T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BLOKZİNCİR VE AKILLI SÖZLEŞMELER KULLANILARAK YENİ BİR GÜVENLİK TASARIMININ IOT İÇİN UYGULANMASI

DOKTORA TEZİ

Emre KARAKOÇ

Enstitü Anabilim Dalı	**:**	**BİLGİSAYAR VE BİLİŞİM MÜHENDİSLİĞİ**
Tez Danışmanı	**:**	**Prof. Dr. Celal ÇEKEN**

Ağustos 2021

T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BLOKZİNCİR VE AKILLI SÖZLEŞMELER KULLANILARAK YENİ BİR GÜVENLİK TASARIMININ IOT İÇİN UYGULANMASI

DOKTORA TEZİ

Emre KARAKOÇ

Enstitü Anabilim Dalı : BİLGİSAYAR VE BİLİŞİM MÜHENDİSLİĞİ

Bu tez 27/08/2021 tarihinde aşağıdaki jüri tarafından oybirliği/oyçokluğu ile kabul edilmiştir.

Jüri Başkanı	**Üye**	**Üye**
D Üye		**Üye**

BEYAN

Tez içindeki tüm verilerin akademik kurallar çerçevesinde tarafımdan elde edildiğini, görsel ve yazılı tüm bilgi ve sonuçların akademik ve etik kurallara uygun şekilde sunulduğunu, kullanılan verilerde herhangi bir tahrifat yapılmadığını, başkalarının eserlerinden yararlanılması durumunda bilimsel normlara uygun olarak atıfta bulunulduğunu, tezde yer alan verilerin bu üniversite veya başka bir üniversitede herhangi bir tez çalışmasında kullanılmadığını beyan ederim.

Emre KARAKOÇ
27.08.2021

TEŞEKKÜR

Doktora eğitimim süresince her türlü desteğini esirgemeyen eşime ve aileme teşekkür ederim. Ayrıca değerli bilgi ve deneyimlerinden yararlandığım, her konuda bilgi ve desteğini almaktan çekinmediğim, araştırmanın planlanmasından yazılmasına kadar tüm aşamalarında yardımlarını esirgemeyen, teşvik eden, aynı titizlikte beni yönlendiren değerli danışman hocam Prof. Dr. Celal ÇEKEN'e teşekkürlerimi sunarım.

İÇİNDEKİLER

SİMGELER VE KISALTMALAR LİSTESİ

AODV	: İsteğe Bağlı Mesafe Vektörü Yönlendirme (Ad-hoc On-Demand Distance Vector)
BS	: Baz İstasyonu (Base Station)
CH	: Küme Başı (Cluster Header)
DApp	: Merkezi Olmayan Uygulama (Decentralized Application)
ED	: Son Cihaz (End Device)
EVM	: Ethereum Sanal Makinesi (Ethereum Virtual Machine)
IPFS	: Gezegenler Arası Dosya Sistemi (InterPlanetary File System)
IoT	: Nesnelerin İnterneti (Internet of Things)
KAA	: Kablosuz Algılayıcı Ağlar
MOSS	: Multi-OPs Spectrum Sharing
MSC	: Üretici Akıllı Sözleşmesi (Manufacturer Smart Contract)
Pop-Es	: Point of Presence-ES / RNP
QoS	: Servis Kalitesi (Quality of Service)
RREP	: Rota İstek Cevap Mesajı (Route Reply Message)
RREQ	: Rota İstek Mesajı (Route Request Message)
SBC	: Tek Kart Bilgisayar (Single Board Computer)
SDN	: Software Defined Networks
SLA	: Hizmet Seviyesi Anlaşması (Service Level Agreement)
SNR	: Sinyal Gürültü Oranı (Signal Noise Ratio)
SSC	: SLA Akıllı Sözleşmesi (SLA Smart Contract)
WSN	: Wireless Sensor Networks
YTA	: Yazılım Tanımlı Ağlar

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

ÖZET

Anahtar kelimeler: Nesnelerin İnterneti, Yazılım Tanımlı Ağlar, Kablosuz Algılayıcı Ağlar, Blokzincir, Akıllı Sözleşmeler, Hizmet Seviyesi Anlaşması, Kara Delik Saldırısı

Kablolu ortamda Yazılım Tanımlı Ağ (YTA) teknolojisinin başarısı, Kablosuz Algılayıcı Ağlar (KAA) üzerinde YTA'nın konuşlandırılmasına ilişkin araştırmaları hızlandırmıştır. Kablosuz ortamın paylaşımlı bir doğası olduğundan, yönlendirme saldırılarına karşı daha savunmasızdır. Kablosuz ortamın fiziksel kısıtlamaları ve ağdaki düğümlerin sınırlı enerjisi, işleme kapasitesi ve belleği olduğundan, mevcut güvenlik protokollerinin çoğu KAA için uygulanamaz.

Bu çalışma ile değişmezlik ve şeffaflık ilkesini temel alan Blokzincir teknolojisi ele alındı. YTA tabanlı KAA üzerinde Blokzincir teknolojisi uygulanarak enerji dostu yeni bir güvenlik modeli geliştirildi. YTA tabanlı KAA'daki olası tehditler araştırıldı ve önerilen modelin güvenirliğini ispat etmek için Kara Delik saldırısı ile model ayrıntılı analiz edildi. Ayrıca önerilen model üzerinde bir Blokzincir teknolojisi olan Akıllı Sözleşmeler kullanıldı. Üretici, hizmet sağlayıcı ve müşterinin taraf olduğu yeni bir güvenli SLA yönetim modeli geliştirildi. Önerilen model güvenilir ürün yaşam döngüsü ve güvenilir SLA sözleşmeleri içermektedir. Önerilen modeldeki yenilikçi yaklaşım sayesinde müşteri ile servis sağlayıcı arasındaki olası anlaşmazlıklar ortadan kaldırıldı. Böylece üretilen ürünün kullanım aşamasında kurcalanabilirlik açısından güvenilirliği sağlandı. Önerilen model var olan modele kıyasla ek veri iletim gecikmesi ve enerji tüketiminde artış meydana getirse de elde edilen simülasyon sonuçları aradaki yüzdesel farkın önemsenmeyecek seviyede (<%1.6) olduğunu göstermektedir. Sonuçlar, önerilen modelin YTA tabanlı KAA'nın güvenliğini sağlamak için umut verici bir çözüm olabileceğini göstermektedir.

IMPLEMENTING A NEW SECURITY DESIGN FOR IOT USING BLOCKCHAIN AND SMART CONTRACTS

SUMMARY

Keywords: Internet of Things, Software Defined Networks, Wireless Sensor Network, Blockchain, Smart Contracts, Service Level Agreement, Black Hole Attacks.

The success of Software Defined Network (SDN) technology in the wired environment has accelerated research into the deployment of SDN in Wireless Sensor Networks (WSN). Since the wireless environment has a shared nature, it is more vulnerable to routing attacks. Because of the physical constraints of the wireless environment and the limited energy, processing capacity and memory of the nodes in the network, many of the existing security protocols cannot be applied to WSN.

In this study, Blockchain technology, which is based on the principle of immutability and transparency, was discussed. A new energy-friendly security model was developed by applying Blockchain technology on SDN-enabled WSN. The possible threats in SDN-enabled WSN were investigated and the model was analyzed in detail with the Black Hole attack to prove the reliability of the proposed model. In addition, Smart Contracts, a Blockchain technology, were used on the proposed model. A new secure SLA management model has been developed in which the manufacturer, service provider and customer are parties. The proposed model includes reliable product lifecycle and reliable SLA contracts. Thanks to the innovative approach in the proposed model, possible conflicts between the customer and the service provider were eliminated. Thus, the reliability of the product in terms of tampering was ensured during the use phase. Although the proposed model causes additional data transmission delay and an increase in energy consumption compared to the existing model, the simulation results show that the percentage difference between them is negligible (<1.6%). The results show that the proposed model could be a promising solution to secure SDN-enabled WSN.

