalardır. JAX-RPC, diğer uygulamaları ile Web hizmetlerini çağırmak için uygulamalar ve Web hizmetleri için daha kolay hale getirmeyi amaçlamaktadır. JAX-RPC, SOAP gelişimi için bir programlama modeli tabanlı uygulamalar sağlar. JAX-RPC programlama modeli SOAP protokolü düzey zamanı mekanizmaları özetleme ve WSDL arasındaki haritalama hizmetleri sunarak geliştirilmesini kolaylaştırır. Tanımlamalarda önceden kullanılmış bazı tanımlayıcıların yeniden kullanıldığı görülebilir. Bunu nedeni SOA'nın bir bütün mimarisinden çok parçaların uygun bileşenlerinde bir üst veya ara sistem oluşturmak olduğu için mimariyi tasvir eden tanımlamalarda önceki tanımlayıcılar kullanılabilir.

### 4.2 SOA Güvenliğinin Yapıtaşları

SOA mimarisinin ikinci katmanı olan *SOA Güvenliğinin Yapıtaşlarının İnşası* ortamın oluşturulması aşamadır. Amaç alt sistemlerin oluşturulması, kurumsal iletişim yolunun hazırlanması, temel güvenlik tedbirlerinin alınmasıdır. Bu katmanda kullanılan tanımlamalar şöyledir.

Çizelge 4. 2 SOA Mimarisi Katmanlarından *SOA Güvenliğinin Yapıtaşlarının İnşasının* Kullandığı Araçlar[26]

| *Kurumsal SOA Güvenliği* |
|---|
| *SOA Güvenliğinin Yapıtaşlarının İnşası*<br>•Web Servis Güvenliği<br>•Kimlik Doğrulama (Kullanıcı İsmi Belirteci/ Sindirmeli Şifre)<br>•Kerberos<br>•XML Şifreleme<br>•XML İmzalama<br>•Web Servis Güvenli Konuşma<br>•JAX-RPC Başlıkları<br>•JAAS<br>•JGSS<br>•Apache XML-Güvenliği |
| *SOA Temelleri* |

*Web Servis Güvenliği:* Web servislerine güvenlik uygulamaları için SOAP'ın bir uzantısıdır. Web servis spesifikasyonunun bir üyesi olarak OASIS tarafından üretilmiştir.

Protokol bütünlüğü, gizlilik ve çeşitli güvenlik belirteci biçimleri ile nasıl iletişimi geçileceğini belirtir. Bu yapılara örnek olarak Güvenlik İddia İşaretleme Dili (SAML), Kerberos, X.509 yapıları örnek verilebilir. WS-Güvenliği, uygulama katmanında çalışan bir SOAP mesajının başlığında güvenlik özellikleri içerir[31]. Anahtar yönetimi, güven önyükleme, federasyon ve teknik detaylar (şifreler, biçimleri, algoritmalar) üzerinde anlaşma WS-Güvenliği kapsamı dışındadır.

*Kimlik Doğrulama:* Sunucunun kendisindeki bilgi veya siteye erişim sağlayanın tam olarak kim olduğunu bilmesi gerektiğinde kimlik doğrulama kullanılır. Kimlik doğrulamasında, kullanıcı veya bilgisayar / sunucu veya istemci kimliğini karşı tarafa kanıtlamak zorundadır. Genellikle, bir sunucu tarafından kimlik doğrulama, kullanıcı adı ve parola kullanımını gerektirir[28]. Kimlik doğrulamanın en önemli dezavantajı web servisiyle iletişime geçmekte izinli olan istemcilerin servisi hangi yetki seviyesinde kullanması gerektiğinin belirlenmemesidir ki, bu sorun yetkilendirme ile çözülmüştür.

*Web Servis Güvenliği Kimlik Doğrulama Çeşitleri:*

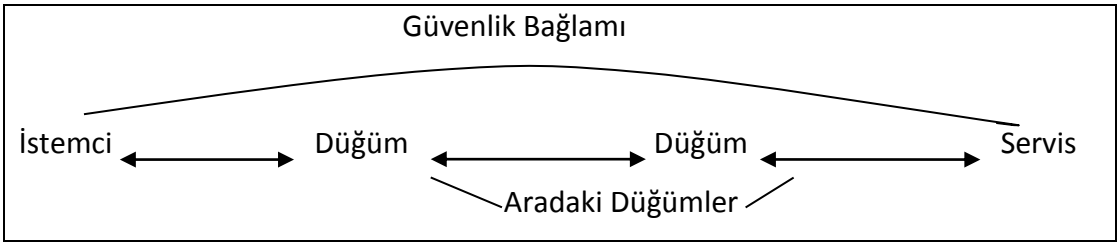
- *Güvenliksiz:* Tüm istekleri herhangi bir güvenlik tedbiri olmadan işler
- *SAML Belirteci*: İstekler mesaj güvenliğini doğrulamak için bir SAML kimlik sağlayıcı ile birlikte talep edilir.
- *İmzalı SAML Belirteci:* İstekler mesaj güvenliğini doğrulamak için bir kimlik sağlayıcı ile birlikte geçerli bir dijital imza SAML belirteci ister.
- *Kullanıcı İsmi Belirteci:* Tüm istekler geçerli bir kullanıcı ismi ve şifre güvenlik tedbiri ile işlenir.
- *İmzalı Kullanıcı İsmi Belirteci:* İstekler mesaj güvenliğini doğrulamak için bir kimlik sağlayıcı ile birlikte geçerli bir dijital imza içerir.
- *Şifreli Kullanıcı İsmi Belirteci*: Tüm istekler geçerli bir şifrelenmiş kullanıcı ismi ve şifre ile kimlik doğrulama sonucu gerçekleştirilir.[27]

*Kerberos:* Kerberos birçok kullanım kimlik ve güvenlik altyapısı önceden kurulmuş bir güvenlik mekanizmasıdır[32]. Bu şartname Web hizmeti güvenlik mimarisi Kerberos güvenlik ortamları ile bütünleşmiş çalışmaktadır. Kerberosun en önemli özelliği doğal olması (daha önceden çoğu sistemde tanımlanmış olması), mimarisinin iyi olması ve çoğu ortamla entegre olabilmesi)dır.

*XML Şifreleme:* XML Güvenliği parçalarından biridir. Bu XML tabanlı mesajlar gizliliğini sağlamaktan sorumludur. Çeşitli standart senaryoları ve özelleştirilmiş uygulamalar benimsenmiştir[31]. XML Şifreleme depolanan XML öğelerinin gizliliğini sağlamak için bir metodu kapsar.

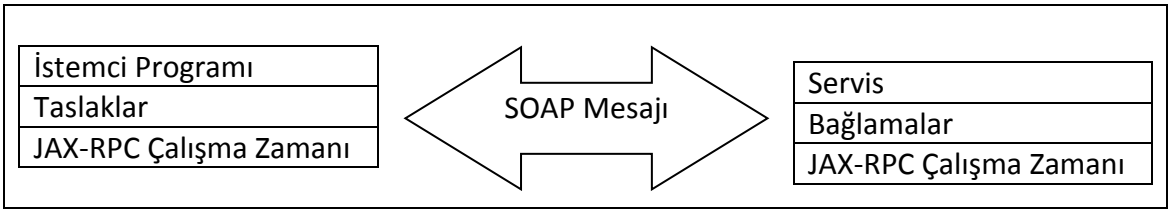
*XML İmzalama:* Bir dijital imza, bir mesaj-imza alıcı olduğunu doğrulamak için izin veren bir veri parçasıdır[32].Mesajın değiştirilmemesi ve sahibinin özel anahtarı tarafından mesaj imza çiftinin oluşturulması XML imzalamanın önemli avantajıdır.

*Web Servis Güvenli Konuşma:* WS-Güveli Konuşma, SOAP mesajları ile güvenli bağlamlarını kurmak ve kullanmak için nispeten yeni bir protokoldür[35]. Öncelikle, güvenli bir bağlam bir istemci arasında kurulan sunucudur.




Şekil 4. 2 Web Servis Güvenli Konuşma Mantığı

*JAX-RPC(XML Tabanlı Uzaktan Yordam Çağrısı (RPC) İçin Java API'si) Başlıkları: Web Hizmetleri ve Web hizmetleri istemcilerine için bir yapı API'sidir. Tip-haritalama sisteminin (Java WSDL) özellikleri; hizmet uç noktası, istisna işleme, mesaj yükleyicileri, hizmet uç nokta bağlama, süre hizmetleri gibi özellikleri vardır.*




Şekil 4. 3 JAX-RPC Web Servisi ve Müşteri Arasındaki İletişim

*JAAS(Java Kimlik Doğrulama ve Yetkilendirme Servisi):* JAAS bir Java ortamında bu yazılım korumaları bina için standart programlama arabirimini tanımlar. JAAS Java güvenlik mekanizmaları kesinlikle kod tabanlıdır[34].

*JGSS (Java Genel Güvenlik Servisi):* JGSS API iletişim mesajların uygulamalar arasında güvenli alışverişini sağlar. JGSS varsayılan güvenlik mekanizması olarak Kerberos V5 kullanan bir API çerçevesidir[35]. JGSS altta yatan farklı güvenlik mekanizmaları

karmaşıklığı ve özelliklerinden kaynaklanan güvenli uygulamalar korur. JGSS kimlik ve mesaj kökeni doğrulaması, mesaj bütünlüğü ve mesaj gizliliği sağlar.

*Apache XML Güvenliği:* İstemci ve sunucu arasındaki iletişim dosyasıyla sağlanmaya çalışılan güvenliğin, programatik (Java platformuna yönelik) olarak sağlanmaya çalışılmasıdır. Java taraflı XML güvenliği olduğundan platform bağımsızlığından uzaklaştırdığı düşünülebilir. Aşağıda bu güvenlik mekanizmasının bazı paketleri gösterilmiştir. Bu paketlerin isimlerinde anlaşılabileceği gibi bu bölümde yer alan tanımlamaların çoğu XML güvenliğinde seçenek olarak sunulmaktadır.

Çizelge 4. 3 Java 1.5.5 API'nin Apache XML Güvenliği Paketlerinin Bir Kısmı[40]

| | |
|---|---|
| javax.xml.crypto | XML kriptografi için ortak sınıfları |
| org.apache.xml.security.signature | Özel XML imza sınıfları |
| org.apache.xml.security.c14n.implementations | Kurallı uygulamaları |
| org.apache.xml.security.exceptions | Bu kütüphane tarafından kullanılan genel istisnalar |
| org.apache.xml.security.utils | Genel yarar sınıfları |
| org.apache.xml.security.keys | Genel anahtar ile ilgili malzeme |

### 4.3 Kurumsal SOA Güvenliği

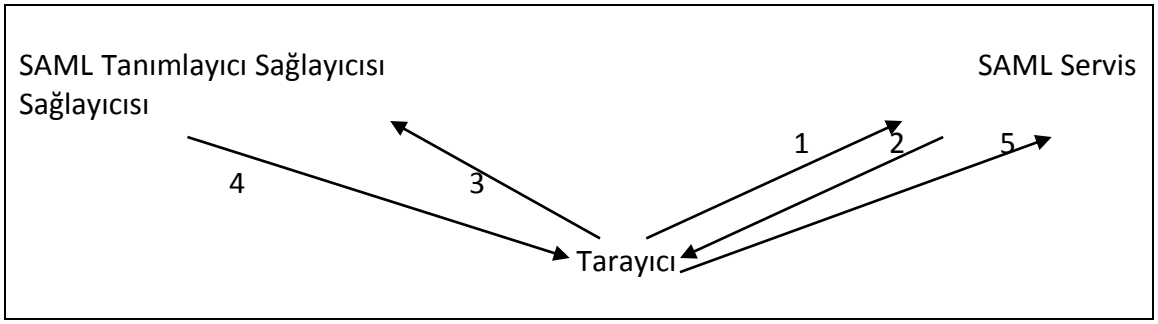
SOA mimarisinin son katmanı olan *Kurumsal SOA Güvenliği katmanı* oluşturulan ortamdan yeni sistemlerin oluşturulması sürecini içeren katmandır. Amaç uygun parçalara bölünüp, temel güvenlik tedbirlerinin alındığı ortamdan yeni sistemlerin uygun olarak oluşturulmasıdır. Bu katmanda kullanılan tanımlamalar şöyledir.

Çizelge 4. 4 SOA Mimarisi Katmanlarından Kurumsal SOA Güvenliğinin Kullandığı Araçlar[30]

| |
|---|
| *Kurumsal SOA Güvenliği*<br>•SAML<br>•Web Servis Güven<br>•AON<br>•Web Servis Güvenlik Politikası<br>•Plana Göre Yerleştirme Mimarisi<br>•XML Saldırıya Maruz Kalma Durumu Yönetimi<br>•OpenSAML |
| *SOA Güvenliğinin Yapıtaşlarının İnşası* |
| *SOA Temelleri* |

*SAML(Güvenlik iddia İşaretleme Dili):* Ws-Güven belirtim protokolünün amacı belirsizlik güvenlik belirteci özelliğine sahip olmasına rağmen, SAML'ın temel işi bu tercih edilen seçenek oluşturmak ve değer protokoller üzerinde çalışabilme kabiliyetidir. SAML belirteci iddialar hizmeti tüketicilere ilişkin bilgilerin üç çeşit sağlanabilme yöntemleri şunlardır:

- Doğrudan Onaylama: Hizmet tüketicisinin bazı kimlik doğrulama kimlik bilgileri (adı, kimliği vb.) için hizmet erişimini sınırlamak.
- Yetki Onaylama: Hizmet tüketicinin bazı yetkilendirme yetkileri için hizmet erişimini sağlamak.
- Onaylama Özelliği: Hizmet tüketicisinin belirli özellikleri için hizmet erişimini sınırlamaktır[42].



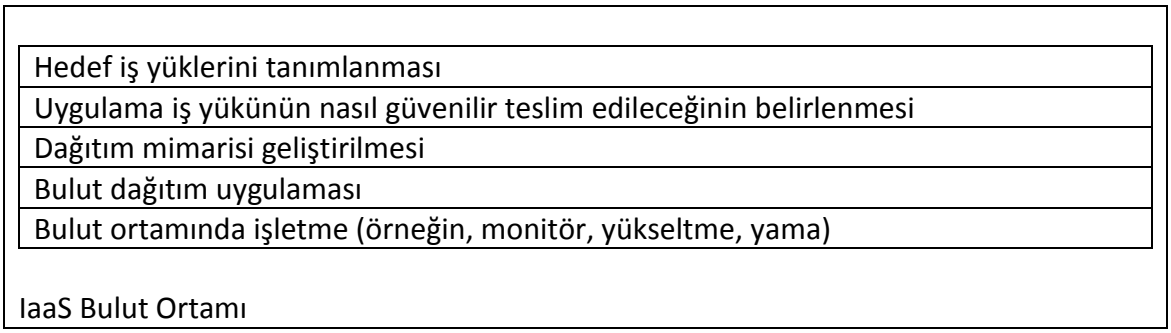

Şekil 4. 4 SAML Özeti(1- Kullanıcı belirli bir kaynak için servis sağlayıcısına bir istekte bulunur. 2-Servis sağlayıcısı kullanıcı için kimlik doğrulaması olması gerektiğini algılar ve bunların SAML Kimlik Sağlayıcı için kullanıcıyı yönlendirir. 3-Kullanıcı kendi tanımlayıcısına erişir ve kimlik doğrulanır. 4-Bir kez doğrulanmış, tanımlayıcı geri servis sağlayıcısı için SAML Yanıtı gönderir. 5-Servis sağlayıcısı, SAML iddiayı işler ve kullanıcı günlüklerine kaydedilir.) [43]

*Web Servis Güven:* WS-Güvenliği bir WS-* spesifikasyonu olup OASIS standartıdır ve WS-Security uzantıları, özellikle verilmesi ile ilgili yenileyen ve güvenlik jeton doğrulamanın yanı sıra, kurmak varlığını değerlendirmek ve güvenli bir ileti alışverişi katılımcılar arasındaki komisyoncu güven ilişkileri için yollar sağlar. WS-Güvenliğinde tanımlanan uzantılarını kullanarak, uygulamaları Web hizmetleri çerçevesinde çalışmak üzere tasarlanmış güvenli iletişim kurulabilir[44].

*AON(Uygulama Odalı Ağ):* SOA mimarisinin daha ileri ve özellikle internet için versiyonu olan AON yeni nesil bir teknolojidir. AON sadece IP cihazlarının IP paket başlıklarına değil aynı zamanda yüklerine müdahale edebilir. AON, modern yazılım ve donanım teknolojileri ile oluşturulmuştur. AON yönlendiriciler, gömülü uygulama zekâ ile internet tasarımı yeniden gözlem için bir şans sağlar[45].

*Web Servis Güvenlik Politikası:* Web servis politikası, Web hizmeti tüketiciler ve Web servis sağlayıcılar arasında birlikte çalışabilirliği sağlamak için meta tanımlayan bir özelliktir. Web servis politika şartnamesi Web hizmeti yönetimin somut bir örneğini oluşturmak için servis yönetişim modellerini otomatikleştirmektedir[46].

*Plana Göre Yerleştirme Mimarisi:* Bir dağıtım mimari bir fiziksel çevre için mantıklı bir mimari haritalama göstermektedir. Fiziksel çevre; bir internet ortamında, CPU, bellek, depolama aygıtları ve diğer donanım ve network cihazları bilgisayar düğümleri içerir. Dağıtım mimarisini tasarlarken teknik gereksinimler aşamasında belirtilen sistem gereksinimlerini karşılamak için gerekli fiziksel kaynakları belirlemek için dağıtım boyutlandırma içerir[47]. Bir dağıtım mimari tasarım tamamlandıktan sonra dağıtım gerçek maliyet proje onayı sırasında değerlendirilir.




Şekil 4. 5: Bir bulut operatörü tipik Bulut Platformu için uygun dağıtım mimarisini belirlemek için adımlar(*Plana Göre Yerleştirme Mimarisi örneği)*

*XML Saldırıya Maruz Kalma Durumu Yönetimi:* Güvenlik yönetimi, özellikle işletim sistemleri, firmware ve uygulamaları tanımlanması, sınıflandırılması, iyileştirilmesiyle ve güvenlik açıklarını azaltmaya döngüsel uygulama olarak sınıflandırılır[48]. Güvenlik açıklarını keşfetmek için birçok yollarından biri güvenlik açığı tarayıcı istihdam etmektir. Bu duyarlı güvenlik açıklarını keşfetmek için birden fazla yollarla hedef varlığın analiz bir güvenlik için bir tarayıcı yazılımıdır.

*OpenSAML:* OpenSAML-Java üreten ve SAML mesajları tüketen, oluşturma ve dijital olarak imzalanmış ve şifrelenmiş içeriği değerlendirilmesi ve SAML bağlamaları ile çalışmak için destek sağlayan Java ile yazılmış bir düşük düzey bir kütüphanedir. SAML meta alıcı için kapsamlı destek de SAML mesajlarının tüketiminin çevrede güvenlik politikalarının oluşturulması için bir API ile birlikte sağlanır[49]. Bu kütüphane SAML kimlik sağlayıcıları, servis sağlayıcıları ve gelişmiş istemcilerin belirli türde yazmaya ihtiyaç duyan kişiler için tasarlanmıştır.

# BÖLÜM 5

# SUNUCU KONFİGÜRASYONU

Bu bölümde Anroid işletim sistemine uygun programlar oluşturulurken genelde kullanılan Java programlama dilinde notasyon kavramına, HTTP için nesne sıralama, dağıtıcı servlet ve spring önyükleme konularına değinilmiştir.

## 5.1 Java Programlama Dilinde Notasyonlar

Java'da kullanılan Spring'i geniş açıda incelemeden önce notasyonlardan bahsedilmiştir. Çünkü notasyonlar anlaşılmadan Spring'in altyapısının anlaşılması çok zordur.

Notasyon, Java kaynak koduna eklenebilen sözdizimsel metadata formatıdır. Sınıflar, metotlar, değişkenler, parametreler ve paketler notasyonla ilişkilendirilebilir[54].

Notasyonların avantajlarından biri geliştiricilerin metotların özelliklerini eklerken belirli bir şemada birleşmesini sağlaması ve ifadelerin manalarının bilinmesidir. Her notasyon "@" işareti ile başlar ve tanımlayıcısı ifadesi ile devam eder. Java derleyicisi notasyonları gördüğü zaman gerekli işlemleri yaparak notasyonlanan yapının özelleşmesini sağlar. Güvenlik tarafından notasyonların önemi ilgi parçalara güvenlik tedbirlerinin eklenebilmesidir. Örneğin bir metota kimlik doğrulama bilgisi eklenebilir.

Notasyonların kullanım amacı üç çeşittir[58]:

- Derleyiciyi bilgilendirmek: Açıklamalar hataları tespit ve uyarıları bastırmak için derleyici tarafından kullanılabilir.
- Derleme zamanı ve dağıtım zamanı işleme: Yazılım araçları, notasyon bilgilerini kod, XML dosyaları gibi formatlı bilgilerden çıkarır.
- Çalışma zamanı işleme: Bazı notasyonlar çalışma zamanında kullanılır.

Son olarak örnek olması açısından @secured (güvenceli) notasyonu incelenmiştir.

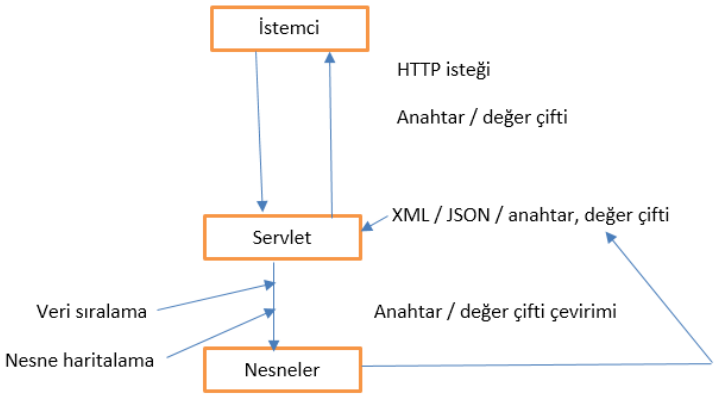
@secured notasyonu, hizmet katmanı güvenlik özelliklerini tanımlamak için kullanılan Java 5 notasyonudur. Güvenceli açıklama iş yöntemleri için güvenlik yapılandırması

özniteliklerin listesini tanımlamak için kullanılır. Güvenlik yapılandırması özniteliklerin listesini döndürür (kullanıcı rolü, yönetici rolü gibi)[67].

## 5.2 HTTP İçin Nesne Sıralama

Bilgisayar biliminde sıralama, veri depolama ve taşımanın uygun formatının hafıza gösterimine taşıma sürecidir. Bilgisayar programların parçalarının birbiri arasında veya programlar arasında verinin taşınması gerektiğinde kullanılır. Sıralama serileştirmeye benzer ve ikisi de uzaktaki nesnelerle iletişim için kullanılır. Sıralama kullanılarak, ilkel iletişim için özel / karmaşık nesneleri kullanarak, karmaşık iletişim kolaylaştırır. Sıralamanın tersi demarshalling olup seri kaldırmaya benzer.

Sıralama, thread (bir programın paralel ve birbirinden bağımsız işlemler yapmasını sağlayan yapılar) ve işlemler arasında veri taşımanın gerekli olduğu durumlarda farklı uygulamaya geçiş uzaktan yordam çağrısında kullanılır. Microsoft Bileşen Nesne Modelleme (COM)'de kullanılan ara yüz göstergeleri COM 'un alt bölümlerine gelmeden önce sıralanması ve .Net'te ayarlanmamış tip ve CLR tipi arasındaki dönüşümde kullanılması kullanım alanlarına örnek verilebilir[68]. Sıralama, komut ve Mozilla uygulama çerçevesinde sağlanan XPCOM teknolojilerini kullanan uygulamalar içinde yaygın bir biçimde kullanılmaktadır.




Şekil 5. 1 HTTP için nesne sıralama şeması

### 5.2.1 Serileştirme ve Sıralama Farkı

Pyhton programlama dilinde aralarında farkı olmayan serileştirme ve sıralama kavramlarının Java dilinde karşılıkları farklıdır[62].

Sıralama kendi durumunu kaydeden kendi örneği yaratıldığında sıralı olduğu bilgisini örneğine ileten bir nesnedir. Uzak ve ya serileştirilmiş her şey sıralanabilir. Sıralama serileştirmeye benzer fakat sıralama kod tabanlı olarak kaydedilir. Aralarındaki fark en çok uzak nesne işlemek üzerinedir.

Sonuç olarak serileştirme bir nesne olup anlamı kendi durumunu bit akışı içinde tekrardan nesne olabilecek şekle dönüşmektir.

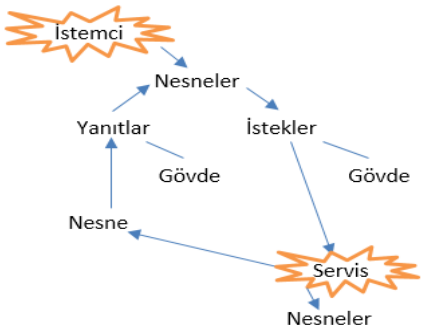
## 5.3 JSON'a Giriş

JSON (JavaScript Nesne Notasyonu) hafif veri değişim formatı olup, insanlar tarafından okunması ve makineler tarafından işlenmesi kolaydır. JSON programlama dilinden bağımsız bir metin formatı olup herhangi bir programlama dilinde kullanılabilmesi için önceden tanımlanması gerekir[63].

JSON iki ilke üzerinde inşa edilir ki:

İsim değer çifti olarak koleksiyon: Çeşitli dillerde bu bir nesne, kayıt, yapı, sözlük, karma tablo, Kama listesi veya ilişkisel dizi olarak gerçekleşmiştir.

İstenilen değerler listesi: Çoğu dilde bu bir dizi, vektör, listede, ya dizisi olarak gerçekleşmiştir.




Şekil 5. 2 JSON iletim

JSON, sadece veri serileştirme biçimi olmasına rağmen, JavaScriptin betik dili olmayan bir alt kümesi olarak çeşitli güvenlik kaygıları oluşturur. Bu endişelerin nedeni JSON metni yürütmek için gömülü JavaScript eklentisi kullanılmasıdır[64].

```
{"employees":[
  {"firstName":"John", "lastName":"Doe"},
  {"firstName":"Anna", "lastName":"Smith"},
  {"firstName":"Peter", "lastName":"Jones"}
]}
```

```
<employees>
  <employee>
    <firstName>John</firstName> <lastName>Doe</lastName>
  </employee>
  <employee>
    <firstName>Anna</firstName> <lastName>Smith</lastName>
  </employee>
  <employee>
    <firstName>Peter</firstName> <lastName>Jones</lastName>
  </employee>
</employees>
```

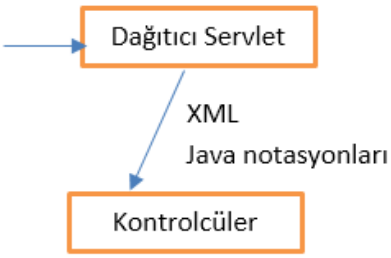
Şekil 5. 3 Aynı veri setinin JSON ve XML formatlı gösterimi [65]

## 5.4 Dağıtıcı Servlet

Dağıtıcı servlet, HTTP isteği işleyicileri / denetleyicileri için merkez dağıtıcısıdır. Örnek olarak UI kontrolleri ve HTTP bazlı uzak servis keşfedicileri verilebilir. Uygun haritalama ve istisna yönetimi faaliyetlerini sağlamak için web isteği işleyicilerin isteğini yürütmek için dağıtılır. Bu servlet çok esnektir. Uygun uyarlayıcı sınıfının yüklenmesinden sonra herhangi bir iş akışında kullanılabilir.

JavaBeans konfigürasyonu bazlıdır. Uygulamanın bir parçası olarak işleyici nesnelerinin isteklerini yönetmeden önce işleyici haritalama eklentisini kullanabilir. İçerdiği çoklu çözümleyici eklentisi sayesinde isteği çoklu parçaya bölerek çözümleyebilir.

Bir web uygulamasında istenilen sayıda dağıtıcı servlet tanımlanabilir. Fakat her servletin kendi isim uzayı olması, kendi uygulamasının haritalama ve işleyici bağlamını yüklemesi gerekir[72].




Şekil 5. 4 Dağıtıcı Servlet Kullanım Şeması ve Çok Kısımlı Veri İşleme

Spring'de çok kısımlı veri işleme için çok kısımlı çözümleyici (MultiPartResolver) ara yüzü kullanılır. Bu arayüz ilk olarak RFC 1867 dokümanında tanımlanmış olup, çok parçalı dosya yüklemek için kullanılır. Uygulama bağlamında veya tek başına bu ara yüzü kullanan sınıflarda kullanılabilir.

Spring'de dağıtıcı servlet varsayılan olarak kullanılmaz ama çok kısımlı istekleri bölümlemek için kullanılır. Tanımlanabilmesi için dağıtıcı servletin uygulama bağlamında çok kısımlı çözümleyicisinin tanımlayıcısı ile oluşturulması gerekir. Bu sayede dağıtıcı servlete gelen ve işlenilmek istenen bütün istekler ilk başta bölümlenir.

Eğer dağıtıcı servlet çok kısımlı isteği bulursa çok kısımlı çözümleyicide tanımlandığı gibi onu çözümler ve HTTP servlet isteğine bölümlenmiş biçimde gönderir.

Web uygulamalarında Spring'in kendi MVC Kütüphanesi kullanılmadığı zaman çok kısımlı çözümleyicinin kullanılması için genellikler web.xml dosyası kullanılır[67].

## 5.5 Spring Önyükleme

Spring önyükleme, tek başına çalışabilen ve Spring bazlı uygulamaların sadece çalıştır komutu verilerek çalıştırılabildiği bir ortamdır. Çoğu uygulamalar çok az Spring konfigürasyonu ile çalışabilir.

Spring önyüklemeyi java-jar veya daha klasik jar plana göre yerleştirme (deployment)lar ile kullanılabileceği gibi komut satırı araçları ile kullanılabilir.

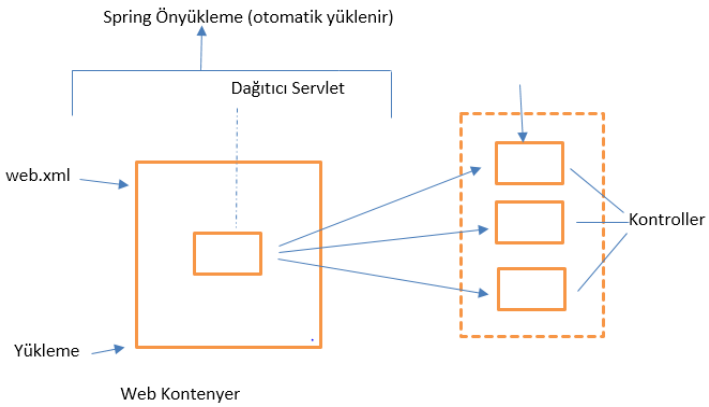
Spring önyükleme esasen dört adımdan oluşur. Bunlar:

1.Web konteyner yükleme,

2.Kontrolleri keşfetme,

3.Dağıtıcıyı yükleme,

4.Veri tabanına vs. bağlanmadır.




Şekil 5. 5 Spring Önyükleme altyapısı

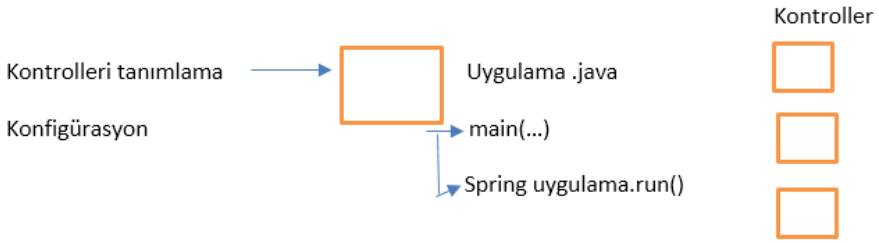
Faydaları şöyledir:

Tüm Spring gelişimi için daha hızlı ve yaygın olarak erişilebilir başlangıç deneyimi sağlanır.

Spring önyükleme kullanılarak oluşturulan uygulamaların yeni isteklere uygun olarak değiştirilmeleri kolay ve hızlıdır.

Projelerin büyük sınıfları (örneğin gömülü sunucular, güvenlik, ölçümler, sağlık kontrolleri, dışa yapılandırma) için fonksiyonel olmayan bir dizi özellik sağlanır.

XML konfigürasyonu için kod ve isteğe gerek yoktur[68].




Şekil 5. 6 Spring Önyükleme çalışma mantığı

# BÖLÜM 6

# SUNUCULARIN BULUT İLE ETKİLEŞİMİ

Bu bölümde Güçlendirme mekanizmasına, Spring'de Bağımlılık Enjeksiyonu Ve Otomatik Bağlantı, Spring Konfigürasyon Notasyonları ile Java Devam Etme API'si konularına değinilmiştir.

## 6.1 Retrofitting (Güçlendirme) - Android Ve Java Programlama Dili İçin Güvenli Tip REST İstemcisi

Şimdiye kadar sunucu taraflı mobil cihazlar için bulut bazlı servislerin inşasından bahsedildi. Fakat mobil cihazların bulut servisleriyle nasıl iletişime geçtiği konusuna değinilmedi. Bu bölümde bulut servislerinin Android tarafı ile iletişimi incelenecektir.

Bu iş için açık kaynak kodlu olan Retrofit kütüphanesinden faydalanılacaktır. Bu kütüphane Java'da bulut bazlı servisler için kuvvetli bir ara yüz sağlar. Ayrıca Retfofit, ara yüzle ve uygun HTTP isteği ile iletişime geçecek parametrelerin dönüşümleri ile ilgilenir. Servise gönderilen istekler, yanıt döndürürken kendini Java türüne dönüştürür. Bu yüzden güçlendirme birçok karışık dönüşüm ve kural çekimini servlette yapmasının yayında bunu istemci tarafında yapar.

Güçlendirmenin neye benzediği ve iletişimin nasıl yapıldığı şu şekilde açıklanabilir. İlk başta iletişime geçecek servisin gösterilebilmesi için Java'da bir arayüz tanımlanır. Daha sonra bulutta bulunacak ve serviste çağrılacak metotlar tanımlanır. Tanımlanan servlet bu metotları gerçekleştirecek ve yürütecektir. Tanımlanan metotlar belirlenen işi yapmalı ve değişken türleri Java'daki türler olmalıdır.

Örneğin kullanıcıların üye olduğu ve üyelerin listelendiği bir uygulama düşünelim. İlk olarak kulacının bilgilerini alan ve kaydeden bir kaydetme metodu ve kayıtlı kullanıcıların olduğu ve verilen URL'den ilgili kişiyi arayan listele metodu olsun. Bunun için ilk olarak bu servisler iletişime geçecek bir Java ara yüzü hazırlanır. Bu uygulamada kuvvetli bir arayüz ve servis ile iletişim için kuvvetli bir Java tip dönüştürücünün olması gerekir.

Şimdi ise HTTP'den sunucuya gönderilecek servis ara yüzünde tanımlanmış listele metodundan istekler alınır ve HTTP isteği olarak gönderilir, cevaplar ise istemcilerin bunu anlamlandırabilmesi için karakter katarı veri türüne dönüştürülür.

Bu arayüz otomatik olarak bilgiyi HTTP isteğine dönüştürür. Bunu servlete gönderir ve cevabı alır. Güçlendirme kütüphanesinin burada arka tarafta bulunan ara yüzü otomatik olarak ara yüzün gerçekleştirildiği nesneye dönüştürülür. Esasında güçlendirme kütüphanesi otomatik olarak her şeyi istenilen türe çevirebilir.

Listele metodu çağrıldığında uygun HTTP isteği otomatik olarak servise gönderilir. Metot çağrısı servletteki HTTP isteği olarak çevrilmek istenirse güçlendirme kütüphanesinin kullanılması gerekir. Bu açıklayıcılar ile aynı yolla olur. Açıklayıcı ile bu yol istek bildirimi ile ve servletin kayıt tanımlayıcı yolu ile oluyordu. Herhangi bir istemci güçlendirme kütüphanesinin haberdar olduğu listele fonksiyonunu çağırarak bir istek gönderir. Bu istek ilgili HTTP metoduna kayıt belirleyici yoluyla belirlenir.

İstek belirlenen yoldan alınır ve geri dönen mesaj karakter katarına döndürülerek geri alınır. Kaydetme metodunda işler biraz farklı olacaktır. Servlete parametreler gönderilecektir. @post notasyonu ile servletin dinlediği kayıt yolun üzerinden yapılır. @formURLEncoded (form URL Çözümleme) notasyonu ile iletim isteğinde URL çözümlemesinde kullanılarak çözümlenecek ve web sayfasındaki forma bakılacaktır. @field (Alan) ile HTTP isteğinde ki parametre haritasından parametrelere döndürülür. Bunu için @fieldURL (Alan URL) ve @fieldDuration (Alan Uzunluğu) notasyonu eklenir.

Esas olarak sadece metot açıklanmasının yanında bunu metot çağrısında nasıl çevrileceğini bildiren notasyonu kullanılır. Bunun içine kaydetme metodu çağrıldığı zaman, güçlendirme fonksiyonu istek iletimi olarak setto(ayarlama) "/" kayıt edilen olarak kabul edilir. Gövde, URL çözümleme vücudu olarak olması gerekir. Sonra her bir

parametre, URL çözümlemenin gövdesine eklenmesi gerekir. Bunun için değer parametreye eklenir. URL'in ise URL parametresi olarak eklenmesi gerekir. İlgili parametreler ilgili değişkenlere eklenir. Güçlendirme kütüphanesinin en önemli avantajından birisi bu işleri otomatik olarak yapmasıdır. Yapılacak işler ara yüzden otomatik olarak HTTP isteklerini yapacak olan nesneler inşa edilir.

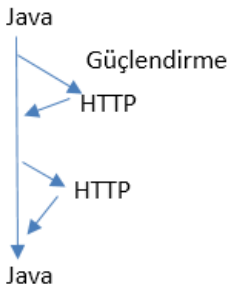
Ara yüzden gerçekleştirilen ve gerçek HTTP isteklerinden metot çağrılarına çevirmede kullanılan nesnelerin oluşturulma dair mekanizması şöyledir: Bu işlem için RestAdapter (Rest Uyarlayıcı) güçlendirmesi yaratılır. Bu güçlendirme kütüphanesinde akıcı bir program ara yüzü bulunur. Oluşturmak için "new RestAdapter.build" ifadesi kullanılır. Sunucunun setEndPoint (varış noktası düzenleme) ile çağrılacağı yer söylenir. Sonra inşası bildirilerek dolaylı olarak RestAdap'tının belirli varış noktası inşa edilir.

Uyarlayıcı oluşturulduğu zaman, servisin ara yüzünün gerçekleştirdiği nesne yaratılır ve servis adaptor.create (uyarlayıcı yaratımı) sayesinde eşitlenir. Arka planda güçlendirme kütüphanesi nesnelerde ki çağrıları ve HTTP isteklerinde belirli arayüz metotlarını otomatik olarak çevirir.

Güçlendirme kütüphanesi çoğu işi basit bir metot çağrısıyla otomatik olarak istemcinin yerine yapar. HTTP isteği döndürülür ve ağa gönderir. Cevap geri geldiği zaman cevabın gövdesi çıkarılır ve kelime katarına dönüştürülür ve istemciye sonuç döndürülür. Benzer şekilde ara yüzdeki diğer metotlar kayıt ekleme için çağrılabilir. URL ve kayıt niteleyicisi gönderilebilir. Güçlendirme kütüphanesi de bu metodu görür.

@field notasyonu ile parametreler eklenir ve HTTP isteğine çevrilerek döndürülür. URL, isteğin gövdesinde bu parametreler çözümlenir ve sunucuya yanıt gönderilir.

Sonuç olarak güçlendirme kütüphanesi iletişim için istek göndermede ve diğer teknolojilerin bulut servislerini kullanması işinde kullanılır. Bu tip kütüphaneler, Java ortamındaki istemcilere çok yardım eder. İstemcinin kendisi dönüşümleri yapmak istemediğinde oldukça kullanışlıdır. Güçlendirme kütüphaneleri arka planda HTTP üzerinden metotları çağırır.




Şekil 6. 1 Güçlendirme ile HTTTP ilişkisi

### 6.2 Spring'de Bağımlılık Enjeksiyonu Ve Otomatik Bağlantı

Kontroller Spring tarafından otomatik olarak fark edilip yaratılır. Kontrollerdeki açıklayıcılara bakılıp, kontrolün haritasındaki istekler belirlenir. Fakat kontroller tanımlanan nesnelere dayanır. İstemciler muhtemelen kontrolcülere her şeyi bildirmek yerine bazı şeylerin kontrolcülerin otomatik yapmasını isterler. Bunu farklı yolları mevcuttur.

Tasarlanan arayüz herhangi bir kayıt sistemi üzerinde işlem yapabilir. Bazen de kayıt sisteminin kullanılan tipini her işlemde değiştirilmek istenebilir. Bu yapılırken kodun kendini düzenleyip yeni versiyonu çalıştırılması istenir.

Uygulama özel yapılandırma ile başlatıldığı zaman otomatik olarak parametre eştirilir ve uygun durum koşulu için uygun kayıt sistemi kullanmak için kontrollere katılır (enjeksiyon edilir).

Spring burada bahsedilenleri bağımlılık katılması olarak adlandırılan terim ile yapar. Bağımlılık katılması ise özel sınıfa bağlı olarak nesne tanımlar. Bu yüzden servis kayıt sisteminin nesnesine bağlı olarak oluşturulur. Bu kayıt sistemi önceden oluşturulan servise bağlıdır.

Bağımlılık katılmasındaki bağımlılık kavramı ile nesne otomatik olarak sistem ara yüzü sayesinde inşası yapılamadan oluşturulur. Bahsedilenin yapılması için AutoWired (otomatik kaplama) olarak adlandırılan notasyonu hafıza değişkenine eklenir. Spring'de kayıt sistemi nesnesi otomatik olarak bağlanmalı veya değişkene eklenmelidir. Spring'in burada yapacağı ona sağlanan konfigürasyona bakarak, sistemi bu yapılandırma içinde geliştirim çalışmasını bulmaktır. Ayrıca nesnenin bir örneğini yaratılarak veya var olan gerçekleştirime yeniden gerçekleştirerek veya yerleştirerek sisteme yerleştirir veya üye değere ekler. Sisteme otomatik olarak bakarak gereklilikleri bulur. Konfigürasyonda

önceden tanımlanmış implementasyonları bulur ve sonra konfigürasyonunda tanımlanmış değeri üye değer atar.

Servisin herhangi sayıdaki diğer nesneye bağlı olarak yaratılabilmesi güzel bir özelliktir. Örneğin, istemcinin bildiği başka bir bağımsız objenin ortamda bulunduğunu ve kullanıcı yönetiminin ara yüzdeki başka bir yerde tanımlandığını düşünülün. @AutoWired notasyonu kullanılsın. Spring sayesinde her bağımlılık için otomatik bağlantı mekanizması otomatik olarak konfigürasyona bakarak uygun arayüz tarafından gerçekleştirilmiş nesneyi bulur. Ayrıca nesneye ulaşılarak hafıza değişkeni değerine eklenir.

Uygulamanın inşası için bağımlılık katılması ile hiyerarşik nesneler otomatik olarak yaratılıp, konfigürasyonları bu hiyerarşiye göre yapılır. Spring'deki farklı yapılandırma nesnelerine bağlı olarak yapılır. Spring'de özel ara yüzlerin haritalaması için nesneler farklı otomatik kaplama çoğalmaları vardır. Sınıflıların yapılandırma nesnesinin otomatik kapsamada olması beklenir.

Sping'de farklı arayüz değişkenlerinin uyumu için haritalamanın tanımlandığı sınıf yaratılabilir. Kontrolcüleri, diğer nesneleri ve ara yüzün kullanılabilmesi için bu sınıfın özel olarak tanımlanması gerekir. Örnek olarak bir sınıfın yaratıldığı ve @Configuration(yapılandırma) notasyonu ile açıkladığını düşünelim. Spring ve bağımlılık katılması tarafından bu sınıfa bakılır ve örneği inşa edilerek otomatik kaplama bağımlılıklarına bakmak için haritalamaya bakılır. Sonra depolama sistemini doldurmak için genel bir metot eklenir. Örneğimizde bu metodun depolama istemi ile çağrıldığını ve yeni yerel depolama ile bir örneğinin oluşturulduğun düşünülsün. Bu yapılandırma dosyası ile depolama sistemi adlı bir metot oluşturulsun. Bu metot çağrıldığı zaman yerel depolamadaki gerçek implementasyonu istemciye otomatik kaplama depolama sisteminde kullanılması için verir. Bu işlerin yapılabilmesi için @Beam notasyonunun tanımlanması gerekir. Bu yüzden çalışma zamanında objenin yeni örneği tanımlanır. @Beam ile metotlar açıklayıcılar sonra değer döndürülmesi ile arayüz tanımlanması ile uygun sınıf implementasyonunda kullanılır. Önceki örneğimizde de bu notasyonu kullanılabilir. Bu sayesinde yerel depolamanın örneği oluşturulur.

Kayıt sistemi türünde üye değeri tanımlanarak otomatik bağlantı notasyonu sayesinde Spring otomatik olarak değeri alır, metoda getirir ve uygun hafıza değeri ile ayarlanır.

Esas olarak otomatik bağlantı notasyonuna sahip çeşitli bağımlılık ve diğer objelere göre notasyonu tanımlanması gerekir. Arayüz sayesinde farklı implementasyonların çeşitli yerel depolama için olduğu gibi bağımlılıklar tanımlanır. Sonra belirli bir uygulamanın tanımlanması için yapılandırma notasyonu tanımlanır. Bu açıklayıcının amacı depolama siteminin farklı implementasyonu için haritalama yapmaktır. @Beam notasyonu ile gerçek implementayonlar tanımlanır ve yaratılır[69].

Bu bölümde yapılmak istenenler ilk başta fazla karmaşık gelebilir. İlk başta yapılmak istenenler uygun adımlara bölünmesi gerekir. Bağımlılık katılması mekanizmasının mantığı anlaşılmasının ardından, Spring'in yeninden kullanılabilir ve daha modüler bir yapıya olanak sağlaması artısı düşünüldüğünde faydalarının fazla olduğu ortaya çıkar. Çalışma zamanında Spring servisi @AutoWired'i görmek için bakar ve okur. Gerekli tipi bildirir. Otomatik kaplanacak üye değişkeninin tipine bakılır. Sonra yapılandırma objesine bakılarak @Beam ile ilişkilendirilmiş notasyon olup olmadığına bakılıp, istenilen uygun tür bildirilir. Eğer bu çağrı istenirse üye değişkeninde eklenmiş değer döndürülür[70].

Spring'de varsayılan olarak bu metot bir kez çağrılır. Lokal depolama metodu çağrılarak bir kopyası yaratılır ve her benzer kaynakla ilgili olarak otomatik bağlantı notasyonu sahipliği ile kullanılmak istenildiğinde benzer nesne referansı tam olarak kullanılır. Bu varsayılan davranıştır. Otomatik bağlantı notasyonu her görüldüğü zaman bu metot çağrılır ve eklenmesinin uygun olduğu zaman yeni nesne örneği oluşturulur. Fakat varsayılan olarak her tekil yerde kayıt sisteminin kullanılması için kayıt sisteminin referansına ulaşılmak istenir. Farklı sınıf dallarının eklenip tekil nesne kullanılması için tekil tasarım deseni kullanılır. Gerçekte gerekli olan şeyler için karmaşık mekanizmaların ortaya çıkarılmasına gerek olur. Spring, otomatik bağlantı notasyonu bazlı bağımlılıklar otomatik olarak ortaya çıkarılır ve konfigürasyona bakılır ve @Beam notasyonu ile tanımlanmış metodun dönüş değeri eşleştirilir. Sonra nesne geri döndürülür ve saklama sistemi referansı bir yerde alır ve bunu üzerinde otomatik bağlantı notasyonu bulunur. Bu Spring'de kuvvetli bir mekanizmadır[71].

### 6.3 Spring Konfigürasyon Notasyonları

Uygulama sınıfının konfigürasyona sahip olduğunu düşünüldüğünde @configuration (yapılandırma) notasyonunun kullanıldığı görülür. Tanımlanan her bir konusal metot @ ile başlayan açıklayıcı ile tanımlanır ve belirli bileşsem ile inşa edilir. Ara yüzün bir dönüş tipi bulunur.

Uygulamanın ara yüzünde @AutoWired gibi notasyonu bulunduğu düşünülsün. Ara yüzün parametreler bulunup ara yüzle ilgili olarak otomatik olarak ilişkilendirilir. @AutWierd'in parametresi bulunup ve yerleştirilir. Varsayılan olarak taranılan paketlerin birer haritalamasının olduğu varsayılacaktır. Bu yüzden eğer paket taranıp ara yüzün çoklu gerçekleştirim olduğu fark edilirse bu otomatik kaplama yapılmak istenildiğinde hangi özel nesne kullanılacağına otomatik olarak karar verilemez.

Tam olarak ara yüzün bir gerçekleştirme veya gerçekleştirimlerinin paketlerine bölümlendirilmesi yerine her ara yüze gerçekleştirim yapılarak Spring'in otomatik olarak hangisini seçeceği ve gidileceği bildirilir. Açıklayıcılar hakkında bilinmesi gereken diğer bir şey de Spring'de @Aoutowired notasyonuna bakılır ve doldurulur. Eğer konfigürasyona belirli açıklayıcı eklenmezse Spring @Autowired notasyonuna bakmaz [72].

Konfigürasyona eklenebilecek diğer bir notasyonda @EnableAoutConfiguration (Atomatik Konfigürasyonu Yapılabilirliği)'dir. Bu notasyon Spring'de @Autowierd notasyonunu arar. Bulunulduğu zaman uygun değer ile doldurulur. @EnableOutConfiguration ve @ComponentScan'nın birlikte kullanımı yaygındır. Bu sayede belirli gerçekleştirim seti bazı paketlerde tanımlanmış olup otomatik olarak konfigürasyonu @EnableAoutConfiguration ile Spring'de @AutoEierd notasyonuna bakar ve doldurur. Konfigürasyon sınıfı basit bir biçimde iki şekilde doldurulabilir. Çünkü bir beanın(JSP 'de sayfalar arası bilgi taşıma işleminde kolaylık sağlayan basit *java* sınıflarıdır) @bean ile tanımlanmasına gerek yoktur ve bir metot uygulamada ki her bir tekil bileşen için bu bean döndürür.

Otomatik kaplama Spring'de otomatik olarak istemci için tanımlanır. Spring'e izin verilerek onların tanımlanması ve kaplanması işi otomatik olarak yapılır. Bütün iş

yapılmadan önce her biri bireysel olarak tanımlanır. Bunun altyapısında paketlerde ve alt paketlerde bütün sınıflar otomatik olarak taranır [73].

Örneğimizde ki com.mobile kullanıldığı zaman com.mobile.data ve com.mobile.http paket olsun ve bunlarla birlikte alt paketler taransın.

Alt paketler düşünüldüğünde taranacak dosya sayısının çok olacağı bellidir ve bu zaman alır. Otomatik kaplama örneklendirilerek her şeyin konfigürasyonunu yapar. Fakat bu anca uygulama başlatıldığı zaman yapılabilir. Çoğu zaman olduğu gibi uygulamanın başlatıldığı zamanki ortam ne kadar kullanılırsa bu sorun olmayacaktır.

Bulut servisinin sağlayıcı ile yeterli isteklerin döndürülmediği zaman bulut ortamının alt seti uygulamayı kapatmaz. Bunu bir örneği Google App (Uygulama) Motor'udur. Bu koşullar performansı etkiler. Fakat çoğu zaman bu koşullar performansı etkilemez [74].

### 6.4 Spring Konfigürasyon Notasyonları

Bulut servisleri yüksek boyutlu verilere erişebilir. Amaç çoklu bilgilere izin vermek (mobil cihazlarla da olabilir) ve mobil istemcilerine bu bilgileri döndürmektir.

Bulut servislerinin mobil cihazların inşasında dikkat edilen noktalardan biride istemci için depo kaynaklarının ve saklanan verilerin kullanımında ki avantajdır. Avantajı sonra aranılacak veri ile özel istemcinin isteği onlara döndürülebilir. Burada ki amaç ile HTTP üzerinden nesnel gönderiminde ki mekanizmadır. Birlikte çalışılacak istemci veya sunucu taraflı nesnelere sahip olunabilir ve bunlar HTTP'nin mesaj gövdesine doğru dizilebilir. Diğer yanda veriler sıralaması bozularak nesneye döndürülür. Ayrıca bütün işlemleri yapmak için nesneler alınır ve HTTP mesajları bulut bazlı servislere gönderilir ve sonra sunucu tarafında nesnelerin gövdelerine çevrilir.

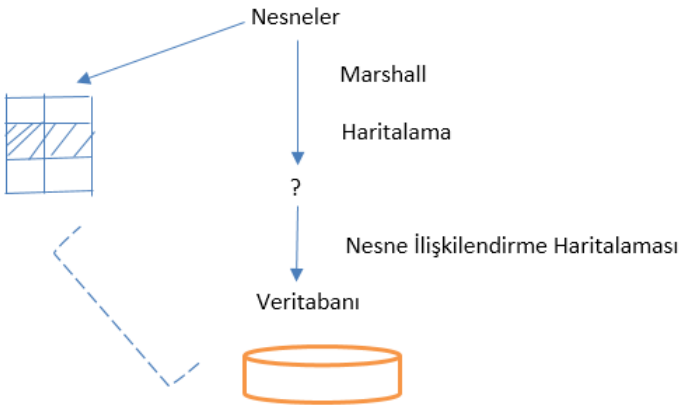
Jackson Kütüphanesi gibi çeşitli kütüphaneler HTTP gövdesine konulmak için JSON'a nesne dönüşümünün otomatik olmasının çeşitli yolları bulunmaktadır. Ayrıca JSON bunu alarak nesneye dönüştürür. Bu iş için güçlendirmenin kullanımı daha önce incelenmişti. Günümüzde sunucuda veri saklanması için benzer yol kullanılır. Sunucu tarafında nesnelerle çalışılır. İstemci taraflı olduğu gibi nesnelere sahip olunur. Veri tabanında saklanması için nesnelere veri taşınmak istenir ve bu veriler bazı formatlara dönüştürülür. Veri tabanının kullanabileceği birden fazla tip ve tür bulunmaktadır.

Fakat nesnelere veri tabanında yeninden sıralananın yapılaması gerekliliği ve veri tabanı formatına dönüştürülme gerekliliği dezavantajıdır.

Veri tabanlarında ki format genellikle satır ve sütunlarla oluşan formattır. Nesnenin veri tabanındaki formatında bu satır ve sütunlar oluşturur. Sonra bu yapılanla kodlar her nesne ve veri tabanı tablolarında nesnelerin haritalamasında kullanılır.

Örneğin, her nesne için haritalar veri tabanındaki satırlara konulduğu ve sütünler nesnenin özel veya üyelik değişkeninin var olduğu düşünülsün. Üye değişken değerlerinin sütunlara konulması için değişik satırlarda nesne örnekleri yaratılarak haritalandırılır. Böyle bir şey yapıldığında, saklanan her bir nesnenin tablo formatına nasıl dönüştürüleceğinin bilinmesi gerektiğinden, yapılması gereken çok sayıda işlem vardır.

Veri tabanında saklanmak istenen veriye karar verildiğinde bu veriler tablo satır, sütunlarından birinde saklanacak şekle dönüştürülür. Bu iş normalde nesne ilişki haritalama olarak adlandırılır ve son olarak veriler geleneksel ilişkisel veri tabanlarında kullanılır. Bu terin noSQL (SQL olmadan) gibi geleneksel olmayan veri tabanları kullanılırken bile kullanılır. Veri tabanına haritalama nesnelerinin saklanması için bütün bu işler günümüzde de yapılır. Bazı literatürler buna ORM derken bazı literatürlere göre noSQL veri tabanları denir. Java nesneleri işlediğinden verilerin nesnelere dönüştürülmesi gerekir [82].




Şekil 6. 2 Veri tabanı haritalama nesnesi alt yapısı

### 6.5 Java Devam Etme API'si

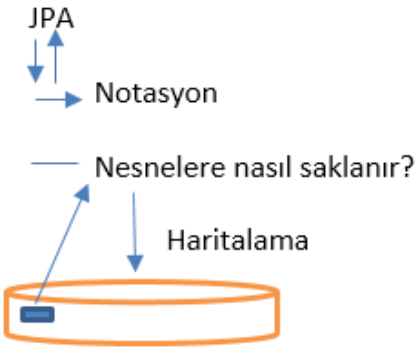
Java'da nesnelerin veri tabanında saklanması ve veri tabanının araştırılıp verinin nesne olarak dönüştürüldüğü yapı bulunmaktadır. Bu araç bir kütüphane olup JPA(Java

Kalıcılık API'si) olarak adlandırılır. JPA'nın arka planında Java açıklayıcıları kullanılarak nesnelerin açıklanması ve JPA'ya veri tabanında nesnelerin saklanması isteği bulunmaktadır. Bu sebeple nesnelere açıklayıcı eklenir. JPA nesne açıklayıcıya otomatik olarak bakar ve nesnelerin veri tabanında ki tablolara dönüştürülmesi için uygun yolları ortaya çıkarır.

JPA bilgileri sıralama veya veri tabanı altındaki objelerin haritalanmasında kullanılır. Nesneler geri döndürülmek istenildiğinde veri tabanından obje dönüşümünde kullanılacak benzer bilgiler kullanılacaktır.

JPA çok iyi bir araç olup haritalamada neredeyse yarı yarıya kod yazılmasını sağlar. Hata yapma ve nesne dönüşümü için düşünülmesine de gerek kalmaz. Gerçekte çok az kod parçası ile veri tabanları bulut servisleriyle JPA sayesinde kullanılır.

JPA'nın gücünden faydalanmak için arka planda çok iş yoktur. Çok basit bazı notasyonları ile nesnelerin veri tabanında saklanması sağlanabilir. Bu bölümde JPA'nın bilgileri nasıl sakladığı işlenecektir.




Şekil 6. 3 JPA işleyicisi

Örnek olarak önceki bölümlerde de örneklendirilen personel bilgileri kayıt ve personel bilgileri listeleyen uygulamayı düşünülsün. Şimdiye kadar mobil servisin kendi mobil istemcileri ile servise kayıt yapıldığı bu bilgilerin sonra objelerde saklandığı, saklanan bilgilerin listede gösterildiği görülmüştü. Şimdi bunu basit bir amaç olduğu anlaşılır. Çünkü servisteki hafıza uygulama çalıştırıldığı zaman ve servis kapatılmadığı sürece o ana kadar saklanan bilgilere nesneler üzerinden erişilebilir. Fakat eğer istenerek veya istenilmeyen bir sebeple bulut servisi durdurulsa kayıtlı bilgilerde kaybolup daha sonraki istemciler bulut servisinin kapatılmasından önceki kayıtlara ulaşamayacaktır.

Bu yüzden nesnelerin örneklerinin oluşturulması gerekilir ve JPA'ya bu bildirilip ileri de bu nesneler kaydedilmelidir. Bu sayede servis kapatılıp yeniden başlatıldığı zaman istemcilerin önceki verilere ulaşabildiği görülecektir. Bunun için basit nesne yaratıldığında veri tabanında bu nesnelerin kullanımlarının basit olduğu görülecektir.

Bütün bu işlerin yapılabilmesi için bir sınıf yaratılıp sonra JPA'nın veri tabanında saklanan sınıf örneğinin gerekliliğini belirten açıklayıcı kullanılıp sınıf yaratılır.

Ardından saklanan nesnelere daha sonra erişmek için onları bir tanımlayıcı ile referans etmektir. Bu sayede normal olarak bazı hafıza adreslerinde nesneler yaratılır. Bu yapılırken fayda olarak tanımlayıcıların eşsiz olup olmadığının düşünülmesine gerek yoktur. Java saklanacak her bir şey için eşsiz hafıza adresi sağlar. Fakat buna benzer şekilde her bir kayıt niteleyicisi sayesinde bulunabilir.

Bahsedilenleri yapmak için, @Id notasyonu ile veya benzersiz bir niteleyicisi ile kayıtlar belirtilir. Bu tanımlayıcıyı JPA'ya bildirmek için tanımlayıcı özelliği kullanılır. Şimdi ID tanımlayıcısı ile herhangi bir hafıza değerine sahip olunabilir. Bu tanımlayıcının türü kelime katarı ise JPA bu eşsiz niteleyiciyi kullanmayı bilir. ID tanımlayıcı ismi zorunlu değildir. Önceki bölümlerde de bahsedilen örnekte herhangi bir alanı tanımlayıcı bildirmek için @Id ile bildirilerek kullanılır.

@Id notasyonu JPA'ya açıklanan özellik veya hafıza değerini nesneye eşsiz tanımlayıcı olduğunu bildirir. Fakat her durumda kullanılabilecek bu eşsiz alanın hangi alan olacağını seçmek zordur.

Tanımlayıcıların eşsiz olduğunu manüel olarak bildirilebilir. Fakat JPA kullanıldığı zaman eşsiz tanımlayıcı yaratmak iş daha kolay olur. JPA sayesinde bu konu üzerinde çok düşünülmez.

İkinci açıklayıcı uzun açıklayıcı olarak eklenebilir. Bu yapılırken uzun açıklayıcıların eklenmesi konusunda çok endişelenmeme gerekir. Anroid'in en büyük geliştirici olan Google'da bu işler kolaydır.

@GeneratedValue (Değer Üretme) notasyonu eşitlik strateji eşitliği için eklenebilir. Fakat bu ekstra açıklayıcı ile JPA bunu için değer doldurur. Bu strateji ile otomatik olarak değer üretilir. Bu açıklayıcı JPA'nın servisin devam edici olmasını sağlar. İsminin imzalanarak sınıf inşası zorunludur. Çoğu zamanda tanımlayıcının bildirilmesi

istenilmez. JPA'nın bu işi otomatik istenmesi gerekir. Bir nesne tanımlanacak olup bu nesnenin getter (elde edici) ve setter (düzenleyici)ı olacaktır.

Değer ismi imzalanıp JPA'da bu objeyi saklaması istenir. Bunun için iki şey yapılacaktır. JPA otomatik olarak bu yaratılan nesneyi alır, veri tabanında ki satır ve sütunlara bu nesne dönüştürülür veya veri tabanı implementasyonuna dönüştürülür. İkinci olarak da yani saklamadan sonra veri tabanının eşsiz tanımlayıcısı belirlenir. Bu eşsiz tanımlayıcı istemci için otomatik olarak belirlenir.

Bu bölümde anlatılan üç notasyonun çoğu zaman ikisi kullanılır.

# BÖLÜM 7

# SPRING GÜVENLİĞİNE GİRİŞ

Bu bölümde Spring Veri REST, oturum yönetimi, Windows tabanlı oturumlar konularına değinilecektir.

## 7.1 Spring Veri REST

Spring Veri REST'i Spring veri projesinin bir parçası olup JPA bazlı depoları RESTful varış noktaları gibi yayımlanmasında kolaylık sağlar.

Spring Veri REST'in önemi düz metinde HTTP REST anlambilimini kullanarak JPA depolama yönetimi varlığı için CRUD operasyonlarının yayımlanmasında temelleri sağlamasıdır. Standart Servlet mimarisi ve Spring MVC kullanılarak yapılan ilk gerçekleştirimde amaç WAR dosyalarının CRUD uygulaması tarafından alan sınıfları ve depolama tanımlarının kullanılarak kolayca gösterilebilmesidir. Gelecek gerçekleştirimlerinin Servlet HTTP ortamını kullanarak depolamak yerine bloklaşmadan giriş çıkış işlemleri ve HTML5'in özelliklerini kullanabilmesi öngörülmektedir.

Özellikleri [76]:

- JPA, MongoDB, Gemfire ve Neo4j depolarını destekler.
- POST kullanılarak yeni varlık yaratımı,
- PUT kullanılarak var olan varlığının güncellenmesi,
- DELETE kullanılarak var olan varlığın silinmesi,
- POST, PUT, DELETE kullanılara varlık ilişkilerinin düzenlenmesi,

- GET kullanılarak servislerin keşfi ve uygun varlıkların listelenmesi, depolama kuyruk metotları kullanarak var olan varlıkların listelenmesi,
- JSR-303 ve Spring Validator beans doğrulayıcılarını içermesi,
- Uygulama olaylarını kullanarak REST geliştiricilerin fonksiyonelliğini arttırmak,
- DSL yapılandırma yardımcısı veya notasyon kullanılara yolun konfigürasyonunu yapma,
- Büyük sonuç setlerini sayfalama,
- Sonuç sıralamadır.

## 7.2 Oturum Yönetimi

Aşağıda görülen çizelgede ki durum yönetiminden sunucu tabanlı durumlardan oturumlar konusu işlenecektir. İstemci tabanlı durumlardan çerezlere önceden değinilmiştir.

Çizelge 7. 1 Windows tabanlı durum yönetimi şeması

| **DURUM YÖNETİMİ** | |
|---|---|
| **SUNUCU TABANLI DURUMLAR** | **İSTEMCİ TABANLI DURUMLAR** |
| Oturum Yönetimi(Session State) | Çerezler |
| Uygulama Yönetimi (Application State) | Viewstate (Görünüm Yönetimi) |

### 7.2.1 Android Tabanlı Oturumlar

Oturum uygulama ile kulacının iletişiminin tekil periyodunu temsil eder. Oturumlar ölçülebilen aktivitenin (ekran görünümleri, olaylar ve kullanışlı konteyneri olarak servis edilir. Varsayılan olarak, Google Analitik oturum süresi otuz dakikadır. Ancak, pek çok geliştirici gibi app arka planda uygulamanın arka plana atılması gibi durumlarda, kendi uygulamasının gerekliliklerini göz önüne olarak bu süreyi uzatabilir veya kısaltabilir.

#### 7.2.1.1 EasyTracker Kullanılarak Otomatik Oturum Yönetimi

EasyTracker geliştirici için yeni oturumlar başlatma çalışmalarını işleyebilir ve otomatik oturum yönetimi sağlar. Burada otomatik oturum yönetimi bakış açısı şöyledir: Varsayılan uygulama 30 saniye oturum zaman aşımı süresi vardır. analytics.xml

dosyasında ga_sessionTimeout (ga oturum zaman aşımı) parametresini değiştirerek zaman aşımı süresini değiştirelir:

Uygulamaya oturum zaman aşımı süresinden daha uzun süre arka planda kalırsa, EasyTracker bayrağı yeni bir oturumun parçası olacaktır[77].

