EV OTOMASYON SİSTEMLERİ İÇİN GÜVENLİK VE GİZLİLİK ODAKLI AĞ GEÇİDİ TASARIMI

Ahmet KAŞİF

T.C.
BURSA ULUDAĞ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

Ev Otomasyon Sistemleri İçin Güvenlik ve Gizlilik Odaklı Ağ Geçidi Tasarımı

Ahmet KAŞİF

(Orcid: 0000-0003-2707-6075)

Dr. Öğr. Üyesi Cengiz TOĞAY

(Danışman)

Prof. Dr. Albert LEVI

(İkinci Danışman)
Sabancı Üniversitesi

YÜKSEK LİSANS TEZİ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

BURSA – 2020

TEZ ONAYI

Ahmet KAŞİF tarafından hazırlanan "Ev Otomasyon Sistemleri için Güvenlik ve Gizlilik odaklı Ağ Geçidi Tasarımı" adlı tez çalışması aşağıdaki jüri tarafından oy birliği ile Bursa Uludağ Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Danışman : Dr. Öğr. Üyesi Cengiz TOĞAY
İkinci Danışman : Prof. Dr. Albert LEVI

Başkan : Dr. Öğr. Üyesi Cengiz TOĞAY — İmza
Bursa Uludağ Üniversitesi, Mühendislik Fakültesi,
Bilgisayar Mühendisliği Anabilim Dalı
Orcid: 0000-0001-5739-1784

Üye : Doç. Dr. Gıyasettin ÖZCAN — İmza
Bursa Uludağ Üniversitesi, Mühendislik Fakültesi,
Bilgisayar Mühendisliği Anabilim Dalı
Orcid: 0000-0002-1166-5919

Üye : Dr. Öğr. Üyesi İzzet Fatih ŞENTÜRK — İmza
Bursa Teknik Üniversitesi, Mühendislik Fakültesi,
Bilgisayar Mühendisliği Anabilim Dalı
Orcid: 0000-0002-1550-563X

Yukarıdaki sonucu onaylarım

Prof. Dr. Hüseyin Aksel EREN
Enstitü Müdürü
.../.../...

U.Ü. Fen Bilimleri Enstitüsü, tez yazım kurallarına uygun olarak hazırladığım bu tez çalışmasında;

- tez içindeki bütün bilgi ve belgeleri akademik kurallar çerçevesinde elde ettiğimi,
- görsel, işitsel ve yazılı tüm bilgi ve sonuçları bilimsel ahlak kurallarına uygun olarak sunduğumu,
- başkalarının eserlerinden yararlanılması durumunda ilgili eserlere bilimsel normlara uygun olarak atıfta bulunduğumu,
- atıfta bulunduğum eserlerin tümünü kaynak olarak gösterdiğimi,
- kullanılan verilerde herhangi bir tahrifat yapmadığımı,
- ve bu tezin herhangi bir bölümünü bu üniversite veya başka bir üniversitede başka bir tez çalışması olarak sunmadığımı

beyan ederim.

.../.../….

Ahmet KAŞİF

ÖZET

Yüksek Lisans Tezi

Ev Otomasyon Sistemleri İçin Güvenlik ve Gizlilik Odaklı Ağ Geçidi Tasarımı

Ahmet KAŞİF

Bursa Uludağ Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Dr. Öğr. Üyesi Cengiz TOĞAY
İkinci Danışman: Prof. Dr. Albert LEVI

Nesnelerin İnterneti (IoT) teknolojisi son yılların en popüler araştırma konularından olup günümüzde sağlık, ulaşım ve eğitim gibi birçok alanda kullanılmaktadır. Bu alanlardan biri olan ev otomasyon sistemleri de IoT'nin yaygınlaşması ile insan hayatında daha çok yer etmeye başlamıştır. IoT'de kullanılan cihazların iletişim protokolleri ve mesaj içerikleri açısından çeşitli ve birbirleri ile uyumsuz olması, donanım kaynaklarının çoğu zaman yetersiz olması sebebiyle güvenlik, gizlilik ve birlikte çalışabilirlik alanlarında sorunlar oluşmaktadır. Sunulan tezde, I) farklı protokolleri kullanan cihazlar arasındaki iletişim probleminin çözülmesi, II) mesaj içeriklerinin gönderici bazında anomali açısından değerlendirilmesi, III) alıcıya uygun hale getirilmesi (anonimleştirme, zenginleştirme ve fakirleştirme), IV) cihazların sadece tanımlı oldukları amaçlar dahilinde dünya ile iletişim kurmasının sağlanması, V) cihazın internetteki ya da aynı ağdaki bir cihaza saldırmasının engellenmesi ve VI) cihazların güncelleme operasyonlarındaki zayıflıkların giderilmesi konularında bir çözüm sunulmaktadır. Bu kapsamda geliştirdiğimiz platform, sunucu, kullanıcı arayüzü olarak bir Android uygulaması ve bir ağ geçidinden meydana gelmektedir. Ağ geçidinde, aracı (broker), kural tabanlı mesaj işleme uygulaması, IP tabanlı kısıtlama ve çoklu SSID yaklaşımları birlikte uygulanmıştır. Platform sayesinde farklı üreticilere ait cihazlar birlikte çalışırken ele geçirilmiş ya da beklendiği gibi çalışmayan bir cihazın bulunduğu ortamdaki diğer cihazlara ya da internetteki bir hedefe atakta bulunmaları engellenmiştir.

Anahtar Kelimeler: ağ geçidi, nesnelerin interneti, ağ güvenliği, veri mahremiyeti, DDoS saldırıları
2020, vii + 60 sayfa.

ABSTRACT

MSc Thesis

Design of Privacy and Security Centric Home Automation System

Ahmet KAŞİF

Bursa Uludağ University
Graduate School of Natural and Applied Sciences
Department of Computer Engineering

Supervisor: Asst. Prof. Dr. Cengiz TOĞAY
Second Supervisor: Prof. Dr. Albert LEVI

Internet of Things (IoT) technology is one of the most popular research topics of recent years and is used in many fields such as health, transportation and education. One of these areas, home automation systems, has become more and more important in human life with the spread of IoT. Problems in security, privacy and interoperability arise because the devices used in IoT are diverse and incompatible with each other in terms of communication protocols and message contents, and hardware resources are often inadequate. In the presented thesis we propose solutions for, I) solving the communication problem between devices using different protocols, II) evaluating the message contents in terms of anomalies on the basis of senders, III) making them suitable for the recipient (anonymisation, enrichment and impoverishment), IV) preventing unauthorized communication of the devices with the world. V) preventing the local devices from attacking a device on the Internet or on the same network; and VI) eliminating vulnerabilities in device update operations. In this context, the platform we developed consists of a server, an Android application as a user interface and a gateway. In the gateway, broker, rule-based message processing, IP-based restriction and multiple SSID approaches are applied together. Thanks to the platform, devices from different manufacturers have been captured while working together or prevented from attacking other devices in the environment where a device does not work as expected, or a target on the Internet.

Key words: gateway, internet of things, network security, data privacy, DDoS attacks
2020, vii + 60 pages.

TEŞEKKÜR

Beraber çalışma ve tecrübelerinden yararlanma fırsatı verdiği ve Yüksek Lisans eğitimim boyunca sabrı, ilgisi, yorumları ve öğütleri ile teknik ve ahlaki olarak gelişmemde sarf ettiği emek ve çabalarından ötürü saygıdeğer danışmanım Dr. Öğr. Üyesi Cengiz Toğay'a sonsuz şükran ve minnetlerimi sunarım.

Tez danışmanlığımı kabul ettiği ve değerli yorumları ile teknik gelişimime katkı sağladığı için saygıdeğer hocam Prof. Dr. Albert Levi'ye ve Bursa Teknik Üniversitesi Bilgisayar Mühendisliği'nde beraber mesai harcadığım ve gerek mesleki gerekse iş ahlakı konusunda önemli tecrübeler kazanmamı sağlayan başta Doç. Dr. Turgay Tugay Bilgin olmak üzere bölümdeki tüm hocalarıma teşekkürlerimi sunarım.

Değerli vaktini ayırarak tez yazımını inceleyen Arş. Gör. Gizem Ortaç'a ve SSLS yapısının tasarımını doktora tez çalışması kapsamında geliştiren ve paylaşan Faik Özgür'e kıymetli desteklerinden ötürü teşekkürlerimi sunar, kendilerine doktora çalışmalarında başarılar dilerim. Çalışmalarım boyunca her daim beni destekleyen sevgili dostum İbrahim Konuk'a teşekkür ederim.

Bu tez çalışması TÜBİTAK-2549 (POLONYA-TÜRKİYE İkili İş Birliği) Projesi kapsamında desteklenmiştir. TÜBİTAK'a projenin finansmanında sağladığı katkılardan ötürü teşekkür ederim.

Hayatım boyunca sevgi ve ilgilerini eksik etmeden her sıkıntımda yanımda olan ve tez yazım sürecinde gösterdikleri sabır ve alakalarından ötürü sevgili aileme sonsuz şükran ve minnetlerimi sunar, bu tez çalışmasını kendilerine armağan ederim.

Ahmet KAŞİF
.../.../2020

İÇİNDEKİLER (TASLAK)

Sayfa

ÖZET i
ABSTRACT ii
TEŞEKKÜR iii
SİMGELER ve KISALTMALAR DİZİNİ iv
ŞEKİLLER DİZİNİ v
ÇİZELGELER DİZİNİ vi
1. GİRİŞ 1
1.1. Genel Bakış 1
1.2. Problem Tanımı 2
1.3. Tezin Hedef ve Katkıları 4
1.4. Tezin Anahatları 5
2. KURAMSAL TEMELLER VE KAYNAK ARAŞTIRMASI 7
2.1. IoT Mimarileri 9
2.2. IoT Güvenlik Ölçütleri 11
2.3. IoT Güvenliğine Yönelik Tehditler 12
2.3.1. Fiziksel (Sensör) Katman Tehditleri 14
2.3.2. Haberleşme (Ağ) Katmanı Tehditleri 15
2.3.3. Uygulama Katmanı Tehditleri 18
3. MATERYAL VE YÖNTEM 20
3.1. Materyal 20
3.1.1. Deney Kurulumu 20
3.1.2. İşletim Sistemi ve Yazılımlar 21
3.2. Yöntem 26
3.2.1. Mimari Tasarım 26
3.2.2. Güvenli Hizmet Katmanı Tanımı 27
3.2.3. Çoklu SSID 30
3.2.4. İletişim Kısıtlaması 31
3.2.5. IoT Cihaz Ekleme 34
3.2.6. MQTT Protokolü ve Cihaz Haberleşmesi 41
3.2.7. Cihaz Güncellemeleri 43
4. BULGULAR 45
4.1. Güvenlik ve Gizlilik 45
4.1. Performans 50
5. TARTIŞMA ve SONUÇ 54
KAYNAKLAR 57
ÖZGEÇMİŞ 60

SİMGELER ve KISALTMALAR DİZİNİ

Simgeler	Açıklama
ms	milisaniye

Kısaltmalar	Açıklama
ACL	Access Control Lists
AIC	Kullanılabilirlik, Bütünlük, Gizlilik
BSSID	Basic Service Set Identifier-Temel Servis Seti Tanımlayıcısı
DoS	Servis Engelleme–Denial of Service
DDoS	Dağıtık Servis Engelleme–Distributed Denial of Service
HMS	Bulut Sunucusu
HTTP	Hyper Text Transfer Protocol
HTTPS	Hyper Text Transfer Protocol Secure
IHG	Ağ Geçidi
IoT	Nesnelerin İnterneti
IEEE	Institute of Electrical and Electronics Engineers
JSON	JavaScript Object Notation
KSA	Kablosuz Sensör Ağlar
MQTT	Telemetri Mesaj İletim Protokolü–Message Queue Telemetry Transport
MS	Milisaniye
M2M	Makine-Makine İletişimi
POC	Kavramın Gerçeklenmesi-Proof of Concept
REST	Representational State Transfer-Temsili Durum Aktarımı
RFID	Radio Frequency Identification
SPOF	Tek Nokta Zafiyeti-Single Point of Failure
SSID	Service Set Identifier-Servis Seti Tanımlayıcısı
SSLS	Güvenli Servis Seviyesi Belirtimi
STS	Saldırı Tespit Sistemi
TCP	Transaction Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
USB	Universal Serial Bus-Evrensel Seri Veriyolu
WLAN	Wireless Local Area Network

ŞEKİLLER DİZİNİ

Sayfa

Şekil 1.1. Yıllara göre internete bağlı IoT cihaz sayısının değişimi 2
Şekil 2.1. Farklı protokollerle haberleşen IoT cihazlar 8
Şekil 2.2. DDoS bant genişliği saldırıları 14
Şekil 2.3. Aradaki adam saldırıları 16
Şekil 2.4. Sybil saldırı modeli 17
Şekil 3.1. Raspberry Pi ile elde edilen ağ geçidi donanımı 21
Şekil 3.2. İstemci/sunucu arasında REST 24
Şekil 3.3. MeteorJS'de veri akışı 26
Şekil 3.4. Ağ geçidi tabanlı IoT mimari tasarım 27
Şekil 3.5. Samsung q60r akıllı tv örnek SSLS 29
Şekil 3.6. OpenWRT çoklu-ssid konfigürasyonu 30
Şekil 3.7. Güvenlik duvarı başlangıç konfigürasyonu 33
Şekil 3.8. Haberleşme kontrol algoritması 34
Şekil 3.9. IoT cihaz ekleme akış diyagramı 36
Şekil 3.10. SSLS giriş ekranı 37
Şekil 3.11. Mobil uygulamada yeni cihaz bildirimi 38
Şekil 3.12. Mobil uygulamada IoT cihaz onaylama ekranı 39
Şekil 3.13. Mobil uygulamada IoT cihaz yönetim ekranı 40
Şekil 3.14. Modern kriptografik algoritmalar 43
Şekil 3.15. Cihaz güncelleme akış diyagramı 44
Şekil 4.1. Kismet ile çevredeki ssid yayınlarının tespiti 47
Şekil 4.2. Ağ geçidi ile iletişim kuran cihazların tespiti 47
Şekil 4.3. Ölçümde kullanılan mesaj yapısı 50
Şekil 4.4. Farklı parametre sayılarında algoritmanın performansı 52
Şekil 4.5. Haberleşme kontrol algoritmasının paket çıktısına etkisi 53

ÇİZELGELER DİZİNİ

Sayfa

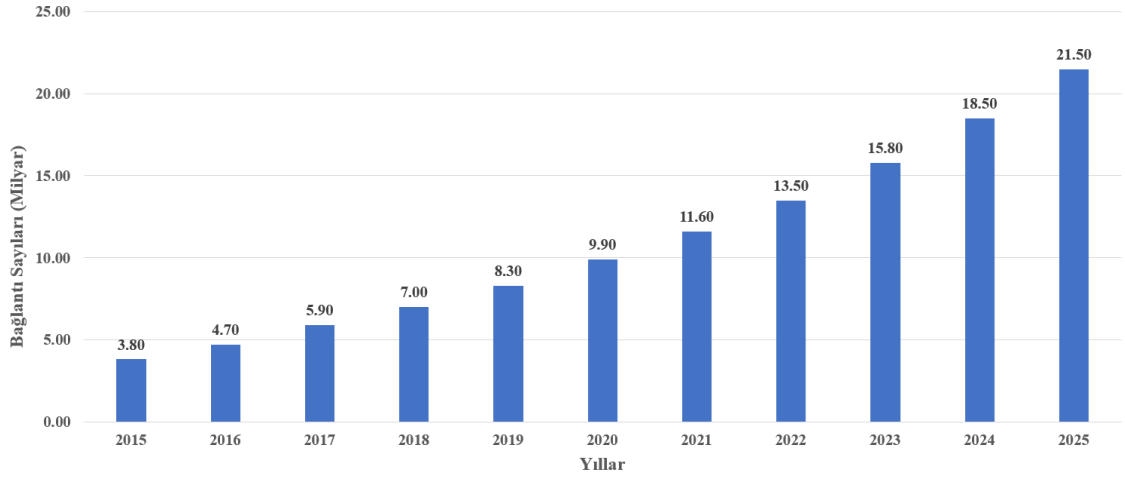
Çizelge 1.1. Tezin katkıları 4
Çizelge 3.1. Raspberry Pi cihaz özellikleri 20
Çizelge 3.2. OpenWRT özellikleri 22
Çizelge 3.3. ExpressJS tabanlı uygulamalarda başlangıç iş akışı 24
Çizelge 4.1. IoT cihazlar için çoklu-ssid ve erişim kısıtlamasının katkıları 45
Çizelge 4.2. IoT güvenlik ölçütlerine sunulan çözümler 48
Çizelge 4.3. Farklı boyutta mesaj içerikleri için operasyon maliyetleri 51

1. GİRİŞ

1.1. Genel Bakış

Bir IoT (Nesnelerin İnterneti) uygulama sahası olarak akıllı ev sistemleri veya diğer adı ile ev otomasyon sistemleri, internete bağlı cihazlar vasıtası ile bir evin güvenliği veya ısıtması gibi amaçlar için uzaktan gözlem, yönetim veya otomasyon hizmetlerinin sağlanmasında kullanılmaktadır. IoT'nin literatüre girişi 1999 yılında Kevin Ashton'ın Procter & Gamble adlı şirkette gerçekleştirdiği sunum ile birlikte olmuştur (Ashton 2009). IoT'den daha eski bir tarihe sahip olan akıllı ev sistemleri, IoT'nin hayatımıza girişi ile hızla yaygınlaşmaya başlamıştır. Cihazların birbirleri ile konuşarak bilgi ürettiği ve kullandığı ilk sistem ise 1991 yılında Cambridge Üniversitesi'nde geliştirilen kahve makinesi yönetim sistemidir (Stafford-Fraser, 1995). Bu uygulama sayesinde ortak bir kahve makinesini kullanan birçok akademisyenin kahve makinesini uzaktan kontrol edebilmesi ve makinenin çok daha yüksek verimlilikle kullanılabilmesi sağlanmıştır.

Geçmişten günümüze insanoğlu üretilen tüm bilginin kaynağı olmuştur. Geçtiğimiz son 20 yılda makine kaynaklı bilginin üretiminde artış kaydedilmektedir. Bu değişim ile birlikte insanlığın rolünün M2M (makine-makine) iletişimini geliştirme, iyileştirme ve bu iletişim sonucunda üretilen verinin işlenmesi olarak değişmesi beklenmektedir (Verma ve ark. 2016). 2018 yılı itibari ile dünya çapında 7 milyar IoT cihazın kullanıldığı ve 2025 yılında bu sayının 21,5 milyar cihaza ulaşmasının beklendiği öngörülmektedir (Lueth 2018) (Şekil 1.1). Görüldüğü üzere hızla yaygınlaşan IoT sistemleri aynı zamanda siber saldırılara karşı güvenliğin sağlanması, cihazların donanımsal çeşitliliğine uygun yazılımların üretilmesi ve veri mahremiyetine yönelik çözümlere olan ihtiyaçları ortaya çıkarmaktadır.



Şekil 1.1. Yıllara göre internete bağlı IoT cihaz sayısının değişimi (https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b (2018)'den değiştirilerek alınmıştır)

1.2. Problem Tanımı

IoT teknolojisi temelde ürün çeşitliliği, kaynak kısıtlı cihazlar ve dinamik ortamlar olmak üzere 3 yapısal özelliğe sahiptir (Oh ve ark. 2017). IoT teknolojisi ortaya atıldığı günden bugüne endüstri alanında büyük bir ilgi görmektedir. Fakat endüstri firmaları değişken pazar koşullarının da etkisi ile piyasaya hızlı ürün sağlamaya büyük önem atfetmektedirler. Bu da üretilen ürünlerin olgunlaşmadan piyasaya sürülmesine ve bakım-destek süreçlerinin gereğinden kısa tutulmasına sebep olmaktadır (Da Xu ve ark. 2014). Böylece piyasadaki aynı sınıf ürünlerde bile birçok farklı teknolojinin kullanılabildiği görülmektedir. Bu ise cihazların makine-makine (M2M) iletişimini sağlaması için gerekli altyapının kurulmasını ve çeşitli güvenlik yamalarının cihazlara ulaştırılmasını zorlaştırmaktadır (Moazzami ve ark. 2017). IoT'de cihazların birlikte çalışabilirliği konusunda yapılan çalışmalarda elde edilen kazanımlar günümüzde oldukça sınırlıdır. Bu nedenle araştırmacıların ve endüstriyel iş ortaklarının daha büyük ilgi ve özellikle ortak inisiyatiflerine ihtiyaç bulunmaktadır. IoT'de kullanılan cihazlar yüksek rekabet koşulları nedeni ile işlemci gücü, hafıza ve pil ömrü gibi özellikleri minimum düzeyde tutularak üretilmektedirler. Bu da geleneksel ağ teknolojileri ile doğrudan bilgisayar ağlarına entegre edildiklerinde çoğu güvenlik ve gizlilik protokolünün uygulanamamasına ve cihazların açık hedefler haline gelmesine sebep

olmaktadır. Bu nedenle kaynak kısıtlı cihazlara özel yaklaşımlar ile yeni güvenlik çözümlerine büyük ihtiyaç bulunmaktadır. Buna yönelik son yıllardaki çalışmalarda donanımsal yük/güvenlik seviyesi dengesinin gözetildiği hafif yöntemlere ağırlık verildiği görülmektedir. Ağ geçitleri ile donanım kısıtlı cihazların sunucuları ile olan iletişiminin güvenli bir şekilde gerçekleştirilmesine yönelik çalışmalar mevcuttur (Toğay ve ark. 2019a). Ancak daha önce önerilen bu sistem internet bağlantısı olmayan ve MODBUS (Hutsing ve ark. 2008) gibi protokolleri temel alan seri iletişim kanalları aracılığı ile veri üreten cihazların sunucular ile olan iletişiminin güvenliğini sağlamayı hedeflemiştir. İnternete bağlanabilen kısıtlı cihazlarında benzer şekilde bir ağ geçidi aracılığı ile güvenliğinin sağlanmasına ihtiyaç bulunmaktadır. Örneğin, bir beyaz eşyanın internet ile olan bağlantısının kurulmasında rol alan modem üretildiğinde TLS 1.0'ı destekliyor ise şu an saldırganların hedefi olmaması ve güncel sunuculara bağlanabilmesi için en az TLS 1.2'yi destekliyor olması gerekmektedir. Ancak, kısıtlı bir donanım olan modem modülünün TLS 1.2'yi desteklemesi sağlanamayabilir. Bu gibi durumların önüne geçilmesi için bir ağ geçidine ihtiyaç bulunmaktadır.