BÖLÜM 1. GİRİŞ

Nesnelerin interneti, gelişen yeni teknolojilerle birlikte akıllı şehirlerde iş ve yaşam alanlarında insanlara yeni imkânlar (bina enerji yönetim sistemleri, akıllı fabrikalar, hassas tarım, e-sağlık sistemleri, akıllı evler, vb.) sunmaktadır [1]. Özellikle Kablosuz Algılayıcı Ağlar (KAA) birbirleriyle etkileşime giren düğümler ve düğüm grupları sayesinde Nesnelerin İnterneti (Internet of Things - IoT) tabanlı sistemlerin dikey alanında yaygın olarak kullanılmaktadır. KAA'nın doğası gereği, kaynak ile hedef arasındaki verilerin iletimi kablosuz olarak gerçekleştirilir. Bu durum, sinyal sahtekarlığı, ağa ağ iletişim mesajlarının yerleştirilmesi, aktarım sırasında mesajların değiştirilmesi ve en kısa yolun atlanması gibi zayıflıklara neden olabilir [2]. Herhangi bir KAA'da şifre kırma potansiyeline sahip çok sayıda saldırı vektörü ve tehdit vardır. Düğümler arasındaki iletişimi kesintiye uğratmak ve yanlış yönlendirmeye neden olmak en önemli saldırılar arasındadır [2]. Ayrıca ağın veri iletimi boyunca birden fazla atlama noktası vardır ve bu da onu çeşitli saldırı türlerine karşı savunmasız kılar [3], [4]. Yaygın kablosuz ağ saldırılarından bazıları şu şekilde sıralanabilir [5].

- Solucan Deliği (Wormhole) Saldırısı
- Obruk (Sinkhole) Saldırısı
- Seçici Yönlendirme Saldırısı
- Kara Delik (Blackhole) Saldırısı
- Gri Delik (Grayhole) Saldırısı
- Hizmeti engelleme saldırısı

Yapılan bir çalışma, Kara Delik saldırılarının LEACH protokolü üzerinde önemli bir etkiye sahip olduğunu göstermiştir [6], [7]. Güvenlik hizmetlerini sağlamak için genellikle üç tür şifreleme ilkesi kullanılır; açık anahtar ilkeleri, özel anahtar ilkeleri ve karma işlevler [8]. Literatürdeki birçok araştırmacı, KAA için bu ilkelerin etkili uygulama tasarımlarını analiz etmiştir. Sensör düğümlerinin düşük enerji kapasitesi, yetersiz depolama alanı ve sınırlı çalışma gücüne sahip olması nedeniyle şifreleme yöntemlerinin uygulanabilirliği zordur ve bu durum yönlendirme mekanizmalarını da kısıtlamaktadır [8]. Ayrıca düğümler üzerinde şifreleme işlemlerinin veya güvenlik protokollerinin uygulanması önemli performans kayıplarına neden olabilir. Bu nedenle, optimize edilmiş ağ operasyonu ve enerji kayıplarını en aza indirmek için yüksek donanımlı Güven Merkezi (Trust Center) gibi merkezi bir otoriteye ihtiyaç vardır. M. Al-Hubaishi ve ark. mevcut geleneksel KAA'dan farklı bir model geliştirmişlerdir. YTA altyapısını KAA üzerinde uygulayarak yeni bir KAA mimarisi olan YTA tabanlı KAA'yı (SDN-enabled WSN) önermişlerdir [9]. Ağdaki düğümlerin cihaz bilgilerinin ve Akış Tablosu bilgilerinin etkileşiminin IEEE 802.15.4 protokolü [10] aracılığıyla sağlandığı yeni bir yönlendirme keşif mekanizması ve özgün bir mimari sunmuşlardır. Modelde, YTA Denetleyici (SDN Controller - SDNC) cihazını ağda merkezi bir otorite olarak atamışlardır. Bu yaklaşımla, yönlendirme ve ağ performansı izleme gibi tüm ağ işlemleri SDNC'ye atanmıştır. Bu stratejinin düşük kapasiteli algılama cihazlarında performansı artırabileceği düşünülse de yeni yaklaşımlar yeni güvenlik açıklarına yol açabilir. Ayrıca kablosuz ağ ortamının belirttiğimiz bu olası riskleri hassas verilerin ağ üzerinde yönetilmesini daha güç hale sokmaktadır. Örneğin dinamik altyapı imkânı sunan KAA mimarisinin kullanıldığı bazı akıllı şehir uygulamalarında veri aktarımının gerçek zamanlı olarak yapılması müşteri tarafından talep edilebilir. Fakat kablosuz ağdaki düğümler düşük enerjili cihazlar olduklarından beklenmedik şekilde enerjileri tükenebilir ve çalışma anında kullanım dışı olabilirler ya da bir saldırgan tarafından ağ manüpüle edilebilir. Bu nedenle kablosuz ağ ortamı kullanıcıları kendi ağ standartlarını karşılayacak veya güvenliğinden emin oldukları bir ağ ortamı talep edebilirler. Bu standartların sağlanması için ağ altyapısını yöneten servis sağlayıcı, ağda meydana gelen tüm değişiklikleri sık bir şekilde izlemesi gerekebilir. Ağdaki

tüm katmanları aynı anda ele almalı, yeni bir istemci düğüm bağladığında trafik yükünü dengeleyecek şekilde yönetimini sağlamalıdır [11]. Bu nedenle kablosuz ağ ortamınında veri güvenliğinin yanı sıra yönetiminin de güvenli şekilde yapılması büyük önem arz etmektedir. Kablosuz ortam için yapılacak bir Hizmet Seviyesi Anlaşması'nda (Service Level Agreement - SLA) kablolu ortama nazaran daha fazla anlaşmazlık çıkması muhtemeldir. Hizmet sağlayıcı ve müşteri, anlaşmazlığın artmasıyla daha maliyetli bir bürokrasi sürecini yaşamak zorunda kalabilirler. Gerekli ödemeyi sağlamak için banka veya finansal hizmetler gibi Güvenilir Üçüncü Taraf (Trusted Third Party) bir şirkete güvenmek zorundadırlar. Bu nedenle, maliyeti düşük karmaşık olmayan ve her iki taraftan ödemenin doğru şekilde yapılmasını sağlayan bir çözüme ihtiyaç vardır. Değişmezlik temel ilkesi üzerine geliştirilen Blokzincir (Blockchain) ve bu sistem üzerinde çalışan Akıllı Sözleşmelerin (Smart Contracts) bizlere güvenilir bir ödeme sistemi ve depolama alanı sağlayarak bu sorunların çözülmesinde yardımcı olabileceği düşünülmektedir. Bu anlamda, esnek geliştirme altyapısı sunan Ethereum gibi bir Blokzincir mimarisi sayesinde SLA'ları ifade etmek için kullanılan kablosuz ağ Servis Kalitesi (Quality of Service – QoS) metriklerini kurcalanmaya karşı bir Akıllı Sözleşme tanımlanabilir. Böylece hizmet ücretleri ve tazminat değerlerinin ödenmesini sağlaması açısından aracı bir kurum ihtiyacı ortadan kalkacaktır. Ayrıca servis sağlayıcılar Akıllı Sözleşme sayesinde güvenilirliğini ispatlayabileceklerdir.

Bu çalışmada YTA tabanlı KAA mimarisi ele alınmış ve Blokzincir teknolojisi kullanarak yeni yaklaşımlar geliştirilmiştir. Birinci çalışmada, saldırılara karşı Blokzincir teknolojisini bu mimari üzerinde uygulayarak yeni bir güvenlik modeli geliştirilmiştir. YTA tabanlı KAA'da daha önce tartışılmayan güvenlik açıkları analiz edilmiştir. Ayrıca önerilen modelin güvenirliğini ve performansını değerlendirebilmek için önerilen model simülasyon üzerinde bir Kara Delik saldırısına karşı kullanılmış ve simülasyon sonuçları elde edilmiştir. Birinci çalışmanın katkıları şu şekilde sıralanabilir;

- Bu çalışma ile YTA tabanlı KAA mimarisi üzerinde literatürde benzeri olmayan Blokzincir-blok yaklaşımına sahip yeni bir güvenlik modeli geliştirildi ve bu mimari üzerinde güvenlik açıkları ilk kez tartışıldı.
- KAA düşük enerji ve çalışma kapasitesine sahip düğümler içerdiğinden, tamamen hafif bir güvenlik protokolü geliştirmek geliştiriciyi kısıtlamaktadır. Bu çalışma ile veri güvenliği ve veri bütünlüğü sağlayan Blokzincir'den ilham alan enerji dostu bir sistem altyapısı önerilmiştir.