#### 7.2.1.2 Manuel Oturum Yönetimi

EasyTracker kullanılarak otomatik oturum yönetimi tek başına yeterli gibi gözükse de bazı ana durumlarda manuel oturum yönetimi daha iyidir. Örneğin uygulamaya kullanıcı her bağlantılığında yeni oturum açılması isteniyorsa manuel oturum yönetimini kullanmak daha yararlıdır. Bu örnekte bu yöntemin kullanımının esas amacı uygulamanın kullanıcının yeniden uygulamaya bağlandığı zaman değişebilmesi olasılığıdır.

Yeni bir oturum oluşturmak için setStartSession(true) (oturum başaltımı düzenlemesi) çağrılır. Bu oturum yönetim uygulamaların çok değişikliğe uyguladığında önemlidir.

### 7.2.2 Windows Tabanlı Oturumlar

Sadece Sorgu dizelerinde değil, oturum state(durum) değişkenleri de başka bir web formuna veri göndermek için kullanılabilir.

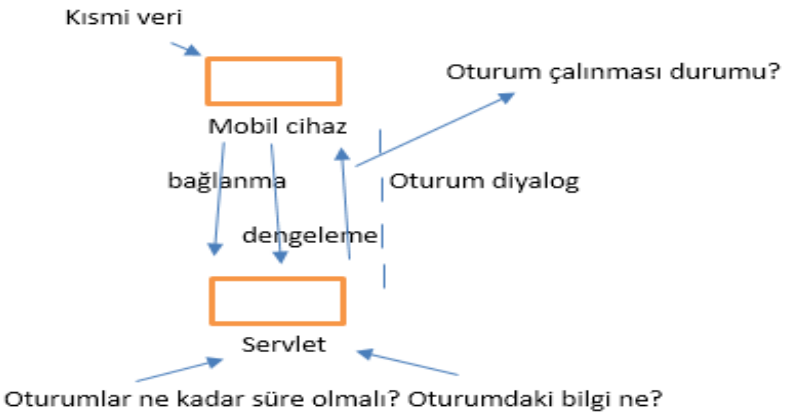
Oturum durumu değişkenleri varsayılan web sunucusunda saklanır ve bir oturumun yaşam süresi boyunca için tutulur. Varsayılan durum modu InProc'tur. Bir oturumun ömrü web.config dosyasında zaman aşımı değeri tarafından belirlenir. Varsayılan süre 20 dakikadır. Zaman aşımı değeri uygulama gereksinimleri ayarlanabilir.

Oturum durumu değişkenleri belirli bir tek oturumda, tüm sayfaların genelinde mevcuttur. Oturum değişkenleri tek kullanıcı küresel veri gibidir.

Uygulama performansı için gerekli değilse, oturum durumu devre dışı geliştirilebilir. Oturum durumu sayfa veya uygulama düzeyinde kapatılabilir.

Sayfa düzeyinde oturum durum açmak için, sayfa yönergesinde ENABLESESSIONSTATE = "False" olarak ayarlanır.

Uygulama düzeyinde oturum devletin açmak için, web.config dosyasında yanlış (FALSE) SessionState modu ayarlanır[84].




Şekil 7. 1 Mobil cihazlarda oturum şeması

### 7.2.3 PKI Yönetimi

PKI ve SSL neredeyse her uygulamayı kapsayan çok geniş bir terim olmasına rağmen, birçok SSL sağlayıcısı hala, yenileyen iptal ve SSL sertifikaları yönetme, yayınlanması üzerinde size daha fazla kontrol veren bir sistemi tanımlamak için Yönetilen PKI kullanır. Güvenilir bir CA'yı kullanarak Yönetilen PKI sisteminin özellikleri genellikle şunlardır:

- SSL sertifikalarının otomatik verilmesi,
- Denetim yetenekleri,
- Tam yaşam döngüsü yönetimi,
- Tüm organizasyon çapında sertifikaların merkezi yönetimidir[86].

## 7.3 Oturum Güvenliği

Bu bölümde HTTP'nin ana kullanım amacı olan oturumlar konusunun güvenliği incelenecektir. İstemcinin sunucu ile iletişim yapmasındaki en önemli zorluklardan biri de çoklu HTTP istekleriyle diyaloğa devam etmesindeki zorluktur. Çünkü istemci sunucuya istek gönderdiğinde sunucunun ilk olarak veya en yakın zamanda ona cevap vermesini ister. Mobil cihaz istemcileri içinde durum aynıdır.

İstemci sunucuya bir istek gönderdiğinde istemci gelen yanıtı alır ortamdan kopar. Sonra yeniden sunucuya istek gönderip yanıtı geri aldığında ve yeninden ortamdan kopmayı istemesi doğal bir durumdur. Fakat bu sefer ikinci isteğin ne zaman gönderilmesinin uygun olacağı sorunu ortaya çıkar. Sorun daha açılırsa sunucu ne

kadar süre için istemci yeninden ortama bağlandığında bağlanan istemcinin aynı istemci olup olmadığını hatırlaması gerekir. IP adresi kullanılırsa bu durum çözülmüş olur. Çünkü aynı IP adresine sahip olan istemci sunucu için aynı istemci olacaktır.

Bazı durumlarda IP kullanılması istenilmez. Bu durumda da çözümü vardır. Fakat bu problem için artık tanımlayıcı türleri yetersiz kalacaktır. Kötü niyetli istemci IP adres veya başka tanımlayıcıları kullanarak sunucuya yanlış veri gönderebilecek veya istemcinin sunucuya göndermek isteği mesajları ortama girip gizleyerek sunucuya gitmesini engelleyebilecektir. Böyle durumda ikinci isteğin aynı istemciden gelip gelmediği sorunu ortaya çıkacaktır.

Problemle ilgili çözümlerden ilki istemcinin sunucudaki ilk isteğine oturum çerezi katmasıdır. Bu durumda mesajın başka istemci tarafından okunmaması için HTTPS protokolü üzerinden gönderilerek mesaj, mesajın yanında bağlanma ve diğer bilgiler olarak sunucuya iletilecektir. Bu çözümde de sunucunun istemcinin kullanıcı ismi ve şifresinin karşı tarafta bilinmesi zorunludur. Bu tanımlayıcılar mesajın gövde veya herhangi bir kısmında olabilir. Mesajın bütünü HTTPS protokolü ile iletileceğinden şifreli olacaktır.

Mesajdaki tanımlayıcılar istemcinin kim olduğunun doğrulanması için yeterlidir. İstemci ile ilgili oturum oluşturulmak istenildiğinde sunucu veri tabanından istemcinin gönderdiği şifreyi kontrol ederek istemciye karar verir. Şifre kontrol edilerek istemcinin gerçekteki istemci olup olmadığı kontrol edilir.

Sunucu geriye çerez göndererek çerezin(az miktarda ki veri) istemcinin göndereceği diğer mesajlarda gerçek istemci olup olmadığını anlar. İkinci istekte sunucu tarafından üretilen çerez istemci tarafından sunucuya verilerek kullanılır. Buradaki çerez, istemcinin hatırlanması için bir parça bilgi olup sunucuya diğer isteklerde bulunurken isteklere eklenir. Sunucu tarafında bu çerez uygun istemci ile eşleştirilir. Sunucu bu çerezin kimin çerezi olduğunu yapılan inceleme sonucunda bulur. Tanımlayıcı veri ile istemci sunucuya kim olduğunu açıklayabilir. Sonraki tüm isteklerde isteğin aynı istemci tarafından gönderildiği anlaşılır.

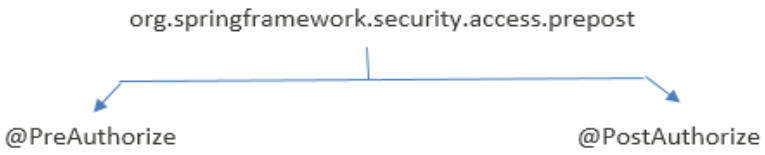
Hep aynı olan istemcinin hakları tutularak yeni istekler istedikçe sunucu tarafından cevaplanır. Çünkü ilk istekte kullanıcı adı ve şifresi bilgisi alınıp, sonra ki isteklerle çerezdeki bilgilerle tanımlama yapılmıştır.

Sunucu çerezleri karıştırmadığı sürece yeni isteği gönderen istemcinin aynı istemci olup olmadığına güvenilir. Bu işlemde HTTPS protokolünün kullanılması oldukça faydalıdır. Bağlantı bilgileri(kullanıcı ismi, şifre) gönderildiğinde bunun korunması gerekir. Buda HTTPS protokolü üzerinden ve şifreli bir biçimde olur. Yanıtların şifreli olmasının bir nedeni oturum çerezinin sunucu ile iletişime geçmesi ve sunucunun özel bir istemci rolüne girebilmesine olanak sağlamasıdır.

Oturum çerezi bir istemci(kötü niyetli) ele geçirirse, artık bu istemci sunucuya göre diğer istemci(çerezi çalınan istemci) olarak görülür. HTTPS'in kullanılmasında ki fayda buradan anlaşılabilir. Bu sayede veriler şifrelenerek gönderildiğinden çalınan çerezde şifreli olur ve kötü niyetli istemci bunu açamadığından sunucuyu kandıramaz. Kısaca iletişimlerde HTTPS kullanılması günümüzde güvenli bir iletişim için zorunludur.

## 7.4 Spring Güvenliği

Spring güvenlik metot açıklayıcıları aşağıdaki şekilde görüldüğü gibi aynı pakette bulunur ve iki çeşittir. Bu açıklayıcılar metot erişim haklarını belirlemek için kullanılır.




Şekil 7. 2 Spring güvenlik metot açıklayıcıları

PreAuthorize(Ön Yetki): Metot çağırımına izinli olunup okunulmadığının kontrolünü yapan metot erişim kontrol açıklayıcısıdır. Türü kelime katarı türü olup, gerekli element değerdir. Spring-EL ifadesi korunmuş yöntemini çağırmadan önce değerlendirilir. Değer "public abstract String value" dir[80].

PostAuthorize(Mesaj Yetki): Metot çağrımından sonra değerlendirilen metot erişim kontrol açıklayıcısıdır. Türü kelime katarı türü olup, gerekli element değerdir. Spring-EL ifadesi korunmuş yöntemini çağırıldıktan sonra değerlendirilir. Değer "public abstract String value" dir[88].

PreAuthorize ile metot içine girmeden önce izin için kontrol edebilirsiniz. PreAuthorize ile rolü temelinde veya yönteme geçirilen argüman üzerinde kontrol edilir. Yöntem yürütülmesinden sonra PostAuthorize açıklayıcısı kullanılır. PostAuthorize rolleri oturum bazında yetkili olabilir, yönteme nesne ve geçit argümanı ile döner.

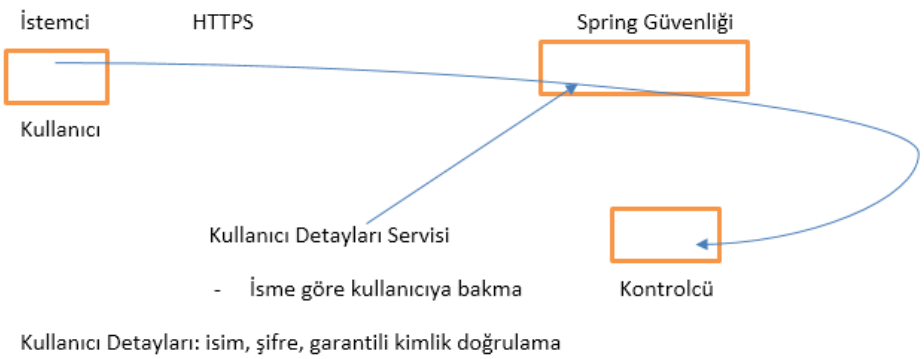
### 7.5 Geleneksel Kullanıcı Detayları Servisi İnşası

**Interface UserDetailsService (Kullanıcı Detayları Ara Yüzü):** Kullanıcıya özel verileri yükleyen çekirdek ara yüzüdür. Kullanıcı DAO olarak çerçeve kullanılmaktadır ve DaoAuthenticationProvider(DAO Kimlik Doğrulama Sağlayıcı) tarafından kullanılan bir stratejidir. Ara yüzü yeni veri erişim stratejileri için destek kolaylaştırır sadece bir salt okunur yöntemi gerektirir.

Alt ara yüzünün ismi UserDetailsManager(Kullanıcı Detayları Yöneticisi)'dir. Alakalı sınıfları ise CachingUserDetailsService(Kullanıcı Detaylarını Önyükleme Servisi) , InMemoryDaoImpl (Hafızada ki DOA Implementasyonu) , InMemoryUserDetailsManager(Hafızada ki Kullanıcı Detayları Yöneticisi Servisi) ,JdbcDaoImpl (JDBC DOA İmplementasyonu) ,JdbcUserDetailsManager (JDBC Kullanıcı Detayları Yöneticisi), LdapUserDetailsManage(LDAP Kullanıcı Detayları Yöneticisi) , LdapUserDetailsService (LDAP Kullanıcı Detayları Servisi) , UserDetailsServiceWrapper (Kullanıcı Bilgileri Servis Sarıcı)'dır.

Metodun türü kullanıcı detayları olup, "loadUserByUsername(String username)" şeklinde kullanılarak kullanıcı ismine bağlı kullanıcı bilgilerinin yerini işaret eder. Asla "null" dönmez.

Kısaca kullanıcı adı ile eşleşen kullanıcıyı bulur. Gerçek uygulamada, arama muhtemelen uygulama örneğinin yapılandırıldığına bağlı harf duyarlı veya harf duyarsız olabilir. Bu durumda, UserDetails(Kullanıcı Detayları) aslında istenenden farklı bir durumda olan kullanıcı adı olabilen bir nesne döndürür.




Şekil 7. 3 Kullanıcı detayları servisi kullanım şeması

# BÖLÜM 8

# ANDROID PLATFORMUNDA MOBİL GÜVENLİK

Bu bölümde geleneksel kullanıcı detayları servisi inşası, OAuth 2. 0, Oauth2. 0 Parola Hibe(Grant) akışı, yatay dengeleme ile durumsuzluk yüklenmesi dengeleme ve durumsal uygulamalar konularına değinilmiştir.

## 8.1 Geleneksel Kullanıcı Detayları Servisi İnşası

Bulut servisine istek gönderen mobil cihazlardaki istemcilerin kimlik doğrulama ve kendilerini hatırlatma için oturum çerezleri kullanmaları çok iyi bir yol değildir. Örneğin bir sunucu düşünelim. Sunucudaki kaynaklara erişebilen bir istemci olsun. Bu istemci sunucunun sunduğu birkaç hizmetten faydalanmak isteyebilir.

Bulut servisi inşasında servisi farklı mobil cihazların kullanma isteği dikkate alınılmalıdır. Bu ortamda Android cihazlar olabileceği gibi iOS cihazlar da olabilir. Yani birbirinden farklı ortamlarda çalışan mobil cihazlar aynı sunucu ile iletişimde olmak isteyebilir. Farklı telefondaki uygulamaların aynı sunucuya ulaşmak istediğini düşünelim. Uygulamalardan biri sunucudaki bir kaynağa başka bir uygulama da sunucudaki farklı bir kaynağa erişmek istesin.

Her bir uygulamanın kendi tanımlayıcılarını sunucu ile ilişkilendirerek kendini tanıttığı ve sunucudaki servislerden nasıl faydalanıldığı bu bölümün konusudur. Eğer bağlantı sayfasında bağlanıldıktan sonra oturum çerezi kullanılarak oturum standardı kullanılmak istenirse her uygulamaya bağlantı sayfası gösterilip kullanıcı ismi ve şifresi istenecektir.

Bazen farklı mobil cihazlar aynı sunucu ile ilgilenirken bazen de istemci farklı uygulamalara bağlanmak isteyebilir. Her uygulama için her zaman bağlantı sayfası ile meşgul olunulması istemciler tarafından istenilen bir durum değildir.

Bazı uygulamalar istemcilerin uygulamaya her bağlanmak istediği durumda kimlik doğrulama yapmasını zorunlu koşmaz. İstemcinin kullanıcı ismi ve şifresini depolar. Böyle olunca da kullanıcı ismi ve şifre bilgilerinin farklı uygulamaların her birinde saklanması durumu ortaya çıkar.

Güvenlikte hiçbir zaman unutulmaması gereken nokta güvenliği en zayıf noktanın belirlediğidir. Örneğin mesaj iletişiminde TripleDes şifreleme algoritması, kimlik doğrulama da ise Des şifreleme algoritmasının kullanıldığını düşünelim. Sistemin güvenliği esasında Des ile korunuyor gibi olur. Kötü niyetli kullanıcı da kimlik doğrulama üzerinden ortamı bozup bilgileri çalmak isteyecektir. Yani en düşük güvenlikli kısım kötü niyetli istemci için davetiye olacaktır. Burada güvenliğin sadece kriptoloji ile sağlanmadığını da unutmamak gerekir. Şifreleme bir arabadır. Pahalı araba kullanan sürücü kaza yapma ihtimali azdır. Fakat eğer sürücü arabayı etkin kullanmayı bilmezse kaza yapma ihtimali düşmeyecektir.

Mobil dünyada da eğer uygulama bizim kimlik bilgilerimizi düz metin, basit şifreleme veya başka basit mekanizmalar ile korunursa kötü niyetliler ilk o uygulama üzerinden saldırır. Bu sebeple birden fazla uygulama için aynı kullanıcı ismi ve şifrenin kullanılması fikri iyi olmaz. Öbür türlü de sunucudaki her bir uygulama için istemci her uygulamaya bağlanmak istediğinde kimlik doğrulama mekanizmasından geçmesi gerekir. Bu durumda oturum çerezinden vazgeçilmiş olması bir dezavantajdır.

İki adet uygulamanın yer aldığını düşünelim. İlk uygulama çok güvenilir olduğu ikinci uygulamanın ise daha az güvenilir olsun. Eğer ikisi de aynı kimlik doğrulamayı kullanıyorsa(aynı bilgilerle) yapılacak bir şey kalmaz. Her ikisi de aynı kullanıcı ismi ve şifre ile sunucuya bağlanır ve istemcinin hesap bilgilerini kullanır.

Her uygulamanın her şeyi yapması istenmez. Belli şeyleri yapması istenebilir. Bazen de sadece olan şeyleri görüntülemek isteyen uygulamalar olabilir.

Uygulamaların normalde her hakka sahip olması istenmez. Uygulamaların gerekliliklerine göre haklara sahip olması beklenir. Çok uygulamaya tanımlayıcı bilgisi

vermek uygulamaların bunları nasıl ve ne şekilde sakladığı bilinmeyen bir ortamda sağlıklı olmayacaktır.

Uygulamalara bizim sahip olduğumuz yetkilere göre erişebildiğimiz kaynakları direk(kimlik doğrulama olmadan) vermek bir çözümdür. Bu yöntem güvenilen uygulamalara ve sadece var olan bilgileri görüp üzerinde oynama yapmak istemeyen uygulamalar için uygundur. Bu model uygulamada kullanıcı ismi ve şifre ile oturum çerezi oluşturarak hesaplara erişim için uygun değildir.

Özetle kullanıcı ismi ve şifreyi ya her uygulama için uygulama kullanılmak istenildiğinde veya uygulama mobil cihazda bunu saklayarak kullanılır. Uygulamada bu tanımlayıcıları başka bir yerde saklar. Aksi takdirde geliştirici istemediği durumda bile istemcileri bağlantı sayfasına yönlendirir. Bu nedenle uygulamanın belirlediği başka bir yerde tanımlayıcılar olur.

Herhangi bir tanecikli yapı türünde hesaba bağlı olarak kaynaklara erişimin düzenlenmesi yoktur. Hesabın farklı kısımlarına farklı uygulamaların erişimi mobil ortamlar için uygun değildir ve bırakılmaya başlanmıştır.

Kullanıcının uygulamaya her bağlandığından kendisini kullanıcı ismi ve şifre ile tanıtmasının bazı dezavantajları vardır. Mobil modelde bu durum çok istenilmez.

Verilerimize erişim için çoklu uygulamalarla çalışıldığını düşünelim. Yani ortamda kullanıcının iletişim kurmak istediği birden fazla uygulama vardır. Baz uygulamaları birden fazla cihaza hizmet etsin. Diğer cihazlar aynı uygulamaya farklı cihaz türleri ile bağlanır. Kullanıcılar birden fazla uygulamaya bağlanır. Bir uygulama için kullanıcıların şifreleri farklı yerlerde saklanır.

Uygulamalar elverişli olmak için şifreleri saklar. Eğer kullanıcı şifresini değiştirmek isterse ki bu genellikler güvenlik için yapılır kendisini günceller ve şifresini değiştirir. Her zaman şifre değiştirilmek istenilsin. Bu durumda her uyulamaya girilmek istenildiğin de şifre değiştirilir. Her şifre değiştirilme isteğinde kullanıcının kendini güncellemesi kullanıcının gözünde zaman alıcı ve boş bir iş olarak gözükmeye başlar. Bu yüzden her zaman değil ara sıra şifre değiştirmek daha uygun gözükür.

Başka bir sorun da uygulamaya bağlantı iznini verilmesi durumundaki ortamdır. Böyle olunca sunucu hangi uygulamanın erişme hakkının olduğunu bilemez. Örneğin bir

uygulamanın bizim politikalarımıza bağlı olarak sadece belli şeylere erişmesi istenebilir. Bu işleme sunucu dâhil edilmez. Uygulamaya kullanıcı adı ve şifre bilgileri verilerek uygulama ortamdan gider ve sunucu ile olan iletişim başlatılır. Sunucu, uygulamanın erişim hakkının olup olmadığı ile ilgilenmez. Sunucu sadece uygulamanın kullanıcı üzerinden erişim bilgileri ile erişilip erişilmediği ile ilgilenir. Arka plan ile ilgilenmez.

Belli bir süre sonra bu uygulamaya güvenilmemeye başlanıldığında alınacak tedbir ve işlemlerde önemli bir konudur. Bir uygulamaya güvenilmeyip diğerlerine güvenmek olası bir durumdur. Bu durumda yapılabilecek şey ilk uygulamadan kullanıcı ismi ve şifre ile sağlanan bağlantı bilgilerini geri çekmektir. Sonrada kullanıcı ismi ve şifre ile oluşan tanımlayıcı bilgilerinin değiştirilerek güvenilmeyen uygulamanın kullanıcının bilgileri ile sunucuya bağlanamaması yapılacak ikinci adımdır. İşlemler yapıldığında artık güvenilen uygulama kullanıcının tanımlayıcılarını kullanarak sunucuya erişemeyecektir.

Normalde kullanıcı ismi ve şifre bilgisi oturum çerezi yaratmak için kullanılır. Fakat bu model mobil dünyadaki kullanıcıların olduğu ve çoklu uygulamaların farklı veri veya veri gruplarına ulaşmak istediği bir ortamda uygun değildir. Bazı uygulamalara diğerlerinden daha çok güvenilebilir. Fakat kullanıcı tanımlayıcıları ile hangi uygulama olursa olsun sunucuya tam erişim yetkisi verilirse bulut bazlı servislerin ideal olmayan durumunu oluşturarak güvenlik problemlerini ortaya çıkarır.

Kullanıcı hesabı ile ilişkili veri kaynaklarına erişimde erişimin kontrolü istenilen bir durumdur. Burada amaç kullanıcının hiçbir uygulamaya kullanıcı tanımlayıcılarını vermemesi değildir. Kullanıcı verilerine hangi uygulamanın ne kadar ulaşacağı önemli bir konudur.

Yapılmak istenen kullanıcının tanımlayıcılarını değiştirirken uygulamaların erişimlerinin bundan etkilenmemesidir. Benzer şekilde belirli bir uygulama için erişimin kaldırılabilmesidir. Bu işlemler bağlantı ekranından girilen verilerle oluşturulan oturum çerezlerinin kullanılarak uygulamalara tam erişim verildiği ortamlarda kolay değildir.

Kullanıcıların kimliğini doğrulama ve kendi oturumlarını tutma arasında sıkı bir ilişki vardır. Doğrulanmış oturumları kaçırma, tespit ve tekrar karşı korunması işleriyle uğraşılırken gereken, anonim oturumları için çok hassas değildir. Aslında, geçerli bir oturum belirteci oturumu doğrular[82].

## 8.2 OAuth2. 0 Giriş

Oauth 2. 0, aslen 2006 yılının sonlarında oluşturulan Oauth protokolünün sonraki evrimdir. Belirli yetkilendirme sağlayan web uygulamaları, masaüstü uygulamaları, cep telefonları ve oturma odası cihazları kullanırken Oauth 2. 0 istemci basitlik üzerine odaklanmaktadır. Bu belirtim IETF Oauth WG içinde geliştirilen Oauth WRAP önerisine dayanmaktadır[83].

Oauth 2. 0'ın önceki versiyonlarından en önemli farklı uygulamalara güvenlik tedbirlerinin eklenmesindeki kolaylaşmadır. Fakat Oauth 1. 0'a geri desteği yoktur. Yani Oauth2. 0'u 1. 0 versiyonun desteklememektedir. Bu teknoloji Android taraflı mobil uygulamalar tarafından desteklendiği kadar Windows taraflı mobil uygulamalar tarafından da desteklenmektedir.

Oauth yaygın olarak, erişim bilgileri tehlikeye düşer endişesi olmadan, web sörfçüsünün kendi Google, Facebook ya da Twitter hesaplarını kullanarak üçüncü taraf web sitelerine oturum açmak için bir yol olarak kullanılmaktadır[84].

Oauth 2. 0 imza, şifreleme, kanal bağlama veya istemci doğrulama desteklemiyor. Bu gizlilik ve sunucu kimlik doğrulaması tamamen SSL'e dayanır[85].

Bu bölümde Oauth 2. 0'ın kimlik doğrulama servisi, jeton servisi ve desteklenen hibe türleri, istemci kayıt ve alan yönetimi faydalarından ziyade konunun mobil tarafı incelenecektir.

### 8.2.1 Mobil Kimlik Doğrulama

Mobil istemcileri (mobil cihaz uygulamaları) gizli olmayan kurumsal istemcilerinin yanında OAM Oauth 2. 0 Servis mobil uygulamalar ilk servisini kullanmak için OAM ile kayıtlı olması gerekir gerektirir.

OAM Oauth 2. 0 Mobil Oauth mobil istemcilere güvenilir erişim sağlamak için geleneksel bir 3 aşamalı Oauth akışı ile çiftlere benzersiz bir mobil uygulama kaydı ve cihaz kimliğini akışı sağlar.

### 8.2.2 Mobil İstemci Güvenliği

Çoğu Oauth istemcisi istemcinin gizli bilgilerini(uygulama şifresi veya gizli anahtar) tutmayan bir tüketici uygulamasıdır. Bu Oauth istemciler genel istemci veya gizli olmayan istemciler olarak adlandırılır.

Çizelge 8. 1 OAuth Kullanımının Faydaları

| Oauth Kullanmanın Faydaları |
|---|
| •Kaynak kontrolü daha kolay |
| •Bireysel uygulama erişim iptali |
| •Uygulamanın neye/kime eriştiğini bildirme |
| •Uygulamaya kullanıcı ismi ve şifre bildirmeme |

## 8.3 Oauth2. 0 Ve Parola Hibe(Grant) Akışı

Web'de yol gösterici anahtarların dağıtılmasının arkasında bir takım prensipler vardır. Bu bölümde mobil kimlik doğrulama için Oauth2'nin kullandığı prensipler incelenecektir.

İlk iş kaynakların düzenleme işini güvenirlik varlığı olan kaynak sunucusundadır. Sonra iş Oauth2 ile erişim kontrolünde kullanılan bilgi bitleri veya sunucuya gelir. Bu varlığa Spring kontrolcüsü adı verilir. Veri tabanında saklanan veriler kaynak sunucusunun bir parçasıdır. Uygulamaya konulmak istenen spring(bahar) bazlı uygulama veya bulut servisi kaynak sunucu olarak düşünülür.

İkinci parça kimlik doğrulama sunucusudur. Bu sunucu kimin neye ulaşabileceğine karar verip ilişkilerin yönetim tiplerini belirler. Bu parça yerine işlerin kombine edildiği bir tekil uygulama kullanılabilir. Örneğin Spring Oauth kullanıldığı zaman Spring kimlik doğrulama sunucusunun yapması gereken işi yapan bir sunucu içerir.

Ayrıca ortamda kullanıcı olarak düşünülebilen kaynak sahibi vardır. Bazen buna sadece kullanıcı da denir. İstemci bazı kaynaklara gidip ulaşmak istediğinde yol gösterici anahtar ile belirli kaynağa erişim hakkına sahip olur. Esasen istemci Oauth2'ye yol gösterici anahtarını almak için kimlik doğrulama sunucusuna sorar veya yol gösterici olarak belirteçler kullanılır.

İstemci gerekli olan belirli belirteç için istemci /Oauth2/token'a istek yollar. Kullanılan belirli bir Oauth2 iş akışına göre istemcinin bir kullanıcı ismi ve şifre sayesinde belirli kaynak sahibine ulaşılmak istenilen bir sistem düşünülsün. Kimlik doğrulama sunucusu bilgiyi doğrulayabilip ve gerekli belirteci ürettiği varsayılsın. Bunun için buluttaki servis için istemci inşa edilip ve istemcinin implementasyonu kontrol edilir ve bu şifre kullanımı kabul edilip güvenilir.

Genel olarak sistemlerde bağlantı ekranı ile kaynak sahibine izin veya kullanıcı ismi ve şifre verilerek istemcinin belirtece veya yol gösterici anahtara sahip olması beklenir. Bu bölümde kimlik doğrulama sunucusu sayesinde kaynak sahibine yanıt döndürdüğü yolu kullanılacaktır. Bağlantı sayfası kaynak sahibine gösterilir.

Kaynak sahibinin bağlantı ekranı ile sisteme bağlanması beklenir. Sonra kimlik doğrulama sunucusu istemcinin erişmek istediğini ve yetkilerinin ne olduğunu bildirir. İşlemlerin başarılı olup olmadığı kontrol edilir.

Sonraki adımda kimlik doğrulama sunucusu sistemin isteğine göre istemciye belirteç verilip verilmeyeceğini belirler. Bu bölümde varsayılan olarak şifrenin tercih edildiği düşünülsün. Kullanıcı istemciye kullanıcı ismi ve şifre içerdiğini kanıtlaması gerekir. Çünkü istemci inşa edilip sunucuda inşa edildiği zaman belirli istemci için güvenilmesi gerekir. İstemciye kanıtlama için kullanıcı ismi ve şifre kullanılır. Bu işleme şifre tercih etme adı verilir.

İstemci kimlik doğrulama sunucusuna istek gönderir. Sonra Spring bu isteği /Oauth2/token yoluna iletir. İletim isteğinin içinde istek vücudunda kullanıcı ismi ve şifre çiftinin URL çözülmüş hali yer alır. Bu değerler kullanıcı tarafından sağlanan kaynak sahibi veya istemci tanımlayıcı olabilir. Bu durum istemcinin tanımlayıcısı olarak kullanılır.

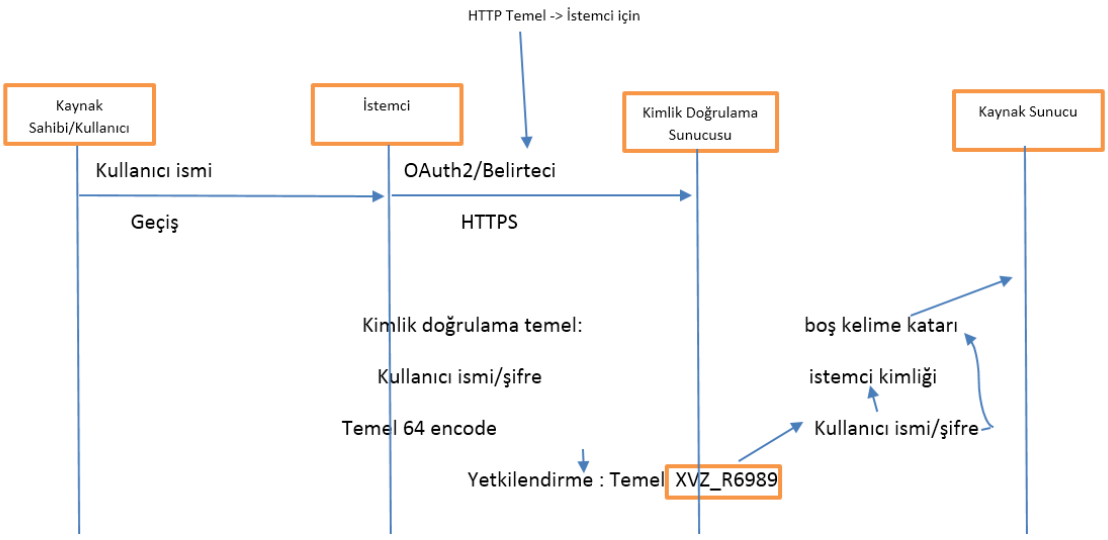
Esas olarak arka planda yatan şey istemcinin çok istemciye güven duyulması zorunluluğunu azaltıp, daha az istemciye güven duyulmasını sağlamaktır. Bu sayede sunucu belirli bir istemci ile ilgili sırları saklayabilir. İstemcide kendi tanımlayıcı ve sırı ile kendini tanımlayabilir.

Mobil cihaza istemci eklenmesinin kontrollü olduğunu fakat bu kontrolü yapacak kişilerin bu kontrolün alt yapısı ile ilgilenmek istemediklerini düşünelim ve eğer sır

istemciye gömülürse herhangi biri uygulamayı çözümleyerek şifreleri ortaya çıkarabilir. Bu yüzdende istemcinin sırlarının olması ve kimlik doğrulamasından geçmesinin önemi kalmamaktadır.

Başka bir modelde HTPP basit kimlik doğrulamasında boş şifre kullanılmasıdır. İstemciler şifre ile ilişkilendirilmez. Bu modelin kullanılmasının tek nedeni Spring'in istemci için basit kimlik doğrulaması kullanılması ve bunun Oauth2'nin içinde yer almasıdır. Fakat bu durumda sıra sahip olmadan istemci boş şifreye sahip olacaktır. Bu durum yani boş şifre kullanmak korkutucu görülebilir ve kimsenin erişemeyeceği anlamına gelebilir. Fakat aslında öyle değildir.

Bir istemcinin geçerli bir kullanıcı ismi ve şifreye normalde sahip olması gerekir. Böyle olunca da istemcilerin belirtece ve yol gösterici anahtara sahip olması durumuna geri dönülür. Böylece istemcinin özel bir istemci olduğu kanıtlayamaz ama bu iş akışı istemci ve sunucu taraflı iş için uygundur.  Kimlik doğrulama sunucusu sağlanan tüm bilgilere bakacaktır. HTTP basit kimlik doğrulamada gerekli kullanıcı ismi ve şifre değerleri geri döner ve istemciye bir belirteç döndürülür. Bu belirteç istemcinin birkaç saniye sonra kullanabileceği bir kelime katarıdır ve kimlik doğrulama ve erişimin sonraki kısmını oluşturur.



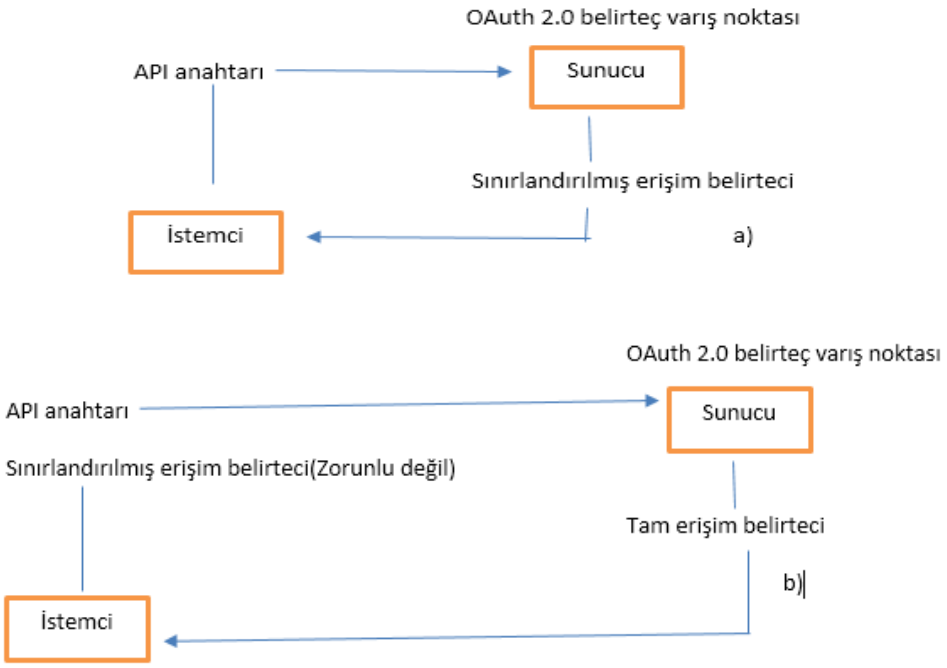

Şekil 8. 1 OAuth 2. 0 ve Parola Hibe (Grant) akış şeması

Özet olarak hibe türleri OAuth 2. 0'ı esnek kılar. Kimlik doğrulama kodu OAuth 1'dekine göre daha az şifreli olarak benzer ve eklenti hibe mobil uygulamalar gibi daha az güvenlik sağlar. Eski uygulamalar adım adım Oauth 2, Müşteri Kimlik Hibe Gömülü

Hizmetler ve arka uç uygulamaları için mükemmel geçmesi için Kaynak Sahibi Şifre Kimliği hibe verir [94].

### 8.4 Oauth2. 0 Ve Parola Hibe(Grant) Akışı

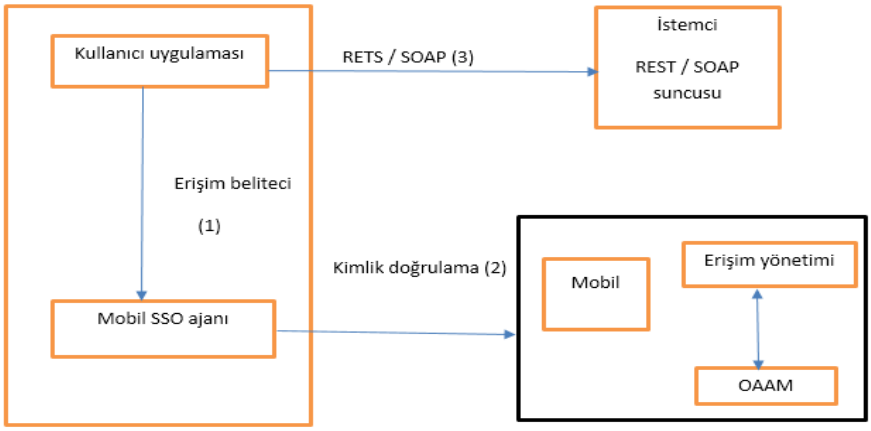
Erişim belirteçleri korumalı kaynaklara erişmek için kullanılan kimlik bilgilerini kullanır. Karakter katarı istemciler için genellikle apaktır. Belirteç, özel kapsamları ve erişim süreleri, kaynak sahibi tarafından verilen ve kaynak sunucu ve yetkilendirme sunucusu tarafından zorunluluğu temsil eder[86].




Şekil 8. 2: Kaynak erişim belirteci kullanım çeşitleri a) Sınırlandırılmış erişim belirteci b) Tam erişim belirteci şekli [98]

OAuth 2. 0 API'ler iş akışları aşağıdaki kaynakları kullanarak uygulanabilir:

- /oauth20/token
- /oauth20/authorize 'dir.




Şekil 8. 3 Mobil SSO Ajanı erişim belirteci kullanım şeması

Bu alanlarda kısıtlı olmamakla birlikte bir erişim belirteci en çok şu alanlarda kullanılır:

- Tanımlayıcı olarak kullanılabilir.
- Belirteç ile ilişkili konu tarafından oluşturulan nesnelerin varsayılan sahibi, birincil grup ve ACL'dir.
- Oturum doğrulama servisi tarafından korunur ve tüm bilgileri (kimlik) bir koleksiyon oturum açarken sağlanan kullanıcı ile kimlik doğrulama paketleri ile doldurulur.
- Kısıtlayan grup tanımlayıcıları (isteğe bağlı) kullanımı: Grupların bu ek set ek erişim izin vermez ama daha da kısıtlar, bu gruplardan birine de izin eğer bir nesneye yalnızca erişim izin verilir[87].

## 8.5 Spring Oauth2. 0 Güvenliği

Oauth veya Oauth2.0, web bazlı servisleri ile ilgili tüm ortamdaki farklı kimlik doğrulama problemlerini çözmek içindir. Oauth2. O ile bulut servislerinde inşa edilecek web bazlı uygulamalarda kimlik doğrulama ve kontrol erişimi için birden fazla akış bulunmaktadır.

Bu bölümde bütün akışların açıklanması yerine buluta erişim uygulamalarında en fazla kullanılan akış üzerinde durulacaktır. Bahsedilecek akış geliştiricinin erişiminin kendi sorumluluğunda olmadığı durumdur. Yani geliştiricinin inşa etmediği bulut servisine erişim ve başka birinin geliştiricinin geliştirdiği bulut servisine erişimi konusuna burada

değinilmeyecektir. Buradaki akış geliştiricinin uygulama ve bağlanılacak buluttaki servisi kendisinin inşa etmesidir.

Oauth'da yukarıda bahsedilen durum kullanıcı şifresi kabulü olarak adlandırılır. Mobil ortamda bu akış daha esnek olarak orijinal kullanıcı ismi ve şifre bilgilerini içeren oturum belirtecinin kullanımıdır. Bu fikrin arka planında kimlik doğrulama sunucu bulunması yatar. Ortamda kaynak sunucusu da bulunur. Bunlar Oauth terimleridir.

Kaynak sunucusu geliştiricinin çok faydalandığı kontrollerden biridir. Kaynak sunucusu erişimi sağlayan mantık kurallarını içerir. Kaynaklar bulut servisine erişim sağlandığında kullanılması yüzünden ilk başta kaynak erişimi yerine kaynak sunucusundan onay alınması gerekir.

Kimlik doğrulama sunucusu ise kaynaklara erişimi kontrol eder. Kontroldeki farklı metotlar içinde kimin kontrolleri erişim için izin verileceğine karar veren ve yönetme işi de kimlik doğrulama sunucusunun görevidir.

Oauth 2 şifre kabulü için kullanılarak istemcilerin sunucudaki kaynaklara erişiminin kontrolleri ve kuralları belirlenir. En son olarak istemciye bağlantı formu gösterilir.

Mobil istemci şifre kabulü için kimlik doğrulama sunucusu isteği gönderilir. Burada önceki bölümlerde gösterildiği gibi sadece kullanıcı adı ve şifre ile ortama bağlanılamaz. Çünkü bu akışta amaç sadece kişiyi tanımlamak değil, kişinin yetkilerine göre hangi kaynaklara erişebileceğini bilerek kontrol etmektir. İstek gönderildikten sonra kimlik doğrulama sunucusu oturum çerezine benzeyen bir belirteç üretir. Bu belirteç ileride kullanılır. Belirteç, internette yol gösterici anahtara denk gelir.

Kimlik doğrulama sunucusundan dönen belirteç yüklenerek istemcinin belirli kaynaklara erişimini sağlar. İstemci bu belirteci edindiği zaman ve belirli bir kaynaklara erişim istendiği zaman bu belirteç kaynak sunucusu tarafından kontrol edilir.

Oturum çerezleri ile belirteçlerin farkı belirli istemcilerin belirli istemci ve kaynaklarla erişiminde oturum çerezlerinin kullanılamamasıdır. Çok istemcinin ortamda olup belli istemcilerin belirli kaynaklara ulaşması istenildiğinde belirteçlerin önemi ortaya çıkar.

Belirteçlerin en önemli özelliklerinden birisi de istemcilere erişimin tamamen açık olması zorunluluğunun bulunmamasıdır.

Belirteç aslında kullanıcı ismi ve şifre bilgisi değildir. Bu duruma örnek olarak istemci şifre bilgisini değiştirmek istediğinde bazı durumlarda belirteçlerin değiştirilmemesi gösterilebilir. Belirteç sayesinde istemcinin bağlanma zamanı da belirtilebilir. Bu sayede istemciye erişim zaman aralıkları bildirilebilir.

Oauth2, klasik şifre bilgisi ile bağlanılan ekranlara benzemesine karşın oturum çerezinin döndürülmesi ve daha karmaşık işler yapabilmesi sayesinde farklıdır. Belirteç bazlı erişim, mobil istemci için daha yüksek güvenilirlik sağlar. Kaynaklara erişim için daha kolay br mekanizma sağlar.

Kaynaklara erişim metotlarının geliştiricinin kontrolünde olması belirtecin bir başka özelliğidir. Bu sayede geliştirici kaynaklara erişim metotlarını kendisi oluşturarak hangi istemcinin hangi zaman aralıklarında hangi kaynaklara erişimine izin vereceğini belirler. Her istemci kendi belirtecini kimlik doğrulama sunucusunu göndererek sunucudaki belirli kaynaklara erişim hakkı kazanır. Bu sayede kaynaklara erişim kontrol altında olur.

Nesneye yönelik programlama içindeki başka akışlar sayesinde geliştiricinin oluşturmadığı uygulamalara diğer istemcilerin erişimi için belirteçler uygulamanın bağlantı ekranı gösterilmeden istemcinin kontrolü sayesinde erişim akışları mevcuttur. Bu konu burada ele alınmamıştır. Çünkü bu akış servisin geliştirici tarafından oluşturulup, istemcilerin hangi kaynaklara hangi zaman aralıklarında erişebileceğinin geliştirici tarafından belirlenebilmesi istenildiği durumda uygun olmamaktadır. Bahsedilen akış esasen güven konusuna girmektedir. Güven, üçüncü parti uygulamaların esas uygulama ile ilgili olan iletişiminin güvenliği ile ilgilenen bir konudur.


Şekil 8. 4 Mobil ortamda güvensiz beliteç kullanımı

Sırada incelemek akış sunucu tarafından belirteç alması uygun olan istemcilerin belirlendiği istemciler için belirteçlerin elde edilediği ve bu belirtecin kaynaklara erişim için kullanıldığı bir akıştır. Bu akışta arka planda yapılan iş çoktur.

Bahsedilen akış Oauth 2 de ve Spring güvenliğinde mevcuttur. Bunun yapımında Oauth 2'un spesifikasyonları ile geliştirici kendi gerçekleştirimi ile yapması yerine var olan gerçekleştirimin kullanıldığı duruma değinilmiştir. Spring güvenliğinin gerçekleştirimleri tüm akışlar için uygundur. Bu akış sadece mobil istemcilerin bulunduğu ortamda şifre yol göstericisinde kullanılmaz[89].

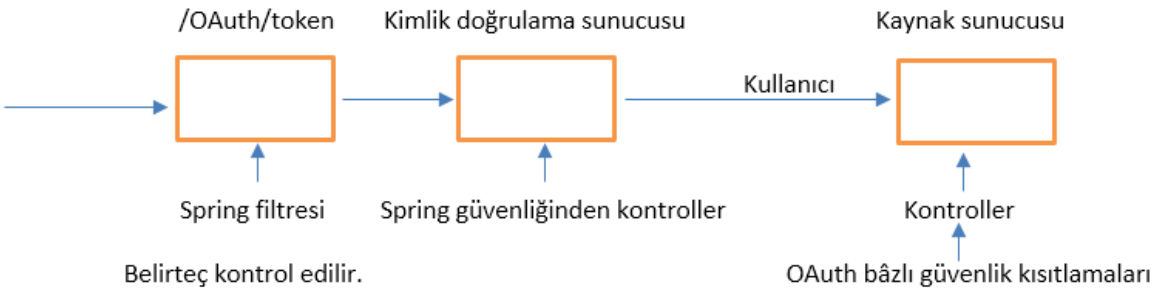
Şifre kabul akışı için Spring bahsedilen durum için gerçekleştirime sahiptir. Konunun geri kalanında bahsedilecek akışta kimlik doğrulama veya yetkilendirme sunucusunun olduğu ve geliştiricinin kaynak sunucusunun olduğu varsayılmıştır. Yine geliştirimin kontrol ve metotları kaynaklara erişimi belirlemiştir. Bu nedenle kontrol ve metotlar kaynak sunucusunda yer alacaktır.

Oauth sunucusu, Spring güvenliği tarafından gelen isteklerden belirteçleri sağlayan ve belirteçleri kontrol eden kontroller bütünüdür. Temel olarak belirteç alınması için varış noktaları belirlenip Oauth/token dizini gibi dizine konfigürasyona bağlı olarak ve istekle alınıp Spring güvenliğindeki kontrolleri parçalara böldür ve belirteçleri dağıtır.

Sonra istek gönderildiği zaman istek filtrelenir ve istek durdurularak belirteç kontrol edilir. Spring güvenliği belirtecin işini kendisi yapar. İstemcinin hangi kaynaklara erişeceğine belirtece bakılarak karar verilir. Diğer önemli olan şey geleneksel güveliğin sağlanabilmesi veya kontrollerdeki notasyonlar sayesinde Oauth bazlı güvenlik şartları temel olarak sağlanmasıdır.

Oauth, eğer istemci belirli kontrolcü metotlarına Oauth sayesinde erişebilirse kontrol içindeki metotlara iddia ekler ve belirteci ilişkili bir alan iler alakaldığını ispatlaması gerekir. İstemcinin erişimi için kontroller veya var olan metotlar tarafından belirli bir alan için tanımlanması gerekir. Bu sayede benzer kullanıcı ismi ve şifre için farklı belirteçler hazırlanıp, farklı alanlarla ilişkilendirilebilir. Sonra kullanıcının farklı belirteçleri belirleyip kontrol ve kontrollere erişimini belirler. Bu sayede istemci her zaman sunucuya belirteç istemiyle gider ve sunucu bu belirtece alan belirlenir.

İstemci belirtecinin belirttiği alanlara erişebilir ve sunucuda istemcinin alana erişebilip erişemeyeceğini belirler. Eğer belirteçte alan belirlenmezse istemci erişim için kontrolcü metotlara erişmesi gerekir. İstemcinin istekleri işlenemez ve istemciye kimlik doğrulama yapılamaz erişim reddedilir.




Şekil 8. 5 Mobil ortamda filtreli olarak beliteç kullanımı

## 8.6 Yatay Dengeleme

Bu bölümde bir uygulama inşa edildiğinde ve bu uygulamanın birden buluta bağlanan günde belki de yüz binlerce kişinin kullandığı bir mobil uygulamaya dönüştüğünde güvenliğin yapılandırılması, kullanıcı trafiğini nasıl yöneteceği konusu işlenmiştir.

Bölümün esas konusu bulut servisleri inşasında ve inşa sonraki süreçte istemcilerin isteklerinin alınıp uygun yanıtların verildiği ortamda istemci sayısının artmasının sunucuyu cevap veremez hale getirmesin önlemektir. İstemci sayısı ile sunucu ilişkisini optimize edilmesini sağlamaktır. Bunu iki yöntemi bulunmaktadır.

İlk yol sunucuya istekler gönderildiğinde istemcilerin sayısı belirli bir sayının altında ise büyük sunuculara gerek olmadığı durumdur. Fakat istek sayısı belirli bir sayıyı aştığında isteklerin cevaplanabilmesi için daha büyük sunuculara ihtiyaç duyulur. Bu yöntemde isteklerin artması sebebiyle sunucu kapsamlarının dikey olarak artması gerekir. Bu da sorunlara neden olacaktır. Belirli bir istek sayısı aşıldığında sunucu artık büyütülemeyeceğinden sorunlar başlayacaktır. Sunucu artık bu durumu destekleyemeyecektir. Tekil makinada bütün istekler işlenemeyecektir.

İkinci yol ise uygulamayı makinelere bölerek ortak olarak çalışmasını sağlamaktır. Bu sayede artan isteklerde sunucunun yeni duruma uyum için kapatılmasına gerek kalmayacaktır. Buna yatay dengeleme denir.

Bulut servislerinin inşasında servislerin yatay olarak dengelenmesi istenilen bir durumdur. Bu sayede yeni yüklemeler geldiğinde bunlar diğer makinelere kolayca bölüştürülebilecektir. Yeni kapasite istendiğinde sunucu olarak yeni makine eklenmesi yeterli olacaktır. Yatay dengeleme kolaylıkla kullanılabilecek bu durum değildir. Çünkü maliyeti arttırmaktadır. Yeni yüklemelerin diğer makinelere bölüştürülmesinin optimize edilmesinin ne zaman yapılacağının iyi karar verilmesi gerekir[89].

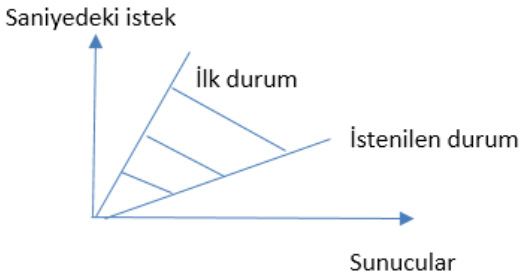
Hesaplama gücü saniyedeki istek olarak ölçülür. Bu sayı uygulamadaki sunucu sayısı ile ilişkilidir. Ortamdaki makine sayısı arttığında hesaplama gücü artar ve bu sayede saniyedeki işlenen istek sayısı artar. Gerçekte hesaplama gücü direk olarak saniyedeki istek işleme ile alakalı olmayabilir. Bu sayı uygulamanın bütünsel olarak uygun bir şekilde istemci isteklerini işlemesi ile ilgilidir. Aslında tek değişkene bağlı değildir.

Gerçekte uygulamalar yatay dengelemeyi kullanmak ister ve buna uygun mimarilere uygun olarak inşa edilirler. Ama bunu makinelerini kullanarak yapmak istemezler. Var olan makineleri en optimize kullanmak isterler. Fakat istemcilerin kendi isteklerine hızlı bir şekilde cevap istemeleri geliştiricileri bir paradoksa sokar. Bir tarafta fazla makine ile artan maliyet bir tarafta hızlı cevap almak isteyenler bulunur. Ayrıca yeni makinelerde mevcut istemci bilgilerinin yüklenmesi durumunda dezavantajıdır.

Durumsal uygulama yapılmak istenerek dezavantajdan kurtulmak istenir. Eğer bir sunucu istemci hakkında bir şey hatırlamazsa bu uygulama durumsal olacaktır. Bu durumda istemcinin istekleri ortamdaki herhangi bir sunucuya gönderilebilecektir. Bu sayede ortamda yeni sunucu eklenip her sunucuya istemciler tanıtılma zorunluluğu ortadan kalkar[90].

Bazen de bazı önemli durumların isteklerle ilişkilendirilmesi gerekir. Burada da yeni makinelerin durumu nasıl elde edeceği, erişimin nasıl olacağı istemcilerin bu bilgilerinin yeni makinelere nasıl taşınacağı ve bunların nasıl aktarılacağı sorunları ortaya çıkacaktır. Bu sorunların çözümü normalden daha zordur. Duyumsuzluğu, uygulamada belirlemek ve durumu saklamak istenen özelliktir.

Sunucuların herhangi bir isteği işleyebilmesi istenen bir durumdur. Yüksek ölçeklenebilir uygulamalar inşa edilmek istendiğinde dezavantaj getirebilecek durumları optimize edimesi gerekir.




Şekil 8. 6 Hesaplama gücü ilk durum ve istenilen durum ilişkisi

Bulutta genellikle çoklu sunucularda çalışan uygulamalar bulunur. Örneğin bazen yüz farklı sunucuda çalışan uygulamalar olabilir. Çeşitli istemcilerin istekleri çeşitli sunuculara iletim yönetimi düşünülmesi gereken bir durumdur. Bunun için istekler her bir özgün sunucunun servleti ile ilişkilendirilip yolunun belirlenmesi gerekir. Fakat her biri farklı makinelerde çalışan çoklu kopyaları varsa gelen isteğin hangi makineye yönlendirileceğinin belirlenmesi gerekir.

Bir makineye erişildiğinde yönlendirmenin dışsal olarak yapıldığı bilinir. Fakat bu yönlendirmenin nasıl olduğu, istekleri hangi makinenin isteği işlediği ve makinelerin ne zaman kapasitesine ulaştığı durumu önemli sorunlardır. Bu sorunlar HTTP yüklenmesi dengeleme konularıdır.

Tipik olarak hangi isteğin ortama geldiğini, bu isteğin değişik makineler için nasıl yönlendirileceği yükleme dengelemenin sorumluluğundadır.

İstek geldiği zaman bu bir makineye yönlendirilir. Diğer istek geldiğinde bu istek başka bir makineye yönlendirilir. Fakat bunun dengeli yapılması gerekir. Servletler bu makinelerde çalışıp durumsuzluk sağlamak istenir. Yani istemcinin hangi makine ile iletişimde olduğu önemli değildir[91].

Tarayıcının belirli bir makine ile iletişim içinde olduğu bir ortam düşünülsün. İstekler bu makineye gönderilerek ve işlenerek, hazırlanan uygun cevap tarayıcıya gönderilsin. İlk başta ilk istek seti geldiği zaman bunlar ortamdaki bir makineye gönderilir. Fakat tüm bu yapılacak işlemlerden önce ilk bağlantı ekranından onay alınması gerekir. Bağlanma işlemi başarı ile sağlanmışsa sıradaki isteklerin makineye iletimi vardır. Burada genellikle dönüşümsel değişimli makinelere iletim yolu kullanılır. Başka istekler geldikçe diğer makinelere dönüşümlü olarak iletilir.

Sunucunun optimum biçimde düzenlenmesinin bir avantajı makinenin istemcinin bağlanıp bağlanmamasından haberi olmamasıdır. İstemci bağlantı sağlandıktan sonra gerekli işlemlerin olmasını bekler. Başka bir avantajlıda isteklerin makinelere dağıtımında yük dengeleme için kör dağıtımının kullanılabilmesidir.

Bütün isteklerdeki yük dengeleme için durumları hatırlama veya istekleri akıllıca yönlendirme kullanılır. Bu çözümlerin herhangi birinin kullanılabilmesi oldukça zordur. Bazen de istemciler belirli makinelerle ilişkilendirilerek o istemcinin istekleri önceden belli olan makineye gönderilebilir.

İstemci mobil cihazdan belirli bir makine ile iletişime başlarsa mobil cihazda durum bilgisi saklanır. Mobil cihazlardan istek gönderen istemcilerin istek seviyeleri farklı olabilir. Bu durum ortamın planlanmasında düşünülmesi gereken bir durumdur.

Mobil cihazların hep aynı makineye yönlendirilmesi bazen bazı makinelerin çok çalışarak bazılarının ise az çalışması sorununa neden olur. Bu durumda diğer makinelere kaydırma yapılır. Fakat o zamanda sunucunun planlamasının iyi yapılandırılması gerekir. Bu durum içinde uygulamanın durum bilgisi kullanılır[103].

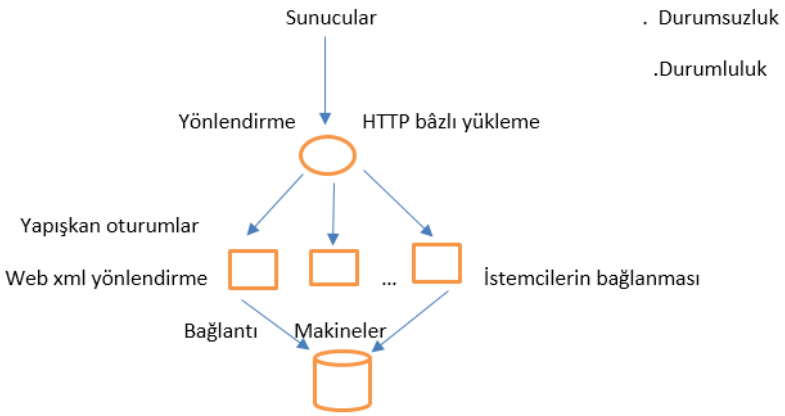
Eğer durumsuzluk olursa yönlendirme daha kolay olacaktır. İstekler herhangi bir makineye kolayca dağıtılabilecektir. Fakat uygulama durumsal ise işler daha zor olacaktır. Bu sefer de dağıtım zor olacaktır. İstemcinin istekleri benzer makinede iletişim içinde olur.

Oturum denen kavram mobil istemcinin makine ile konuşurken hep aynı makineyi kullanması durumudur. Temel olarak isteklerin bireysel sunuculara yönlendirilip işlenmesidir.

Kullanılabilecek başka bir yol ise isteğin herhangi bir bireysel makineye yönlendirilmesidir. Bu da oturum dağıtılmasına benzer. Her ikisinde kimin sisteme bağlanıp bağdalanamadığı bilgilerin merkez veri tabanında saklar. Diğer bir istek geldiğinde ilk başta istek sahibi istemcinin sisteme daha önce bağlanıp bağlanmadığı kontrol edilir. Bu bilginin kontrolünde veri tabanı kullanılır.

Merkezi sürekli mekanizması sayesinde farklı sunucular üzerinden sağlanarak isteklerin nasıl dağıtıldığının düşünülmesine gerek kalmaz. Bu katmanda durum bilgisi saklanmaz. Onun yerine daha alt katmanda saklanır.

HTTP yükleme dengelemesinin çeşitli sunucularda çeşitli kriterlere bakılarak uygulamanın durumsuz veya durumsal olduğuna karar verilebilir. Eğer uygulama durumsal ise durumun yönetimi işini düzenlemek ve durumun tekil makinedeki hafızada saklanması oldukça zordur.




Şekil 8. 7 Durumsuzluk yüklenmesi dengeleme ve durumsal uygulamalar şeması

## 8.7 Otomatik Dengeleme

Bulutta yatay dengelemenin bir özelliği ise bulutta inşaatın elastik olması, maliyetlerin azaltılması ve uygulamada ki isteklere göre yanıtların oluşturulabilmesindeki otomatikliktir. Uygulamada yatay dengeleme kullanılıyorsa yeni makineler eklenerek yük dağılımı yürütülür.

Bölümün konusu uygulamanın kendi veri merkezinde çalıştığı durumda veya uygulamayı desteklemek için yeni birimler alındığında dengelemenin nasıl olacağıdır. Yani makinelere yeni bir makine eklenmesi durumunda bunun mümkün olan en az değişiklikle ortama hızlı bir şekilde adapte olmasını sağlamaktır.

Kendi veri merkezini oluşturmak veya makinenin yeni mekanizmaya uyumu ve makineyi yeni alt yapının desteklemesi otomatik dengelemenin konularıdır.

Normalde ortama yeni makine eklenmesi için belirli bir seviyeye gelişilmesi gerekir. Yani geliştirici tarafından belirlenen bir eşik değeri aşıldığında artık yeni bir makinenin ortama katılması gerektiği anlaşılır.

Benzer şekilde yeni makine alınıldığında bu makineninde istekleri optimum şekilde yürütülmesi gerekir. Diğer taraftan yeni makine ortamdaki en büyük maliyet kalemidir.

Bulutta yeni makine adaptastonu kolaydır. Buluttaki elastiklik özelliği sevilen bir özelliktir. Çünkü bulutta fiziksel olarak makine alınmasına gerek kalmaz ve yeni makinenin ortama uyumu kolaydır.

Bulut sayesinde yapılacak değişiklik planları fiziksel altyapıdan çok soyutsa planlamadır. Fakat buluttaki soyut makinelerin alımda maliyetli olması, maliyet dezavantajını tam olarak ortadan kaldıramamaktadır.

Esasen birinin bir konsol üzerinden ortamdaki yüklemeyi inceleyerek gerekli düzenlemeri yapmasını bekleme, en iyi durumdur. Fakat normal hayatta bu çok normal değildir. Örneğin sunucuların ne olduğu bilinir ve davranışları az çok tahmin edilir. Eğer CPU'nun yükleme kapasitesi, isteğe yanıt döndürme süresi gibi özelliklerin miktarları ölçülebilirse bu bilgiler ışığında bir plan uygulanıp yeni makinenin ortama eklenmesi gerektiği otomatik olarak belirlenebilir.

Otomatik yükleme için internette kapasite verebileceği zamanı karar veren bir akıl kullanılır. Bu akıl bulut ile iletişime geçerek yeni makineler eklenmesi işini makinelerin ne kadar sonra ömrünü tükettiğinin bakılmasına otomatik olarak karar verir. Tanımlayıcı kullanılması ile akılcılığın inşasında otomatik olarak kapasite eklenir veya belirli bir kapasite kaldırılır. Buna otomatik yükleme denir.

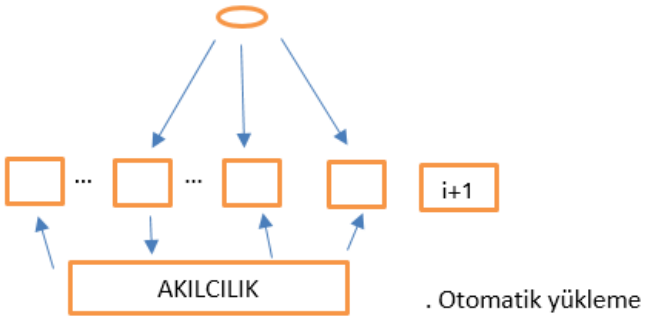
Otomatik yükleme bulutun efektif olarak kullanılmamasını sağlayarak isteği artması durumunda otomatik olarak gerekli tedbirleri alır. Bu kavramın en önemli gereksinimi farkına varmaktır.

Bulut ortamında makinelerin hızında bir limit vardır. Yeni makinelerin uygulamanın ortamına uyarlanması zaman ve maliyet alacaktır. Bu akılcılığın inşası önce kurulacak ortam hakkında toplanılabilecek kadar bilgi toplanması gerekir. Eğer planlama iyi yapılırsa yeni makinelerin ortama eklenmesi kısa sürede olur.

Başka bir sorunda buluttaki mevcut yükün yanı sıra gelecekte ortaya çıkacak yükü de ölçmektir. Eklenecek düğümlerin analizin iyi yapılarak, ne zaman ekleneceğinin tahmin edilebilmesi gerekir. Uygulamada yatay dengeleme kullanılacaksa yeni makine ekleneceğinde var olan makinelerin bağlantısının ne zaman koparılacağının

düşünülmesi gerekir. Bu bağlantı koparılmasını önlemek için otomatik dengeleme yöntemi kullanılır. Fakat işlerin otomatik olarak yapılmasına karar verecek karar alma mekanizmasını kurallarının iyi belirlenmesi gerekir. Bulut ortamında yaratılacak sanal makinelerde otomatik dengeleme yöntemini kullanmak daha kolaydır[93].

Akıllı mekanizma sadece makine eklemede kullanılmaz. Ortamda özellikle makinelerde sorun olursa uygulanacak çözümlere karar verme, istemcilerin isteklerini sınıflandırıp öncelik sağlayabilmesi gibi görevleri vardır. Esas amacı ortamın çalışmasını otomatik olarak sağlamaktır.




Şekil 8. 8 Otomatik yükleme mekanizması

## 8.8 Bulut Hizmet Çeşitleri

Bu bölümde mobil ortamda bulut servis modellerinden olan platform olarak servis ve altyapı olarak servis modelleri ve farklılıkları incelenecek.