Cihazlar ile olan iletişimin sağlanması sırasında mesajların anonimleştirilmesi, zenginleştirilmesi/fakirleştirilmesi ya da belirli bir formata göre mesaj içeriğine uygunluğunun kontrol edilmesi cihazı ya da mesajı işleyecek uygulamayı korumayı hedeflemektedir. Bu kapsamda sunucu makinesinde kurulu olan aracı sunucusu (broker) ile kontrollerin sağlanmasına yönelik çalışmalar yapılmıştır (Toğay 2018, Toğay ve ark. 2019a). Sunulan tezde, önceki çalışmada sunucuda gerçekleştirilmekte olan kural işleme mekanizmasının ağ geçidinde çalışmasının sağlanması gerekmektedir. Ayrıca, değiştirilmiş bir aracı sunucu uygulaması yerine herhangi bir aracı sunucusu ile çalışacak şekilde sistemin kurgulanmasına ihtiyaç bulunmaktadır. Mesajların belirli kurallara göre işlenmesi sonrasında hedeflere gönderilmesi için kuralların belirli bir formatta cihaz bazında hazırlanması gerekmektedir.

Ele geçirilen IoT cihazları, güvenlik duvarı arkasında olsa dahi ağdaki diğer cihazlar, uygulamalar, sunucular ve kullanıcılar için güvenlik riski oluşturmaktadırlar. Ele geçirilen bu cihazlar ağdaki diğer cihazlara saldırabilir, ağ hakkında bilgi toplayabilir, diğer cihaz ve uygulamaların doğru bir şekilde çalışmasını engelleyebilirler. Dolayısı ile

bu cihazların diğer cihaz ve ağ bileşenlerinden soyutlanması gerekmektedir. Bir yaklaşım olarak sadece güvenlik duvarları kullanılabilir. Ancak güvenlik duvarları sadece IP tabanlı kural yaklaşımlarını uygulamaktadırlar. Dolayısı ile alınacak bir IP ile farklı yetkilere sahip olunabilir. Bu nedenle cihazların diğer ağ bileşenlerinden tamamen soyutlayacak bir mekanizmaya ihtiyaç bulunmaktadır.

IoT'nin yapısal özelliklerinden biri de dinamik ortamlar oluşturmasıdır. Cihazların çalışırlık ve bağlantı durumları ve mesaj içerikleri sürekli değişim göstermektedir. Bu nedenle esneklik ve ölçeklenebilirlik özelliklerine sahip ağ topolojileri ve IoT mimarilerinin tasarlanması ve IoT'nin güvenlik gereksinimlerini karşılayacak kapsamlı güvenlik çözümlerine olan ihtiyaç her geçen gün artmaktadır.

1.3. Tezin Hedef ve Katkıları

Bu çalışmada kaynak/güç kısıtlı IoT cihazlara yönelik tehditler, IoT güvenlik gereksinimleri ve güvenlik açıklarına karşı literatürdeki mevcut çözümler incelenmiş ve IoT yapısal özelliklerine uygun çok katmanlı bir güvenlik ve gizlilik çözümü önerilmiştir. Önerilen çözüm kapsamında, güvenlik ve gizliliğin artırılmasına yönelik yapılanlar Çizelge 1.1'de sunulmaktadır.

Çizelge 1.1. Tezin katkıları

#	**Katkılar**
1	IoT cihazlarının yeni güvenlik gereksinimleri kapsamında geciken ya da gerçekleştirilemeyen yamaların ağ geçidi sayesinde gerçekleştirilmesi sağlanmıştır.
2	IoT cihazlarının diğer ağ bileşenlerinden soyutlanması için çoklu SSID yöntemi ile uygulanmıştır.

Çizelge 1.1. Tezin katkıları (Devamı)

3	IoT cihazlarının, DDoS saldırılarda kullanılmalarının önüne geçebilmek amacı ile cihazların birbirleri ve yerel ağdaki diğer cihazlar ile doğrudan iletişim kurmalarının yanı sıra internete doğrudan çıkışları güvenlik duvarı aracılığı ile engellenmiştir.
4	IoT cihazlarının iletişim içeriğinin tanımlanması ve sunulması için iletişim arayüz tanımı olan SSLS'ler (Secure Service Layer Specification, Güvenli Hizmet Katmanı Tanımı) hazırlanmıştır.
5	IoT cihazlara gelen ve giden tüm trafik, üretici firma tarafından önceden belirlenmesi gereken SSLS'ler aracılığı ile kontrol edilmekte, SSLS dışı trafik engellenerek cihazların olağan dışı trafik oluşturması önlenmektedir.
6	IoT cihazlar ile ağ geçidi arasındaki tüm trafik simetrik şifreleme ile korunmakta ve özet (hash) kullanımı ile verinin bütünlüğü sağlanmaktadır.
7	IoT cihazlarının güncellenmesine yönelik olarak geliştirdiğimiz Ağ geçidi uygulamasının güncelleme dosyasını indirme ve doğrulama sonrasında cihaza yüklenmesi sağlanmıştır.
8	Cihazların ağ geçidindeki aracı yazılımı ile olan iletişiminde konu bazlı yetki kontrolü gerçekleştirilmektedir. Sunulan tezde yetkiler SSLS'lerden elde edilerek otomatik olarak aracı yazılıma uygulanmaktadır.

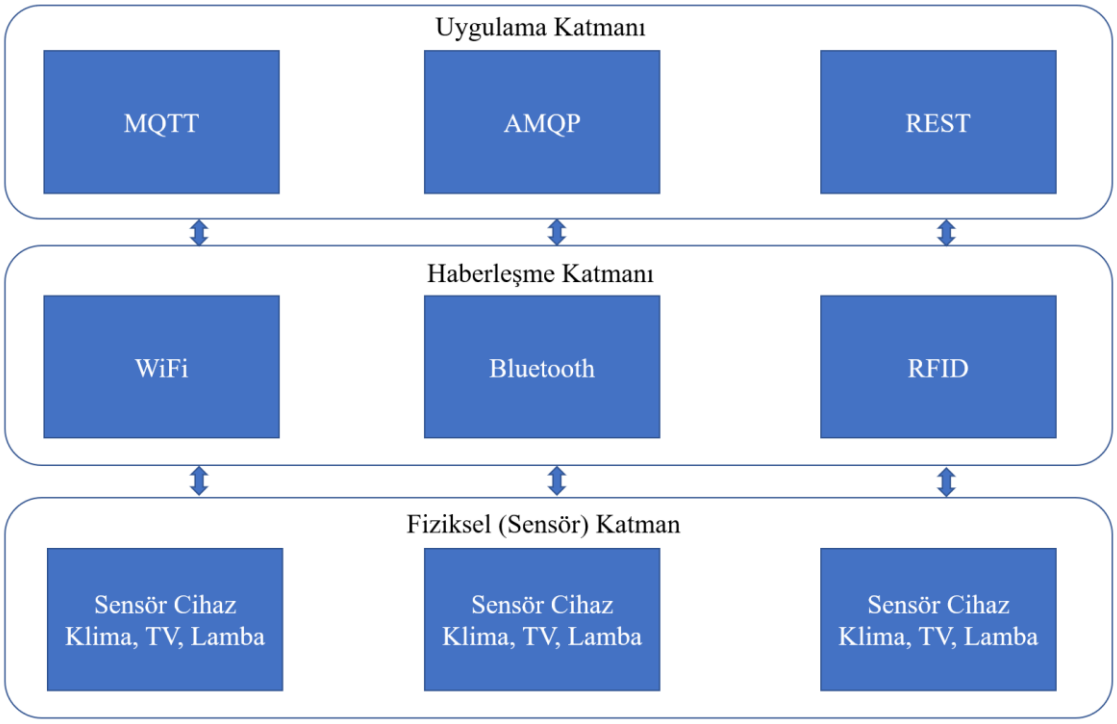
1.4. Tezin Anahatları

Tez çalışması beş bölümden oluşmaktadır. "Giriş" bölümünde IoT teknolojisinin ortaya çıkışı ve yapısal özelliklerinden bahsedilmiştir. Daha sonra bu yapısal özelliklerden kaynaklanan sorunlar açıklanmakta ve tezin hedef ve katkıları kısmında da sunulan çözümlere değinilmektedir. "Kuramsal Temeller ve Kaynak Araştırması" bölümünde, IoT katmanları uygulama katmanı, haberleşme katmanı ve sensör katmanı olarak 3 başlık altında incelenmektedir. Kablosuz Sensör Ağlar'da (KSA) kullanılan geleneksel güvenlik çözümlerinin IoT'de neden yeterli gelmediği ele alınmakta ve IoT'ye özgü güvenlik gereksinimleri açıklanmaktadır. Daha sonra, IoT'de kullanılmakta olan en popüler mimari tasarımlar sunulmaktadır. IoT güvenliğine yönelik tehditlerin çeşitli ağ

katmanlarında sınıflandırıldığı ve çözüm öneri ile bağıntılı bir şekilde ele alındığı bölümde son olarak IoT'nin güncel sorunlarından olan biri olan veri mahremiyetinin sağlanmasına yönelik tehditler ve çözüm önerileri incelenmektedir. "Materyal ve Yöntem" bölümü "materyal" ve "yöntem" olmak üzere iki başlıkta sunulmaktadır. Materyal alt başlığında deney ortamı, deneyde kullanılan donanımlar ve kurulum aşaması anlatılmaktadır. Ayrıca ilgili donanımlarda kullanılan işletim sistemi ve yazılımlar tanıtılmaktadır. Yöntem alt başlığında ise çalışmanın mimari tasarımı, SSLS tasarımı, çoklu SSID tekniği, MQTT protokolü ve cihaz haberleşmesi, güvenlik duvarının kullanımı ile iletişim kısıtlaması, şifreli mesajlaşma ve cihaz güncellemelerine dair çözümler sunulmaktadır. "Bulgular" bölümünde tez kapsamında elde edilen çözüm ile çeşitli zafiyetlere yönelik elde edilen geliştirmeler ve yapılan ölçüm ve gözlem sonuçları paylaşılmakta ve kullanılan çözüm ve algoritmaların çalışma performansları incelenmektedir. Son olarak "Sonuç ve Tartışma" bölümünde ise çalışmanın literatüre katkıları yorumlanmaktadır.

2. KURAMSAL TEMELLER ve KAYNAK ARAŞTIRMASI

Yapısal özelliklerinin de etkisi ile IoT'de birçok amaç için farklı teknolojiler bir arada kullanılabilmektedir. Bu teknolojilerin katmansal olarak sınıflandırılması konusunda yakın zamana kadar uygulamaya geçen bir görüş birliği söz konusu olamamıştır. Bu nedenle de araştırmacılar tarafından uygulamanın bağlamına göre farklı sınıflandırmalar ile değerlendirilebildiği sıklıkla görülmektedir (Alaba ve ark. 2017). Bu kapsamda, katmansal sınıflandırma adına son yıllarda yapılan çalışmalar önem arz etmektedir. Lin ve ark. (2016), IoT'nin uygulama, sensör (algı, fiziksel) ve haberleşme olmak üzere 3 katmanda incelenmesinin faydalı olacağını belirtmişlerdir. Buna göre uygulama katmanı son kullanıcının sistem ile etkileşiminin olduğu ve sistemin bakım, yönetim ve iyileştirme süreçlerinin gerçekleştirildiği en üst IoT katmanıdır. Sensör katmanında elde edilen veriler bu katmanda IoT'nin uygulandığı bağlama göre değişen amaçlar doğrultusunda kullanılmaktadır. Uygulama katmanını oluşturan uygulamalar bulut işlem birimi, ara katman ve M2M haberleşme protokolü kullanılarak oluşturulmaktadır (Yaqoob ve ark. 2017). Uygulama katmanında karşılaşılan en büyük problemlerden biri; IoT'de kullanılan cihaz özelliklerinin ve haberleşme yöntemlerinin çeşitliliğinin çok fazla olması, tekil bir yönetim arayüzünün olmaması ve böyle bir arayüzün etkili ve güvenli bir şekilde geliştirilmesinin zorluğudur (Şekil 2.1). Bu zorluk IoT'nin karmaşık hiyerarşik yapısından kaynaklanmak ile birlikte, cihaz ve hizmet üretici tarafında ortak bir inisiyatifin oluşturulamamasının da önemli etkisi bulunmaktadır.



Şekil 2.1. Farklı protokoller ile haberleşen IoT cihazlar

Sensör katmanı ile uygulama katmanı arasındaki haberleşmenin sağlandığı katman, haberleşme (ağ) katmanı olarak belirtilmiştir. Bu katmanın asli görevi, yerel IoT uygulamalarının bulut tabanlı uygulama katmanı arasındaki trafiğin IoT uygun olarak güvenli, sürekli ve gizli bir şekilde sağlamaktır. Haberleşme katmanında çok çeşitli haberleşme yöntem ve protokolleri kullanılabilmektedir. Bu durum IoT'nin saldırı yüzeyinin genişlemesine ve tehditlerin sayı ve çeşitliliğinde artışa sebep olmaktadır. Geleneksel güvenlik çözümleri IoT'nin kaynak-kısıtlı heterojen cihazlardan mütevellit özgün yapısı nedeni ile yeterli olmamakta ve IoT'ye özgü bir çözümü zorunlu kılmaktadır.

Sensör (algı) katmanı ise çeşitli ortamlarda algılama ve uygulama faaliyetlerinin IoT cihazları aracılığı ile gerçekleştirildiği en alt seviye IoT katmanıdır. Bu katmanda gerçekleştirilen faaliyetler, haberleşme katmanı aracılığı ile yönetim birimlerinin bulunduğu uygulama katmanına iletilmekte; uygulama katmanından gelen komutlar ise ortamda IoT cihazlar aracılığı ile uygulanmaktadır. Bu katmanda değerlendirilen IoT

cihazların en önemli özellikleri arasında düşük güç tüketimi, yetersiz hafıza/işlem gücü ve çok çeşitli haberleşme protokol/yöntemlerinin (MQTT (Locke, 2010), AMQP (Naik, 2017), COAP (http://tools.ietf.org/html/draft-ietf-core-coap-12, 2012)) kullanımı gelmektedir.

2.1. IoT Mimarileri

IoT'de şimdiye kadar bahsedilen araştırma alanlarında kapsayıcı çözümlerin geliştirilmesi ise en başta mimari tasarımın iyi oluşturulmasına bağlı olmakta ve IoT'ye özgü motiflerin değerlendirilmesini zorunlu kılmaktadır. Fakat, IoT gibi çok çeşitli cihazların bir arada kullanılabildiği bir ortamda kapsayıcı yazılım geliştirmek zorlu bir süreçtir. Bu nedenle mimari seçimi konusunda da araştırmacıların ve endüstrinin uygulamaya bağlı ve kapsayıcılıktan uzak günlük çözümleri sıklıkla kullanıldığı görülmektedir. Bununla birlikte, son yıllarda bu konuda ortak çözüm üretmeye yönelik inisiyatiflerin arttığı görülmektedir. Bu kapsamda Lin ve Bergman (2016) tarafından yapılan çalışmada, en sık kullanılan IoT mimarileri "ara katman", "bulut" ve "ağ geçidi" mimarileri olarak belirtilmiştir.

Ara katman mimarileri donanım, ağ katmanı veya işletim sistemi kaynaklı zorlukları uygulama katmanından soyutlayarak geliştiricinin tüm eforunu geliştirilecek arayüze harcamasını temin etmektedir. Ara katman mimarilerinin gereksinimleri işlevsel ve işlevsel olmayan türde olmak üzere 2 sınıfta incelenmektedir (Razzaque ve ark. 2015). İşlevsel olmayan gereksinimler, IoT gereksinimleri başlığında detaylı açıklandığından burada ayrıca bahsedilmemiştir. İşlevsel gereksinimler ise kaynakların otonom keşfi, kaynak yönetimi, veri yönetimi, olay yönetimi, yazılım yönetimi olarak sınıflandırılmaktadır. Bunlardan en önemlileri kaynakların otonom keşfi, olay yönetimi ve kaynak yönetimidir. Sürekli artış gösteren cihaz sayısı ve çeşitliliği sebebi ile insan eliyle IoT kaynaklarının sisteme girdi olarak sağlanması sürdürülebilir bir çözüm değildir. Bu nedenle bir ara katman mimari sistemin kaynak keşfini otonom yapabilmelidir. Hayatın her alanında olmak ile birlikte özellikle sağlık, ulaşım gibi yüksek gecikme süreleri ve hata oranlarına tahammül payının az olduğu alanlarda da kullanılan IoT için veri ve olay yönetiminin, dolayısı ile gerçek zamanlılığın önemi

büyüktür. Zira gerçekleşen her durumda anlık tepki verilmesi, verinin gerekli ön ve son işlem aşamalarından geçirilmesi ve güvenli bir şekilde saklanması gerekmektedir. Bütün gereksinimlerin yanı sıra, uygulanabilir olması için ara katman mimarilerinin güvenlik ve gizlilik eksikliklerinin tamamlanması gerekmektedir. Fakat ara katman mimarilerinin çoğunluğunda hala kimlik doğrulama tabanlı kısmi güvenlik kullanılmaktadır. Bu durum hem IoT'nin gereksinimlerini karşılamamakta hem de cihazlara kaldırabileceğinden fazla işlem yükü oluşturmaktadır. Yazılım katmanlarının sayısının fazlalaşması ise işlem karmaşıklığını arttırmakta ve yine haberleşme performansını etkilemektedir (Lin ve Bergman. 2016).

Bir diğer mimari olan bulut mimarileri, IoT cihazlarının ihtiyaç duyduğu işlem gücünün bulut işlem birimleri tarafından sağlanması fikri üzerine geliştirilmektedir. Bu sayede IoT cihazlarında gözlem, veri toplama, saklama ve işleme süreçlerinde ihtiyaç duyulan işlem gücü sunucu tarafından sağlanmaktadır. Böylece protokol bazlı cihazlar arası iletişim problemlerinin pratikte aşılması sağlanabilmektedir (Toğay ve ark. 2019b). Fakat bulut mimarisinin merkeziyetçi yaklaşımı buluta yapılan saldırılarda tüm IoT cihazlarının etkilenmesini sağlayabilmektedir (Lin ve Bergman. 2016). Ağ geçidi mimarilerinde, IoT'nin ihtiyaç duyduğu gereksinimlerin tamamının yerel ağda karşılanması amaçlanmaktadır. Bu maksatla IoT cihazlara göre donanım kaynakları yönünden zengin olan ağ geçidi cihazların kullanımı önerilmektedir. Ağ geçidi cihazların etkin kullanımı ile farklı üreticilerden birçok cihazın yönetimi, haberleşmesi ve birlikte çalışabilmesi mümkün hale gelmektedir (Lin ve Bergman 2016). Ağ geçidi mimarilerinin bulut mimari ile birlikte kullanımı oldukça yaygındır. Bulutu ağ geçidi için tamamlayıcı olarak kullanan uygulamalarda, bulut sisteminde meydana gelecek bir kesintide ağ geçidi yerel ağda hizmet vermeye devam edebilmekte ve cihazların doğrudan saldırıya maruz kalmasının önüne geçilebilmektedir. Tüm bu faydalar sağlanırken ara katman mimarisindeki yazılım karmaşıklığı oluşmamaktadır. Gajewski ve ark. (2019) çalışması, ağ geçidi ve bulut mimarilerin birlikte etkin kullanımına iyi bir örnek oluşturmaktadır. Çalışmada, ev otomasyon sistemlerinde makine öğrenmesi destekli STS (Saldırı Tespit Sistemi) kullanarak servis sağlayıcı seviyesinde ve yerel ağ geçidi seviyesinde olmak üzere iki seviyeli izinsiz giriş tespit sistemi elde etmişlerdir. Bu sayede hem işlem yükü kullanıcı tarafına dağıtılmakta, hem de servis sağlayıcıda

herhangi bir zafiyet yaşanması durumunda SPOF (Single Point of Failure, Tek Nokta Zafiyeti) oluşmasının önüne geçilmekte, ağ geçidi vasıtası ile yerel güvenlikte herhangi bir zafiyet oluşmamaktadır.

Bull ve ark. (2016) IoT'de SDN (Software Defined Networks, Yazılım Tanımlı Ağlar) mimarisini kullanarak ağ geçitleri ile akış tabanlı bir güvenlik çözümü önermişlerdir. Ağ geçidinde geliştirdikleri kontrolcü sayesinde, cihazlara yönelik Dağıtık Servis Engelleme (DDoS) türünde tehditlerin saptanması ve bertaraf edilmesini sağlamaya çalışmış, bir DDoS türü olan TCP flood saldırısı ile sistem başarımını test etmişlerdir. Çalışmada sistem içerisindeki IoT cihazlara uygulanan DDoS ataklarının tespitinde ağ geçitlerini kullanması açısından faydalı olmak ile birlikte, DDoS harici ataklara karşı kapsamlı bir çözüm getirilmemektedir.

Deniz E. (2019), IoT ağlarına yeni düğüm (cihaz) eklenmesi süreçlerini daha güvenli hale getirecek kötücül düğüm tanıma sistemi geliştirmiştir. Buna göre ağa katılacak düğümler için öncelikle bulutta kayıt oluşturulmakta ve sistem tarafından düğüm için bir şifreleme anahtarı oluşturulmaktadır. Düğüm yerel ağda bağlantı isteğinde bulunduğunda koordinatör düğüm bulut ile bağlantı kurmakta, istekte bulunan düğümün buluttaki şifreleme anahtarını sorgulamakta ve bağlantı bilgilerini bu anahtar ile şifreleyerek iletmektedir. Aynı şifreleme anahtarına üretim zamanında sahip olduğu düşünülen düğüm, gelen paketi çözerek bağlantı bilgilerini elde etmekte ve ağa katılmaktadır.

2.2. IoT Güvenlik Ölçütleri

IoT güvenliği kapsamında ele alınması gereken ikinci önemli husus güvenlik ölçütlerinin belirlenmesidir. KSA'da geleneksel bilgi güvenliği gereksinimlerini belirten AIC üçlemesi (Gizlilik, Bütünlük, Kullanılabilirlik), geleneksel ağlardan daha farklı ihtiyaçları olan IoT uygulamalar için yeterli değildir. Son dönemlerde IoT güvenlik ölçütleri sınıflandırması konusunda ciddi ilerlemeler kaydedilmiştir. Lin ve ark. (2017), IoT güvenlik ölçütlerinin, AIC'ye ek olarak kimlik doğrulama ve yetkilendirme, mahremiyet ve güven kavramlarını da içerecek şekilde genişletilmesini önermektedir.

Gizlilik, sistem bünyesinde elde edilen verinin sadece yetkili kişilerce kullanılabilmesi ve yetkisiz kişilerin eline geçmemesi için gerekli güvenlik önlemlerinin alınması gerekliliği ile ilgilidir. Bütünlük, verinin saklama ve taşıma süreçlerinde değişime karşı korunumunu ifade etmektedir. Uygulama alanına göre önemi değişmemek ile birlikte özellikle sağlık ve ulaşım gibi alanlarda ise son derece önemli ve gerekli olan bir ölçüttür. Zira, sistemin tek bir noktasında oluşacak hasarlı veri hizmetin aksaması veya sistemin/insanların zarar görmesi ile sonuçlanabilmektedir. Kullanılabilirlik, IoT sistemlerde veri akışı sürekliliğini belirtmektedir. Akışta yaşanacak bir aksaklık, bağlı sistem unsurlarının tamamında hizmet verememezlik durumunun oluşmasına neden olabilmektedir. Kullanılabilirlik ölçütü için en büyük tehdit günümüzde oldukça yaygın olan DDoS saldırılarıdır. Kimlik Doğrulama ve Yetkilendirme, sensör (algılayıcı) cihazlardan servis uygulamalarına ve kavramlardan son kullanıcıya kadar, her tarafın kimliklendirilmesi ve sadece kendi alanı ile yetkilendirilmesini ifade etmektedir. Bu sayede izinsiz ve yetkisiz haberleşmenin önüne geçilebilmekte ve ağda meşru haberleşme yapılması sağlanabilmektedir. Mahremiyet, verinin gerek depolama aşamasında gerekse işlenme sürecinde sadece ilgili taraflarca kullanılabilmesidir. Gizlilikten farklı olarak, kullanıcının veri üzerinde hakkı ölçüsünde kontrol sahibi olabilmesi ve izni olmayan verilere erişememesi anlamına gelmektedir. Güven ise tek başına bir ölçüt olmayıp, diğer tüm güvenlik ölçütlerinin birbirleri ile uyumlu ve tamamlayıcı olarak çalışmasını ifade etmektedir. Bunun için IoT'de katman içi ve katmanlar arası güvenlik ve gizlilik protokollerinin birbirleri ile arasındaki güvenin ve son kullanıcı ile IoT arası güvenin sağlanması gerekmektedir (Andrea ve ark. 2015).