İkinci çalışmada ise YTA tabanlı KAA mimarisinin Akıllı Şehirler gibi yeni teknolojilerin yaygın kullanıldığı alanlarda kulllanımının artacağını öngörerek son kullanıcıyı koruyan Akıllı Sözleşme tabanlı fütürist bir SLA Yönetim modeli geliştirilmiştir. Hem müşterilere taahhüt edilen ağ metriklerinin takip edilmesini hem de müşteri ve servis sağlayıcı arasında üçüncü taraf gerektirmeden SLA ödeme ve tazminat sürecinin otomatize şekilde yürütülmesi sağlanmıştır. Geliştirdiğimiz bu model sayesinde ağ metrik verilerinin Akıllı Sözleşme'ye aktarımı esnasında dışarıdan herhangi bir kurcalamaya karşı verilerin doğruluğu garanti edilmiştir. Böylelikle hem müşteri hem de servis sağlayıcı için tüm SLA çıktıları güvenilir kabul edilmiştir. İkinci çalışmanın katkıları şu şekilde sıralanabilir;

- Literatürde özellikle son zamanlarda Akıllı Sözleşmeler kullanılarak çeşitli modeller önerilmiş olsa da önerilen model Akıllı Sözleşmelerin KAA mimarisine uygulanabilir olduğunu kapsamlı olarak gösteren ilk çalışmadır.
- Düşük enerjili ve performans açısından kısıtlı düğümlerin ağ metriklerinin denetlenebileceği Akıllı Sözleşme tabanlı bir model geliştirilmiştir. Böylelikle kablosuz ağların esnek yapısı sayesinde Akıllı Şehir uygulamalarının birçok alanında uygulanabilir olduğunu göstererek yeni bir bakış açısı kazandıracaktır.
- Akıllı Sözleşmeler uygulamalara her ne kadar değişmezlik ve tarafsızlık sağlasa da verilerin Akıllı Sözleşmeye gönderilmesi aşamasında üçüncü taraf güvensizliği ortaya çıkmaktadır. Önerilen model ile bu sorun aşılmış ve çift

taraflı kontrol mekanizması sayesinde verinin doğrulanabilir olması sağlanmıştır. Ayrıca sözleşmeye gönderilen veri sayısı teyit edilmiştir. Böylelikle art niyetli olaylara sebebiyet vermemek için servis sağlayıcı ve müşteri güvence altına alınmıştır.

- Önerilen modelde, kablosuz sensör düğümlerin üretiminden satışına, satışından müşteriye kadar tüm aşamaları kapsayan ve tedarik zincirini de içeren küresel bir ekosistem önerilmiştir. Modelde yer alan tedarik zinciri sayesinde bir cihazın orijinal olup olmadığı hususunda üretiminden müşterinin kullanım aşamasına kadar güven sağlanmıştır.

BÖLÜM 2. LİTERATÜR İNCELEMESİ

Literatürde KAA'ların güvenliği üzerine çeşitli çalışmalar yapılmıştır. Bu çalışmalar genellikle ağ iletişimini bozmaya ve değiştirmeye yönelik saldırılara karşı geliştirilen güvenlik modellerini kapsamaktadır. Özellikle İsteğe Bağlı Mesafe Vektörü Yönlendirme (Ad-Hoc On-Demand Distance Vector - AODV) protokolü üzerinde yapılan saldırıların araştırıldığı birçok çalışma bulunmaktadır. Bu nedenle Blokzincir teknolojisi tabanlı önerilen modeller ise genellikle AODV protokolü üzerinde çalışılarak oluşturulmuştur. YTA tabanlı KAA mimarisi yeni bir yaklaşım olduğundan güvenlik üzerine mevcut bir çalışma bulunmamaktadır. Ayrıca mimarisinde YTA Denetleyici gibi güçlü performansa sahip bir cihaz bulunması Blokzincir yaklaşımının uygulanabilirliğini diğer KAA'lara göre artırmıştır. Bu kazanımla Blokzincir teknolojisi farklı bir yaklaşım ile kullanılmış ve KAA'larda SLA yönetim altyapısı oluşturulmuştur. Bu sayede YTA tabanlı KAA'da SLA yönetimini Blokzincir teknolojisi ile gerçekleştiren ilk çalışma olmuştur.

2.1. Kara Delik Saldırısını Önlemeye Yönelik Önerilen Geleneksel Güvenlik Modelleri

[7], [12], [13], [14], [15], [16], [17] çalışmalarında, ağdaki düğümler gruplara ayrılmış ve yüksek verimli görev düğümleri olan Küme Lideri (Cluster Header - CH), her grup için belirlenmiştir. Bu düğümler grup içinde merkezi bir yetkiye sahiptir ve komşu tablosu ve itibar tablosu gibi gruptaki diğer düğümlerin bilgileri tutulmuştur. Ayrıca CH, gerektiğinde düğümün geçmiş etkileşimlerini kontrol ederek komşuların güvenini belirlemiştir. Bu çalışmalardan bazılarında Rota İstek (Route Request - RREQ) mesajı paketine id içeren bir alan eklenerek rota güvenliği amaçlanmıştır. Bu bilgiler kullanılarak zararlı düğüm tespiti yapılabilir ve gerekirse gruplar yeniden düzenlenebilir. [15], [16], [18] ve [19] çalışmalarında Baz İstasyonu

(Base Station - BS), merkezi otorite olarak seçilmiştir. BS, ağdaki düğümleri yetkilendirmiş ve anahtar paylaşımını sağlamıştır. Aynı zamanda BS, CH seçiminden ve enerji seviyelerinin izlenmesinden sorumludur. Seçilen CH'lerin veya düğümlerin enerjilerinde ani bir düşüş olması kötü niyetli olarak kabul edilmiştir. [20] ve [21]'de kontrol paketleri grup liderlerini ve komşu düğümleri belirlemek için kullanılmıştır. Kara Delik tespiti, kontrol paketleri aracılığıyla belirli eşik değerleri kullanılarak yapılmıştır. Kara Delik tespiti ile ilgili başka bir çalışmada [22], iki düğüm arasında güven yönetimi için bulanık mantık kullanılmıştır. Her düğüm için bir güven değeri belirlenmiş ve komşu düğümlerin güven değerlerini korumaktan sorumlu olduğu belirtilmiştir. Güven değeri belirli bir eşik değerin altına düşen düğümler zararlı olarak tanımlanmıştır. [23]'da geliştirilen modelde ise CH, güven değerlerine bağlı olarak gruptaki kayıtlı düğümlere anahtarlar atamıştır. Eski düğümler, yalnızca güven değerleri eşikten yüksek olduğunda yeni anahtarlar alabilmişlerdir. Bu parametreler kullanarak kötü düğümleri iyi olanlardan ayırmak amaçlanmıştır.

Saldırı tespiti için düğümlerin yanıt süresi eşikleri de bazı çalışmalarda ana faktör olarak kabul edilmiştir. Bu çalışmaların birinde [24], her düğüm komşu düğümlerin bir tablosunu tutmuştur. Her düğümü ziyaret eden ve gelen trafiğin akışını izleyen kontrol aracıları adı verilen özel düğümler ağa eklenmiştir. [25]'de ise HT (Hybrid Techniques) adı verilen bir teknik kullanılmıştır. Saldırı, paket ve zaman gecikme değerleri yardımıyla belirlenmiştir. Kaynak düğüm ağa RREQ gönderir ve belirli bir süre bekler. Her iki çalışmada da başarılı iletişim hedeflendiğinden düğümlerin süresi dolmadan yanıt vermesi beklenmiştir. Aksi takdirde, düğüm Kara Delik olarak işaretlenmiştir.

[8], [26], [27] ve [28] çalışmalarında, karma şifreleme algoritmasının diğer şifreleme algoritmalarından daha iyi performansa sahip olduğu ileri sürülmüştür. Tüm bu çalışmalar, KAA'daki cihazların sınırlı hafızaya ve düşük enerjiye sahip olduğuna işaret etmektedir. Bu nedenle ağ iletişimini anahtarlamak için tek yönlü bir karma zincir yöntemi önermişlerdir.