### 8.8.1 IaaS (Bir Hizmet Olarak Altyapı)

Bu servis modeliyle bulut hesaplama servisleri ile sanallaştırılmış ortamda kaynaklar üzerinde işlem yapılabilmek için genellikle internet üzerinden genel erişim noktalarından erişimi sağlar. Modelde donanım olarak hesaplama altyapısında sanallaştırılmış donanım kullanılır. Sanal sunucu uzayı, ağ bağlantıları, bant genişliği, IP adresleri ve yerel dengeleyici gibi altyapı kavramları bu modelde kullanılır. İstemcinin kendi IT platformları üzerinde sanallaştırılmış bileşenlerin inşası için erişimine izni verilir[107].

IaaS'ın en büyük avantajı dengeleme ve yeni ortamlara hızlı adapte olmasıyken servisi altyapıyla uğraşmaya itmesi de dezavantajıdır. En iyi model olmamasının nedeni her

uygulamanın farklı önemlikli olması ve gereken güvenlik düzeylerinin aynı olmamasıdır. Bazı uygulamalarda uygulamaya sanal makine eklenmesine gerek duyulmaz.

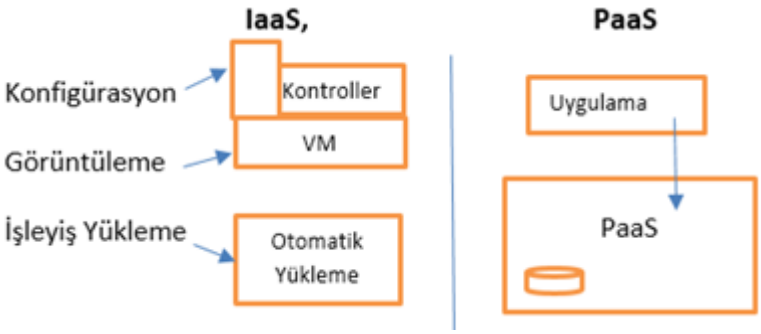
### 8.8.2 PaaS (Bir Hizmet Olarak Platform)

Tüketicilerin araçları veya sağlayıcılardaki kütüphaneler kullanılarak uygulama veya servis oluşturduğu servis modelidir. Tüketici yazılım kontrollerini düzenlemenin yanında yapılandırma ayarlamalarını kontrol eder. Sağlayıcılara örnek olarak network, sunucu gibi tüketicininim uygulamasını sunması gereken araçlardır. Bu model sayesinde donanımı, işletim sisteminin ve depolama araçlarını ve network kapasitesini internet üzerinden kiralanabilir. Servis dağıtıcısı olarak kullanılan bu model tüketicilere sanal sunucuları kiralayabilme ve uygulamayana var olan servislere eklenen yenilerini test etmede kullanılır[108].

PaaS yazılım geliştirme sürecinde baskın bir yaklaşım olacağı iddia edilmektedir. Avantajları olarak süreçleri otomatik yenileme yeteneği, tanımlanmış bileşenleri üretim için otomatik bağlamak gösterilebilirken uygulama performansının altyapı ve donanım ile sağlandığı ortamlara adaptesinin zor olması dezavantajıdır[109].

### 8.9.3 SaaS (Bir Hizmet Olarak Yazılım)

IaaS ile daha çok sistem uzmanları, PaaS (Platform olarak servis) ile uygulama geliştiriciler, SaaS ile de daha çok son kullanıcıları tarafından kullanıldığından en iyi servis modeli amaca uygun olarak değişir. Uygulamanın büyüklüğü küçük ise SaaS tek başına yeterli olabilirken karmaşıklığa göre her üçü de kullanılabilmektedir. Üçünü kullananların koruması gerekenler farklı olduğu için güvenlik mekanizmaları farklı olacaktır. Bu da uygulamanın büyüklüğüne bağlı olacaktır.




Şekil 8. 9 IaaS ve PaaS mekanizmaları

# BÖLÜM 9

# WINDOWS PHONE PLATFORMU İÇİN BULUT KAVRAMI

Bu bölümde önceki bölümlerde incelenen Android platformu için bulut kavramı, mobil istemci inşası, mobil taraflı servis kullanımı, servislere bağlanma kavramlarının Windows tabanlı ortamdaki nasıl olduğu ve aralarındaki farklılıklara değinilmiştir. Eğer bir özellik iki platform için geçerliyse platform ismi belirtilmeyip genellikle farklı durumlarda platform ismi belirtilmiştir.

## 9.1 Windows Phone'nun Özelikleri Ve Bulut İle Etkileşim Ortamı

Windows Phone İşletim Sisteminin amacı mobil uygulama olarak bulut ile iletişimindeki elemanların mimarisinin nasıl oluşturulacağı ve dizaynı ile ilgilenir.

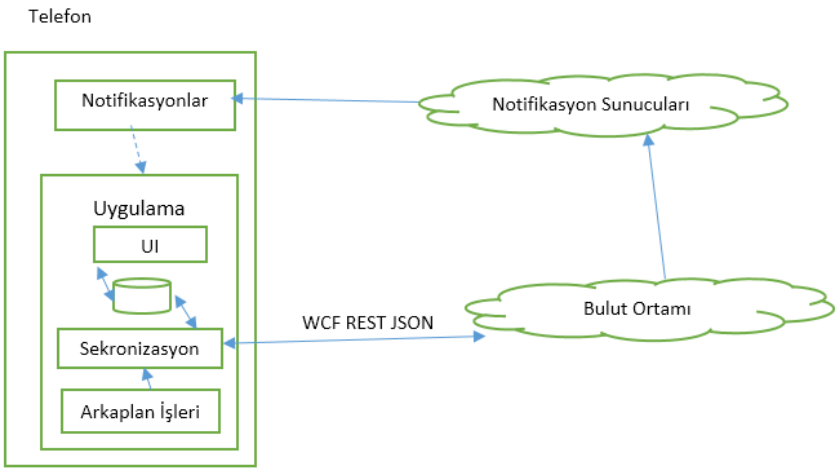
Günümüzde maalesef mobil cihazların birçoğu ancak belirli aralıklarla internete bağlanabiliyor. Bu yüzden mobil istemci uygulamaları buluta veri gönderirken gönderilme işlemi gerçekleşene kadar bu verilerin güvenliği, mobil cihazlarda kullanıcıların kullanım şifrelerinin güvenli biçimde cihaz içinde saklanması, kullanıcıların tanımlayıcı bilgilerinin veriyi buluta veri taşımada kullanılabilirliği gibi konular bu mobil cihazların buluta bağlanmasında cihaza görev düşenlerden bazılarıdır.

Windows taraflı bulut ile iletişime geçen mobil uygulamaların kurulmasında göz önünde tutulması gereken özellikler güvenilirlik, güvenlik ve bağlanılabilirliktir. Android tarafında ise bu dengeleme ve güvenlikti. Her iki platformda aslında kriterlerinde güvenliği ayrı bir başlık olarak tuturak, güvenliğin her ortamda en önemli bileşen olduğu görülür. Bulut ortamının sınırlarının bilinmemesi en önemli özelliği olduğundan her zaman güvenlik en önemli ölçüt olacaktır.

Android ortamında istemci ve sunucular arasında iletişim arasında REST iletişim modeli desteklenirken bu Windows tarafında ise SOAP ve REST desteklenir. SOAP desteklenmesi sayesinde kurumsal uygulamalara daha kolay adapte uygulamalar gerçekleştirilmeside sağlanır.

Her iki platformda bütün servis modelleri desteklenirken, Android'e Iaas ve PaaS daha Microsoft'a ise daha çok SaaS uygundur. Bu sayede Android platformu daha çok uygulamanın alt yapıları düşünülen uygulamalar için uygunken Microsoft ise daha çok üst yapılar (hazır yapılar destekli) kullanılarak yapılması düşünülen uygulamalar için uygundur.

Microsoft Phone'da iletişimde WCF kullanılabilir[159]. WCF sayesinde SOAP ve REST iletişim modelleri desteklenir. WCF'in sağladığı yapılandırma dosyası üzerinde otomatik değişiklik sağlayan arayüz kullanılarak mobil geliştiricilerin dosyanın karmaşasından kurutulu ara yüzün yönlendirmesi sayesinde daha kolay bir biçimde yapılandırma dosyasının oluşturulmasını sağlar. Anroid'de de SOAP kullanılabilir. Fakat güvenlik mekanizmaları daha çok REST için uygundur. Yani Android'de ortam ayarlamaları daha karmaşık olup bu ortamda SAOP kullanılması artık REST'e bir avantaj sağlamayıp kısa mesaj iletişimlerinde dezavantaj olmaktadır.




Şekil 9. 1 Microsoft Phone bulut iletişim ana şeması

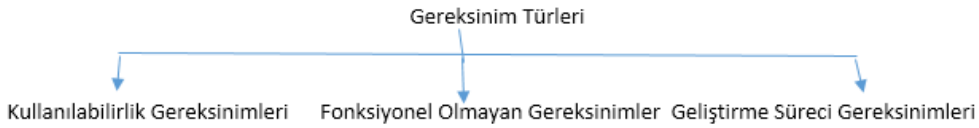
## 9.2 Windows Phone Platformu İçin Mobil İstemci İnşası

Bu bölümde gereksinimlerin belirlenmesi ve sınıflandırılması, mobil istemci uygulamasının bileşenleri, istemci taraflı uygulamaların içyapısı, sayfa yönlendirme, pivot kontrolü ile veri görüntüleme konularına değinilecektir.

### 9.2.1 Gereksinimlerin Belirlenmesi ve Sınıflandırılması

Bu bölümde gereksinimlerin sınıflandırılmasından önce mobil uygulama projelerinde başarı için gereksinimlerin belirlenmesinde uyulacak kriterlere örnekler verilmiştir. Bunlar:

- Genel, jenerik hedefler yerine mobil uygulama için net strateji ve hedefler (iş gereksinimleri) belirleme,
- Proje boyunca iş gereksinimlerine odaklanarak kapsam kaymalarını engelleme,
- Mobil uygulamanın kapsamını belirlerken "uygulama özellikleri ne olsun?" sorusu yerine "hangi kullanıcı ihtiyaçlarını karşılamalıyım?" sorusuna odaklanma,
- Gereksinimleri ve içeriği belirlerken anlık, temel kullanıcı ihtiyaçlarını karşılamaya odaklanma,
- Mobil kullanıcı ara yüzlerinin tasarımında görsellikten önce kullanılabilirliğe öncelik verme,
- Kullanıcı gereksinimleri (use cases), fonksiyonel gereksinimleri ve içeriği belirlerken mobil kullanıma uygunluğu dikkate alma ve vakit alan ve fazla veri girişi gerektiren işlemleri mobile dâhil etmeme,
- Analiz sonrasında ilk örnek tasarımları üzerinden son kullanıcılarla fonksiyonel ve kullanılabilirlik testleri yapma ve kullanıcıların mobil uygulamayı ilk kez kullanıcı kabul testlerinde görmesi olarak sıralanabilir[100].




Şekil 9. 2Gereksinim türleri

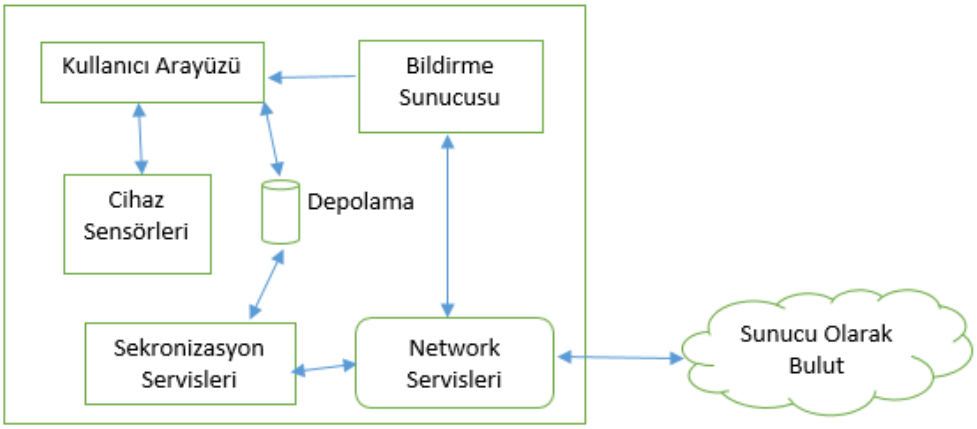
Kullanılabilirlik Gereksinimlerine; ekran büyüklüğü, kullanıcının fiziksel hareketinden etkilenmeme, kullanım alanını bölme, kaynak kısıtlayıcılarını ve limitli kullanıcı deneyimlerini uygun değer kullanma örnek olarak verilebilir[101].

Fonksiyonel Olmayan Gereksinimlerin alt kategorileri ve kategori ile ilgili örnek şöyledir. Kullanılabilirlik (Hızlı erişim menüleri, kısa yollar test edenlerin önerileri dikkate alınarak oluşturulmalı), güvenilirlik (Veri tabanının göçmesi, elektrik kesintisi gibi durumlarda sistem bütünlüğü için veri tabanının loglama özelliğinden faydalanılacak.), performans (Veri tabanı bağlantısı, gelen sonuçların hızı, ağdaki bağlantı kapasitesi ne olursa olsun bilgisayar başında çalışan kişinin fark edeceği şekilde olmamalı), desteklenebilirlik (Projede kullanılan veri tabanının transaction ve foreign key desteğinin olması gerekli), implementasyon (Sistem için çizdiğimiz UML diyagramları, Java kodlarına uyarlanmalı.), arayüz (Kullanıcı düğmelere basarak veya formları doldurarak işlem yapacaktır.) kategorileridir [102].

Geliştirme Süreci Gereksinimleri, platformda karar verme, mobil geliştirme, tarz modelleme ve yaratıcılıktır. Bir mobil strateji kurarken, bu genel hedefleri bilmek önemlidir[103].

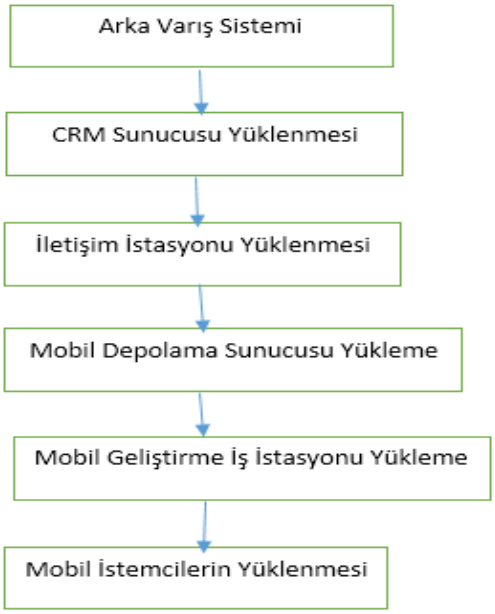
## 9.3 Mobil İstemci Uygulamasının Bileşenleri

Uygulama ortamında üç temel ana bileşen bulunur. Bunlar kullanıcı ara yüzü, depolama alt sistemi ve sekronizasyon sunucusudur. Ortamda Windows Phone platformunun bileşenlerinden kamera, mikrofon, bildirim servisleri ve network sunucularında sunucu servisleri ile iletişiminde kullanılabilir.




Şekil 9. 3 İstemci uygulaması bileşenleri

Şekil 8. 4'de SAP ortamında mobil istemci bileşenlerinin yüklenmesinin şeması gösterilmiştir.



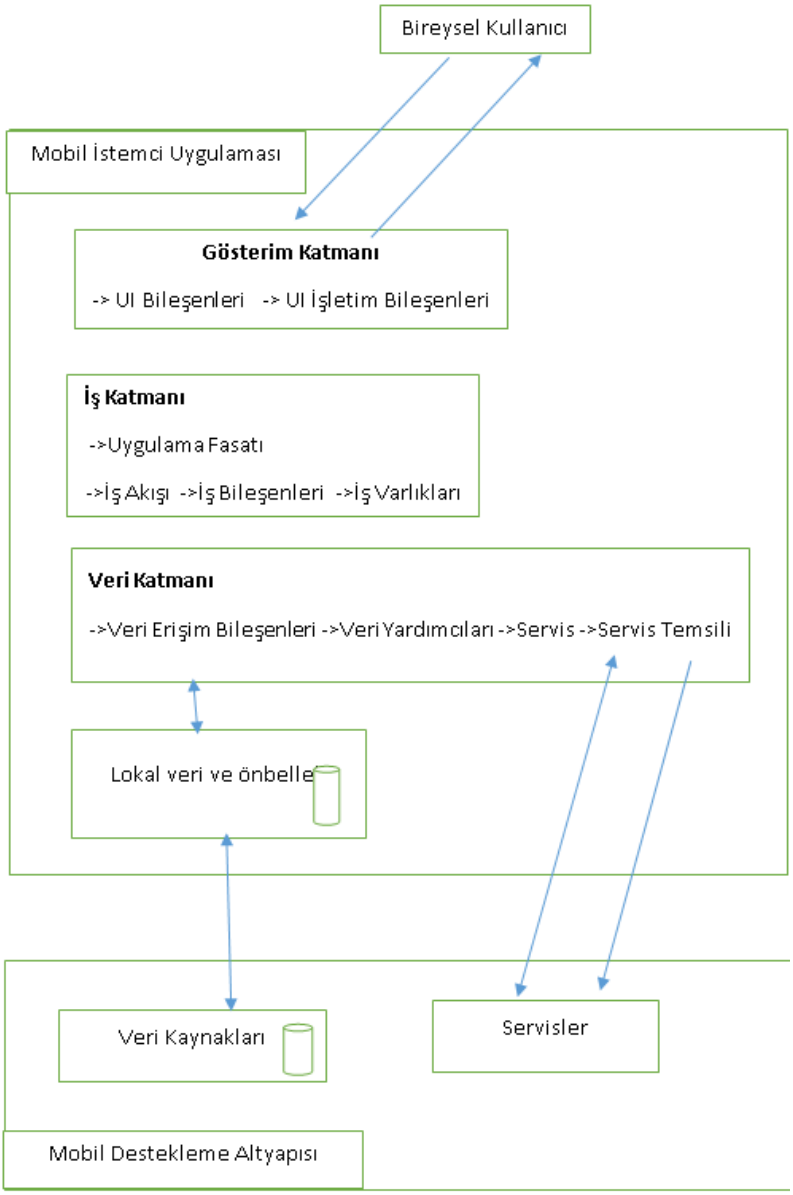

Şekil 9. 4 Mobil istemci bileşenlerinin yüklenmesi[106]

## 9.4 İstemci Taraflı Uygulamaların İç Yapısı

Bu bölümde MVVM tasarım desenine göre bölünmüş istemci taraflı uygulamaların içyapısı yerine (daha ileride bahsedilecek) katmanlamaya göre bölünmüş şekli Şekil 9. 4'te gösterilmiştir.

Bir mobil uygulama kullanıcı deneyimi, iş ve veri katmanları olarak çoklu katmanlama olarak inşa edilebilir. Bir mobil uygulama geliştirildiğinde istemcileri basit Web bazlı istemci veya zengin istemci olarak belirleyebilir. Eğer zengin istemci oluşturulursa, iş ve veri servis katmanları cihazda saklanır. Eğer basit istemci oluşturulmak istenirse iş ve

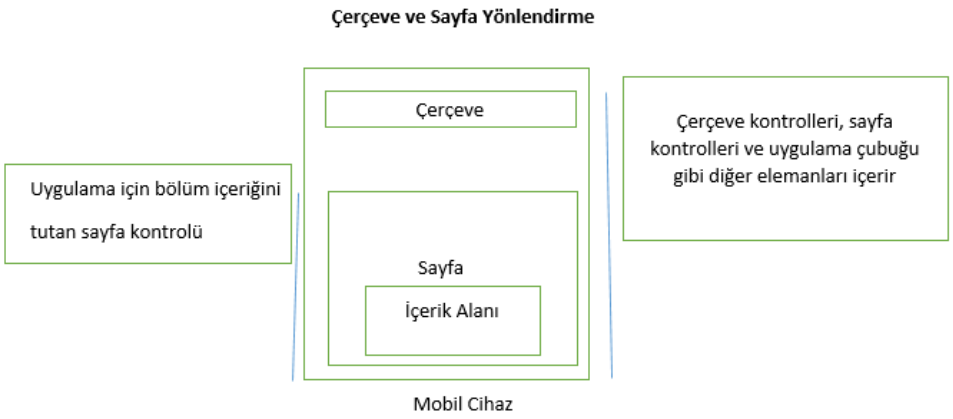
veri katmanları sunucuda olacaktır. Aşağıdaki şekilde zengin istemci için oluşturulmuş istemci taraflı uygulamaların içyapısı şeması gösterilmiştir.




Şekil 9. 5 Windows Phone platformunda istemci taraflı uygulamaların iç yüzü [105]

## 9.5 Sayfa Yönlendirme

Sayfa yönlendirme, kaliteli uygulama inşasında dikkat edilmesi gereken önemli bir açıdır. Çünkü çoğu zaman uygulamalar tekil (tek uygulama) değildir. Uygulamalarda sayfalar arası geçiş yapılır ve bunu farklı stiller için kullanılır. Aşağıdaki şekilde sayfaların yerleştirilme hiyerarşisi gösterilmiştir.




Şekil 9. 6 Çerçeve ve sayfa yönlendirme

Çerçeve en üstte bulunan konteynerdir. Uygulamalarda App.xaml.cs dosyasına bakılırsa önyükleme zamanı olarak uygulamanın ilk yaratacağı nesne çerçeve nesnesidir. Çerçeve içerikleri uygulamadaki tüm sayfaların ve diğer kontrolleri normalde programlanmaz ve sadece gösterilir. Bu daha sonra bütün çerçeveye döndürülür.

Windows Phone'deki XAML uygulamaları web sayfa modeli ve URI tarafından tanımlanan her sayfaya benzer bir biçimine benzeyen sayfa yönlendirme modelini kullanılır[117].

## 9.6 Kullanıcı Ara yüzü Tanımlama

Windows Phone 2. 0 sürümü ile kullanıcı ara yüzü ve iletişim yönlendiricinin modası artık geçmiştir. Kullanıcı ara yüzü tanımlama oluşturulurken altı ilke takip edilmelidir. Bunlar:

1) **Windows Phone Platformu:** Windows Phone kullanılarak oluşturulabilecek uygulama tipleri belirlenmeli ve her bir türün özel kullanıcıları bilinmeli ve yapılacak uygulama türünün hedef kitle ile uyumu iyi olmalıdır.
2) **Windows Phone İçin Uygulama Dizayn İşlemi:** Mobil cihazların uygulama dizaynı cihazlardaki varlıklıların etkilerinin nasıl olacağına dair fikirleri içerir.
3) **Windows Phone İçin Uygulama Mimarisi ve Yönlendirme Modelleri:** Platformun farklı uygulama yönlendirme modelleri ve her stil için en iyi uygulamaların farklı türleri hakkında bilgileri içerir.

4) **Windows Phone İçin Kullanıcı Ve Platform İletişimleri:** Windows Phone için belirli alanlar ve kullanıcı iletişimlerin yönlendirmeleri içerir. Bu yönlendirmelerine şemalar, animasyonlar, yönlendirme, hizalama, grafikler örnek verilebilir.

5) **Windows Phone İçin Kontrol Dizaynı Kontrolleri:** Windows Phonede özel kontroller gibi aynı platformda Silverlight kontrollerinin de yol göstericileridir.

6) **Ek:** Diğer tasarım kaynaklarına (araç kutusu, simgeler, tasarım örneği ve öğretici) bağlantılar içermelidir[118].

## 9.7 Kullanıcı Ara yüzü Elemanları

Bir uygulama standart kontrollerden oluşup uygulamanın görünüm ve davranışı Windows Phone'nın standart görünüm ve davranışı ile eşleştirilir. Bu bölümde elemanlardan Pivot (eksen) kontrolüne değinilmiştir.

### 9.7.1 Pivot Kontrolü (Uygulama İndeks Etiketi)

Pivot kontrolünü kullanıcıların aralarında gezindiği kullanıcı ara yüzü için çoklu sayfalarda kullanılır. Bu stil şeması uygulamada kullanılması pratik olarak uygundur.

Her sayfa uygulamada gösterilen kullanıcı ile ilgili veriler gösterilir. Uygulama indeksi etiketi stili bitirilmiş uygulamayı zenginleştirir veya uygulamada alt alanda kullanılır. Örneğin uygulamanın birinci seviyesinde panorama kontrolüne sahip olunabilir. Uygulama indeksi stili ile kullanıcının alt bölümlere yönlendirilmesinde kullanılır[119].


Şekil 9. 7 Uygulama indeksi etiketi filtreleme ilişkisi

## 9.8 Veri Görüntüleme

Günümüzde mobil ve bulut hesaplama teknolojileri eskiden bilgisayar dünyası denince akla gelen PC'nin yerini almaya başlamıştır. Yani PC'de yapılan her şey bu teknolojiler ile de yapılmak istenir. Hesaplamalı devretme (Android platformu) ve veri bağlama (Windows Phone platformu) akıllı telefon gibi düşük enerjili cihazlar için elastikiyetlerini arttırmak için besleme (saklama) yapan temel teknolojilerdir.

Hesaplamalı devretme kod seviyesinde işlenir. Bir mobil uygulama genellikle belirli kısımlara (metot ve sınıflar gibi) bölünür ve önsel veya çalışma zamanlı olarak analiz edilir. Tanımlanan ve kapalı yüklenen çoğu hesaplama işlemi pahalı (yazılımsa ve donanımsal olarak) operasyonlardır. Bir mobil operasyon devretmeli olabilir veya olmayabilir. Bu mobil kaynaklar üzerindeki etkiye bağlıdır.

Devretme isteğe bağlı bir işlem olup, eğer mobil operasyonun yüksek hesaplama işlemini isteyip aynı zamanda iletişimde az miktarda veri iletişiminin olduğu durumda idealdir. Diğer durumda eğer devretme desteklenmezse buluta veri gönderme işinde daha çok enerji ve harcanılan zaman miktarı artar. Bunun yanında hesaplamalı devretme uygulamaya eklenen eklentiler (resim, video işleme teknikleri) tarafından tanımlanır.

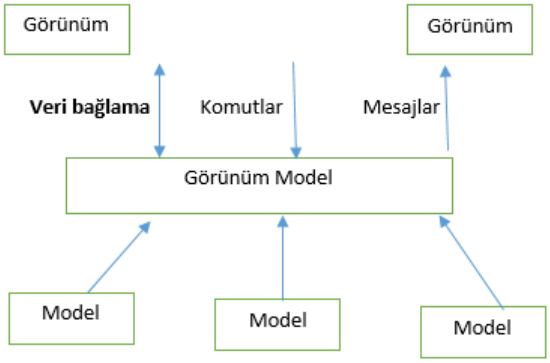
Veri bağlamada işleme farklı yöntemle yapılır. Bulut ile düşük enerjili cihazlar arasındaki veri bağlama farklı şemalar üzerinden gerçekleştirilir. En çok kullanılan şema Web API üzerinden buluta erişimin her zaman devam edileceği varsayımına dayanan servis bazlı entegrasyondur. Görev temsilcisi, servis tarafından direk olarak asenkron mobil işlerinde kullanılır. Çevrim dışı cihazlar için temsilcisiz işler bu iş doğalında zaman tüketen işler olup programlanabilir ve paralellenebilir olup çoklu sunucularda çalışması iyi değildir. Fakat temsilci iş belirli bir iş üzerinde odaklanıp ortama özel işler yapmayı sağlar. Burada amaç operasyonun daha yoğun olarak cihazın içeriği ile olmasını sağlamak gibi görünse de tersine bağlamadan sonra verinin yayınlanması devretmede opsiyonel olup bağlamada zorunlu bir iş olmaya başlar.

Mobil işlemlerin temsilciliği veri bağlamını yayımlamada harcanan gücü analizi olmadan herhangi bir zaman kullanılabilir. Diğer bir anlamda veri bağlama yerel ve uzak yerlerle iletişiminde hangi iletişim algoritmasının (veri kapalı yükleme de var) kullanılacağını arayüz mantığında ki algoritmaları ile karar verir. Fakat bu mantıkta en önemli amaç mobil ağda veri trafiğinin en aza indirilmesini sağlar. Bu yüzden mobilde en gerekli iletişim işleminin arttırılması gerekir. Dolayısıyla cihazların enerji ömürlerini arttırması gerekir.

Veri bağlama teknikleri genel olarak alan saklamayı arttırmak (DropBox gibi), veriyi tekrarlama ve merkezileştirme (Fabl gibi), var olan uzaktan algoritmaları yüksek veri

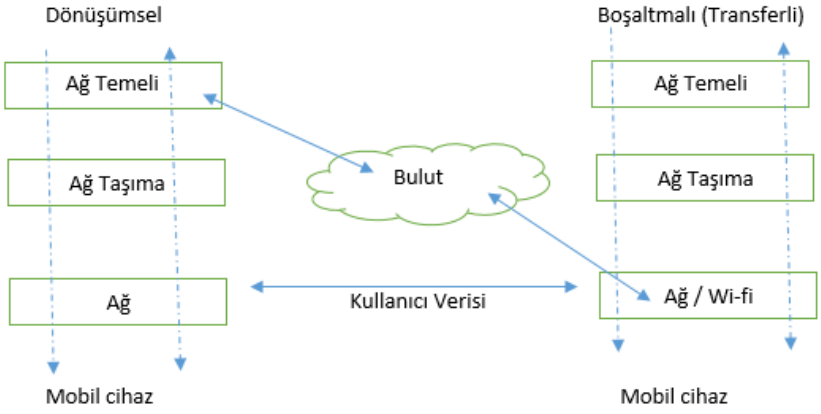
depolarıyla uyumunun ayarlanması (aktif tanımlama – mobilden doğrulama) sayılabilir[109].

Veri bağlama uygulama bazlı mimarisel seçenekleri ve iletişimi belirler. Hesaplamalı kapalı yüklemenin aksine veri bağlama daha büyük gereksinimleri olan uygulamalarda daha uygundur. Windows Phone'nun daha büyük verilerle çalışmasının daha uygun olmasının tek nedeni bu değildir. WCF'in iletişimde kullanılabilirliği ve SOAP'ın desteklenmesi de sayılabilir.




Şekil 9. 8 Windows Phone'da MVVM deseni şeması [111]

Her iki yönteminde bulut gücüyle mobil cihazların uyumunu ve kapasitesini arttırmaya çalışırken ikisinde farklı sorunlara ve farklı teknolojilerin desteklenmesinin mobil cihaz için altyapısına yoğunlaşmıştır.

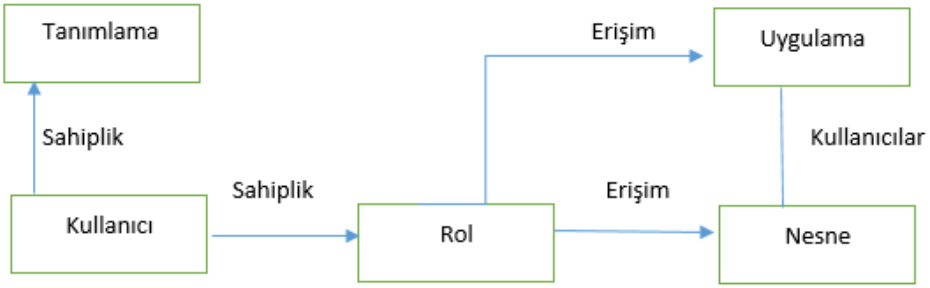



Şekil 9. 9 Saklı yükleme yöntemi ile mobil cihaz uyumu şeması

## 9.9 Servislere Erişim Modelleri

Günümüzde bulut için temsilcili ve saklı yüklemeli mobil operasyonlarının bulut hesaplama için uyumu giderek artmıştır. Mobil temsilcide, çoklu bulut problemlerini çözmek için (SaaS seviyesinde) çözmeyi ister. Daha karmaşık çözümler çoklu bulutların uyumu sorununda adresleme ile çözmeye çalışır. Mobil şeffaf (görünür) temsilciliği ve

asenkron düzenleme işleri bulut altyapısının kaynak yoğunluk işleme ve dinamik alanıdır. Tersine, çoğu çalışma mobilden bulut çevresi ile ilgili saklı yükleme yönü ve çözüm için adresleme sorunu olarak saklı yüklemenin mobilden buluta ne, ne zaman ve nasıl olacağıdır.




Şekil 9. 10 Erişimde rol, sahipli ve erişim ilişkisi [110]

### 9.9.1 Bulut İçin Saklı Yüklemeli Mobil İşlemler

Kodun saklı yüklenmesi, mobil cihazların bulut etkileşimi konusunda özellikle akıllı telefonların çıkmasıyla önem kazanmıştır. Mobilde kod başlatılması temel olarak, mobil uygulamayı bölerek uzaktan işletime hazır hale getirmede ve buna geliştiricinin karar verebilmesidir. Bu sayede uygulamaya yeni ihtiyaçlara uygun eklemelerde en az değişiklik yapılmasını sağlar. Yani sadece önceden bölümlenmiş uygulamanın sadece otomatik olarak ilgili bölüm güncellenir. Bu otomatik güncellemenin analizine saklı yükleme karar motoru karar verir. Bu makine bir mobil bileşenine saklı yükleme uygulanıp uygulanmayacağına karar verir.

Temel olarak karar motoru cihazın çoklu yönüne uygun olarak (bağlantı bant genişliği, verini boyutu, vb.) profiller ve motorun ölçebileceği şekilde (liner programlama gibi) uygular. Bu sayede bulut ve cihazla oluşan iletişim ortamının uygun değer gereksinimleri (gereken enerjinin az olması, sürenin mümkün olabildiğince az olması gibi) olur.

Mobil uygulama bölümleme otomatik mekanizmalarda esnekliği sağlamak için çalışma zamanında aynı uygulamalarda farklı cihazlarda donanım özellikleriyle tercih edilir. Burada amaç var olan gelişimin aynı anda özel bir cihaza gereksinim olduğunda gereklidir. Yani uygulamanın değişmeyip, güncellemelerin bu uygulamaya göre yapılması gerekir [112].

### 9.9.2 Bulut İçin Devredici Mobil İşlemler

İş devretme günümüzde herhangi bir platformda çoklu mekanizma (Web soketleri, REST bazlı istekler, vb.) ile kullanılan bir operasyon olmuştur. Farklı mobil platformlar veya aynı platformun farklı versiyonları ağ iletişimini yönetmede farklı ilkeler edinmiştir. Örneğin Android 10. seviye platforma REST bazlı istekler asenkron işletilirken 15. seviye ve üstü versiyonlarında geliştirici eğer farklı bir ortamda bulunan iş ile ağ iletişimi kurmak isterse AsyncTask (asenkron iş) sınıfını kullanması gerekir.

Bulut isteği bazen zaman tüketici bir iş olabilir. Bu yüzden iletişim için farklı uygun bir mekanizma ile iletişime ihtiyaç duyar. Yoksa bu mobil kaynaklarının israfına yol açabilir.

Bulut fonksiyonelliğinin mobil ile entegrasyonunda farklı Web API'leri farklı bulut vektörlerini doğal (native) mobil platformunda kullanılır. Vektörler genel olarak Web API'lerini dinamik programlamanın altyapısını oluşturan paralel programlamayı destekleyen bir arayüz olarak kabul edilir. Bir Web API'nın mobil cihazda gösterilmesi derleyici limitleri, kodun bağımsızlığı gibi nedenlerden dolayı mobil işletim sistemlerine göre değişir. Bu yüzden çoğu zaman geliştirim başarılamaz. Tersine bulutla iletişime geçecek yazılım uygulamaları çoklu Web API'ler ile iletişime zorlanır. Web servisleri genellikle SaaS seviyesinde sağlanarak bulutun fonksiyonelliğinin arttırılmasıyla sağlanır.

# BÖLÜM 10

# WINDOWS PHONE PLATFORMU VERİ İLETİMİ

Bu bölümün esas konusu Windows Phone platformu ile oluşturulan servislerin istemciler tarafından kullanılmasıdır. Mobil istemcilerin verinin devamı olarak izole depolamayı nasıl kullandıkları ve mobil cihazda veri güvenliği konusu işlenmiştir. İşletim sisteminin aktif olmaması durumunda uygulamanın yeninden aktif duruma döndürülmeye çalışıldığında gerekli olan durum bilgisini nasıl sakladığı konusu işlenilen bir başka konudur.

Bulut servisi bazen de istemciden veri alıp, istemciye veri döndürür. Bu bölümde uygulamanın cihaz ve bulu arasındaki veri uyumuna da değinilmiştir. Ön planda ve arka planda çalışan işlerin uyumu değinilen başka bir konudur.

## 10.1 Mobil Cihazda İzole Depolamanın Kullanımı

Verinin yerel olarak saklanmasının iki yolu vardır. İlk yol isim/değer çifti şeklinde biçimlendirilen İzole Depolama Ayarlayıcı (Isolated Stroge Setting) sınıfını kullanmaktır.

Uygulamalarda çoğu amaç için sadece istenilen az miktardaki veriler için depolama yapılmak istenebilir. İzole Depolama Ayarlayıcı sınıfı belirli bir klasörde (yükleme olmaksızın) verilerin isim değer çifti olarak saklanabilmesine olanak sağlar. Bu veriler uygulamanın başlaması, sonlanması, enerji kesilmesi gibi olaylardan etkilenmez. Kullanıcı uygulamayı kaldırdığında ya da kullanıcı bu verileri silindiğinde bu veriler silinir. Kütüphaneye eklenmeden bir veriye erişilemez.

Bu sınıf kullanılırken dikkat edilmesi gereken şeyler şunlardır:

- Eğer sınıftan eklenmemiş bir veri çekilmeye çalışılırsa bir hata (istisna değil) ile karşılaşılır.
- Ayarlamalar kısmında istenilen veri türünde veri kaydedilebilir.
- Veri döndürüldüğü zaman açıkça türü belirtilmesi gerekir.
- Ayarlama değeri kütüphaneye ekleme değeri ile aynıdır[113].

### 10.1.1 Güvenlik

Windows Phone'de genellikle kullanıcı şifreleri şifreli metin olarak izole depolarda kullanılır. İzole depolama cihazdaki diğer uygulamalardan farklı olarak birinci seviye veri güvenliği sağlar. Fakat önce uygulamanın gerekli güvenlik düzeyi belirlenir.

Cihazda veri saklanmasında iki adet implementasyon güvenliği bulunmaktadır. İlk tip diğer uygulamaların potansiyel olarak uygulamanın verilerine erişip başkasına taşıma için kullanmasıdır. Windows Phone'da izole saklama modeli kendi saklama sistemini kullanacak limitli uygulamalarda kullanılır. Fakat uygulamadaki verileri saklamak için izole saklama kullanmak beklenmeyen sonuç doğrulacağı düşünülüyorsa kullanılmamalıdır. Örneğin çok miktarlı veri (Limitli kullanılabilir. Miktar limite yaklaştıkça hız yavaşlar.) kullanıldığında, verilerin daha detaylı olarak işlenmesi gerektiğinde kullanılmamalıdır.

İkinci tipte cihazdaki uygulamaya kimlik doğrulama olmadan kullanıcıların erişimi durumudur. Bu durumda cihazda veri şifreli olarak saklanmalıdır.

Windows Phone API'si Veri Koruma API'sine erişim sağlayabilir. Veri Koruma API'si kullanıcı ve telefon sağlayıcıları (şifreleme ve de şifreleme için) kullanılarak kriptografik anahtar saklanır. Veri koruma sınıfı Veri Koruma API'sine erişim sağlar ve korumasız (unprotected) metotları kullanır.

Mobil cihazda her uygulama kendi de şifreleme anahtarına (uygulama ilk defa çalıştırıldığında üretilen) sahiptir. Korumalı ve korumasız metotlar de şifreleme anahtarını kullanır ve uygulamadaki veriler (şifreli olarak) uygulamaya özgüdür. Ek olarak, de şifreleme anahtarı uygulama güncellemelerinde taşınır[114].

## 10.2 Uygulamanın Aktif Olma Ve Aktif Olmama Durumu

Bu bölümde Windows Phone uygulamalarının yaşam döngüsü ve uygulamanın aktif olma ve aktif olmama durumlarını işleyişi incelenmiştir. Windows Phone'de esas zamanlı olarak aslında tek uygulama çalışabilir. Bu uygulamanın düzgün olarak cevap verebilmesi ve uygulanabilir olması için önemlidir. Çünkü sadece bir uygulama ön planda çalışır ve kullanıcı başka bir uygulamaya geçtiğinde yani yeni bir uygulama çalıştırmak istediğinde eski uygulamanın bilgilerinin nasıl saklanacağı ve istenildiği zaman çalışmaya nasıl hazır olacağı bu bölümün konusudur. Windows Phone platformu modelleri, API'ler ile ilişkili olan kullanıcılara aktif olma ve aktif olmama durumları kullanıcının deneyimlerinde yaralanarak tutarlı ve sezgisel olarak işlemesine izin verir.

## 10.3 Asenkron Çağrılar

Asenkron programlama normalde çok geniş bir konudur. Burada kısa bir özet geçilmiştir. Web istek ve web yanıt konularına daha önceki bölümlerde incelenmiştir. Bir asenkron çağrısı şöyle yapılır:

Asenkron başlatma metodu temsili olarak veya istenirse bazı durum bilgileri geri dönüşlü metodu çağrılır.

Arka planda çağrı istenildiğinde kontrol uygulamayı arka planda işler. Bu sayede kullanıcı ara yüzü duyarlılığını devam ettirir. Kullanıcılar geri dönüşleri beklemeden diğer işleri işleyebilir. Bu durum web istemci tarafından meydana getirilen olayın beklenmesine benzer.

Geri dönüşlü metot çağrıldığı zaman sonuç elde etmek için başlama metodu tersine sonuç metodu çağrılır. Bu metotun gönderilme değeri olarak asenkron sonuç değeridir[115].

Yukarıda belirtilen adımlardan başka daha çok çeşitti mevcut olmasına karşın bu bölümde bu özetine değinilmiştir.

## 10.4 Windows Phone ve Bulut Arasında Asenkron Veri İletimi

Mobil ortamda uygulamalar genellikle çevrimiçi ve çevrim dışı ortamlarda çalışmak için üretilirler. Bulut veri sekronizasyon mekanizması sayesinde bu mekanizma ile oluşturulmuş bir uygulama internetin olmadığı bir ortamda çalıştırılmak istenildiğinde internet yok mesajı ile karşılaşılmaz. Bulut veri senkronizasyonu sayesinde yerel veri tabanları ve karmaşık senkronizasyonlar ile uğraşmak zorunda kalınmaz. Veriler bulutta tutulur. Esasen bahsedilen zor işleri bu mekanizma kendisi yapar.

Bulut veri sekronizasyon mekanizması sayesinde bulut nesnelerinin yerel depolamada saklanmasına izin verilir. Lokal ve bulut depolamanın eşzaman olması sağlanır[128].

Mekanizma ile herhangi bir bulut nesnesi ve herhangi bir mobil uygulamada kullanılabilir. Bağlantı uygun ise veri sunucu ile senkronize olması beklenir.

Otomatik ve manuel olmak üzere iki çeşit sekronizasyon türü bulunmaktadır. Otomatik sekronizasyon mobil istemci uygulaması ve bulut uygulamasını arka planda temsilcili olarak işletilir. Arka plan temsilcisi uygulama arka planda çalışırken veya çalışmazken işler yürütülür.

Manuel sekronizasyon, yöneticinin sunucu ve istemci arasındaki ver transferinde kendi kurallarına göre yönetimini belirlediği ve gelen verilerinde uygulama içinde çalışmasının düzenlendiği sekronizasyon türüdür[120]. Kullanım alanının dar olduğu düşünülmesine karşın aslında geniştir.

# BÖLÜM 11

## WINDOWS PHONE İÇİN SERVİSLERE BAĞLANMA

Bu bölümde mobil istemcilerin dışsal servislerle (uygulama servisi ve üçüncü kısım servisler arasında ki) iletişiminin çeşitli yolları incelenmiştir. Mobil istemcilerin dışsal servislere bağlantısında bir takım tasarım prensipleri ve kurallar bulunur. Windows Phone uygulamlarının en çok bağlandığı bulut servisi Azure'dir.

Mobil istemci uygulamasının şu özelliklere sahip olması beklenir:

- Değişken ağ bağlantısında güvenilir olmalıdır.
- Ağ bant genişliğinin mümkün olduğu kadar az olması gerekir.
- Uygulama mümkün olduğu kadar az güç harcaması gerekir.

Çevrimiçi servis bileşenlerinin şu özelliklere sahip olması beklenir:

- Uygun seviye güvenlik sağlamalıdır.
- İstemci uygulamalarının hazırlanmasında kolay geliştirilebilir olmalıdır.
- Çeşitli istemci platformlarını desteklemelidir.

### 11.1 Servis İçin Kimlik Doğrulama

Bir mobil uygulamadaki istemcilerin bulut servislere bağlanmasında kimin servise bağlandığının kontrolü yani kimlik doğrulama çeşitli nedenlerle kullanılmaktadır. Burada amaç uygulamayı hangi istemcileri kullandığının belirlenmesidir. İstemcilerin servislerdeki hangi kaynaklara erişeceği başka bir konu olup buna yetkilendirme denir.

Mobil uygulamalar geliştirilirken yapılacak güncellemeler karşısında kimlik doğrulama mekanizmalarının çok değiştirilmemesi beklenir. Bunun yanında uygulamanın kimlik doğrulama ve yetkilendirme konularınızdaki gelişmelere karşı güncellenmesi bile istenir. Bu sayede uygulamanın esnekliği artar. Yeni gelişmelere uygulanabilir olur. Mobil uygulamaların farklı platformlarla uyumu zordur. Bunun için bulut servisleri kullanılır. Çünkü bulut servislerinde iletişim mesaj formatı belirli bir standart olduğundan bu formatta mesaj iletişimi kabul eden yeni bir platformda yaratılabilir.

Günümüzde bulut depolama platformlarında kullanıcıların özel bilgileride saklandığından kötü niyetli kullanıcılar tarafından saldırıya maruz kalmaktadır. Burada platform veya kullanıcının hatası olabilir. En son yapılan saldırılarda kullanıcının hatalı olduğu düşünülmektedir. Fakat bu saldırılarda benzer diğer uygulamalar veya daha güvenlik gereken uygulamalarda bu saldırılar karşısında kendi güvenlik mekanizmalarını değiştirmek isterler[119]. Çoğunda var olan güvenlik mekanizmasında sorun olmaz ama şüphesi bile diğer uygulamaları etkilemiş olur.


Şekil 11. 1 Windows Phone uygulaması ile Google OAuth2 bağlantısı [120]

Mobil cihaz kimlik doğrulama ve yetkilendirme mekanizması için normalde kullanıcı ismi ve şifre kullanılır. Mobil istemci sağlayıcıların tutarlılığı kontrol edilemez. Kullanıcı uygulama yönetim sayfasına yönlendirilir. Şifre ise izole depolamada saklanmadan önce şifrelenir. Uygulamalarda istenilen düzeye göre şifrelerin düzeni belirlenebilir.

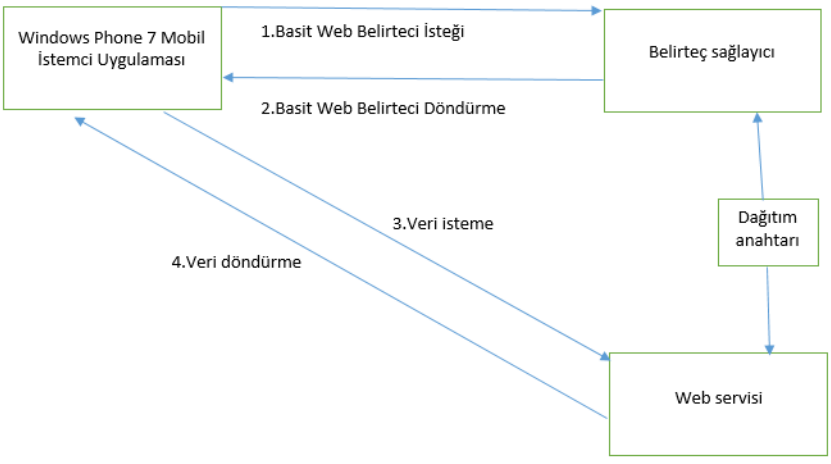
Mobil taklit (mock) kimlik doğrulama kullanılarak kullanıcı ismiyle tanımlanmış şifre bilgisinin kullanımını kontrol edilebilir. Uygulama kimlik doğrulama metodu değiştirilmek istenirse şu özelliklere dikkat edilmelidir:

1. Mobil istemciye sağlayıcı göndermek için geleneksel HTTP başlığında düzenlenmelidir.
2. Kimlik doğrulama modülü eklemek için sağlayıcılar doğrulanmalı ve sağlayıcı prensibi nesnesi yaratılmalıdır.

### 11.1.1 İddia Bazlı Kimlik Doğrulama Mekanizması

Mobil cihazlarda kullanılan kimlik doğrulama mekanizmalarından biri kullanıcı ismi ve şifre bilgileri ile oluşturulan iddia bazlı yaklaşımdır. Bunun bir yolu da SWT ve OAuth 2. 0'ı protokolünü kullanmaktır. Bu protokollerin kullanımının faydaları şunlardır:

- Kimlik doğrulama işlemi uygulamadan ayrı bir hazır araç (üçüncü kısım uygulamalar gibi) sayesinde yapılır.
- Kimlik doğrulama işlemi belirli standartlar üzerinden yürütülür. Geliştirici kendi standartı belirlemediği için esas ürün üzerine odaklanır.
- Protokoller üzerinde güncelleme yapıldığında otomatik olarak uygulamada da güncelleme olmuş olur. Geliştirici bu durumda etkilenmez.




Şekil 11. 2 Windows Phone web servisinde kimlik doğrulama için SWT kullanımı [127]

Dezavantajı olaraktan kullanılan protokollerin güçlü olmasına karşın kötü niyetli kullanıcıların kendilerini kötü olarak tanıtabilmesi için ilk olarak bu prokollerin

açıklıklarını bulmaya çalışacakları açıktır. Çünkü bu sayede saldırabilecekleri uygulama artar.

Senaryoda, mobil istemci uygulaması web servisini SWT içeren iddia ile çağrılır. Bu iş sağlayıcıya bir istek belirteci gönderilerek yapılır. Sağlayıcının yukarıdaki örnekte SWT'yi tanıması zorunludur. Bu istekte istemci tanımlayıcısı, istemci sırrı, kullanıcı ismi ve şifre bilgileri yer alır. Anroid platformunda istemci sırrı bilgisi normalde kullanılmaz. SWT, Android platformunda servis oluşturulmuş olup Windows platformunda istemcinin bulunduğu zaman kullanılabilir.

İstemci tanımlayıcısı ve istemci sırrı sağlayıcının uygulamanın isteğinin SWT olup olmadığını tanımlar. Sağlayıcı kullanıcı ismi ve şifreyi kullanıcının kimlik doğrulamasında kullanılır.

Belirteç sağlayıcısı SWT'yi kullanıcı tanımlayıcı olarak kullanılacak ve diğer iddialarını tüketici uygulamaları için inşa edilir. Ayrıca sağlayıcı sır anahtarını servis ile paylaşmak için kullanılır. İstemci uygulaması servisten veri istediğinde isteğin kimlik doğrulama başlığındaki SWT'ye bakılır.

Servis isteği aldığında SWT kimlik doğrulama modülünden kimlik doğrulama başlığı çıkarılır. Son olarak servis kimlik doğrulama için iddiaları kullanarak verinin ne olduğuna karar verilir. Yanıt istemciye gönderilir.

### 11.2 Windows Phone'da Notifikasyonlar

Windows Phone'da yerel veya bulut servisi kullanılarak notifikasyonlar yaratılır veya programa konulur. Notifikasyonlar başlangıç ekranında, alarm göstermelerinde ve hatırlatıcılarda Toast gönderme ve Tile (Türkçe karşılıkları karşılamıyor) güncellemeyi için gönderilir.

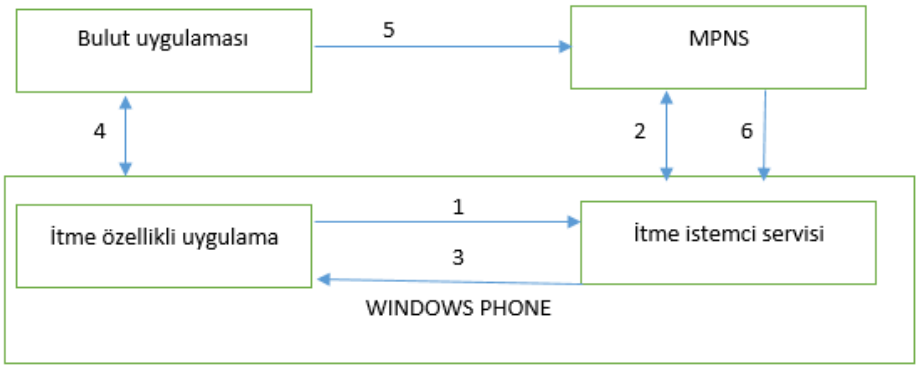
**Lokal notifikasyonlar:** Tile ve Toast notifikasyonu için yaratılan notifikasyonlardır. Bir zaman veya zaman arlıklarında tekrarlanması için alarm veya hatırlatıcı yaratmak için kullanılır.

**İtme notifikasyonları:** Tile'ın güncellemesi için Toast notifikasyonu yaratır veya bulut servisinden itme notifikasyonlarını kullanarak uygulamaya hammadde verisi gönderir. Bu notifikasyonlar sayesinde Tile güncelleme, Toast notifikasyonu yaratma ve bulut

servisinden itme notifikasyonları kullanarak hammadde verileri uygulamaya yönlendirilebilir.

### 11.2.1 İtme Notifikasyonları (Bildirme)

Windows Phone'daki Microsoft İtme Notifikasyonu Servisi üçüncü parti uygulama geliştirmek isteyen geliştiriciler için asenkron çalışan bir servistir. Bir kanal üzerinden verimi bir şekilde mobil cihaz ve bulut servisi arasında veri iletişimini sağlar. Aşağıdaki şekilde itme notifikasyonunun iletimi gösterilmiştir.




Şekil 11. 3 Windows Phone'da itme notifikasyonu iletimi [121] (Şekilde ayrıca itme notifikasyonları kullanım ortamının bileşenleri gözükmektedir.)

1. Uygulama itme istemci servisinden itme notifikasyonu URI'si istenir.
2. İtme istemci servisi, MPNS ile birlikte çalışır (müzakere eder) ve itme istemci servisine notofikasyon URI'si döndürür.
3. İtme istemci servisi, uygulamaya notifikasyon URI'si gönderir.
4. Uygulamadan bulut servisine notifikasyon URI'si gönderilir.
5. Bulut servisi uygulamaya bilgiye gönderildiği zaman bu notifikasyon URI'si MPNS'e itme notifikasyonu gönderir.
6. MPNS, uygulamaya itme notifikasyonunu yönlendir.

### 11.2.2 Notifikasyon Gönderimi

İtme notifikasyonları kanal URI'ye HTTP POST mesajı yollanarak iletilir. Mesaj başlıklarına örnek olarak mesaj yapılandırma notifikasyonu ve notifikasyonun mesaj gövdesi (title ve toast notifikasyonu gibi) olarak örnek verilebilir. Merkezi itme

notifikasyonunun statüsünü bilgilendirme için başlıklarının dâhil olduğu bir mesajlar karşılık verir. İstek ve yanıt arasındaki uyum için tanımlayıcı başlığı kullanılabilir.

Bir web servisine veya uygulamaya iletim notifikasyonlarını gönderme kriterleri şunlardır:

- Notifikasyon gönderilmek istenen her bir Windows Phone cihazı için POST mesajı yaratılmalıdır.
- Uygun notifikasyon tipine uygun olarak mesajlar formatlanır. Bir uygulama sunucuya aynı anda birden fazla notifikasyon tipi iletemez. Sadece birini iletebilir. Eğer çoklu notifikasyon tipleri aynı istemci cihazına aynı zamanda gönderilmek istenirse, her bir notifikasyon türü için POST mesajı yaratılmalıdır.
- Windows Phone 7. 0 işletim sistemi istemcileri için mesaj özellikleri olup bu işletim sisteminde desteklenmeyenler eklenmez Mesajın derecesi düşürülür. Böyle durumun sorun yaratmaması için geliştiricilerin web servisi üzerinden iletişimde mesaja işletim sistemi sürüm bilgisi eklenmelidir.
- Mesajlar, iletişim notifikasyon servisine iletilmelidir[122].

Notifikasyon açısından iki platform arasında bazı isim değişiklikleri olsada esasında çok farklı bir mimari farkı yoktur. Anroid daha kontrolcü olup servletleri kullanmasına karşın, ürün bazlı Windows Phone'de sunucular kullanılır.

### 11.3 İtme Notifikasyonları (Bildirme)

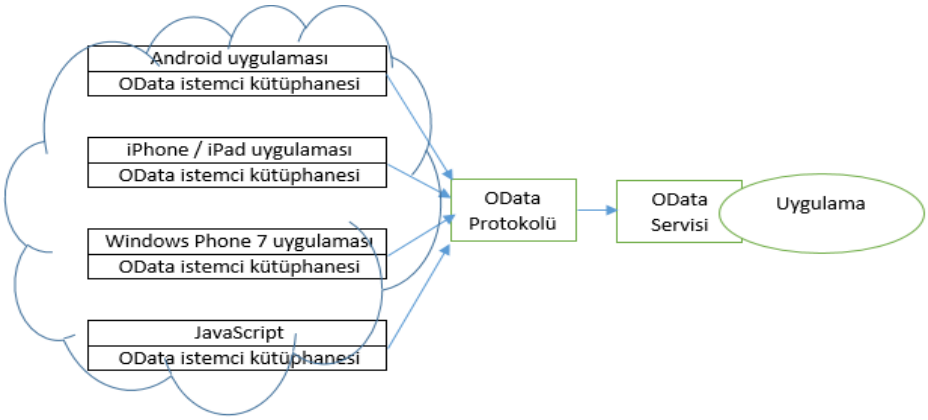
Bu bölümde iki mobil platformunda buluttaki verilere erişim bir protokol üzerinden nasıl yapıldığı üzerinde durulmuştur. Veri transferinde OData protokolü üzerinden açıklanmıştır. Bu protokolün kullanılmasının amacı iki platformdada kullanılabilen bu protokol olmasıdır. Amaç aynı protokolün platformlara göre farklılaşıp farklılaşmadığı ve platformların bu protokole uyumuna değinilmiştir.

OData kaynak erişimi, veri modeli içinde ve URL'le tanımlama işinde kullanılan REST API'lerinin yaratılması ve tüketilmesinde kullanılır. Yayımlama ve tanımlama işlemi HTTP mesajları sayesinde Web istemcileri tarafından kullanılmasını sağlar[123].

Günümüzde genellikle Web uygulamalarına tarayıcılardan erişilir. Daha çok oranda normal istemci uygulamaları özellikle mobil cihazlarda tarayıcılarda yer alır. İstemci uygulamalarında Web uygulamalarına erişim için REST iletişim standartı kullanılır.

Servislerindeki verilere erişim zordur. Eğer veri dönüştürülmeden yapılmak istenirse Web uygulama geliştiricinin veri modeli, kuyruk dili, istemci kütüphaneleri ve araçlara ihtiyaç duyar.

Eğer uygulama kendi verilerini bir protokol üzerinden paylaşmak istenirse (burada OData protokolü) işler daha kolay olacaktır. Aşağıdaki şekilde protokolün diğer ortamlarla olan uyumu gösterilmiştir.




Şekil 11. 4 Yeni dönem iletişim protokollerinin (Şekilde OData protokolü) bulut ile etkileşimi [123]

Yukardaki şekilde bir web uygulamasının farklı türde mobil cihazlar (Andorid, iPhone / iPad, Windows Phone 7) üzerinde bulunan istemci uygulamalarının erişiminin şekli gösterilmiştir. OData istemci kütüphaneleri Microsoft tarafından aynı veri modeli ile uygulanabilir hale getirilmiştir. OData protokolü üzerinden bir istek verisine ulaşıldığında her uygulama kendine uygun formatı seçebilir. Genellikle XML ile Atom veya JSON formatı kullanılır.

Bir istemci ham veriyi görmez. Aksine sadece OData servisi tarafından sağlanan veri modelini görür. OData veri modeli için haritalamanın nasıl yapılacağı geliştiriciye bağlıdır.

Veri kaynağı ilişkisel tablolar olursa geliştirici bir ya da daha fazla tabloyu uygulamanın OData veri modeline uygun olarak düzenleyebilir. Fakat bazı tabloların sütunları

dikkate alınmayabilir. Çünkü alternatif olarak veri tablosu ile OData Veri Model'inin farklı olduğu her yerde farklı haritalamanın uygulanması gerekir.

Geliştirici iki durumdan birini seçmekte serbesttir. OData kullanma uygulamanın verilerini istemcilerin direk göreceği anlamına gelmez.

## BÖLÜM 12

# WCF'DE MESAJ SEVİYE GÜNVELİKTE KULLANILAN ŞİFRELEME ALGORİTMALARI

Bu bölümde WCF'de mesaj seviye güvenlikte bağlamların sözdizimi[136] ve bağlamlarla kullanılabilen kullanılabilen şifreleme algoritmalarına değinilecektir. WsHttpBinding bağlamı ile HTTP protokolü, NetTcpBinding bağlamı ile TCP protokolü temsil edilmektedir.

```
<wsHttpBinding>
    <binding
        allowCookies="Boolean"
        bypassProxyOnLocal="Boolean"
        closeTimeout="TimeSpan"
        hostNameComparisonMode="StrongWildCard/Exact/WeakWildcard"
        maxBufferPoolSize="integer"
        maxReceivedMessageSize="Integer"
        messageEncoding="Text/Mtom"
                name="string"
        openTimeout="TimeSpan"
        proxyAddress="URI"
        receiveTimeout="TimeSpan"
        sendTimeout="TimeSpan"

textEncoding="UnicodeFffeTextEncoding/Utf16TextEncoding/Utf8TextEncodi
ng"
        transactionFlow="Boolean"
        useDefaultWebProxy="Boolean">
        <reliableSession ordered="Boolean"
           inactivityTimeout="TimeSpan"
           enabled="Boolean" />
        <security
mode="Message/None/Transport/TransportWithCredential">
           <transport
clientCredentialType="Basic/Certificate/Digest/None/Ntlm/Windows"
                proxyCredentialType="Basic/Digest/None/Ntlm/Windows"
                realm="string" />
          <message
```

```
algorithmSuite="Basic128/Basic192/Basic256/Basic128Rsa15/Basic256Rsa15
/TripleDes/TripleDesRsa15/Basic128Sha256/Basic192Sha256/TripleDesSha25
6/Basic128Sha256Rsa15/Basic192Sha256Rsa15/Basic256Sha256Rsa15/TripleDe
sSha256Rsa15"

clientCredentialType="Certificate/IssuedToken/None/UserName/Windows"
            establishSecurityContext="Boolean"
            negotiateServiceCredential="Boolean" />
        </security>
       <readerQuotas             maxArrayLength="Integer"
maxBytesPerRead="Integer"           maxDepth="Integer"
maxNameTableCharCount="Integer"
maxStringContentLength="Integer" />
    </binding>
</wsHttpBinding>
```

Şekil 12. 1 WsHttpBinding bağlamanın sözdizimi

```
<netTcpBinding>
   <binding
      closeTimeout="TimeSpan"
      hostNameComparisonMode="StrongWildCard/Exact/WeakWildcard"
      listenBacklog="Integer"
      maxBufferPoolSize="integer"
      maxBufferSize="Integer"
      maxConnections="Integer"
      maxReceivedMessageSize="Integer"
      name="string"
      openTimeout="TimeSpan"
      portSharingEnabled="Boolean"
      receiveTimeout="TimeSpan"
      sendTimeout="TimeSpan"
      transactionFlow="Boolean"

transactionProtocol="OleTransactions/WSAtomicTransactionOctober2004"

transferMode="Buffered/Streamed/StreamedRequest/StreamedResponse"

      <reliableSession ordered="Boolean"
            inactivityTimeout="TimeSpan"
            enabled="Boolean" />
      <security mode="None/Transport/Message/Both">
            <message
clientCredentialType="None/Windows/UserName/Certificate/CardSpace"
                defaultProtectionLevel="None/Sign/EncryptAndSign"
algorithmSuite="Basic128/Basic192/Basic256/Basic128Rsa15/Basic256Rsa15
/TripleDes/TripleDesRsa15/Basic128Sha256/Basic192Sha256/TripleDesSha25
6/Basic128Sha256Rsa15/Basic192Sha256Rsa15/Basic256Sha256Rsa15/TripleDe
sSha256Rsa15" />
            <transport clientCredentialType="None/Windows/Certificate"
                protectionLevel="None/Sign/EncryptAndSign" />
      </security>
```

```
      <readerQuotas             maxArrayLength="Integer"
maxBytesPerRead="Integer"           maxDepth="Integer"
maxNameTableCharCount="Integer"
maxStringContentLength="Integer" />   </binding>
</netTcpBinding>
```

Şekil 12. 2 NetTcp bağlama sözdizimi

Bağlamalarla mesaj seviye güvenlikte kullanılabilen şifreleme türleri şöyledir[137].

1. Basic128: Şifreleme için Basic128 (AES şifreleme algoritması), mesaj özeti için SHA1, anahtar değişimi için Rsa-oaep-mgf1p kullanılır.
2. Basic 192: Şifreleme için Basic128 (AES şifreleme algoritması), mesaj özeti için SHA1, anahtar değişimi için Rsa-oaep-mgf1p kullanılır.
3. Basic 256: Şifreleme için Basic256 (AES şifreleme algoritması), mesaj özeti için SHA1, anahtar değişimi için Rsa-oaep-mgf1p kullanılır.
4. Basic256Rsa15: Şifreleme için Basic256 (AES şifreleme algoritması), mesaj özeti için SHA1, anahtar değişimi için Rsa15 kullanılır.
5. Basic192Rsa15: Şifreleme için Basic192 (AES şifreleme algoritması), mesaj özeti için SHA1, anahtar değişimi için Rsa15 kullanılır.
6. TripleDes: Şifreleme için TripleDes, mesaj özeti için SHA1, anahtar değişimi için Rsa-oaep-mgf1p kullanılır.
7. Basic128Rsa15: Şifreleme için Basic128 (AES şifreleme algoritması), mesaj özeti için SHA1, anahtar değişimi için Rsa15 kullanılır.
8. TripleDesRsa15: Şifreleme için TripleDes, mesaj özeti için Sha256, anahtar değişimi için Rsa15 kullanılır.
9. Basic128Sha256: Şifreleme için Basic128 (AES şifreleme algoritması), mesaj özeti için Sha256, anahtar değişimi için Rsa-oaep-mgf1p kullanılır.
10. Basic192Sha256: Şifreleme için Basic192 (AES şifreleme algoritması), mesaj özeti için Sha256, anahtar değişimi için Rsa-oaep-mgf1p kullanılır.
11. Basic256Sha256: Şifreleme için Basic192 (AES şifreleme algoritması), mesaj özeti için Sha256, anahtar değişimi için Rsa-oaep-mgf1p kullanılır.