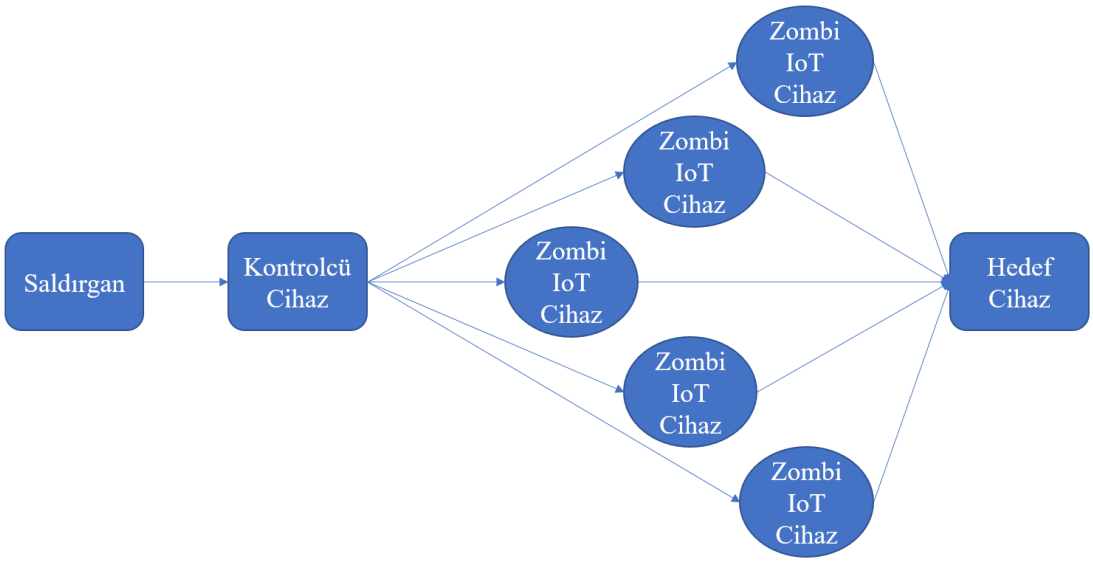
2.3. IoT Güvenliğine Yönelik Tehditler

KSA'ya yönelik tehditler, IoT'yi de doğrudan ilgilendirmek ile birlikte, cihaz çeşitliliği ve bu cihazlardaki donanımsal zafiyetler, ağ katmanlarındaki zafiyetler, bilinçsiz kullanıcı hataları, yazılımsal zafiyetler gibi birçok parametre sebebi ile IoT çok daha geniş bir saldırı yüzeyine sahiptir (Khan ve Salah. 2017). Parametre sayısının fazla olması, saldırı türlerinin sınıflandırılması ihtiyacını ortaya çıkarmıştır. Araştırmacıların genel görüşü, saldırıları IoT katmanları temelinde sınıflandırmak yönündedir. Bu

kapsamda IoT'ye yönelik tehditler fiziksel (sensör) katman, haberleşme (ağ) katmanı ve uygulama katmanına yönelik tehditler olmak üzere 3 sınıfta incelenmektedir.

Her 3 katmanın ortak problemi olan Servis Engelleme (DoS) saldırıları ise IoT için en önemli tehditlerden olması sebebiyle ayrı bir kapsamda değerlendirilmektedir. DoS veya günümüzde saldırganlar tarafından daha popüler hale gelen DDoS saldırıları, son yıllarda sıklıkla kullanılmaya başlanan bir saldırı türü olup, belirli bir sistem veya kaynağa yönelik giden ve gelen tüm trafiğin engellenmesi veya bozulması amacı ile yapılan saldırıların tümüne verilen isimdir (Gasti ve ark. 2013). Teknik olarak kolay fakat etkisi çok büyük olan DDoS saldırılarına yönelik önlemler gelişmiş Saldırı Tespit Sistemleri (STS) gibi çözümlere olan ihtiyacı arttırmaktadır. Saldırganlar arasında en popüler yöntem olan bant genişliği hedefli saldırılarda saldırgan, zombi cihazlar, hedef servis veya kaynak olmak üzere 3 taraf bulunmaktadır (Şekil 2.2). Bu türdeki saldırıların önkoşulu saldırganın hedef cihaza saldırıda kullanacağı cihazları belirlemesi ve bu cihazları zombi cihaz olarak kullanmak üzere ele geçirmesidir. Asıl saldırı başlatıldığında zombi cihazlar aynı anda hedefe yönelik bağlantı isteğinde bulunmakta ve hedef ortamda hafıza taşması, güç yetersizliği ve/veya işlem gücünün tamamının tüketilmesi gibi sonuçlar ortaya çıkmaktadır. IoT cihazlar, kısıtlı kaynakları ile internet ortamında çok kolay ele geçirilebilir ve sayıca çok olmaları sebebi ile DDoS saldırılarında sıklıkla zombi cihazlar olarak değerlendirilmektedirler.

Swan ve ark. (2012) kablosuz ağlardaki bant genişliğinin kullanıcı itibar seviyesine göre ayarlanabilir servis kalitesi (QoS) ile sağlayabilmek amacı ile çoklu SSID (Service Set Identifier) kullanımını öneren bir patent geliştirmişlerdir. Ağa bağlanacak olan cihazların bant genişliği ve güvenlik gereksinimlerine göre QoS'nin, sürekli veya süreksiz, ayrıca dinamik olarak sağlanması, böylece ağdaki cihazların mahremiyet ve güvenlik seviyelerinin özelleştirilmesi önerilmiştir. Birden fazla SSID kullanımı ile de bir cihaza birden fazla QoS hizmetinin verilebilmesi mümkün hale gelmektedir. Bu yöntem, IoT cihazların kullanıcı cihazlarından soyutlanması ve doğrudan internet bağlantılarının kesilmesiyle desteklenerek, cihazların uzaktan ele geçirme ve bozma saldırılarına maruz kalmalarının engellenmesi yönünde tez kapsamında çalışma yapılmıştır.



Şekil 2.2. DDoS bant genişliği saldırıları

DDoS saldırılarına karşı çeşitli çözümler önerilmektedir. Çoğunluğu saldırıya göre çok daha maliyetli olan çözümlerin yanında, günümüzde daha az maliyetli çözümler de kullanılmaya başlanmıştır (Zhang ve Green, 2015). Bunlardan biri geri-yayılım tekniği olup, herhangi bir yönlendiricinin bir alan adına yönelik bant genişliğinin aşıldığını veya devam eden saldırı bulunduğunu fark etmesi durumunda ilgili alan adına paket gönderen adresleri diğer yönlendiriciler ile paylaşarak paketlerin saldırgana geri gönderilmesini hedeflemektedir. Bu sayede saldırganın kendisini servis dışı bırakmakta ve hedefin zombi cihazlar tarafından daha fazla meşgul edilmesinin de önüne geçmektedir (Gasti ve ark. 2013).

2.3.1. Fiziksel (Sensör) Katman Tehditleri

IoT cihazlarının bulunduğu fiziksel katmandaki tehditler, cihazların donanımsal olarak ele geçirilmesi, aktif-pasif dinleme faaliyetleri ve haberleşmenin sekteye uğratılması (Jammer saldırıları) gibi çeşitli saldırılardan oluşmaktadır. Kablosuz ağlarda dinleme faaliyetleri, ortam hakkında bilgi toplama ve saldırıların sonraki aşamalarının planlanması amacı ile gerçekleştirilmekte ve saldırıların ilk adımını teşkil etmektedir.

Dinleme faaliyetleri, aktif tarama ve pasif tarama olmak üzere 2 türde incelenmektedir (Anh ve Shorey 2005). Aktif taramada ilgili ağa paket gönderimi yapılmakta ve cevaplar analiz edilmektedir. Bu şekilde ağdaki açıklar ile ilgili çok faydalı bilgiler elde edilebilmektedir. Aktif taramanın en önemli dezavantajı, SSID yayını yapmayan kapalı ağlarda kullanılamıyor oluşudur. Ayrıca aktif tarama yapıldığında saldırganın ağ adaptörleri ifşa olmaktadır. Bu durumda eğer ilgili güvenlik önlemleri alınmış ise saldırgan ağda asli saldırıyı gerçekleştirmeden yakalanmasına neden olabilmektedir. Pasif dinleme faaliyetlerinde saldırgan ağ adaptörünü RF-MON moduna almakta ve düğümler ile yönlendirici arasındaki trafiği sadece uzaktan yakalamaktadır. Kablosuz haberleşmenin yapısı sebebi ile pasif dinleme faaliyetlerinin fark edilmesi ve saldırganın yakalanması çok zor olmaktadır. Ağda şifreli haberleşmenin kullanılması, paket bütünlüğünü doğrulayıcı tedbirlerin alınması (özet değerleri), mümkünse kapalı SSID ile haberleşmenin sağlanması en önemli tedbirler olarak belirtilmektedir.

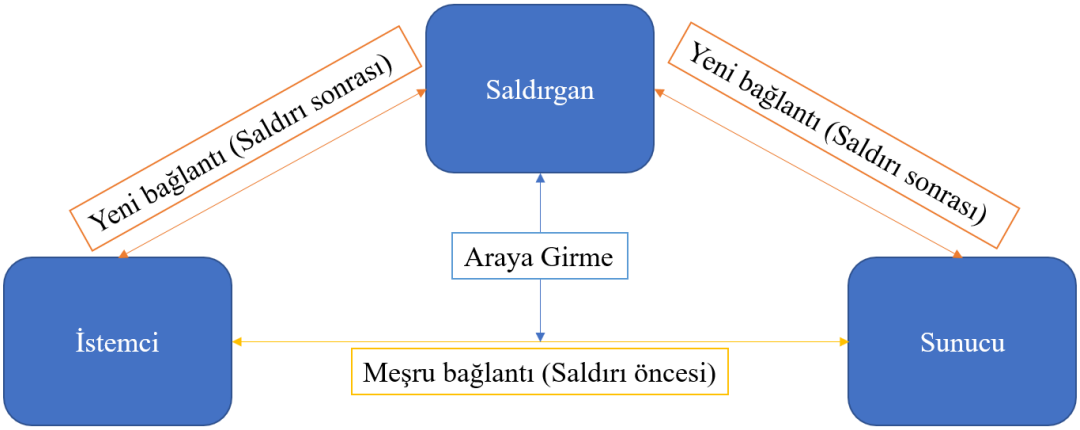
Jammer saldırıları, kablosuz ağlarda fiziksel olarak ağ yayınlarını engellemek ve IoT sistemi hizmet veremez duruma getirmek için kullanılır. “Continuous Jammer”, “Deceptive Jammer”, “Random Jammer”, ve “Reactive Jammer” olmak üzere 4 farklı türde incelenmektedir (Wu ve ark. 2006). Continuous Jammer, kanalın sürekli radyo sinyalleri ile meşgul edilmesi ve meşru sinyallerin kanala ulaşamamasını hedefler. Deceptive Jammer ile sürekli ve aynı boyutta paket gönderimi yapılır. Alıcı, meşru bir haberleşme olduğuna ikna edilerek paket okuma durumunda tutulur. Paketler gönderilmeye devam ettiği sürece cevap gönderemeyen alıcı, saldırganın gönderdiği paketleri okumaya devam ettiğinden, diğer taraflar ile haberleşemez. Random Jammer tekniğinde rastgele zamanlarda sinyal gönderilir ve saldırganın güç kaynağı sınırlı ise kullanılır. Reactive Jammer kanalda haberleşme yok iken uyku vaziyetine geçerken, haberleşme başlayınca etkinleşir. Bu davranışı sebebi ile tespit edilmesi de zordur.

2.3.2. Haberleşme (Ağ) Katmanı Tehditleri

Haberleşme katmanı IoT cihazları ve uygulama katmanı arasındaki haberleşmenin sağlandığı katmandır ve haberleşmenin türüne (kablolu, kablosuz) göre farklı tehditler

oluşabilmektedir. IoT ortamlarda cihazların mobil özelliklerinin olabilmesi gibi sebeplerle genellikle kablosuz haberleşme kullanılmaktadır.

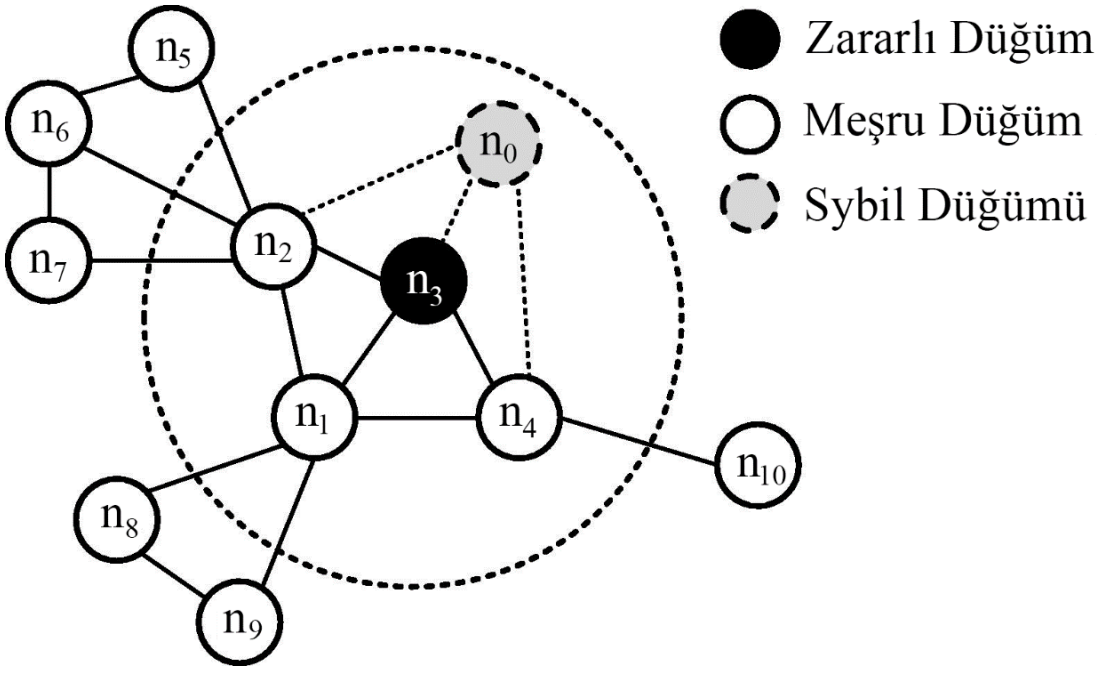
Aradaki adam (Man In The Middle, MITM) saldırılarında, saldırgan fiziksel veya yazılımsal yöntemler ile haberleşen iki tarafın arasına meşru bir kullanıcı gibi girmektedir (Şekil 2.3). ARP poisoning (ARP zehirleme), DNS poisoning (DNS zehirleme), ICMP yönlendirme ve HTTPS dinleme gibi yöntemler kullanılarak ağ katmanlarının çeşitli yüzeylerinde uygulanabilmektedir (Nayak ve Samaddar, 2010). Daha sonra elde edilen bilgi, ilgili ağda gözlemleme (sniffing) veya haberleşen taraflardan birine etki etme (spoofing) amacı ile kullanılabilmektedir. MITM'e karşı oluşturduğu bu geniş saldırı yüzeyinden dolayı tek bir çözümle koruma sağlamak zordur. Bu nedenle MITM saldırılarında özel durumlara yönelik çözümler geliştirilmektedir. Nayak ve Samaddar (2010), ARP poisoning saldırılarına karşı kullanıcı cihazlarının ARP önbelleklerini periyodik olarak güncellemelerini içeren bir çalışma önermektedir. Bu sayede saldırganların cihazlardaki ARP önbelleğini istedikleri adresler ile değiştirip cihazın haberleşmesini ele geçirememeleri hedeflenmiştir. Cho ve Jeon (2016), haberleşmeye başlayacak olan iki taraf arasındaki anahtar takası öncesinde zaman damgalı tek kullanımlık şifre (OTP) kullanarak, MITM saldırılarının önüne geçmeyi hedeflemiştir. Önerilen yöntemde gönderici ve alıcı taraflar aynı zaman damgası ile OTP üretmekte ve alıcı göndericinin ürettiği OTP'yi doğrulamaktadır. Bu sayede saldırganın anahtar takası öncesinde açığa çıkması mümkün hale gelmekte ve güvenli anahtar takası sonrası tarafların meşru iletişim kurabilmesi sağlanmaktadır.



Şekil 2.3. Aradaki adam saldırıları

Haberleşme katmanında sıklıkla görülen bir diğer saldırı türü olan Sybil saldırıları, Sybil düğümler kullanılarak ağda haberleşmenin engellenmesi ve koruma mekanizmalarına duyulan güvenin yok edilmesine neden olur (Sarigiannidis ve ark. 2015). Sybil düğüm, ele geçirilmiş veya ağa herhangi bir yolla sızmayı başarmış zararlı düğümlere verilen isimdir. Bu düğümler, ağ içerisinde oluşturdukları sahte kimlikleri ağdaki meşruiyeti kullanarak illegal operasyonların yürütülmesi veya sonraki saldırıların önünün açılması amacı ile kullanır ve ağda etki (influence) üstünlüğünü elde etmeye çalışırlar (Şekil 2.4).

Sybil saldırılarına engel olmak için ağdaki kimliklerin itibar seviyesine göre yetkilendirilmesi, yüksek kimlik oluşturma maliyetlerinin kullanımı ile birçok sahte kimlik oluşturulmasının saldırgan açısından kullanışsız hale getirilmesi, ağdaki etki üstünlüğünün meşru olmayan taraflara geçişinin engellenmesi amacı ile ağa yeni eklenecek düğümlerin çeşitli doğrulama mekanizmaları ile ağ içi düğümler tarafından onaylanması gibi çözümler önerilmektedir. Ayrıca günümüzde STS aracılığı ile Sybil saldırılarına karşı koruma seviyesi etkin ve sürekli olarak arttırılabilmektedir.



Şekil 2.4. Sybil saldırı modeli (Sarigiannidis ve ark. 2015)

2.3.3. Uygulama Katmanı Tehditleri

IoT cihazlardan elde edilen ham veriler uygulama katmanına iletilmekte ve istenilen ortama göre işlenmektedir. Uygulama katmanında kullanılan protokollerdeki zafiyetler ve yönetim panellerindeki yazılımsal zafiyetlerle birlikte, kullanıcı kimlik bilgilerinin çalınması ve veri mahremiyeti gibi çözüm ihtiyacı bulunan çeşitli sorunlar görülmektedir.

Uygulama katmanında IoT için en önemli problemlerden biri veri mahremiyetidir. IoT, verinin yoğun bir şekilde üretilip kullanıldığı bir alan olup bu nedenle ham ve işlenmiş verilere sadece yetkili ve meşru tarafların erişim sağlaması ve veri sahibinin rızası olmadan verilerin kullanılmasının önüne geçilmesi açısından büyük önem taşımaktadır. Günümüzde veri mahremiyeti üretici firmalar tarafından cihazlara entegre edilen yazılımlar vasıtası ile şahıslar veya kurumsal firmalar tarafından sağlanan hizmetler aracılığı ile veya IoT güvenlik açıklarından faydalanan saldırganlar tarafından ihlal edilebilmekte ve ticari amaçlar ile kullanılabilmektedir. Bu nedenle ham verinin IoT cihazdan bulut sunucusuna kadar süren yolculuğunda ve bulutta depolanışına kadar tüm süreçlerin ayrı ayrı gizlilik ihlallerine karşı korunması gerekmektedir. Depolama ve trafiğin şifreli hale getirilmesi bu konudaki en önemli çözümlerden biridir. Fakat IoT cihazların kısıtlı kaynaklara sahip olması sebebi ile güçlü şifreleme tekniklerinin kullanımı çoğu zaman tercih edilmemektedir. Bu kapsamda Katagi ve Moriai (2008) IoT cihazlarda hafif şifreleme tekniğinin (lightweight cryptography) kullanımını önerilmişlerdir. Hafif şifrelemede simetrik anahtar kullanılmakta ve işlem yükü düşük algoritmaların kullanımıyla IoT cihazlara uygun şifreleme çözümü sunulması ve mahremiyetin geliştirilmesini amaçlanmaktadır.

Uygulama katmanındaki bir diğer sorun yazılımsal açıklardır. Yazılım geliştirme süreçleri gün geçtikçe daha karmaşık ve çok katmanlı süreçler halini almaktadır, bu da yazılımların yönetim ve bakımını zorlaştırmaktadır. 2018 yılında IoT'de kullanılan yazılımlarda açık kaynaklı yazılım oranı %77'dir (https://www.helpnetsecurity.com/2018/05/22/open-source-code-security-risk/, 2018).

Açık kaynak yazılımlarda en büyük problemlerden biri her yazılımın diğer yazılımlar ile bağımlılık ilişkisi kurmasıdır. Dolayısı ile kullanılan bağımlılıkların sahip olduğu yazılımsal zafiyetler, kullanan yazılımların da zafiyeti haline gelebilmektedir. Buna karşı çeşitli paket yönetim sistemleri son yıllarda yapay zekâ kullanımı ile yazılım zafiyetlerini geliştiricilere raporlamaya başlamışlardır. Örnek olarak uygulamalar tarafından çok yaygın olarak kullanılan güvenlik yazılım kütüphanesi OpenSSL'deki bir zafiyetten dolayı ilgili versiyonu kullanan sunucuların belleğinden veri sızıntısının olmuştur (https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0160, 2013).

3. MATERYAL VE YÖNTEM

3.1. Materyal

Materyal bölümünde çalışmanın gerçek dünyada uygulanması için gerekli donanım konfigürasyonlarının seçimi ve kurulum süreci anlatılmakta ve kullanılan işletim sistemi ve yazılımlar tanıtılmaktadır.