2.2. Kara Delik Saldırısını Önlemeye Yönelik Önerilen Blokzincir Tabanlı Güvenlik Modelleri

Literatürde Blokzincir yaklaşımı kullanılarak KAA'da güvenliği sağlamayı amaçlayan bazı çalışmalar yer almaktadır. Yönlendirme düğümleri arasındaki güvenilirliği artırmak için Blokzincir teknolojisinin kurcalamaya karşı korumalı ve izlenebilir özellikleri kullanılarak çeşitli güvenlik modelleri önerilmiştir. Bu modeller ise mevcut yönlendirme algoritmalarıyla birleştirilerek oluşturulmuştur [29], [30], [31], [32]. Alok Kumar ve ark. [33] KAA'da Blokzincir yaklaşımı kullanarak düğümlerin yanlış raporlarının BS'ye iletilmesini önleyen bir model önermişlerdir. Bu yaklaşımda ağ modeline ağdaki normal algılama düğümlerinden farklı olarak yüksek işlem gücüne sahip düğümler olan h-düğümleri eklenmiştir. Herhangi bir veri aktarım işleminden önce veriler belirli bir h-düğümüne rapor edilir. Sonuç olarak h-düğümü bunu ekler ve özel bir Blokzincir ağına rapor verilir. Önerdikleri modelde her biri madenci olarak hareket eden ve fikir birliğine varan 3 adet h düğümü yer almıştır. Sadhu Ram Basnet ve Subarna Shakya benzer bir çalışmada küçük bir veri merkezi tasarlayarak YTA altyapısı ile Open Stack ile işbirliği yapan bir Blokzincir teknolojisini uygulamışlardır [34]. Blokzincir sayesinde güvenilmeyen bir üyenin doğrulama olmaksızın diğer üyelerle etkileşime girmesi engellenmiştir. Alt ağ bloklara bölünmüş ve bir üye başka bir üyeye dosya göndermek istediğinde bu işlem ilgili işlem bloğuna eklenerek tüm üyelere dağıtılmıştır. Veri aktarım işlemine mutabakat halinde izin verilmiştir. Pradip Kumar Sharma ve ark. YTA mimarisindeki Akış Tablosu sürümlerini güncel ve güvenilir bir şekilde diğer IoT cihazlarına aktarmak için IoT ağlarında Blockchain teknolojisini kullanmışlardır [35]. Modellerinin performanslarını ise Kullanıcı Veri Birimi Protokolü (User Datagram Protocol – UDP) ile Sel Saldırısı altında araştırmışlardır.

Blokzincir teknolojisi yüksek performanslı hesaplamalar içerdiğinden düşük enerjili cihazlara uygulanması oldukça zordur. Bu nedenle, çalışmaların çoğu Blokzincir yaklaşımını geleneksel KAA veya YTA mimarisine uygulamaktadır. Bazı çalışmalarda, KAA ağının tamamı bir Blokzincir ağına dönüştürülür ve her düğüm bir defter olarak kullanılır. Bu yaklaşımı özellikle düşük depolama ve performans

kapasitesine sahip cihazlar için kullanışlı olmadığı düşünülmektedir. Bu nedenle önerilen modelde, düşük enerjili düğümlere sahip YTA tabanlı KAA mimarisi için yüksek işlem gücü gerektirmeyecek Blokzincir-blok yaklaşımını kullandık. Diğerlerinden farklı olarak, düğümlerde aşırı enerji tüketimi gerektirmeyecek ve ağ performansını önemli ölçüde etkilemeyecek yeni bir model geliştirdik.

2.3. Kablosuz Ağlarda Blokzincir Tabanlı Önerilen SLA Yönetim Modelleri

Literatürde SLA yönetimi için Blokzincir ve Akıllı Sözleşme kullanılarak önerilen birçok çalışma mevcuttur. Yapılan iki çalışmada [36], [37] ağların bakımı, yönetimi, planlaması ve geliştirilmesiyle ilgili çeşitli hizmetler sunan Point of Presence-ES / RNP (Pop-Es) üzerinde SLA izleme sürecini otomatikleştirmeyi amaçlayan Akıllı Sözleşmelerle güçlendirilmiş bir model önermişlerdir. Geliştirdikleri Akıllı Sözleşme'yi, halka açık bir Ethereum test ağı olan Ropsten ağında saklamışlardır. Önerdikleri sistemde Blokzincir ağından saklanması maliyetli olan dosyaları merkezi olmayan bir dosya sistemi Gezegenler Arası Dosya Sistemi (InterPlanetary File System - IPFS)'de saklamışlardır. Diğer çalışmada [38], müşteri ve servis sağlayıcı arasında olası bir tazminat sürecini otomatize eden Ethereum tabanlı Akıllı Sözleşme geliştirmişlerdir. Önerdikleri model ile hem servis sağlayıcı hem de müşteri Akıllı Sözleşme'de yer alan SLA'nın şartlarını baştan kabul ederek Blokzincir ağına yüklemişlerdir. SLA şartlarını test edebilmek için bir web sunucu üzerinde yer alan bir PHP siteyi referans alarak cevap verme sürelerini kullanmışlardır. Başka bir çalışmada ise [39], Akıllı Sözleşmeler ile dinamik SLA'ları dağıtılmış bir şekilde yönetmek için bir framework önermişlerdir. Bu framework ile izleme verilerinin toplanmasını, doğrulanmasını, ağ kalitesindeki değişiklikleri ve hizmet ödeme sistemini yönetmişlerdir. Kapalı zincir ve genel zincir olmak üzere iki farklı Blokzincir ağ kullanmışlardır. İlk ağı, SLA'ların yönetimi ve gerekli ağır hesaplamalar için kullanılırken, ikinci ağı Akıllı Sözleşme'yi yürütmek için kullanmışlardır. Yazarlar benzer bir çalışma daha yapmışlardır. Bu çalışmada ise önceki çalışmaya [40] ilaveten Akıllı Sözleşmelere dönüştürülmüş olan dinamik SLA'ların tanımlanması ve yönetimi için yeni bir dil ve ilişkili bir çatı mimari geliştirmişlerdir. SLA uygulaması için Oracle'dan yararlanılarak karar verici bir yapı

eklenmiştir. Diğer bir çalışmada ise [41], Bulut Bilişim iletişim standartlarının yönetimi için kullanılan SLA'ların güvenli bir aracı platformdan yoksun olduklarını belirterek bunun için Akıllı Sözleşmeleri önermişlerdir. Ayrıca SLA'daki olası ihlalleri Blokzincir'e kaydetmeden önce doğru veri olduğunu kanıtlamanın oldukça zor olduğunu belirtmişlerdir. Bu zorluğun üstesinden gelmek için, oyun teorisini kullanan bir tanık modeli önermişlerdir. Tanıkların hedeflenen güvenilirliğini sağlaması için tanıklara bir teşvik olarak belirli bir ücret ödeme taahhüdünde bulunmuşlardır. Başka bir çalışmada [42] ise yazarlar Akıllı Sözleşmelerin, merkezi şekilde yönetilmesinin yaygın olduğunu ve bu nedenle erişilebilirliği ve veri bütünlüğünün manüpülasyona açık olduğunu belirtmişlerdir. Bu yüzden Akıllı Sözleşmelerin gruplanabileceği ve ortak değişkenlerin veya verilerin toplu olarak belirlenen değere göre birbirleriyle etkileşimde bulundukları merkezi olmayan bir çalışma geliştirmişlerdir. Birden fazla üye arasında fikir birliğine varmak için asenkron oylama yapılarak oy sayısı yeterli olduğunda bir fikir birliğine vardıkları bir uzlaşı mekanizması önermişlerdir. Böylelikle istemci ve Akıllı Sözleşme arasındaki veri geçişinin güvenirliğini sağlamak istemişlerdir. Benzer başka bir çalışmada [43] ise SLA'ların tazminat sürecini yönetmenin bürokrasi gerektiren zor bir süreç olduğunu belirtmişlerdir. Başlatılan bir SLA'de meydana gelen ihlali çözmek için sermaye ve emek harcandığını ve işlemlerin genel olarak manuel yapıldığını vurgulamışlardır. Bunun için güvenilir ve üçüncü bir şahıs müdahalesi gerektirmeyen yapısı nedeniyle Akıllı Sözleşmeleri SLA ihlal yönetimi için kullanmışlardır.

Blokzincir ve Akıllı Sözleşmeler, ağ dilimleme ve kanal tahsis yönetimi için birkaç çalışmada kullanılmıştır. Yapılan bir çalışmada [44] Blokzincir'in özelliklerinin, 5G ağ dilim yönetiminin üst katman seviyesinde birçok süreci doğası gereği destekleyebileceğini iddia etmişler ve Akıllı Sözleşmeler ile sistemin 5G'de sanallaştırılmış ağ işlevleri için yönetilebilir bir platform olarak nasıl çalışabileceğini açıklamışlardır. Benzer diğer çalışmada [45], geleneksel ağ paylaşım anlaşmalarına bağlı kalmadan yeni iş modellerinin yeni ihtiyaçlarını karşılamak amacıyla Blokzincir teknolojisinden yararlanan yeni bir ağ dilimleme teknolojisi NSBchain'i önermişlerdir. Önerdikleri modelde ağ kaynaklarını otomatize ve ölçeklenebilir bir

şekilde kiracılar arasında tahsis edebilmek için Akıllı Sözleşmeler kullanmışlardır. Diğer bir çalışmada [46] ise spectrum dağıtımında olası güvenlik sorunları ele alınmış ve operatörlerin spektrum paylaşımını güvenliği ve mahremiyetinin korunması için güvenilir bir aracıya ihtiyaç duyulacağını belirtmişlerdir. Bu nedenle Akıllı Sözleşmeler sayesinde açık artırma ve pazaryeri altyapısı sağlayan ve kablosuz ağlarda bağımsız şekilde sprektrum paylaşmasını sağlayan Multi-OPs Spectrum Sharing (MOSS) adını verdikleri bir framework tanıtmışlardır.