12. TripleDesSha256: Şifreleme için TripleDes, mesaj özeti için Sha256, anahtar değişimi için Rsa-oaep-mgf1p kullanılır.
13. Basic128Sha256Rsa15: Şifreleme için Basic128 (AES şifreleme algoritması), mesaj özeti için Sha256, anahtar değişimi için Rsa15 kullanılır.
14. Basic192Sha256Rsa15: Şifreleme için Basic192 (AES şifreleme algoritması), mesaj özeti için Sha256, anahtar değişimi için Rsa15 kullanılır.
15. Basic256Sha256Rsa15: Şifreleme için Basic256 (AES şifreleme algoritması), mesaj özeti için Sha256, anahtar değişimi için Rsa15 kullanılır.
16. TripleDesSha256Rsa15: Şifreleme için TripleDes, mesaj özeti için Sha256, anahtar değişimi için Rsa15 kullanılır.

Şifreleme takımlarında geçen alt şifreleme algoritmaları şöyledir.

DES: İlk tasarılığında donanım uygulamalarında kullanılması amaçlanmıştır. İletişim amaçlı kullanımda hem gönderen, hem de alıcı şifreleme ve deşifrelemede kullanılan aynı gizli anahtar üzerinde anlaşmış olmalıdır. Gizli anahtarın güvenli bir biçimde dağıtımı için açık anahtarlı sistem kullanılabilir[136]. Bulut ortamının tam olarak kurulmasından sonra kullanımı azalacak belkide kullanılmayacaktır.

AES: Des'e göre daha güvenli bir sistemdir. 128 bit, 192 bit ve 256 bit olmak üzere üç farklı anahtar uzunluğuna sahip olabilir. AES'in DES'in aksine donanımda ve yazılımda hızlı olması, daha kolay uygulanabilir olması ve çok daha az hafızaya gerek duyması güçlü yönleri olarak söylenebilir. Günümüzde bilinen tüm akademik, pratik ve doğrudan (brute force) saldırılara karşı dayanıklı olduğu düşünülmektedir. En yaygın olarak kullanılan simetrik şifreleme algoritmasıdır[137] .

RSA: Göndericinin bir metotla ve herkesçe bilinen açık bir anahtarla mesajlarını şifrelediği bir şifre sistemi olarak tanımlanır. Simetrik anahtarlı sistemlerin aksine anahtarı bilmek deşifre anahtarını ortaya çıkarmaz. Bu sistem hem gizlilik hem de dijital imza sağlamak amaçlı kullanılabilir[138]. Türlerinden olan Rsa15 ve Rsa-oaep-mgf1p mesaj seviye güvenlikte kullanılabilir. Türlerinden olan Rsa-oaep-mgf1p, Xml düzeninde verilerinin RSA şifreleme algoritmasıyla şifreli olarak iletilmesi için kullanılır. SOAP iletişim formatında XML'e benzediğinden mesaj seviye güvenlikte kullanılabilmetedir.

SHA: Genellikle MD5 ile birlikte kullanılan karışık bir şifreleme algoritmasıdır. EN büyük özelliği şifreleme anahtarından tesrisine şifreleme anahtarı elde edilememesidir. Bir türü olup özetleme fonksiyonlarından olan SHA-1 herhangi bir uzunluktaki bir metinin sabit uzunluktaki özetini oluşturur. Bu özet, veri bütünlüğü ve kimlik doğrulaması ile ilgili uygulamalarda temel yapı taşı haline gelmiştir.

# BÖLÜM 13

# WCF'DE MESAJ SEVİYE GÜNVELİKTE KULLANILAN ŞİFRELEME ALGORİTMALARININ PROTOKOLLE İLİŞKİSİ ÜZERİNE BİR ÇALIŞMA

Bu bölümde web servis güvenlik tipleri, protokol bazında test yapılarak test sonuçlarına değinilmiştir.

## 13.1 Web Servis Güvenlik Tipleri

Web servis güvenliği, web servislerinin güvenliğini sağlamaya yönelik bir SOAP uzantısıdır [135]. SOAP şemasına uygun olarak hazırlanmış mesajlar XML formatına dönüştürülerek iletilir. Web servis güvenliği dediğimizde aklımıza ilk gelen koruma kalitesi ile mesaj güvenirliliği, gizliliği ve tekillik doğrulamasıdır [125].

Web servis güvenliği üç kısımda incelenir: taşıma katmanı güvenliği (TLS), mesaj katmanı güvenliği ve uygulama katmanı güvenliği.

Taşıma katmanı güvenliği (TLS), internet üzerinden uygulama ve kullanıcılar arasında iletişim gizliliği sağlayan bir protokoldür [105]. Ortamda sadece istemci ve sunucu bulunur; üçüncü bir bileşen bulunsa da iletişimle ilgisi yoktur. Yani mesaj iletişimi sadece istemci ve sunucu arasında etkileşim ile sağlanır. Bu türe verilebilecek en güzel örnek; bir web sitesine bağlantı gerçekleştirme istenildiği, zaman karşı tarafta kullanıcı ismi ve şifresi sorulması ile gerçekleştirilen kimlik doğrulamasıdır. Bu durumda istemci (kullanıcı) ile sunucu (web sitesi) haricinde bir istemci veya sunucu bulunmaz.

Mesaj katmanı güvenliğinde, güvenlik bilgileri SOAP iletisinin veya güvenlik bilgi mesajı eki ile birlikte iletim sağlayan SOAP mesaj ekinin içerisinde bulunur. Burada, sadece istemci ve sunucu arasında mesaj iletişimi olmaz; birden çok istemci ve sunucu olabilir. Bazen de ortamın sınırları dahi bilinmez. Örneğin, e-ticaret sitesi üzerinden alışveriş

yapıldığında istemci kullanıcı olarak bağlanırken site sunucu, e-ticaret sitesi alışveriş tutarını ilgili bankadan tahsil edince istemci olur. Bankaların sayısı birden fazla olabileceğinden ve kullanıcılar da aynı anda site üzerinde alışveriş yapabileceğinden ortam çok sunuculu ve çok istemcili bir ortama dönüşür.

Uygulama seviyesi güvenliği ise bu iki güvenlik tipinin belirli yerlerde kullanılması üzerine kuruludur. Taşıma seviyesinde önemli olan kimin bağlantı yapmak istediğiyken, mesaj seviyesi güvenliğinde hangi kullanıcının hangi verilere ne kadar süreyle ulaşmak istediği önemlidir. Bazı üst sistemlerde ise istemci ve sunucu arasındaki ilişki bire bir, birden çoğa ve çoktan çoğa olacağından, hangi güvenlik tipinin nerede kullanılacağı sorusu üzerine çözüm bulmaya çalışan güvenlik tipidir. Ayarlamanın tam olarak sağlanması zor olduğundan sistemin bazen gereğinden az, bazen de gereğinden fazla güvenlik unsurları taşımasına neden olabilir.

### 13.1.1 Mesaj Seviye Güvenlik

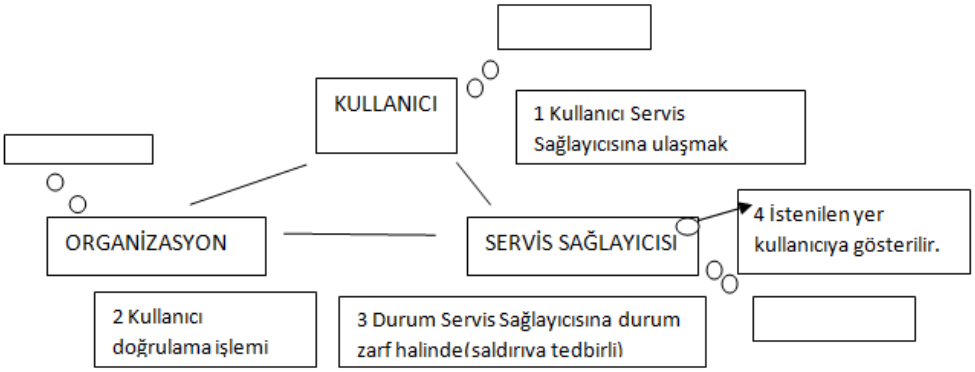
Mesaj düzey güvenlik, belge sistemleri ve iş ortakları arasında dijital olarak imzalanmış veya şifrelenmiş mesaj iletişimi sağlar [126]. TLS'den farklı olarak bu tanımda imzalama olayı ve iş ortağı değil iş ortakları kelimeleri bulunmaktadır. Mesaj katmanı güvenliğinde kimlik doğrulama amacıyla adresleme işleyicisi(addressing handler) kullanmaktır. Burada amaç, ortamda bulunan çoklu bileşenlerin(çoklu istemci ve çoklu sunucu) haberleşmesinde uygun protokol kullanıldığını, kullanılan adres kelimelerini (string) kontrol edip (genişletilen adres kelimeleri taranarak), doğrulama yaparak iletişime izin vermektir. Adres kelimelerinin amacı uygun adresleri genişletmek, yani yenilerini türetmektir [127]. İletişim sırasında adres kelimeleri genişletilmez, genişlemeye izin verilseydi, karmaşık olan mesaj seviye güvenliği daha da karmaşıklaşır ve sayısı bilinmeyen sunucuların birinin iletişimi bitmeden diğerleriyle iletişim kurulmaya çalışıldığında belli bir seviyeden sonra sunucular çalışamaz duruma gelebilirdi. Adres kelimesi üretme ara yüzü, üretilen adres kelimesinin doğru formatta olup olmadığının kontrolünü gerçekleştiren fonksiyonlardan oluşur.

Mesaj düzeyinde diğerinde kullanılan kimlik doğrulama yöntemlerinden birisi de vekil (Proxy) kimlik doğrulama yöntemidir. Vekil kimlik doğrulama, ilke dosyasında düzenlemesi belirtilmişse gereklidir [110]. Bu yöntemin en önemli avantajı kendini

sistemdeki diğer sunuculardan gizlemesidir. Bu durumda, isteğin aslında kimden geldiği konusunda karışıklığa düşünüleceği izlenimini doğursa da, araya istemci veya sunucu olarak girmeye çalışanın gerçek hedefe değil, oluşturulan sanal konuma ulaşmasını sağlar. Başka bir avantajı da önbellek (cache) mekanizması sayesinde, sunucuların aynı koşullardaki aynı istekler için hazırlanmış cevapları göndererek rahatsız edilmemesini ve üst sistemin daha hızlı çalışmasını sağlamasıdır. Ancak, araya girme çabası başarılı olursa, gerçek konumu yerine sanal bir konuma ulaşılacağı için saldırganın belirlenmesi zorlaşır. Günümüzde şifreleme sistemlerinin güçlülüğü ve sistemlerin aşırı çalışmasını önlemesi sayesinde tercih edilme oranı oldukça fazladır. Kimlik doğrulama erişiminin işletilebilmesi için farklı kategorideki her URL isteği için proxyde artan numaranın kullanılması gerekir [128]. Böylece aynı anda cevap veremeyen sunucu artan numaraları bir sıraya koyar ve sonrasında işler ve/veya önbelleğinde varsa direk cevap verir.

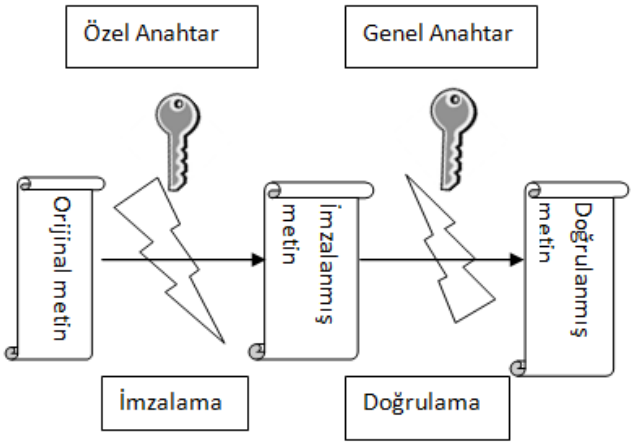
Son olarak SAML kimlik doğrulama yöntemi kullanılabilmektedir. SAML(Güvenlik İddiası İşaretleme Dili) kimlik doğrulama işleyicisi, HTTP post bağlamayı kullanan protokoldür. Yapılabilen işleri; imzalama ve mesajları şifreleme, CRX kullanıcılar ve grupların otomatik oluşturulması, hizmet sağlayıcı ve kimlik sağlayıcı başlatılan kimlik doğrulamasıdır [129].

SAML ile kimlik doğrulamada servletlerden faydalanılır. Servlet, HTTP istemlerine cevap veren servlet ara yüzüyle uyumlu çalışan sınıftır. Yaşam döngüsü incelendiğinde yapılan iş daha iyi anlaşılacaktır.




Şekil 12. 1 SAML kimlik doğrulama işleyicisi çalışma mekanizması (Ortamda birden fazla kullanıcı / organizasyon / servis sağlayıcısı olabilir.)

Veri güvenliğinde önemli konularından biri olan veri imzalama konusu burada imzalama işleyicisi kullanılarak uygulanır. Normal veri imzalamadan farklı verilerinin hepsinin değil istenilen kısmının imzalanması gerçekleştirilebilir. Bu da istenilen sunuculara veya istemcilere verinin ilgili kısmının imzalanarak sunulmasını sağlar. Bu sayede, hem hızda artış olur hem de, gereksiz veri gereksiz ortama hiç ulaşmamış olur.




Şekil 12. 2 Basit sayısal imzalama

İmzalama işleyicisinin detayı şu şekilde verilebilir: istemci tarafında istekleri imzalamak için kullanılırken, sunucu tarafında da yanıtlar imzalanır. Bir XML belgesindeki düğüm imzalanabileceği gibi, yalnızca belirli öğelerin imzalanması için de kullanılabilir. Bir web servisi sürecinde, hiçbir ad tanımlayıcı veya bir kimlik bileşeni ayarlanmaz veya bir kök eleman bulunmazsa; zarfın içindeki tüm unsurlar için imzalı bir web servis kimliği atanır. Herhangi bir elemanın(kimlik tarafından düzenlenen) üzerinde oynama yapılırsa sunucu tarafında işler karışacağından önceki imzalar geçersiz olur.

## 13.2 Test İşlemleri

Bu bölümde WCF'de Mesaj Seviye Güvenlikte Şifreleme Algoritmalarının çeşitli parametrelere etkileri konusunda Windows Phone ve Androide istemcisine bağlı olarak test işlemi uygulanarak elde edilen sonuçlar gösterilecektir. Test işlemi ana yapı olarak iki kısma ayrılmıştır. İlk kısımda istemci sunucudan değişken uzunlukta kelime katarı üretmesini ve sonuç olarak bu kelime katarını istemciye döndürmesini istemiş, ikinci kısımda ise istemci sunucudan aynı uzunluktaki kelime katarlarını istemiş sunucuda bu kelime katarlarını istemciye göndermiştir. Her iki testte de mesaj seviye güvenlik şifreleme türleri farklı bağlayıcılarla (HTTP, TCP) uygulanmıştır.

Test edilen bilgisayarın özellikleri: 2GHz DDR2 işlemci, 160 GB HDD bellek, 4 MB L2 ön bellek.

Test sonuçlarının gösterildiği çizelgelerde geçen kavramlar şöyledir. Adet parametresi ile istemcinin sunucundan üretmesini istediği kelim katarının uzunluğunu, T ile istemci ve sunucu arasındaki sayısını, B ile istemci ve sunucu arasındaki mesajlaşmanın boyutunu KB olarak, S ile ile istemci ve sunucu arasındaki mesajlaşmanın süresini saniye cinsinden gösterirken TVerim, SVerim ve BVerim ile ilgili parametrenin Adet değişkenine bölümü ile elde edilmiştir.

Uygulamanın TCP ve HTTP protokolleriyle test edilmesinin sebebi istemci ve sunucunun bulut ile iletişiminde esas olarak HTTP protokolünün kullanılmasının yanında büyük veride TCP protokülünün kullanılmasıdır [149]. Büyük veride TCP protokolünün kullanılmasının nedeni şu an büyük veride güvenlikten çok hızlı veri işleme konusunda yoğunlaşılmasıdır. Test işlemlerinde vurgulanmak istenen başka bir noktada büyük boyuttaki mesajlaşmalarda sunuculara yüklenen görevi çeşitli parametreler ile incelemektir.

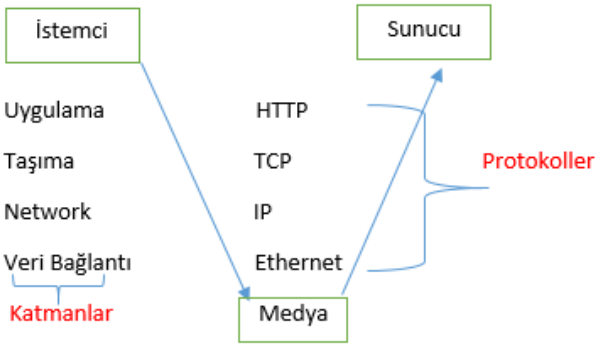
WCF ortamına Windows Phone ve Android istemcilerinin bağlanması sunucu açısından farketmediğinden test sonuçları her ikisi içinde ortak gösterilmiştir. Fakat Andorid istemciler için özel ayarlamaların gerektiği durumlarda ayarlamalar yapılmıştır.

### 13.3.1 Test Edilen Protokoller

Çalışmada kullanılan HTTP protokolü, dağıtılmış, işbirlikçi, çoklu ortam bilgi sistemleri için bir uygulama protokolü [141] olup bulut ortamının veri iletişiminde artık bir

standard olarak kabul edilmiştir. Bu protokolü desteklemeyen ortam yok gibidir. WCF'de bu protocol temelde basicHttpBinding (Temel HTTP Bağlama) ve wsHttpBinding ile temsil edilir. Bu çalışmada basicHttpBinding'in katılmamasının nedeni bu bağlamanın güvenlik mekanizmalarına açık olmaması ve günümüz ortamında güvenliksiz very iletişiminin kabul edilmemesi yönünden artık kullanımının eski işletim sistemlerinde azalması ve yeni işletim sistemlerinde desteklenmemesidir. TCP, TCP/IP protokol takımının iki aktarım katmanı protokolünden birisidir. Gelişmiş bilgisayar ağlarında paket anahtarlamalı bilgisayar iletişiminde kayıpsız veri gönderimi sağlayabilmek için TCP protokolü oluşturulmuştur [131].

Bu iki protokol arasındaki en önemli fark HTTP'nin daha çok mesaj seviye güvenliği ile ilgilenirken TCP'nin daha çok taşıma seviyesi güvenlikle ilgilenir. Esasında iki protokol tipi de bu iki güvenlik tipinin karışımını kullanır [134]. Fakat oranları farklılık gösterir. Örneğin TCP istemci ve sunucu arasında iletişime geçmeden önce kullanılacak veri iletişim yolunun test eder. HTTP ise daha standart bir protokol olduğundan TCP kadar özellikleri (örn. TCP kabul paketi, bağlantılı protokol olması) kendisinde varsayılan olarak barındıramaz.




Şekil 12. 3 Protokollerin katmanla ilişkisi [133]

Test ortamında elde edilen sonuçlar eklerde gösterilmiştir. Bölümde sadece elde edilen sonuçların yorumlarına yer verilmiştir.

### 13.2.1 Test Ortamında Elde Edilen Verilerin Genel Veri Özeti

Bu bölümde verilen değerlerin hangi şifreleme türüne ait oldukları EK-B'de gösterilen mesaj seviye güvenliğin şifreleme algoritmalarına başlıklarındaa verilen isimdir. Verilmemiş olan değerler güvenliksiz ortamı göstermiştir. Örneğin B1 ile Basic128 şifreleme algoritması ile oluşturulmuş mesajlaşma boyutunu KB. cinsinden değeri

gösterilmiştir. TVerim2 ile Basic128Rsa15 şifreleme algoritması ile oluşturulmuş TVerim değeri ilgili bağlamadaki değerleri gösterilmiştir. Genel veri özeti olarak serinin minimum, maksimum, ortalama, medyan ile 1. ve 3. çeyrek değerleri kullanılmıştır.

#### 13.2.1.1 Tek İstemcili NetTcpBinding Bağlamalı Ortamda Elde Edilen Genel Veri Özeti Değerleri

Tek istemcili NetTcpBinding bağlamalı ortamda elde edilen genel veri özeti değerleri incelendiğinde güvenliksiz ortamda değişen mesajlaşma boyutları karşısında iletişim süresi en hızlı olurken değişen şifrele algoritmalarına göre iletişim süresi belirtilen duyarlılıkla (1. sn.) değişmemiştir. İletişim mesaj adedi parametresinde güvenliksiz ortamda 2 adet olurken, test edilen mesaj algoritmasına göre değişmeyip 10 adet değerini almıştır. İletişim mesaj boyutu da güvenliksiz ortamda en az değeri alırken bu sefer şifreleme türüne göre iletişim mesaj boyutunun aldığı değer farklı görülmüştür.

#### 13.2.1.2 Tek İstemcili WsHttpBinding Bağlamalı Ortamda Elde Edilen Genel Veri Özeti Değerleri

Tek istemcili WsHtppBinding bağlamalı ortamda elde edilen genel veri özeti değerleri incelendiğinde güvenliksiz ortamda değişen mesajlaşma boyutları karşısında iletişim süresi en hızlı olurken değişen şifrele algoritmalarına göre iletişim süresi belirtilen duyarlılıkla (1. sn.) değişmemiştir. İletişim mesaj adedi paretmetreside güvenliksiz ortamda 2 adet olurken, test edilen mesaj algoritmasına göre değişmeyip 10 adet değerini almıştır. İletişim mesaj boyutu da güvenliksiz ortamda en az değeri alırken bu sefer şifreleme türüne göre iletişim mesaj boyutunun aldığı değer farklı görülmüştür.

#### 13.2.1.3 Çok İstemcili NetTcpBinding Bağlamalı Ortamda Elde Edilen Genel Veri Özeti Değerleri

Çok istemcili NetTcpBinding bağlamalı ortamda elde edilen genel veri özeti değerleri incelendiğinde güvenliksiz ortamda değişen mesajlaşma boyutları karşısında iletişim süresi en hızlı olurken değişen şifreleme algoritmalarına göre iletişim süresi değişkendir. İletişim mesaj adedi parametresi de güvenliksiz ortamda 2 adet olurken, test edilen mesaj algoritmasına göre değişmeyip 10 adet değerini almıştır. İletişim mesaj boyutuda da güvenliksiz ortamda en az değeri alırken bu sefer şifreleme türüne göre iletişim mesaj boyutunun aldığı değer farklı görülmüştür. İletişim mesaj boyutu ve

iletişim mesaj süreleri değerlerinin hangi şifreleme algoritmalarında birbirlerine yakın değerler elde edildiğine benzerlik çizelgelerinde değinilecektir.

#### 13.2.1.4 Çok İstemcili WsHttpBinding Bağlamalı Ortamda Elde Edilen Genel Veri Özeti Değerleri

Çok istemcili wsHttpBinding bağlamalı ortamda elde edilen genel veri özeti değerleri incelendiğinde güvenliksiz ortamda değişen mesajlaşma boyutları karşısında iletişim süresi en hızlı olurken değişen şifrele algoritmalarına göre iletişim süresi değişkendir. İletişim mesaj adedi parametresinde güvenliksiz ortamda 2 adet olurken, test edilen mesaj algoritmasına göre değişmeyip 10 adet değerini almıştır. İletişim mesaj boyutuda da güvenliksiz ortamda en az değeri alırken bu sefer şifreleme türüne göre iletişim mesaj boyutunun aldığı değer farklı görülmüştür. İletişim mesaj boyutu ve iletişim mesaj süreleri değerlerinin hangi şifreleme algoritmalarında birbirlerine yakın değerler elde edildiğine benzerlik çizelgelerinde değinilecektir.

### 13.2.2 Test Ortamında Elde Edilen Verilerin İlişki Tablosu

Bu bölümde verilen değerlerin hangi şifreleme türüne ait oldukları ilgili mesaj seviye güvenliğin şifreleme algoritmalarısı isimdir. Verilmemiş olan değerler güvenliksiz ortamı göstermiştir. Örneğin B1 ile Basic128 şifreleme algoritması ile oluşturulmuş mesajlaşma boyutunu KB. cinsinden değeri gösterilmiştir. TVerim2 ile Basic128Rsa15 şifreleme algoritması ile oluşturulmuş TVerim değeri ilgili bağlamadaki değerleri gösterilmiştir. İlişki tablosu olarak bir parametrenin aynı türde farklı şifreleme algoritması ile test edilmiş ve aralarındaki benzerlik değerleri gösterilmiştir. Değerin 1'e yaklaşması benzerliğin daha çok olduğunu, O'a yaklaşması ise benzerliğin azaldığını gösterir.

#### 13.2.2.1 Tek İstemcili NetTcpBinding Bağlamalı Ortamda Elde Edilen İlişki Tabloları Test Verileri

Tek istemcili NetTcpBinding bağlamalı ortamda elde edilen ilişki tablosu değerleri incelendiğinde iletişim mesaj boyutu olarak şifreleme algoritmaları temel olarak 4 guruba ayrılmıştır.

Çizelge 13. 1 Tek istemcili NetTcpBinding bağlamalı ortamda elde edilen verilere göre şifreleme algoritmalarını sınıflandırılaması çizelgesi

| |
|---|
| Basic128, Basic128Rsa15, Basic192 |
| Basic128Sha256, Basic128Sha256Rsa15, Basic192Sha256, Basic192Sha256Rsa15, Basic256, Basic256Rsa15, TripleDes, TripleDesRsa15 |
| Basic256Sha256, Basic256Sha256Rsa15 |
| TripleDesSha256, TripleDesSha256Rsa15 |

#### 13.2.2.2 Tek İstemcili WsHttpBinding Bağlamalı Ortamda Elde Edilen İlişki Tablosu Değerleri

Tek istemcili WsHttpBinding bağlamalı ortamda elde edilen ilişki tablosu değerleri incelendiğinde iletişim mesaj boyutu olarak 3 guruba ayrılmıştır.

Çizelge 13. 2 Tek istemcili WsHttpBinding bağlamalı ortamda elde edilen verilere göre şifreleme algoritmalarını sınıflandırılaması çizelgesi

| |
|---|
| Basic128, Basic128Rsa15, Basic192, Basic192Rsa15, Basic192Sha256, Basic192Sha256Rsa15, Basic256, Basic256Rsa15, TripleDes, TripleDesRsa15 |
| Basic128Sha256, Basic128Sha256Rsa15 |
| Basic256Sha256, Basic256Sha256Rsa15, TripleDesSha256, TripleDesSha256Rsa15 |

#### 13.2.2.3 Çok İstemcili NetTcpBinding Bağlamalı Ortamda Elde Edilen İlişki Tablosu Değerleri

Çok istemcili NetTcpBinding bağlamalı ortamda elde edilen ilişki tablosu değerleri incelendiğinde iletişim mesaj boyutu olarak 3 guruba ayrılmıştır.

Çizelge 13. 3 Çok istemcili NetTcpBinding bağlamalı ortamda elde edilen verilere göre şifreleme algoritmalarını sınıflandırılaması çizelgesi

| |
|---|
| Basic128Sha256 |
| TripleDesRsa15, TripleDesSha256 |
| Diğer şifreleme algoritmaları |

İletişim mesaj süreleri arasındaki fark sınır değerinden (0.001) fazladır.

#### 13.2.2.4 Çok İstemcili WsHttpBinding Bağlamalı Ortamda Elde Edilen İlişki Tablosu Değerleri

Çok istemcili WsHttpBinding bağlamalı ortamda elde edilen ilişki tablosu değerleri incelendiğinde iletişim mesaj boyutu olarak 5 guruba ayrılmıştır.

Çizelge 13. 4 Çok istemcili WsHttpBinding bağlamalı ortamda elde edilen verilere göre şifreleme algoritmalarını sınıflandırılaması çizelgesi

| |
|---|
| Basic128Rsa15 |
| Basic128Sha256 |
| TripleDesRsa15 |
| TripleDesSha256Rsa15 |
| Diğer şifreleme algoritmaları |

İletişim mesaj süreleri arasındaki fark sınır değerinden (0.001) fazladır.

# BÖLÜM 14

# SONUÇLAR

Test edilen iki mobil ortamın (Anroid, WindowsPhone) bulut ile iletişimde herhangi bir sorun bulunmamaktadır. Ama genel olarak Android buluta bağlanmada varsayılan değerleri daha geniş aralıklıdır. Windows Phone ise varsayılan değerleri daha düşük tutarak platform olarak daha temkinli davranmıştır. Örneğin bağlantı düşme süresi Android'de daha fazladır. Fakat her iki ortamda bağlantı düşme süreleri istenilen değer solarak ayarlanabilir. İki ortamda iletişim için HTTP protokolünü kullanır. Bu sayede platformlar arasındaki iletişimde sorun olmaz. HTTP protokolü günümüzde mobil ortamda iletişim olarak kullanılmasının yanında bulut için en çok kullanılan iletişim protokolüdür.

Mobil ve bulut ortamında HTTP'nin TCP'ye tercih edilmesi sebebiyle günümüzde uygulamaların donanımsal olmaktan uygulama üzerinde değiştiğini gözlemlemek mümkündür.

Sonraki test aşamasında HTTP yerine TCP protokolünün bulut ortamında kullanılmasının sonuçları incelenmiştir. TCP protokolü mobil ortamda kullanım oranının günümüzde hala neden artmadığı üzerinde durulmaya çalışılmıştır. TCP'nin bulut ortamına uyumunun zor olduğu bilinildiği halde bu çalışmanın yapılmasının nedeni TCP'nin daha hızlı bir protokol olmasına karşın geliştirilebilirliğinin daha düşük olması etkenlerinin karşılaştırılmasını yapmaktır. Yani testte ölçülecek hız farklılıklarının geliştirilebilirdik kalite unsuru yanında tercih edilmemesinin oranını bulmaktır. Yani yazılım kalite unsurlarının önemliliklerinin ortamdan ortama fark ettiği ve her unsurun farklı ortamlarda farklı öneme sahip olduğu sonucuna ulaşılır.

Test işlemlerinde TCP protokolü kullanıldığında iletişim mesaj boyutu büyümesine karşın, iletişim mesaj süreleri HTTP protokolüne göre daha azdır. HTTP protokolünün TCP protokolü kullandığı bilinmesine karşın WCF'de TCP kullanılarak veri iletişiminde mesaj boyutu artmıştır.

Günümüzde HTTP protokolünün istemci ve sunucu arasındaki iletişimlerde daha fazla kullanılmasının nedeni bu protokolün daha geniş çevrelerce tanınması ve kullanımının daha eskiye dayanmasından dolayı kullanıcılarda uyandırdığı güven birikimidir. WCF mesaj seviye güvenlikte HTTP protokolünün varsayılan şifreleme türü Basic128 iken, TCP protokolünün ki güvenliksizdir. Her iki protokolde WCF mesaj seviye güvenlikteki protokolleri desteklemektedir. TCP'in iletişim boyutunda kötü çıkmasının nedeni kendisinin varsayılan olarak taşıma seviye güvenliği çağırması, mesaj seviye güvenliği ise kendisine eklenmek edildiğinde kullanmasıdır. Bu sebeple eklenmek istenen özellik iletişim mesajına yeni etiketler eklenmesine neden olacağından iletişim mesajının boyutunu arttırmıştır. Süre olarak daha iyi olması ise iletişimin belirli bir istemciden sunucuya olduğundan ve bu konu esas olarak transfer seviye güvenliği ilgilendirdiğinden bu alanda daha iyi çalışmaktadır. WCF mesaj seviyesi güvenlikte doğrusal karmaşıklığa sahiptir. Yani programın çalışma hızı ve boyutu iletişim mesaj sayısının artması karşısında sabit olarak artar. Bu durum çalışma hızı verimi ve iletişim mesaj boyutu verimi parametreleri ile gözlemlenmiştir. Fakat iletişim mesaj sayısı aynı kalıp tek seferde iletişim mesaj boyutu arttırıldığından bu sefer iletişim mesaj boyutu çok fazla değişmemiştir. Çünkü SOAP'da iletişim mesaj boyutlarındaki düzenleme etiketleri (her mesajlaşmada kullanılan mesaj hariç etiketler) çok fazla olduğundan iletişim mesaj boyutu bu değişkeni çok fazla değiştirememiştir. İletişim ortalama 8.000 karakterden fazla olduğunda WCF iletişimin dosya ile olmasını istemiştir.

WCF üzerinde SOAP kurallarına uygun olarak iletişim yapılmak istenildiğinde varsayılan değerlerin kurumsal uygulamalar için yetersiz olduğu gözlemlenmiştir. Örnek olarak varsayılan iletişim bağlantı süresi bir dakika olup. Bu değer kurumsal uygulamalar için yetersiz kalmaktadır. İletişim süresi en fazla yaklaşık bir saat olabilmektedir.

Bulut ortamında hâlâ TCP protokolü kullanılmaya tam olarak uygun değildir. Bulutta uygulama katmanında çalışmak istenildiğinden HTTP protokolünün kullanılması daha uygundur. TCP'nin daha hızlı bir protokol olmasına karşın iletişimde taşınan mesajın

parçalarının kaybolabileceği düşüncesi bu protokolün bulut ortamındaki kullanım oranını düşürmüş. HTTP protokolünün ise bu durumu garanti altına alması ve çoğu sistemde tanımlanması ise bu protokolün bulut ortamında kullanımını arttırmıştır. HTTP'nin önemli bir avantajı da uygulama katmanı protokolü olduğu için programcının alt seviye konularla (taşıma gibi) ilgilenmesinin soyutlamasıdır.

SOAP'da mesaj seviye güvenlikte asimetrik şifreleme algoritmalarının çalışma sürelerinin mesaj iletişim sayısına göre simetrik şifreleme algoritmaları ile kıyaslandığında daha düzensiz olduğu görülmüştür. Bu sebeple SOAP'da mesaj seviye güvenlikte simetrik şifreleme algoritması kullanmak daha iyi olacaktır. Çünkü SOAP'da mesaj iletişimde iletilecek mesajla birlikte birçok standart metin iletildiğinden bu da iletişim mesajını şişirdiğinden iletişim süresinin değişmesi istemcide sistemin iyi çalışmadığı fikrini doğuracaktır.

Mobil ortamlarda ise istemcinin ortamdan her an kopabileceği fikri günümüzde hâlâ sürdüğünden iletişimde SOAP kullanılması yerine REST kullanılmasına gidilmiştir. REST'in avantajı iletişim mesajlarını daha kısa bir biçimde (fazla başlık eklemeden) ortama verip iletişimi sağlamasıdır. Fakat o zamanda uzun mesaj iletişimde mesaj çok kez parçaya bölüneceğinden getireceği fayda azalacak belli bir uzunluktan sonra SOAP kullanımı daha kârlı sonuç vermeye başlayacaktır.

Günümüzde hâlâ bulut ortamı tam olarak oluşmadığı sonucuna varılabilir. Çünkü tam anlamıyla bulut ortamı yakalandığı zaman günümüzde sunucunun cevap verdiği istemci türüne göre cevabı değişmediği halde cevabın iletim formatının farklı olması ve istemcilerin kategorize edilmesi bulut ortamının esasında tam olarak oluşturulamadığı sonucu ortaya çıkar. Bu duruma örnek olarak bazı sitelerin mobil istemciler için farklı içerik göndermesi gösterilebilir.

Mobil uygulamalarda kullanılabilen fakat kurumsal PC'lerde bulunmayan GPS desteği gibi destekler mobil uygulamalara artı bir avantaj sağlayıp sadece mobil istemcilere cevap veren uygulamaların olmasına sebep olan bir durum örneğidir. Bu sebeple sunucuların her çeşit istemciye cevap vermemesine neden olur.

Günümsüde veri iletiminde mobil taraflı asenkron iletişimin, kurumsal olarak eşzaman iletişimin kullanılması bulut ortamının tam olarak kurulamadığının bir başka kanıtıdır.

Günümüzde çerezlerin kullanım amacı kötü olmasada güvenlik açığı oluşturabilmesi ileri yıllarda kullanımının azalacağı belkide bulut ortamının tam olarak oluşturulabilmesi sonucunda kullanımının kaldırılabileceği düşüncesi savunulabilir. Windows Phone tarafında durum Görünüm Yönetimi istemci taraflı durum yönetimi durumu ile çözülmek istenilmiştir.

Çıkan sonuçlar toplandığında varılacak en nihai sonuç şudur. Eğer gelişilmek istenilen uygulama kurumsal olmayan bir uygulama ise HTTP protokolü kullanımının uygun olduğu fakat uygulama kurumsal ise bulut ile etkileşiminde HTTP protokolü kullanımının avantajlı olduğu fakat kurum içinde istemci ve sunucu arasındaki iletişimde TCP protokolü avantajlı olabileceği sonucunda ulaşılır. Çünkü WCF, TCP protokolünün kullanılmasıyla oluşturulan uygulamalarda kurum içi iletişimde mesaj kaybı yaşatmayacaktır. Bu durum test işlemleri üzerinde hiçbir veri kaybı olmaması ile gözlemlenmiştir.

## KAYNAKLAR


[1] Windows Phone, What Is Cloud? ,http://www.windowsphone.com/en-us/how-to/wp8/basics/what-is-the-cloud , 01 Ekim 2014 .

[2] How Google Cloud Platform Works? ,https://cloud.google.com, 01 Ekim 2014.

[3] Samantha Morris (2011), "Cloud Types: Private, Public And Hybrid.", Asigra, http://www.asigra.com/blog/cloud-types-private-public-and-hybrid, 14 Kasım 2014.

[4] Hansen, P., "The programming langugae Concurrent Pascal", (2014), IEEE Trans. Software Eng., SE-1:199 – 207.

[5] Marsden (1986), Why are standards necessary?, Uses BSC as an example to show the need for both standard protocols and a standard framework, Communication network protocols 2nd Edition.

[6] Nielsen, R., Gettys, J., Mogul, J., Nielsen, H. Ve T. Berners-Lee, (1997), "Hyper Text Transfer Protocol – HTTP/1.1", RFC 2068.

[7] Beal, V., Cloud computing (the cloud), Webopedia, http://www.webopedia.com/TERM/C/cloud_computing.html, 2013.

[8] All, A., Business And Procurement Leaders Take A More Active Role In Outsourcing, Nearshore Americas, http://www.nearshoreamericas.com/procurement-leaders-involved-inoutsourcing ,27 Kasım 2014.

[9] IDC Analyze the Future (2010). "Bulut Bilişim Dosyası", Telapi Telecom Dergisi: 69-98.

[10] Binder, R., (2014). "How to Release Rock-solid RESTful APIs and Ice the Testing BackBlob", System Verification Associates:1-42.

[11] Kızıl, B., HTTP Protokolü Nedir?, CRC64, http://www.crc64.com/http-protokolu-nedir.html, 13 Kasım 2014.

[12] Fieldin, B., RFC 2616 - IETF Tools, POST, Standarts Track, https://tools.ietf.org/html/rfc2616#section-9.5, 13 Kasım 2014.

[13] Fieldin (1999), PUT, Standarts Track, https://tools.ietf.org/html/rfc2616#section-9.6, 13 Kasım 2014.

[14] Khare, R. ve Lawrence, S., (2000). Internet RFC 2817, Upgrading to TLS Within HTTP/1. a1.

[15] Dusseault, L. ve Snell, J.,(2010), "PATCH Method for HTTP", IETF, RFC 5789: 2-10.

[16] Linhart, C. Ve Heled, R., (2005). "HTTP Request Smuggling", Watchfire. 7, WhitePaper: 1-22.

[17] Linthicum, D., (2012), "White Paper- 3 First Steps İn Building Your Own Cloud Services", Cloud Computing, InfoWorld.

[18] Comer, D., (2000). Interworking with TCP / IP: Principles, Protocols and Architecture", Prentice Hall.

[19] Kuzu, A., (2011).Bilgisayar Ağları ve İletişim, Nobel Yayınları.

[20] Kanneganti, R., Chadavarapu, P., (2008), Part 1- SOA basics", SOA Security, Manning Publishing.

[21] Environment Information Excahnge Network (2014), "The Basic of XML", EPA(United Syayes Environmental Protection Agency, www.epa.gov/cdx/about/xmlfactsheet.pdf, 26 Kasım 2014.

[22] Carlisle, D., Costello, R., (2014), "White-Paper - The XML Namespace It is Implicitly Declared It Contains 4 Great Attributes",XFront.

[23] w3schools (2014), XPath Tutorial, http://www.w3schools.com/XPath.

[24] Koftikian, J., (2001), "Project Paper Simple Object Access Protocol (SOAP)",STS Software System: 1-46.

[25] Java Sun (2008) ," Java API for XML Processing", JAVA API FOR XML PROCESSING.

[26] Kanneganti, R., Chadavarapu P., (2008), Part 2- Building blocks of SOA security, Part 3- Enterprise SOA Securityw, SOA Security, Manning Publishing.

[27] Lascelles, F. ve Flint A.,(2010), " WS Security Performance. Secure Conversation versus the X509 Profile", Jboss World Chicago.

[28] Çeviri: Mesut Timur, M., (2007).OWASP GUIDE 2.0.1", OWASP Guide Project.

[29] WSS4J 1.5.5 API (2009), Hierarchy For Package, org.apache.ws.security.components.crypto, 27 Ekim 2014.

[30] Libera, D., Microsoft ve Dixon,B., (2003)," Web Services Security Kerberos Binding", IBM Corporation, Microsoft Corporation: 1-25.

[31] Somorovsky, J. Schwenk, J.," Technical Analysis of Countermeasures against Attack on XML Encryption or Just Another Motivation for Authenticated Encryption", Horst G¨ortz Institute for IT Security Ruhr-University Bochum.

[32] Smeets, B., (2005),"Can we rely on XML Signatures?",Secure Legal Information Managment.

[33] IBM, Microsoft, RSA Security and VeriSign (2002), “Web Services Secure Conversation Language (WS-SecureConversation)”, IBM, Microsoft Version 1. 0: 1-321

[34] Oracle Corporation (2012), "Java Authentication and Authorization Service (JAAS) Reference Guide”, Java SE Documentation (Oracle).

[35] IBM WebSphere Real Time 1. 0 (2014), “ Java Generic Security Service (JGSS)” ,IBM Security Guide.

[36] Marti, M., (1993-2014)-“Single Sign-on Using Kerberos in Java”, JAVA SE Documentation (Oracle).

[37] Learn REST: A Tutorial (2014), How Simple Is REST?, http://rest.elkstein.org/, 4 Ocak 2015.

[38] Zhao, W., (2008). “Trsut managment for web services”, Web Services, 2008. ICWS '08. IEEE International Conference on :818-821, Beijing.

[39] Saleforce Developers,Single Sign-On for Desktop and Mobile Applications using SAML and OAuth, http://developer.salesforce.com/page/Single_Sign-On_for_Desktop_and_Mobile_Applications_using_SAML_and_OAuth, 12 Ocak 2015.

[40] Sosnoski, D., (2010),“Java web services: WS-Trust and WS-SecureConversation “,Apache CXF, CFX and Security Articles.

[41] Tian, X.; Cheng, Y. Ve Liu, B., (2008), “Multicast with an application-oriented networking (AON) approach,”, ICC ’08, Beijing.

[42] IBM WebSphere DataPower SOA Appliances (2009), “Understanding Web Services Policy”, IBM Corporation.

[43] Sun Java Entrprise System Deployment Planning White Paper(2004), http://docs.oracle.com/cd/E19199-01/817-5759/copyright.html, 15 Ocak 2015.

[44] Vulnerability Management Framework Development (2014), http://www.dts-solution.com/solutions/compliance-monitoring/vulnerability-management/, 13 Kasım 2014 .

[45] Shibboleth Open SAML-Java, https://shibboleth.net/products/opensaml-java.html, 13 Kasım 2014.

[46] Fielding, R., (2000). Architectural Styles And The Design Of Network-Based Software Architecture, University Of California, Irvine.

[47] Bada Developer , Samsung Services, http://developer.bada.com/devtools/samsungservices, 13 Kasım 2014.

[48] Bada Developer (2014), “Push Messaging Guide”, bada Master, http://developer.bada.com/article/push-messaging-guide, 13 Kasım 2014 .

[49] Harris, N., What are Push Notifications?, Azure Mobile Services, http://azure.microsoft.com/tr-tr/documentation/videos/cloud-cover-ep73-nick-harris-push-notificatiins/ .

[50] Siddappa, D., (2013), “ADF Mobile Push Notification With Google Cloud Messaging (GCM) Part 1”, Oracle Mobile Developer’s Guide On Push Notifications.

[51] Parse, Android Push Notifications, Anrodi Tutorials, https://parse.com/tutorials/android-push-notifications, 14 Kasım 2014.

[52] Servlet Tutorial, Simply Easy Learning by tutorialspoint.com, tutorialspoint.com, 22 Aralık 2014.

[53] Oracle Corporation(2011). “Interface Servlet”, Servlet (Java EE 6).

[54] Hall, M., (2003). Servlet Basics, Core Servlets and JavaServer Pages, Prentice Hall ve Sun Microsytems Press.

[55] Studytonight , Servlet life Cycle, Servlet Technology, Steps to create servlet using Eclipse IDE | Studytonight, http:///www.studytonight.com/servlet/servlet-life-cycle.php, 12 Aralık 2014.

[56] Java Interview Questions: - Major Difference Between Servlet And Restfull Web Service?, http://javainterviewseries.over-blog.com/article-java-interview-questions-major-difference-between-servlet-and-restfull-web-service-82568597.html, 27 Aralık 2014.

[57] Stuttard, D. ve Pinto, M., (2008). Handling Client – Side Data Securely, The Web Application Hacker’s Handbook: Discovering and Exploiting Security Flaws.

[58] Sun Microsystems (2005). Java(TM) Language Specification, Prentice Hall.

[59] Oracle Java Documentation (2014), “Lesson: Annotations”, The Java™ Tutorials.

[60] Provatal, S., org.springframework.security.access.annotation, Annotation Type Secured, http://docs.spring.io/autorepo/docs/spring-security/3.2.2.RELEASE/apidocs/org/springframework/security/access/annotation/Secured.html, 15 Kasım 2014 .

[61] .NET Framework 1.1 (2014) , Interop Marshaling Overview, Microsoft Developer Work.

[62] The Python Software Foundation (2014), 11. 5. marshal — Internal Python object serialization”, https://docs.python.org/2/library/marshal.html, 14 Kasım2014.

[63] Introducing JSON, http://www.json.org/, 13 Aralık 2014.

[64] Crockford, D., (2006). "IANA Considerations". Guidelines for Writing an IANA Considerations Sections in RFCs: 1-27.

[65] W3Schools Home (2014), JSON Tutorial, http://www.w3schools.com/json/, 13 Aralık 2014.

[66] Spring Framework 4.1.2.RELEASE API, Class DispatcherServlet, http://docs.spring.io/spring/docs/current/javadoc-api/overview-summary.html, 15 Kasım 2014 .

[67] Spring Framework 4.1.2.RELEASE API, Interface MultipartResolver, http://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/web/multipart/MultipartResolver.html, 15 Kasım 2014 .

[68] Webb, P., Syer, D., Long, J. ve Nicoll, S., (2013-2014). "Spring Boot Reference Guide", http://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#boot-features-security .

[69] Fowler, M., (2015), "Inversion of Control Containers and the Dependency Injection pattern", http://martinfowler.com, 12 Şubat 2015.

[70] Schwarz, N., Lungu, M., Nierstrasz, O., (2012), "Seuss: Decoupling responsibilities from static methods for fine-grained configurability", Journal of Object Technology: 1-23.

[71] Simple Easy Learning (2011), "Spring Dependency Injection", Spring Framework Tutorial.

[72] Webb, P., "Annotation Type EnableAutoConfiguration", http://docs.spring.io/autorepo/docs/spring-boot/1.0.0.RC4/api/org/springframework/boot/autoconfigure/EnableAutoConfiguration.html, 15 Kasım 2014.

[73] Beams, C. Ve Hoeller, J., "Annotation Type ComponentScan", http://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/context/annotation/ComponentScan.html, 15 Kasım 2014.

[74] Johson, R. Ve Beams, C., Annotation Type Configuration, http://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/context/annotation/Configuration.html, 15 Kasım 2014.

[75] Ambler, S., Lines, M., (2013), Disclilined Agile Delivery(DAD), An IBM Press Book.

[76] Brisbin, J. ve Gierke, O., (2012-204), " Spring Data REST Reference Documentation", Spring Documentation.

[77] Google Developers, Sessions - Android SDK v2 (Legacy), https://developers.google.com/analytics/devguides/collection/android/v2/sessions?hl=tr, 27 Kasım 2014.

[78] SSL Shopper, Public Key Infrastructure (PKI) Overview, https://www.sslshopper.com/public-key-infrastructure-pki-overview.html, 15 Kasım 2014.

[79] Ben, A. ve Luke, T., (2013), "Spring Security, Reference Documentation.", Spring Docs 3.0.8 Versiyonu.

[80] Tomcat Administration (2014), Annotation Type PreAuthorize, Spring Docs, http://docs.spring.io/autorepo/docs/spring-security/3.2.2 RELEASE/apidocs/overview-summar.html, 23 Ekim 2014.

[81] Tomcat Administration (2014), Annotation Type PostAuthorize, Spring Docs, http://docs.spring.io/autorepo/docs/spring-security/3.2.2 RELEASE/apidocs/overview-summar.html, 23 Ekim 2014.

[82] Wilander, J., (2011), "The Relation Between Sessions and Authentication, REST and Stateless Session IDs", Apps and Security.

[83] OAuth (2014), OAuth 2. 0, http://oauth.net/2/, 15 Kasım 2014 .

[84] Hammer, E.i (2010). "Introducing OAuth 2. 0", Huniverse: 1-20.

[85] Slaesforce Success Community (2014), "OAuth 2. 0 Username - Password Flow", OAuth 2. 0 Username – Password Flow.

[86] OAuthlib (2014), "Grant Types", OAuth 2. 0, National Institutes of Health.

[87] OAuth Token Overview, OAuth 2.0 APIs, https://developers.digitalriver.com/book/export/html/1665, 16 Kasım 2014.

[88] Oracle (2014), "37 Understanding Mobile and Social", Fusion Middleware Administrator's Guide for Oracle Access Managment.

[89] Microsoft Technet (2003), "How Access Tokens Work", Authorization and Access Control Technolohies.

[90] Jenkov, J., OAuth 2.0 Tutorial", http://tutorials.jenkov.com/oauth2/index.html, 16 Kasım 2014.

[91] Singing Horse Stuio, Horizontal vs Vertical Scaling", http://singinghorsestudio.com/horizontal-vs-vertical-scaling/, 16 Kasım 2014.

[92] Kriushanth, M., Arockiam, L. ve Mirobi, G., (2013), "Auto Scaling in Cloud Computing: An Overview", International Journal of Advanced Research in Computer and Communication Engineering, Vol. 2, IJARCCE: 1-6.

[93] Openstack Cloud Software, Stateless vs. Stateful Services, http://docs.openstack.org/high-availability-guide/content/stateless-vs-stateful.html, 16 Kasım 2014 .

[94] Brocade Communications Systems (2009), "Stateless Server Load Balancing", Server Iron SLB Guide.

[95] Luis, M., Buyya, R., (2011), "Dynamically Scaling Applications in the Cloud", ACM SIGCOMM Computer Communication Review: 45-51.

[96] Amies, A., Sluiman, H., Tong, G., Liu, N., (2012), Infrastructure as a Service Cloud Concepts, Developing and Hosting Applications on the Cloud, IBM Press.

[97] Mell, P., (2011). "The NIST Definition of Cloud Computing", National Institute of Science and Technology: 1-3.

[98] Diversity Limited (2011) ,"UNDERSTANDING The Cloud Computing Stack SaaS, Paas, IaaS", Cloud: 1-20, Unites States.

[99] Britch, D. Ve Cheung, F., (2012). Developing an Advanced Windows® Phone 7. 5 App that Connects to the Cloud.Microsoft.

[100] UX Services (2014), Mobil Uygulama projelerinde başarı için 10 Uluslararası İş Analizi ve Tasarım Prensibi, http://www.uxservices.com/blog/mobil-

uygulama-projelerinde-basari-icin-10-uluslararasi-analizi-ve-tasarim-prensibi/, 20 Ocak 2015.

[101] Nokia Developer, Usability Testing: Key for developing high quality mobile applications, http://developer.nokia.com/community/wiki/Usability_Testing:_Key_for_developing_high_quality_mobile_applications, 12 Ocak 2015.

[102] Postacı, B., Sertçelik, M., (2009). Yazılım Gereksinim Dökümanı, BCM Hastane Otomasyonu, Fatih Ünivesitesi Yazılım Mühendisliği.

[103] Web Ascender, Our Mobile Application Strategy, http://www.webascender.com/Services/Mobile-Development, 16 Kasım 2014.

[104] The Best-Run Businesses Run SAP (2014). "Installation of Mobile Client Components", Implementation of CRM Mobile Client Applications, Versiyon 7.0.

[105] Meier, J. ve Homer, A., (2008), "Mobile Application Architecture", Mobile Application Architecture Guide Application Architecture Pocket Guide Series, Microsoft.

[106] Saxena, S., (2013), "Windows Phone: Page Navigation", The Code Project Open License (CPOL) 1. 02.

[107] Windows 8 Design And Coding Guidelines Updated For Windows 8. 1, Microsoft, www.go.microsoft.com/fwlink/p/?linkid=258743, 10 Ocak 2015.

[108] Microsoft MSDN (2014) ,App tabs (Pivot control) for Windows Phone, Pivot Control For Windows Phone 8.

[109] Flores, H. ve Srirama, S., (2014), " Computational Offloading or Data Binding? Bridging the Cloud Infrastructure to the Proximity of the Mobile User" ,Mobile Cloud Computing, Services, and Engineering (MobileCloud), 2014 2nd IEEE International Conference: 10-18.

[110] Microsoft Tech Net, Simple Data Binding in XAML, http://social.technet.microsoft.com/wiki/contents/articles/9645.simple-data-binding-in-xaml.aspx, 16 Kasım 2014.

[111] Emotive Docs V3.0, Distributing Applications to Users, http://docs.emotive.com/pages/advanced-acl-intro.html, 16 Kasım 2014.

[112] Sarah, P., (2010), "Mobile Data Traffic Surge: 40 Exabytes by 2014", ReadWrite.

[113] Blankenburg, J., (2010), "31 Days Of Wındows Phone | Day #15: Isolated Storage", http://jeffblankenburgdotcom.wordpress.com/2010/10/15/31-days-of-windows-phone-day-15-isolated-storage/, 16 Kasım 2014.

[114] Microsoft Patterns & Practices, Using Isolated Storage on the Phone", http://msdn.microsoft.com/en-us/library/hh821020.aspx, 16 Kasım 2014.

[115] Lecrenski, N. ve Watson, K., (2011), "Making Asynchronous Calls, Windows Phone 7 Application Development", Wiley Publishing.

[116] Bourne, J., (2012). A Future Claims-Based Approach", Windows Phone 7 Developer Guide.

[117] Rad Cloud Data Sync, UI for Windows Phone", http://www.telerik.com/products/windows-phone/overview/all-controls/clouddatasync.aspx, 16 Kasım 2014 .

[118] Kaspersky Lab (2014), "White Paper - Kaspersky Administration Kit 8. 0", Why Administration Agent fails to establish connection with the Administration Server.

[119] Milliyet, Dropbox Hacklendi! 7 Milyon Şifre Çalındı!, http://www.milliyet.com.tr/dropbox-hacklendi-7-milyon-sifre-internet-1954247/, 14 Ekim 2014 .

[120] Rycke's, R., "Google OAuth 2. 0 on Windows Phone, REST Windows Phone 8", http://pieterderycke.wordpress.com/2013/03/11/google-oauth-2-0-on-windows-phone/, 21 Ocak 2015.

[121] Microsoft Msdn, Push notifications for Windows Phone 8, http://msdn.microsoft.com/en-us/library/windows/apps/ff402558(v=vs.105).aspx, 16 Kasım 2014.

[122] Microsoft Msdn, Sending push notifications for Windows Phone 8, http://msdn.microsoft.com/en-us/library/windows/apps/hh202945(v=vs.105).aspx, 16 Kasım 2014.

[123] Chappell, D., (2011), "Introducing Odata, Data Access For The Web, The Cloud, Mobile Devices, And More", Data Developer Center Articles, Microsoft Msdn, David Chappel & Associates: 1-24.

[124] Lascelles, F., (2006). "Aaron Flint: WS Security Performance", Secure Conversation versus the X509 Profile: 17-22.

[125] Korkmaz, Y., *.NET Framework* / Web Service 'lerinde Güvenlik, 14 Kasım 2014.

[126] Rouse, M., Transport Layer Security (TLS), http://searchsecurity.techtarget.com/defination/Transport-Layer-Security-TLS, 20 Kasım 2014..

[127] Defination of TLS, http://www.pcmag.com/encyclopedia/term/52944/tls, 20 Kasım 2014 .

[128] Wike, A., (2014), Difference between SSL and TLS, http://www.wolfssl.com/wolfSSL/Blog/Entries/2010/10/7_Differences_between_SSL_and_TLS_Protocol_Versions.html, 23 Kasım 2014.

[129] Marquez, J., (2001), "Kerberos: Secure Authentıcatıon", SANS Institude InfoSec Reading Room: 2-10.

[130] Fielding, T. ve Gettys, J., (1999). "*Hypertext Transfer Protocol", HTTP/1.1*. IETF. RFC 2616, Haziran 1999.

[131] Vinton G. ve Kahn, R., (1974), "A Protocol for Packet Network Intercommunication". IEEE Transactions on Communications, 22:2-13.

[132] Mirsat Yeşiltepe (2015), “Servis Odaklı Mimari Güvenliğinde Güvenlik Tiplerinin Karşılaştırılması”, Akademik Bilişim 2015, 31 Ocak – 6 Şubat 2015, Eskişehir.

[133] Allen, S., (2012), “A Software Developer's Guide to HTTP Part III– Connections”, PluralSight Hardcore Dev And IT Training.

[134] <wsHttpBinding>, <netTcpBinding>, Developer Network, https://msdn.microsoft.com/en-us/library/ms731399(v=vs.110).aspx, 20 Nisan 2015 .

[135] <bindings>, WCF Configuration Schema, https://msdn.microsoft.com/en-us/library/ms731346%28v=vs.110%29.aspx, 20 Nisan 2015.

[136] Ayyıldız, E., Doğanaksoy, A., (2004), Kriptolojiye Giriş Ders Notları, Uygulamalı Matematik Enstitüsü, Kriptoloji Bölümü ODTÜ.

[137] İTÜ Bilgi İşlem Daire Başkanlığı, “Seyir Defteri, Şifreleme Yöntemleri, Seyir Defteri”, (2013), İTÜ Bilgi İşlme Daire Başkanlığı, http://bidb.itu.edu.tr/seyirdefteri/blog/2013/09/07/%C5%9Fifreleme-y%C3%B6ntemleri, 20 Nisan 2015.

[138] Yanpei, C., Rean, G., (2012). “Understanding TCP Incast and Its Implications for Big Data Workloads”, USENIX; Login Magazine, Haziran 2012.

## EK-A

## TEST ORTAMININ OLUŞTURULMASI

Bu bölümde test ortamının sunucu, Windows Phone istemcisi ve Android istemcisi açısından kurulmasında izlenen adımlar incelenecektir. Bu test ortamında amaç bir bulut servisi oluşturulacak farklı işletim sitemlerinin bu buluta bağlanım şekilleri ve ortamla olan iletişiminin izlenmesidir. Örnek gösterilen kod parçacıkları test edilen bir durumda elde edilmiş kod parçacığıdır. Test durumunun değişmesi ilgili kod parçacıklarının değişmesine neden olmuştur.

### A-1 Test Ortamındaki Sunucunun Oluşturulması

WCF ortamında oluşturulacak sunucunun inşasında şu adımlar izlenmiştir.

1. İlk olarak sunucunun sunacağı hizmetlerin sadece isimlerinin ve yollarının belirleneceği arayüz oluşturulup, bu arayüz temel alınarak hizmet sınıfı oluşturulmuştur.

GetMessage() fonksiyonun ilk parametresi mesajlaşmanın kaç kere yapılacağı ikinci parametresi ise mesajlaşmada kullanılacak şansal (rassal) karakterlerle oluşturulmuş mesajın karakter uzunluğunu belirtir. Bu fonksiyon web servisi kullanan istemciler tarafından sadece isim ve aldığı parametreler ile bilinecektir. RandomString fonksiyonun parametresi şansal karakterlerle oluşturulacak iletişim mesajının karakter uzunluğu parametresidir.

```
[ServiceContract]
public interface ISimpleService
{
    [OperationContract]
    string GetMessage(double name, double name1);

    string RandomString(double size);
}
```

Şekil A. 1 Oluşturulan arayüz kod parçacığı

```
public string GetMessage(double name, double name1)
{
    return name+ RandomString(name1);
}


public string RandomString(double size)
{
    StringBuilder builder = new StringBuilder();
    char ch;

    for (double i = 0; i < size; i++)
    {
        ch = Convert.ToChar(Convert.ToInt32(Math.Floor(26 * random.NextDouble() + 65)));
        builder.Append(ch);
    }

    return builder.ToString();
}
```

Şekil A. 2 Arayüz tarafından oluşturulan hizmet sınıfı

2. İstemci ve sunucu arasındaki iletişimde köprü görevini üstlenecek App.Config yapılandırma (yapılandırma) dosyası oluşturulur.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <system.diagnostics>
    <sources>
      <source name="System.ServiceModel.MessageLogging" switchValue="Warning, ActivityTracing">
        <listeners>
          <add type="System.Diagnostics.DefaultTraceListener" name="Default">
            <filter type="" />
          </add>
          <add name="ServiceModelMessageLoggingListener">
            <filter type="" />
          </add>
        </listeners>
      </source>
    </sources>
    <sharedListeners>
      <add initializeData="D:\app_messages.svclog"
type="System.Diagnostics.XmlWriterTraceListener, System, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089"
        name="ServiceModelMessageLoggingListener" traceOutputOptions="Timestamp">
        <filter type="" />
      </add>
```

```
    </sharedListeners>
    <trace autoflush="true" />
  </system.diagnostics>
  <system.serviceModel>
    <diagnostics performanceCounters="All">
      <messageLogging logEntireMessage="true" logMalformedMessages="true"
        logMessagesAtTransportLevel="true" maxMessagesToLog="999999999"
        maxSizeOfMessageToLog="999999999" />
    </diagnostics>
    <bindings>
      <wsHttpBinding>
        <binding name="netTcp" maxBufferPoolSize="99999999" maxReceivedMessageSize="99999999">
          <security mode="Message">
            <message algorithmSuite="TripleDesSha256" />
          </security>
        </binding>
      </wsHttpBinding>
    </bindings>
    <behaviors>
      <serviceBehaviors>
        <behavior name="mexBehavior">
          <serviceMetadata httpGetEnabled="true" />
        </behavior>
      </serviceBehaviors>
    </behaviors>
    <services>
      <service behaviorConfiguration="mexBehavior" name="SimpleService.SimpleService">
        <endpoint address="SimpleService"
  binding="wsHttpBinding"
  contract="SimpleService.ISimpleService"
  bindingConfiguration="netTcp"/>

        <host>
          <baseAddresses>
            <add baseAddress="http://localhost:8080" />

          </baseAddresses>
        </host>
      </service>
    </services>
  </system.serviceModel>
</configuration>
```

Şekil A. 3 Testte kullanılan yapılandırma dosyasının bir durum için örneği

Şekildeki yapılandırma dosyası kullanılan test durumuna göre şifreleme algoritmasının çeşidi, kullanılan bağlamanın değişmesi gibi durumlarda güncellenerek sunucunun yeni ortama uyumu sağlanmıştır.

3. Servis dosyasının oluşturulur. Servis parametresi ile sunulacak hizmetin (servisin) bulunduğu konum belirtilir.

```
<%@ ServiceHost Language="C#" Debug="true" Service="SimpleService.SimpleService" %>
```

Şekil A. 4 Svc uzantılı servis dosyasının içeriği

4. Sunucunun yayımlanmasında ISS'ın TCP protokolü kabul etmemesi yüzünden, WAS yayımlama yöntemi seçilmiştir. WAS, İnternet Bilgi Servisleri sürüm 7. 0

içinde tanıtılan işlem aktivasyon mekanizmasıdır. Windows servisi yayınlama yönteminin seçilmemesinin nedeni Android istemcilerininde ortama bağlanabilmesini sağlayabilmektir.

Sonuç olarak TCP protokülü kullanılabildiği yerlerde ortama Andorid istemcisinin katılnması sağlanmış, diğer durumlarda Android istemcisi sadece HTTP protokolünün kullanıldığı zamanda ortama katılması sağlanmıştır.

**A-2 Test Ortamındaki Windows Phone İstemcisinin Oluşturulması**

1. Uygulamaya sunucudaki servis, servis referansı olarak (servisin hizmet vereceği URL adresi belirtilerek) eklenir. Bu URL sayesinde ilgili WSDL dosyasına ulaşılır.

```
for (int i = 1; i <= 10000; i++)
{
    SimpleService.SimpleServiceClient client = new SimpleService.SimpleServiceClient();
    ListBox1.Items.Add(client.GetMessage(i, 6));
    client.Close();
}
```

Şekil A. 5 Windows Phone istemcisinin servisi çağırma kod parçası örneği

Servisi kullanan istemci oluşturulup, servisteki hizmet fonksiyonu ilgili parametreler kullanılarak çağrılmıştır. client.Close() fonksiyonu ile istemcinin test ortamında kopması sağlanmıştır.

**A-3 Test Ortamındaki Android İstemcisinin Oluşturulması**

1. Web servis referansı ile sunucudaki hizmet URL üzerinden bağlanılır. Esasen bu URL ile bağlantı sunucu tarafından oluşturulan hizmetin WSDL dosyası üzerinden sağlanır. Wsdl dosyası sunucu tarafından oluşturulan bilgiler üzerinden düzenlendiği için burada sadece dosyanın bir bölümü gösterilmiştir. Wsdl dosyası istemci ve sunucu arasındaki iletişimde iletişim kurallarının yer aldığı bir sözleşme olarak düşünülebilir.

```
▼<wsdl:service name="SimpleService">
  ▼<wsdl:port name="WSHttpBinding_ISimpleService" binding="tns:WSHttpBinding_ISimpleService">
     <soap12:address location="http://localhost:4228/SimpleService.svc/SimpleService"/>
    ▼<wsa10:EndpointReference>
      ▼<wsa10:Address>
         http://localhost:4228/SimpleService.svc/SimpleService
       </wsa10:Address>
      ▼<Identity xmlns="http://schemas.xmlsoap.org/ws/2006/02/addressingidentity">
         <Upn>GALATA\MUHTIP</Upn>
       </Identity>
     </wsa10:EndpointReference>
   </wsdl:port>
 </wsdl:service>
</wsdl:definitions>
```

Şekil A. 6 Wsdl dosyası kod parçası örneği

2. Hizmet servisi uygulama sayfasında bildirilerek ilgili hizmetler uygulamada kullanılır.