3.1.1. Deney Kurulumu

IoT cihazların kısıtlı kaynaklara sahip olması nedeni ile güvenlik çözümlerinin görece daha güçlü donanımsal özelliklere sahip cihazlar olan ağ geçidi cihazlar üzerinden yönetilmesi gerekmektedir. Bu amaçla ağ geçidi rolünde Raspberry Pi (RPi) 3B+ cihazının kullanımı değerlendirilmiştir (Harrington, 2015). RPi ARMv8 mimarili 64-bit 1.4 GHz işlemci, 1 GB bellek, kamera, ekran, harici görüntü çıkışları, USB (Universal Serial Bus- Evrensel Seri Veriyolu) bağlantı girişleri, geliştirme pinleri ve kablosuz ağ adaptörüne sahip mini bir bilgisayar olup, cihazın detaylı özellikleri Çizelge 3.1'de sunulmaktadır. Sabit depolama birimi bulunmayan RPi'da bu amaçla kullanılmak üzere SD kart okuyucu yuvası bulunmaktadır. Ayrıca RPi birçok farklı Gömülü Linux (Embedded) dağıtımı ile uyumlu çalışmaktadır.

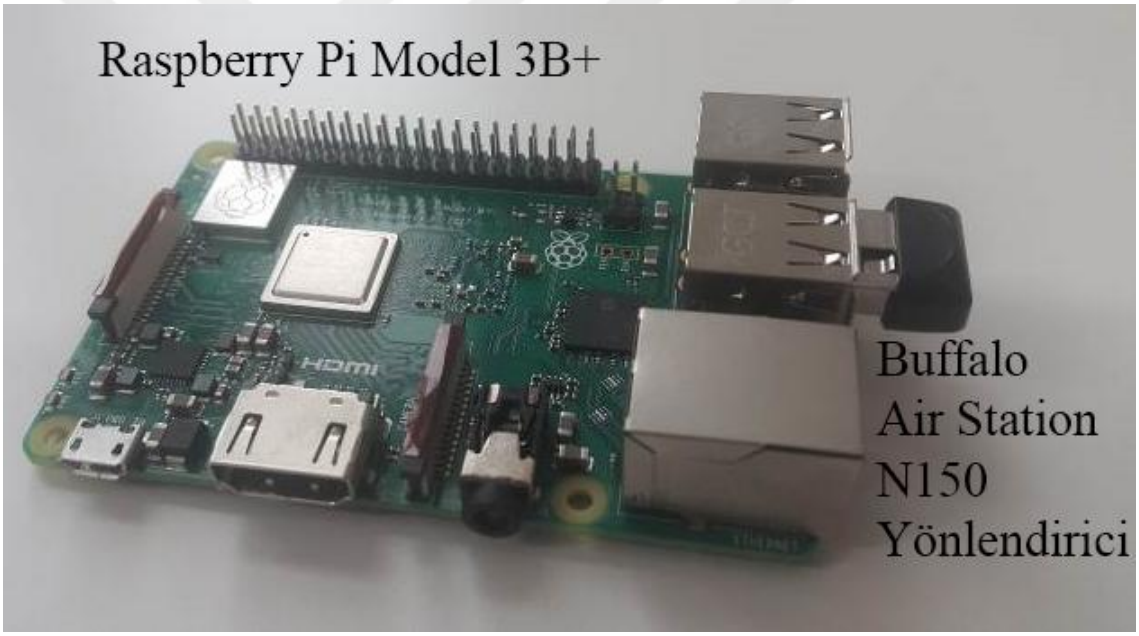
Çizelge 3.1. Raspberry Pi cihaz özellikleri (https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus, 2019)

#	Cihaz	Marka ve model
1	İşlemci	Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
2	Ram	1GB LPDDR2 SDRAM
3	Genişletme yuvaları	40-pin GPIO başlığı
4	Görüntü çıkışı	HDMI
5	Kamera desteği	RPi modüler kamera yuvası
6	Depolama	1 Mikro SD yuvası

Çizelge 3.1. Raspberry Pi cihaz özellikleri (https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus, 2019) (Devamı)

7	Güç	5V/2.5A DC güç girişi
8	Kablosuz Adaptörü	2.4GHz ve 5GHz IEEE 802.11.b/g/n/ac kablosuz LAN, Bluetooth 4.2, BLE

Çalışma kapsamında RPi donanımı OpenWRT işletim sistemi ile birlikte kullanılmıştır. RPi cihazının mevcut donanımları çoklu SSID'nin uygulanmasını desteklemediğinden, çoklu SSID desteği bulunan Buffalo AirStation N150 model USB kablosuz yönlendirici (router) RPi ile birlikte kullanılmıştır (Şekil 3.1).



Şekil 3.1. Raspberry Pi ile elde edilen ağ geçidi donanımı

3.1.2. İşletim Sistemi ve Yazılımlar

Deneyde kullanılacak işletim sistemi olarak OpenWRT ve Raspbian olmak üzere 2 seçenek değerlendirilmiştir. OpenWRT gömülü sistemler için açık kaynaklı yazılımlar vasıtası ile kablosuz ağ iletişimini desteklemek üzere tasarlanmış olan Linux tabanlı bir

işletim sistemi sistemidir (https://openwrt.org/, 2019a). Ağ geçidini donanımsal olarak meydana getiren RPi cihazlar için de geniş bir paket desteği ve dökümantasyonu bulunan OpenWRT'nin temel özellikleri Çizelge 3.2'de görülmektedir. OpenWRT, daha çok yönlendiricilerde kullanılmakta ve yerel ağ güvenliğine sağladığı katkı nedeni ile tercih edilmektedir. Yaptığımız gözlemlere göre işletim sisteminin diskte kapladığı alan ilk yükleme sonrası 100MB civarında olup, bu rakam çeşitli üçüncü parti yazılımların eklenmesi ile 300MB seviyelerine ulaşmaktadır. Hafıza kullanımı ağdaki trafiğe bağlı olarak değişmekle birlikte cihaz yük altında değilken 30MB civarındadır. Ağ yönetimi konusunda da birçok arayüzün tanımlı olduğu işletim sisteminde, rakipleri kullanıldığında en baştan tasarlanması gereken birçok model hazır gelmekte ve işletim sistemindeki mevcut yüksek seviye arayüzler vasıtası ile en düşük seviyede özelleştirilebilmektedir. Örneğin, güvenlik duvarı yazılımı bu modellerden sadece biridir ve ağ yönetimi konusunda yönetim karmaşıklığını oldukça azaltmaktadır. Cihazların birbirleri ile haberleşmesinin kesilmesi, yerel ağ yönetimi ve yerel ağların internet çıkışlarının güvenlik duvarı ile kontrolü OpenWRT yazılımları sayesinde mümkün olabilmektedir.

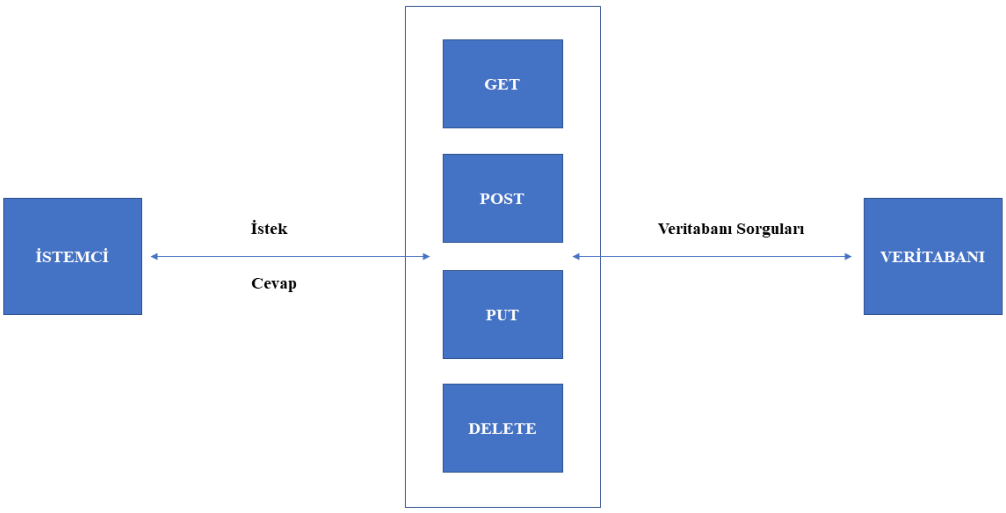
Çizelge 3.2. OpenWRT özellikleri

#	**Özellikler**
1	Özgün ve kapsamlı paket yönetim sistemi (opkg)
2	Donanımda düşük kaynak gereksinimi
3	Geniş donanım uyumluluğu
4	Web tabanlı yüksek seviye kullanıcı arayüzü
5	Modüler tasarımı ile kapsamlı özelleştirme

Raspbian işletim sistemi, Raspberry Pi ürününün geliştiricileri tarafından desteklenen bir işletim sistemi olup, kullanıcı arayüzü ile birlikte gelmektedir (Harrington, 2015). Dolayısı ile geliştiriciler, cihazı bir ekran ile birlikte kullandıklarında masaüstü bilgisayar ortamından farksız bir geliştirme deneyimi elde edebilmektedirler. Fakat bu kullanıcı arayüzünün ve kurulumla birlikte yüklü gelen yazılımların işletim sistemi üzerinde oluşturduğu yük oldukça fazladır. İşletim sisteminin ilk kurulum sonrası diskte

kapladığı alan yaklaşık 4GB'tır. Raspbian, Debian tabanlı bir gömülü Linux işletim sisteminin klonu olduğundan yönlendirici cihazlara yönelik yazılımlar içeren bir paket yönetim sistemi bulunmamakta, yönlendirici problemlerine has çözümlerin uygulanması da sıklıkla mümkün olmamaktadır. Sonuç olarak, donanım olarak RPi ürününün kullanılması ve bu üründe işletim sistemi olarak OpenWRT'den yararlanılmasının hızlı ve istikrarlı bir araştırma-geliştirme sürecinin sağlanması için uygun olacağı değerlendirilmiştir.

Tez çalışması kapsamında geliştirilen mobil kullanıcı uygulaması, IHG kontrolcü uygulaması, HMS REST uygulaması gibi bileşenlerin tamamı JavaScript dilinde NodeJS tabanlı uygulamalar olarak hazırlanmıştır. NodeJS, JavaScript dilinde olay tabanlı programlama yaklaşımının kullanıldığı ölçeklenebilir ve asenkron haberleşme arayüzüdür (Tilkov ve Vinovski, 2010). Bu sayede NodeJS ile geliştirilen uygulamalar birçok alıcıya eş zamanlı hizmet verebilmektedir. NodeJS ayrıca dünyanın en büyük paket yönetim sistemlerinden biri olan npm ekosistemini içinde barındırmaktadır. Dünyanın her yerinden açık kaynak topluluğu tarafından geliştirilen milyonlarca paket, kütüphane veya kod parçası npm aracılığı ile yine topluluğun kullanımına sunulmaktadır. Bu şekilde npm gibi paket yönetim merkezleri aracılığı ile projelere eklenen yazılımlara "bağımlılık" adı verilmektedir. Bu bağımlılıkların güvenlik, birlikte çalışmaya karşı uyuşmazlık kontrolü ve versiyon güncellemeleri npm tarafından otomatik olarak kontrol edilmekte ve geliştirici tarafından gerekli tedbirlerin alınması sağlanmaktadır. Dünyanın her yerinde açık kaynaklı yazılımlar ile kurumsal firmalardan bireysel geliştiricilere kadar yazılım geliştirme süreçlerine dair birçok problem açık kaynak felsefesi vasıtası ile çözülmekte ve çözümler yine açık kaynak olarak piyasaya sunulmaktadır. Sunucu uygulamaları, Fielding ve Taylor (2002) tarafından önerilmiş olan REST (Representational State Transfer-Temsili Durum Aktarımı) mimari kullanılarak geliştirilmiştir. REST mimarisinin sunduğu avantajlardan bazıları, istemci ve sunucu tarafların birbirlerinden soyutlanması, durumdan bağımsız sorgulama ve tüm istemciler için tek sunucu arayüzü kullanımıdır. Bu sayede istemci uygulamalar ile sunucu arasındaki birlikte çalışabilirlik sorunları aşılabilmektedir. REST uygulamaları için akış şeması Şekil 3.2'de görülmektedir.



Şekil 3.2. İstemci/sunucu arasında REST

Tez kapsamında elde edilen REST uygulamalarının geliştirme sürecinde ExpressJS kütüphanesi kullanılmıştır (Mardan, 2014). ExpressJS, sunucu tarafında yazılım geliştirme amaçlı kullanılan, NodeJS tabanlı bir web kütüphanesidir. ExpressJS'in NodeJS'e göre tercih edilmesindeki en önemli sebep, sağladığı işlevsel kod parçaları ile daha yüksek seviyeli bir yazılım geliştirme ortamı oluşturmasıdır. Bu da geliştirme sürecini daha yönetilebilir ve hızlı bir hale getirmektedir. ExpressJS'de MVC özelliği olmamak ile birlikte veri tabanı, ara katman ve arayüz kütüphanelerini içeren çeşitli çözümlerle birlikte kullanılarak bu özelliği kazanması da sağlanabilmektedir. ExpressJS kullanılan uygulamalarda bir başlangıç noktası bulunmaktadır (entry point). Bu noktaya gelindiğinde, Çizelge 3.3'te sunulan iş akışı işletilmektedir.

Çizelge 3.3. ExpressJS tabanlı uygulamalarda başlangıç iş akışı

#	**İş parçacıkları**
1	3.parti yazılımlar ve projedeki diğer modül, kod parçacıkları ve veri modelleri eklenir.
2	ExpressJS sayfa taslakları ve uygulama konfigürasyonları yüklenir.

Çizelge 3.3. ExpressJS tabanlı uygulamalarda başlangıç iş akışı (Devamı)

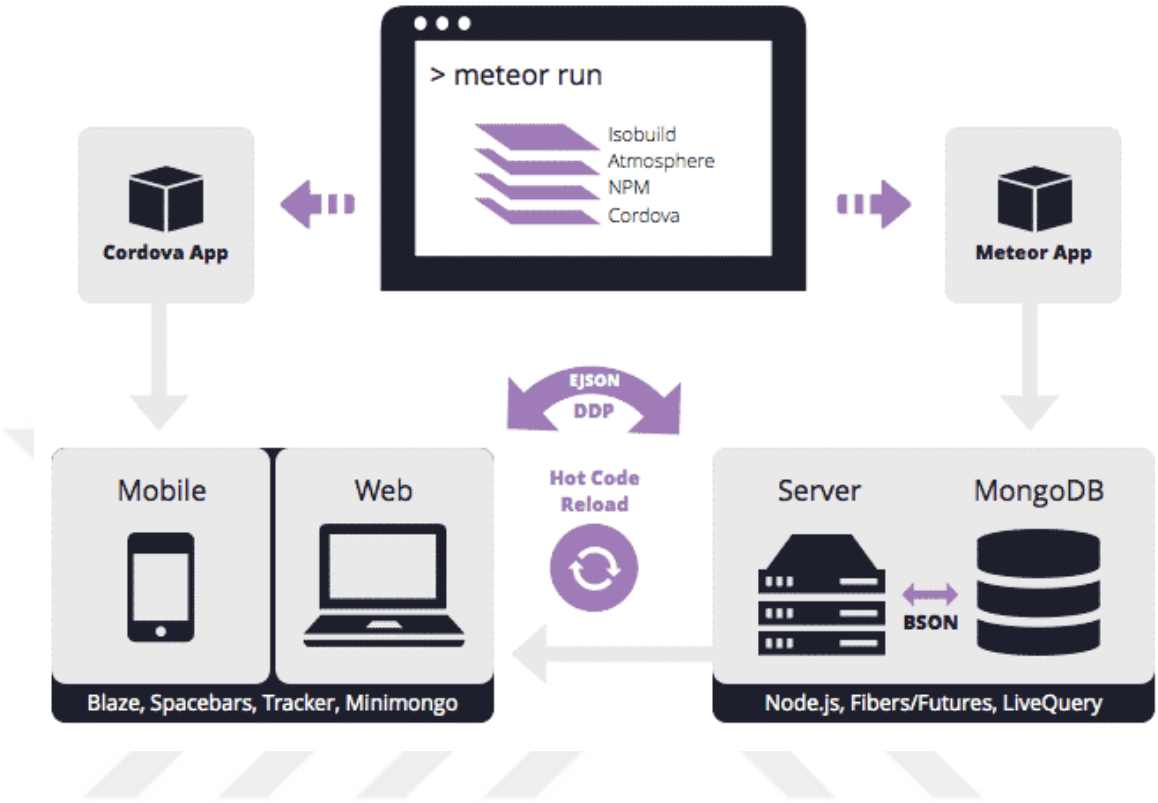
3	Hata yakalayıcılar, çerezler, statik dosyalar gibi ara katman tanımlamaları yapılır.
4	Rotalar belirlenir.
5	Veri tabanı bağlantısı gerçekleştirilir (MongoDB, Redis, MySQL).
6	Uygulama başlatılır.

Başlangıç iş akışı tamamlandıktan sonra ExpressJS uygulaması, belirtilen rotalar üzerinden gelen istekleri dinlemektedir. Herhangi bir istek yakalandığında ise tanımlı fonksiyonlar çalışmakta veya ara katman (middleware) yazılımlar tetiklenmekte, elde edilen sonuç ise istekte bulunan tarafa iletilmektedir.

Son kullanıcının etkileşim sağladığı mobil uygulama bir başka açık kaynak Web geliştirme arayüzü olan MeteorJS kütüphanesi ile geliştirilmiştir. MeteorJS, kullanıcı ve mobil tarafların tamamında hem önyüz hem de sunucu tarafında sadece JavaScript kullanarak geliştirme yapmaya imkân sağlayan NodeJS tabanlı bir JavaScript kütüphanesidir (Strack, 2015). Bu amaçla piyasaya ilk çıktığı 2012 yılında hem kurumsal firmalar hem de bireysel geliştiriciler tarafından büyük bir ilgi ile karşılanmıştır ve bugün de aynı ilgi ile aktif kullanımına devam edilmektedir. MeteorJS'de reaktif programlama modeli kullanılmaktadır. Bu sayede istemci ve sunucu sadece verinin gösterimini değil aynı zamanda veride meydana gelen değişimleri de dinlemektedir. Nesnelerin güncel durumları istemcilere kolaylıkla aktarılabilmekte ve uygulama geliştirme süreci oldukça kısalmaktadır (Şekil 3.3).

MeteorJS'in bir diğer özelliği, mobil uygulama geliştiriciler için sağladığı Apache Cordova entegrasyonudur. Cordova çoklu platform destekli bir mobil uygulama geliştirme kütüphanesidir (Wargo, 2015). Bu sayede web uygulama geliştiricilerinin ek bir efor sarf etmeden bir mobil uygulama geliştiricisinden daha hızlı, işlevsel ve görsel mobil uygulamalar geliştirmesi mümkün olabilmektedir. Bu nedenlerden ötürü çalışma kapsamında geliştirilen tüm kullanıcı uygulamaları MeteorJS kullanılarak web

uygulaması olarak geliştirilmiş, Android/IOS mobil uygulama kodları MeteorJS'nin Cordova eklentisi kullanılarak elde edilmiştir.



Şekil 3.3. MeteorJS'de veri akışı (https://joshowens.dev/what-is-meteor-js (2014a)'dan değiştirilerek alınmıştır.)

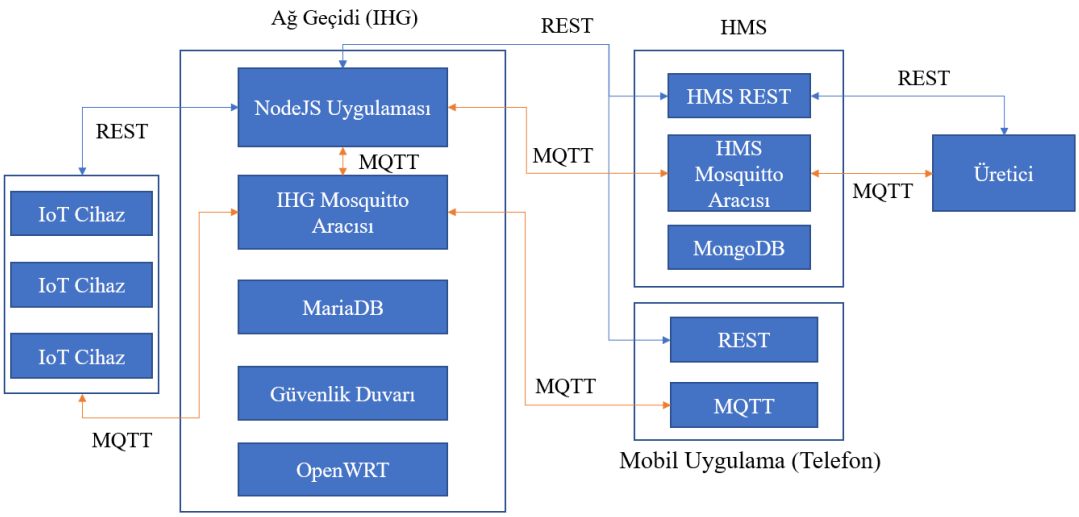
3.2. Yöntem

Bu kısımda mimari tasarım, SSLS tasarımı ve kullanımı, Çoklu SSID tekniği, güvenlik duvarları vasıtasıyla IoT cihazların internet kaynaklı saldırılara karşı korunması, şifreli MQTT haberleşmesi ve iletişim kontrolü kapsamında uygulanan çözümler ve cihaz ekleme ve güncelleme süreçleri açıklanmaktadır.

3.2.1. Mimari Tasarım

Mimari tasarımı Şekil 3.4'de gösterilen sistem, bulut sunucu uygulaması (HMS), ağ geçidi (IHG), son kullanıcı mobil uygulaması ve IoT bileşenlerinden meydana gelmektedir. IoT cihazlar MQTT ve REST bağlantı yeteneklerine sahip olan cihazlardır.

REST özellikle cihazların sisteme tanıtılması aşamasında kullanılmaktadır. Sonrasında cihazlar diğer bileşenler ile MQTT protokolü ile iletişim kurmaktadır. IHG'de bulunan NodeJS tabanlı Kontrol uygulaması IoT cihazlarının sisteme eklenmesi sırasında REST arayüzü ile hizmet verirken, IoT cihazlarının tüm MQTT bazlı trafiğini denetlemek/zenginleştirmek/anonimleştirmek için MQTT istemci arayüzünü kullanmaktadır. IHG aynı zamanda ağ bazında IoT cihazları ile evdeki diğer cihazların (telefon, bilgisayar vb.) soyutlanmasını sağlamaktadır. Telefon uygulaması IoT cihazlarına benzer şekilde IHG kontrol uygulaması aracılığı ile sisteme kendisini tanıtırken sonrasında IoT cihazları ile olan iletişimini MQTT protokolü ile gerçekleştirmektedir. HMS ise IoT cihaz ve telefon uygulamaları için gerekli kimlik doğrulama operasyonlarının yanı sıra IoT cihazlarını tanıtan meta verilerin saklanması ve sunulması operasyonlarından sorumludur. Son olarak üretici firmalar ürettikleri cihazlara ait meta verileri HMS'ye REST arayüzü ile ekler ve sonrasında cihazlarından yetkilendirilmiş olan verileri MQTT protokolü ile alabilir ya da cihazlara veri/komut sağlayabilir.