Başka çalışmalarda ise Akıllı Sözleşmeler, IoT ağ üzerinde kimlik doğrulama ve veri paylaşım yönetimi için kullanılmıştır. IoT cihazlarının yaşam döngüsünde benzersiz ve küresel bir dijital kimlik oluşturmayı ve Blokzincir ağında saklamayı önermişlerdir [47]. Diğer bir çalışmada ise Akıllı Sözleşmeler, IoT cihazları arasındaki işlemlerin koordinasyonunu sağlaması ve parasal işlemlerin otomatize olarak yönetilmesi için kullanılmıştır [48]. Benzer çalışmada [49] ise IoT sistemlerine birden fazla IoT cihazı bağlandığında erişim yönetiminin iyi derecede dağıtılması gerektiğini belirtmişler bunun için ölçeklenebilir bir yapı olması gerektiğini vurgulamışlardır. Bu mimari için Blokzincir ve Akıllı Sözleşme tabanlı bir erişim kontrolü mimarisi önermişlerdir. Aynı yaklaşım ile benzer diğer çalışmada [50] ise birçok IoT cihazının yönetiminde Akıllı Sözleşme işleme maliyetlerini azaltmak için hub tabanlı katman ve havuz katmanı da olmak üzere çok katmanlı bir yönetim mekanizması önermişlerdir. Başka bir çalışmada ise [51], Akıllı Sözleşme kullanılarak veri kiralama sistemi önermişlerdir. Bunun için gerekli olan veri bütünlüğünü ve güvenliğini artırmak için geleneksel modellere alternatif olarak Akıllı Sözleşme ve Blokzincir kullanmışlardır. Farklı bir çalışmada ise yazarlar [52], IoT sistemleri için dağıtılmış ve güvenilir erişim kontrolü sağlamak için erişim kontrol sözleşmeleri, yargıç sözleşmesi ve kayıt sözleşmesi barındıran Akıllı Sözleşmelere dayalı bir model önermişlerdir. Erişim kontrol sözleşmesi, tanımlı kurallara dayalı olarak doğrulama sağlayarak dinamik erişim izinlerini yönetir. Yargıç sözleşmesi, erişim kontrol sözleşmelerinden hatalı davranış bilgilerini (çok sayıda hatalı giriş vb.) alır ve değerlendirir gerekirse cezayı sahibine iade ederek bir davranış değerlendirme yöntemi uygular. Böylelikle erişim kontrol sözleşmelerin işleyişine katkıda bulunur. Kayıt sözleşmesi ise erişim kontrol sözleşmelerine ait

değerlendirme çıktılarını Akıllı Sözleşmelerine kaydeder. Diğer bir çalışmada [53] ise yazarlar Akıllı Sözleşme tabanlı akıllı ev sistemi modeli önermişlerdir. Bu modelde acil durumlarda IoT sensörlerden gelen veriler Akıllı Sözleşme tabanlı geliştirdikleri ev servis sağlayıcısına iletilmiştir. Ev servis sağlayıcı ile ev sahibi arasındaki iletişim için Meteor adlı bir framework kullanmışlardır. DDoS saldırısı ve diğer saldırılardan korunmak için Tek Kullanımlık Parolayı (Once Time Password) kullanmışlardır. Başka bir çalışmada [54] ise IoT sistemlerinde sensör verilerinin güvenliği ve izlenebilirliği için güvenilir üçüncü bir taraf gerektiğini belirtmişler. Bu güveni sağlaması için Akıllı Sözleşme ve DLT (Distributed Ledger Technologies) adını verdikleri bir model önermişlerdir. Sensörlerden gelen verilerin sadece sağlama toplamı (checksum) değerlerinin doğrulanabileceği bir Dağıtılmış Uygulama (Decentralized Application - DApp) geliştirmişlerdir. Akıllı Sözleşme tabanlı IoT üzerine bir başka yaklaşım [55] ise paylaşım ekonomisi üzerine olmuştur. Birçok paylaşım ekonomisi platformunun güvenilirlik sağlamak amacıyla derecelendirme sistemlerini uygulamaya koyduklarını fakat kullanıcıların güvenilir bir profil ortaya koysalar bile bireysel yönden risk taşıdıklarını belirtmişlerdir. Bu nedenle paylaşım ekonomisine güvenilir altyapı sağlamak amacıyla Akıllı Sözleşmelerin merkezi olmayan yapısı sayesinde güvenilir bir altyapı sağlayacaklarını savunmuşlardır. Paylaşım ekonomisi üzerinde Akıllı Sözleşmeler için örnek bir model belirleyerek SLA kavramını uygulamışlardır. Böylelikle SLA yaşam döngüsünü otomatize etmişler ve tüm kuralları şeffaf hale getirmişlerdir.

Önerilen bazı çalışmalar ihlallerin kurcalanmadan Akıllı Sözleşme'ye aktarılmasını garanti ediyor olsa da uygulanabilirlik açısından oldukça karışıktır. Bazı çalışmalar ise kaydedilmiş veriyi daha sonra müşteriye kiralama imkânı sunmuştur. Bu durumda ise müşterinin veriye gerçek zamanlı olarak ihtiyaç duyması durumunda isteğin karşılanması mümkün olmayacaktır. Ayrıca Blokzincir teknolojisi yüksek performanslı hesaplamalar içerdiğinden sensör düğümü gibi düşük enerjili cihazlara uygulanması oldukça zordur [1]. Bu nedenle, IoT tabanlı çalışmaların hemen hemen hepsinde RasberryPi gibi Tek Kart Bilgisayar (Single Board Computer - SBC) kullanılmıştır. Hızlı demonstrasyon yapmak için pratik olan bu küçük bilgisayarlar üzerinde araştırmacılar istedikleri gibi algoritmik yöntemlerini test edebilmişlerdir.

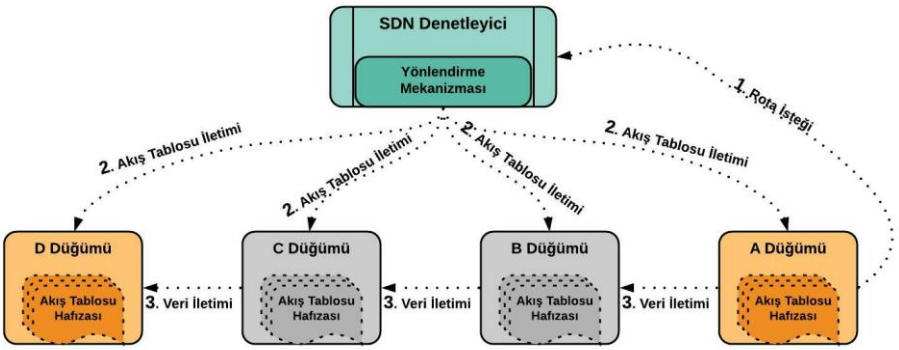
Pratikte mümkün olan bu yaklaşımlar ne yazık ki sadece kablosuz sensörlerin oluşturduğu mesh ağlar gibi gerçek dünya sistemlerinde uygulanması mümkün olmamaktadır. Bu nedenle yaptığımız çalışma, diğer çalışmalardan farklı olarak sadece kablosuz sensörlerden oluşan bir KAA'yı model almıştır. Bu çalışma, KAA üzerinde Akıllı Sözleşmelerin karmaşık olmayacak bir şekilde uygulanabilir olduğunu göstermektedir.