```
private static String getMessage(java.lang.Double name, java.lang.Double name1) {
    org.tempuri.SimpleService service = new org.tempuri.SimpleService();
    org.tempuri.ISimpleService port = service.getWSHttpBindingISimpleService();
    return port.getMessage(name, name1);
}
```

String a= getMessage(symbol,symbol1);

Şekil A. 7 Hizmet sınıfının uygulama üzerinde bildirimi ve bu hizmetin kullanım örneği

## EK-B

## TEST ORTAMINDA ELDE EDİLEN VERİLER

Bu bölümde WCF'de mesaj seviye güvenlikte kullanılan şifreleme algoritmalarına, bu algoritmalara kullanılarak elde edilen verilere yer verilecektir. Bölüm 12'de ise elde edilen veriler değerlendirilmiştir.

### B - 1 WCF'de Mesaj Seviye Güvenlikte Kullanılan Şifreleme Algoritmaları

Çizelge B.1 WCF'de Mesaj Seviye Güvenlikte Kullanılan Şifreleme Algoritmaları

| WCF'de Mesaj Seviye Güvenlikte Kullanılan Şifreleme Algoritmaları | |
|---|---|
| 1-Basic128 | 9-Basic256 |
| 2-Basic128Rsa15 | 10-Basic256Rsa15 |
| 3-Basic128Sha256 | 11-Basic256Sha256 |
| 4-Basic128Sha256Rsa15 | 12-Basic256Sha256Rsa15 |
| 5-Basic192 | 13-TripleDes |
| 6-Basic192Rsa15 | 14-TripleDesRsa15 |
| 7-Basic192Sha256 | 15-TripleDesSha256 |
| 8-Basic192Sha256Rsa15 | 16-TripleDesSha256Rsa15 |

Aşağıdaki grafiklerde yer alan kısaltmaların açıklamaları ilgili bölümlerde değinilmiştir.

### B - 2 Tek İstemcili Değişken Kelime Katarlı Test İşlemi Verileri

### B - 2. 1 NetTcpBinding Bağlama Kullanılarak Elde Edilen Veriler

Çizelge B.2 Güvenliksiz Elde Edilen Test Verileri

| 0-Güvenliksiz | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 100 | 2 | 0,186 | 4 | 0,02 | 0,00186 | 0,04 |
| 200 | 2 | 0,189 | 4 | 0,01 | 0,000945 | 0,02 |
| 300 | 2 | 0,192 | 4 | 0,00667 | 0,00064 | 0,01333 |
| 400 | 2 | 0,186 | 4 | 0,005 | 0,000465 | 0,01 |
| 500 | 2 | 0,184 | 4 | 0,004 | 0,000368 | 0,008 |

Çizelge B.2 Güvenliksiz Elde Edilen Test Verileri (devamı)

| Adet | T | S | B | Tverim | Sverim | Bverim |
|---|---|---|---|---|---|---|
| 4000 | 2 | 0,19 | 7 | 4,75E-05 | 4,8E-05 | 0,00175 |
| 5000 | 2 | 0,199 | 8 | 3,98E-05 | 4E-05 | 0,0016 |
| 6000 | 2 | 0,219 | 9 | 3,65E-05 | 3,7E-05 | 0,0015 |
| 7000 | 2 | 0,198 | 10 | 2,83E-05 | 2,8E-05 | 0,00143 |
| 8000 | 2 | 0,224 | 11 | 0,000028 | 2,8E-05 | 0,00138 |

Çizelge B. 3 Basic128 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 1- Basic128 | | | | | | |
|---|---|---|---|---|---|---|
| Adet | T | S | B | Tverim | Sverim | Bverim |
| 100 | 10 | 1 | 47 | 0,1 | 0,01 | 0,47 |
| 200 | 10 | 1 | 47 | 0,05 | 0,005 | 0,235 |
| 300 | 10 | 1 | 47 | 0,033333333 | 0,003333333 | 0,156666667 |
| 400 | 10 | 1 | 47 | 0,025 | 0,0025 | 0,1175 |
| 500 | 10 | 1 | 48 | 0,02 | 0,002 | 0,096 |
| 4000 | 10 | 1 | 52 | 0,0025 | 0,00025 | 0,013 |
| 5000 | 10 | 1 | 53 | 0,002 | 0,0002 | 0,0106 |
| 6000 | 10 | 1 | 55 | 0,001666667 | 0,000166667 | 0,009166667 |
| 7000 | 10 | 1 | 56 | 0,00142857 | 0,00014285 | 0,008 |
| 8000 | 10 | 1 | 57 | 0,00125 | 0,000125 | 0,007125 |

Çizelge B. 4 Basic128Rsa15 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 2- Basic128Rsa15 | | | | | | |
|---|---|---|---|---|---|---|
| Adet | T | S | B | Tverim | Sverim | Bverim |
| 100 | 10 | 1 | 47 | 0,1 | 0,01 | 0,47 |
| 200 | 10 | 1 | 47 | 0,05 | 0,005 | 0,235 |
| 300 | 10 | 1 | 47 | 0,033333333 | 0,003333333 | 0,156666667 |
| 400 | 10 | 1 | 47 | 0,025 | 0,0025 | 0,1175 |
| 500 | 10 | 1 | 48 | 0,02 | 0,002 | 0,096 |
| 4000 | 10 | 1 | 52 | 0,0025 | 0,00025 | 0,013 |
| 5000 | 10 | 1 | 53 | 0,002 | 0,0002 | 0,0106 |
| 6000 | 10 | 1 | 55 | 0,001666667 | 0,000166667 | 0,009166667 |
| 7000 | 10 | 1 | 56 | 0,001428571 | 0,000142857 | 0,008 |
| 8000 | 10 | 1 | 57 | 0,00125 | 0,000125 | 0,007125 |

Çizelge B. 5 Basic128Sha256 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 3- Basic128Sha256 | | | | | | |
|---|---|---|---|---|---|---|
| Adet | T | S | B | Tverim | Sverim | Bverim |
| 100 | 10 | 1 | 48 | 0,1 | 0,01 | 0,48 |
| 200 | 10 | 1 | 48 | 0,05 | 0,005 | 0,24 |

Çizelge B. 5 Basic128Sha256 Şifreleme Algoritması İle Elde Edilen Test Verileri (devamı)

| Adet | T | S | B | Tverim | Sverim | Bverim |
|---|---|---|---|---|---|---|
| 300 | 10 | 1 | 48 | 0,033333333 | 0,003333333 | 0,16 |
| 400 | 10 | 1 | 48 | 0,025 | 0,0025 | 0,12 |
| 500 | 10 | 1 | 48 | 0,02 | 0,002 | 0,096 |
| 4000 | 10 | 1 | 53 | 0,0025 | 0,00025 | 0,01325 |
| 5000 | 10 | 1 | 54 | 0,002 | 0,0002 | 0,0108 |
| 6000 | 10 | 1 | 56 | 0,001666667 | 0,000166667 | 0,009333333 |
| 7000 | 10 | 1 | 57 | 0,001428571 | 0,000142857 | 0,008142857 |
| 8000 | 10 | 1 | 58 | 0,00125 | 0,000125 | 0,00725 |

Çizelge B. 6 Basic128Sha256Rsa15 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 4-Basic128Sha256Rsa15 | | | | | | |
|---|---|---|---|---|---|---|
| Adet | T | S | B | Tverim | Sverim | Bverim |
| 100 | 10 | 1 | 48 | 0,1 | 0,01 | 0,48 |
| 200 | 10 | 1 | 48 | 0,05 | 0,005 | 0,24 |
| 300 | 10 | 1 | 48 | 0,033333333 | 0,003333333 | 0,16 |
| 400 | 10 | 1 | 48 | 0,025 | 0,0025 | 0,12 |
| 500 | 10 | 1 | 48 | 0,02 | 0,002 | 0,096 |
| 4000 | 10 | 1 | 53 | 0,0025 | 0,00025 | 0,01325 |
| 5000 | 10 | 1 | 54 | 0,002 | 0,0002 | 0,0108 |
| 6000 | 10 | 1 | 56 | 0,001666667 | 0,000166667 | 0,009333333 |
| 7000 | 10 | 1 | 57 | 0,001428571 | 0,000142857 | 0,008142857 |
| 8000 | 10 | 1 | 58 | 0,00125 | 0,000125 | 0,00725 |

Çizelge B. 7 Basic192 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 5-Basic192 | | | | | | |
|---|---|---|---|---|---|---|
| Adet | T | S | B | Tverim | Sverim | Bverim |
| 100 | 10 | 1 | 47 | 0,1 | 0,01 | 0,47 |
| 200 | 10 | 1 | 47 | 0,05 | 0,005 | 0,235 |
| 300 | 10 | 1 | 47 | 0,033333333 | 0,003333333 | 0,156666667 |
| 400 | 10 | 1 | 47 | 0,025 | 0,0025 | 0,1175 |
| 500 | 10 | 1 | 48 | 0,02 | 0,002 | 0,096 |
| 4000 | 10 | 1 | 52 | 0,0025 | 0,00025 | 0,013 |
| 5000 | 10 | 1 | 53 | 0,002 | 0,0002 | 0,0106 |
| 6000 | 10 | 1 | 55 | 0,001666667 | 0,000166667 | 0,009166667 |
| 7000 | 10 | 1 | 56 | 0,001428571 | 0,000142857 | 0,008 |
| 8000 | 10 | 1 | 57 | 0,00125 | 0,000125 | 0,007125 |

Çizelge B. 8 Basic192Rsa15 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 6-Basic192Rsa15 | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 100 | 10 | 1 | 47 | 0,1 | 0,01 | 0,47 |
| 200 | 10 | 1 | 47 | 0,05 | 0,005 | 0,235 |
| 300 | 10 | 1 | 47 | 0,033333333 | 0,003333333 | 0,156666667 |
| 400 | 10 | 1 | 47 | 0,025 | 0,0025 | 0,1175 |
| 500 | 10 | 1 | 48 | 0,02 | 0,002 | 0,096 |
| 4000 | 10 | 1 | 52 | 0,0025 | 0,00025 | 0,013 |
| 5000 | 10 | 1 | 53 | 0,002 | 0,0002 | 0,0106 |
| 6000 | 10 | 1 | 55 | 0,001666667 | 0,000166667 | 0,009166667 |
| 7000 | 10 | 1 | 56 | 0,001428571 | 0,000142857 | 0,008 |
| 8000 | 10 | 1 | 57 | 0,00125 | 0,000125 | 0,007125 |

Çizelge B. 9 Basic192Sha256 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 7-Basic192Sha256 | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 100 | 10 | 1 | 48 | 0,1 | 0,01 | 0,48 |
| 200 | 10 | 1 | 48 | 0,05 | 0,005 | 0,24 |
| 300 | 10 | 1 | 48 | 0,033333333 | 0,003333333 | 0,16 |
| 400 | 10 | 1 | 48 | 0,025 | 0,0025 | 0,12 |
| 500 | 10 | 1 | 48 | 0,02 | 0,002 | 0,096 |
| 4000 | 10 | 1 | 53 | 0,0025 | 0,00025 | 0,01325 |
| 5000 | 10 | 1 | 54 | 0,002 | 0,0002 | 0,0108 |
| 6000 | 10 | 1 | 56 | 0,001666667 | 0,000166667 | 0,009333333 |
| 7000 | 10 | 1 | 57 | 0,001428571 | 0,000142857 | 0,008142857 |
| 8000 | 10 | 1 | 58 | 0,00125 | 0,000125 | 0,00725 |

Çizelge B. 10 Basic192Sha256Rsa15 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 8-Basic192Sha256Rsa15 | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 100 | 10 | 1 | 48 | 0,1 | 0,01 | 0,48 |
| 200 | 10 | 1 | 48 | 0,05 | 0,005 | 0,24 |
| 300 | 10 | 1 | 48 | 0,033333333 | 0,003333333 | 0,16 |
| 400 | 10 | 1 | 48 | 0,025 | 0,0025 | 0,12 |
| 500 | 10 | 1 | 48 | 0,02 | 0,002 | 0,096 |
| 4000 | 10 | 1 | 53 | 0,0025 | 0,00025 | 0,01325 |
| 5000 | 10 | 1 | 54 | 0,002 | 0,0002 | 0,0108 |
| 6000 | 10 | 1 | 56 | 0,001666667 | 0,000166667 | 0,009333333 |
| 7000 | 10 | 1 | 57 | 0,001428571 | 0,000142857 | 0,008142857 |
| 8000 | 10 | 1 | 58 | 0,00125 | 0,000125 | 0,00725 |

Çizelge B. 11 Basic256 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 9-Basic256 | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 100 | 10 | 1 | 47 | 0,1 | 0,01 | 0,47 |
| 200 | 10 | 1 | 47 | 0,05 | 0,005 | 0,235 |
| 300 | 10 | 1 | 47 | 0,033333333 | 0,003333333 | 0,156666667 |
| 400 | 10 | 1 | 47 | 0,025 | 0,0025 | 0,1175 |
| 500 | 10 | 1 | 47 | 0,02 | 0,002 | 0,094 |
| 4000 | 10 | 1 | 52 | 0,0025 | 0,00025 | 0,013 |
| 5000 | 10 | 1 | 53 | 0,002 | 0,0002 | 0,0106 |
| 6000 | 10 | 1 | 55 | 0,001666667 | 0,000166667 | 0,009166667 |
| 7000 | 10 | 1 | 56 | 0,001428571 | 0,000142857 | 0,008 |
| 8000 | 10 | 1 | 57 | 0,00125 | 0,000125 | 0,007125 |

Çizelge B. 12 Basic256Rsa15 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 10-Basic256Rsa15 | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 100 | 10 | 1 | 47 | 0,1 | 0,01 | 0,47 |
| 200 | 10 | 1 | 47 | 0,05 | 0,005 | 0,235 |
| 300 | 10 | 1 | 47 | 0,033333333 | 0,003333333 | 0,156666667 |
| 400 | 10 | 1 | 47 | 0,025 | 0,0025 | 0,1175 |
| 500 | 10 | 1 | 47 | 0,02 | 0,002 | 0,094 |
| 4000 | 10 | 1 | 52 | 0,0025 | 0,00025 | 0,013 |
| 5000 | 10 | 1 | 53 | 0,002 | 0,0002 | 0,0106 |
| 6000 | 10 | 1 | 55 | 0,001666667 | 0,000166667 | 0,009166667 |
| 7000 | 10 | 1 | 56 | 0,001428571 | 0,000142857 | 0,008 |
| 8000 | 10 | 1 | 57 | 0,00125 | 0,000125 | 0,007125 |

Çizelge B. 13 Basic256Sha256 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 11-Basic256Sha256 | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 100 | 10 | 1 | 48 | 0,1 | 0,01 | 0,48 |
| 200 | 10 | 1 | 48 | 0,05 | 0,005 | 0,24 |
| 300 | 10 | 1 | 48 | 0,033333333 | 0,003333333 | 0,16 |
| 400 | 10 | 1 | 48 | 0,025 | 0,0025 | 0,12 |
| 500 | 10 | 1 | 48 | 0,02 | 0,002 | 0,096 |
| 4000 | 10 | 1 | 53 | 0,0025 | 0,00025 | 0,01325 |
| 5000 | 10 | 1 | 54 | 0,002 | 0,0002 | 0,0108 |

Çizelge B. 13 Basic256Sha256 Şifreleme Algoritması İle Elde Edilen Test Verileri (devamı)

| Adet | T | S | B | Tverim | Sverim | Bverim |
|---|---|---|---|---|---|---|
| 6000 | 10 | 1 | 55 | 0,001666667 | 0,000166667 | 0,009166667 |
| 7000 | 10 | 1 | 57 | 0,001428571 | 0,000142857 | 0,008142857 |
| 8000 | 10 | 1 | 58 | 0,00125 | 0,000125 | 0,00725 |

Çizelge B. 14 Basic256Sha256Rsa15 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 12-Basic256Sha256Rsa15 | | | | | | |
|---|---|---|---|---|---|---|
| Adet | T | S | B | Tverim | Sverim | Bverim |
| 100 | 10 | 1 | 48 | 0,1 | 0,01 | 0,48 |
| 200 | 10 | 1 | 48 | 0,05 | 0,005 | 0,24 |
| 300 | 10 | 1 | 48 | 0,033333333 | 0,003333333 | 0,16 |
| 400 | 10 | 1 | 48 | 0,025 | 0,0025 | 0,12 |
| 500 | 10 | 1 | 48 | 0,02 | 0,002 | 0,096 |
| 4000 | 10 | 1 | 53 | 0,0025 | 0,00025 | 0,01325 |
| 5000 | 10 | 1 | 54 | 0,002 | 0,0002 | 0,0108 |
| 6000 | 10 | 1 | 55 | 0,001666667 | 0,000166667 | 0,009166667 |
| 7000 | 10 | 1 | 57 | 0,001428571 | 0,000142857 | 0,008142857 |
| 8000 | 10 | 1 | 58 | 0,00125 | 0,000125 | 0,00725 |

Çizelge B. 15 TripleDesRsa15 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 13-TripleDes | | | | | | |
|---|---|---|---|---|---|---|
| Adet | T | S | B | Tverim | Sverim | Bverim |
| 100 | 10 | 1 | 47 | 0,1 | 0,01 | 0,47 |
| 200 | 10 | 1 | 47 | 0,05 | 0,005 | 0,235 |
| 300 | 10 | 1 | 47 | 0,033333333 | 0,003333333 | 0,156666667 |
| 400 | 10 | 1 | 47 | 0,025 | 0,0025 | 0,1175 |
| 500 | 10 | 1 | 47 | 0,02 | 0,002 | 0,094 |
| 4000 | 10 | 1 | 52 | 0,0025 | 0,00025 | 0,013 |
| 5000 | 10 | 1 | 53 | 0,002 | 0,0002 | 0,0106 |
| 6000 | 10 | 1 | 55 | 0,001666667 | 0,000166667 | 0,009166667 |
| 7000 | 10 | 1 | 56 | 0,001428571 | 0,000142857 | 0,008 |
| 8000 | 10 | 1 | 57 | 0,00125 | 0,000125 | 0,007125 |

Çizelge B. 16 TripleDesRsa15 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 14-TripleDesRsa15 | | | | | | |
|---|---|---|---|---|---|---|
| Adet | T | S | B | Tverim | Sverim | Bverim |
| 100 | 10 | 1 | 47 | 0,1 | 0,01 | 0,47 |
| 200 | 10 | 1 | 47 | 0,05 | 0,005 | 0,235 |
| 300 | 10 | 1 | 47 | 0,033333333 | 0,003333333 | 0,156666667 |

Çizelge B. 16 TripleDesRsa15 Şifreleme Algoritması İle Elde Edilen Test Verileri (devamı)

| Adet | T | S | B | Tverim | Sverim | Bverim |
|---|---|---|---|---|---|---|
| 400 | 10 | 1 | 47 | 0,025 | 0,0025 | 0,1175 |
| 500 | 10 | 1 | 47 | 0,02 | 0,002 | 0,094 |
| 4000 | 10 | 1 | 52 | 0,0025 | 0,00025 | 0,013 |
| 5000 | 10 | 1 | 53 | 0,002 | 0,0002 | 0,0106 |
| 6000 | 10 | 1 | 55 | 0,001666667 | 0,000166667 | 0,009166667 |
| 7000 | 10 | 1 | 56 | 0,001428571 | 0,000142857 | 0,008 |
| 8000 | 10 | 1 | 57 | 0,00125 | 0,000125 | 0,007125 |

Çizelge B. 17 TripleDesSha256 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 15-TripleDesSha256 | | | | | | |
|---|---|---|---|---|---|---|
| Adet | T | S | B | Tverim | Sverim | Bverim |
| 100 | 10 | 1 | 48 | 0,1 | 0,01 | 0,48 |
| 200 | 10 | 1 | 48 | 0,05 | 0,005 | 0,24 |
| 300 | 10 | 1 | 48 | 0,033333333 | 0,003333333 | 0,16 |
| 400 | 10 | 1 | 48 | 0,025 | 0,0025 | 0,12 |
| 500 | 10 | 1 | 48 | 0,02 | 0,002 | 0,096 |
| 4000 | 10 | 1 | 53 | 0,0025 | 0,00025 | 0,01325 |
| 5000 | 10 | 1 | 54 | 0,002 | 0,0002 | 0,0108 |
| 6000 | 10 | 1 | 55 | 0,001666667 | 0,000166667 | 0,009166667 |
| 7000 | 10 | 1 | 57 | 0,001428571 | 0,000142857 | 0,008142857 |
| 8000 | 10 | 1 | 58 | 0,00125 | 0,000125 | 0,00725 |

Çizelge B. 18 TripleDesSha256Rsa15 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 16-TripleDesSha256Rsa15 | | | | | | |
|---|---|---|---|---|---|---|
| Adet | T | S | B | Tverim | Sverim | Bverim |
| 100 | 10 | 1 | 48 | 0,1 | 0,01 | 0,48 |
| 200 | 10 | 1 | 48 | 0,05 | 0,005 | 0,24 |
| 300 | 10 | 1 | 48 | 0,033333333 | 0,003333333 | 0,16 |
| 400 | 10 | 1 | 48 | 0,025 | 0,0025 | 0,12 |
| 500 | 10 | 1 | 48 | 0,02 | 0,002 | 0,096 |
| 4000 | 10 | 1 | 53 | 0,0025 | 0,00025 | 0,01325 |
| 5000 | 10 | 1 | 54 | 0,002 | 0,0002 | 0,0108 |
| 6000 | 10 | 1 | 55 | 0,001666667 | 0,000166667 | 0,009166667 |
| 7000 | 10 | 1 | 57 | 0,001428571 | 0,000142857 | 0,008142857 |
| 8000 | 10 | 1 | 58 | 0,00125 | 0,000125 | 0,00725 |

### B - 2. 2 WsHttpBinding Bağlama Kullanılarak Elde Edilen Veriler

Çizelge B. 19 Güvenliksiz İle Elde Edilen Test Verileri

| 16-TripleDesSha256Rsa15 | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 100 | 2 | 0,193 | 3 | 0,02 | 0,00193 | 0,03 |
| 200 | 2 | 0,191 | 3 | 0,01 | 0,000955 | 0,015 |
| 300 | 2 | 0,192 | 3 | 0,006666667 | 0,00064 | 0,01 |
| 400 | 2 | 0,193 | 4 | 0,005 | 0,0004825 | 0,01 |
| 500 | 2 | 0,196 | 4 | 0,004 | 0,000392 | 0,008 |
| 4000 | 2 | 0,195 | 7 | 0,0005 | 0,00004875 | 0,00175 |
| 5000 | 2 | 0,198 | 8 | 0,0004 | 0,0000396 | 0,0016 |
| 6000 | 2 | 0,209 | 9 | 0,000333333 | 3,48333E-05 | 0,0015 |
| 7000 | 2 | 0,196 | 10 | 0,000285714 | 0,000028 | 0,001428571 |
| 8000 | 2 | 0,197 | 11 | 0,00025 | 0,000024625 | 0,001375 |

Çizelge B.20 Basic128 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 1-Basic128 | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 100 | 10 | 1 | 47 | 0,1 | 0,01 | 0,47 |
| 200 | 10 | 1 | 47 | 0,05 | 0,005 | 0,235 |
| 300 | 10 | 1 | 47 | 0,033333333 | 0,003333333 | 0,156666667 |
| 400 | 10 | 1 | 47 | 0,025 | 0,0025 | 0,1175 |
| 500 | 10 | 1 | 47 | 0,02 | 0,002 | 0,094 |
| 4000 | 10 | 1 | 52 | 0,0025 | 0,00025 | 0,013 |
| 5000 | 10 | 1 | 53 | 0,002 | 0,0002 | 0,0106 |
| 6000 | 10 | 1 | 54 | 0,001666667 | 0,000166667 | 0,009 |
| 7000 | 10 | 1 | 56 | 0,001428571 | 0,000142857 | 0,008 |
| 8000 | 10 | 1 | 57 | 0,00125 | 0,000125 | 0,007125 |

Çizelge B.21 Basic128Rsa15 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 2-Basic128Rsa15 | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 100 | 10 | 1 | 47 | 0,1 | 0,01 | 0,47 |
| 200 | 10 | 1 | 47 | 0,05 | 0,005 | 0,235 |
| 300 | 10 | 1 | 47 | 0,033333333 | 0,003333333 | 0,156666667 |
| 400 | 10 | 1 | 47 | 0,025 | 0,0025 | 0,1175 |
| 500 | 10 | 1 | 47 | 0,02 | 0,002 | 0,094 |
| 4000 | 10 | 1 | 52 | 0,0025 | 0,00025 | 0,013 |
| 5000 | 10 | 1 | 53 | 0,002 | 0,0002 | 0,0106 |
| 6000 | 10 | 1 | 54 | 0,001666667 | 0,000166667 | 0,009 |
| 7000 | 10 | 1 | 56 | 0,001428571 | 0,000142857 | 0,008 |
| 8000 | 10 | 1 | 57 | 0,00125 | 0,000125 | 0,007125 |

Çizelge B.22 Basic128Sha256 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 3-Basic128Sha256 | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 100 | 10 | 1 | 47 | 0,1 | 0,01 | 0,47 |
| 200 | 10 | 1 | 48 | 0,05 | 0,005 | 0,24 |
| 300 | 10 | 1 | 48 | 0,033333333 | 0,003333333 | 0,16 |
| 400 | 10 | 1 | 48 | 0,025 | 0,0025 | 0,12 |
| 500 | 10 | 1 | 48 | 0,02 | 0,002 | 0,096 |
| 4000 | 10 | 1 | 53 | 0,0025 | 0,00025 | 0,01325 |
| 5000 | 10 | 1 | 54 | 0,002 | 0,0002 | 0,0108 |
| 6000 | 10 | 1 | 55 | 0,001666667 | 0,000166667 | 0,009166667 |
| 7000 | 10 | 1 | 56 | 0,001428571 | 0,000142857 | 0,008 |
| 8000 | 10 | 1 | 58 | 0,00125 | 0,000125 | 0,00725 |

Çizelge B.23 Basic128Sha256Rsa15 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 4-Basic128Sha256Rsa15 | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 100 | 10 | 1 | 47 | 0,1 | 0,01 | 0,47 |
| 200 | 10 | 1 | 48 | 0,05 | 0,005 | 0,24 |
| 300 | 10 | 1 | 48 | 0,033333333 | 0,003333333 | 0,16 |
| 400 | 10 | 1 | 48 | 0,025 | 0,0025 | 0,12 |
| 500 | 10 | 1 | 48 | 0,02 | 0,002 | 0,096 |
| 4000 | 10 | 1 | 53 | 0,0025 | 0,00025 | 0,01325 |
| 5000 | 10 | 1 | 54 | 0,002 | 0,0002 | 0,0108 |
| 6000 | 10 | 1 | 55 | 0,001666667 | 0,000166667 | 0,009166667 |
| 7000 | 10 | 1 | 56 | 0,001428571 | 0,000142857 | 0,008 |
| 8000 | 10 | 1 | 58 | 0,00125 | 0,000125 | 0,00725 |

Çizelge B.24 Basic192 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 5-Basic192 | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 100 | 10 | 1 | 47 | 0,1 | 0,01 | 0,47 |
| 200 | 10 | 1 | 47 | 0,05 | 0,005 | 0,235 |
| 300 | 10 | 1 | 47 | 0,033333333 | 0,003333333 | 0,156666667 |
| 400 | 10 | 1 | 47 | 0,025 | 0,0025 | 0,1175 |
| 500 | 10 | 1 | 47 | 0,02 | 0,002 | 0,094 |
| 4000 | 10 | 1 | 52 | 0,0025 | 0,00025 | 0,013 |
| 5000 | 10 | 1 | 53 | 0,002 | 0,0002 | 0,0106 |
| 6000 | 10 | 1 | 54 | 0,001666667 | 0,000166667 | 0,009 |
| 7000 | 10 | 1 | 56 | 0,001428571 | 0,000142857 | 0,008 |
| 8000 | 10 | 1 | 57 | 0,00125 | 0,000125 | 0,007125 |

Çizelge B.25 Basic192Rsa15 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 6-Basic192Rsa15 | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 100 | 10 | 1 | 47 | 0,1 | 0,01 | 0,47 |
| 200 | 10 | 1 | 47 | 0,05 | 0,005 | 0,235 |
| 300 | 10 | 1 | 47 | 0,033333333 | 0,003333333 | 0,156666667 |
| 400 | 10 | 1 | 47 | 0,025 | 0,0025 | 0,1175 |
| 500 | 10 | 1 | 47 | 0,02 | 0,002 | 0,094 |
| 4000 | 10 | 1 | 52 | 0,0025 | 0,00025 | 0,013 |
| 5000 | 10 | 1 | 53 | 0,002 | 0,0002 | 0,0106 |
| 6000 | 10 | 1 | 54 | 0,001666667 | 0,000166667 | 0,009 |
| 7000 | 10 | 1 | 56 | 0,001428571 | 0,000142857 | 0,008 |
| 8000 | 10 | 1 | 57 | 0,00125 | 0,000125 | 0,007125 |

Çizelge B.26 Basic192Sha256 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 7-Basic192Sha256 | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 100 | 10 | 1 | 48 | 0,1 | 0,01 | 0,48 |
| 200 | 10 | 1 | 48 | 0,05 | 0,005 | 0,24 |
| 300 | 10 | 1 | 48 | 0,033333333 | 0,003333333 | 0,16 |
| 400 | 10 | 1 | 48 | 0,025 | 0,0025 | 0,12 |
| 500 | 10 | 1 | 48 | 0,02 | 0,002 | 0,096 |
| 4000 | 10 | 1 | 53 | 0,0025 | 0,00025 | 0,01325 |
| 5000 | 10 | 1 | 54 | 0,002 | 0,0002 | 0,0108 |
| 6000 | 10 | 1 | 55 | 0,001666667 | 0,000166667 | 0,009166667 |
| 7000 | 10 | 1 | 57 | 0,001428571 | 0,000142857 | 0,008142857 |
| 8000 | 10 | 1 | 58 | 0,00125 | 0,000125 | 0,00725 |

Çizelge B.27 Basic192Sha256Rsa15 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 8-Basic192Sha256Rsa15 | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 100 | 10 | 1 | 48 | 0,1 | 0,01 | 0,48 |
| 200 | 10 | 1 | 48 | 0,05 | 0,005 | 0,24 |
| 300 | 10 | 1 | 48 | 0,033333333 | 0,003333333 | 0,16 |
| 400 | 10 | 1 | 48 | 0,025 | 0,0025 | 0,12 |
| 500 | 10 | 1 | 48 | 0,02 | 0,002 | 0,096 |
| 4000 | 10 | 1 | 53 | 0,0025 | 0,00025 | 0,01325 |
| 5000 | 10 | 1 | 54 | 0,002 | 0,0002 | 0,0108 |
| 6000 | 10 | 1 | 55 | 0,001666667 | 0,000166667 | 0,009166667 |
| 7000 | 10 | 1 | 57 | 0,001428571 | 0,000142857 | 0,008142857 |
| 8000 | 10 | 1 | 58 | 0,00125 | 0,000125 | 0,00725 |

Çizelge B.28 Basic256 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 9-Basic256 | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 100 | 10 | 1 | 47 | 0,1 | 0,01 | 0,47 |
| 200 | 10 | 1 | 47 | 0,05 | 0,005 | 0,235 |
| 300 | 10 | 1 | 47 | 0,033333333 | 0,003333333 | 0,156666667 |
| 400 | 10 | 1 | 47 | 0,025 | 0,0025 | 0,1175 |
| 500 | 10 | 1 | 47 | 0,02 | 0,002 | 0,094 |
| 4000 | 10 | 1 | 52 | 0,0025 | 0,00025 | 0,013 |
| 5000 | 10 | 1 | 53 | 0,002 | 0,0002 | 0,0106 |
| 6000 | 10 | 1 | 54 | 0,001666667 | 0,000166667 | 0,009 |
| 7000 | 10 | 1 | 56 | 0,001428571 | 0,000142857 | 0,008 |
| 8000 | 10 | 1 | 57 | 0,00125 | 0,000125 | 0,007125 |

Çizelge B.29 Basic256Rsa15 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 10-Basic256Rsa15 | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 100 | 10 | 1 | 47 | 0,1 | 0,01 | 0,47 |
| 200 | 10 | 1 | 47 | 0,05 | 0,005 | 0,235 |
| 300 | 10 | 1 | 47 | 0,033333333 | 0,003333333 | 0,156666667 |
| 400 | 10 | 1 | 47 | 0,025 | 0,0025 | 0,1175 |
| 500 | 10 | 1 | 47 | 0,02 | 0,002 | 0,094 |
| 4000 | 10 | 1 | 52 | 0,0025 | 0,00025 | 0,013 |
| 5000 | 10 | 1 | 53 | 0,002 | 0,0002 | 0,0106 |
| 6000 | 10 | 1 | 54 | 0,001666667 | 0,000166667 | 0,009 |
| 7000 | 10 | 1 | 56 | 0,001428571 | 0,000142857 | 0,008 |
| 8000 | 10 | 1 | 57 | 0,00125 | 0,000125 | 0,007125 |

Çizelge B.30 Basic256Sha256 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 11-Basic256Sha256 | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 100 | 10 | 1 | 47 | 0,1 | 0,01 | 0,47 |
| 200 | 10 | 1 | 48 | 0,05 | 0,005 | 0,24 |
| 300 | 10 | 1 | 48 | 0,033333333 | 0,003333333 | 0,16 |
| 400 | 10 | 1 | 48 | 0,025 | 0,0025 | 0,12 |
| 500 | 10 | 1 | 48 | 0,02 | 0,002 | 0,096 |
| 4000 | 10 | 1 | 52 | 0,0025 | 0,00025 | 0,013 |
| 5000 | 10 | 1 | 54 | 0,002 | 0,0002 | 0,0108 |
| 6000 | 10 | 1 | 55 | 0,001666667 | 0,000166667 | 0,009166667 |
| 7000 | 10 | 1 | 56 | 0,001428571 | 0,000142857 | 0,008 |
| 8000 | 10 | 1 | 58 | 0,00125 | 0,000125 | 0,00725 |

Çizelge B.31 Basic256Sha256Rsa15 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 12-Basic256Sha256Rsa15 | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 100 | 10 | 1 | 47 | 0,1 | 0,01 | 0,47 |
| 200 | 10 | 1 | 48 | 0,05 | 0,005 | 0,24 |
| 300 | 10 | 1 | 48 | 0,033333333 | 0,003333333 | 0,16 |
| 400 | 10 | 1 | 48 | 0,025 | 0,0025 | 0,12 |
| 500 | 10 | 1 | 48 | 0,02 | 0,002 | 0,096 |
| 4000 | 10 | 1 | 52 | 0,0025 | 0,00025 | 0,013 |
| 5000 | 10 | 1 | 54 | 0,002 | 0,0002 | 0,0108 |
| 6000 | 10 | 1 | 55 | 0,001666667 | 0,000166667 | 0,009166667 |
| 7000 | 10 | 1 | 56 | 0,001428571 | 0,000142857 | 0,008 |
| 8000 | 10 | 1 | 58 | 0,00125 | 0,000125 | 0,00725 |

Çizelge B.32 TripleDes Şifreleme Algoritması İle Elde Edilen Test Verileri

| 13-TripleDes | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 100 | 10 | 1 | 47 | 0,1 | 0,01 | 0,47 |
| 200 | 10 | 1 | 47 | 0,05 | 0,005 | 0,235 |
| 300 | 10 | 1 | 47 | 0,033333 | 0,003333 | 0,156667 |
| 400 | 10 | 1 | 47 | 0,025 | 0,0025 | 0,1175 |
| 500 | 10 | 1 | 47 | 0,02 | 0,002 | 0,094 |
| 4000 | 10 | 1 | 52 | 0,0025 | 0,00025 | 0,013 |
| 5000 | 10 | 1 | 53 | 0,002 | 0,0002 | 0,0106 |
| 6000 | 10 | 1 | 54 | 0,001667 | 0,000167 | 0,009 |
| 7000 | 10 | 1 | 56 | 0,001429 | 0,000143 | 0,008 |
| 8000 | 10 | 1 | 57 | 0,00125 | 0,000125 | 0,007125 |

Çizelge B.33 TripleDesRsa15 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 14-TripleDesRsa15 | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 100 | 10 | 1 | 47 | 0,1 | 0,01 | 0,47 |
| 200 | 10 | 1 | 47 | 0,05 | 0,005 | 0,235 |
| 300 | 10 | 1 | 47 | 0,033333333 | 0,003333333 | 0,156666667 |
| 400 | 10 | 1 | 47 | 0,025 | 0,0025 | 0,1175 |
| 500 | 10 | 1 | 47 | 0,02 | 0,002 | 0,094 |
| 4000 | 10 | 1 | 52 | 0,0025 | 0,00025 | 0,013 |
| 5000 | 10 | 1 | 53 | 0,002 | 0,0002 | 0,0106 |
| 6000 | 10 | 1 | 54 | 0,001666667 | 0,000166667 | 0,009 |
| 7000 | 10 | 1 | 56 | 0,001428571 | 0,000142857 | 0,008 |
| 8000 | 10 | 1 | 57 | 0,00125 | 0,000125 | 0,007125 |

Çizelge B.34 TripleDesSha256 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 15-TripleDesSha256 | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 100 | 10 | 1 | 47 | 0,1 | 0,01 | 0,47 |
| 200 | 10 | 1 | 48 | 0,05 | 0,005 | 0,24 |
| 300 | 10 | 1 | 48 | 0,033333 | 0,003333 | 0,16 |
| 400 | 10 | 1 | 48 | 0,025 | 0,0025 | 0,12 |
| 500 | 10 | 1 | 48 | 0,02 | 0,002 | 0,096 |
| 4000 | 10 | 1 | 52 | 0,0025 | 0,00025 | 0,013 |
| 5000 | 10 | 1 | 54 | 0,002 | 0,0002 | 0,0108 |
| 6000 | 10 | 1 | 55 | 0,001667 | 0,000167 | 0,009167 |
| 7000 | 10 | 1 | 56 | 0,001429 | 0,000143 | 0,008 |
| 8000 | 10 | 1 | 58 | 0,00125 | 0,000125 | 0,00725 |

Çizelge B.35 TripleDesSha256Rsa15 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 16-TripleDesSha256Rsa15 | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 100 | 10 | 1 | 47 | 0,1 | 0,01 | 0,47 |
| 200 | 10 | 1 | 48 | 0,05 | 0,005 | 0,24 |
| 300 | 10 | 1 | 48 | 0,033333333 | 0,003333333 | 0,16 |
| 400 | 10 | 1 | 48 | 0,025 | 0,0025 | 0,12 |
| 500 | 10 | 1 | 48 | 0,02 | 0,002 | 0,096 |
| 4000 | 10 | 1 | 52 | 0,0025 | 0,00025 | 0,013 |
| 5000 | 10 | 1 | 54 | 0,002 | 0,0002 | 0,0108 |
| 6000 | 10 | 1 | 55 | 0,001666667 | 0,000166667 | 0,009166667 |
| 7000 | 10 | 1 | 56 | 0,001428571 | 0,000142857 | 0,008 |
| 8000 | 10 | 1 | 58 | 0,00125 | 0,000125 | 0,00725 |

**B - 3 Çok İstemcili Sabit Kelime Katarlı Test İşlemi Verileri**

**B - 3. 1 NetTcpBinding Bağlama Kullanılarak Elde Edilen Veriler**

Çizelge B.36 Güvenliksiz Elde Edilen Test Verileri

| 0-Güvenliksiz | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 6 | 12 | 0,301 | 18 | 2 | 0,050166667 | 3 |
| 7 | 14 | 0,394 | 21 | 2 | 0,056285714 | 3 |
| 8 | 16 | 0,304 | 24 | 2 | 0,038 | 3 |
| 9 | 18 | 0,338 | 27 | 2 | 0,037555556 | 3 |
| 10 | 20 | 0,307 | 30 | 2 | 0,0307 | 3 |
| 3500 | 7000 | 8 | 10403 | 2 | 0,002285714 | 2,972285714 |
| 3600 | 7200 | 8 | 10693 | 2 | 0,002222222 | 2,970277778 |
| 3700 | 7400 | 8 | 10997 | 2 | 0,002162162 | 2,972162162 |
| 3800 | 7600 | 8 | 11294 | 2 | 0,002105263 | 2,972105263 |
| 3900 | 7800 | 8 | 11592 | 2 | 0,002051282 | 2,972307692 |
| 11000 | 22000 | 24 | 32700 | 2 | 0,002181818 | 2,972727273 |
| 12000 | 24000 | 26 | 35674 | 2 | 0,002166667 | 2,972833333 |
| 13000 | 26000 | 28 | 38649 | 2 | 0,002153846 | 2,973 |
| 14000 | 28000 | 30 | 41610 | 2 | 0,002142857 | 2,972142857 |
| 15000 | 30000 | 34 | 44598 | 2 | 0,002266667 | 2,9732 |

Çizelge B.37 Basic128 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 1-Basic128 | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 6 | 60 | 1 | 279 | 10 | 0,166666667 | 46,5 |
| 7 | 70 | 1 | 326 | 10 | 0,142857143 | 46,57142857 |
| 8 | 80 | 1 | 372 | 10 | 0,125 | 46,5 |
| 9 | 90 | 1 | 418 | 10 | 0,111111111 | 46,44444444 |
| 10 | 100 | 1 | 465 | 10 | 0,1 | 46,5 |
| 3500 | 35000 | 84 | 162967 | 10 | 0,024 | 46,562 |
| 3600 | 36000 | 86 | 167630 | 10 | 0,023888889 | 46,56388889 |
| 3700 | 37000 | 89 | 172288 | 10 | 0,024054054 | 46,56432432 |
| 3800 | 38000 | 91 | 176908 | 10 | 0,023947368 | 46,55473684 |
| 3900 | 39000 | 93 | 181595 | 10 | 0,023846154 | 46,56282051 |
| 11000 | 110000 | 263 | 512355 | 10 | 0,023909091 | 46,57772727 |
| 12000 | 120000 | 289 | 558941 | 10 | 0,024083333 | 46,57841667 |
| 13000 | 130000 | 308 | 605516 | 10 | 0,023692308 | 46,57815385 |
| 14000 | 140000 | 333 | 652133 | 10 | 0,023785714 | 46,58092857 |
| 15000 | 150000 | 356 | 698245 | 10 | 0,023733333 | 46,54966667 |

Çizelge B.38 Basic128Rsa15 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 2-Basic128Rsa15 | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 6 | 60 | 1 | 279 | 10 | 0,166666667 | 46,5 |
| 7 | 70 | 1 | 326 | 10 | 0,142857143 | 46,57142857 |
| 8 | 80 | 1 | 372 | 10 | 0,125 | 46,5 |
| 9 | 90 | 1 | 418 | 10 | 0,111111111 | 46,44444444 |
| 10 | 100 | 1 | 465 | 10 | 0,1 | 46,5 |
| 3500 | 35000 | 84 | 162968 | 10 | 0,024 | 46,56228571 |
| 3600 | 36000 | 86 | 167526 | 10 | 0,023888889 | 46,535 |
| 3700 | 37000 | 89 | 172281 | 10 | 0,024054054 | 46,56243243 |
| 3800 | 38000 | 91 | 176947 | 10 | 0,023947368 | 46,565 |
| 3900 | 39000 | 94 | 181592 | 10 | 0,024102564 | 46,56205128 |
| 11000 | 110000 | 262 | 512344 | 10 | 0,023818182 | 46,57672727 |
| 12000 | 120000 | 285 | 558923 | 10 | 0,02375 | 46,57691667 |
| 13000 | 130000 | 311 | 605207 | 10 | 0,023923077 | 46,55438462 |
| 14000 | 140000 | 334 | 652001 | 10 | 0,023857143 | 46,5715 |
| 15000 | 150000 | 357 | 698748 | 10 | 0,0238 | 46,5832 |

Çizelge B.39 Basic128Sha256 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 3-Basic128Sha256 | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 6 | 60 | 1 | 284 | 10 | 0,166666667 | 47,33333333 |
| 7 | 70 | 1 | 331 | 10 | 0,142857143 | 47,28571429 |
| 8 | 80 | 1 | 379 | 10 | 0,125 | 47,375 |
| 9 | 90 | 1 | 426 | 10 | 0,111111111 | 47,33333333 |
| 10 | 100 | 1 | 473 | 10 | 0,1 | 47,3 |
| 3500 | 35000 | 91 | 165953 | 10 | 0,026 | 47,41514286 |
| 3600 | 36000 | 95 | 170698 | 10 | 0,026388889 | 47,41611111 |
| 3700 | 37000 | 97 | 175411 | 10 | 0,026216216 | 47,40837838 |
| 3800 | 38000 | 99 | 180175 | 10 | 0,026052632 | 47,41447368 |
| 3900 | 39000 | 103 | 184924 | 10 | 0,026410256 | 47,41641026 |
| 11000 | 110000 | 281 | 521800 | 10 | 0,025545455 | 47,43636364 |
| 12000 | 120000 | 306 | 569230 | 10 | 0,0255 | 47,43583333 |
| 13000 | 130000 | 331 | 616687 | 10 | 0,025461538 | 47,43746154 |
| 14000 | 140000 | 357 | 664141 | 10 | 0,0255 | 47,43864286 |
| 15000 | 150000 | 385 | 711601 | 10 | 0,025666667 | 47,44006667 |

Çizelge B.40 Basic128Sha256Rsa15 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 4-Basic128Sha256Rsa15 | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 6 | 60 | 1 | 284 | 10 | 0,166666667 | 47,33333333 |

Çizelge B.40 Basic128Sha256Rsa15 Şifreleme Algoritması İle Elde Edilen Test Verileri (devamı)

| 4-Basic128Sha256Rsa15 | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 7 | 70 | 1 | 333 | 10 | 0,142857143 | 63 |
| 8 | 80 | 1 | 379 | 10 | 0,125 | 47,375 |
| 9 | 90 | 1 | 426 | 10 | 0,111111111 | 47,33333333 |
| 10 | 100 | 1 | 473 | 10 | 0,1 | 47,3 |
| 3500 | 35000 | 89 | 165943 | 10 | 0,025428571 | 47,41228571 |
| 3600 | 36000 | 90 | 170697 | 10 | 0,025 | 47,41583333 |
| 3700 | 37000 | 93 | 175441 | 10 | 0,025135135 | 47,41648649 |
| 3800 | 38000 | 94 | 180181 | 10 | 0,024736842 | 47,41605263 |
| 3900 | 39000 | 96 | 184932 | 10 | 0,024615385 | 47,41846154 |
| 11000 | 110000 | 269 | 521736 | 10 | 0,024454545 | 47,43054545 |
| 12000 | 120000 | 292 | 569226 | 10 | 0,024333333 | 47,4355 |