Şekil 3.4. Ağ geçidi tabanlı IoT mimari tasarım

3.2.2. Güvenli Hizmet Katmanı Tanımı

IoT'nin güvenlik sorunlarının tüm tarafların etkin katılımı ile çözülebileceği noktasından hareketle, üretici kaynaklı oluşturulacak SSLS'lerin değerlendirilmesi düşünülmüştür. SSLS'de meta veriler (cihazın üretici, ürün tipi, model bilgileri, SSLS tarihi), yayınla/abone ol (ing. publish/subscribe) konu listesi (ing. topic), mesaj şablonları ve mesaj içeriğindeki alanlar ile ilgili kısıtları içeren tanımlamalar ve son olarak kurallar yer almaktadır. SSLS'ler, IHG uygulaması tarafından yeni cihaz ekleme sürecinde yayınla/abone ol yetkilerinin oluşturulması, mesaj içeriklerinin kontrolü, bu ikisinde meydana gelebilecek anomalilerin tespiti ve IHG sistemine yeni bir cihaz eklenirken cihaz ile ilgili bilgi ve kullanıcı onayı gerektiren konuların ne olduğunun tespit edilmesinde kullanılmaktadır. Bu kapsamda oluşturulan örnek televizyon SSLS'i Şekil 3.5'te görülmektedir. Örnekte üretici firma, ürünün televizyon olduğu, ürün modeli, üretici tarafından MQTT ile iletişimde kullanması istenilen yayınla/abone ol başlığı isimleri ve mesaj içeriğinde bulunması gereken parametreler, bunların tipleri ve alabilecekleri değer aralıkları ve anonimleştirmeye yönelik mesaj içerikleri hakkında bilgi verilmektedir. SSLS'ler JSON (JavaScript Nesne Formatı-JavaScript Object Notation) formatında hazırlanmış olup HMS'de NoSQL bir veritabanı olan MongoDB üzerinde saklanmaktadır.

Yeni bir cihazın IHG'ye eklenmesi gerektiğinde cihaz ağa bağlanmakta ve eklenme isteğinde bulunmaktadır. Sonrasında ağ geçidinde cihazla ilgili SSLS, HMS'den talep edilmektedir. Elde edilen SSLS ile son kullanıcıya meta bilgileri ve onay gerektiren konular mobil uygulama arayüzü aracılığı ile sunulmaktadır. Kullanıcının cihazı ve yetkilerini onaylamasının ardından IHG'de bulunan Mosquitto aracısında (https://mosquitto.org/, 2019b) konu ve yetki kapsamındaki tanımlamalar gerçekleştirilir. Buna göre IHG kontrolcü uygulaması, IoT cihazların gönderdiği istek ve durum bilgilerini içeren tüm mesajları, üretici tarafından önceden tanımlanmış SSLS'te belirtilen mesaj içeriği kısıtları ile karşılaştırmaktadır. Mesaj içerisindeki alan sayısı, alan adları ve bu alanların her birinde gönderilebilecek veri tipleri birebir eşleme yöntemi ile kontrol edilmektedir. Örnek SSLS'teki durum mesajları, "status" başlığı ile haberleşmektedir ve bu başlıktan gelen mesajlarda "channel", "volume", "brightness" ve "ai_upscale" adlı 4 özelliğinin olduğu, bu alanların ilk üçünde tam sayı değer beklendiği, son değerin ise doğru/yanlış değeri olması gerektiği, tamsayı değer beklenen

alanların ayrıca 0 ile 100 arasında değer içermesi gerektiği SSLS'de tanımlı "Modelcheck" kuralı ile gerçekleştirilmektedir. Ayrıca, eğer anonimleştirme uygulanacak ise "brightness" ve "volume" alanları hariç diğer alanların filtrelenmesi ve HMS ile paylaşılacak olan mesaj içeriğinde yer almayacak şekilde gelen mesajın değiştirilmesi sağlanmaktadır. Örnek SSLS'e göre "software_update_request" başlığı ile haberleşme sağlanan güncelleme isteklerinde ise, mesaj içeriğinin model ismi ve versiyon numarasından oluşması gerektiği belirtilmekte, eksik ya da fazla alan içeren mesajlar bu sayede saptanarak anomali bildirimi yapılabilmektedir.

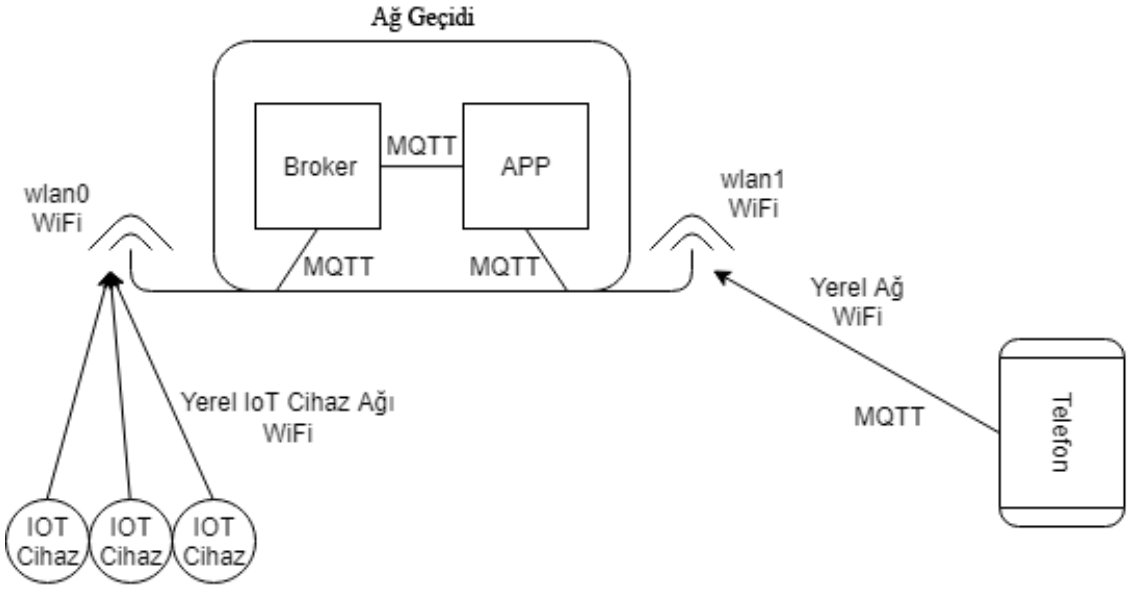
```
{
  "_id": "Dxd974wmGayHrGWPn",
  "manufacturer": "samsung",
  "model": "q60r",
  "type": "tv",
  "ssls_date": "10.09.2015",
  "publishTopics": ["update", "software_update"],
  "subscribeTopics": ["status", "software_update_request"],
  "properties": [
    { "type": "boolean", "name": "working_status","value": false },
    { "type": "number", "name": "channel","value": 0 },
    { "type": "number", "name": "volume", "value": 0 },
    { "type": "number", "name": "brightness", "value": 0 },
    { "type": "boolean", "name": "ai_upscale", "value": true }
  ],
  "rules":{
    "modelCheck":{
      "parameters": ["channel", "brightness", "volume"],
      "body": "channel<100 && channel>0 && brightness<100 && brightness>0 && volume<100 &&
        volume>0) ? return true; : return false;"
    },
    "anonymity":{
      "parameters": ["channel", "brightness", "volume"],
      "body": "return ({success: true, 'payload': {brightness, volume}})"
    }
  },
  "message_format": {
    "status": [
      "open", "close", "channel", "volume", "brightness", "ai_upscale", "version_update"
    ],
    "update": [
      "working_status", "channel", "volume", "brightness", "ai_upscale" ],
    "software_update_request": [
      "modelname", "version"
    ],
    "software_update": [
      "software_version", "software_address", "software_date"
    ]
  }
}
```

Şekil 3.5. Samsung q60r akıllı tv için örnek SSLS

Üreticinin SSLS oluşturması için dinamik form yapısı kullanılan MeteorJS tabanlı bir Web uygulaması geliştirilmiştir. Uygulama, üreticinin belirttiği tüm bilgileri JSON tabanlı bir nesneye dönüştürmekte ve bulut sistemindeki MongoDB sunucusuna kaydedilmek üzere iletmektedir.

3.2.3. Çoklu SSID

IoT cihazlarda veri aktarımı çoğunlukla kablosuz haberleşme kullanılarak yapılmaktadır. Bu nedenle, ikinci bölümde detaylı incelemesi verilen çeşitli saldırılara maruz kalabilmektedirler. IoT cihazlarının güvenlik ya da servis kalitesi gereksinimleri nedeni diğer ağlardaki cihazlardan soyutlanmaları iki veya daha fazla yerel ağ oluşturulması ile sağlanabilmektedir. Bu nedenle çalışma kapsamında Şekil 3.6'da görüldüğü üzere iki SSID'li (Service Set Identifier) bir ağ yapısı kurulmuştur. RPi'nin mevcut kablosuz ağ adaptörü birden fazla SSID kurulumuna imkân vermediği için çoklu SSID desteği bulunan Buffalo AirStation N150 model USB kablosuz yönlendiricinin RPi ile birlikte kullanımı değerlendirilmiştir.



Şekil 3.6. OpenWRT çoklu-ssid konfigürasyonu

Elde edilen iki SSID'li yapıda, IoT cihazları (IP kamera, termostat, klima, vb.) wlan0 (ihg2-iot) ağında haberleşirken, kullanıcı cihazları (telefon, bilgisayar, vb.) wlan1 (ihg2-home) ağında haberleşmektedirler. Her iki ağ da birer kablosuz yerel ağ (WLAN) olarak hizmet vermektedir. Kullanıcı ve sensör cihazlar arasındaki bu soyutlama sayesinde kullanıcı cihazları ve IoT cihazlar arasında yerel ağda doğrudan iletişim kurma imkânı ortadan kalkmaktadır. Böylece herhangi bir şekilde ele geçirilen bir kullanıcı cihazı (ev bilgisayarı, vb.) üzerinden IoT cihazlarının; ele geçirilen bir IoT cihaz üzerinden ise kullanıcı cihazlarının tehdit altında kalması önlenmektedir. Ayrıca wlan0 yerel ağının, sonraki bölümde ifade edildiği üzere, güvenlik duvarının etkin kullanımı aracılığı ile internet çıkışı engellenmekte ve IoT cihazların doğrudan internet kaynaklı saldırılara karşı korunması sağlanmaktadır. IoT ortamlarda bütünlük ve erişilebilirliğe yönelik en büyük tehditlerden olan DDoS saldırılarının etkinliği bu yaklaşımın kullanıldığı ağlarda azalacaktır. Çünkü ortamdaki cihazların dış ağa çıkış izinleri olmadığından yapılacak saldırılarda zombi cihaz olarak kullanılamayacak ve yine doğrudan bu cihazlara yönelik dış ağ kaynaklı bir saldırıda IHG tarafından engellenmiş olacaktır.

3.2.4. İletişim Kısıtlaması

Güvenlik duvarları belirli bir ortamın korunması amacı ile gelen ve giden tüm trafiği gözlemleyen ve kural tabanlı iş akışı ile ilgili paket trafiğinin engelleme ya da izin verme kararını alan yazılımlar olup ağ güvenliğinde en önemli noktalarından birini oluşturmaktadırlar. Scarfone ve Hoffman'a (2009) göre tüm güvenlik duvarları 3 temel özelliği içermelidir. Buna göre dış ağdan gelen ve iç ağdan giden tüm trafiğin gözlemlenmesi, sadece yetkili bağlantılara izin vermesi, yetkisiz veya zararlı olduğunu belirlediği bağlantıları engellemesi ve doğrudan saldırıya açık herhangi bir zafiyet barındırmaması gerekmektedir. Modern güvenlik duvarları, temel filtre mekanizması olarak kural tabanlı paket filtreleme modeli kullanmakta ve öncelikle OSI katmanlarından "network" ve "transport" katmanlarının korunmasında görev almaktadırlar. Çalışmada OpenWRT işletim sistemi ile birlikte gelen Linux IPTables güvenlik duvarı uygulaması kullanılmıştır. Linux IPTables paket filtreleme tabanlı bir güvenlik duvarıdır (Purdy, 2004). Paket filtreleme tabanlı güvenlik duvarları sıralı bir kural seti barındıran yönergeler zincirinden oluşmaktadır. Kurallar filtreleme alanları ve

bir aksiyondan oluşmaktadır. Filtreleme alanları paketin ilgili kurala tabi olup olmadığını belirlemek amacı ile kullanılmakta ve çeşitli varyasyonları bulunmak ile birlikte genellikle protokol, kaynak IP adresi, kaynak Port numarası, hedef IP adresi, hedef Port numarasından ve güvenlik duvarı alan bilgilerinden oluşmaktadır. Aksiyon ise ilgili kural ile eşleşen paketler için uygulanacak eylemi belirtmektedir. Linux IPTables güvenlik duvarlarında bu eylemler ACCEPT, DROP, REJECT aksiyonlarını oluşmaktadır. Gelen ve giden tüm paketler, sırayla kural setindeki tüm kuralları dolaşmakta ve filtreleme alanları eşleşen ilk kuralın aksiyonunu almaktadırlar.

IoT cihazlarının tabi olduğu güvenlik duvarı kuralları Şekil 3.7'de görülmektedir. Buna göre ağa bağlanacak cihazların REST uygulamasına istekte bulunabilmesi için "iot-rest" kuralı tanımlanmış, diğer tüm trafik güvenlik duvarında DROP işlemine tabi tutulmaktadır.

Traffic Rules

Name	Match	Action	Enable
Allow-DHCP-Renew	IPv4-UDP From *any host* in *wan* To *any router IP* at port *68* on *this device*	*Accept input*	☑
Allow-Ping	IPv4-ICMP with type *echo-request* From *any host* in *any zone* To *any router IP* on *this device*	*Accept input*	☑
Allow-IGMP	IPv4-IGMP From *any host* in *wan* To *any router IP* on *this device*	*Accept input*	☑
Allow-DHCPv6	IPv6-UDP From IP *fc00::/6* in *wan* To IP *fc00::/6* at port *546* on *this device*	*Accept input*	☑
Allow-MLD	IPv6-ICMP with types *130/0, 131/0, 132/0, 143/0* From IP *fe80::/10* in *wan* To *any router IP* on *this device*	*Accept input*	☑
Allow-ICMPv6-Input	IPv6-ICMP with types *echo-request, echo-reply, destination-unreachable, packet-too-big, time-exceeded, bad-header, unknown-header-type, router-solicitation, neighbour-solicitation, router-advertisement, neighbour-advertisement* From *any host* in *wan* To *any router IP* on *this device*	*Accept input* and limit to *1000* pkts. per *second*	☑
Allow-ICMPv6-Forward	IPv6-ICMP with types *echo-request, echo-reply, destination-unreachable, packet-too-big, time-exceeded, bad-header, unknown-header-type* From *any host* in *wan* To *any host* in *any zone*	*Accept forward* and limit to *1000* pkts. per *second*	☑
Allow-IPSec-ESP	IPv4 and IPv6-IPSEC-ESP From *any host* in *wan* To *any host* in *lan*	*Accept forward*	☑
Allow-ISAKMP	IPv4 and IPv6-UDP From *any host* in *wan* To *any host*, port *500* in *lan*	*Accept forward*	☑
Allow-iot dns	IPv4 and IPv6-TCP, UDP From *any host* in *iot* To *any router IP* at port *53* on *this device*	*Accept input*	☑
Allow-iot dhcp	IPv4 and IPv6-UDP From *any host* in *iot* To *any router IP* at ports *67-68* on *this device*	*Accept input*	☑
Allow-iot rest/mqtt	IPv4 and IPv6-TCP From *any host* in *iot* To *any router IP* at ports *4000-4001* on *this device*	*Accept input*	☑

Şekil 3.7. Güvenlik duvarı konfigürasyonu

Tez kapsamında geliştirilen güvenlik çözümlerinden bir diğeri tüm MQTT trafiğinin IHG üzerinden geçirilmesi ve mesajların üretici tarafından önceden belirlenmiş SSLS'lere uygunluğunun kural-tabanlı bir filtreleme sistemi ile denetlenmesidir. IoT cihazların IHG kontrolcü uygulamaya gönderdiği tüm istek ve durum bilgileri, üretici tarafından önceden tanımlanmış SSLS'te belirtilen mesaj içeriği kısıtları ile karşılaştırılmaktadır (Şekil 3.5). Sözde kodu Şekil 3.8'de verilmiş olan haberleşme kontrol algoritmasında, mesaj içeriğindeki alanların adları, tipleri ve toplam alan sayısının SSLS'te belirtilen formata uygunluğu kontrol edilmektedir. Sonrasında, bu alanlarda gönderilen verilerinin istenilen aralıklarda olup olmadıkları belirlenmektedir. Tüm kontrolleri geçen paketler meşru sayılmakta ve diğer taraflara iletilmektedir.

```
validated = true
for m in message
   count = 0
   for f in message_format
      if m.name == f.name
         count = 1
      end
   end
   if count != 1
      validated = false
   end
end
```

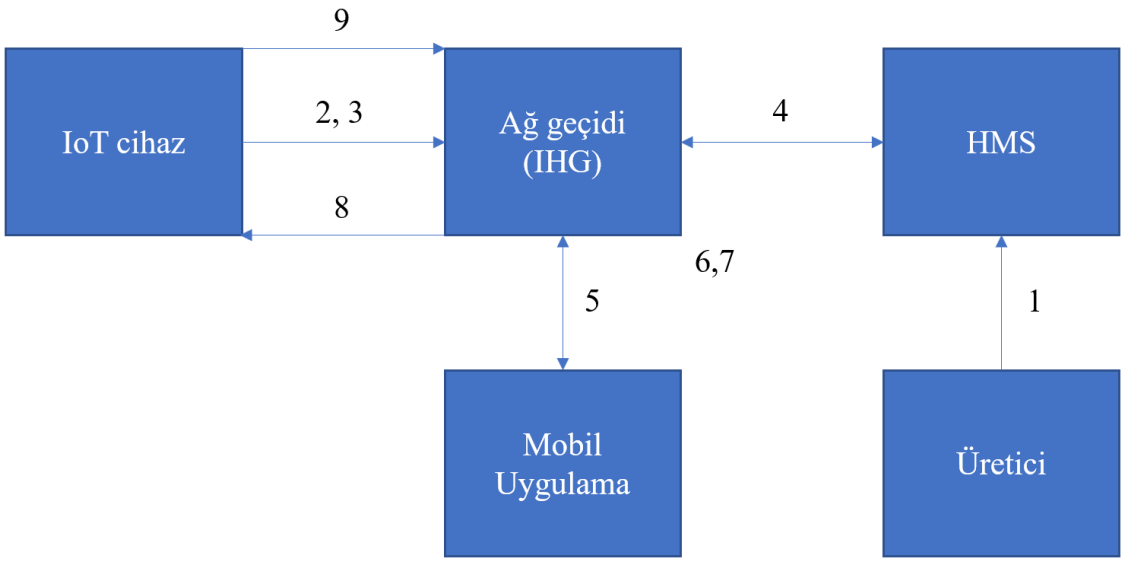
Şekil 3.8. Haberleşme kontrol algoritması

3.2.5. IoT Cihaz Ekleme

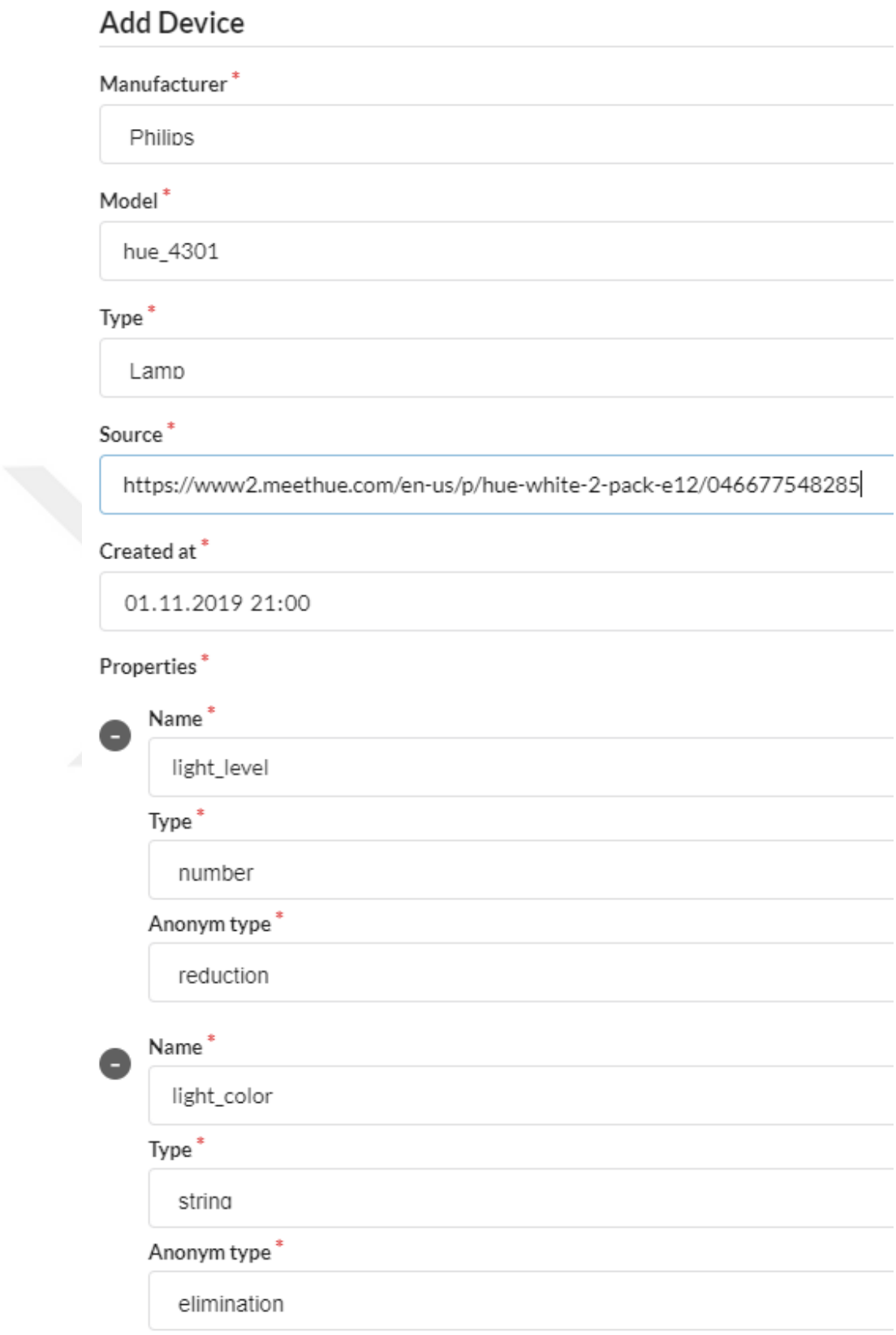
IoT'de en önemli problemlerden biri ağa yeni cihazların güvenli bir şekilde eklenmesidir. Zira zararlı düğümler olarak bilinen cihazların herhangi bir şekilde ağa sızması, ağdaki tüm cihazların ele geçirilebilmesine sebep olmaktadır. Bu nedenle çalışma kapsamında üretici ve son kullanıcının etkin bir şekilde katılım gösterdiği

güvenli bir cihaz ekleme senaryosunun kullanımı önerilmektedir. Senaryonun akış diyagramı Şekil 3.9'da sunulmuştur. Şekildeki numaralar listedeki aşamalara karşılık gelmektedir.