BÖLÜM 3. KULLANILAN TEKNOLOJİLER VE ÖNERİLEN YÖNTEMLER

3.1. Kullanılan Teknolojiler

3.1.1. YTA tabanlı KAA

Bu çalışmada, Riverbed Modeler Simülasyon yazılımı kullanılarak modellenen YTA tabanlı KAA kullanılmıştır. Bu mimarinin ayrıntıları [9], [56] ve [57]'de yer almaktadır. Bu mimaride iki tür düğüm vardır: Birincisi, tüm ağ bilgilerini içeren ve tüm ağ kontrolünden ve yönetim işlemlerinden sorumlu olan SDNC'dir. SDNC ayrıca bulanık tabanlı Dijkstra algoritmasını kullanan bir yönlendirme keşif mekanizmasına sahiptir. Kaynak ve hedef arasındaki en verimli yolu belirlerken kullandığı yönlendirme algoritması, yol üzerindeki düğümlerin kalan enerjisini ve komşu düğümlerin Sinyal Gürültü Oranı (Signal Noise Ratio - SNR) değerini dikkate alır. Ağdaki ikinci düğüm ise, Sink işlevleri için bir uygulama katmanına sahip olan Son Cihaz (End Device - ED)'dır. Herhangi bir ED, ağ ve uygulama gereksinimlerine uygun olarak bir yönlendirme cihazı olarak da davranabilir. SDNC ve ED arasındaki kontrol mesajlaşması, [56]'de kapsamlı bir şekilde açıklanan WSANFlow Protokolü tarafından sağlanır.



Şekil 3.1. YTA tabanlı KAA'nın ana bileşenleri ve veri aktarımı yaşam döngüsü

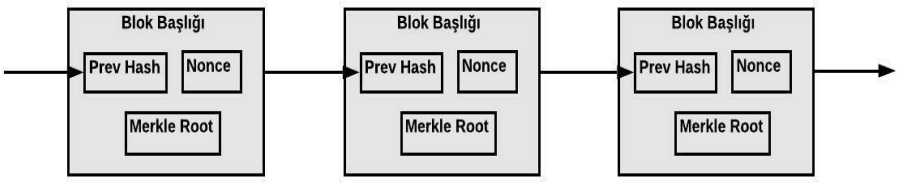
Önerilen mimaride, herhangi bir veri transferini başlatabilmek için iki işlem gerçekleştirilmelidir. Birincisi, kaynak düğümden SDNC'ye bir yol talebinin gönderilmesidir. İkincisi ise elde edilen yol bilgisinin ilgili düğümlere SDNC tarafından dağıtılmasıdır. Şekil 3.1. YTA tabanlı KAA'nın ana bileşenlerini ve önerilen mimaride bir veri aktarım işleminin sıra diyagramını göstermektedir. A düğümü, verileri D düğümüne göndermesi gerektiği bilgisini bir yol talebi olarak SDNC'ye iletir. SDNC bu talebi aldığında, bulanık tabanlı Dijkstra algoritmasını kullanarak A ve D düğümleri arasındaki en uygun yolu belirler ve bu bilgiyi bir uygulama akış kuralları olarak Akış Tablosu adı verilen objelerde saklar. Bu akış Tablosu daha sonra yoldaki düğümlere aktarılır. Akış Tablosu'nu alan düğümler, uygulamaya ait veri aktarımı işlemine katılmaya hazırdır. Ağdaki herhangi bir düğüm bir paket aldığında, ilk olarak Akış Tablosu'nu kontrol eder, herhangi bir eşleşen kural olması durumunda, istenen işlemi gerçekleştirmek için eylem alanına bakar. Örneğin, bir sonraki düğüme iletir veya paketi düşürür. Herhangi bir ED'nin Akış Tablosu'ndaki kuralları dinamik olarak değişebilir. Ağın hayatta kaldığı süre boyunca, enerji tüketimi, kesinti veya saldırı durumunda ED'ler devre dışı bırakılabilir. Bu gibi durumlarda ağ istihbaratına sahip SDNC, yeni rotaları keşfedip ilgili ED'lere teslim etmekte ve veri aktarım işleminin kesintisiz olarak devam etmesini sağlamaktadır.

3.1.2. Blokzincir

Blokzincir sistemi 2008 yılında Bitcoin kripto para sistemi mimarisi olarak Satoshi Nakamoto tarafından yayınlandı [58]. Blokzincir kriptografik kanıt tabanlı ve üçüncü bir güvenilir tarafa gerek duymadan direkt olarak iki tarafın birbiriyle işlem yapabilmelerini olanak sağlayan bir elektronik ödeme sistemidir [58]. Blokzincir'i kayıtları zaman damgalı bloklar halinde toplanan ve her bloğa ait unique bir hash kodu bulunan bir defter olarak düşünebiliriz.

3.1.2.1. Blokzincir-Blok algoritması

Kullanıcı tarafından seçilen şifreleri kullanarak şifreleme düzeni güvenlik açıklarının üstesinden gelen Blokzincir-blok fikrinin temeli olan karma (hash) zinciri, ilk olarak Lamport [59], [60] tarafından önerilmiştir. Bir veri parçasına bir şifreleme karma işlevinin art arda uygulanmasıdır. Blokzincir, blokların karma zinciri sürümüdür. Her blok oluşturulduğunda önceki bloğun karma kodu referans alınır. Böylece bir Blokzincir oluşturulur [61]. Yeni bloğu tamamlayan düğüm ise diğer tüm düğümlere yeni blok hakkında haber verir ve eşleme tamamlanır. Sistemin dağıtık yapıda olması sayesinde herhangi bir yerde veya birden fazla yapılan saldırı girişimi sistemin güvenirliğini etkilemez. Bundan dolayı Blokzincir ağında her düğümün birbirine güvenmesi gerekmez. Şekil 3.2.'de gösterildiği gibi, her blok, işlemlerin [62] Merkle kökünü içerir ve bu, işlemlerin tarihsel doğrulama mekanizmasını kolaylaştırır.



Şekil 3.2. Blokzincir Blokları

Bir blok karma kodu oluşturulduğunda sürüm, Prev Hash, Merkle Root, Time, Bits, Nonce bilgileri kullanılır ve bu değerler birleştirilir [58]. Bloklardan birini değiştirmek istiyorsak, her bloğun blok karma değerlerini baştan sona değiştirmemiz

gerekir. Ek olarak, küresel olarak dağıtılmış defter (ledger) bu değişiklikleri kabul etmeyecektir. Bu, blokların değişmez olmasını sağlar.

3.1.2.2. Akıllı Sözleşmeler

Akıllı Sözleşmeler ise Blokzincir ağı üzerinde değiştirilemez şekilde saklanan ve hali hazırda çalışan kod parçacıklarıdır [63]. Bu kod parçacıkları Blokzincir ağı güvencesi altındadırlar ve değiştirilemezler. Dolayısıyla Akıllı Sözleşmeler dışardan yapılabilecek herhangi bir manipülasyona kapalıdırlar. Akıllı Sözleşmeler veri depolayabilirler, bir karar mekanizması işletebilirler, bir hesaba sahip olup başka bir hesaba para aktarabilirler ve diğer sözleşmelerle iletişim kurabilirler [49]. Akıllı Sözleşmeler, sahibi tarafından Blokzincir üzerinde kalıcı olarak oluşturulur. Blokzincir altyapısını kullanan birden fazla kripto para platformu mevcut. Bunlardan belki de en önemlisi Ethereum diyebiliriz. Ethereum, Blokzincir tabanlı ve Akıllı Sözleşme protokolü özelliğini sağlayan açık kaynaklı bir kripto para mimarisidir. İşlem temelli durum geçişleri sayesinde Blokzincir uzlaşma (consensus) protokolünün güncellenmiş bir sürümünü destekler [64]. Ethereum'da Akıllı Sözleşmelerin yürütülmesi Ethereum Sanal Makinesi (Ethereum Virtual Machine - EVM) üzerinde gerçekleşir. EVM ise Ethereum ağında çalışan bir Turing yazılımıdır. Yeterli zaman ve verilen programlama diline bakılmaksızın herhangi bir programın platformda çalıştırılmasını sağlar. Her işlem operasyonu için belirli bir EVM işletim ücreti vardır ve 'gas' olarak adlandırılır. Bu yüzden Akıllı Sözleşme üzerinde algoritmik yöntem oluştururken yüksek işlem ücretlerinden kaçınmak amaçlı sözleşmenin en uygun işlem ve değişken içermesi sağlanmalıdır. Ayrıca Ethereum, geliştiricilerin DApp oluşturmasına ve dağıtmasına olanak sağlar. DApp'lar ise Ethereum sayesinde farklı farklı endüstrilerde aracı hizmetleri kaldırmada büyük kolaylık sağlar.