| 13000 | 130000 | 322 | 616702 | 10 | 0,024769231 | 47,43861538 |
| 14000 | 140000 | 342 | 664139 | 10 | 0,024428571 | 47,4385 |
| 15000 | 150000 | 368 | 711597 | 10 | 0,024533333 | 47,4398 |

Çizelge B.41 Basic192 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 5-Basic192 | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 6 | 60 | 1 | 276 | 10 | 0,166666667 | 46 |
| 7 | 70 | 1 | 326 | 10 | 0,142857143 | 46,57142857 |
| 8 | 80 | 1 | 372 | 10 | 0,125 | 46,5 |
| 9 | 90 | 1 | 419 | 10 | 0,111111111 | 46,55555556 |
| 10 | 100 | 1 | 465 | 10 | 0,1 | 46,5 |
| 3500 | 35000 | 85 | 163042 | 10 | 0,024285714 | 46,58342857 |
| 3600 | 36000 | 88 | 167701 | 10 | 0,024444444 | 46,58361111 |
| 3700 | 37000 | 90 | 172357 | 10 | 0,024324324 | 46,58297297 |
| 3800 | 38000 | 92 | 177015 | 10 | 0,024210526 | 46,58289474 |
| 3900 | 39000 | 95 | 181681 | 10 | 0,024358974 | 46,58487179 |
| 11000 | 110000 | 266 | 512489 | 10 | 0,024181818 | 46,58990909 |
| 12000 | 120000 | 290 | 559099 | 10 | 0,024166667 | 46,59158333 |
| 13000 | 130000 | 315 | 605719 | 10 | 0,024230769 | 46,59376923 |
| 14000 | 140000 | 338 | 652311 | 10 | 0,024142857 | 46,59364286 |
| 15000 | 150000 | 363 | 698953 | 10 | 0,0242 | 46,59686667 |

Çizelge B.42 Basic192Rsa15 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 6-Basic192Rsa15 | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 6 | 60 | 1 | 279 | 10 | 0,166666667 | 46,5 |
| 7 | 70 | 1 | 316 | 10 | 0,142857143 | 45,14285714 |
| 8 | 80 | 1 | 372 | 10 | 0,125 | 46,5 |
| 9 | 90 | 1 | 419 | 10 | 0,111111111 | 46,55555556 |
| 10 | 100 | 1 | 465 | 10 | 0,1 | 46,5 |
| 3500 | 35000 | 86 | 163037 | 10 | 0,024571429 | 46,582 |
| 3600 | 36000 | 88 | 167701 | 10 | 0,024444444 | 46,58361111 |
| 3700 | 37000 | 90 | 172350 | 10 | 0,024324324 | 46,58108108 |
| 3800 | 38000 | 93 | 177021 | 10 | 0,024473684 | 46,58447368 |
| 3900 | 39000 | 95 | 181675 | 10 | 0,024358974 | 46,58333333 |
| 11000 | 110000 | 267 | 512489 | 10 | 0,024272727 | 46,58990909 |
| 12000 | 120000 | 292 | 559101 | 10 | 0,024333333 | 46,59175 |
| 13000 | 130000 | 317 | 605724 | 10 | 0,024384615 | 46,59415385 |
| 14000 | 140000 | 340 | 652332 | 10 | 0,024285714 | 46,59514286 |
| 15000 | 150000 | 368 | 698803 | 10 | 0,024533333 | 46,58686667 |

Çizelge B.43 Basic192Sha256 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 7-Basic192Sha256 | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 6 | 60 | 1 | 284 | 10 | 0,166666667 | 47,33333333 |
| 7 | 70 | 1 | 332 | 10 | 0,142857143 | 47,42857143 |
| 8 | 80 | 1 | 379 | 10 | 0,125 | 47,375 |
| 9 | 90 | 1 | 426 | 10 | 0,111111111 | 47,33333333 |
| 10 | 100 | 2 | 474 | 10 | 0,2 | 47,4 |
| 3500 | 35000 | 90 | 166015 | 10 | 0,025714286 | 47,43285714 |
| 3600 | 36000 | 91 | 170768 | 10 | 0,025277778 | 47,43555556 |
| 3700 | 37000 | 93 | 175508 | 10 | 0,025135135 | 47,43459459 |
| 3800 | 38000 | 95 | 180254 | 10 | 0,025 | 47,43526316 |
| 3900 | 39000 | 98 | 184999 | 10 | 0,025128205 | 47,43564103 |
| 11000 | 110000 | 271 | 521946 | 10 | 0,024636364 | 47,44963636 |
| 12000 | 120000 | 296 | 569404 | 10 | 0,024666667 | 47,45033333 |
| 13000 | 130000 | 319 | 616848 | 10 | 0,024538462 | 47,44984615 |
| 14000 | 140000 | 345 | 664354 | 10 | 0,024642857 | 47,45385714 |
| 15000 | 150000 | 371 | 711685 | 10 | 0,024733333 | 47,44566667 |

Çizelge B.44 Basic192Sha256Rsa15 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 8-Basic192Sha256Rsa15 | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 6 | 60 | 1 | 284 | 10 | 0,166666667 | 47,33333333 |
| 7 | 70 | 1 | 332 | 10 | 0,142857143 | 47,42857143 |
| 8 | 80 | 1 | 379 | 10 | 0,125 | 47,375 |

Çizelge B.44 Basic192Sha256Rsa15 Şifreleme Algoritması İle Elde Edilen Test Verileri (devamı)

| 8-Basic192Sha256Rsa15 | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 9 | 90 | 1 | 426 | 10 | 0,111111111 | 47,33333333 |
| 10 | 100 | 2 | 474 | 10 | 0,2 | 47,4 |
| 3500 | 35000 | 87 | 166024 | 10 | 0,024857143 | 47,43542857 |
| 3600 | 36000 | 89 | 170768 | 10 | 0,024722222 | 47,43555556 |
| 3700 | 37000 | 93 | 175505 | 10 | 0,025135135 | 47,43378378 |
| 3800 | 38000 | 96 | 180259 | 10 | 0,025263158 | 47,43657895 |
| 3900 | 39000 | 98 | 185004 | 10 | 0,025128205 | 47,43692308 |
| 11000 | 110000 | 273 | 521924 | 10 | 0,024818182 | 47,44763636 |
| 12000 | 120000 | 298 | 569408 | 10 | 0,024833333 | 47,45066667 |
| 13000 | 130000 | 322 | 616861 | 10 | 0,024769231 | 47,45084615 |
| 14000 | 140000 | 348 | 664343 | 10 | 0,024857143 | 47,45307143 |
| 15000 | 150000 | 371 | 711825 | 10 | 0,024733333 | 47,455 |

Çizelge B.45 Basic256 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 9-Basic256 | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 6 | 60 | 1 | 278 | 10 | 0,166666667 | 46,33333333 |
| 7 | 70 | 1 | 325 | 10 | 0,142857143 | 46,42857143 |
| 8 | 80 | 1 | 371 | 10 | 0,125 | 46,375 |
| 9 | 90 | 1 | 417 | 10 | 0,111111111 | 46,33333333 |
| 10 | 100 | 1 | 464 | 10 | 0,1 | 46,4 |
| 3500 | 35000 | 86 | 162575 | 10 | 0,024571429 | 46,45 |
| 3600 | 36000 | 89 | 167230 | 10 | 0,024722222 | 46,45277778 |
| 3700 | 37000 | 91 | 171884 | 10 | 0,024594595 | 46,45513514 |
| 3800 | 38000 | 94 | 176520 | 10 | 0,024736842 | 46,45263158 |
| 3900 | 39000 | 96 | 181173 | 10 | 0,024615385 | 46,45461538 |
| 10000 | 100000 | 244 | 464620 | 10 | 0,0244 | 46,462 |
| 11000 | 110000 | 270 | 511084 | 10 | 0,024545455 | 46,46218182 |
| 12000 | 120000 | 293 | 557566 | 10 | 0,024416667 | 46,46383333 |
| 13000 | 130000 | 317 | 604037 | 10 | 0,024384615 | 46,46438462 |
| 14000 | 140000 | 341 | 650523 | 10 | 0,024357143 | 46,46592857 |
| 15000 | 150000 | 365 | 697040 | 10 | 0,024333333 | 46,46933333 |

Çizelge B.46 Basic256Rsa15 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 10-Basic256Rsa15 | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 6 | 60 | 1 | 278 | 10 | 0,166666667 | 46,33333333 |
| 7 | 70 | 1 | 325 | 10 | 0,142857143 | 46,42857143 |
| 8 | 80 | 1 | 371 | 10 | 0,125 | 46,375 |
| 9 | 90 | 1 | 417 | 10 | 0,111111111 | 46,33333333 |
| 10 | 100 | 1 | 464 | 10 | 0,1 | 46,4 |
| 3500 | 35000 | 86 | 162590 | 10 | 0,024571429 | 46,45428571 |
| 3600 | 36000 | 88 | 167237 | 10 | 0,024444444 | 46,45472222 |
| 3700 | 37000 | 91 | 171884 | 10 | 0,024594595 | 46,45513514 |
| 3800 | 38000 | 93 | 176531 | 10 | 0,024473684 | 46,45552632 |
| 3900 | 39000 | 95 | 181172 | 10 | 0,024358974 | 46,45435897 |
| 11000 | 110000 | 268 | 511093 | 10 | 0,024363636 | 46,463 |
| 12000 | 120000 | 292 | 557525 | 10 | 0,024333333 | 46,46041667 |
| 13000 | 130000 | 315 | 604025 | 10 | 0,024230769 | 46,46346154 |
| 14000 | 140000 | 340 | 650498 | 10 | 0,024285714 | 46,46414286 |
| 15000 | 150000 | 363 | 697015 | 10 | 0,0242 | 46,46766667 |

Çizelge B.47 Basic256Sha256 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 11-Basic256Sha256 | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 6 | 60 | 1 | 283 | 10 | 0,166666667 | 47,16666667 |
| 7 | 70 | 1 | 331 | 10 | 0,142857143 | 47,28571429 |
| 8 | 80 | 1 | 378 | 10 | 0,125 | 47,25 |
| 9 | 90 | 1 | 425 | 10 | 0,111111111 | 47,22222222 |
| 10 | 100 | 1 | 478 | 10 | 0,1 | 47,8 |
| 3500 | 35000 | 88 | 165567 | 10 | 0,025142857 | 47,30485714 |
| 3600 | 36000 | 90 | 170305 | 10 | 0,025 | 47,30694444 |
| 3700 | 37000 | 93 | 175024 | 10 | 0,025135135 | 47,30378378 |
| 3800 | 38000 | 95 | 179711 | 10 | 0,025 | 47,29236842 |
| 3900 | 39000 | 98 | 184500 | 10 | 0,025128205 | 47,30769231 |
| 11000 | 110000 | 274 | 520522 | 10 | 0,024909091 | 47,32018182 |
| 12000 | 120000 | 301 | 567867 | 10 | 0,025083333 | 47,32225 |
| 13000 | 130000 | 325 | 615181 | 10 | 0,025 | 47,32161538 |
| 14000 | 140000 | 352 | 662523 | 10 | 0,025142857 | 47,32307143 |
| 15000 | 150000 | 374 | 709915 | 10 | 0,024933333 | 47,32766667 |

Çizelge B.48 Basic256Sha256Rsa15 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 12-Basic256Sha256Rsa15 | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 6 | 60 | 1 | 283 | 10 | 0,166666667 | 47,16666667 |
| 7 | 70 | 1 | 331 | 10 | 0,142857143 | 47,28571429 |
| 8 | 80 | 1 | 378 | 10 | 0,125 | 47,25 |
| 9 | 90 | 1 | 425 | 10 | 0,111111111 | 47,22222222 |
| 10 | 100 | 1 | 472 | 10 | 0,1 | 47,2 |
| 3500 | 35000 | 87 | 165571 | 10 | 0,024857143 | 47,306 |
| 3600 | 36000 | 90 | 170305 | 10 | 0,025 | 47,30694444 |
| 3700 | 37000 | 93 | 175026 | 10 | 0,025135135 | 47,30432432 |
| 3800 | 38000 | 96 | 179767 | 10 | 0,025263158 | 47,30710526 |
| 3900 | 39000 | 97 | 184501 | 10 | 0,024871795 | 47,30794872 |
| 11000 | 110000 | 274 | 520540 | 10 | 0,024909091 | 47,32181818 |
| 12000 | 120000 | 298 | 567860 | 10 | 0,024833333 | 47,32166667 |
| 13000 | 130000 | 323 | 615191 | 10 | 0,024846154 | 47,32238462 |
| 14000 | 140000 | 348 | 662513 | 10 | 0,024857143 | 47,32235714 |
| 15000 | 150000 | 375 | 709905 | 10 | 0,025 | 47,327 |

Çizelge B.49 TripleDes Şifreleme Algoritması İle Elde Edilen Test Verileri

| 13-TripleDes | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 6 | 60 | 1 | 279 | 10 | 0,166666667 | 46,5 |
| 7 | 70 | 1 | 325 | 10 | 0,142857143 | 46,42857143 |
| 8 | 80 | 1 | 371 | 10 | 0,125 | 46,375 |
| 9 | 90 | 2 | 418 | 10 | 0,222222222 | 46,44444444 |
| 10 | 100 | 2 | 464 | 10 | 0,2 | 46,4 |
| 3500 | 35000 | 97 | 162634 | 10 | 0,027714286 | 46,46685714 |
| 3600 | 36000 | 103 | 167283 | 10 | 0,028611111 | 46,4675 |
| 3700 | 37000 | 105 | 171901 | 10 | 0,028378378 | 46,45972973 |
| 3800 | 38000 | 109 | 176574 | 10 | 0,028684211 | 46,46684211 |
| 3900 | 39000 | 108 | 181234 | 10 | 0,027692308 | 46,47025641 |
| 11000 | 110000 | 302 | 511266 | 10 | 0,027454545 | 46,47872727 |
| 12000 | 120000 | 329 | 557651 | 10 | 0,027416667 | 46,47091667 |
| 13000 | 130000 | 356 | 604270 | 10 | 0,027384615 | 46,48230769 |
| 14000 | 140000 | 385 | 650657 | 10 | 0,0275 | 46,4755 |
| 15000 | 150000 | 412 | 697336 | 10 | 0,027466667 | 46,48906667 |

Çizelge B.50 TripleDesRsa15 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 14-TripleDesRsa15 | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 6 | 60 | 1 | 279 | 10 | 0,166666667 | 46,5 |
| 7 | 70 | 1 | 325 | 10 | 0,142857143 | 46,42857143 |

Çizelge B.50 TripleDesRsa15 Şifreleme Algoritması İle Elde Edilen Test Verileri (devam)

| Adet | T | S | B | Tverim | Sverim | Bverim |
|---|---|---|---|---|---|---|
| 8 | 80 | 1 | 371 | 10 | 0,125 | 46,375 |
| 9 | 90 | 2 | 418 | 10 | 0,222222222 | 46,44444444 |
| 10 | 100 | 2 | 464 | 10 | 0,2 | 46,4 |
| 3500 | 35000 | 97 | 162641 | 10 | 0,027714286 | 46,46885714 |
| 3600 | 36000 | 99 | 167289 | 10 | 0,0275 | 46,46916667 |
| 3700 | 37000 | 102 | 171935 | 10 | 0,027567568 | 46,46891892 |
| 3800 | 38000 | 106 | 176591 | 10 | 0,027894737 | 46,47131579 |
| 3900 | 39000 | 108 | 181299 | 10 | 0,027692308 | 46,48692308 |
| 11000 | 110000 | 303 | 511268 | 10 | 0,027545455 | 46,47890909 |
| 12000 | 120000 | 329 | 557751 | 10 | 0,027416667 | 46,47925 |
| 13000 | 130000 | 358 | 604263 | 10 | 0,027538462 | 46,48176923 |
| 14000 | 140000 | 385 | 650775 | 10 | 0,0275 | 46,48392857 |
| 15000 | 150000 | 410 | 697312 | 10 | 0,027333333 | 46,48746667 |

Çizelge B.51 TripleDesSha256 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 15-TripleDesSha256 | | | | | | |
|---|---|---|---|---|---|---|
| Adet | T | S | B | Tverim | Sverim | Bverim |
| 6 | 60 | 1 | 284 | 10 | 0,166666667 | 47,33333333 |
| 7 | 70 | 1 | 331 | 10 | 0,142857143 | 47,28571429 |
| 8 | 80 | 1 | 378 | 10 | 0,125 | 47,25 |
| 9 | 90 | 2 | 425 | 10 | 0,222222222 | 47,22222222 |
| 10 | 100 | 2 | 472 | 10 | 0,2 | 47,2 |
| 3600 | 36000 | 102 | 170308 | 10 | 0,028333333 | 47,30777778 |
| 3700 | 37000 | 105 | 174999 | 10 | 0,028378378 | 47,29702703 |
| 3800 | 38000 | 107 | 179739 | 10 | 0,028157895 | 47,29973684 |
| 3900 | 39000 | 110 | 184508 | 10 | 0,028205128 | 47,30974359 |
| 10000 | 100000 | 280 | 473192 | 10 | 0,028 | 47,3192 |
| 11000 | 110000 | 309 | 520529 | 10 | 0,028090909 | 47,32081818 |
| 12000 | 120000 | 338 | 567836 | 10 | 0,028166667 | 47,31966667 |
| 13000 | 130000 | 365 | 615170 | 10 | 0,028076923 | 47,32076923 |
| 14000 | 140000 | 394 | 662510 | 10 | 0,028142857 | 47,32214286 |
| 15000 | 150000 | 421 | 709882 | 10 | 0,028066667 | 47,32546667 |

Çizelge B.52 TripleDesSha256Rsa15 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 16-TripleDesSha256Rsa15 | | | | | | |
|---|---|---|---|---|---|---|
| Adet | T | S | B | Tverim | Sverim | Bverim |
| 6 | 60 | 1 | 284 | 10 | 0,166667 | 47,33333 |
| 7 | 70 | 1 | 331 | 10 | 0,142857 | 47,28571 |
| 8 | 80 | 1 | 378 | 10 | 0,125 | 47,25 |
| 9 | 90 | 2 | 425 | 10 | 0,222222 | 47,22222 |
| 10 | 100 | 2 | 472 | 10 | 0,2 | 47,2 |

Çizelge B. 52 TripleDesSha256Rsa15 Şifreleme Algoritması İle Elde Edilen Test Verileri (devamı)

| 16-TripleDesSha256Rsa15 | | | | | | |
|---|---|---|---|---|---|---|
| Adet | T | S | B | Tverim | Sverim | Bverim |
| 3500 | 35000 | 99 | 165569 | 10 | 0,028286 | 47,30543 |
| 3600 | 36000 | 102 | 175043 | 10 | 0,028333 | 48,62306 |
| 3700 | 37000 | 105 | 179775 | 10 | 0,028378 | 48,58784 |
| 3800 | 38000 | 108 | 179775 | 10 | 0,028421 | 47,30921 |
| 3900 | 39000 | 111 | 184500 | 10 | 0,028462 | 47,30769 |
| 11000 | 110000 | 311 | 520543 | 10 | 0,028273 | 47,32209 |
| 12000 | 120000 | 337 | 567875 | 10 | 0,028083 | 47,32292 |
| 13000 | 130000 | 365 | 615710 | 10 | 0,028077 | 47,36231 |
| 14000 | 140000 | 396 | 662518 | 10 | 0,028286 | 47,32271 |
| 15000 | 150000 | 422 | 709905 | 10 | 0,028133 | 47,327 |

### B - 3. 2 WsHttpBinding Bağlama Kullanılarak Elde Edilen Veriler

Çizelge B. 53 Güvenliksiz Elde Edilen Test Verileri

| 0-Güvenliksiz | | | | | | |
|---|---|---|---|---|---|---|
| Adet | T | S | B | Tverim | Sverim | Bverim |
| 6 | 12 | 0,294 | 17 | 2 | 0,049 | 2,833333333 |
| 7 | 14 | 0,299 | 19 | 2 | 0,042714286 | 2,714285714 |
| 8 | 16 | 0,358 | 22 | 2 | 0,04475 | 2,75 |
| 9 | 18 | 0,302 | 25 | 2 | 0,033555556 | 2,777777778 |
| 10 | 20 | 0,306 | 27 | 2 | 0,0306 | 2,7 |
| 3500 | 7000 | 8 | 9418 | 2 | 0,002285714 | 0,002690857 |
| 3600 | 7200 | 8 | 9687 | 2 | 0,002222222 | 0,002690833 |
| 3700 | 7400 | 8 | 9957 | 2 | 0,002162162 | 0,002691081 |
| 3800 | 7600 | 9 | 10226 | 2 | 0,002368421 | 0,002691053 |
| 3900 | 7800 | 9 | 10495 | 2 | 0,002307692 | 0,002691026 |
| 11000 | 22000 | 24 | 29605 | 2 | 0,002181818 | 0,002691364 |
| 12000 | 24000 | 27 | 32299 | 2 | 0,00225 | 0,002691583 |
| 13000 | 26000 | 29 | 34992 | 2 | 0,002230769 | 0,002691692 |
| 14000 | 28000 | 31 | 37682 | 2 | 0,002214286 | 0,002691571 |
| 15000 | 30000 | 34 | 40379 | 2 | 0,002266667 | 0,002691933 |

Çizelge B. 54 Basic128 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 1-Basic128 | | | | | | |
|---|---|---|---|---|---|---|
| Adet | T | S | B | Tverim | Sverim | Bverim |
| 6 | 60 | 1 | 277 | 10 | 0,166666667 | 46,16666667 |
| 7 | 70 | 1 | 323 | 10 | 0,142857143 | 46,14285714 |
| 8 | 80 | 1 | 370 | 10 | 0,125 | 46,25 |
| 9 | 90 | 1 | 416 | 10 | 0,111111111 | 46,22222222 |
| 10 | 100 | 1 | 462 | 10 | 0,1 | 46,2 |
| 3500 | 35000 | 89 | 161934 | 10 | 0,025428571 | 46,26685714 |

Çizelge B. 54 Basic128 Şifreleme Algoritması İle Elde Edilen Test Verileri (devam)

| | | | | | | |
|---|---|---|---|---|---|---|
| 3600 | 36000 | 92 | 166561 | 10 | 0,025555556 | 46,26694444 |
| 3700 | 37000 | 94 | 171182 | 10 | 0,025405405 | 46,26540541 |
| 3800 | 38000 | 96 | 175811 | 10 | 0,025263158 | 46,26605263 |
| 3900 | 39000 | 98 | 180411 | 10 | 0,025128205 | 46,25923077 |
| 11000 | 110000 | 278 | 509026 | 10 | 0,025272727 | 46,27509091 |
| 12000 | 120000 | 303 | 555301 | 10 | 0,02525 | 46,27508333 |
| 13000 | 130000 | 328 | 601600 | 10 | 0,025230769 | 46,27692308 |
| 14000 | 140000 | 354 | 647900 | 10 | 0,025285714 | 46,27857143 |
| 15000 | 150000 | 381 | 694204 | 10 | 0,0254 | 46,28026667 |

Çizelge B. 55 Basic128Rsa15 Şifreleme Algoritması İle Elde Edilen Test Verileri

| **2-Basic128Rsa15** | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 6 | 60 | 1 | 277 | 10 | 0,166666667 | 46,16666667 |
| 7 | 70 | 1 | 324 | 10 | 0,142857143 | 46,28571429 |
| 8 | 80 | 1 | 370 | 10 | 0,125 | 46,25 |
| 9 | 90 | 1 | 416 | 10 | 0,111111111 | 46,22222222 |
| 10 | 100 | 1 | 462 | 10 | 0,1 | 46,2 |
| 3500 | 35000 | 89 | 161927 | 10 | 0,025428571 | 46,26485714 |
| 3600 | 36000 | 92 | 171188 | 10 | 0,025555556 | 47,55222222 |
| 3700 | 37000 | 95 | 175813 | 10 | 0,025675676 | 47,51702703 |
| 3800 | 38000 | 97 | 175813 | 10 | 0,025526316 | 46,26657895 |
| 3900 | 39000 | 99 | 180439 | 10 | 0,025384615 | 46,26641026 |
| 11000 | 110000 | 277 | 509029 | 10 | 0,025181818 | 46,27536364 |
| 12000 | 120000 | 304 | 555294 | 10 | 0,025333333 | 46,2745 |
| 13000 | 130000 | 329 | 601603 | 10 | 0,025307692 | 46,27715385 |
| 14000 | 140000 | 358 | 647910 | 10 | 0,025571429 | 46,27928571 |
| 15000 | 150000 | 378 | 694214 | 10 | 0,0252 | 46,28093333 |

Çizelge B. 56 Basic128Sha256 Şifreleme Algoritması İle Elde Edilen Test Verileri

| **3-Basic128Sha256** | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 6 | 60 | 1 | 282 | 10 | 0,166666667 | 47 |
| 7 | 70 | 1 | 323 | 10 | 0,142857143 | 46,14285714 |
| 8 | 80 | 1 | 376 | 10 | 0,125 | 47 |
| 9 | 90 | 1 | 423 | 10 | 0,111111111 | 47 |
| 10 | 100 | 1 | 470 | 10 | 0,1 | 47 |
| 3500 | 35000 | 90 | 164641 | 10 | 0,025714286 | 47,04028571 |
| 3600 | 36000 | 92 | 169339 | 10 | 0,025555556 | 47,03861111 |
| 3700 | 37000 | 95 | 174043 | 10 | 0,025675676 | 47,03864865 |
| 3800 | 38000 | 98 | 174043 | 10 | 0,025789474 | 45,80078947 |
| 3900 | 39000 | 100 | 178755 | 10 | 0,025641026 | 45,83461538 |

Çizelge B. 56 Basic128Sha256 Şifreleme Algoritması İle Elde Edilen Test Verileri (devamı)

| 3-Basic128Sha256 | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 11000 | 110000 | 281 | 517531 | 10 | 0,025545455 | 47,04827273 |
| 12000 | 120000 | 306 | 564588 | 10 | 0,0255 | 47,049 |
| 13000 | 130000 | 332 | 611650 | 10 | 0,025538462 | 47,05 |
| 14000 | 140000 | 358 | 658706 | 10 | 0,025571429 | 47,05042857 |
| 15000 | 150000 | 383 | 705799 | 10 | 0,025533333 | 47,05326667 |

Çizelge B.57 Basic128Sha256Rsa15 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 4-Basic128Sha256Rsa15 | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 6 | 60 | 1 | 282 | 10 | 0,166666667 | 47 |
| 7 | 70 | 1 | 329 | 10 | 0,142857143 | 47 |
| 8 | 80 | 1 | 376 | 10 | 0,125 | 47 |
| 9 | 90 | 1 | 423 | 10 | 0,111111111 | 47 |
| 10 | 100 | 1 | 470 | 10 | 0,1 | 47 |
| 3500 | 35000 | 90 | 164633 | 10 | 0,025714286 | 47,038 |
| 3600 | 36000 | 93 | 169346 | 10 | 0,025833333 | 47,04055556 |
| 3700 | 37000 | 95 | 174050 | 10 | 0,025675676 | 47,04054054 |
| 3800 | 38000 | 98 | 178758 | 10 | 0,025789474 | 47,04157895 |
| 3900 | 39000 | 101 | 183459 | 10 | 0,025897436 | 47,04076923 |
| 11000 | 110000 | 283 | 517545 | 10 | 0,025727273 | 47,04954545 |
| 12000 | 120000 | 310 | 564571 | 10 | 0,025833333 | 47,04758333 |
| 13000 | 130000 | 333 | 611648 | 10 | 0,025615385 | 47,04984615 |
| 14000 | 140000 | 360 | 658708 | 10 | 0,025714286 | 47,05057143 |
| 15000 | 150000 | 383 | 705799 | 10 | 0,025533333 | 47,05326667 |

Çizelge B. 58 Basic192 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 5-Basic192 | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 6 | 60 | 1 | 277 | 10 | 0,166667 | 46,16667 |
| 7 | 70 | 1 | 324 | 10 | 0,142857 | 46,28571 |
| 8 | 80 | 1 | 370 | 10 | 0,125 | 46,25 |
| 9 | 90 | 1 | 416 | 10 | 0,111111 | 46,22222 |
| 10 | 100 | 1 | 462 | 10 | 0,1 | 46,2 |
| 3500 | 35000 | 90 | 162041 | 10 | 0,025714 | 46,29743 |
| 3600 | 36000 | 93 | 166672 | 10 | 0,025833 | 46,29778 |
| 3700 | 37000 | 95 | 171294 | 10 | 0,025676 | 46,29568 |
| 3800 | 38000 | 97 | 175897 | 10 | 0,025526 | 46,28868 |
| 3900 | 39000 | 100 | 180557 | 10 | 0,025641 | 46,29667 |
| 11000 | 110000 | 282 | 509376 | 10 | 0,025636 | 46,30691 |

Çizelge B. 58 Basic192 Şifreleme Algoritması İle Elde Edilen Test Verileri (devam)

| Adet | T | S | B | Tverim | Sverim | Bverim |
|---|---|---|---|---|---|---|
| 12000 | 120000 | 308 | 555603 | 10 | 0,025667 | 46,30025 |
| 13000 | 130000 | 333 | 602009 | 10 | 0,025615 | 46,30838 |
| 14000 | 140000 | 359 | 648333 | 10 | 0,025643 | 46,3095 |
| 15000 | 150000 | 384 | 694693 | 10 | 0,0256 | 46,31287 |

Çizelge B. 59 Basic192Rsa15 Şifreleme Algoritması İle Elde Edilen Test Verileri (devamı)

| 6-Basic192Rsa15 | | | | | | |
|---|---|---|---|---|---|---|
| Adet | T | S | B | Tverim | Sverim | Bverim |
| 6 | 60 | 1 | 277 | 10 | 0,166667 | 46,16667 |
| 7 | 70 | 1 | 324 | 10 | 0,142857 | 46,28571 |
| 8 | 80 | 1 | 370 | 10 | 0,125 | 46,25 |
| 9 | 90 | 1 | 416 | 10 | 0,111111 | 46,22222 |
| 10 | 100 | 1 | 462 | 10 | 0,1 | 46,2 |
| 3500 | 35000 | 89 | 162037 | 10 | 0,025429 | 46,29629 |
| 3600 | 36000 | 92 | 166672 | 10 | 0,025556 | 46,29778 |
| 3700 | 37000 | 94 | 171303 | 10 | 0,025405 | 46,29811 |
| 3800 | 38000 | 96 | 175935 | 10 | 0,025263 | 46,29868 |
| 3900 | 39000 | 99 | 180556 | 10 | 0,025385 | 46,29641 |
| 11000 | 110000 | 280 | 509310 | 10 | 0,025455 | 46,30091 |
| 12000 | 120000 | 304 | 555669 | 10 | 0,025333 | 46,30575 |
| 13000 | 130000 | 330 | 602001 | 10 | 0,025385 | 46,30777 |
| 14000 | 140000 | 356 | 648341 | 10 | 0,025429 | 46,31007 |
| 15000 | 150000 | 382 | 694668 | 10 | 0,025467 | 46,3112 |

Çizelge B. 60 Basic192Sha256 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 7-Basic192Sha256 | | | | | | |
|---|---|---|---|---|---|---|
| Adet | T | S | B | Tverim | Sverim | Bverim |
| 6 | 60 | 1 | 282 | 10 | 0,166666667 | 47 |
| 7 | 70 | 1 | 329 | 10 | 0,142857143 | 47 |
| 8 | 80 | 1 | 376 | 10 | 0,125 | 47 |
| 9 | 90 | 1 | 423 | 10 | 0,111111111 | 47 |
| 10 | 100 | 1 | 470 | 10 | 0,1 | 47 |
| 3500 | 35000 | 92 | 164744 | 10 | 0,026285714 | 47,06971429 |
| 3600 | 36000 | 95 | 169455 | 10 | 0,026388889 | 47,07083333 |
| 3700 | 37000 | 97 | 174155 | 10 | 0,026216216 | 47,06891892 |
| 3800 | 38000 | 99 | 178867 | 10 | 0,026052632 | 47,07026316 |
| 3900 | 39000 | 102 | 183576 | 10 | 0,026153846 | 47,07076923 |
| 11000 | 110000 | 287 | 517883 | 10 | 0,026090909 | 47,08027273 |
| 12000 | 120000 | 314 | 564964 | 10 | 0,026166667 | 47,08033333 |
| 13000 | 130000 | 340 | 612039 | 10 | 0,026153846 | 47,07992308 |

Çizelge B. 60 Basic192Sha256 Şifreleme Algoritması İle Elde Edilen Test Verileri (devamı)

| 7-Basic192Sha256 | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 14000 | 140000 | 365 | 659158 | 10 | 0,026071429 | 47,08271429 |
| 15000 | 150000 | 392 | 706299 | 10 | 0,026133333 | 47,0866 |

Çizelge B. 61 Basic192Sha256Rsa15 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 8-Basic192Sha256Rsa15 | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 6 | 60 | 1 | 282 | 10 | 0,166666667 | 47 |
| 7 | 70 | 1 | 329 | 10 | 0,142857143 | 47 |
| 8 | 80 | 1 | 376 | 10 | 0,125 | 47 |
| 9 | 90 | 1 | 423 | 10 | 0,111111111 | 47 |
| 10 | 100 | 2 | 470 | 10 | 0,2 | 47 |
| 3500 | 35000 | 92 | 164742 | 10 | 0,026285714 | 47,06914286 |
| 3600 | 36000 | 94 | 169451 | 10 | 0,026111111 | 47,06972222 |
| 3700 | 37000 | 97 | 174162 | 10 | 0,026216216 | 47,07081081 |
| 3800 | 38000 | 99 | 178866 | 10 | 0,026052632 | 47,07 |
| 3900 | 39000 | 102 | 183573 | 10 | 0,026153846 | 47,07 |
| 11000 | 110000 | 288 | 517868 | 10 | 0,026181818 | 47,07890909 |
| 12000 | 120000 | 314 | 564950 | 10 | 0,026166667 | 47,07916667 |
| 13000 | 130000 | 340 | 612054 | 10 | 0,026153846 | 47,08107692 |
| 14000 | 140000 | 366 | 659149 | 10 | 0,026142857 | 47,08207143 |
| 15000 | 150000 | 397 | 706303 | 10 | 0,026466667 | 47,08686667 |

Çizelge B. 62 Basic256 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 9-Basic256 | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 6 | 60 | 1 | 277 | 10 | 0,166666667 | 46,16666667 |
| 7 | 70 | 1 | 323 | 10 | 0,142857143 | 46,14285714 |
| 8 | 80 | 1 | 369 | 10 | 0,125 | 46,125 |
| 9 | 90 | 1 | 415 | 10 | 0,111111111 | 46,11111111 |
| 10 | 100 | 1 | 461 | 10 | 0,1 | 46,1 |
| 3500 | 35000 | 90 | 161524 | 10 | 0,025714286 | 46,14971429 |
| 3600 | 36000 | 93 | 166130 | 10 | 0,025833333 | 46,14722222 |
| 3700 | 37000 | 95 | 170752 | 10 | 0,025675676 | 46,14918919 |
| 3800 | 38000 | 98 | 175368 | 10 | 0,025789474 | 46,14947368 |
| 3900 | 39000 | 101 | 179941 | 10 | 0,025897436 | 46,13871795 |
| 12000 | 120000 | 309 | 553797 | 10 | 0,02575 | 46,14975 |
| 13000 | 130000 | 335 | 599957 | 10 | 0,025769231 | 46,15053846 |
| 14000 | 140000 | 361 | 651485 | 10 | 0,025785714 | 46,53464286 |
| 15000 | 150000 | 399 | 692485 | 10 | 0,0266 | 46,16566667 |

Çizelge B. 63 Basic256Rsa15 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 10-Basic256Rsa15 | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 6 | 60 | 1 | 277 | 10 | 0,166667 | 46,16667 |
| 7 | 70 | 1 | 323 | 10 | 0,142857 | 46,14286 |
| 8 | 80 | 1 | 369 | 10 | 0,125 | 46,125 |
| 9 | 90 | 1 | 415 | 10 | 0,111111 | 46,11111 |
| 10 | 100 | 1 | 461 | 10 | 0,1 | 46,1 |
| 3500 | 35000 | 91 | 161514 | 10 | 0,026 | 46,14686 |
| 3600 | 36000 | 93 | 166137 | 10 | 0,025833 | 46,14917 |
| 3700 | 37000 | 95 | 170748 | 10 | 0,025676 | 46,14811 |
| 3800 | 38000 | 98 | 175366 | 10 | 0,025789 | 46,14895 |
| 3900 | 39000 | 100 | 179990 | 10 | 0,025641 | 46,15128 |
| 11000 | 110000 | 283 | 507753 | 10 | 0,025727 | 46,15936 |
| 12000 | 120000 | 308 | 553885 | 10 | 0,025667 | 46,15708 |
| 13000 | 130000 | 333 | 600087 | 10 | 0,025615 | 46,16054 |
| 14000 | 140000 | 359 | 646292 | 10 | 0,025643 | 46,16371 |
| 15000 | 150000 | 387 | 692520 | 10 | 0,0258 | 46,168 |

Çizelge B. 64 Basic256Sha256 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 11-Basic256Sha256 | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 6 | 60 | 1 | 281 | 10 | 0,166666667 | 46,83333333 |
| 7 | 70 | 1 | 328 | 10 | 0,142857143 | 46,85714286 |
| 8 | 80 | 1 | 375 | 10 | 0,125 | 46,875 |
| 9 | 90 | 1 | 422 | 10 | 0,111111111 | 46,88888889 |
| 10 | 100 | 1 | 469 | 10 | 0,1 | 46,9 |
| 3500 | 35000 | 92 | 164229 | 10 | 0,026285714 | 46,92257143 |
| 3600 | 36000 | 95 | 168925 | 10 | 0,026388889 | 46,92361111 |
| 3700 | 37000 | 98 | 173619 | 10 | 0,026486486 | 46,92405405 |
| 3800 | 38000 | 101 | 178304 | 10 | 0,026578947 | 46,92210526 |
| 3900 | 39000 | 103 | 183001 | 10 | 0,026410256 | 46,92333333 |
| 11000 | 110000 | 288 | 516141 | 10 | 0,026181818 | 46,92190909 |
| 12000 | 120000 | 316 | 563186 | 10 | 0,026333333 | 46,93216667 |
| 13000 | 130000 | 342 | 610141 | 10 | 0,026307692 | 46,93392308 |
| 14000 | 140000 | 369 | 657117 | 10 | 0,026357143 | 46,93692857 |
| 15000 | 150000 | 395 | 704084 | 10 | 0,026333333 | 46,93893333 |

Çizelge B.65 Basic256Sha256Rsa15 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 12-Basic256Sha256Rsa15 | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 6 | 60 | 1 | 281 | 10 | 0,166666667 | 46,83333333 |
| 7 | 70 | 1 | 328 | 10 | 0,142857143 | 46,85714286 |
| 8 | 80 | 1 | 375 | 10 | 0,125 | 46,875 |
| 9 | 90 | 1 | 422 | 10 | 0,111111111 | 46,88888889 |
| 10 | 100 | 1 | 469 | 10 | 0,1 | 46,9 |
| 3500 | 35000 | 93 | 164191 | 10 | 0,026571429 | 46,91171429 |
| 3600 | 36000 | 96 | 168920 | 10 | 0,026666667 | 46,92222222 |
| 3700 | 37000 | 98 | 173612 | 10 | 0,026486486 | 46,92216216 |
| 3800 | 38000 | 100 | 178304 | 10 | 0,026315789 | 46,92210526 |
| 3900 | 39000 | 103 | 182993 | 10 | 0,026410256 | 46,92128205 |
| 11000 | 110000 | 292 | 516262 | 10 | 0,026545455 | 46,93290909 |
| 12000 | 120000 | 316 | 563165 | 10 | 0,026333333 | 46,93041667 |
| 13000 | 130000 | 341 | 610122 | 10 | 0,026230769 | 46,93246154 |
| 14000 | 140000 | 369 | 657105 | 10 | 0,026357143 | 46,93607143 |
| 15000 | 150000 | 395 | 704132 | 10 | 0,026333333 | 46,94213333 |

Çizelge B. 66 TripleDes Şifreleme Algoritması İle Elde Edilen Test Verileri

| 13-TripleDes | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 6 | 60 | 1 | 277 | 10 | 0,166666667 | 46,16666667 |
| 7 | 70 | 1 | 323 | 10 | 0,142857143 | 46,14285714 |
| 8 | 80 | 1 | 369 | 10 | 0,125 | 46,125 |
| 9 | 90 | 1 | 415 | 10 | 0,111111111 | 46,11111111 |
| 10 | 100 | 1 | 461 | 10 | 0,1 | 46,1 |
| 3500 | 35000 | 101 | 161683 | 10 | 0,028857143 | 46,19514286 |
| 3600 | 36000 | 104 | 166307 | 10 | 0,028888889 | 46,19638889 |
| 3700 | 37000 | 108 | 170925 | 10 | 0,029189189 | 46,19594595 |
| 3800 | 38000 | 111 | 175550 | 10 | 0,029210526 | 46,19736842 |
| 3900 | 39000 | 113 | 180169 | 10 | 0,028974359 | 46,19717949 |
| 11000 | 110000 | 317 | 508287 | 10 | 0,028818182 | 46,20790909 |
| 12000 | 120000 | 345 | 554499 | 10 | 0,02875 | 46,20825 |
| 13000 | 130000 | 375 | 600696 | 10 | 0,028846154 | 46,20738462 |
| 14000 | 140000 | 404 | 646952 | 10 | 0,028857143 | 46,21085714 |
| 15000 | 150000 | 432 | 693004 | 10 | 0,0288 | 46,20026667 |

Çizelge B. 67 TripleDesRsa15 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 14-TripleDesRsa15 | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 6 | 60 | 1 | 277 | 10 | 0,166667 | 46,16667 |
| 7 | 70 | 1 | 323 | 10 | 0,142857 | 46,14286 |
| 8 | 80 | 1 | 369 | 10 | 0,125 | 46,125 |
| 9 | 90 | 1 | 415 | 10 | 0,111111 | 46,11111 |
| 10 | 100 | 1 | 461 | 10 | 0,1 | 46,1 |
| 3500 | 35000 | 101 | 161686 | 10 | 0,028857 | 46,196 |
| 3600 | 36000 | 104 | 166307 | 10 | 0,028889 | 46,19639 |
| 3700 | 37000 | 107 | 170925 | 10 | 0,028919 | 46,19595 |
| 3800 | 38000 | 110 | 175546 | 10 | 0,028947 | 46,19632 |
| 3900 | 39000 | 112 | 180172 | 10 | 0,028718 | 46,19795 |
| 11000 | 110000 | 314 | 508277 | 10 | 0,028545 | 46,207 |
| 12000 | 120000 | 343 | 553491 | 10 | 0,028583 | 46,12425 |
| 13000 | 130000 | 373 | 600734 | 10 | 0,028692 | 46,21031 |
| 14000 | 140000 | 411 | 646828 | 10 | 0,002929 | 46,202 |
| 15000 | 150000 | 428 | 693170 | 10 | 0,028533 | 46,21133 |

Çizelge B. 68 TripleDesSha256 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 15-TripleDesSha256 | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 6 | 60 | 1 | 281 | 10 | 0,166666667 | 46,83333333 |
| 7 | 70 | 1 | 328 | 10 | 0,142857143 | 46,85714286 |
| 8 | 80 | 1 | 375 | 10 | 0,125 | 46,875 |
| 9 | 90 | 1 | 422 | 10 | 0,111111111 | 46,88888889 |
| 10 | 100 | 1 | 469 | 10 | 0,1 | 46,9 |
| 3500 | 35000 | 102 | 164352 | 10 | 0,029142857 | 46,95771429 |
| 3600 | 36000 | 105 | 169050 | 10 | 0,029166667 | 46,95833333 |
| 3700 | 37000 | 108 | 173747 | 10 | 0,029189189 | 46,95864865 |
| 3800 | 38000 | 111 | 178439 | 10 | 0,029210526 | 46,95763158 |
| 3900 | 39000 | 114 | 183133 | 10 | 0,029230769 | 46,95717949 |
| 11000 | 110000 | 321 | 516634 | 10 | 0,029181818 | 46,96672727 |
| 12000 | 120000 | 351 | 563477 | 10 | 0,02925 | 46,95641667 |
| 13000 | 130000 | 379 | 610591 | 10 | 0,029153846 | 46,96853846 |
| 14000 | 140000 | 409 | 657598 | 10 | 0,029214286 | 46,97128571 |
| 15000 | 150000 | 449 | 704589 | 10 | 0,029933333 | 46,9726 |

Çizelge B. 69 TripleDesSha256Rsa15 Şifreleme Algoritması İle Elde Edilen Test Verileri

| 16-TripleDesSha256Rsa15 | | | | | | |
|---|---|---|---|---|---|---|