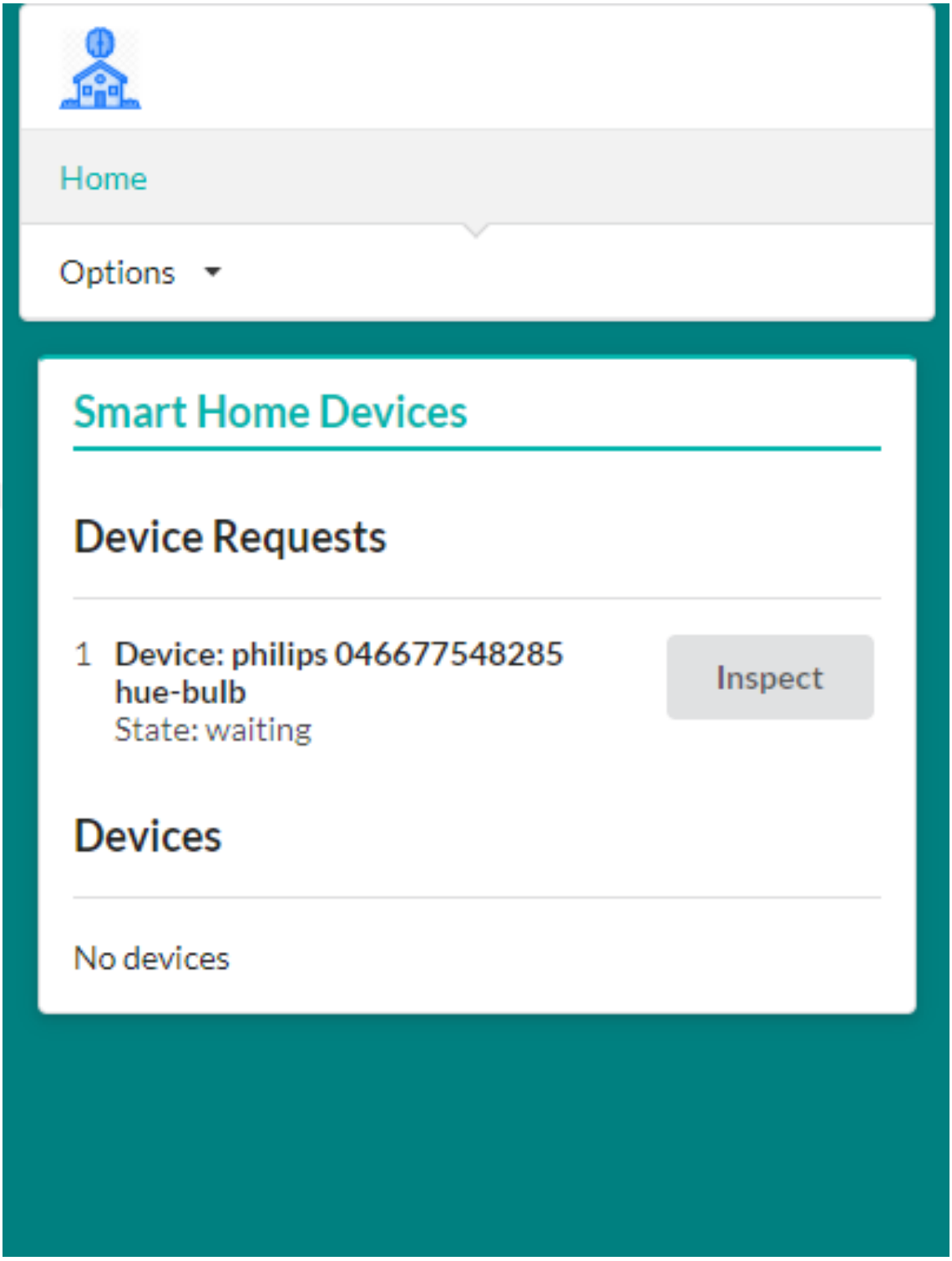
1. Üretici firma, geliştirmiş olduğu her cihaz için SSLS oluşturur. Tanımlanan SSLS'ler HMS_PANEL web uygulaması aracılığı ile HMS'deki veritabanına aktarılır (Şekil 3.10). Burada servis operatörü HMS'ler ayrıca gerektiğinde denetleyebilir.
2. Cihazın Wlan0'a bağlanması için gerekli olan SSID ve parola bilgileri cihaza kullanıcı tarafından eklenir. Cihazın Wlan0'ya bağlanması sağlanır.
3. Cihaz ağa katılma isteğini, ağ geçidindeki NodeJS uygulamasına iletir. Ağ geçidi, HMS'den cihaz ile ilgili SSLS'i talep eder.
4. Ağ geçidi, cihaza uygun SSLS'i edinir, kullanıcı mobil uygulamasına MQTT aracılığı ile cihaz onaylama bildirimini iletir (Şekil 3.11).
5. Kullanıcı, cihazın bilgilerini doğruladıktan sonra Şekil 3.12'de gösterildiği üzere cihazı onayladığını MQTT aracılığı ile ağ geçidi NodeJS uygulamasına iletir.
6. Ağ geçidi NodeJS uygulaması, MQTT üzerinden gelen onay mesajı üzerine IoT cihazı için çalışma zamanında oluşturulan kullanıcı adı ve şifre ile ACL (Erişim Kontrol Listesi, Access Control List) tanımlamaları MQTT aracısına iletilir.
7. HMS'den daha önce çekilmiş olan SSLS'in bir kopyası, ağ geçidindeki SQL tabanlı sqlite veritabanına kaydedilir.
8. Cihazın MQTT iletişiminde kullanacağı kullanıcı adı, parola, cihaza özel simetrik şifreleme anahtarı ve ağ geçidi MQTT aracısının IP adresi cihazın REST isteğine cevap olarak döndürülür.
9. Cihaz isteğine cevap aldıktan sonra elde ettiği erişim bilgilerini ve şifreleme anahtarını kullanarak şifreli trafik üretmeye başlayabilmektedir. Son kullanıcı cihazdan gelen verileri Mobil uygulama aracılığı ile inceleyebilmekte ve cihaza komut gönderimi yapabilmektedir (Şekil 3.13).



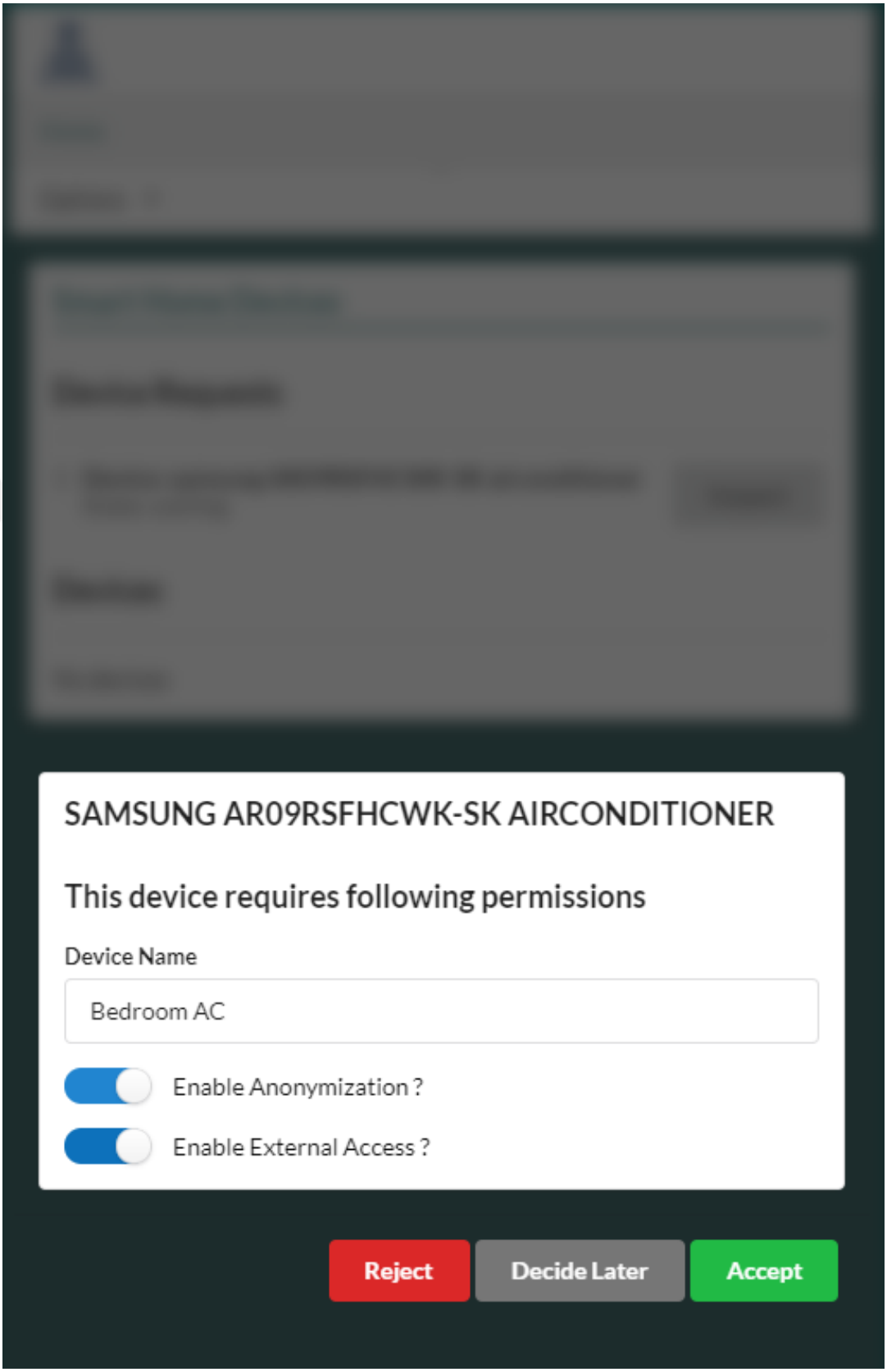
Şekil 3.9. IoT cihaz ekleme akış diyagramı



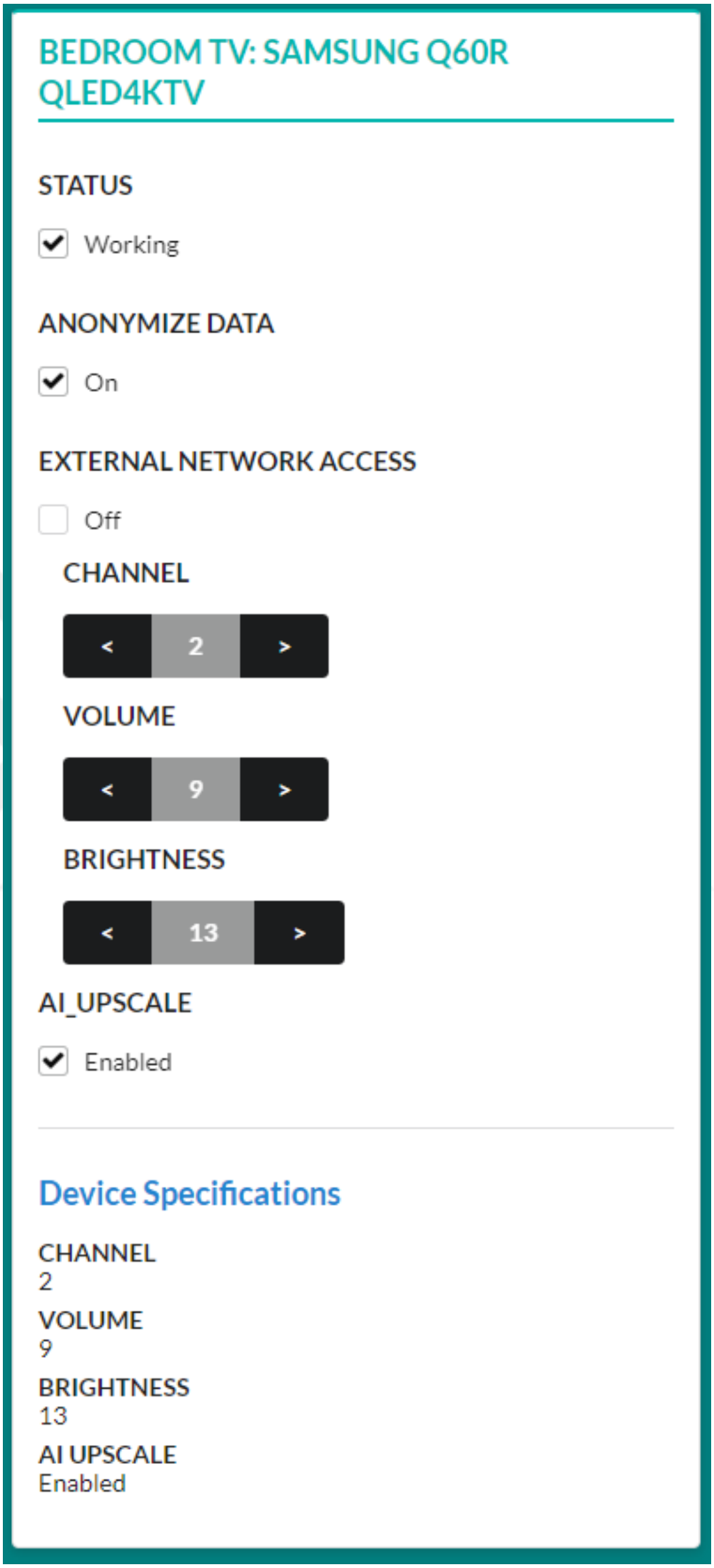
Şekil 3.10. SSLS giriş ekranı



Şekil 3.11. Mobil uygulamada yeni cihaz bildirimi



Şekil 3.12. Mobil uygulamada IoT cihaz onaylama ekranı



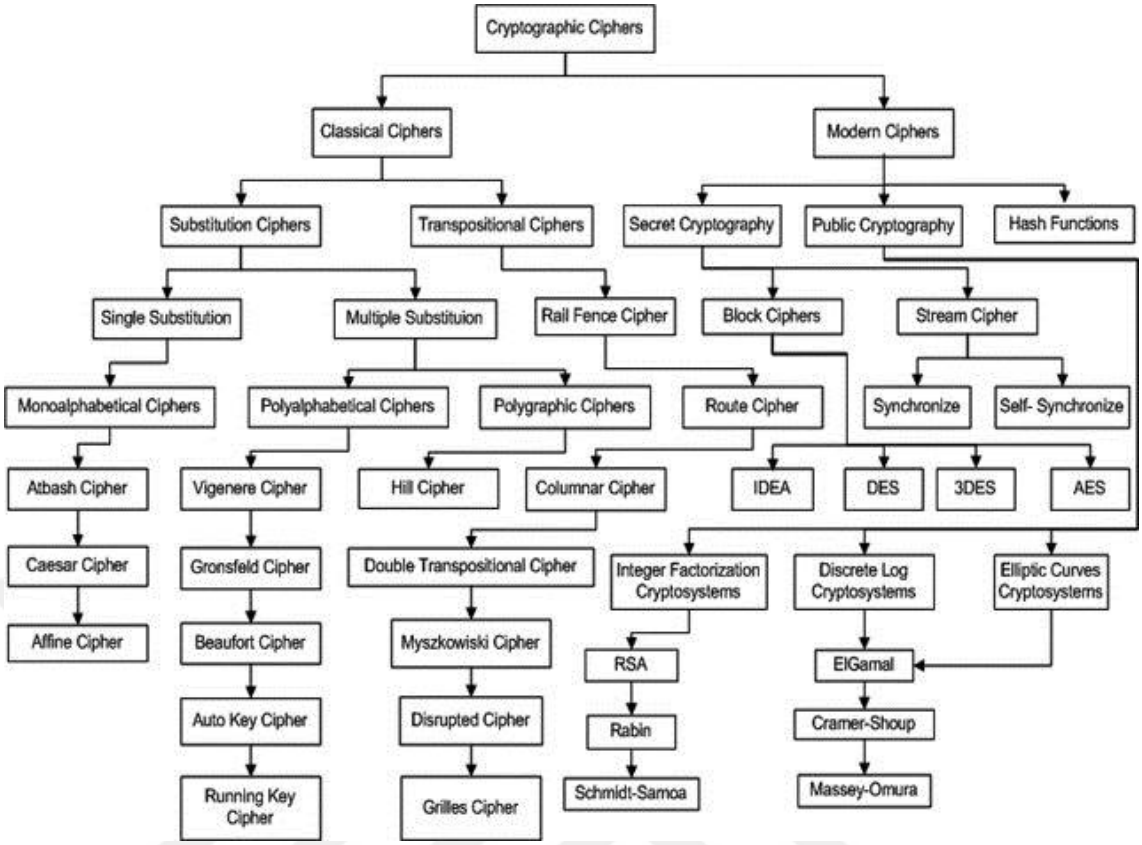
Şekil 3.13. Mobil uygulamada IoT cihaz yönetim ekranı

3.2.6. MQTT Protokolü ve Cihaz Haberleşmesi

Sunulan çalışmada taraflar arasındaki haberleşmede nesnelerin interneti çözümlerinde yoğun olarak kullanılmakta olan Telemetri Mesaj İletim Protokolü (MQTT) protokolü kullanılmıştır (http://mqtt.org/, 2019). MQTT, 1999 yılında IBM tarafından geliştirilmiş olup, yayınla/abone ol (publish/subscribe) mantığı ile çalışan ve düşük bant genişliği olan ortam ve cihazları hedefleyen hafif M2M (Makine-Makine) mesajlaşma protokollerinden birisidir. MQTT'de merkezi bir aracı (broker) ve istemciler bulunmaktadır. Bir istemci tarafından bir başlığa (topic) yayınlanmış olan mesaj, başlığa abone olan istemcilere aracı tarafından iletilir. Eclipse Mosquitto, HiveMQ (https://www.hivemq.com/docs/4.2/hivemq/introduction.html, 2019c) ve ActiveMQ (https://activemq.apache.org/getting-started, 2019d) bilinen aracı sunucu yazılımları, gerektiğinde farklı protokoller (MQTT, AMQP (Vinoski 2006), OpenWire, TCP, vb.) arasında protokol değişimleri yaparak mesajı istemcilere iletirler. Çalışma kapsamında, açık kaynak kodlu ve geniş dokümantasyon desteği bulunan Eclipse Mosquitto aracı yazılımının kullanımına karar verilmiştir. IoT cihazlar MQTT aracısı ile durum bilgisi iletmek, son kullanıcı ve üretici kaynaklı komutlar dinlemek, yazılım güncellemelerine erişmek ve yazılım güncellemesi talebinde bulunmak amacı ile Şekil 3.5'de ifade edildiği üzere SSLS'te belirtilen 4 başlık üzerinden trafik oluşturmaktadırlar. Bu başlıkların isimleri üretici tarafından sağlanmak ile birlikte kullanım amaçları belirtilen amaçlarla sabittir.

IoT cihazları ve IHG kontrolcü uygulama arasındaki trafiğin şifreli hale getirilmesi IoT güvenlik ölçütlerinden gizlilik ve mahremiyet açısından önem taşımakta ve yerel ağda pasif dinleme yapan veya ağa sızma girişiminde bulunan saldırganlara karşı güvenlik sağlamaktadır. Modern şifreleme teknikleri simetrik ve asimetrik şifreleme olmak üzere iki başlıkta incelenmektedir (Katz ve Lindell, 2014). Simetrik şifrelemede hem şifreleme hem de çözmede tek bir şifreleme anahtarı kullanılmaktadır. Bu durum asimetrik şifrelemeye göre çok daha hızlı olmasını sağlamaktadır. Buna karşılık en önemli dezavantaj ise anahtar dağıtımı ve anahtar yönetimi problemleridir. Asimetrik şifreleme ise açık ve özel anahtar olmak üzere iki şifreleme anahtarı kullanılmaktadır. Özel anahtar sadece şifreli metni çözecek tarafta bulunmakta, açık anahtar ise çözücü ile

iletişim kuracak tüm taraflara dağıtılmaktadır. Asimetrik şifreleme güvenlik açısından çok daha etkili olmak ile birlikte simetrik şifrelemeye göre daha fazla güç ve yer gerektirmekte ve IoT ortamlarında sık kullanılmamaktadır. Günümüzde her iki teknikte kullanılan birçok algoritma bulunmaktadır (Şekil 3.14). Fakat kaynak kısıtlı cihazlarda oluşturacağı yük dikkate alındığında simetrik şifreleme tabanlı yöntemler öne çıkmaktadır (Katagi ve Moriai 2008, Toğay ve ark. 2019a). Simetrik şifreleme için iletişim kuracak her iki taraf arasında ortak bir anahtarın şifreleme ve çözme için belirlenmesi gerekmektedir. Asimetrik şifreleme protokolleri tabanlı anahtar takası oldukça yaygındır. Asimetrik şifreleme protokolü ile anahtar takası sonrasında simetrik şifreleme protokolleri ile şifreli veri transferi yöntemlerinden olan Transmission Control Protocol (TLS), aracı ve istemciler arasındaki iletişimde kullanılmak üzere Mosquitto ve uyumlu istemci kütüphaneleri tarafından desteklenmektedir. Ancak, 1) cihazların kısıtlı cihazlar olması ve asimetrik şifreleme için gerekli açık anahtarları saklama gereksinimi, 2) sertifikaların gerektiğinde geri çekilmesindeki zorluklar, 3) her IHG içerisinde ortak bir sertifika ve özel anahtar barındırmaya bağlı güvenlik riskleri nedeni ile TLS tabanlı yaklaşımın uygulanması pratikte uygun değildir. Simetrik şifreleme ile iletişim için anahtarlar taraflarca daha önceden ilgili taraflarca paylaşılması durumunda doğrudan oturum bazlı üretilmiş bir anahtar ile şifreli iletişim kurulabilir. Bu yöntemde anahtarın cihazdan elde edilmesi riski bulunmaktadır. Bu kapsamda güvenli ek donanımlarda anahtarlar güvenle saklanabilir ve ilgili anahtar gerek yetkilendirmede gerek oturum anahtarının belirlenmesinde kullanılabilir (Toğay ve ark. 2019). Sunulan tezde her cihaz için ayrı bir anahtar cihaz ekleme sürecinde belirlenmektedir. Dolayısı ile herhangi cihaz diğer cihazların trafiğini görememekte ya da değiştirememektedir.