3.1.3. KAA'larda SLA kullanımı

SLA'ler servis sağlayıcı ve müşteri arasındaki hizmet niteliklerini barındıran anlaşmalardır. Bu nitelikler; servis sağlayıcının müşteriye sunduğu hizmetin kalite

standartlarını belirlemektedir. Bir SLA kapsamında sağlanan hizmetlerin niteliği, miktarı, yedekleme, destek, hizmetin teslim süresi ve sorun çözme süreleri gibi kriterler yer alabilir. Önerilen modelde KAA üzerine çalıştık. Bu nedenle kablosuz ağların ihtiyaç duyabileceği teknik kriterlerin neler olduğu incelenmiştir. KAA'da SLA sözleşmesi için kullanılabilecek bazı parametreler aşağıdaki gibidir [65].

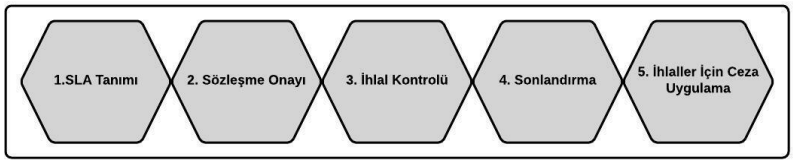
- Servis Kullanılabilirliği (Service availability)
- Sistem Kapalı Kalma Süresi (Down-Time)
- Ağ Hata Oranı (Network Failure Rate)
- Ölçüm Periyodu (Measurement Period)
- Gecikme (Latency)
- Ağdaki Düğüm Sayısı (Number of Nodes in Network)
- Enerji Tüketimi (Energy Consumed)

Önerilen modelde bu parametrelerin denetimini sağlayan bir altyapı sunulmaktadır. Müşteri istediği takdirde belirtilen parametreleri SLA Sözleşmesine dahil edebilmektedir. Ayrıca SLA ler içerisinde farklı aşamalar barındırırlar ve SLA yaşam döngüsünü oluştururlar. Şekil 3.3'te bir SLA yaşam döngüsü ve önerilen modelin hangi aşamaları kapsadığı gösterilmektedir. Önerilen model 5 aşamadan oluşmaktadır ve şu aşamaları içermektedir;

- SLA tanım aşaması
- SLA in kabul edilmesi
- İhlallerin kontrol edilmesi
- SLA sonlandırılması
- Cezaların yürürlüğe girmesi şeklindedir.

Önerilen modelde kriterleri belirleyen müşteridir. Servis sağlayıcı, müşterinin belirlediği şartları eğer kabul ediyorsa SLA Akıllı Sözleşme'sini oluşturur. Akıllı Sözleşme'nin oluşturulması servis sağlayıcının görevidir. Müşteri sözleşmeyi onaylar ve süreç başlar. Çalışma süresince belirli aralıklarla kablosuz ağ içerisindeki veri aktarım kalitesinin belirlenen eşik değerlerini aşıp aşmadığı kontrol edilir. Bu

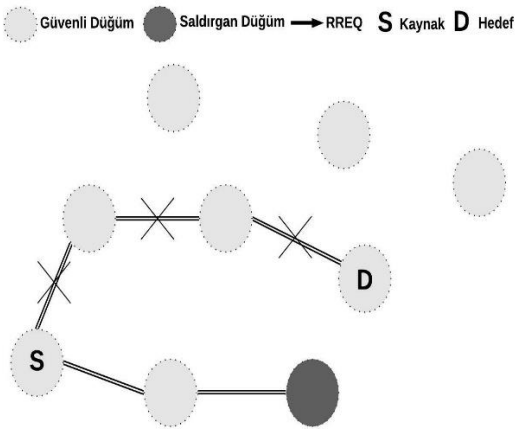
kontroller kaydedilir ve veri aktarımı tamamlandığında toplam ihlal için ceza bedeli hesaplanarak müşteriye iade edilir. Böylelikle otomatize şekilde SLA yaşam döngüsü tamamlanır.



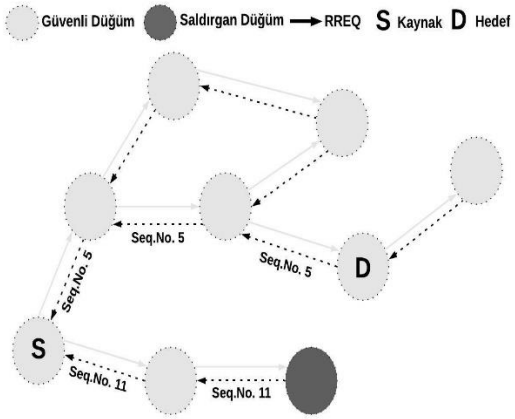
Şekil 3.3. KAA'larda SLA Yaşam Döngüsü

3.1.4. Kara Delik Saldırısı

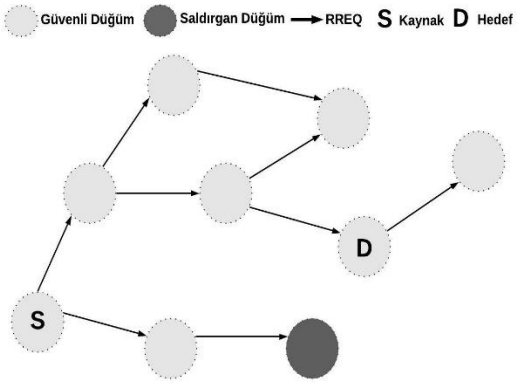
Kara Delik, AODV protokolünü kullanan yaygın bir saldırı türüdür. AODV protokolünde, rota isteğe bağlı olarak otomatik olarak oluşturulur. Bir düğüm başka bir düğüme bir veri paketi gönderdiğinde, RREQ ve Rota İstek Cevap (Request Reply – RREP) mesajları ile yenilenen yönlendirme tablosunu kullanır. Yeni rota olması durumunda düğüm, Rota Keşfi işlemini başlatır. AODV üzerindeki Rota Bulma İşlemi, RREQ ve RREP olmak üzere iki kontrol mesajına sahiptir. Bu mesajların her ikisi de yeni rotayı tanımlamak için kullanılır. Yönlendirme keşif işlemi tamamlandıktan sonra kaynak düğümün hedefe veri iletmesine izin verilir [66].



Şekil 3.4. Kaynak düğümden RREQ paketi yayınlanması



Şekil 3.5. Saldırgan düğümden ve hedef düğümden RREP paketi yayımlanması



Şekil 3.6. Kaynak düğüm ile saldırgan düğüm arasında veri iletiminin başlaması

AODV protokolünde Kara Delik saldırısı şu şekilde gerçekleştirilir: Şekil 3.4.'te gösterildiği üzere kaynak düğüm gönderilecek veriye sahip olduğunda, öncelikle yönlendirme keşif işlemini başlatır ve komşularına bir RREQ paketi yayınlar. RREQ paketini alan herhangi bir komşu, kendi adresini ekleyerek paketi hedef noktaya iletir. Saldırgan düğümü ise kaynak düğüme (en yüksek sıra numarası ve en düşük atlama sayısı ile) yanıt olarak yanlış bir RREP paketi gönderebilir ve bir hedef düğüm görevi görebilir. Kaynak düğüm birden fazla yanıt aldığında, RREP paketlerinin sıra numaralarını karşılaştırır. En yüksek sıra numarasına sahip rota seçilir. Gelen paketlerin her iki RREP'si aynı sıra numarasına sahipse minimum atlama sayısı dikkate alınır. Şekil 3.5.'te gösterildiği üzere kaynak düğüm hem hedef düğümden hem de saldırgan düğümden bir RREP paketi almaktadır. Ancak saldırgan düğüm, hedef düğümün RREP paketine göre yüksek sıra numarası (sıra no. 11)

içeren bir RREP paketi göndermiştir. Şekil 3.6.'da gösterildiği gibi, saldırgan düğümü hedefliyormuş gibi davrandığından, tüm veri paketlerini kendisine iletmiştir ve böylece doğru veri yolunu bloke etmiştir. Bu saldırının bir sonucu olarak kaynak düğüm ve gerçek hedef düğüm birbiriyle iletişim kuramamaktadır [67].

3.2. Önerilen Yöntemler

Bu tez kapsamında, Blokzincir altyapısı kullanılarak YTA tabanlı KAA mimarisi üzerinde güvenliği artırmak amacıyla iki farklı yöntem belirlenmiştir.

3.2.1. Blokzincir-blok yaklaşımı kullanılarak Kara Delik saldırısına karşı Akış Tablosu ve veri güvenliğinin sağlanması

Önerilen model, YTA tabanlı KAA'daki güvenlik gereksinimlerini karşılamak için veri güvenliği ve veri bütünlüğü aşamalarını içerir. Bu çalışmada, her iki aşamada da Blokzincir-blok yapısına benzer bir mimari uygulanmıştır. Bu yapı sayesinde üretilen tüm Akış Tabloları ve girişleri karma hale getirilerek blok haline getirilir. Hafif hesaplama gücü gerektiren bu yaklaşımla performans kaybının en aza indirilmesi hedeflenmiştir.