| **Adet** | **T** | **S** | **B** | **Tverim** | **Sverim** | **Bverim** |
| 6 | 60 | 1 | 281 | 10 | 0,166666667 | 46,83333333 |
| 7 | 70 | 1 | 328 | 10 | 0,142857143 | 46,85714286 |
| 8 | 80 | 1 | 375 | 10 | 0,125 | 46,875 |
| 9 | 90 | 1 | 422 | 10 | 0,111111111 | 46,88888889 |
| 10 | 100 | 1 | 469 | 10 | 0,1 | 46,9 |
| 3500 | 35000 | 104 | 164352 | 10 | 0,029714286 | 46,95771429 |
| 3600 | 36000 | 106 | 169048 | 10 | 0,029444444 | 46,95777778 |
| 3700 | 37000 | 109 | 173743 | 10 | 0,029459459 | 46,95756757 |
| 3800 | 38000 | 112 | 178436 | 10 | 0,029473684 | 46,95684211 |
| 3900 | 39000 | 114 | 183134 | 10 | 0,029230769 | 46,9574359 |
| 11000 | 110000 | 321 | 516645 | 10 | 0,029181818 | 46,96772727 |
| 12000 | 120000 | 351 | 563602 | 10 | 0,02925 | 46,96683333 |
| 13000 | 130000 | 380 | 610581 | 10 | 0,029230769 | 46,96776923 |
| 14000 | 140000 | 412 | 657874 | 10 | 0,029428571 | 46,991 |
| 15000 | 150000 | 439 | 704662 | 10 | 0,029266667 | 46,97746667 |

## B - 4 Tek Ortamında Elde Edilen İlişki Tabloları Test Verileri

### B - 4. 1 Tek İstemcili NetTcpBinding Bağlamalı Ortamda Elde Edilen İlişki Tabloları Test Verileri

Çizelge B. 70 Tek İstemcili NetTcpBinding Bağlamalı Ortamda Elde Edilen İlişki Tabloları Test Verileri

| | B | B1 | B2 | B3 | B4 | B5 | B6 |
|---|---|---|---|---|---|---|---|
| B | 1.0000000 | 0.9955378 | 0.9955378 | 0.9977783 | 0.9977783 | 0.9955378 | 0.9955378 |
| B1 | 0.9955378 | 1.0000000 | 1.0000000 | 0.9973799 | 0.9973799 | 1.0000000 | 1.0000000 |
| B2 | 0.9955378 | 1.0000000 | 1.0000000 | 0.9973799 | 0.9973799 | 1.0000000 | 1.0000000 |
| B3 | 0.9977783 | 0.9973799 | 0.9973799 | 1.0000000 | 1.0000000 | 0.9973799 | 0.9973799 |
| B4 | 0.9977783 | 0.9973799 | 0.9973799 | 1.0000000 | 1.0000000 | 0.9973799 | 0.9973799 |
| B5 | 0.9955378 | 1.0000000 | 1.0000000 | 0.9973799 | 0.9973799 | 1.0000000 | 1.0000000 |
| B6 | 0.9955378 | 1.0000000 | 1.0000000 | 0.9973799 | 0.9973799 | 1.0000000 | 1.0000000 |
| B7 | 0.9977783 | 0.9973799 | 0.9973799 | 1.0000000 | 1.0000000 | 0.9973799 | 0.9973799 |
| B8 | 0.9977783 | 0.9973799 | 0.9973799 | 1.0000000 | 1.0000000 | 0.9973799 | 0.9973799 |
| B9 | 0.9977783 | 0.9973799 | 0.9973799 | 1.0000000 | 1.0000000 | 0.9973799 | 0.9973799 |
| B10 | 0.9977783 | 0.9973799 | 0.9973799 | 1.0000000 | 1.0000000 | 0.9973799 | 0.9973799 |
| B11 | 0.9981186 | 0.9948253 | 0.9948253 | 0.9974307 | 0.9974307 | 0.9948253 | 0.9948253 |
| B12 | 0.9981186 | 0.9948253 | 0.9948253 | 0.9974307 | 0.9974307 | 0.9948253 | 0.9948253 |
| B13 | 0.9977783 | 0.9973799 | 0.9973799 | 1.0000000 | 1.0000000 | 0.9973799 | 0.9973799 |
| B14 | 0.9977783 | 0.9973799 | 0.9973799 | 1.0000000 | 1.0000000 | 0.9973799 | 0.9973799 |
| B15 | 0.9981186 | 0.9948253 | 0.9948253 | 0.9974307 | 0.9974307 | 0.9948253 | 0.9948253 |
| B16 | 0.9981186 | 0.9948253 | 0.9948253 | 0.9974307 | 0.9974307 | 0.9948253 | 0.9948253 |

| | B7 | B8 | B9 | B10 | B11 | B12 | B13 |
|---|---|---|---|---|---|---|---|
| B | 0.9977783 | 0.9977783 | 0.9977783 | 0.9977783 | 0.9981186 | 0.9981186 | 0.9977783 |
| B1 | 0.9973799 | 0.9973799 | 0.9973799 | 0.9973799 | 0.9948253 | 0.9948253 | 0.9973799 |
| B2 | 0.9973799 | 0.9973799 | 0.9973799 | 0.9973799 | 0.9948253 | 0.9948253 | 0.9973799 |
| B3 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 | 0.9974307 | 0.9974307 | 1.0000000 |
| B4 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 | 0.9974307 | 0.9974307 | 1.0000000 |
| B5 | 0.9973799 | 0.9973799 | 0.9973799 | 0.9973799 | 0.9948253 | 0.9948253 | 0.9973799 |
| B6 | 0.9973799 | 0.9973799 | 0.9973799 | 0.9973799 | 0.9948253 | 0.9948253 | 0.9973799 |
| B7 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 | 0.9974307 | 0.9974307 | 1.0000000 |
| B8 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 | 0.9974307 | 0.9974307 | 1.0000000 |
| B9 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 | 0.9974307 | 0.9974307 | 1.0000000 |
| B10 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 | 0.9974307 | 0.9974307 | 1.0000000 |
| B11 | 0.9974307 | 0.9974307 | 0.9974307 | 0.9974307 | 1.0000000 | 1.0000000 | 0.9974307 |
| B12 | 0.9974307 | 0.9974307 | 0.9974307 | 0.9974307 | 1.0000000 | 1.0000000 | 0.9974307 |
| B13 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 | 0.9974307 | 0.9974307 | 1.0000000 |
| B14 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 | 0.9974307 | 0.9974307 | 1.0000000 |
| B15 | 0.9974307 | 0.9974307 | 0.9974307 | 0.9974307 | 1.0000000 | 1.0000000 | 0.9974307 |
| B16 | 0.9974307 | 0.9974307 | 0.9974307 | 0.9974307 | 1.0000000 | 1.0000000 | 0.9974307 |

Çizelge B. 70 Tek İstemcili NetTcpBinding Bağlamalı Ortamda Elde Edilen İlişki Tabloları Test Verileri (devamı)

| | B14 | B15 | B16 |
|---|---|---|---|
| B | 0.9977783 | 0.9981186 | 0.9981186 |
| B1 | 0.9973799 | 0.9948253 | 0.9948253 |
| B2 | 0.9973799 | 0.9948253 | 0.9948253 |
| B3 | 1.0000000 | 0.9974307 | 0.9974307 |
| B4 | 1.0000000 | 0.9974307 | 0.9974307 |
| B5 | 0.9973799 | 0.9948253 | 0.9948253 |
| B6 | 0.9973799 | 0.9948253 | 0.9948253 |
| B7 | 1.0000000 | 0.9974307 | 0.9974307 |
| B8 | 1.0000000 | 0.9974307 | 0.9974307 |
| B9 | 1.0000000 | 0.9974307 | 0.9974307 |
| B10 | 1.0000000 | 0.9974307 | 0.9974307 |
| B11 | 0.9974307 | 1.0000000 | 1.0000000 |
| B12 | 0.9974307 | 1.0000000 | 1.0000000 |
| B13 | 1.0000000 | 0.9974307 | 0.9974307 |
| B14 | 1.0000000 | 0.9974307 | 0.9974307 |
| B15 | 0.9974307 | 1.0000000 | 1.0000000 |
| B16 | 0.9974307 | 1.0000000 | 1.0000000 |

| | Bverim | Bverim1 | Bverim2 | Bverim3 | Bverim4 | Bverim5 |
|---|---|---|---|---|---|---|
| Bverim | 1.0000000 | -0.6815454 | -0.6815454 | -0.6801486 | -0.6801486 | -0.6815454 |
| Bverim1 | -0.6815454 | 1.0000000 | 1.0000000 | 0.9999911 | 0.9999911 | 1.0000000 |
| Bverim2 | -0.6815454 | 1.0000000 | 1.0000000 | 0.9999911 | 0.9999911 | 1.0000000 |
| Bverim3 | -0.6801486 | 0.9999911 | 0.9999911 | 1.0000000 | 1.0000000 | 0.9999911 |
| Bverim4 | -0.6801486 | 0.9999911 | 0.9999911 | 1.0000000 | 1.0000000 | 0.9999911 |
| Bverim5 | -0.6815454 | 1.0000000 | 1.0000000 | 0.9999911 | 0.9999911 | 1.0000000 |
| Bverim6 | -0.6815454 | 1.0000000 | 1.0000000 | 0.9999911 | 0.9999911 | 1.0000000 |
| Bverim7 | -0.6801486 | 0.9999911 | 0.9999911 | 1.0000000 | 1.0000000 | 0.9999911 |
| Bverim8 | -0.6801486 | 0.9999911 | 0.9999911 | 1.0000000 | 1.0000000 | 0.9999911 |
| Bverim9 | -0.6801077 | 0.9999909 | 0.9999909 | 1.0000000 | 1.0000000 | 0.9999909 |
| Bverim10 | -0.6801077 | 0.9999909 | 0.9999909 | 1.0000000 | 1.0000000 | 0.9999909 |
| Bverim11 | -0.6801986 | 0.9999912 | 0.9999912 | 0.9999999 | 0.9999999 | 0.9999912 |
| Bverim12 | -0.6801986 | 0.9999912 | 0.9999912 | 0.9999999 | 0.9999999 | 0.9999912 |
| Bverim13 | -0.6801077 | 0.9999909 | 0.9999909 | 1.0000000 | 1.0000000 | 0.9999909 |
| Bverim14 | -0.6801077 | 0.9999909 | 0.9999909 | 1.0000000 | 1.0000000 | 0.9999909 |
| Bverim15 | -0.6801986 | 0.9999912 | 0.9999912 | 0.9999999 | 0.9999999 | 0.9999912 |
| Bverim16 | -0.6801986 | 0.9999912 | 0.9999912 | 0.9999999 | 0.9999999 | 0.9999912 |

Çizelge B. 70 Tek İstemcili NetTcpBinding Bağlamalı Ortamda Elde Edilen İlişki Tabloları Test Verileri (devamı)

| | Bverim6 | Bverim7 | Bverim8 | Bverim9 | Bverim10 | Bverim11 |
|---|---|---|---|---|---|---|
| Bverim | -0.6815454 | -0.6801486 | -0.6801486 | -0.6801077 | -0.6801077 | -0.6801986 |
| Bverim1 | 1.0000000 | 0.9999911 | 0.9999911 | 0.9999909 | 0.9999909 | 0.9999912 |
| Bverim2 | 1.0000000 | 0.9999911 | 0.9999911 | 0.9999909 | 0.9999909 | 0.9999912 |
| Bverim3 | 0.9999911 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 | 0.9999999 |
| Bverim4 | 0.9999911 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 | 0.9999999 |
| Bverim5 | 1.0000000 | 0.9999911 | 0.9999911 | 0.9999909 | 0.9999909 | 0.9999912 |
| Bverim6 | 1.0000000 | 0.9999911 | 0.9999911 | 0.9999909 | 0.9999909 | 0.9999912 |
| Bverim7 | 0.9999911 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 | 0.9999999 |
| Bverim8 | 0.9999911 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 | 0.9999999 |
| Bverim9 | 0.9999909 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 | 0.9999999 |
| Bverim10 | 0.9999909 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 | 0.9999999 |
| Bverim11 | 0.9999912 | 0.9999999 | 0.9999999 | 0.9999999 | 0.9999999 | 1.0000000 |
| Bverim12 | 0.9999912 | 0.9999999 | 0.9999999 | 0.9999999 | 0.9999999 | 1.0000000 |
| Bverim13 | 0.9999909 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 | 0.9999999 |
| Bverim14 | 0.9999909 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 | 0.9999999 |
| Bverim15 | 0.9999912 | 0.9999999 | 0.9999999 | 0.9999999 | 0.9999999 | 1.0000000 |
| Bverim16 | 0.9999912 | 0.9999999 | 0.9999999 | 0.9999999 | 0.9999999 | 1.0000000 |

| | Bverim12 | Bverim13 | Bverim14 | Bverim15 | Bverim16 |
|---|---|---|---|---|---|
| Bverim | -0.6801986 | -0.6801077 | -0.6801077 | -0.6801986 | -0.6801986 |
| Bverim1 | 0.9999912 | 0.9999909 | 0.9999909 | 0.9999912 | 0.9999912 |
| Bverim2 | 0.9999912 | 0.9999909 | 0.9999909 | 0.9999912 | 0.9999912 |
| Bverim3 | 0.9999999 | 1.0000000 | 1.0000000 | 0.9999999 | 0.9999999 |
| Bverim4 | 0.9999999 | 1.0000000 | 1.0000000 | 0.9999999 | 0.9999999 |
| Bverim5 | 0.9999912 | 0.9999909 | 0.9999909 | 0.9999912 | 0.9999912 |
| Bverim6 | 0.9999912 | 0.9999909 | 0.9999909 | 0.9999912 | 0.9999912 |
| Bverim7 | 0.9999999 | 1.0000000 | 1.0000000 | 0.9999999 | 0.9999999 |
| Bverim8 | 0.9999999 | 1.0000000 | 1.0000000 | 0.9999999 | 0.9999999 |
| Bverim9 | 0.9999999 | 1.0000000 | 1.0000000 | 0.9999999 | 0.9999999 |
| Bverim10 | 0.9999999 | 1.0000000 | 1.0000000 | 0.9999999 | 0.9999999 |
| Bverim11 | 1.0000000 | 0.9999999 | 0.9999999 | 1.0000000 | 1.0000000 |
| Bverim12 | 1.0000000 | 0.9999999 | 0.9999999 | 1.0000000 | 1.0000000 |
| Bverim13 | 0.9999999 | 1.0000000 | 1.0000000 | 0.9999999 | 0.9999999 |
| Bverim14 | 0.9999999 | 1.0000000 | 1.0000000 | 0.9999999 | 0.9999999 |
| Bverim15 | 1.0000000 | 0.9999999 | 0.9999999 | 1.0000000 | 1.0000000 |
| Bverim16 | 1.0000000 | 0.9999999 | 0.9999999 | 1.0000000 | 1.0000000 |

### B - 4. 2 Tek İstemcili WsHttpBinding Bağlamalı Ortamda Elde Edilen İlişki Tablosu Değerleri

Çizelge B. 71 Tek İstemcili WsHttpBinding Bağlamalı Ortamda Elde Edilen İlişki Tabloları Test Verileri

| | B | B1 | B2 | B3 | B4 | B5 | B6 |
|---|---|---|---|---|---|---|---|
| B | 1.0000000 | 0.9923005 | 0.9923005 | 0.9923596 | 0.9923596 | 0.9923005 | 0.9923005 |
| B1 | 0.9923005 | 1.0000000 | 1.0000000 | 0.9947950 | 0.9947950 | 1.0000000 | 1.0000000 |
| B2 | 0.9923005 | 1.0000000 | 1.0000000 | 0.9947950 | 0.9947950 | 1.0000000 | 1.0000000 |
| B3 | 0.9923596 | 0.9947950 | 0.9947950 | 1.0000000 | 1.0000000 | 0.9947950 | 0.9947950 |
| B4 | 0.9923596 | 0.9947950 | 0.9947950 | 1.0000000 | 1.0000000 | 0.9947950 | 0.9947950 |
| B5 | 0.9923005 | 1.0000000 | 1.0000000 | 0.9947950 | 0.9947950 | 1.0000000 | 1.0000000 |
| B6 | 0.9923005 | 1.0000000 | 1.0000000 | 0.9947950 | 0.9947950 | 1.0000000 | 1.0000000 |
| B7 | 0.9923005 | 1.0000000 | 1.0000000 | 0.9947950 | 0.9947950 | 1.0000000 | 1.0000000 |
| B8 | 0.9923005 | 1.0000000 | 1.0000000 | 0.9947950 | 0.9947950 | 1.0000000 | 1.0000000 |
| B9 | 0.9923005 | 1.0000000 | 1.0000000 | 0.9947950 | 0.9947950 | 1.0000000 | 1.0000000 |
| B10 | 0.9923005 | 1.0000000 | 1.0000000 | 0.9947950 | 0.9947950 | 1.0000000 | 1.0000000 |
| B11 | 0.9923721 | 0.9931768 | 0.9931768 | 0.9970527 | 0.9970527 | 0.9931768 | 0.9931768 |
| B12 | 0.9923721 | 0.9931768 | 0.9931768 | 0.9970527 | 0.9970527 | 0.9931768 | 0.9931768 |
| B13 | 0.9923005 | 1.0000000 | 1.0000000 | 0.9947950 | 0.9947950 | 1.0000000 | 1.0000000 |
| B14 | 0.9923005 | 1.0000000 | 1.0000000 | 0.9947950 | 0.9947950 | 1.0000000 | 1.0000000 |
| B15 | 0.9923721 | 0.9931768 | 0.9931768 | 0.9970527 | 0.9970527 | 0.9931768 | 0.9931768 |
| B16 | 0.9923721 | 0.9931768 | 0.9931768 | 0.9970527 | 0.9970527 | 0.9931768 | 0.9931768 |

| | B7 | B8 | B9 | B10 | B11 | B12 | B13 |
|---|---|---|---|---|---|---|---|
| B | 0.9923005 | 0.9923005 | 0.9923005 | 0.9923005 | 0.9923721 | 0.9923721 | 0.9923005 |
| B1 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 | 0.9931768 | 0.9931768 | 1.0000000 |
| B2 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 | 0.9931768 | 0.9931768 | 1.0000000 |
| B3 | 0.9947950 | 0.9947950 | 0.9947950 | 0.9947950 | 0.9970527 | 0.9970527 | 0.9947950 |
| B4 | 0.9947950 | 0.9947950 | 0.9947950 | 0.9947950 | 0.9970527 | 0.9970527 | 0.9947950 |
| B5 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 | 0.9931768 | 0.9931768 | 1.0000000 |
| B6 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 | 0.9931768 | 0.9931768 | 1.0000000 |
| B7 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 | 0.9931768 | 0.9931768 | 1.0000000 |
| B8 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 | 0.9931768 | 0.9931768 | 1.0000000 |
| B9 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 | 0.9931768 | 0.9931768 | 1.0000000 |
| B10 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 | 0.9931768 | 0.9931768 | 1.0000000 |
| B11 | 0.9931768 | 0.9931768 | 0.9931768 | 0.9931768 | 1.0000000 | 1.0000000 | 0.9931768 |
| B12 | 0.9931768 | 0.9931768 | 0.9931768 | 0.9931768 | 1.0000000 | 1.0000000 | 0.9931768 |
| B13 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 | 0.9931768 | 0.9931768 | 1.0000000 |
| B14 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 | 0.9931768 | 0.9931768 | 1.0000000 |
| B15 | 0.9931768 | 0.9931768 | 0.9931768 | 0.9931768 | 1.0000000 | 1.0000000 | 0.9931768 |
| B16 | 0.9931768 | 0.9931768 | 0.9931768 | 0.9931768 | 1.0000000 | 1.0000000 | 0.9931768 |

| | B14 | B15 | B16 |
|---|---|---|---|
| B | 0.9923005 | 0.9923721 | 0.9923721 |
| B1 | 1.0000000 | 0.9931768 | 0.9931768 |
| B2 | 1.0000000 | 0.9931768 | 0.9931768 |
| B3 | 0.9947950 | 0.9970527 | 0.9970527 |
| B4 | 0.9947950 | 0.9970527 | 0.9970527 |
| B5 | 1.0000000 | 0.9931768 | 0.9931768 |
| B6 | 1.0000000 | 0.9931768 | 0.9931768 |
| B7 | 1.0000000 | 0.9931768 | 0.9931768 |
| B8 | 1.0000000 | 0.9931768 | 0.9931768 |
| B9 | 1.0000000 | 0.9931768 | 0.9931768 |
| B10 | 1.0000000 | 0.9931768 | 0.9931768 |
| B11 | 0.9931768 | 1.0000000 | 1.0000000 |
| B12 | 0.9931768 | 1.0000000 | 1.0000000 |
| B13 | 1.0000000 | 0.9931768 | 0.9931768 |
| B14 | 1.0000000 | 0.9931768 | 0.9931768 |
| B15 | 0.9931768 | 1.0000000 | 1.0000000 |
| B16 | 0.9931768 | 1.0000000 | 1.0000000 |

Çizelge B. 71 Tek İstemcili WsHttpBinding Bağlamalı Ortamda Elde Edilen İlişki Tabloları Test Verileri (devamı)

| | Bverim | Bverim1 | Bverim2 | Bverim3 | Bverim4 | Bverim5 | Bverim6 |
|---|---|---|---|---|---|---|---|
| Bverim | 1.0000000 | 0.9966864 | 0.9966864 | 0.9966409 | 0.9966409 | 0.9966864 | 0.9966864 |
| Bverim1 | 0.9966864 | 1.0000000 | 1.0000000 | 0.9999352 | 0.9999352 | 1.0000000 | 1.0000000 |
| Bverim2 | 0.9966864 | 1.0000000 | 1.0000000 | 0.9999352 | 0.9999352 | 1.0000000 | 1.0000000 |
| Bverim3 | 0.9966409 | 0.9999352 | 0.9999352 | 1.0000000 | 1.0000000 | 0.9999352 | 0.9999352 |
| Bverim4 | 0.9966409 | 0.9999352 | 0.9999352 | 1.0000000 | 1.0000000 | 0.9999352 | 0.9999352 |
| Bverim5 | 0.9966864 | 1.0000000 | 1.0000000 | 0.9999352 | 0.9999352 | 1.0000000 | 1.0000000 |
| Bverim6 | 0.9966864 | 1.0000000 | 1.0000000 | 0.9999352 | 0.9999352 | 1.0000000 | 1.0000000 |
| Bverim7 | 0.9966890 | 1.0000000 | 1.0000000 | 0.9999358 | 0.9999358 | 1.0000000 | 1.0000000 |
| Bverim8 | 0.9966890 | 1.0000000 | 1.0000000 | 0.9999358 | 0.9999358 | 1.0000000 | 1.0000000 |
| Bverim9 | 0.9966864 | 1.0000000 | 1.0000000 | 0.9999352 | 0.9999352 | 1.0000000 | 1.0000000 |
| Bverim10 | 0.9966864 | 1.0000000 | 1.0000000 | 0.9999352 | 0.9999352 | 1.0000000 | 1.0000000 |
| Bverim11 | 0.9966447 | 0.9999341 | 0.9999341 | 0.9999999 | 0.9999999 | 0.9999341 | 0.9999341 |
| Bverim12 | 0.9966447 | 0.9999341 | 0.9999341 | 0.9999999 | 0.9999999 | 0.9999341 | 0.9999341 |
| Bverim13 | 0.9966864 | 1.0000000 | 1.0000000 | 0.9999352 | 0.9999352 | 1.0000000 | 1.0000000 |
| Bverim14 | 0.9966864 | 1.0000000 | 1.0000000 | 0.9999352 | 0.9999352 | 1.0000000 | 1.0000000 |
| Bverim15 | 0.9966447 | 0.9999341 | 0.9999341 | 0.9999999 | 0.9999999 | 0.9999341 | 0.9999341 |
| Bverim16 | 0.9966447 | 0.9999341 | 0.9999341 | 0.9999999 | 0.9999999 | 0.9999341 | 0.9999341 |

| | Bverim7 | Bverim8 | Bverim9 | Bverim10 | Bverim11 | Bverim12 | Bverim13 |
|---|---|---|---|---|---|---|---|
| Bverim | 0.9966890 | 0.9966890 | 0.9966864 | 0.9966864 | 0.9966447 | 0.9966447 | 0.9966864 |
| Bverim1 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 | 0.9999341 | 0.9999341 | 1.0000000 |
| Bverim2 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 | 0.9999341 | 0.9999341 | 1.0000000 |
| Bverim3 | 0.9999358 | 0.9999358 | 0.9999352 | 0.9999352 | 0.9999999 | 0.9999999 | 0.9999352 |
| Bverim4 | 0.9999358 | 0.9999358 | 0.9999352 | 0.9999352 | 0.9999999 | 0.9999999 | 0.9999352 |
| Bverim5 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 | 0.9999341 | 0.9999341 | 1.0000000 |
| Bverim6 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 | 0.9999341 | 0.9999341 | 1.0000000 |
| Bverim7 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 | 0.9999347 | 0.9999347 | 1.0000000 |
| Bverim8 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 | 0.9999347 | 0.9999347 | 1.0000000 |
| Bverim9 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 | 0.9999341 | 0.9999341 | 1.0000000 |
| Bverim10 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 | 0.9999341 | 0.9999341 | 1.0000000 |
| Bverim11 | 0.9999347 | 0.9999347 | 0.9999341 | 0.9999341 | 1.0000000 | 1.0000000 | 0.9999341 |
| Bverim12 | 0.9999347 | 0.9999347 | 0.9999341 | 0.9999341 | 1.0000000 | 1.0000000 | 0.9999341 |
| Bverim13 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 | 0.9999341 | 0.9999341 | 1.0000000 |
| Bverim14 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 | 0.9999341 | 0.9999341 | 1.0000000 |
| Bverim15 | 0.9999347 | 0.9999347 | 0.9999341 | 0.9999341 | 1.0000000 | 1.0000000 | 0.9999341 |
| Bverim16 | 0.9999347 | 0.9999347 | 0.9999341 | 0.9999341 | 1.0000000 | 1.0000000 | 0.9999341 |

| | Bverim14 | Bverim15 | Bverim16 |
|---|---|---|---|
| Bverim | 0.9966864 | 0.9966447 | 0.9966447 |
| Bverim1 | 1.0000000 | 0.9999341 | 0.9999341 |
| Bverim2 | 1.0000000 | 0.9999341 | 0.9999341 |
| Bverim3 | 0.9999352 | 0.9999999 | 0.9999999 |
| Bverim4 | 0.9999352 | 0.9999999 | 0.9999999 |
| Bverim5 | 1.0000000 | 0.9999341 | 0.9999341 |
| Bverim6 | 1.0000000 | 0.9999341 | 0.9999341 |
| Bverim7 | 1.0000000 | 0.9999347 | 0.9999347 |
| Bverim8 | 1.0000000 | 0.9999347 | 0.9999347 |
| Bverim9 | 1.0000000 | 0.9999341 | 0.9999341 |
| Bverim10 | 1.0000000 | 0.9999341 | 0.9999341 |
| Bverim11 | 0.9999341 | 1.0000000 | 1.0000000 |
| Bverim12 | 0.9999341 | 1.0000000 | 1.0000000 |
| Bverim13 | 1.0000000 | 0.9999341 | 0.9999341 |
| Bverim14 | 1.0000000 | 0.9999341 | 0.9999341 |
| Bverim15 | 0.9999341 | 1.0000000 | 1.0000000 |
| Bverim16 | 0.9999341 | 1.0000000 | 1.0000000 |

### B - 4. 3 Çok İstemcili NetTcpBinding Bağlamalı Ortamda Elde Edilen İlişki Tablosu Değerleri

Çizelge B. 72 Çok İstemcili NetTcpBinding Bağlamalı Ortamda Elde Edilen İlişki Tabloları Test Verileri

| | B | B1 | B2 | B3 | B4 | B5 | B6 |
|---|---|---|---|---|---|---|---|
| B | 1.0000000 | 0.9999999 | 0.9999999 | 0.9999997 | 1.0000000 | 0.9999998 | 1.0000000 |
| B1 | 0.9999999 | 1.0000000 | 0.9999999 | 0.9999996 | 0.9999999 | 0.9999998 | 1.0000000 |
| B2 | 0.9999999 | 0.9999999 | 1.0000000 | 0.9999996 | 0.9999999 | 0.9999998 | 0.9999999 |
| B3 | 0.9999997 | 0.9999996 | 0.9999996 | 1.0000000 | 0.9999997 | 0.9999995 | 0.9999997 |
| B4 | 1.0000000 | 0.9999999 | 0.9999999 | 0.9999997 | 1.0000000 | 0.9999998 | 1.0000000 |
| B5 | 0.9999998 | 0.9999998 | 0.9999998 | 0.9999995 | 0.9999998 | 1.0000000 | 0.9999999 |
| B6 | 1.0000000 | 1.0000000 | 0.9999999 | 0.9999997 | 1.0000000 | 0.9999999 | 1.0000000 |
| B7 | 1.0000000 | 1.0000000 | 1.0000000 | 0.9999997 | 1.0000000 | 0.9999999 | 1.0000000 |
| B8 | 1.0000000 | 0.9999999 | 1.0000000 | 0.9999997 | 1.0000000 | 0.9999999 | 1.0000000 |
| B9 | 0.9999998 | 0.9999998 | 0.9999998 | 0.9999996 | 0.9999998 | 0.9999998 | 0.9999999 |
| B10 | 1.0000000 | 0.9999999 | 1.0000000 | 0.9999997 | 1.0000000 | 0.9999999 | 1.0000000 |
| B11 | 1.0000000 | 0.9999999 | 1.0000000 | 0.9999997 | 1.0000000 | 0.9999999 | 1.0000000 |
| B12 | 1.0000000 | 0.9999999 | 1.0000000 | 0.9999997 | 1.0000000 | 0.9999999 | 1.0000000 |
| B13 | 1.0000000 | 0.9999999 | 1.0000000 | 0.9999997 | 1.0000000 | 0.9999999 | 1.0000000 |
| B14 | 0.9999999 | 0.9999998 | 0.9999998 | 0.9999995 | 0.9999999 | 0.9999998 | 0.9999999 |
| B15 | 1.0000000 | 0.9999999 | 1.0000000 | 0.9999997 | 1.0000000 | 0.9999999 | 1.0000000 |
| B16 | 0.9999855 | 0.9999857 | 0.9999853 | 0.9999854 | 0.9999856 | 0.9999858 | 0.9999858 |

| | B7 | B8 | B9 | B10 | B11 | B12 | B13 |
|---|---|---|---|---|---|---|---|
| B | 1.0000000 | 1.0000000 | 0.9999998 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 |
| B1 | 1.0000000 | 0.9999999 | 0.9999998 | 0.9999999 | 0.9999999 | 0.9999999 | 0.9999999 |
| B2 | 1.0000000 | 1.0000000 | 0.9999998 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 |
| B3 | 0.9999997 | 0.9999997 | 0.9999996 | 0.9999997 | 0.9999997 | 0.9999997 | 0.9999997 |
| B4 | 1.0000000 | 1.0000000 | 0.9999998 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 |
| B5 | 0.9999999 | 0.9999999 | 0.9999998 | 0.9999999 | 0.9999999 | 0.9999999 | 0.9999999 |
| B6 | 1.0000000 | 1.0000000 | 0.9999999 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 |
| B7 | 1.0000000 | 1.0000000 | 0.9999999 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 |
| B8 | 1.0000000 | 1.0000000 | 0.9999999 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 |
| B9 | 0.9999999 | 0.9999999 | 1.0000000 | 0.9999999 | 0.9999999 | 0.9999999 | 0.9999999 |
| B10 | 1.0000000 | 1.0000000 | 0.9999999 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 |
| B11 | 1.0000000 | 1.0000000 | 0.9999999 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 |
| B12 | 1.0000000 | 1.0000000 | 0.9999999 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 |
| B13 | 1.0000000 | 1.0000000 | 0.9999999 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 |
| B14 | 0.9999999 | 0.9999999 | 0.9999998 | 0.9999999 | 0.9999999 | 0.9999999 | 0.9999999 |
| B15 | 1.0000000 | 1.0000000 | 0.9999999 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 |
| B16 | 0.9999857 | 0.9999857 | 0.9999855 | 0.9999858 | 0.9999857 | 0.9999857 | 0.9999857 |

| | B14 | B15 | B16 |
|---|---|---|---|
| B | 0.9999999 | 1.0000000 | 0.9999855 |
| B1 | 0.9999998 | 0.9999999 | 0.9999857 |
| B2 | 0.9999998 | 1.0000000 | 0.9999853 |
| B3 | 0.9999995 | 0.9999997 | 0.9999854 |
| B4 | 0.9999999 | 1.0000000 | 0.9999856 |
| B5 | 0.9999998 | 0.9999999 | 0.9999858 |
| B6 | 0.9999999 | 1.0000000 | 0.9999858 |
| B7 | 0.9999999 | 1.0000000 | 0.9999857 |
| B8 | 0.9999999 | 1.0000000 | 0.9999857 |
| B9 | 0.9999998 | 0.9999999 | 0.9999855 |
| B10 | 0.9999999 | 1.0000000 | 0.9999858 |
| B11 | 0.9999999 | 1.0000000 | 0.9999857 |
| B12 | 0.9999999 | 1.0000000 | 0.9999857 |
| B13 | 0.9999999 | 1.0000000 | 0.9999857 |
| B14 | 1.0000000 | 0.9999999 | 0.9999857 |
| B15 | 0.9999999 | 1.0000000 | 0.9999856 |
| B16 | 0.9999857 | 0.9999856 | 1.0000000 |

Çizelge B. 72 Çok İstemcili NetTcpBinding Bağlamalı Ortamda Elde Edilen İlişki Tabloları Test Verileri (devamı)

| | Bverim | Bverim1 | Bverim2 | Bverim3 | Bverim4 | Bverim5 |
|---|---|---|---|---|---|---|
| Bverim | 1.00000000 | 0.06024231 | 0.06868328 | 0.07477762 | 0.270522690 | -0.15285036 |
| Bverim1 | 0.06024231 | 1.00000000 | 0.99423191 | 0.81910489 | 0.036271975 | 0.70792404 |
| Bverim2 | 0.06868328 | 0.99423191 | 1.00000000 | 0.81770084 | 0.038032612 | 0.70604353 |
| Bverim3 | 0.07477762 | 0.81910489 | 0.81770084 | 1.00000000 | -0.177753420 | 0.60626843 |
| Bverim4 | 0.27052269 | 0.03627198 | 0.03803261 | -0.17775342 | 1.000000000 | 0.02896951 |
| Bverim5 | -0.15285036 | 0.70792404 | 0.70604353 | 0.60626843 | 0.028969507 | 1.00000000 |
| Bverim6 | -0.24381278 | 0.30146222 | 0.29906234 | 0.45767726 | -0.939002092 | 0.23157499 |
| Bverim7 | 0.08712598 | 0.85290860 | 0.85167776 | 0.95664689 | 0.009337178 | 0.65273950 |
| Bverim8 | 0.08582656 | 0.85204868 | 0.85187003 | 0.95610199 | 0.008977346 | 0.65312981 |
| Bverim9 | -0.02965074 | 0.80336222 | 0.80344202 | 0.90325041 | -0.020773642 | 0.63956736 |
| Bverim10 | 0.01630852 | 0.85012088 | 0.84943594 | 0.95499185 | -0.009860467 | 0.66656583 |
| Bverim11 | 0.26782150 | 0.64777659 | 0.64873189 | 0.65941361 | -0.034604703 | 0.52235640 |
| Bverim12 | 0.09947245 | 0.86996959 | 0.86984169 | 0.92025362 | -0.009492032 | 0.68395352 |
| Bverim13 | 0.07647030 | 0.79877953 | 0.79935192 | 0.93262410 | -0.047746414 | 0.48595120 |
| Bverim14 | 0.08183955 | 0.78638200 | 0.78719773 | 0.91823720 | -0.045175255 | 0.47783139 |
| Bverim15 | 0.16391774 | 0.85630221 | 0.85637075 | 0.90872244 | -0.017398671 | 0.52492996 |
| Bverim16 | -0.12089428 | 0.31544206 | 0.27335286 | 0.33008661 | -0.052662327 | 0.23159420 |

| | Bverim6 | Bverim7 | Bverim8 | Bverim9 | Bverim10 | Bverim11 |
|---|---|---|---|---|---|---|
| Bverim | -0.2438128 | 0.087125981 | 0.085826558 | -0.02965074 | 0.016308524 | 0.2678215 |
| Bverim1 | 0.3014622 | 0.852908595 | 0.852048684 | 0.80336222 | 0.850120877 | 0.6477766 |
| Bverim2 | 0.2990623 | 0.851677761 | 0.851870031 | 0.80344202 | 0.849435940 | 0.6487319 |
| Bverim3 | 0.4576773 | 0.956646887 | 0.956101987 | 0.90325041 | 0.954991846 | 0.6594136 |
| Bverim4 | -0.9390021 | 0.009337178 | 0.008977346 | -0.02077364 | -0.009860467 | -0.0346047 |
| Bverim5 | 0.2315750 | 0.652739497 | 0.653129810 | 0.63956736 | 0.666565831 | 0.5223564 |
| Bverim6 | 1.0000000 | 0.289236955 | 0.289563492 | 0.30213462 | 0.307607861 | 0.2499284 |
| Bverim7 | 0.2892370 | 1.000000000 | 0.999698284 | 0.94254331 | 0.996851891 | 0.7552990 |
| Bverim8 | 0.2895635 | 0.999698284 | 1.000000000 | 0.94221827 | 0.996850931 | 0.7549168 |
| Bverim9 | 0.3021346 | 0.942543313 | 0.942218273 | 1.00000000 | 0.950289405 | 0.6905914 |
| Bverim10 | 0.3076079 | 0.996851891 | 0.996850931 | 0.95028940 | 1.000000000 | 0.7386746 |
| Bverim11 | 0.2499284 | 0.755298993 | 0.754916772 | 0.69059142 | 0.738674560 | 1.0000000 |
| Bverim12 | 0.3150302 | 0.941915719 | 0.941816852 | 0.88562220 | 0.938082965 | 0.7486312 |
| Bverim13 | 0.3346548 | 0.953181840 | 0.953285635 | 0.90110742 | 0.951639869 | 0.6774583 |
| Bverim14 | 0.3289292 | 0.939058439 | 0.939230646 | 0.89009100 | 0.940389208 | 0.6686184 |
| Bverim15 | 0.3147129 | 0.919975028 | 0.919814746 | 0.85844043 | 0.911697321 | 0.7216241 |
| Bverim16 | 0.1602476 | 0.331869494 | 0.330396600 | 0.31171257 | 0.348489518 | 0.2279028 |

| | Bverim12 | Bverim13 | Bverim14 | Bverim15 | Bverim16 |
|---|---|---|---|---|---|
| Bverim | 0.099472451 | 0.07647030 | 0.08183955 | 0.16391774 | -0.12089428 |
| Bverim1 | 0.869969591 | 0.79877953 | 0.78638200 | 0.85630221 | 0.31544206 |
| Bverim2 | 0.869841687 | 0.79935192 | 0.78719773 | 0.85637075 | 0.27335286 |
| Bverim3 | 0.920253616 | 0.93262410 | 0.91823720 | 0.90872244 | 0.33008661 |
| Bverim4 | -0.009492032 | -0.04774641 | -0.04517525 | -0.01739867 | -0.05266233 |
| Bverim5 | 0.683953518 | 0.48595120 | 0.47783139 | 0.52492996 | 0.23159420 |
| Bverim6 | 0.315030161 | 0.33465481 | 0.32892918 | 0.31471291 | 0.16024763 |
| Bverim7 | 0.941915719 | 0.95318184 | 0.93905844 | 0.91997503 | 0.33186949 |
| Bverim8 | 0.941816852 | 0.95328563 | 0.93923065 | 0.91981475 | 0.33039660 |
| Bverim9 | 0.885622200 | 0.90110742 | 0.89009100 | 0.85844043 | 0.31171257 |
| Bverim10 | 0.938082965 | 0.95163987 | 0.94038921 | 0.91169732 | 0.34848952 |
| Bverim11 | 0.748631188 | 0.67745834 | 0.66861838 | 0.72162413 | 0.22790281 |
| Bverim12 | 1.000000000 | 0.89701124 | 0.88400715 | 0.97574966 | 0.35169096 |
| Bverim13 | 0.897011243 | 1.00000000 | 0.98664163 | 0.92607457 | 0.32890754 |
| Bverim14 | 0.884007149 | 0.98664163 | 1.00000000 | 0.91479710 | 0.33943863 |
| Bverim15 | 0.975749658 | 0.92607457 | 0.91479710 | 1.00000000 | 0.34576997 |
| Bverim16 | 0.351690958 | 0.32890754 | 0.33943863 | 0.34576997 | 1.00000000 |

Çizelge B. 72 Çok İstemcili NetTcpBinding Bağlamalı Ortamda Elde Edilen İlişki Tabloları Test Verileri (devamı)

| | Bverim | Bverim1 | Bverim2 | Bverim3 | Bverim4 | Bverim5 |
|---|---|---|---|---|---|---|
| Bverim | 1.00000000 | 0.06024231 | 0.06868328 | 0.07477762 | 0.270522690 | -0.15285036 |
| Bverim1 | 0.06024231 | 1.00000000 | 0.99423191 | 0.81910489 | 0.036271975 | 0.70792404 |
| Bverim2 | 0.06868328 | 0.99423191 | 1.00000000 | 0.81770084 | 0.038032612 | 0.70604353 |
| Bverim3 | 0.07477762 | 0.81910489 | 0.81770084 | 1.00000000 | -0.177753420 | 0.60626843 |
| Bverim4 | 0.27052269 | 0.03627198 | 0.03803261 | -0.17775342 | 1.000000000 | 0.02896951 |
| Bverim5 | -0.15285036 | 0.70792404 | 0.70604353 | 0.60626843 | 0.028969507 | 1.00000000 |
| Bverim6 | -0.24381278 | 0.30146222 | 0.29906234 | 0.45767726 | -0.939002092 | 0.23157499 |
| Bverim7 | 0.08712598 | 0.85290860 | 0.85167776 | 0.95664689 | 0.009337178 | 0.65273950 |
| Bverim8 | 0.08582656 | 0.85204868 | 0.85187003 | 0.95610199 | 0.008977346 | 0.65312981 |
| Bverim9 | -0.02965074 | 0.80336222 | 0.80344202 | 0.90325041 | -0.020773642 | 0.63956736 |
| Bverim10 | 0.01630852 | 0.85012088 | 0.84943594 | 0.95499185 | -0.009860467 | 0.66656583 |
| Bverim11 | 0.26782150 | 0.64777659 | 0.64873189 | 0.65941361 | -0.034604703 | 0.52235640 |
| Bverim12 | 0.09947245 | 0.86996959 | 0.86984169 | 0.92025362 | -0.009492032 | 0.68395352 |
| Bverim13 | 0.07647030 | 0.79877953 | 0.79935192 | 0.93262410 | -0.047746414 | 0.48595120 |
| Bverim14 | 0.08183955 | 0.78638200 | 0.78719773 | 0.91823720 | -0.045175255 | 0.47783139 |
| Bverim15 | 0.16391774 | 0.85630221 | 0.85637075 | 0.90872244 | -0.017398671 | 0.52492996 |
| Bverim16 | -0.12089428 | 0.31544206 | 0.27335286 | 0.33008661 | -0.052662327 | 0.23159420 |

| | S | S1 | S2 | S3 | S4 | S5 | S6 |
|---|---|---|---|---|---|---|---|
| S | 1.0000000 | 0.9988920 | 0.9991067 | 0.9990332 | 0.9991265 | 0.9991616 | 0.9993075 |
| S1 | 0.9988920 | 1.0000000 | 0.9999120 | 0.9998868 | 0.9998393 | 0.9999091 | 0.9998783 |
| S2 | 0.9991067 | 0.9999120 | 1.0000000 | 0.9999264 | 0.9999615 | 0.9999862 | 0.9999739 |
| S3 | 0.9990332 | 0.9998868 | 0.9999264 | 1.0000000 | 0.9999151 | 0.9999333 | 0.9999243 |
| S4 | 0.9991265 | 0.9998393 | 0.9999615 | 0.9999151 | 1.0000000 | 0.9999503 | 0.9999502 |
| S5 | 0.9991616 | 0.9999091 | 0.9999862 | 0.9999333 | 0.9999503 | 1.0000000 | 0.9999774 |
| S6 | 0.9993075 | 0.9998783 | 0.9999739 | 0.9999243 | 0.9999502 | 0.9999774 | 1.0000000 |
| S7 | 0.9990660 | 0.9998426 | 0.9999120 | 0.9999200 | 0.9998914 | 0.9999108 | 0.9998945 |