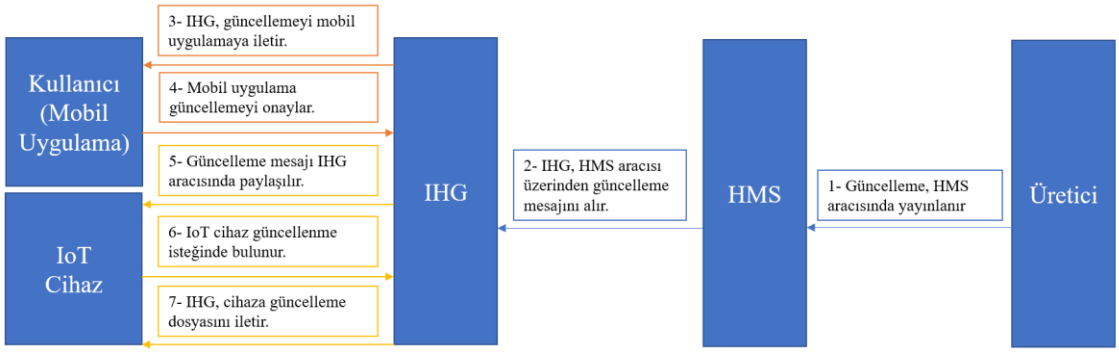


Şekil 3.14. Modern kriptografik algoritmalar (Ramakrishnan (2018)'den uyarlanarak alınmıştır)

Çalışma kapsamında IoT cihazların durum bilgilerinin çeşitli taraflara (son kullanıcı, üretici firma, çeşitli bulut servisleri vb.) gönderilmesi, son kullanıcının mobil uygulama aracılığı ile oluşturduğu komutların IoT cihazlara iletilmesi ve üretici kaynaklı komutların IoT cihazlara iletilmesinde IHG kontrolcü uygulaması aracı vazife görmektedir. MQTT protokolü üzerinden oluşturulan tüm trafik, bir simetrik şifreleme algoritması (AES-128) kullanılarak şifrelenmektedir. Bu sayede veri gizliliği ve mahremiyet seviyesinde iyileştirme sağlanabilmektedir.

3.2.7. Cihaz Güncellemeleri

IoT'de cihaz güncellemelerinin güvenli bir şekilde üreticiden IoT cihaza aktarılması en önemli problemlerden biri olarak görülmektedir. Günümüzde zararlı yazılım içeren güncellemeler vasıtası ile birçok cihaz kolaylıkla ele geçirilmekte ve çeşitli amaçlar ile kullanılmaktadır. Güncellemelerin güvenli bir şekilde yapılabilmesi için tüm tarafların ve haberleşme kanallarının güvenilir olması şarttır. IoT cihazlar ile üretici firma arasında ağ geçidi cihazın kullanımı ile bu güven sağlanmaya çalışılmıştır. Şekil 3.15'de görüldüğü üzere üretici tarafından MQTT aracılığı ile güncelleme bilgisi cihaza özel güncelleme başlığında HMS sunucusunda paylaşılmaktadır. Ağ geçidi, ağda kayıtlı bulunan tüm IoT cihazlar için güncelleme başlık bilgisine SSLS'ler aracılığı ile sahip olduğundan sadece kendi cihazlarına yönelik güncelleme mesajlarını alabilmektedir. Bir güncelleme mesajı ulaştığında üreticinin mesajda belirttiği indirme adresinden güncelleme dosyası indirilmekte ve istendiğinde cihaza iletilmek üzere yerel dosya sisteminde saklanmaktadır. Sonrasında cihaza güncelleme bulunduğuna dair ağ geçidi MQTT aracısı üzerinden güncelleme mesajı oluşturulmakta ve cihaz SSLS'inde belirtilen başlık adına göre cihaza/cihazlara iletilmektedir. Güncellemeyi edinmek isteyen cihazlar, ağ geçidine güncelleme talebi için SSLS'lerinde belirtilen başlıkta güncelleme isteğinde bulunmakta ve ağ geçidi dosyayı istekte bulunan cihazın dosya sistemine aktarabilmektedir.



Şekil 3.15. Cihaz güncellemeleri akış diyagramı

4. BULGULAR

4.1. Güvenlik ve Gizlilik

Bu kısımda çalışma kapsamında önerilen güvenlik çözümlerinin IoT güvenlik gereksinimlerindeki karşılıkları sunulmaktadır. IoT cihazlarının, internet erişimine sahip olması ve diğer IoT ve kullanıcı cihazları ile doğrudan haberleşmesinin oluşturduğu güvenlik riskleri sebebiyle çoklu SSID tekniği ve güvenlik duvarları kullanım yaklaşımı uygulanmıştır. Böylece cihazların, ağ geçidindeki REST uygulaması ve MQTT aracı uygulaması ile olan trafiğin dışındaki tüm iletişim engellenmiştir. Bu kapsamda elde edilen kazanımlar Çizelge 4.1'de listelenmiştir.

Cihazlar doğrudan internet kaynaklı erişime kapatıldığından internet kaynaklı saldırılar etkisiz hale getirilmiştir. Ayrıca, fiziksel olarak ağdaki bir cihazın ele geçirilmesi ya da WIFI ağ şifresinin ele geçirilmesi durumunda cihazların bulunduğu ağa erişim sağlanabilir. Ancak, cihazlar doğrudan sadece aracı uygulama ve cihaz tarafından bilinen bir şifre ile iletişim kurması nedeni ile araya girme saldırısı gerçekleştirilemez. Bunun dışında ağ geçidi üzerinden cihazlar arasındaki doğrudan iletişim güvenlik duvarı aracılığı ile engellendiğinden cihazların birbirlerine doğrudan bir saldırısı mümkün değildir.

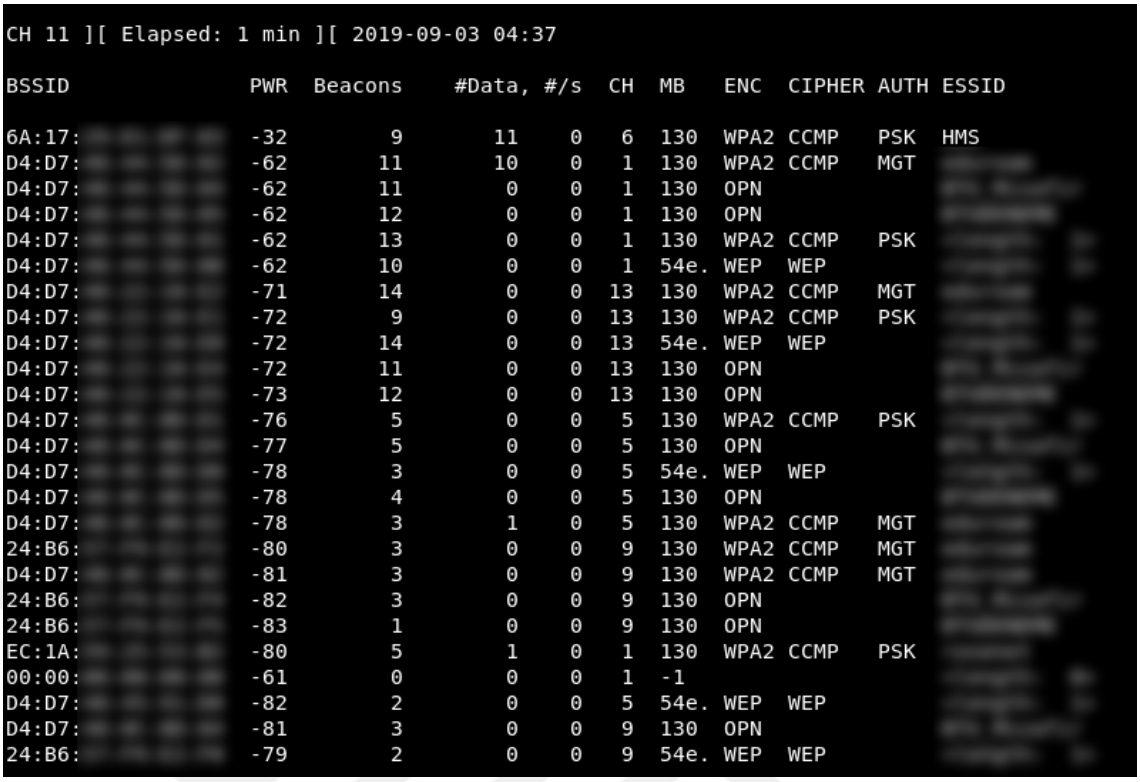
Çizelge 4.1. IoT cihazlar için çoklu-ssid ve erişim kısıtlamasının katkıları

#	IoT saldırı yüzeyleri	Önerilen Yöntem	Eski Yöntem
1	Evdeki tüm cihazlara ait konum tespiti	Evet	Evet
2	Evdeki tüm cihazların marka/model tespiti	Evet	Evet
3	Evdeki tüm cihazlara ait içeriğin dinlenmesinin engellenmesi	**Evet**	Hayır
4	Evdeki kullanıcı cihazlarına (bilgisayar, telefon vb.) ilişkin şifresiz paketlerin (UDP/TCP) dinlenememesi	**Evet**	Hayır
5	IoT cihazlarına internet kaynaklı saldırıların engellenmesi	**Evet**	Hayır

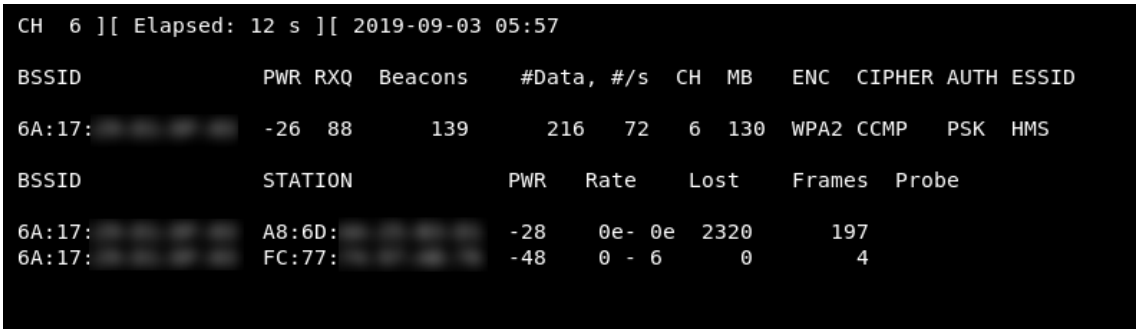
Çizelge 4.1. IoT cihazlar için çoklu-ssid ve erişim kısıtlamasının katkıları (Devamı)

6	IoT cihazlarının doğrudan internet erişiminin ve saldırılarda kullanılmalarının engellenmesi	**Evet**	Hayır
7	IoT cihazlarının birbirleri ile doğrudan haberleşmesi yoluyla saldırı gerçekleştirilmesinin engellenmesi	**Evet**	Hayır
8	Kullanıcı cihazları ve IoT cihazlarının doğrudan haberleşmesinin engellenmesi	**Evet**	Hayır

Aynı ağda olma koşulu ile ağ trafiğindeki şifreli ve şifresiz tüm trafik elde edilebilmektedir. Farklı SSID'li ağların anahtarı farklı olacağı için farklı ağlara ait şifresiz veriler diğer ağdaki istemciler tarafından ağ şifresi ile şifreli olacağı için erişilemez. Ancak, paketlerin başlıklarında cihazlar hakkında içeriği bilinmese dahi bazı temel bilgiler pasif dinleyiciler için dahi erişilebilir durumdadır. Örnek olarak Çizelge 4.1'de ifade edildiği üzere MAC adresi hangi üreticiye ait bir cihaz olduğu ya da cihazın dinleyici cihaza göre sinyal gücü ile göreceli konumu tahmin edilebilir. Pasif dinleme yazılımlarından olan Kismet (Haines ve Thornton, 2008) kablosuz ağ trafiği izleme aracı ile yapılan denemeler sonucunda çevredeki kablosuz ağların gizli ya da açık SSID fark etmeksizin ağ geçidi BSSID (temel servis seti tanımlayıcısı) değerlerinin bulunabildiği görülmüştür (Şekil 4.1). BSSID değeri, erişim noktasının kullandığı ağ adaptörünün MAC adresini ifade etmektedir. BSSID ve kanal değerleri kullanılarak ilgili ağ geçidi ile iletişim kuran cihazların konum tespiti, MAC adresleri ele geçirilebilmektedir (Şekil 4.2). MAC adresleri kullanılarak cihazların kullandığı ağ adaptörlerinin ve bu sayede cihazların kendisinin marka/model tespiti yapılabilmektedir.



Şekil 4.1. Kismet ile çevredeki ssid yayınlarının tespiti



Şekil 4.2. Ağ geçidi ile iletişim kuran cihazların tespiti

Linux IPTables güvenlik duvarının etkin kullanımı ile IoT cihazların tamamının sadece REST sunucusu ve MQTT aracısına erişim izni tanımlanabilmektedir. Cihazlar doğrudan internet erişimine sahip olmadıkları için, internet kaynaklı saldırılardan etkilenmemektedirler. Bu sayede IoT güvenlik gereksinimlerinden gizlilik, kullanılabilirlik ve güven ölçütlerinde geliştirme yapılmış olmaktadır. Elde edilen

güvenlik duvarı konfigürasyonunda REST sunucusuna erişim izni veren ve diğer tüm trafiği engelleyen 2 kural bulunmaktadır. Eklenen her cihaz için MQTT aracısına erişebilmesi için güvenlik duvarına bir kural oluşturulmaktadır.

IoT güvenlik gereksinimlerine dair önerilen çözümler Çizelge 4.2'de sunulmaktadır. Gizlilik gereksinimine çoklu SSID tekniği ve kural tabanlı mesaj kontrol algoritması vasıtasıyla sunulan çözüm sayesinde IoT cihazların belirli bir kanalda belirli mesaj formatı haricinde mesajlaşma imkanını ortadan kaldırarak ağ içerisinde yetkisiz işlem imkânı bırakmamaktadır. Bütünlüğe yönelik olarak MQTT mesajların AES-128 ile şifrelenmesi ve SHA-256 ile özet değer oluşturulup mesaj içeriğiyle birlikte gönderilmesi önerilmiştir. Cihazların çeşitli saldırılar ile pil ömrü, bant genişliği, işlem gücü gibi özelliklerinin tüketilmesini engellemek ve cihazın işlevsel kalmasını sağlayarak kullanılabilirliği sağlamak amacıyla yine Çoklu SSID tekniği kullanılmıştır. Bu teknik cihazlara yakın mesafeden fiziksel katman üzerinden yapılan saldırılara karşı bir koruma sağlamasa da internet kaynaklı doğrudan saldırıların engellenmesi sağlanabilmektedir.

Üretici firma tarafından oluşturulup HMS'ye yüklenen SSLS'ler IHG'de taraflar arasındaki mesajların belirli kurallara göre aktarılmasında etkin bir şekilde kullanılmaktadır. SSLS'ler ile tarafların hangi mesajları hangi başlığa yayıncı ya da abone olacağı, içeriğinin ne olacağı ile ilgili tüm detay bilgileri içermektedir.

Çizelge 4.2. IoT güvenlik ölçütlerine sunulan çözümler

Güvenlik Ölçütü	**Problem**	**Önerilen Yöntem**
Gizlilik	IoT cihazlarının ürettiği ya da alacağı verilerin içeriğinin ele geçirilmesi	Çoklu SSID tekniği, cihaz bazında şifreli iletişim, IHG tarafından mesajların iletileceği tarafların sınırlandırılması
Bütünlük	MITM vb. saldırılar ile mesaj içeriğinin bozulması	Haberleşmede simetrik şifreleme ve özet (hash) değerlerinin kullanımı

Çizelge 4.2. IoT güvenlik ölçütlerine sunulan çözümler (Devamı)

Kullanılabilirlik	Cihazların saldırı (DDoS) altındayken servis sunamaması	Çoklu SSID tekniği ve IHG tarafından güvenlik duvarı ile doğrudan trafiklerin engellenmesi
Kimliklendirme ve Yetkilendirme	Taraflar bir mesaj geldiğinde mesajın kim tarafından gönderildiğini doğrulama şansı bulunmamaktadır. Dolayısı ile taraflar aracı sunucuya güvenmek zorundadır.	IoT cihazları sisteme kullanıcı tarafından doğrulandıktan sonra eklenmekte ve bu sırada cihaza özel kullanıcı adı ve parolası belirlenmektedir. Cihazın iletişim kuracağı başlıklar ve içerikleri ile ilgili olarak SSLS baz alınarak aracı yazılımda gerekli tanımlar yapılmaktadır.
Mahremiyet	IoT cihazlarda üretilen veri çeşitli taraflar tarafından kullanıcı rızası olmadan ticari kaygılar ile değerlendirilmektedir.	Orijinal verinin ağ geçidinde silme, bozma veya zenginleştirme teknikleri ile değiştirilerek buluta aktarılması sağlanmıştır.
Güven	Sistemdeki tarafların iddia ettikleri kişi olmaması durumu sistemdeki diğer taraflara zarar verebilir. Örnek olarak, ele geçirilmiş bir klima hırsız alarmını kapatabilir.	SSLS çözümü ile taraflar arasındaki iletişimin içeriğinin ne olması gerektiği aracı tarafından denetlenmektedir. Aracıdan gelen içeriğin doğru olduğu kabul edilmektedir. Buna göre aracı cihazları kullanıcısı aracılığı ile doğruladıktan sonra sisteme dahil etmektedir.

4.2. Performans

IoT cihazlar tarafından IHG kontrolcü uygulamaya gönderilen durum ve istek içerikli MQTT mesajlarının tamamının cihaz SSLS'ine uygun olması gerekmektedir. Bu kapsamda Şekil 3.8'de sözde kodu verilen haberleşme kontrol algoritması geliştirilmiştir. Algoritma temelde JSON formatında gelen mesaj içeriklerini beklenen şablon değerleri ile karşılaştırmaktadır. Ölçümlerde karşılaştırmaların yapılabilmesi için 5, 10, 20, 30, 40 ve 50 parametre sayısına ve yapay değerlere sahip mesajlar kullanılarak, mesaj boyutunun IHG'deki mesaj işleme süresine etkisinin ölçülmesi amaçlanmıştır. Sistemdeki mesajların JSON tabanlı olacağı öngörüldüğünden testlerde senaryoya uygun olması açısından oluşturulan yapay mesajlar JSON formatında tasarlanmıştır. Şekil 4.3'te test için oluşturduğumuz 5 parametreli bir mesajın JSON içeriği görülmektedir.

```
{
  "parametre01":"value01",
  "parametre02":"value02",
  "parametre03":"value03",
  "parametre04":"value04",
  "parametre05":"value05"
}
```

Şekil 4.3. Ölçümde kullanılan 5 parametreli mesaj yapısı

Çizelge 4.3'de mesajlarda parametre sayısına bağlı olarak JSON dönüştürme operasyonu ve gelen mesajın beklenen mesaja uygunluğunun kontrolüne ilişkin karşılaştırma operasyonunun işleme süreleri sunulmuştur. Buna göre, JSON dönüştürme ve mesaj içeriğinin SSLS kuralları ile karşılaştırılması operasyonlarının mesaj boyutu ile ilişkili olarak doğrusal artış gösterdiği görülmektedir. Ölçümlerde JavaScript'in ön tanımlı string eşleştirme algoritmaları kullanılmıştır.

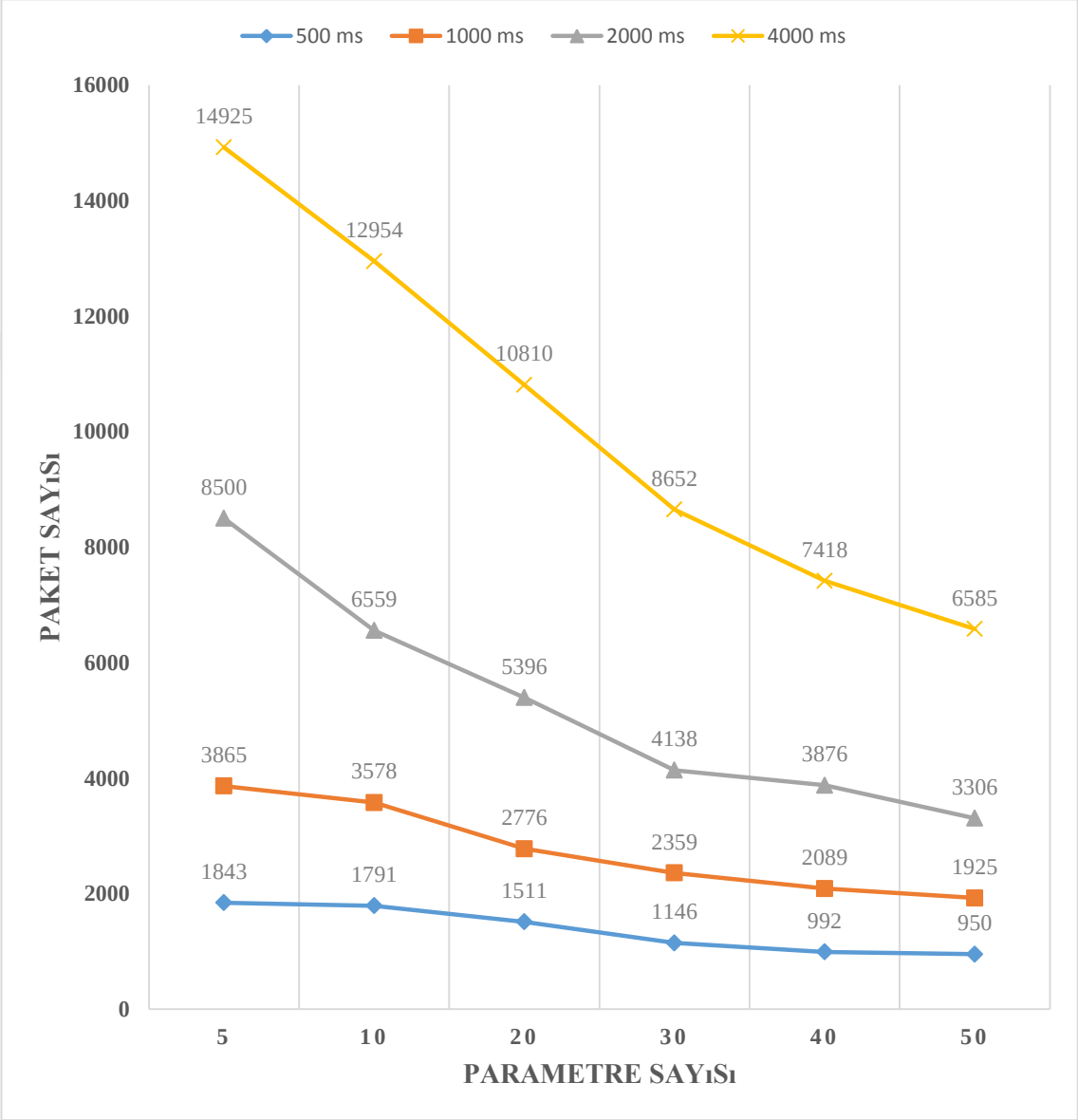
Çizelge 4.3'te gösterilen operasyonlar dışında toplam mesaj işleme süresini etkileyen operasyonlardan biri de cihazın SSLS'inden ilgili kuralların çekilmesidir. Veritabanından yapılan isteklerin asenkron çalışacak şekilde tanımlanması nedeni ile veritabanın operasyonların ne kadar olduğuna ilişkin sağlıklı ölçümler yapılamamıştır. Ancak ortalama toplam işlem süresi gözönüne alındığında yaklaşık 0.07 milisaniye olduğu görülmektedir.