| S8 | 0.9990133 | 0.9999596 | 0.9999694 | 0.9999138 | 0.9999057 | 0.9999603 | 0.9999364 |
| S9 | 0.9991210 | 0.9999093 | 0.9999706 | 0.9999275 | 0.9999132 | 0.9999749 | 0.9999484 |
| S10 | 0.9991195 | 0.9999316 | 0.9999835 | 0.9999298 | 0.9999435 | 0.9999786 | 0.9999618 |
| S11 | 0.9991422 | 0.9998972 | 0.9999773 | 0.9999025 | 0.9999286 | 0.9999756 | 0.9999595 |
| S12 | 0.9992755 | 0.9998877 | 0.9999771 | 0.9999269 | 0.9999392 | 0.9999778 | 0.9999800 |
| S13 | 0.9990757 | 0.9998655 | 0.9999331 | 0.9999311 | 0.9999214 | 0.9999388 | 0.9999175 |
| S14 | 0.9985571 | 0.9994816 | 0.9995125 | 0.9995266 | 0.9994494 | 0.9994473 | 0.9994757 |
| S15 | 0.9991821 | 0.9999066 | 0.9999836 | 0.9999180 | 0.9999410 | 0.9999864 | 0.9999719 |
| S16 | 0.9991696 | 0.9998813 | 0.9999779 | 0.9999276 | 0.9999272 | 0.9999783 | 0.9999614 |

| | S7 | S8 | S9 | S10 | S11 | S12 | S13 |
|---|---|---|---|---|---|---|---|
| S | 0.9990660 | 0.9990133 | 0.9991210 | 0.9991195 | 0.9991422 | 0.9992755 | 0.9990757 |
| S1 | 0.9998426 | 0.9999596 | 0.9999093 | 0.9999316 | 0.9998972 | 0.9998877 | 0.9998655 |
| S2 | 0.9999120 | 0.9999694 | 0.9999706 | 0.9999835 | 0.9999773 | 0.9999771 | 0.9999331 |
| S3 | 0.9999200 | 0.9999138 | 0.9999275 | 0.9999298 | 0.9999025 | 0.9999269 | 0.9999311 |
| S4 | 0.9998914 | 0.9999057 | 0.9999132 | 0.9999435 | 0.9999286 | 0.9999392 | 0.9999214 |
| S5 | 0.9999108 | 0.9999603 | 0.9999749 | 0.9999786 | 0.9999756 | 0.9999778 | 0.9999388 |
| S6 | 0.9998945 | 0.9999364 | 0.9999484 | 0.9999618 | 0.9999595 | 0.9999800 | 0.9999175 |
| S7 | 1.0000000 | 0.9998740 | 0.9999191 | 0.9999312 | 0.9998874 | 0.9999181 | 0.9998930 |
| S8 | 0.9998740 | 1.0000000 | 0.9999510 | 0.9999672 | 0.9999528 | 0.9999498 | 0.9999346 |
| S9 | 0.9999191 | 0.9999510 | 1.0000000 | 0.9999801 | 0.9999595 | 0.9999690 | 0.9999225 |
| S10 | 0.9999312 | 0.9999672 | 0.9999801 | 1.0000000 | 0.9999790 | 0.9999774 | 0.9999388 |
| S11 | 0.9998874 | 0.9999528 | 0.9999595 | 0.9999790 | 1.0000000 | 0.9999666 | 0.9999235 |
| S12 | 0.9999181 | 0.9999498 | 0.9999690 | 0.9999774 | 0.9999666 | 1.0000000 | 0.9999477 |
| S13 | 0.9998930 | 0.9999346 | 0.9999225 | 0.9999388 | 0.9999235 | 0.9999477 | 1.0000000 |
| S14 | 0.9994526 | 0.9995004 | 0.9995762 | 0.9995267 | 0.9995030 | 0.9995306 | 0.9994414 |
| S15 | 0.9999056 | 0.9999561 | 0.9999714 | 0.9999853 | 0.9999918 | 0.9999770 | 0.9999397 |
| S16 | 0.9999106 | 0.9999498 | 0.9999747 | 0.9999792 | 0.9999816 | 0.9999776 | 0.9999442 |

Çizelge B. 72 Çok İstemcili NetTcpBinding Bağlamalı Ortamda Elde Edilen İlişki Tabloları Test Verileri (devamı)

| | S14 | S15 | S16 |
|---|---|---|---|
| S | 0.9985571 | 0.9991821 | 0.9991696 |
| S1 | 0.9994816 | 0.9999066 | 0.9998813 |
| S2 | 0.9995125 | 0.9999836 | 0.9999779 |
| S3 | 0.9995266 | 0.9999180 | 0.9999276 |
| S4 | 0.9994494 | 0.9999410 | 0.9999272 |
| S5 | 0.9994473 | 0.9999864 | 0.9999783 |
| S6 | 0.9994757 | 0.9999719 | 0.9999614 |
| S7 | 0.9994526 | 0.9999056 | 0.9999106 |
| S8 | 0.9995004 | 0.9999561 | 0.9999498 |
| S9 | 0.9995762 | 0.9999714 | 0.9999747 |
| S10 | 0.9995267 | 0.9999853 | 0.9999792 |
| S11 | 0.9995030 | 0.9999918 | 0.9999816 |
| S12 | 0.9995306 | 0.9999770 | 0.9999776 |
| S13 | 0.9994414 | 0.9999397 | 0.9999442 |
| S14 | 1.0000000 | 0.9994803 | 0.9995024 |
| S15 | 0.9994803 | 1.0000000 | 0.9999891 |
| S16 | 0.9995024 | 0.9999891 | 1.0000000 |

**B - 4. 4 Çok İstemcili WsHttpBinding Bağlamalı Ortamda Elde Edilen İlişki Tablosu Değerleri**

Çizelge B. 73 Çok İstemcili WsHttpBinding Bağlamalı Ortamda Elde Edilen İlişki Tabloları Test Verileri

| | B | B1 | B2 | B3 | B4 | B5 | B6 |
|---|---|---|---|---|---|---|---|
| B | 1.0000000 | 0.7374200 | 0.7365960 | 0.7381883 | 0.7374152 | 0.7374254 | 0.7374179 |
| B1 | 0.7374200 | 1.0000000 | 0.9999857 | 0.9999859 | 1.0000000 | 1.0000000 | 1.0000000 |
| B2 | 0.7365960 | 0.9999857 | 1.0000000 | 0.9999735 | 0.9999858 | 0.9999858 | 0.9999858 |
| B3 | 0.7381883 | 0.9999859 | 0.9999735 | 1.0000000 | 0.9999857 | 0.9999858 | 0.9999857 |
| B4 | 0.7374152 | 1.0000000 | 0.9999858 | 0.9999857 | 1.0000000 | 1.0000000 | 1.0000000 |
| B5 | 0.7374254 | 1.0000000 | 0.9999858 | 0.9999858 | 1.0000000 | 1.0000000 | 1.0000000 |
| B6 | 0.7374179 | 1.0000000 | 0.9999858 | 0.9999857 | 1.0000000 | 1.0000000 | 1.0000000 |
| B7 | 0.7374201 | 1.0000000 | 0.9999857 | 0.9999858 | 1.0000000 | 1.0000000 | 1.0000000 |
| B8 | 0.7374014 | 1.0000000 | 0.9999857 | 0.9999858 | 1.0000000 | 1.0000000 | 1.0000000 |
| B9 | 0.7378271 | 0.9999918 | 0.9999770 | 0.9999782 | 0.9999917 | 0.9999918 | 0.9999918 |
| B10 | 0.7374284 | 1.0000000 | 0.9999857 | 0.9999857 | 1.0000000 | 1.0000000 | 1.0000000 |
| B11 | 0.7374226 | 1.0000000 | 0.9999858 | 0.9999857 | 1.0000000 | 1.0000000 | 1.0000000 |
| B12 | 0.7374304 | 1.0000000 | 0.9999857 | 0.9999858 | 1.0000000 | 1.0000000 | 1.0000000 |
| B13 | 0.7374108 | 1.0000000 | 0.9999857 | 0.9999857 | 1.0000000 | 1.0000000 | 1.0000000 |
| B14 | 0.7373706 | 0.9999997 | 0.9999856 | 0.9999853 | 0.9999997 | 0.9999997 | 0.9999997 |
| B15 | 0.7374223 | 1.0000000 | 0.9999858 | 0.9999857 | 1.0000000 | 1.0000000 | 1.0000000 |
| B16 | 0.7374621 | 0.9999999 | 0.9999857 | 0.9999856 | 0.9999998 | 0.9999998 | 0.9999998 |

Çizelge B. 73 Çok İstemcili WsHttpBinding Bağlamalı Ortamda Elde Edilen İlişki Tabloları Test Verileri (devamı)

| | B7 | B8 | B9 | B10 | B11 | B12 | B13 |
|---|---|---|---|---|---|---|---|
| B | 0.7374201 | 0.7374014 | 0.7378271 | 0.7374284 | 0.7374226 | 0.7374304 | 0.7374108 |
| B1 | 1.0000000 | 1.0000000 | 0.9999918 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 |
| B2 | 0.9999857 | 0.9999857 | 0.9999770 | 0.9999857 | 0.9999858 | 0.9999857 | 0.9999857 |
| B3 | 0.9999858 | 0.9999858 | 0.9999782 | 0.9999857 | 0.9999857 | 0.9999858 | 0.9999857 |
| B4 | 1.0000000 | 1.0000000 | 0.9999917 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 |
| B5 | 1.0000000 | 1.0000000 | 0.9999918 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 |
| B6 | 1.0000000 | 1.0000000 | 0.9999918 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 |
| B7 | 1.0000000 | 1.0000000 | 0.9999917 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 |
| B8 | 1.0000000 | 1.0000000 | 0.9999917 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 |
| B9 | 0.9999917 | 0.9999917 | 1.0000000 | 0.9999918 | 0.9999919 | 0.9999918 | 0.9999919 |
| B10 | 1.0000000 | 1.0000000 | 0.9999918 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 |
| B11 | 1.0000000 | 1.0000000 | 0.9999919 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 |
| B12 | 1.0000000 | 1.0000000 | 0.9999918 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 |
| B13 | 1.0000000 | 1.0000000 | 0.9999919 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 |
| B14 | 0.9999997 | 0.9999997 | 0.9999917 | 0.9999997 | 0.9999997 | 0.9999997 | 0.9999997 |
| B15 | 1.0000000 | 1.0000000 | 0.9999919 | 1.0000000 | 1.0000000 | 1.0000000 | 1.0000000 |
| B16 | 0.9999998 | 0.9999998 | 0.9999926 | 0.9999999 | 0.9999999 | 0.9999999 | 0.9999998 |

| | B14 | B15 | B16 |
|---|---|---|---|
| B | 0.7373706 | 0.7374223 | 0.7374621 |
| B1 | 0.9999997 | 1.0000000 | 0.9999999 |
| B2 | 0.9999856 | 0.9999858 | 0.9999857 |
| B3 | 0.9999853 | 0.9999857 | 0.9999856 |
| B4 | 0.9999997 | 1.0000000 | 0.9999998 |
| B5 | 0.9999997 | 1.0000000 | 0.9999998 |
| B6 | 0.9999997 | 1.0000000 | 0.9999998 |
| B7 | 0.9999997 | 1.0000000 | 0.9999998 |
| B8 | 0.9999997 | 1.0000000 | 0.9999998 |
| B9 | 0.9999917 | 0.9999919 | 0.9999926 |
| B10 | 0.9999997 | 1.0000000 | 0.9999999 |
| B11 | 0.9999997 | 1.0000000 | 0.9999999 |
| B12 | 0.9999997 | 1.0000000 | 0.9999999 |
| B13 | 0.9999997 | 1.0000000 | 0.9999998 |
| B14 | 1.0000000 | 0.9999998 | 0.9999995 |
| B15 | 0.9999998 | 1.0000000 | 0.9999998 |
| B16 | 0.9999995 | 0.9999998 | 1.0000000 |

| | Bverim | Bverim1 | Bverim2 | Bverim3 | Bverim4 |
|---|---|---|---|---|---|
| Bverim | 1.00000000 | 0.22314699 | -0.04873521 | -0.01837169 | -0.97687652 |
| Bverim1 | 0.22314699 | 1.00000000 | 0.37092251 | 0.11604190 | -0.17779271 |
| Bverim2 | -0.04873521 | 0.37092251 | 1.00000000 | 0.09615294 | 0.02866538 |
| Bverim3 | -0.01837169 | 0.11604190 | 0.09615294 | 1.00000000 | 0.05492855 |
| Bverim4 | -0.97687652 | -0.17779271 | 0.02866538 | 0.05492855 | 1.00000000 |
| Bverim5 | 0.17488118 | 0.98375241 | 0.38404944 | 0.04293558 | -0.13091066 |
| Bverim6 | 0.17198344 | 0.98361406 | 0.38500128 | 0.03584630 | -0.12974224 |
| Bverim7 | -0.99001299 | -0.18268751 | 0.03174280 | 0.05405658 | 0.99593067 |
| Bverim8 | -0.93688455 | -0.17305091 | 0.01774365 | 0.06823415 | 0.93991205 |
| Bverim9 | 0.36814385 | 0.89574030 | 0.28977950 | 0.09857633 | -0.30124887 |
| Bverim10 | 0.26266056 | -0.07515297 | -0.08721936 | 0.05553068 | -0.22506189 |
| Bverim11 | 0.14583531 | 0.53979857 | 0.17009625 | 0.18821232 | -0.09089986 |
| Bverim12 | 0.14262535 | 0.53862245 | 0.15831532 | 0.19579400 | -0.08310798 |
| Bverim13 | 0.31608486 | 0.97837380 | 0.35341495 | 0.07814707 | -0.26810636 |
| Bverim14 | 0.32566213 | 0.97325611 | 0.35612916 | 0.07132968 | -0.28051980 |
| Bverim15 | -0.25895430 | 0.43997147 | 0.19111898 | 0.18657470 | 0.30880038 |
| Bverim16 | -0.23800636 | 0.42335952 | 0.18205427 | 0.17480835 | 0.27793436 |

Çizelge B. 73 Çok İstemcili WsHttpBinding Bağlamalı Ortamda Elde Edilen İlişki Tabloları Test Verileri (devamı)

| | Bverim5 | Bverim6 | Bverim7 | Bverim8 | Bverim9 | Bverim10 |
|---|---|---|---|---|---|---|
| Bverim | 0.17488118 | 0.1719834 | -0.99001299 | -0.93688455 | 0.36814385 | 0.26266056 |
| Bverim1 | 0.98375241 | 0.9836141 | -0.18268751 | -0.17305091 | 0.89574030 | -0.07515297 |
| Bverim2 | 0.38404944 | 0.3850013 | 0.03174280 | 0.01774365 | 0.28977950 | -0.08721936 |
| Bverim3 | 0.04293558 | 0.0358463 | 0.05405658 | 0.06823415 | 0.09857633 | 0.05553068 |
| Bverim4 | -0.13091066 | -0.1297422 | 0.99593067 | 0.93991205 | -0.30124887 | -0.22506189 |
| Bverim5 | 1.00000000 | 0.9996810 | -0.13583496 | -0.12828143 | 0.88285647 | -0.11228473 |
| Bverim6 | 0.99968105 | 1.0000000 | -0.13378524 | -0.12795365 | 0.88211679 | -0.11363042 |
| Bverim7 | -0.13583496 | -0.1337852 | 1.00000000 | 0.94174249 | -0.31416814 | -0.23286174 |
| Bverim8 | -0.12828143 | -0.1279536 | 0.94174249 | 1.00000000 | -0.30326219 | -0.22233711 |
| Bverim9 | 0.88285647 | 0.8821168 | -0.31416814 | -0.30326219 | 1.00000000 | 0.05653736 |
| Bverim10 | -0.11228473 | -0.1136304 | -0.23286174 | -0.22233711 | 0.05653736 | 1.00000000 |
| Bverim11 | 0.48550971 | 0.4849199 | -0.10115805 | -0.09779060 | 0.55776967 | 0.44683081 |
| Bverim12 | 0.48457199 | 0.4837631 | -0.09485131 | -0.09291845 | 0.55433320 | 0.44643564 |
| Bverim13 | 0.96770452 | 0.9673839 | -0.27397340 | -0.26092009 | 0.92297298 | 0.02912984 |
| Bverim14 | 0.96273328 | 0.9617725 | -0.28624402 | -0.27009185 | 0.91864492 | 0.03106157 |
| Bverim15 | 0.40731616 | 0.4075157 | 0.30129856 | 0.28568673 | 0.39906748 | 0.33082229 |
| Bverim16 | 0.39055754 | 0.3905041 | 0.27564807 | 0.24484363 | 0.42187500 | 0.32065386 |

| | Bverim11 | Bverim12 | Bverim13 | Bverim14 | Bverim15 | Bverim16 |
|---|---|---|---|---|---|---|
| Bverim | 0.14583531 | 0.14262535 | 0.31608486 | 0.32566213 | -0.2589543 | -0.2380064 |
| Bverim1 | 0.53979857 | 0.53862245 | 0.97837380 | 0.97325611 | 0.4399715 | 0.4233595 |
| Bverim2 | 0.17009625 | 0.15831532 | 0.35341495 | 0.35612916 | 0.1911190 | 0.1820543 |
| Bverim3 | 0.18821232 | 0.19579400 | 0.07814707 | 0.07132968 | 0.1865747 | 0.1748083 |
| Bverim4 | -0.09089986 | -0.08310798 | -0.26810636 | -0.28051980 | 0.3088004 | 0.2779344 |
| Bverim5 | 0.48550971 | 0.48457199 | 0.96770452 | 0.96273328 | 0.4073162 | 0.3905575 |
| Bverim6 | 0.48491992 | 0.48376315 | 0.96738393 | 0.96177252 | 0.4075157 | 0.3905041 |
| Bverim7 | -0.10115805 | -0.09485131 | -0.27397340 | -0.28624402 | 0.3012986 | 0.2756481 |
| Bverim8 | -0.09779060 | -0.09291845 | -0.26092009 | -0.27009185 | 0.2856867 | 0.2448436 |
| Bverim9 | 0.55776967 | 0.55433320 | 0.92297298 | 0.91864492 | 0.3990675 | 0.4218750 |
| Bverim10 | 0.44683081 | 0.44643564 | 0.02912984 | 0.03106157 | 0.3308223 | 0.3206539 |
| Bverim11 | 1.00000000 | 0.99658578 | 0.59050257 | 0.58533745 | 0.9068870 | 0.8703265 |
| Bverim12 | 0.99658578 | 1.00000000 | 0.58888488 | 0.58464090 | 0.9073227 | 0.8696862 |
| Bverim13 | 0.59050257 | 0.58888488 | 1.00000000 | 0.99535294 | 0.4521665 | 0.4345211 |
| Bverim14 | 0.58533745 | 0.58464090 | 0.99535294 | 1.00000000 | 0.4474764 | 0.4258393 |
| Bverim15 | 0.90688700 | 0.90732268 | 0.45216646 | 0.44747643 | 1.0000000 | 0.9318745 |
| Bverim16 | 0.87032654 | 0.86968621 | 0.43452112 | 0.42583931 | 0.9318745 | 1.0000000 |

| | S | S1 | S2 | S3 | S4 | S5 | S6 |
|---|---|---|---|---|---|---|---|
| S | 1.0000000 | 0.9996851 | 0.9995996 | 0.9996810 | 0.9997072 | 0.9996620 | 0.9996695 |
| S1 | 0.9996851 | 1.0000000 | 0.9999490 | 0.9999832 | 0.9999854 | 0.9999807 | 0.9999909 |
| S2 | 0.9995996 | 0.9999490 | 1.0000000 | 0.9999640 | 0.9999303 | 0.9999579 | 0.9999493 |
| S3 | 0.9996810 | 0.9999832 | 0.9999640 | 1.0000000 | 0.9999813 | 0.9999857 | 0.9999904 |
| S4 | 0.9997072 | 0.9999854 | 0.9999303 | 0.9999813 | 1.0000000 | 0.9999723 | 0.9999851 |
| S5 | 0.9996620 | 0.9999807 | 0.9999579 | 0.9999857 | 0.9999723 | 1.0000000 | 0.9999896 |
| S6 | 0.9996695 | 0.9999909 | 0.9999493 | 0.9999904 | 0.9999851 | 0.9999896 | 1.0000000 |
| S7 | 0.9996764 | 0.9999848 | 0.9999481 | 0.9999862 | 0.9999835 | 0.9999849 | 0.9999905 |
| S8 | 0.9997252 | 0.9998670 | 0.9997792 | 0.9998639 | 0.9998819 | 0.9998170 | 0.9998547 |
| S9 | 0.9997533 | 0.9998806 | 0.9997364 | 0.9998194 | 0.9998706 | 0.9998148 | 0.9998530 |
| S10 | 0.9996894 | 0.9999941 | 0.9999383 | 0.9999833 | 0.9999855 | 0.9999830 | 0.9999891 |
| S11 | 0.9996868 | 0.9999857 | 0.9999720 | 0.9999902 | 0.9999827 | 0.9999811 | 0.9999837 |
| S12 | 0.9995933 | 0.9999793 | 0.9999537 | 0.9999760 | 0.9999719 | 0.9999701 | 0.9999799 |
| S13 | 0.9996612 | 0.9999856 | 0.9999694 | 0.9999936 | 0.9999744 | 0.9999818 | 0.9999860 |
| S14 | 0.9995307 | 0.9998118 | 0.9999006 | 0.9998549 | 0.9998057 | 0.9998568 | 0.9998446 |
| S15 | 0.9997613 | 0.9998476 | 0.9997501 | 0.9998277 | 0.9998648 | 0.9998330 | 0.9998544 |
| S16 | 0.9996741 | 0.9999796 | 0.9999744 | 0.9999873 | 0.9999724 | 0.9999855 | 0.9999847 |

Çizelge B. 73 Çok İstemcili WsHttpBinding Bağlamalı Ortamda Elde Edilen İlişki Tabloları Test Verileri (devamı)

| | S7 | S8 | S9 | S10 | S11 | S12 | S13 |
|---|---|---|---|---|---|---|---|
| S | 0.9996764 | 0.9997252 | 0.9997533 | 0.9996894 | 0.9996868 | 0.9995933 | 0.9996612 |
| S1 | 0.9999848 | 0.9998670 | 0.9998806 | 0.9999941 | 0.9999857 | 0.9999793 | 0.9999856 |
| S2 | 0.9999481 | 0.9997792 | 0.9997364 | 0.9999383 | 0.9999720 | 0.9999537 | 0.9999694 |
| S3 | 0.9999862 | 0.9998639 | 0.9998194 | 0.9999833 | 0.9999902 | 0.9999760 | 0.9999936 |
| S4 | 0.9999835 | 0.9998819 | 0.9998706 | 0.9999855 | 0.9999827 | 0.9999719 | 0.9999744 |
| S5 | 0.9999849 | 0.9998170 | 0.9998148 | 0.9999830 | 0.9999811 | 0.9999701 | 0.9999818 |
| S6 | 0.9999905 | 0.9998547 | 0.9998530 | 0.9999891 | 0.9999837 | 0.9999799 | 0.9999860 |
| S7 | 1.0000000 | 0.9998542 | 0.9998315 | 0.9999840 | 0.9999876 | 0.9999784 | 0.9999855 |
| S8 | 0.9998542 | 1.0000000 | 0.9998317 | 0.9998716 | 0.9998571 | 0.9998501 | 0.9998501 |
| S9 | 0.9998315 | 0.9998317 | 1.0000000 | 0.9998805 | 0.9998354 | 0.9998061 | 0.9998223 |
| S10 | 0.9999840 | 0.9998716 | 0.9998805 | 1.0000000 | 0.9999812 | 0.9999815 | 0.9999843 |
| S11 | 0.9999876 | 0.9998571 | 0.9998354 | 0.9999812 | 1.0000000 | 0.9999737 | 0.9999917 |
| S12 | 0.9999784 | 0.9998501 | 0.9998061 | 0.9999815 | 0.9999737 | 1.0000000 | 0.9999801 |
| S13 | 0.9999855 | 0.9998501 | 0.9998223 | 0.9999843 | 0.9999917 | 0.9999801 | 1.0000000 |
| S14 | 0.9998283 | 0.9996430 | 0.9995662 | 0.9998027 | 0.9998510 | 0.9998159 | 0.9998435 |
| S15 | 0.9998367 | 0.9997933 | 0.9998747 | 0.9998573 | 0.9998273 | 0.9998042 | 0.9998094 |
| S16 | 0.9999788 | 0.9998371 | 0.9998181 | 0.9999801 | 0.9999858 | 0.9999708 | 0.9999855 |

| | S14 | S15 | S16 |
|---|---|---|---|
| S | 0.9995307 | 0.9997613 | 0.9996741 |
| S1 | 0.9998118 | 0.9998476 | 0.9999796 |
| S2 | 0.9999006 | 0.9997501 | 0.9999744 |
| S3 | 0.9998549 | 0.9998277 | 0.9999873 |
| S4 | 0.9998057 | 0.9998648 | 0.9999724 |
| S5 | 0.9998568 | 0.9998330 | 0.9999855 |
| S6 | 0.9998446 | 0.9998544 | 0.9999847 |
| S7 | 0.9998283 | 0.9998367 | 0.9999788 |
| S8 | 0.9996430 | 0.9997933 | 0.9998371 |
| S9 | 0.9995662 | 0.9998747 | 0.9998181 |
| S10 | 0.9998027 | 0.9998573 | 0.9999801 |
| S11 | 0.9998510 | 0.9998273 | 0.9999858 |
| S12 | 0.9998159 | 0.9998042 | 0.9999708 |
| S13 | 0.9998435 | 0.9998094 | 0.9999855 |
| S14 | 1.0000000 | 0.9997470 | 0.9998935 |
| S15 | 0.9997470 | 1.0000000 | 0.9998388 |
| S16 | 0.9998935 | 0.9998388 | 1.0000000 |

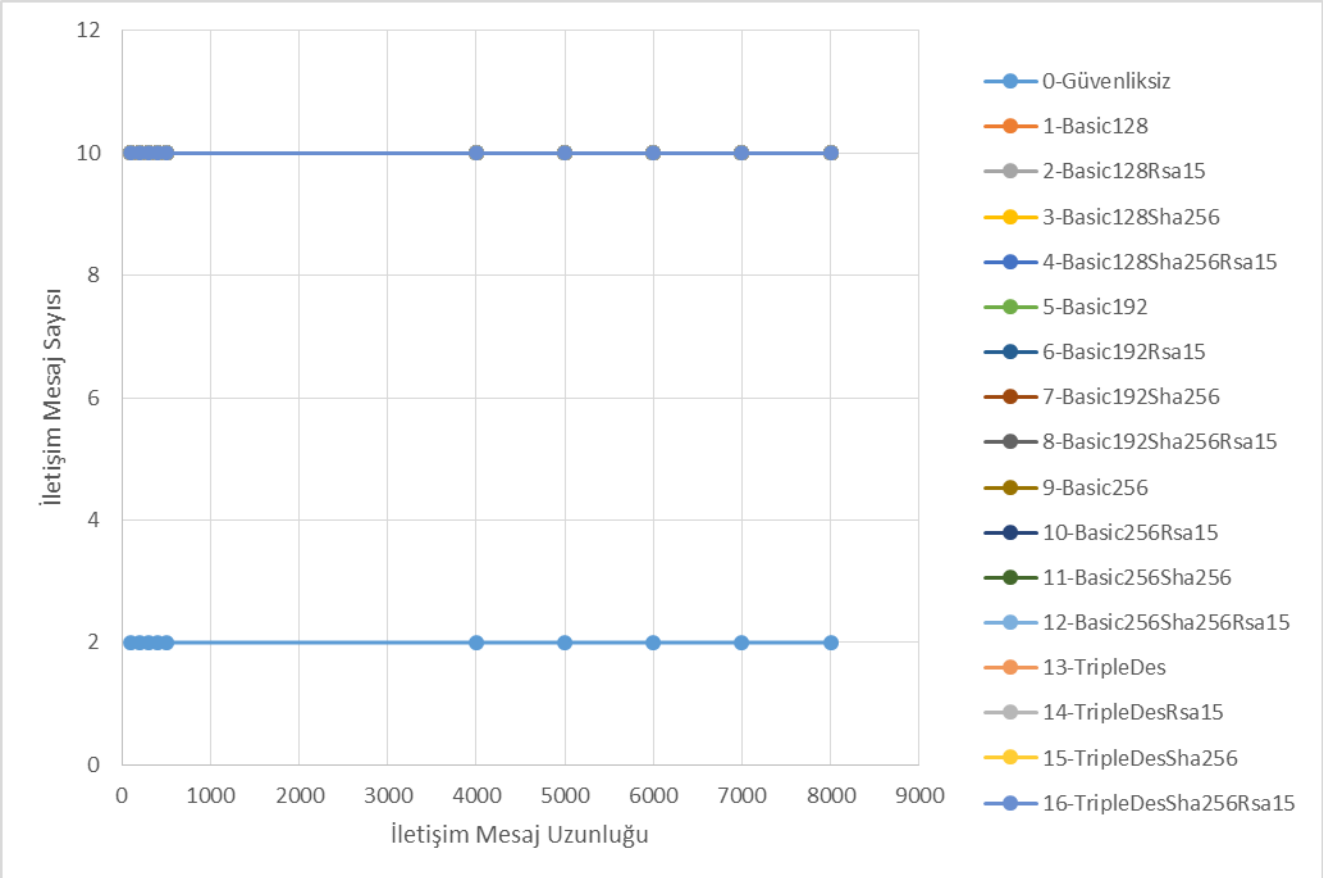
# TEST ORTAMINDA ELDE EDİLEN VERİLER KULLANILARAK OLUŞTURULAN YORUM GRAFİKLERİ

Bu bölümde EK-B'de elde edilen verilerin yorumlanması için oluşturulan grafikler ve yorumlara yer verilmiştir.

## C -1 Tek İstemcili Değişken Kelime Katarlı Test İşlemi Verileri

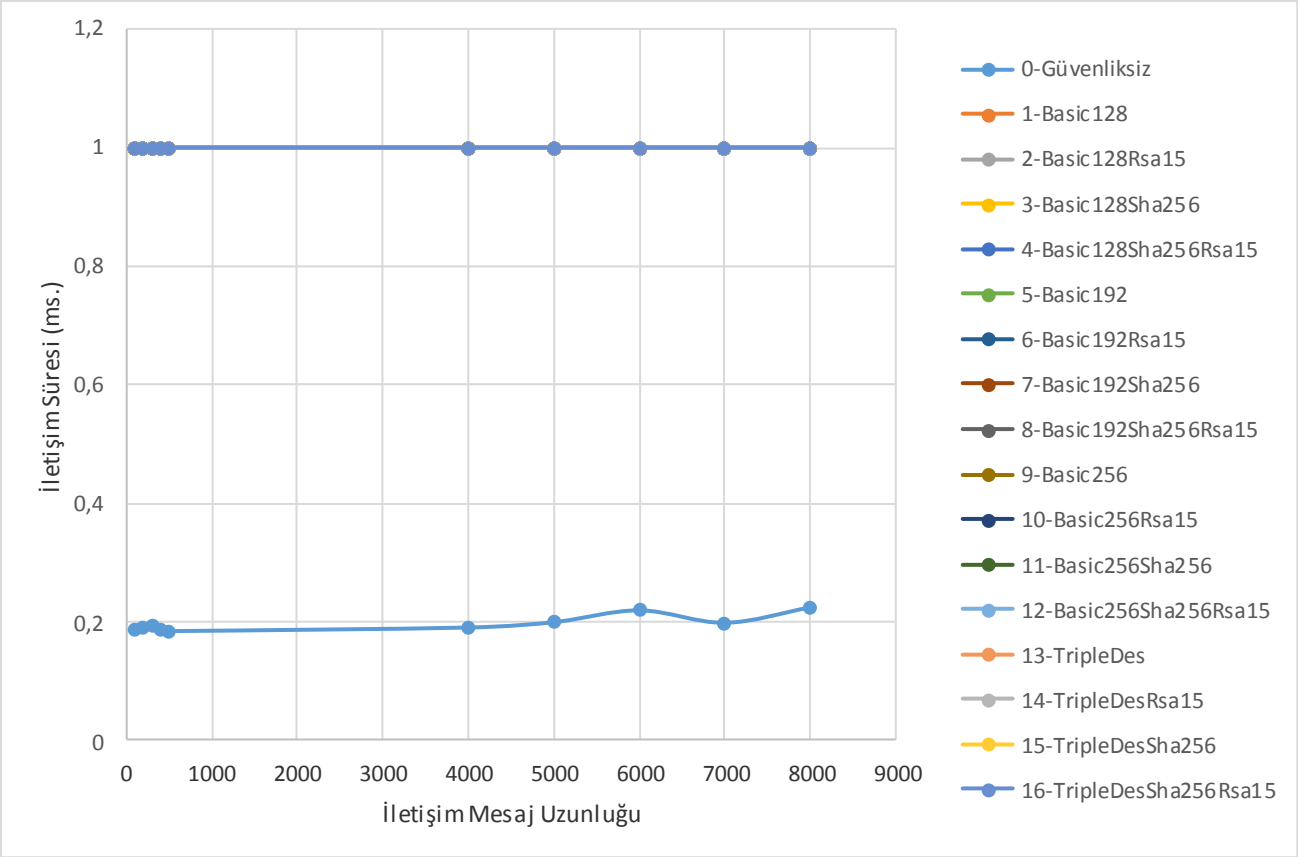
### C -1. 1 NetTcpBinding Bağlama Kullanılarak Elde Edilen Veriler

Çizelge C. 1 TCP protokolü kullanılarak çeşitli şifreleme algoritmaları ile iletişim mesaj uzunluğu – iletişim sayısı ilişkisi çizelgesi



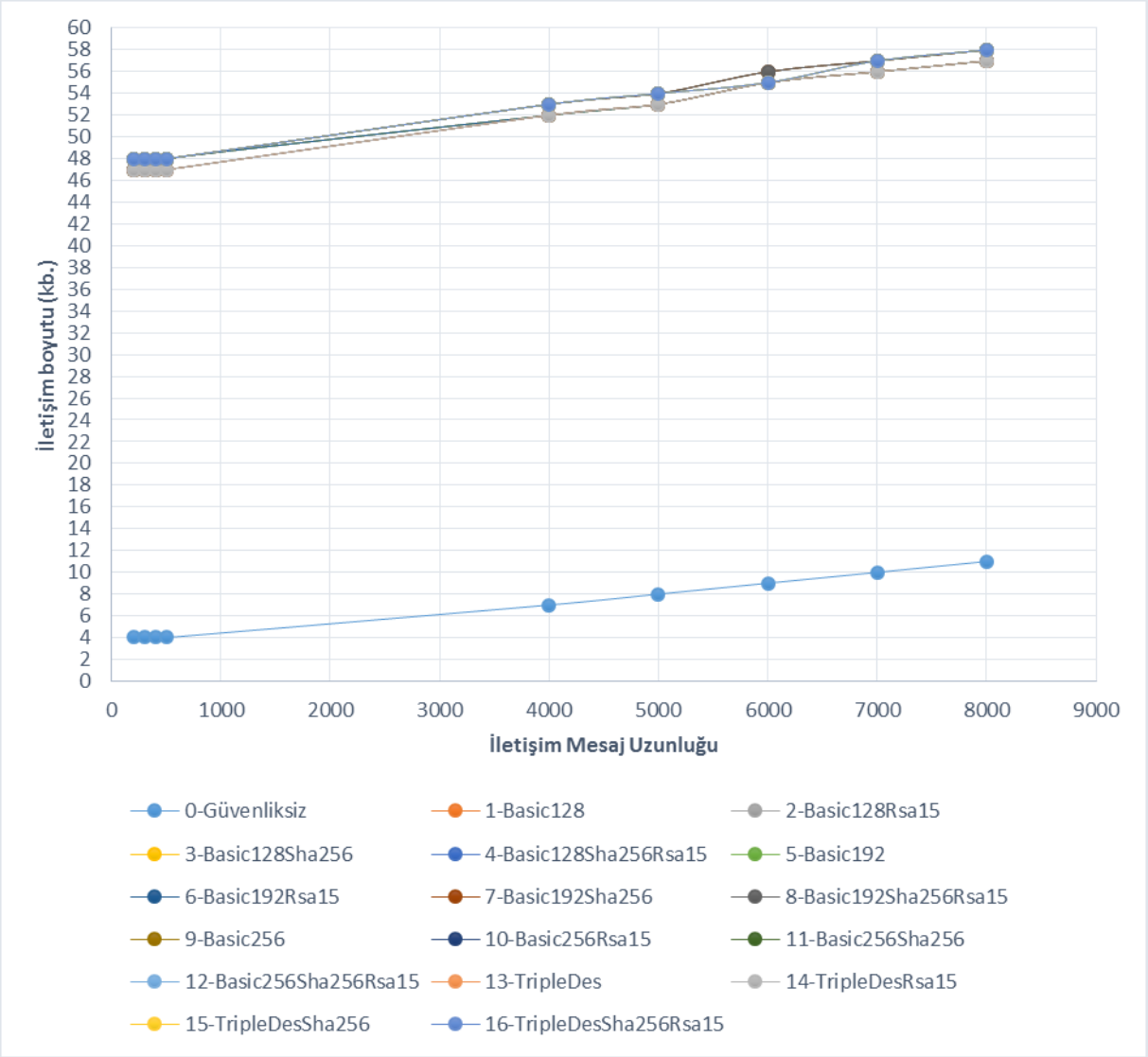
Çizelge C. 1 incelendiği zaman güvenliksiz ortamda iletilen mesajların düz metin durumlarıyla iletişim yapıldığında iletişim mesaj sayısı sadece iki iken, şifrelenmiş mesajlarla iletişim yapılmak istenildiği zaman iletişim sayısı beş kat artarak on olmuştur.

Çizelge C. 2 TCP protokolü kullanılarak çeşitli şifreleme algoritmaları ile iletişim mesaj uzunluğu –iletişim süresi (ms.) ilişkisi çizelgesi



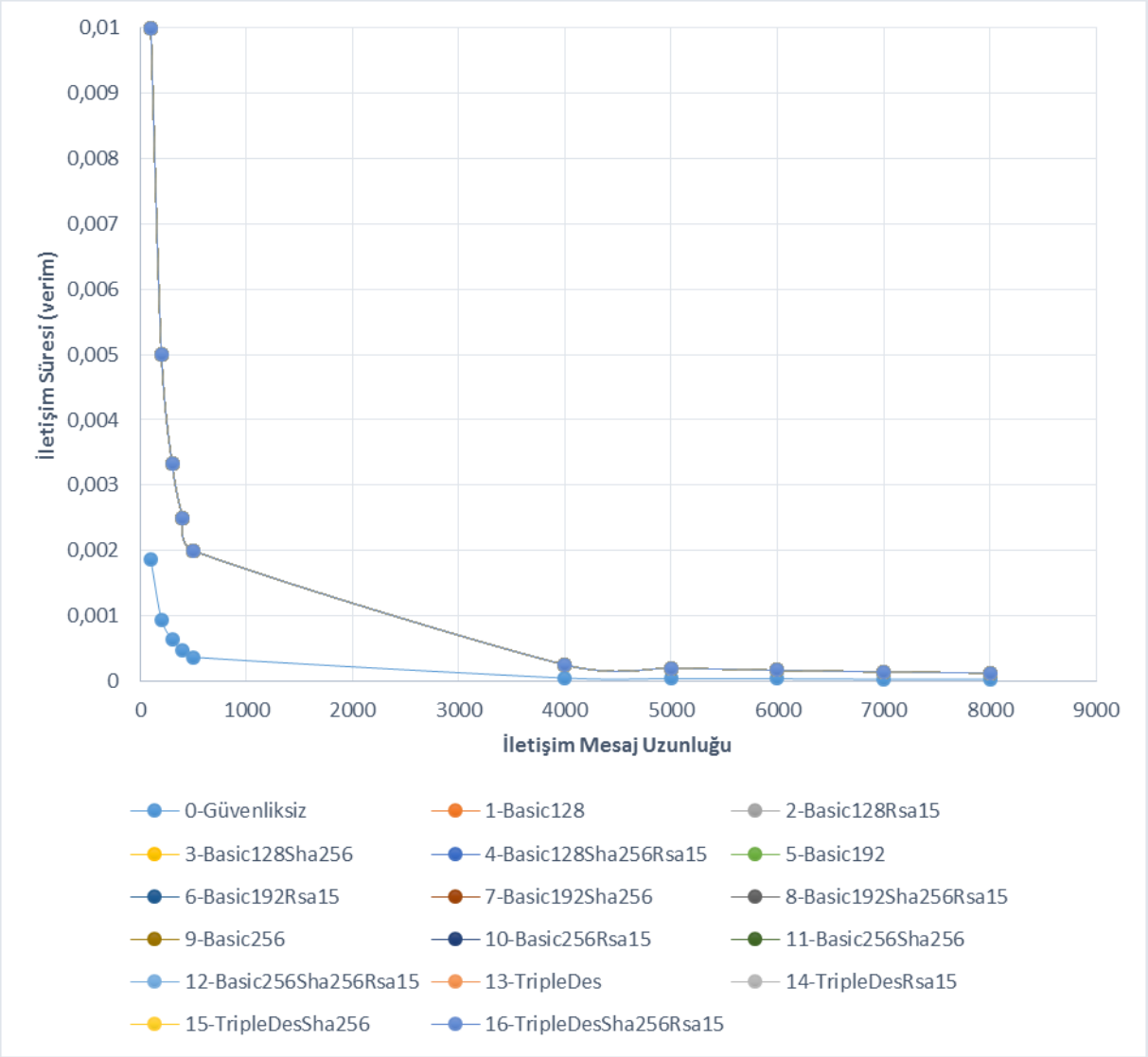

Çizelge C. 2 incelendiğinde güvenliksiz ortamda iletişim süresi 0. 2 milisaniye civarında olmasına karşın diğer şifreleme algoritmalarıyla oluşturulmuş ortamlarda süre 1 mili saniye civarına çıkmıştır.

Çizelge C. 3 TCP protokolü kullanılarak çeşitli şifreleme algoritmaları ile iletişim mesaj uzunluğu –iletişim boyutu (kb.) ilişkisi çizelgesi



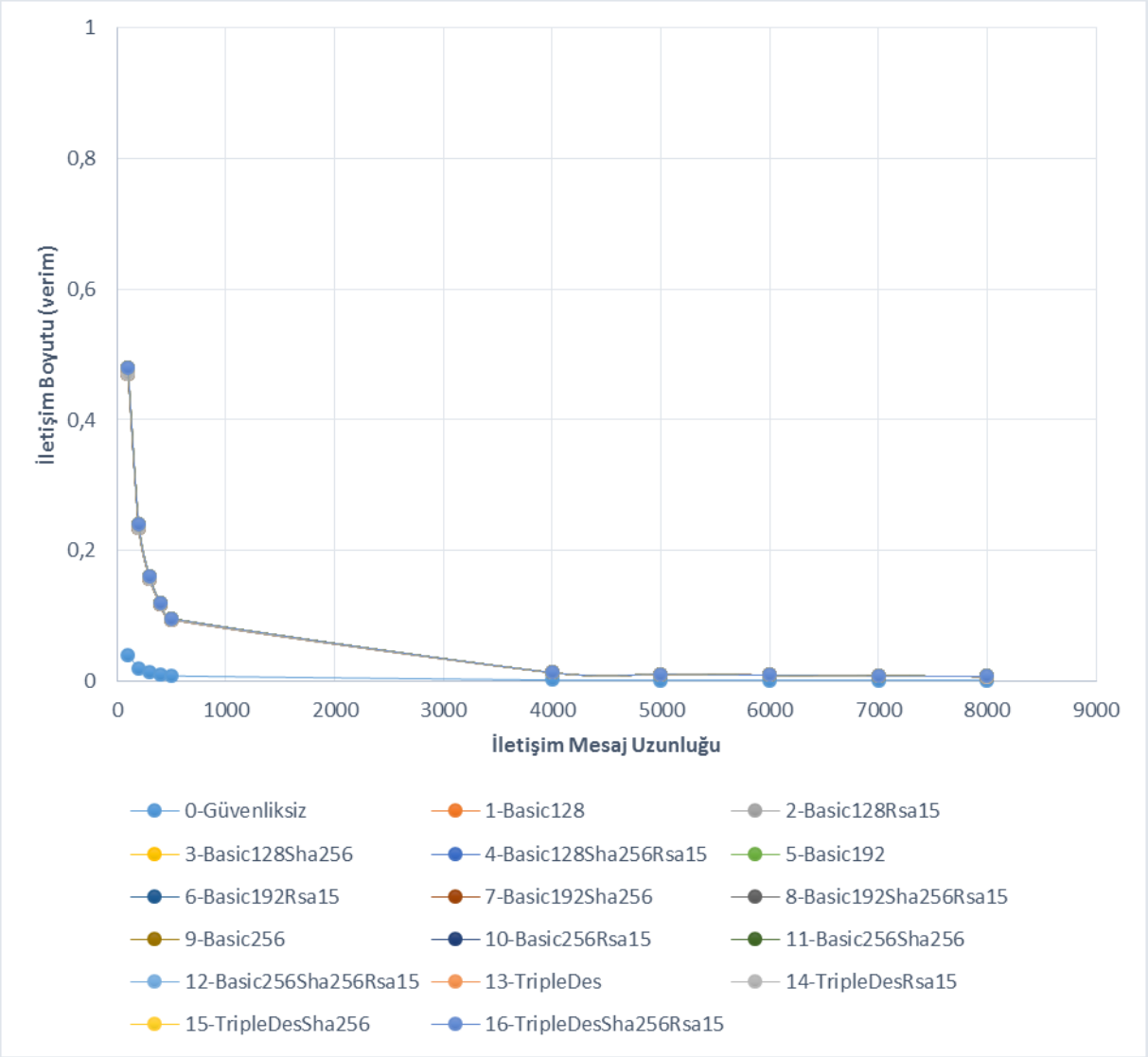

Çizelge C. 3 incelendiğinde güvenliksiz ve simetrik şifreleme algoritmalarının değişen mesaj boyutuna karşın iletişim boyutu düzenli olarak artmasına karşın asimetrik şifreleme algoritmasıyla oluşturulmuş ortamlarda ise düzenin çok az bozulduğu gözlemlenmiştir.

Çizelge C. 4 TCP protokolü kullanılarak çeşitli şifreleme algoritmaları ile iletişim mesaj uzunluğu –iletişim süresi (verim) ilişkisi çizelgesi




Çizelge C.4 incelendiğinden SOAP protokolünde iletişim mesajının karakter sayısının arttırılması ortamın karakter başına iletişim süresinde daha da verimli olmasına sağlamıştır. Fakat bu verim iletişim mesajının karakter sayısının artılımasınız bu ortamdaki verimliliği giderek azalmıştır.

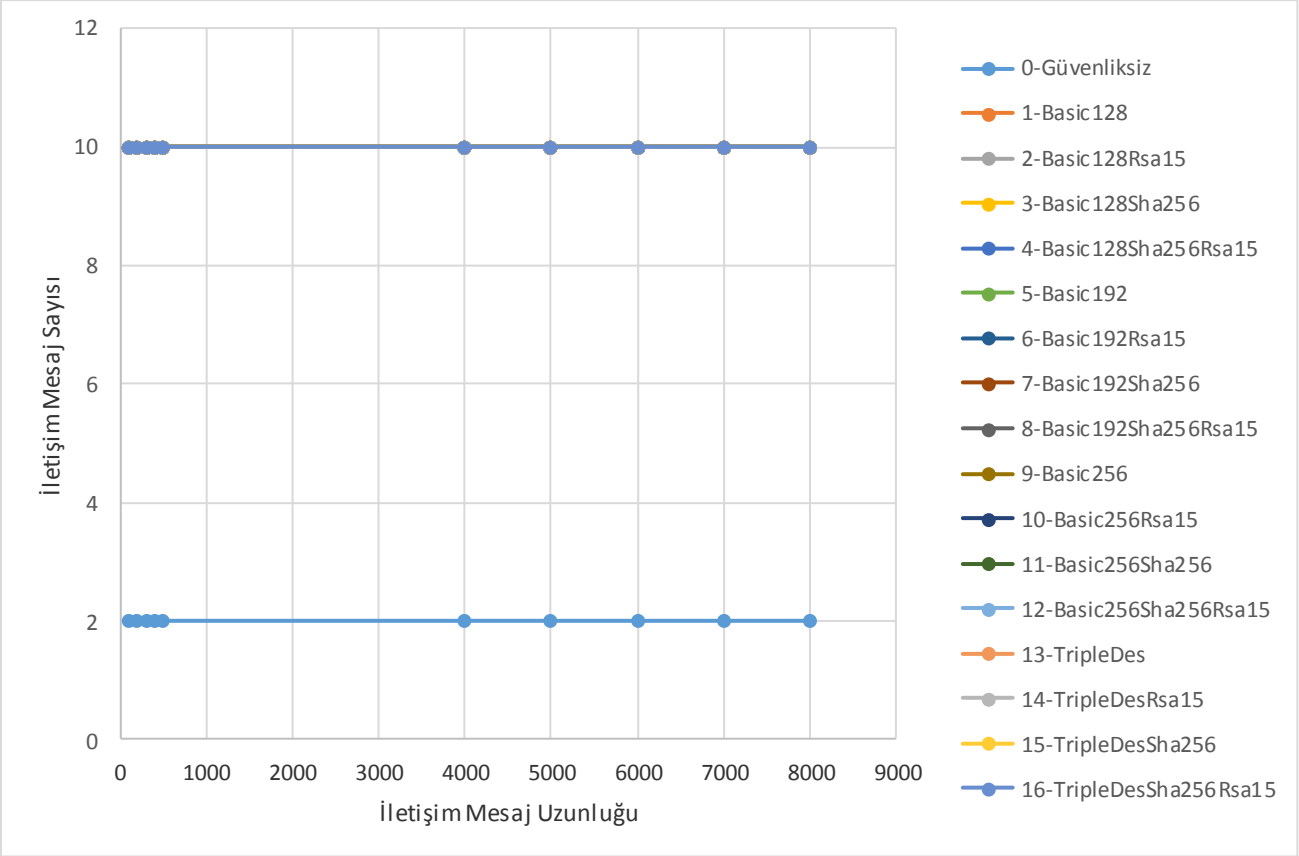
Çizelge C. 5 TCP protokolü kullanılarak çeşitli şifreleme algoritmaları ile iletişim mesaj uzunluğu –iletişim boyutu (verim) ilişkisi çizelgesi




Çizelge C.4 incelendiğinden SOAP protokolünde iletişim mesajının karakter sayısının arttırılması ortamın karakter başına iletişim boyutunun daha da verimli olmasına sağlamıştır. Fakat bu verim iletişim mesajının karakter sayısının artılımasınız bu ortamdaki verimliliği giderek azalmıştır.

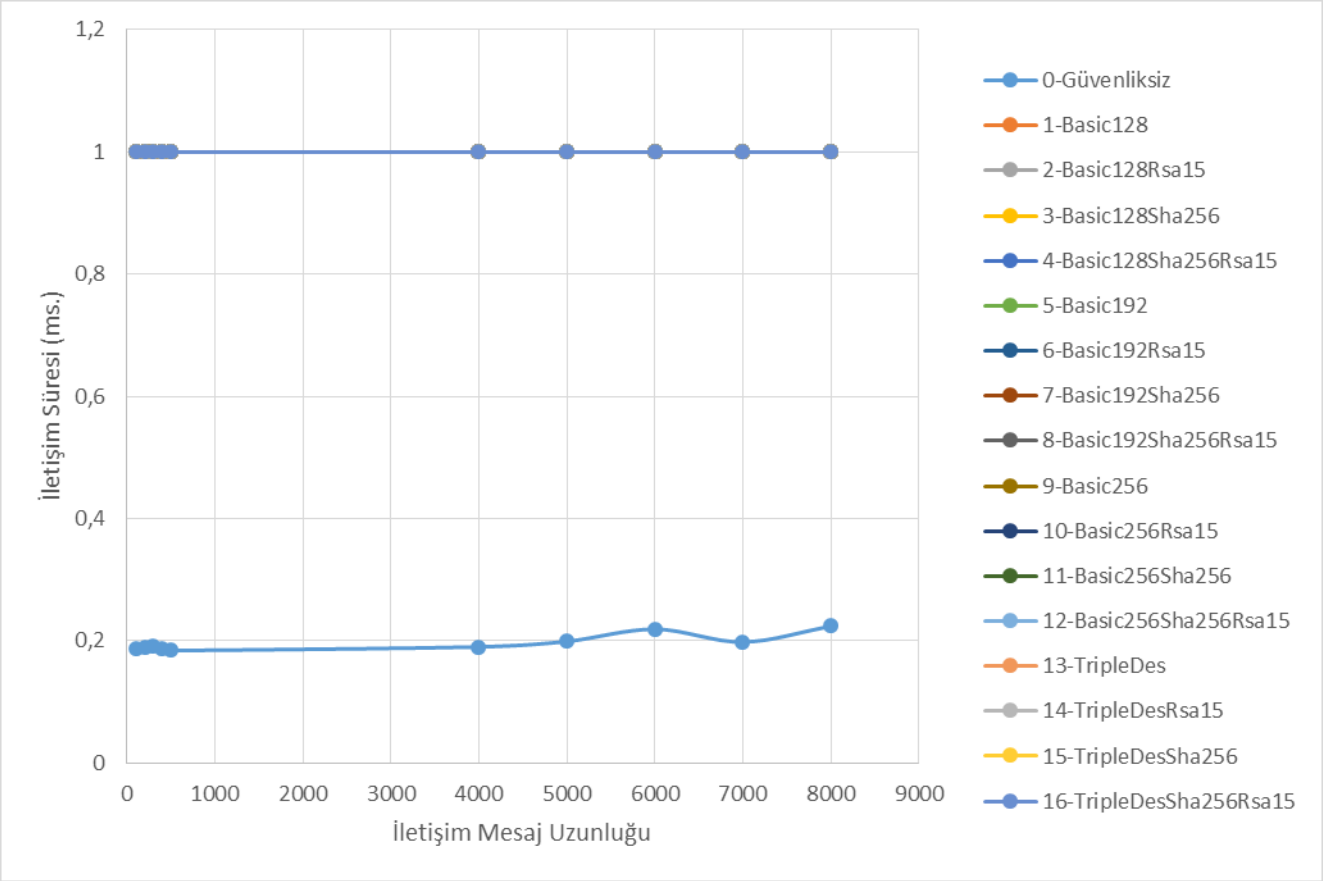
### C - 1. 2 WsHttp Binding Bağlama Kullanılarak Elde Edilen Veriler

Çizelge C. 6 HTTP protokolü kullanılarak çeşitli şifreleme algoritmaları ile iletişim mesaj uzunluğu – iletişim sayısı ilişkisi çizelgesi



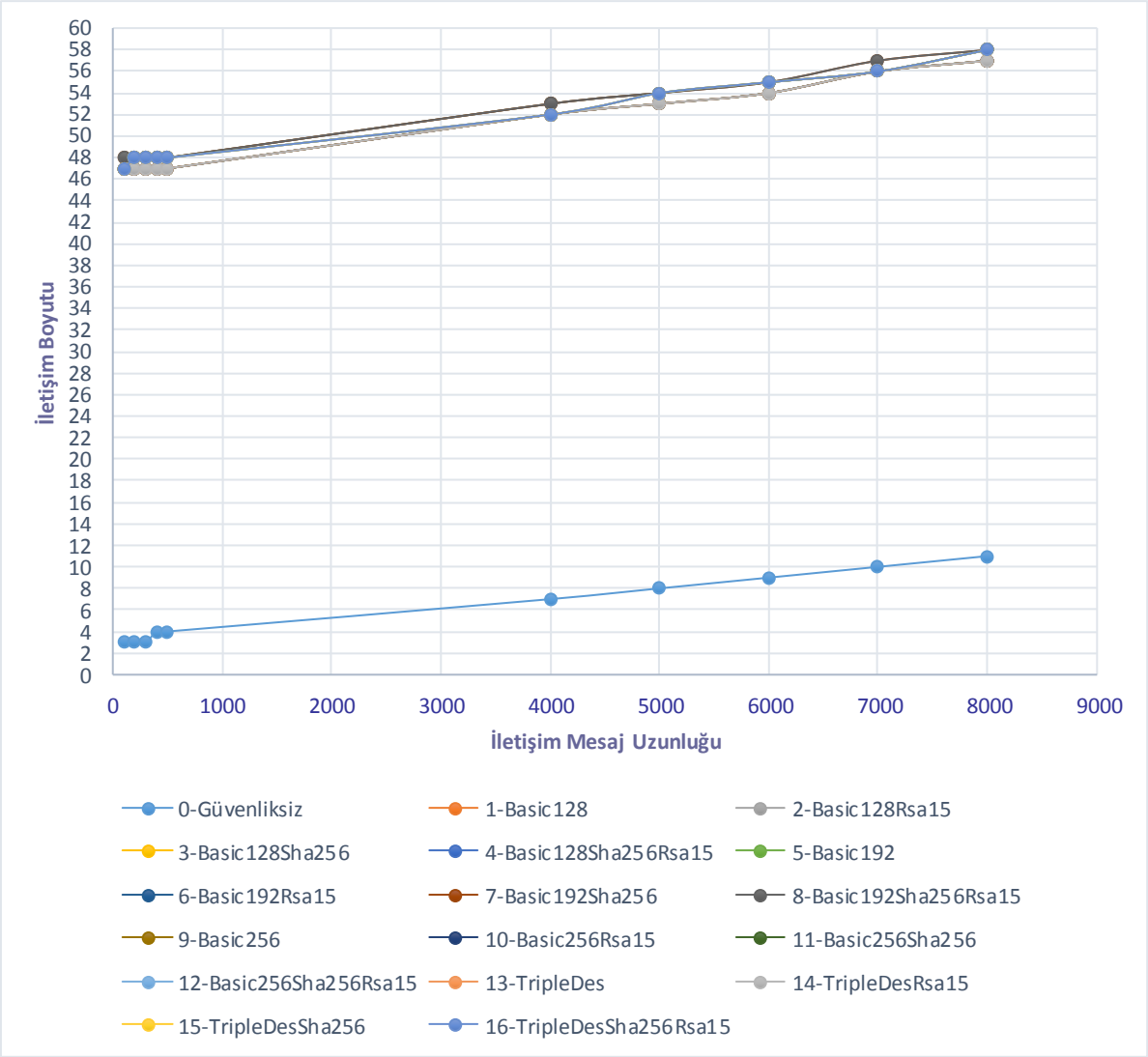

Çizelge C. 6 incelendiği zaman güvenliksiz ortamda iletilen mesajların düz metin durumlarıyla iletişim yapıldığında iletişim mesaj sayısı sadece iki iken, şifrelenmiş mesajlarla iletişim yapılmak istenildiği zaman iletişim sayısı beş kat artarak on olmuştur.

Çizelge C. 7 HTTP protokolü kullanılarak çeşitli şifreleme algoritmaları ile iletişim mesaj uzunluğu –iletişim süresi (ms.) ilişkisi çizelgesi



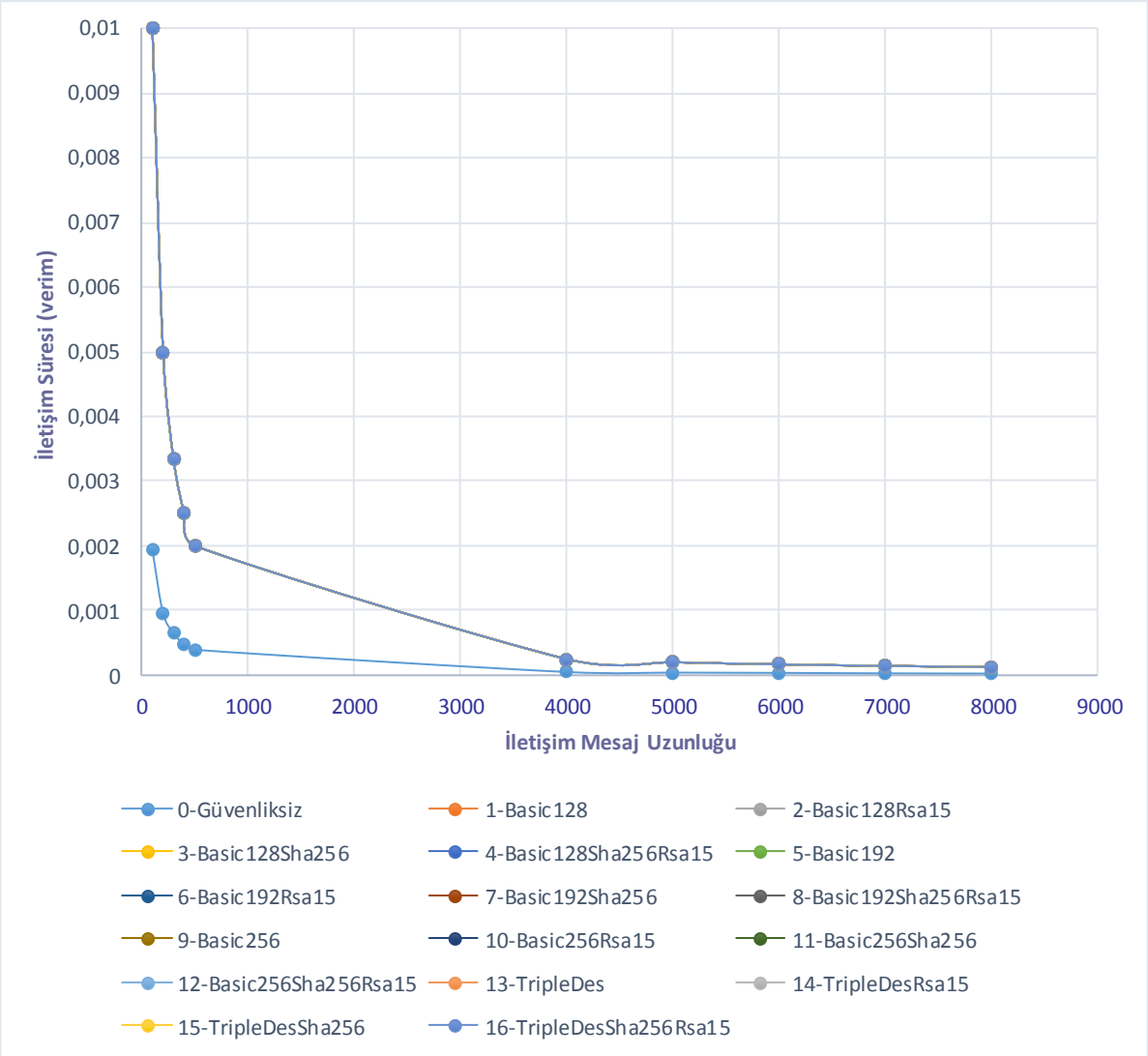

Çizelge C. 7 incelendiğinde güvenliksiz ortamda iletişim süresi 0. 2 milisaniye civarında olmasına karşın diğer şifreleme algoritmalarıyla oluşturulmuş ortamlarda süre 1 mili saniye civarına çıkmıştır.

Çizelge C. 8 HTTP protokolü kullanılarak çeşitli şifreleme algoritmaları ile iletişim mesaj uzunluğu –iletişim boyutu (kb.) ilişkisi çizelgesi



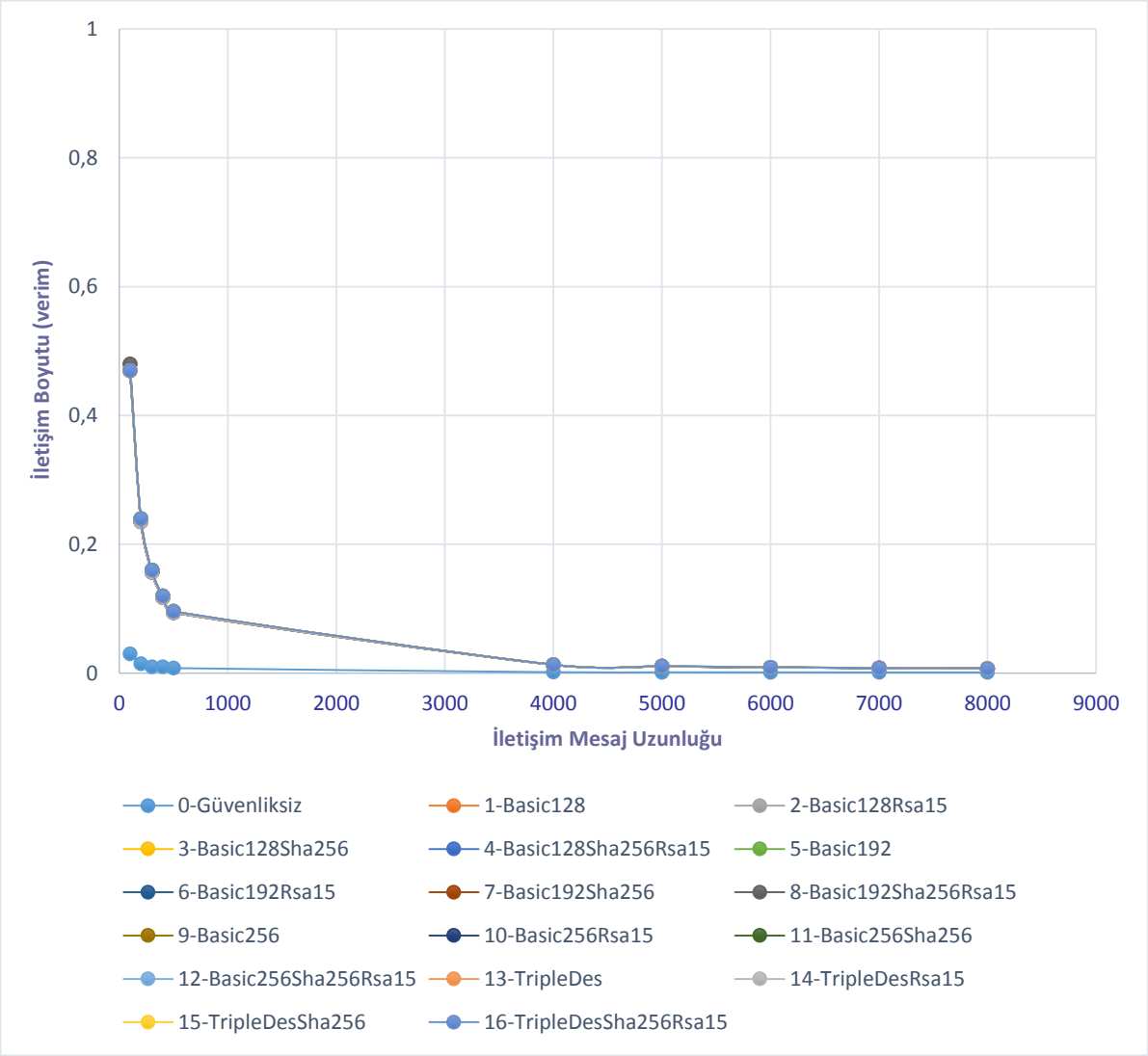

Çizelge C. 8 incelendiğinde güvenliksiz ve simetrik şifreleme algoritmalarının değişen mesaj boyutuna karşın iletişim boyutu düzenli olarak artmasına karşın asimetrik şifreleme algoritmasıyla oluşturulmuş ortamlarda ise düzenin çok az bozulduğu gözlemlenmiştir.

Çizelge C. 9 HTTP protokolü kullanılarak çeşitli şifreleme algoritmaları ile iletişim mesaj uzunluğu –iletişim süresi (verim) ilişkisi çizelgesi




Çizelge C. 9 incelendiğinden SOAP protokolünde iletişim mesajının karakter sayısının arttırılması ortamın karakter başına iletişim süresinde daha da verimli olmasına sağlamıştır. Fakat bu verim iletişim mesajının karakter sayısının artılımasınız bu ortamdaki verimliliği giderek azalmıştır.

Çizelge C. 10 HTTP protokolü kullanılarak çeşitli şifreleme algoritmaları ile iletişim mesaj uzunluğu – iletişim boyutu (verim) ilişkisi çizelgesi



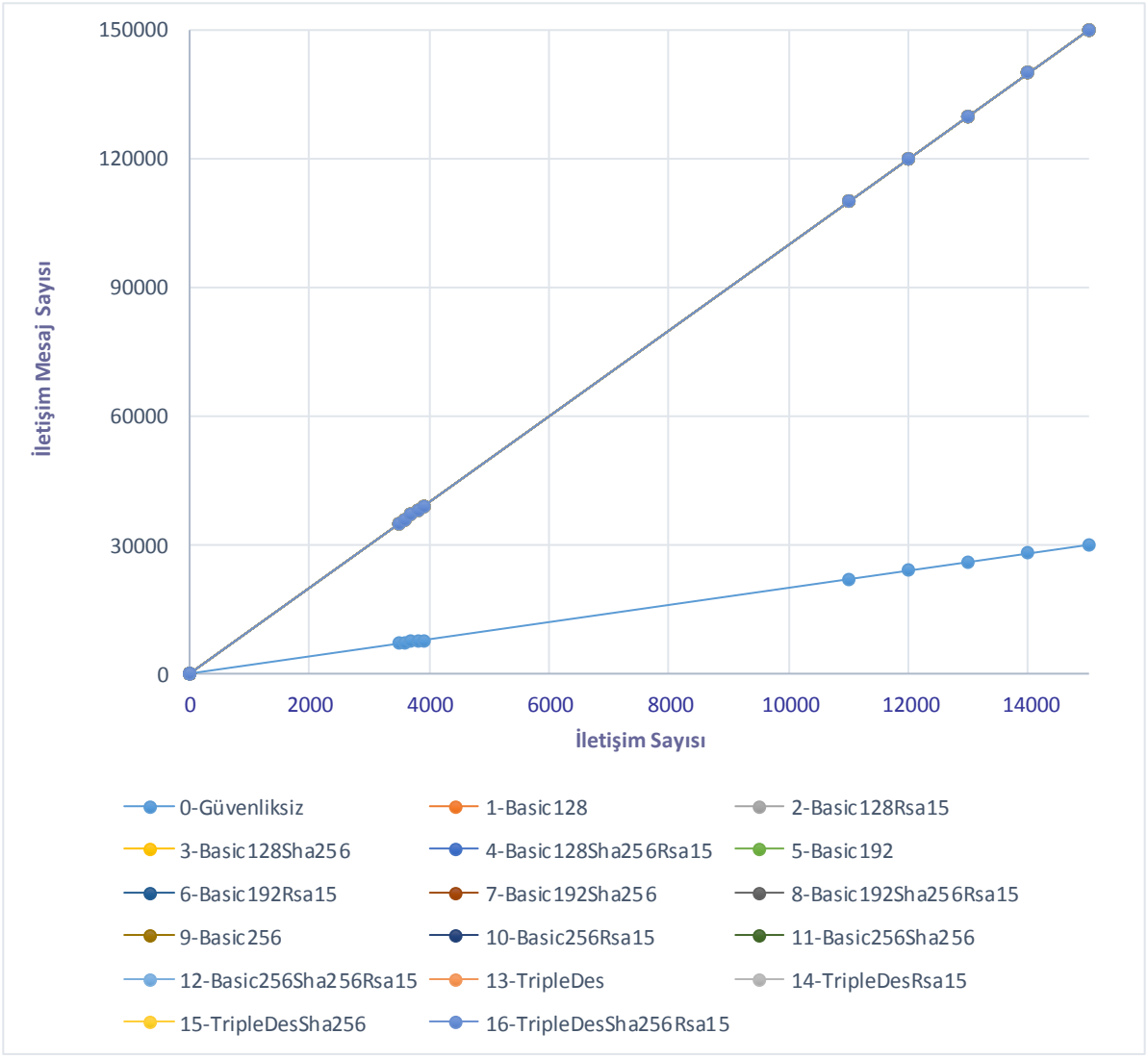

Çizelge C. 10 incelendiğinden SOAP protokolünde iletişim mesajının karakter sayısının arttırılması ortamın karakter başına iletişim boyutunun daha da verimli olmasına sağlamıştır. Fakat bu verim iletişim mesajının karakter sayısının artılımasınız bu ortamdaki verimliliği giderek azalmıştır.

Sonuç olarak ortamda tek seferlik mesaj iletişiminde değişen iletişim mesaj karakter sayısının test edildiği durumda kullanılan protokol çıkan test sonuçlarını pek etkilememiştir. Birbirine yakın test sonuçları elde eden şifreleme algoritmalarına bu bölümde değil yakınlık (ilişki) grafikleri gösterildiği bölümde incelenmiştir.

**C - 2 Çok İstemcili Sabit Kelime Katarlı Test İşlemi Verileri**

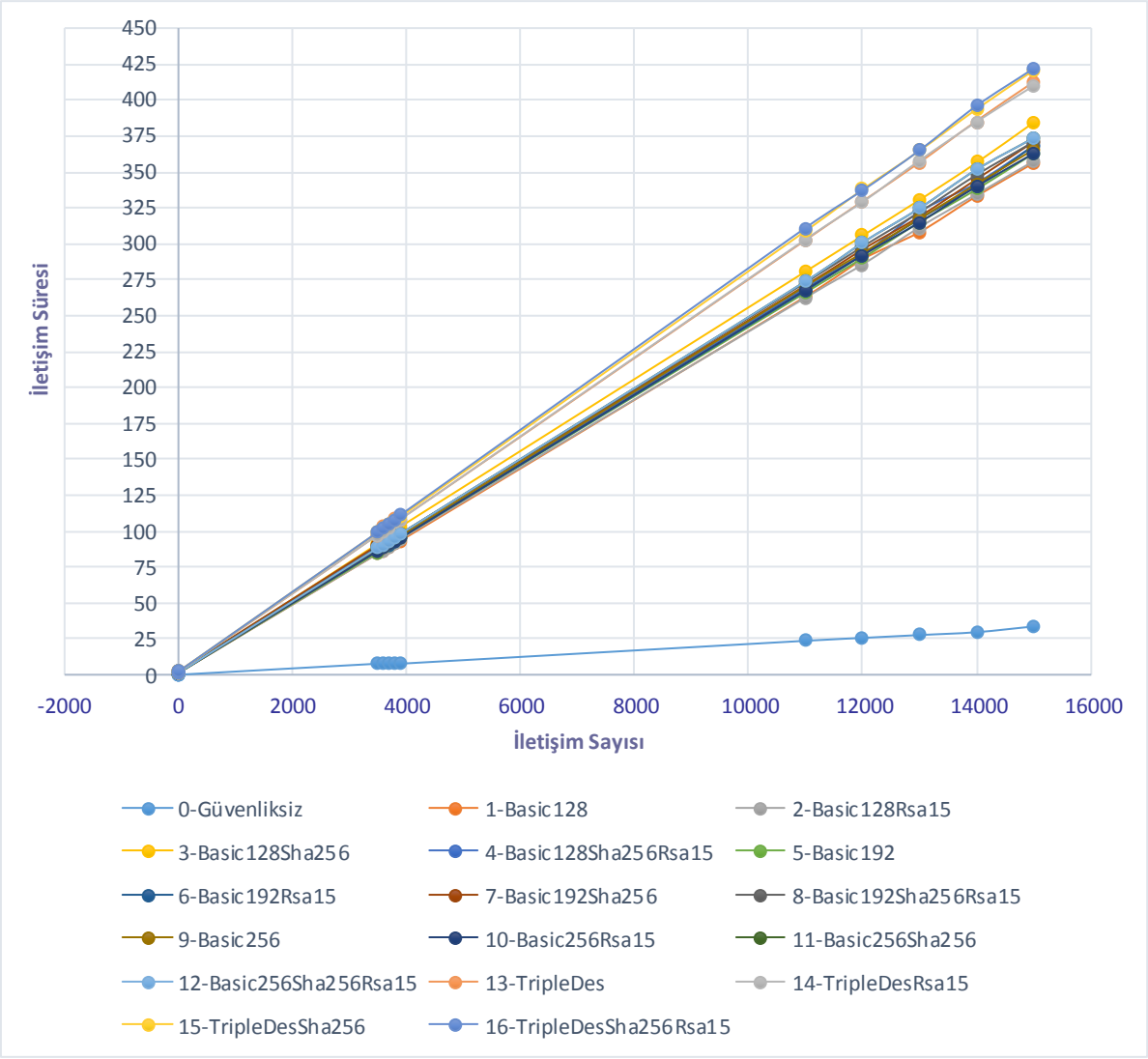
**C - 2. 1 NetTcpBinding Bağlama Kullanılarak Elde Edilen Veriler**

Çizelge C. 11 TCP protokolü kullanılarak çeşitli şifreleme algoritmaları ile iletişim mesaj sayısı – iletişim sayısı ilişkisi çizelgesi



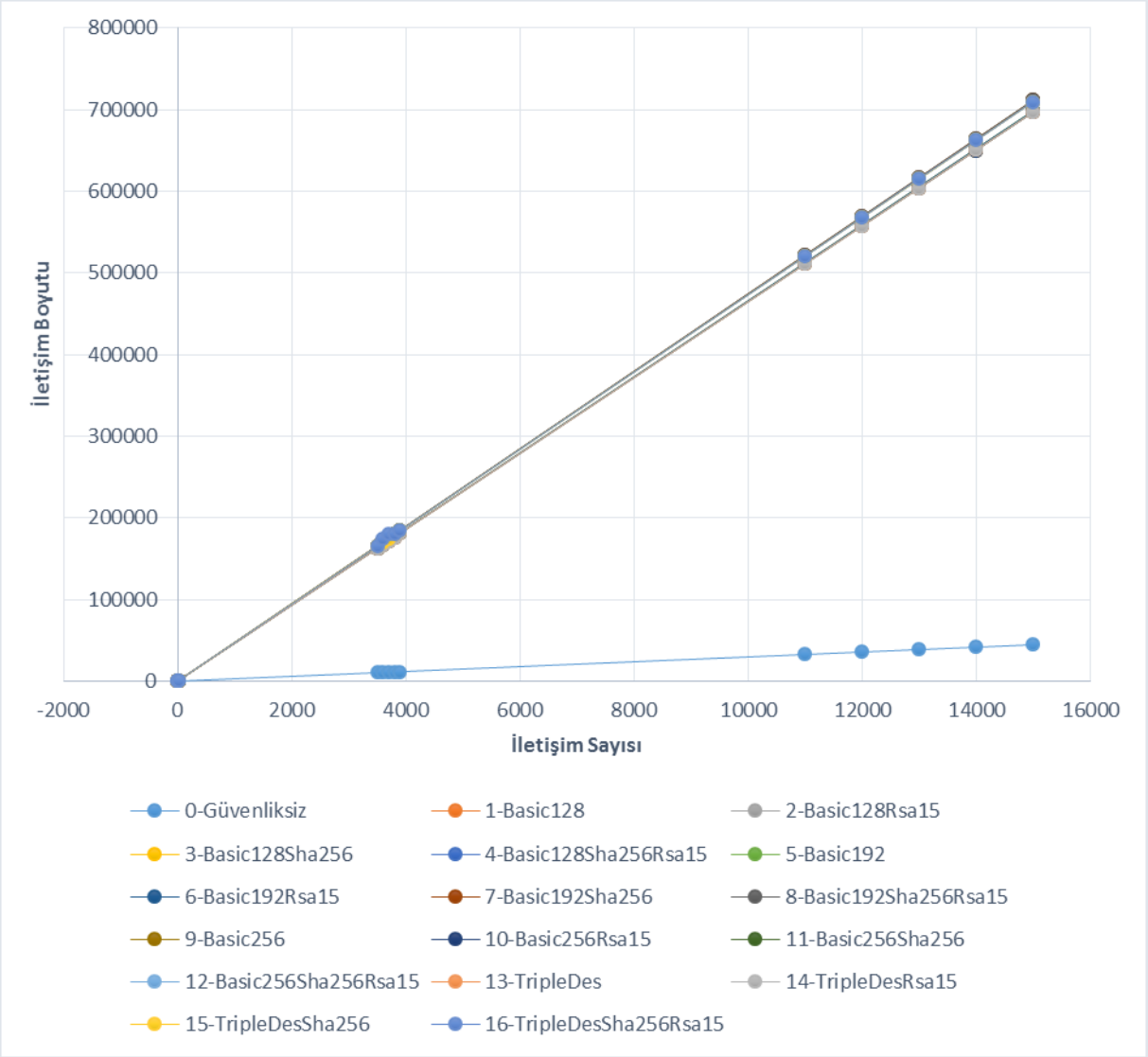

Çizelge C. 11 incelendiği zaman güvenliksiz ortamda iletişim mesaj sayısı, iletişim sayısının iki katı iken; şifreli iletişimin bulunduğu ortamda iletişim mesaj sayısı, iletişim sayısının on katı olmuştur.

Çizelge C. 12 TCP protokolü kullanılarak çeşitli şifreleme algoritmaları ile iletişim süresi – iletişim sayısı ilişkisi çizelgesi



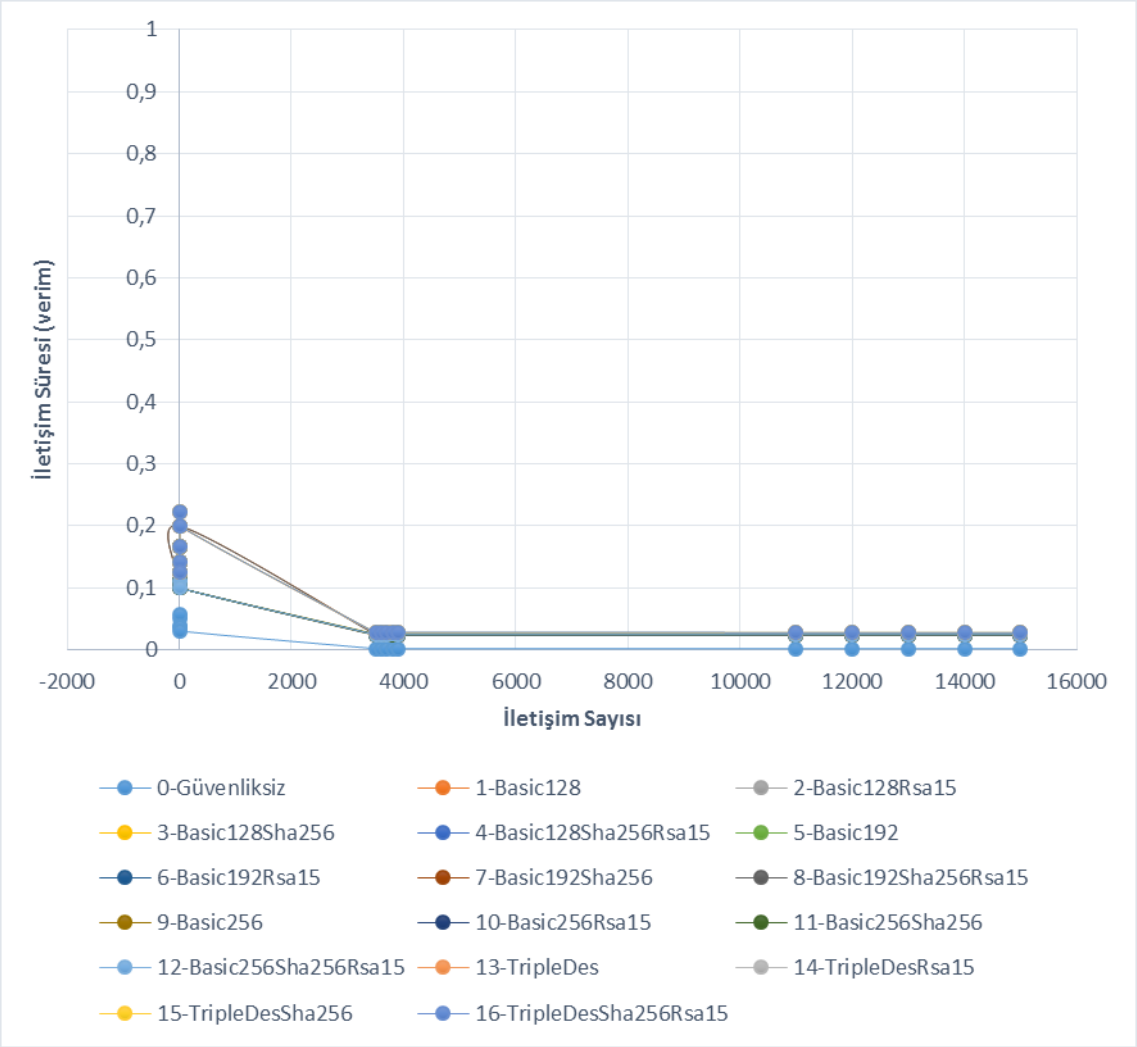

Çizelge C. 12 incelendiği zaman güvenliksiz ve şifreli iletişimin olduğu ortamlarda iletişim sayısı arttırıldığında iletişim süresinin doğrusal olarak arttığı gözlemlenmiştir. Güvenliksiz ortam ile şifreli ortamdaki elde edilen değerler giderek artmıştır.

Çizelge C. 13 TCP protokolü kullanılarak çeşitli şifreleme algoritmaları ile iletişim boyutu – iletişim sayısı ilişkisi çizelgesi



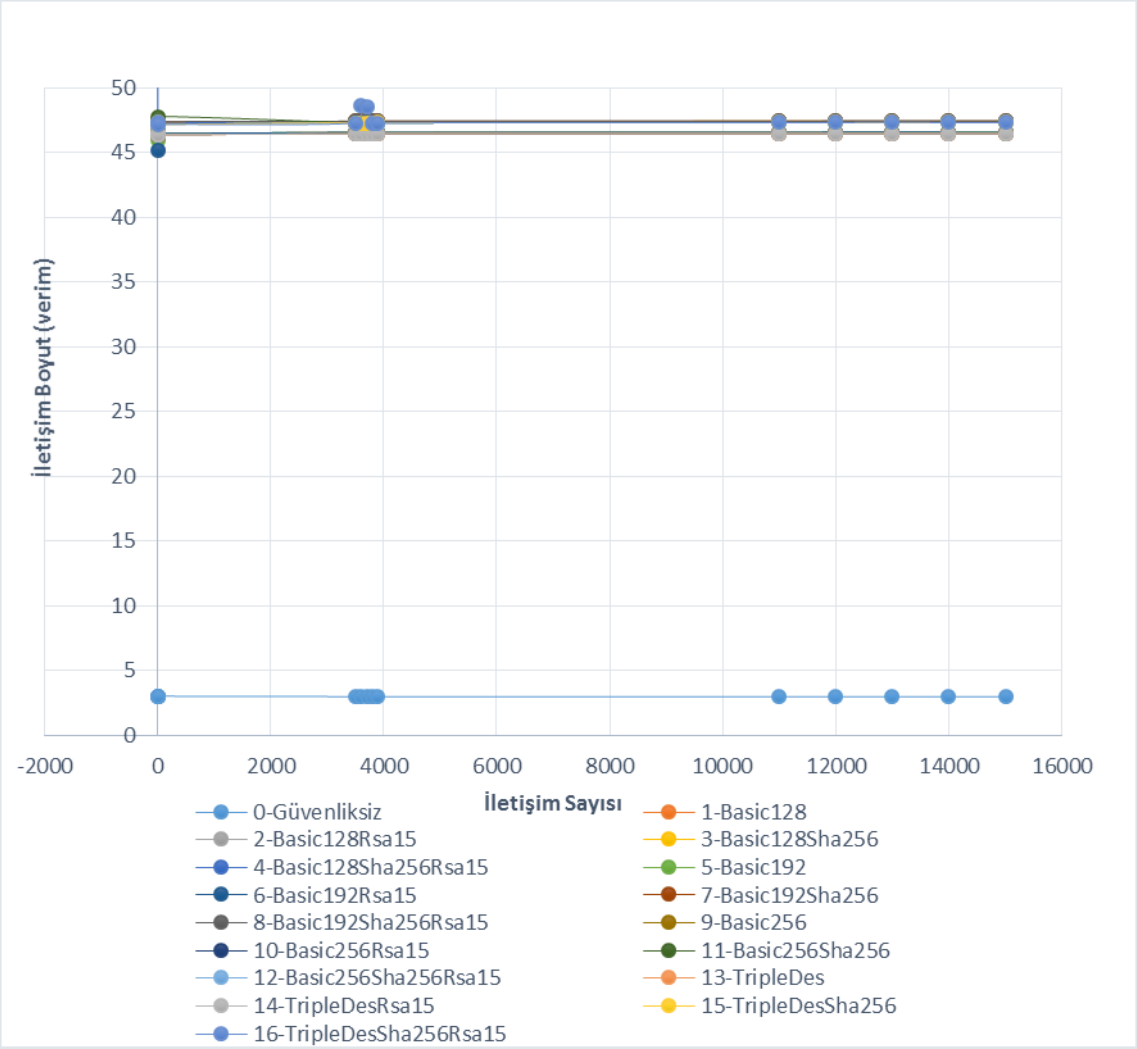

Çizelge C. 13 incelendiği zaman güvenliksiz ve şifreli iletişimin olduğu ortamlarda iletişim sayısı arttırıldığında iletişim süresinin doğrusal olarak arttığı gözlemlenmiştir. Güvenliksiz ortam ile şifreli ortamdaki elde edilen değerler giderek artmıştır.

Çizelge C. 14 TCP protokolü kullanılarak çeşitli şifreleme algoritmaları ile iletişim süresi (verim) – iletişim sayısı ilişkisi çizelgesi




Çizelge C. 14 incelendiği zaman güvenliksiz ve şifreli iletişimin olduğu ortamlarda iletişim sayısı arttırıldığında iletişim süresinin verimliliği azalmış belli bir seviyeden sonra sabitlenmiştir. Asimetik şifreleme algoritmaları ile kurulan test ortamında ise iletişim süresinin verimliliği iletişim süresinin çok az asimetrik olduğundan dolayı negatif değerler alabilir.

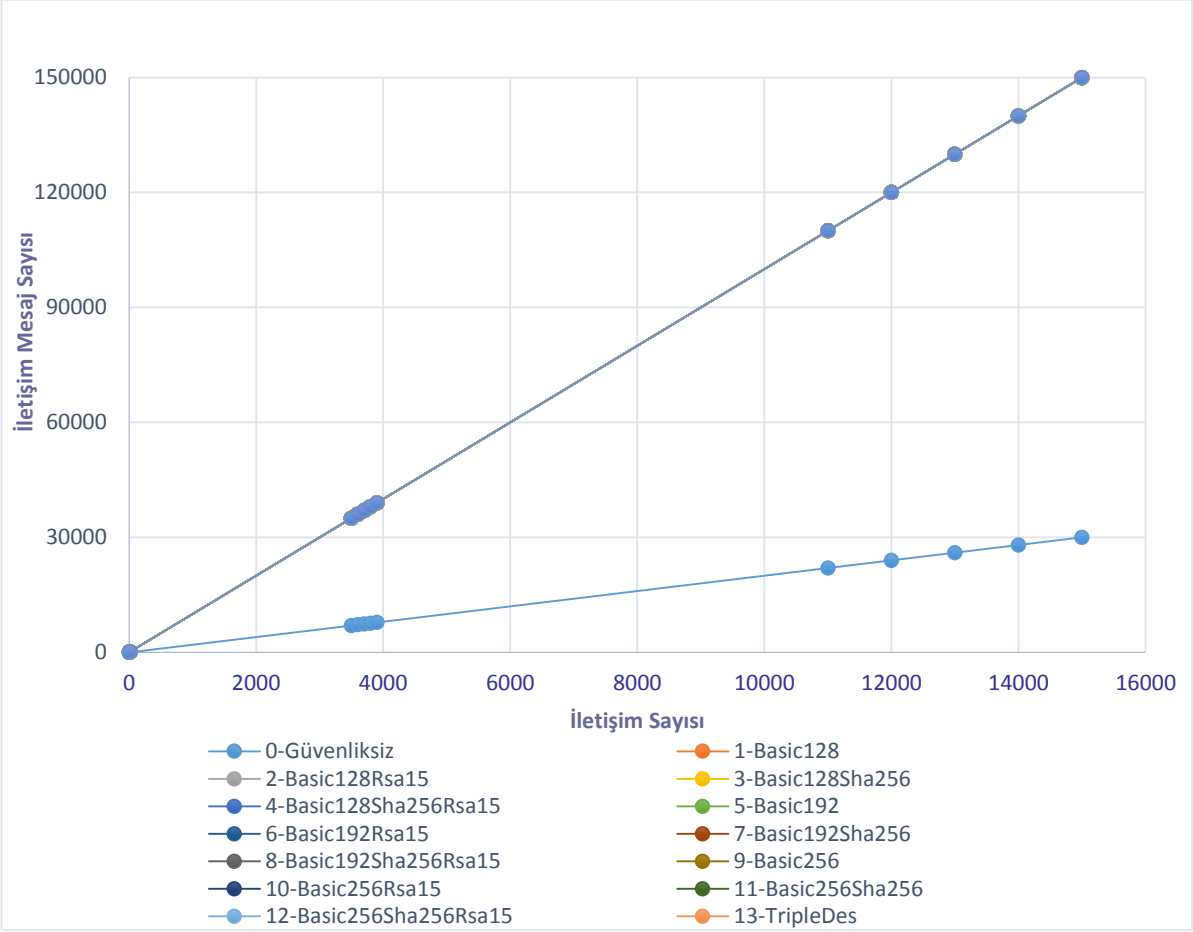
Çizelge C. 15 TCP protokolü kullanılarak çeşitli şifreleme algoritmaları ile iletişim boyut (verim) – iletişim sayısı ilişkisi çizelgesi




Çizelge C. 15 incelendiği zaman güvenliksiz ve şifreli iletişimin olduğu ortamlarda iletişim sayısı arttırıldığında iletişim boyutu verimliliği sabit olduğu gözlemlenmiştir.

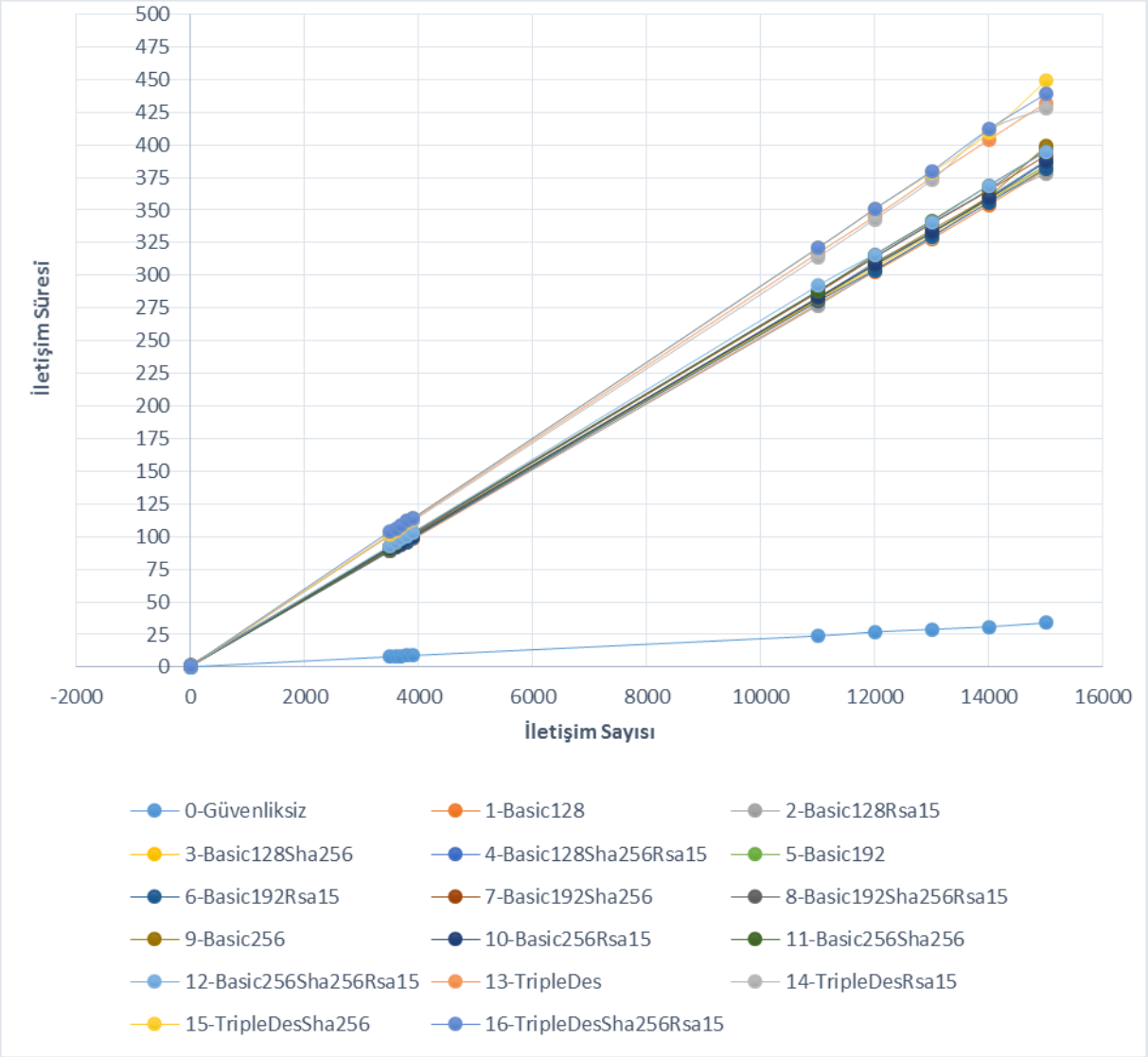
### C - 2. 2 WsHttpBinding Bağlama Kullanılarak Elde Edilen Veriler

Çizelge C. 16 HTTP protokolü kullanılarak çeşitli şifreleme algoritmaları ile iletişim mesaj sayısı – iletişim sayısı ilişkisi çizelgesi



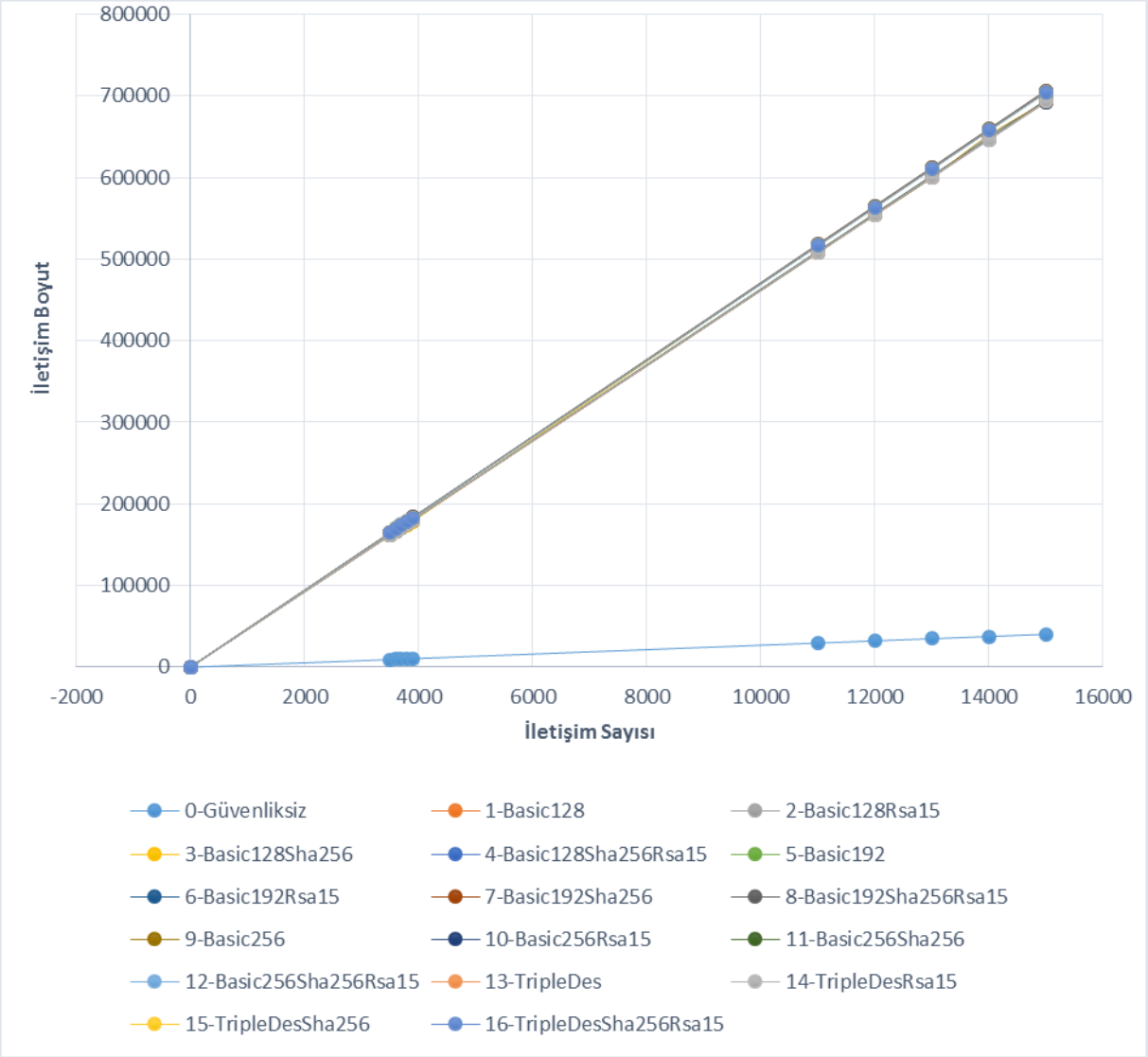

Çizelge C. 16 incelendiği zaman güvenliksiz ortamda iletişim mesaj sayısı, iletişim sayısının iki katı iken; şifreli iletişimin bulunduğu ortamda iletişim mesaj sayısı, iletişim sayısının on katı olmuştur.

Çizelge C. 17 HTTP protokolü kullanılarak çeşitli şifreleme algoritmaları ile iletişim süresi – iletişim sayısı ilişkisi çizelgesi



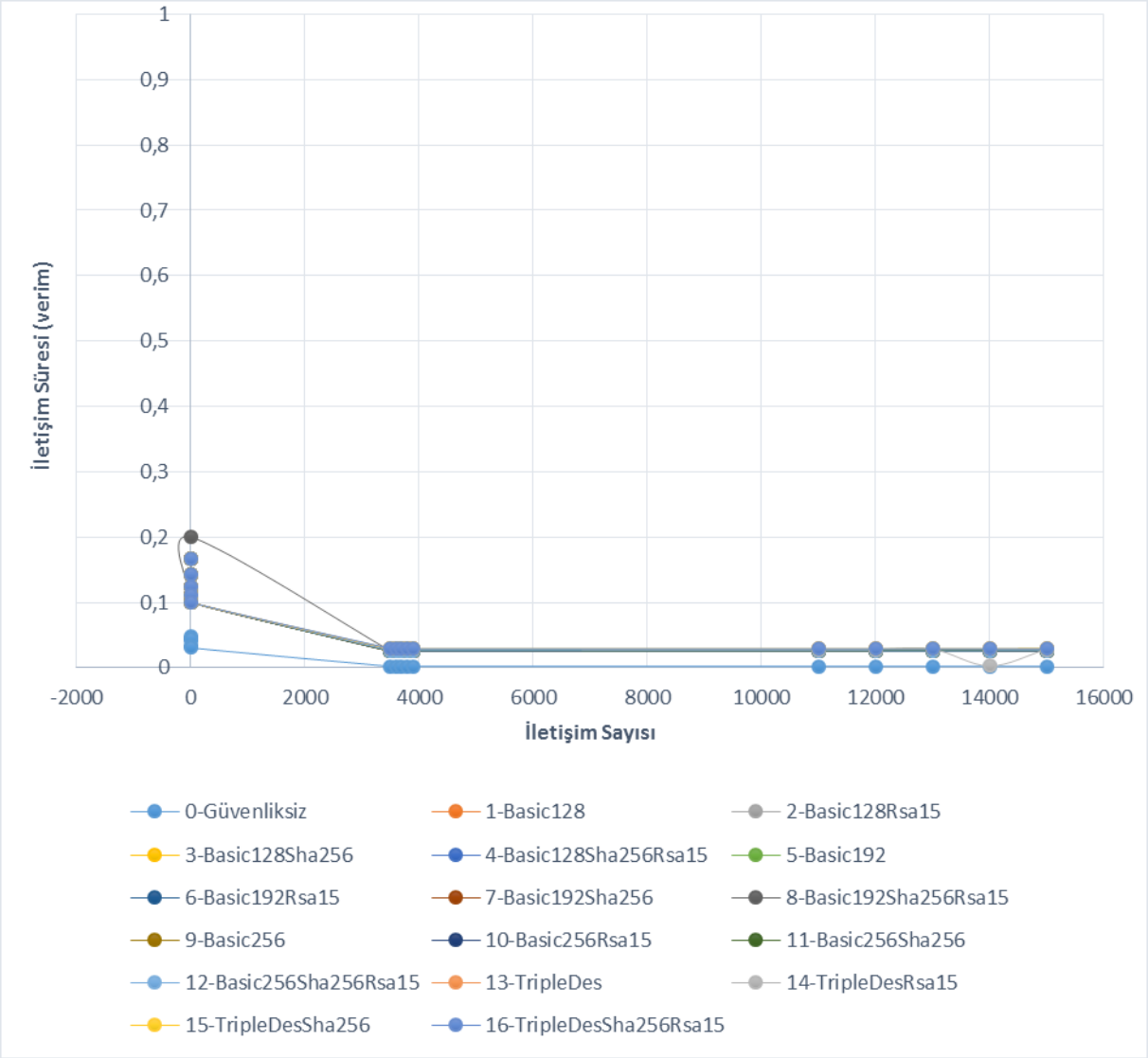

Çizelge C. 17 incelendiği zaman güvenliksiz ve şifreli iletişimin olduğu ortamlarda iletişim sayısı arttırıldığında iletişim süresinin doğrusal olarak arttığı gözlemlenmiştir. Güvenliksiz ortam ile şifreli ortamdaki elde edilen değerler giderek artmıştır.

Çizelge C. 18 HTTP protokolü kullanılarak çeşitli şifreleme algoritmaları ile iletişim boyutu – iletişim sayısı ilişkisi çizelgesi



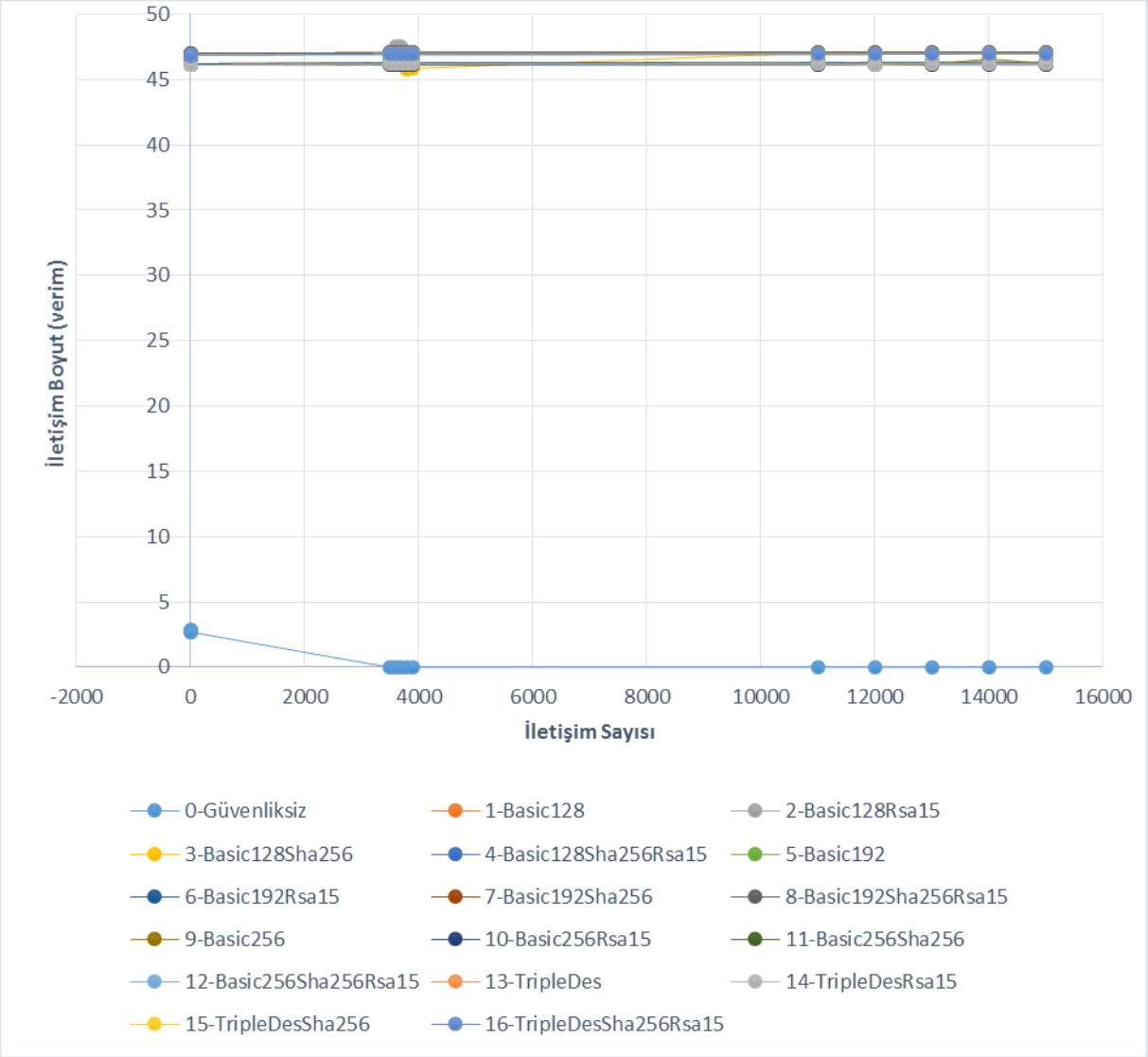

Çizelge C. 18 incelendiği zaman güvenliksiz ve şifreli iletişimin olduğu ortamlarda iletişim sayısı arttırıldığında iletişim süresinin doğrusal olarak arttığı gözlemlenmiştir. Güvenliksiz ortam ile şifreli ortamdaki elde edilen değerler giderek artmıştır.

Çizelge C. 19 HTTP protokolü kullanılarak çeşitli şifreleme algoritmaları ile iletişim süresi (verim) – iletişim sayısı ilişkisi çizelgesi




Çizelge C. 19 incelendiği zaman güvenliksiz ve şifreli iletişimin olduğu ortamlarda iletişim sayısı arttırıldığında iletişim süresinin verimliliği azalmış belli bir seviyeden sonra sabitlenmiştir. Asimetik şifreleme algoritmaları ile kurulan test ortamında ise iletişim süresinin verimliliği iletişim süresinin çok az asimetrik olduğundan dolayı negatif değerler alabilir.

Çizelge C. 20 HTTP protokolü kullanılarak çeşitli şifreleme algoritmaları ile iletişim boyut (verim) – iletişim sayısı ilişkisi çizelgesi




Çizelge C. 20 incelendiği zaman güvenliksiz ve şifreli iletişimin olduğu ortamlarda iletişim sayısı arttırıldığında iletişim boyutu verimliliği sabit olduğu gözlemlenmiştir.

**C - 3 İki Bağlama Kullanılarak Elde Edilen Fark Verileri**

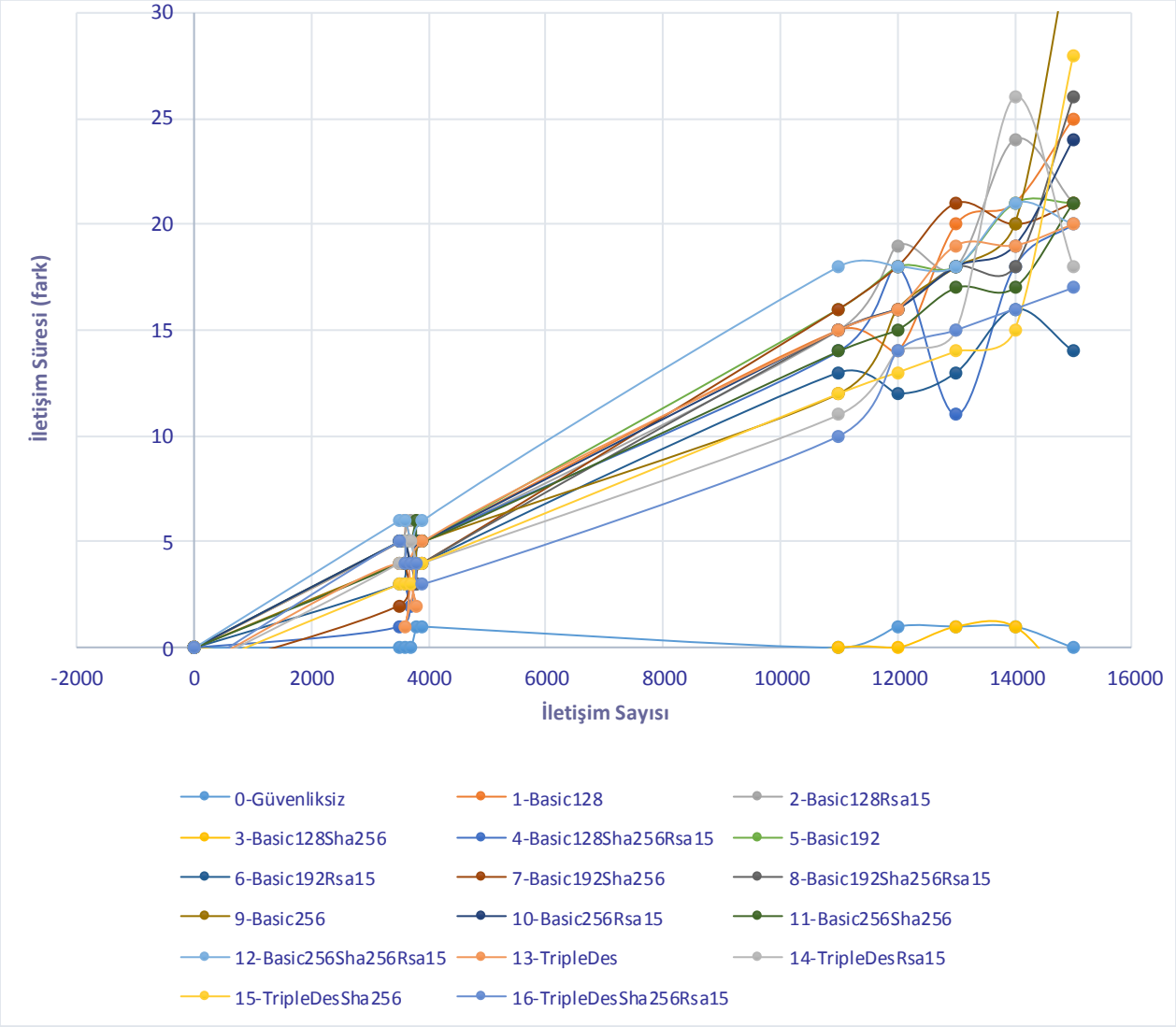
**C - 3. 1 Tek İstemcili Değişken Kelime Katarlı Test İşlemi Fark Verileri**

Bu bölümde iki bağlama kullanılarak değişen iletişim mesaj uzunluğunda farklı protokollere bağlı olarak yapılan iletişimde çok az fark olduğundan fark grafikleri verilmemiştir.

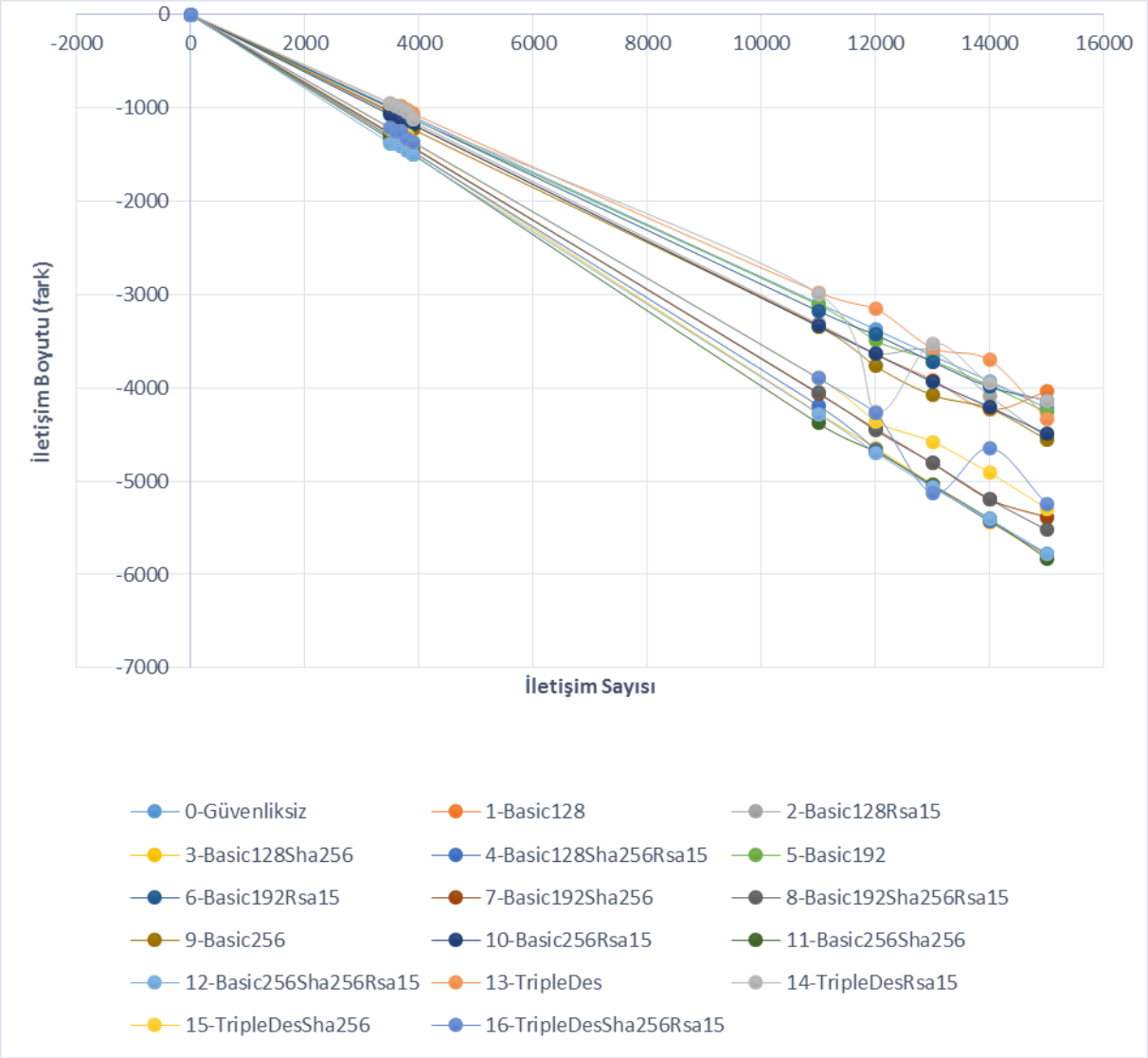
**C - 3. 2 Çok İstemcili Değişken Kelime Katarlı Test İşlemi Fark Verileri**

Çizelge C.21 incelendiği zaman iletişimde kullanılan protokolün çeşitli şifreleme algoritmaları üzerinde etkisi fazla iken çeşitli şifreleme algoritmaları üzerinde azdır. Genel olarak TCP protokolünün HTTP protokolünden daha hızlı çalıştığı gözlemlenmiştir. Asimetrik şifreleme algoritmalarında ise iletişim süresi asimetrik olduğundan iki protokol arasındaki iletişim süresi farkı bazen azalıp bazende artmıştır.

Çizelge C. 21 Güvenliksiz olarak ve çeşitli şifreleme algoritmaları ile iletişim süresi (fark) – iletişim sayısı ilişkisi çizelgesi

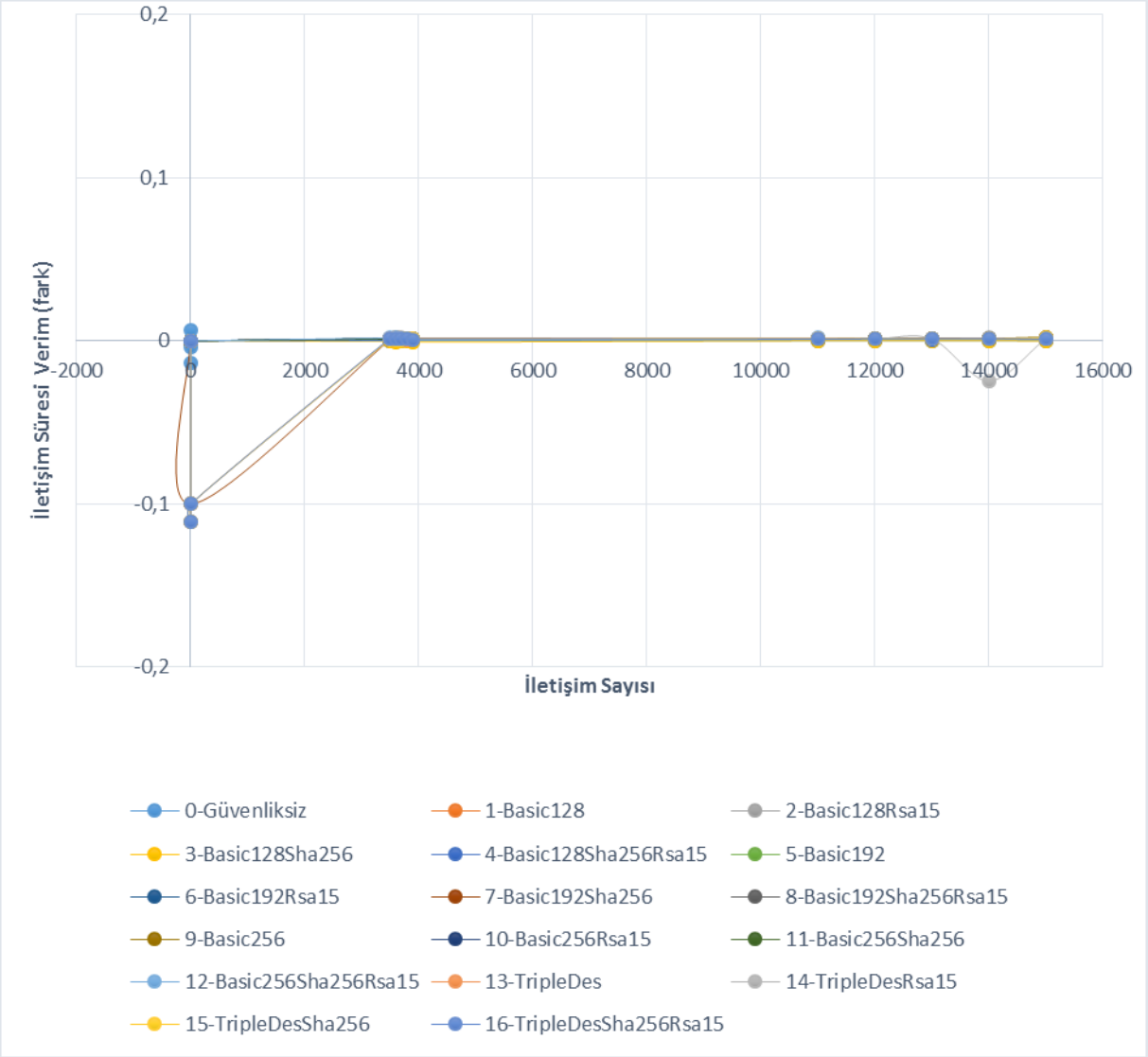


Çizelge C. 22 Güvenliksiz olarak ve çeşitli şifreleme algoritmaları ile iletişim boyutu (fark) – iletişim sayısı ilişkisi çizelgesi



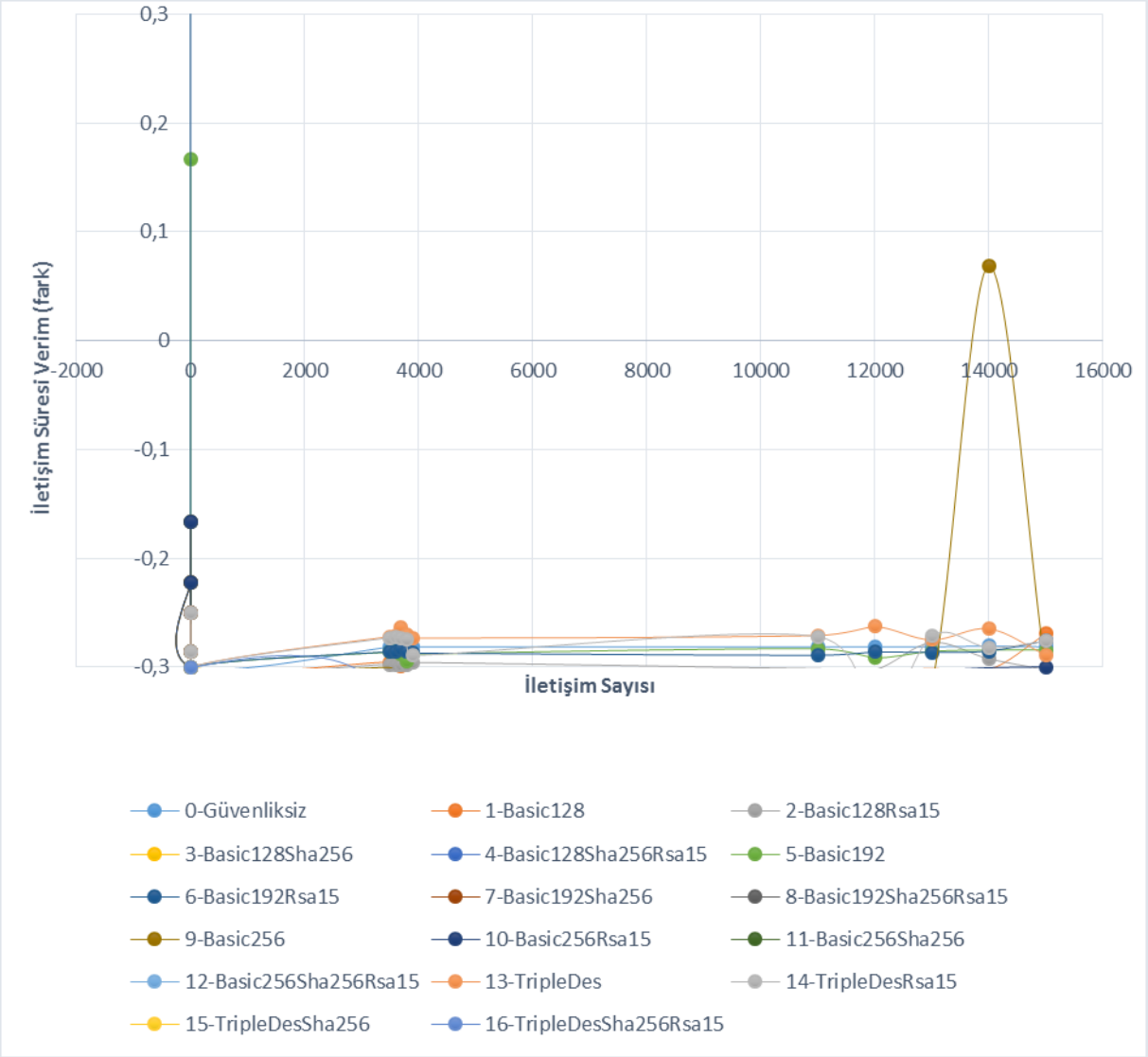

Çizelge C.22 incelendiği zaman iletişimde kullanılan protokolün çeşitli şifreleme algoritmaları üzerinde etkisi fazla iken çeşitli şifreleme algoritmaları üzerinde azdır. Genel olarak TCP protokolü ile iletişim yapıldığında HTTP protokolüne göre mesajların daha çok boyut aldığı gözlemlenmiştir. Asimetrik şifreleme algoritmalarında ise iletişim süresi asimetrik olduğundan iki protokol arasındaki iletişim süresi farkı bazen azalıp bazende artmıştır.

Çizelge C. 23 Güvenliksiz olarak ve çeşitli şifreleme algoritmaları ile iletişim boyutu verim (fark) – iletişim sayısı ilişkisi çizelgesi




Çizelge C.23 incelendiği zaman iletişimde kullanılan protokolün çeşitli şifreleme algoritmaları üzerinde etkisi azdır. Her iki protokolde de iletişim süresi verimi çok iyidir.

Çizelge C. 24 Güvenliksiz olarak ve çeşitli şifreleme algoritmaları ile iletişim süresi verim (fark) – iletişim sayısı ilişkisi çizelgesi




Çizelge C. 24 incelendiği zaman iletişimde kullanılan protokolün çeşitli şifreleme algoritmaları üzerinde etkisi azdır. Her iki protokolde de iletişim süresi verimi çok iyidir. Bazı asimetrik şifreleme algoritmalarında pozitif ne negatif değerler elde edilmiştir.

# ÖZGEÇMİŞ

**KİŞİSEL BİLGİLER**


**Adı Soyadı** : Mirsat YEŞİLTEPE
**Doğum Tarihi ve Yeri** : 29/01/1989
**Yabancı Dili** : İngilizce
**E-posta** : mirsaty@yildiz.edu.tr


**ÖĞRENİM DURUMU**

| Derece | Alan | Okul/Üniversite | Mezuniyet Yılı |
|---|---|---|---|
| Lisans | Bilgisayar Müh. | Trakya Üni. | 2012 |
| | Maliye | Anadolu Üni. | 2013 |
| Lise | Fen Bilimleri | Dede Korkut Anadolu Lis. | 2007 |

**İŞ TECRÜBESİ**

| Yıl | Firma/Kurum | Görevi |
|---|---|---|
| 2015 | Yıldız Teknik Üni. Kimya Metalurji Fak. | Araştırma Görevlisi |

**YAYINLARI**

**Bildiri**

1. “WCF Ortamında Oluşan İstemci Ve/Veya Sunucu Hatalarının İstemci ve Sunucularda Bildirimi” - Mirsat Yeşiltepe, Ömer Özgür Bozkurt; Bilişim 2014, TBD 31. Ulusal Bilişim Kurultayı, 6-9 Kasım 2014, Ankara.
2. “Servis Odaklı Mimari Güvenlik Tipleri Karşılaştırılması” - Mirsat Yeşiltepe, Ö. Özgür Bozkurt; XVII. Akademik Bilişim Konferansı, AB 2015, 31 Ocak – 6 Şubat 2015, Eskişehir.
3. “Bulut Ortamında Doğru Web Servis Güvenliği Kripto Bileşenlerinin Seçilmesi” - Mirsat Yeşiltepe, Ö. Özgür Bozkurt; International Symposium On Digital Forensics And Security, ISDFS-2015, 11-12 Mayıs 2015, Ankara.
4. “WCF’de Throttling (Azaltma) Mekanizmasının Protokol Bazlı İncelenmesi”- Mirsat Yeşiltepe, Muhammet Kurulay; International Symposium On Digital Forensics And Security, ISDFS-2015, 11-12 Mayıs 2015, Ankara.
5. “Security Type Comparision In Service Oriented Architecture Security” - Mirsat Yeşiltepe, Ö. Özgür Bozkurt; Istanbul Unıversity World Conference On Technology, Innovation And Entrepreneurship 2015, 28 – 30 Mayıs 2015, İstanbul.