Sistemin çoklu iş parçacıklı ve asenkron çalışmaya olanak tanıması nedeni ile yük altındaki mesaj performansı Çizelge 4.3'ün üçüncü satırında görüldüğü üzere JSON dönüştürme ve karşılaştırma işlemleri toplam maliyetlerinin altında gerçekleştirilebilmektedir. Örnek olarak 50 parametreli bir mesajın sadece JSON dönüştürme ve karşılaştırma maliyetleri 3,86 milisaniye sürmesine rağmen yük altında ortalama mesaj işleme süresi 0,61 milisaniye sürmektedir. Şekil 4.4'te ve Çizelge 4.3'dün son satırında görüldüğü üzere mesajlardaki parametre sayısı arttıkça işlenebilen paket sayısının azaldığı gözlemlenmiştir. Bununla birlikte IHG tarafından akıllı ev sistemlerinin ihtiyaç duyacağı kapasitenin çok üzerinde bir performans sergilendiği de görülmektedir. Gereken durumlarda daha yüksek işlem gücüne sahip donanımlar kullanılarak sonuçlarda iyileştirme sağlanabileceği görülmektedir.

Çizelge 4.3. Farklı boyutta mesaj içerikleri için operasyon maliyetleri

	Parametre Sayısı					
	5	**10**	**20**	**30**	**40**	**50**
JSON Dönüştürme İşlemi	0,1 MS	0,12 MS	0,18 MS	0,24 MS	0,31 MS	0,33 MS
Karşılaştırma İşlemi	0,1 MS	0,26 MS	0,54 MS	1,31 MS	2,32 MS	3,53 MS
Toplam ortalama süre	0,27 MS	0,31 MS	0,37 MS	0,46 MS	0,54 MS	0,61 MS
Saniyede	3865	3578	2776	2359	2089	1925

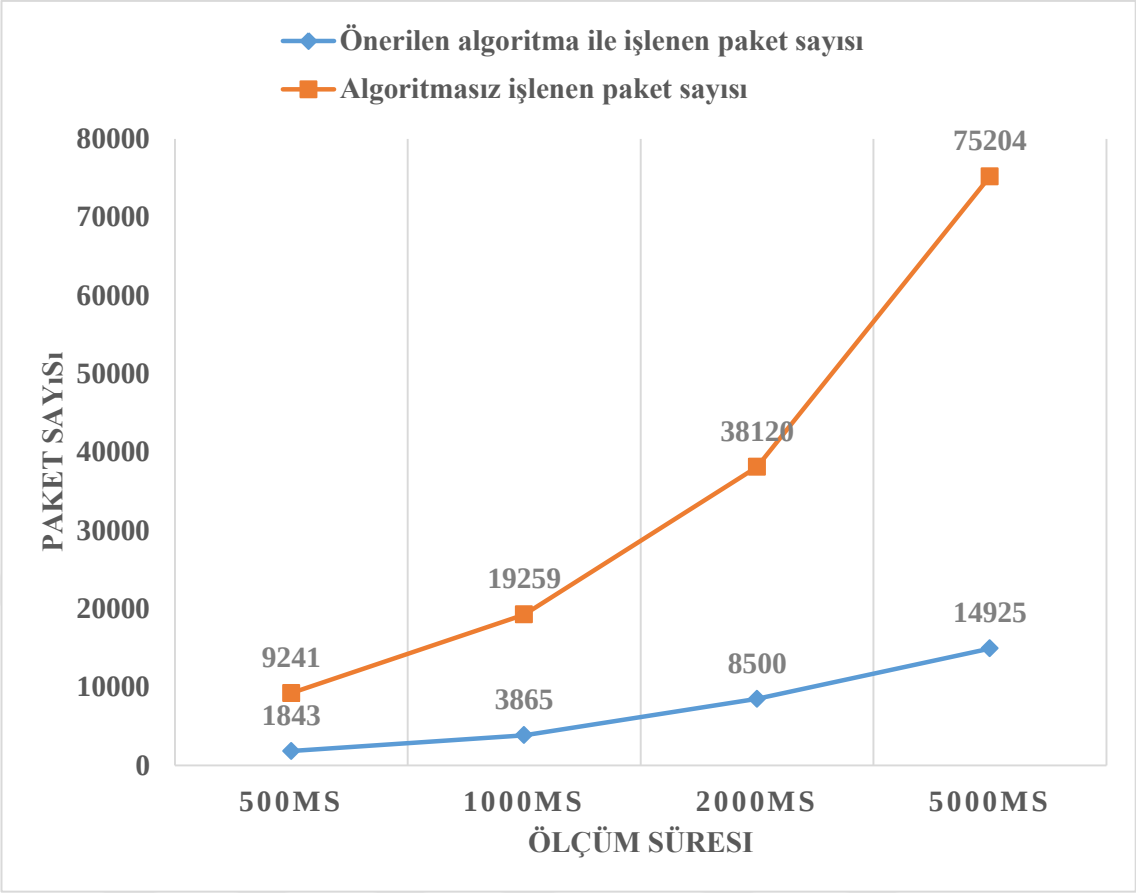
işlenen mesaj sayısı	mesaj/sn	mesaj/sn	mesaj/sn	mesaj/sn	mesaj/sn	mesaj/sn



Şekil 4.4. Farklı parametre sayılarında algoritmanın performansı

Şekil 4.5'te 10 parametreli mesaj trafiği için çalıştırılan kuralların hiç kural olmayan IHG'ye göre bir yük oluşturduğu gözlemlenmektedir. Haberleşme kontrol algoritmasının gecikme süresine etkisinin ölçülebilmesi amacıyla kuralların uygulandığı

ve uygulanmadığı iki duruma ait performans ölçümleri Şekil 4.5'te görülmektedir. Algoritmanın kullanılmadığı durumda sistemin mesaj işleme performansı, algoritmanın 5 parametreli mesajlarda değerlendirildiği duruma göre yaklaşık 4.43 kat daha hızlı olmaktadır. Ancak, tezimizde sunulan yaklaşıma ait paket işleme kapasitesinin (saniyede 3865 mesaj) akıllı ev sistemleri kapsamında ihtiyacın çok üzerinde olması nedeni ile günlük kullanım için uygun olduğu değerlendirilmektedir. Çizelge 4.3'de görüldüğü üzere işlem süresinin büyük çoğunluğunu tip dönüşümü ve string tabanlı karşılaştırma işlemleri almaktadır. Performansın arttırılması amacı ile JSON tabanlı tip dönüştürme işlemleri yerine byte ya da karakter tabanlı karşılaştırma yaklaşımları uygulanabilir. Tezimizde gerekli platformun kurulması hedeflenmiştir. Gerek üretici ve gerekse operatörler tarafından hazırlanacak kurallar sistem performansını doğrudan etkiler. Bu nedenle operatörlerin üreticilerden gelecek kuralları IHG'lere yansıtmadan önce değerlendirmesinde fayda bulunmaktadır. Performans ölçümlerinde bir mesaja sadece bir kural uygulanmıştır. Birden fazla kural sıra ile ya da paralel bir şekilde uygulanması mümkündür. Bir mesaja uygulanacak kural sayısı yine performansı etkileyecek unsurlardandır.



Şekil 4.5. Haberleşme kontrol algoritmasının paket çıktısına etkisi

5. TARTIŞMA ve SONUÇ

Sunulan Tez çalışmasında geleneksel güvenlik yöntemlerinin IoT cihazlarına ya da onların diğer unsurlara vereceği zararların engellenmesinde etkisiz kaldığı probleminden yola çıkılarak güvenlik ve gizlilik odaklı ağ geçidi mimari tasarım oluşturulmuştur. Evdeki bilgisayar ve telefon gibi kullanıcıları tarafından etkin bir şekilde kullanılan cihazlar yeteli güvenlik önlemlerinin alınmaması nedeni ile çoğu zaman trojan gibi yazılımlar aracılığı ile evdeki diğer güvenlik anlamında zayıf IoT cihazlarına zarar vermede kullanılmaktadır. Benzer şekilde nesnelerin interneti cihazlarının tespit edilen açıkları kapatmaması ya da geç kapatmasına bağlı olarak evdeki ve internetteki diğer cihazlara atakta kullanılabilmektedir. IoT cihazlarının merkezde olduğu atakların engellenmesi için tezimizde geliştirdiğimiz platform kullanılabileceği yaptığımız incelemelerde görülmüştür. Tez kapsamında sunulan

yöntemlerin denenmesi amacı ile kullanıcı gereksinimlerinin sağlanmasına yönelik mobil uygulama, IHG ve HMS yazılımları geliştirilmiştir.

Uygulama sahası olarak akıllı ev sistemleri kullanılmıştır. Prototip çalışmaları için RPi cihaz ve USB kablosuz ağ adaptörü genişletmesi ile birlikte OpenWRT işletim sistemi kullanılarak ağ geçidi cihaz donanımı elde edilmiş ve çoklu SSID yaklaşımı uygulanmıştır. IoT cihazların yönlendirilmiş saldırılarda kullanılmalarının önüne geçebilmek amacı ile birbirleri ile ve yerel ağdaki diğer cihazlar ile doğrudan iletişim kurmaları ve internet kaynaklı saldırılardan korunmaları amacı ile doğrudan internete çıkışları Linux IPTables güvenlik duvarı aracılığı ile engellenmiştir. Böylece IoT cihazları birbirlerinden, evdeki diğer cihazlardan ve internetten soyutlanmıştır.

Cihazların bir başka cihaz ya da kullanıcı gibi davranarak sisteme zarar vermesi cihazların üreticileri tarafından tanımlı tanımlar (SSLS) kapsamında hareket etmeleri sağlanması zorunlu kılınarak sağlanmıştır. Cihazın ne olduğu en başta cihazı sisteme ekleyecek kullanıcı tarafından tanımlanmaktadır. Dolayısı ile kullanıcısı tarafından belirli marka ve model'e ait olduğu belirlenen cihaz sadece üreticisi tarafından tanımlı konu başlıklarına belirli mesaj içeriği yapısı ile yazabilir ya da okuyabilir. Böylece, bir klimanın güvenlik sistemini kapatması gibi bir durum söz konusu olamaz. Sunulan platformun bir diğer özelliği ise sistemdeki cihazların veri güvenliğine yönelik olarak pratikte problemli olan asimetrik şifreleme ile anahtar takası yerine doğrudan cihaz bazlı ve cihazın sisteme eklenmesi sırasında belirlenen bir simetrik anahtar ile gerçekleştirmesi sağlanmıştır. Veri mahremiyeti SSLS'lerde tanımlı kurallar çerçevesinde mesaj içeriklerinin fakirleştirilmesi, anonimleştirilmesi ya da gerektiğinde zenginleştirilmesi IHG ile sağlanabilmektedir.

Cihazların güncellenmesi üreticilerin yeterli önlem almadığı takdirde bir güvenlik açığına dönüşmektedir. Çoğu zaman cihazların kısıtlı olmasına bağlı olarak güncellemenin doğrulanmadan yüklenmesi cihazın saldırgan tarafından ele geçirilmesi, başka sistemlere atakta kullanılması gibi sonuçlar ortaya çıkarmaktadır. Sunulan çalışmada IHG üreticisi tarafından ilan edilmiş olan güncellemenin doğrulanması ve sonrasında cihaza ulaştırılmasında aracılık etmektedir.

Yaptığımız ölçümlerde tezimizde sunulan yaklaşımların uygulanması neticesinde saniyede 3865 mesajın içerik kontrolünden geçirildikten sonra yönlendirilmesi sağlanmıştır. Akıllı ev sistemleri için gerekli performans gereğinin çok üzerinde bir performans RPi doanımlı IHG üzerinde elde edilmiştir. Cihaz sayısı ve mesajların artması durumunda birden fazla daha iyi işlemci gücüne sahip IHG veya birden fazla IHG'nin sistemde kullanılması değerlendirilebilir.

Sonuç olarak sunulan platform sayesinde IoT'nin yapısal sorunlarına yönelik çok katmanlı bir çözüm önerilmektir. Kısıtlı donanıma sahip IoT cihazlarının uğrayacağı veya çevresine vereceği zararlar engellenmiştir. Cihazların kimliklerinin ve yetkilerinin ne olacağının belirlenmesi önemli problemlerdendir. Sunulan tez ile kullanıcısı tarafından ne olduğu sisteme tanıtılan cihazın sadece belirli bir şablonda tanımlı konu başlıklarına mesaj atmasının sağlanması ile cihazların etki alanı azaltılmıştır. Ancak, eve dahil olacak olan ürün doğrudan satıcısı tarafından değiştirilmiş bir yazılama sahip ya da atak için üretilmiş olabilir. Böyle bir ürün sisteme eklendikten sonra ağdaki cihazlar hakkında bilgi toplayabilir ve bu bilgileri bir mesajın içerisinde belli etmeden dışarı aktarabilir. IHG ile aynı SSID'ye sahip ağ oluşturup diğer cihazların kendisine bağlanmasını sağlama, ağda gürültü oluşturma, IHG'de DOS atağı gerçekleştirme gibi yöntemler ile ağda servislerin aksamasına neden olabilir. Sisteme dahil olacak üreticilerin güvenilen firmalar olmasının sağlanması, IHG mesaj içeriklerinin açık kapı bırakmayacak şekilde iyi tanımlanması gibi yaklaşımlar saldırganlar ile gerçekleştirilen savaşta kullanılabilir. Tezde sunulan yaklaşım ile IoT cihazlarının zayıf yönlerinin güçlendirilmesine yönelik bir adım atılmıştır.

KAYNAKLAR

Anonim, 2014a. What is MeteorJS. https://joshowens.dev/what-is-meteor-js/-(Erişim Tarihi: 1.12.2019).

Anonim, 2019a. OpenWRT Documentation. https://openwrt.org/docs/start/-(Erişim Tarihi: 1.12.2019).

Anonim, 2019b. Eclipse Mosquitto Documentation https://www.mosquitto.org/-(Erişim Tarihi: 1.12.2019).

Anonim, 2019c. HiveMQ Dökümantasyonu. https://www.hivemq.com/docs/4.2/hivemq/introduction.html-(Erişim Tarihi: 1.12.2019)

Anonim, 2019d. ActiveMQ Dökümantasyonu. https://activemq.apache.org/getting-started-(Erişim Tarihi: 1.12.2019).

Ashton, K. 2009. That 'internet of things' thing. *RFID journal*, 22(7): 97-114.

Atzori, L., Iera, A., Morabito, G. 2010. The internet of things: A survey. *Computer networks*, 54(15): 2787-2805.

Alaba, F. A., Othman, M., Hashem, I. A. T., Alotaibi, F. 2017. Internet of Things security: A survey. *Journal of Network and Computer Applications*, 88: 10-28.

Anh, N. T., Shorey, R. 2005. Network sniffing tools for WLANs: merits and limitations. IEEE International Conference on Personal Wireless Communications, 23-25 January 2005, New Delhi, India.

Andrea, I., Chrysostomou, C., Hadjichristofi, G. 2015. Internet of Things: Security vulnerabilities and challenges. IEEE Symposium on Computers and Communication (ISCC), 6-9 July 2015, Larnaca, Cyprus.

Bull, P., Austin, R., Popov, E., Sharma, M., Watson, R. 2016. Flow based security for IoT devices using an SDN gateway. IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud), 22-24 August 2016, Vienna, Austria

Cho, T. H., Jeon, G. M. 2016. A method for detecting man-in-the-middle attacks using time synchronization one time password in interlock protocol based internet of things. *Journal of Applied and Physical Sciences*, 2(2): 37-41.

Da Xu, L., He, W., Li, S. 2014. Internet of things in industries: A survey. *IEEE Transactions on industrial informatics*, 10(4): 2233-2243.

Deniz, E. 2019. Nesnelerin İnternetinde Gizlilik ve Güvenlik Yönetimi. *Yüksek Lisans Tezi*, AÜ Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Anabilim Dalı, Ankara

Fielding, R. T., Taylor, R. N. 2002. Principled design of the modern Web architecture. *ACM Transactions on Internet Technology (TOIT)*, 2(2): 115-150.

Gajewski, M., Batalla, J. M., Levi, A., Togay, C., Mavromoustakis, C. X., Mastorakis, G. 2019. Two-tier anomaly detection based on traffic profiling of the home automation system. *Computer Networks*, 158: 46-60.

Gasti, P., Tsudik, G., Uzun, E., Zhang, L. 2013. DoS and DDoS in named data networking. 22nd International Conference on Computer Communication and Networks (ICCCN), 30 July-2 August 2013, Nassau, Bahamas.

Haines, B., Thornton, F. 2008. Kismet hacking. Syngress Publishing, New Hampshire, USA, 272pp.

Harrington, W. 2015. Learning Raspbian. Packt Publishing Ltd, Birmingham, UK, 154pp.

Huitsing, P., Chandia, R., Papa, M., Shenoi, S. 2008. Attack taxonomies for the Modbus protocols. *International Journal of Critical Infrastructure Protection*, 1: 37-44.

IETF, 2015. https://tools.ietf.org/html/rfc7519-(Erişim Tarihi: 1.12.2019)

Karhula, P. (2016). Internet of Things: a gateway centric solution for providing IoT connectivity. *Yüksek Lisans Tezi*, KU Consortium Chydenius, Department of Mathematical Information Technology, Kokkola

Katagi, M., Moriai, S. 2008. Lightweight cryptography for the internet of things. *Sony Corporation*, 7-10.

Katz, J., Lindell, Y. 2014. Introduction to modern cryptography. Chapman and Hall/CRC, Boca Raton, USA, 120pp.

Lin, H., Bergmann, N. 2016. IoT privacy and security challenges for smart home environments. *Information*, 7(3): 44.

Lin, J., Yu, W., Zhang, N., Yang, X., Zhang, H., Zhao, W. 2017. A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications. *IEEE Internet of Things Journal*, 4(5): 1125-1142.

Locke, D. 2010. Mq telemetry transport (mqtt) v3. 1 protocol specification. IBM developerWorks Technical Library, 15.

Lueth, K. L. 2018. State of the IoT 2018: Number of IoT devices now at 7B – Market accelerating. https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b-(Erişim Tarihi: 1.12.2019).

Mardan, A. 2014. Express. js Guide: The Comprehensive Book on Express. js. Leanpub, Victoria, Canada, 315pp.

Moazzami, M. M., Xing, G., Mashima, D., Chen, W. P., Herberg, U. 2016. SPOT: A smartphone-based platform to tackle heterogeneity in smart-home IoT systems. IEEE 3rd World Forum on Internet of Things (WF-IoT), 12-14 December 2016, Reston, USA.

Naik, N. 2017. Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP. IEEE international systems engineering symposium (ISSE), 11-13 October 2017, Vienna, Austria.

Nayak, G. N., Samaddar, S. G. 2010. Different flavours of man-in-the-middle attack, consequences and feasible solutions. 3rd International Conference on Computer Science and Information Technology, 9-11 July 2010, Chengdu, China.

Oh, S. R., Kim, Y. G. 2017. Security requirements analysis for the IoT. 2017 International Conference on Platform Technology and Service (PlatCon), 13-15 February 2017, Busan, Korea.

Purdy, G. N. 2004. Linux iptables Pocket Reference: Firewalls, NAT & Accounting. O'Reilly Media, Inc, California, USA, 96pp.

Ramakrishnan, S. 2018. Cryptographic and Information Security Approaches for Images and Videos. CRC Press, Boca Raton, USA, 129pp.

Razzaque, M. A., Milojevic-Jevric, M., Palade, A., Clarke, S. 2015. Middleware for internet of things: a survey. *IEEE Internet of things journal*, 3(1): 70-95.

Red Hat Inc., 2013. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0160-(Erişim Tarihi: 1.12.2019)

Sarigiannidis, P., Karapistoli, E., Economides, A. A. 2015. Detecting Sybil attacks in wireless sensor networks using UWB ranging-based information. *Expert Systems with Applications*, 42(21): 7560-7572.

Stafford-Fraser, Q. 1995. The trojan room coffee pot. Np, May.

Strack, I. 2015. Getting Started with Meteor. js JavaScript Framework. Packt Publishing Ltd, Birmingham, UK, 130pp.

Swan, T. L., McKinney, D. U., 2012. Providing quality of service (QOS) using multiple service set identifiers (SSID) simultaneously. U.S. Patent and Trademark Office, U.S. Patent No. 8,184,530, Washington.
Tilkov, S., Vinoski, S. 2010. Node. js: Using JavaScript to build high-performance network programs. *IEEE Internet Computing*, 14(6): 80-83.
Togay, C. 2018. A New Message Processing Mechanism for Internet of Things. *Uludağ University Journal of The Faculty of Engineering*, 23(2): 55–66.
Togay, C., Mutlu, G., Kurtuluş, D., Özgür, F. 2019a. Secure Gateway for the Internet of Things. *Avrupa Bilim ve Teknoloji Dergisi*, 16: 414-426.
Togay, C., Ozgur, F., Kiraz, G., Kurtulus, D., 2019b. Rule Verification Mechanism for IoT, Ulusal Yazılım Mühendisliği Sempozyumu (UYMS), 23-25 September 2019, Urla, İzmir.
Wargo, J. M. 2015. Apache Cordova 4 Programming. Pearson Education, Indiana, USA, 79pp.
Verma, P. K., Verma, R., Prakash, A., Agrawal, A., Naik, K., Tripathi, R., Alsabaan, M., Khalifa, T., Abdelkader, T., Abogharaf, A. 2016. Machine-to-Machine (M2M) communications: A survey. *Journal of Network and Computer Applications*, 66: 83-105.
Vinoski, S. 2006. Advanced message queuing protocol. *IEEE Internet Computing*, 6: 87-89.
Xu, W., Ma, K., Trappe, W., Zhang, Y. 2006. Jamming sensor networks: attack and defense strategies. *IEEE network*, 20(3): 41-47.
Yaqoob, I., Hashem, I. A. T., Mehmood, Y., Gani, A., Mokhtar, S., Guizani, S. 2017. Enabling communication technologies for smart cities. *IEEE Communications Magazine*, 55(1): 112-120.
Zeljka, Z., 2018. Open Source Code Security Risk. https://www.helpnetsecurity.com/2018/05/22/open-source-code-security-risk/-(Erişim Tarihi: 1.12.2019).
Zhang, C., Green, R. 2015. Communication security in internet of thing: preventive measure and avoid DDoS attack over IoT network. Society for Computer Simulation International, 12-15 April 2015, Virginia, USA.

ÖZGEÇMİŞ

Adı Soyadı : Ahmet KAŞİF
Doğum Yeri ve Tarihi : Osmangazi / Bursa, 1995
Yabancı Dil : İngilizce

Eğitim Durumu
Lise : Ahmet Vefik Paşa Anadolu Lisesi, Bursa, 2013
Lisans : Celal Bayar Üniversitesi Bilgisayar Mühendisliği, 2017
Yüksek Lisans : Bursa Uludağ Üniversitesi Bilgisayar Mühendisliği, 2020

Çalıştığı Kurum/Kurumlar (Devam Etmekte) : Bursa Teknik Üniversitesi, Araştırma Görevlisi, 2018 –

İletişim (e-posta) : ahmet.kasif@btu.edu.tr

Yayınları :