Blokzincir güvenliğinde genellikle iki ana şifreleme tekniği kullanılır. Bunlar, Elips Eğrisi RSA şifreleme ve SHA-256 Hash kodlama teknikleridir. Çalışmamızdaki cihazlar sınırlı işleme kapasitesine sahip olduğundan, önerilen modelde SHA-256 Hash kodlama yaklaşımının kullanılması tercih edilmiştir. Önerilen model iki ana aşamayı hedeflemektedir. Birincisi, Akış Tablosu'nun kurcalanmasını önlemektir. İkincisi, veri paketinin ağ üzerinde doğru rotada ve değişmeden iletilmesini sağlamaktır.

Akış Tablosu güvenlik aşaması iki adımdan oluşur:

- SDNC tarafından Akış Tablosu imzasının (hash değeri) elde edilmesi
- Akış Tablosu'nu alan düğüm tarafından imzanın (hash değeri) doğrulanması

Akış Tablosu bir veya daha fazla akış kuralı içerir. Bir Akış Tablosu imzası elde etmek için her akış kuralına özel bir imza değeri hesaplanır. Hesaplanan değerler ve ilgili düğüme ait özel bir gizli anahtar kullanılarak genel Akış Tablosu imzası oluşturulur. Tüm imza değerleri hesaplanırken çeşitli karma fonksiyonlardan geçirilirler. Ortaya çıkan Akış Tablosu imzası, Akış Tablosu içeriğiyle birlikte ilgili düğüme gönderilir. Ardından ilgili düğüm bu değeri doğrular.

Veri güvenliği aşaması, verilerin geçtiği tüm düğümleri kapsar. Veri güvenliği aşaması iki adımdan oluşur:

- Veriyi gönderen düğüm tarafından veri imzasının elde edilmesi
- Veriyi alan düğüm tarafından veri imzasının doğrulanması

Bu aşamada verilerin değişmezliğini korumak için paket çerçevesine yeni bir alan eklenmiştir. Bu alanda, her veriye özel bir jeton (RouteDataToken) taşınır. Bu jeton oluşturulurken verinin gideceği düğümün gizli anahtarının karma kodu kullanılır. Bu karma kod daha önce SDNC tarafından akış kuralı yoluyla ilgili düğüme gönderilir. Bu karma kod, akış kuralındaki bir alanda (PNToken) saklanır. RouteDataToken, bu karma kodu ve veri içeriğini çeşitli fonksiyonlardan geçirerek elde edilir. Elde edilen bu değer, verileri alan komşu düğüm tarafından doğrulanır. Bu prosedürlerin ayrıntıları aşağıdaki alt bölümlerde verilmiştir.

Tablo 3.1. Önerilen güvenlik modeli için notasyon bilgileri

Sembol	Notasyon
$h()$	Karma fonksiyon
N_i	i numaralı düğümün kimlik numarası
K_i	i numaralı düğümün özel anahtarı
c_f	Akış Tablosu imzası
c_{fs}	Düğümlerde ve SDNC'de saklanan doğrulama imzası
c_e	Akış kuralı imzası
c_{et}	Geçici akış kuralı imzası

Tablo 3.1. (Devamı)

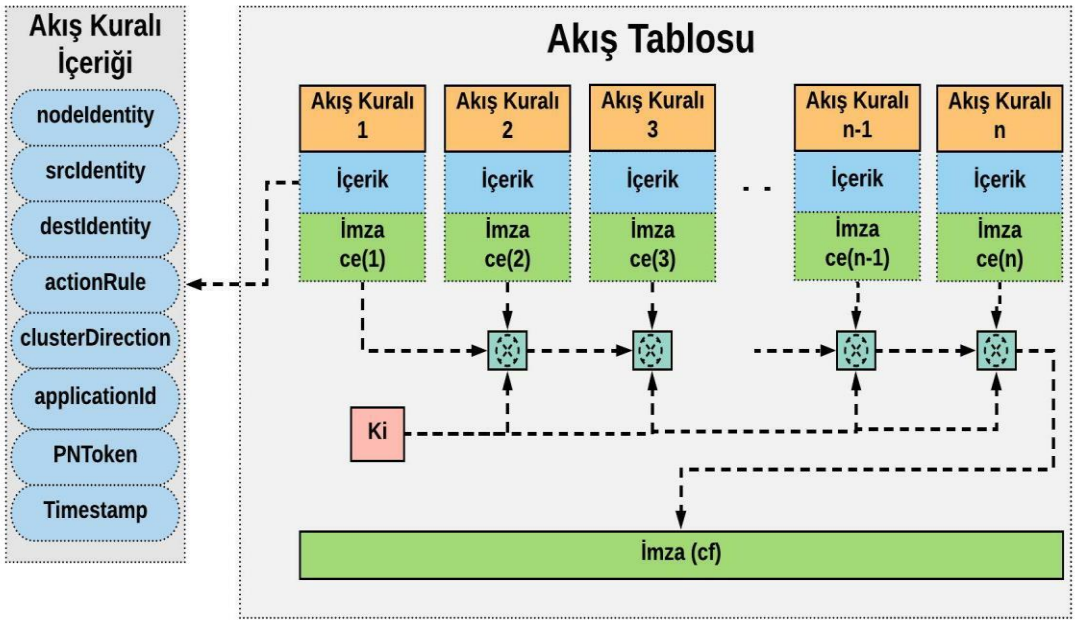
Sembol	Notasyon
c_r	RouteDataToken
c_d	Veri imzası
c_{ds}	Düğümlerde saklanan veri imzası
$\oplus$	XOR işlemi
$\|\|$	Birleştirme (Concatenation) işlemi

Tablo 3.1.'de önerilen modelin parametrelerini göstermektedir. KAA'da düğümlerin $(N_1, N_2, N_3 \ldots N_n)$ olduğu varsayıldığında, ağdaki tüm düğümler SDNC tarafından kaydedilir ve bu düğümlerin kimlik numarası ve gizli anahtarları bilinmektedir.

3.2.1.1. Akış Tablosu güvenliği

Buradaki ana amaç, doğru Akış Tablosu bilgisinin doğru zamanda doğru düğüme gittiğini Akış Tablosu imzasını kullanarak doğrulamaktır. Bu aşama iki bölümden oluşur. Birincisi, Akış Tablosu'nun oluşturulması ve doğrulanmasıdır. Şekil 3.7. bir Akış Tablosu imzasının nasıl üretildiğini gösterir. Her akış kuralının içeriğinde farklı alanlar saklanır. Bu alanlar;

- NodeIdentity: Hangi düğüm akış kuralının depolanması gerektiğini belirten alandır.
- SrcIdentity: Paketin hangi düğümden gelmesi gerektiğini belirten alandır.
- DestIdentity: Paketin hangi düğüme gitmesi gerektiğini belirten alandır.
- ActionRule: Düğüme gelen paketle ne yapılacağı hakkında bilgi içerir. (Drop, Forward, vb.)
- ClusterDirection: Ağda birden fazla küme varsa, paketin hangi kümeden geçmesi gerektiğini belirtir.
- Uygulama Kimliği: Farklı uygulama verileri aktarılabilir. Bu alan, her uygulamaya özgü benzersiz bir kimlik içerir.
- PNToken: Paketin geldiği düğümün gizli anahtarının karma değeri.
- Zaman Damgası: Akış kuralının oluşturulma zamanı.



Şekil 3.7. N_i düğümü için Akış Tablosu imza üretimi

Örnek olarak, N_i kaynak düğümü, N_j ise hedef düğüm olarak kabul edelim. Akış Tablosu oluşturma işlemi aşağıdaki gibidir: N_i, N_j 'ye veri aktarımı için SDNC'den yol bilgisini talep eder. SDNC, Denklem 3.1 ve 3.2 kullanarak N_i 'den N_j 'ye en kısa yolu belirler. Ardından SDNC, Akış Tablosu'nu ilgili düğümlere gönderir (N_i 'den N_j 'ye yoldaki tüm düğümler). Denklemde 3.1 yalnızca bir akış kuralı için imza değerinin nasıl hesaplanacağını gösterir. Denklem 3.2'de, Akış Tablosu imzasının akış kuralı imzaları kullanılarak nasıl hesaplandığını gösterir.

$$h\left(c_{e_1} \oplus h(K_i)\right) = c_{e_1}^i \tag{3.1}$$

$$h\left(c_{e_2} \oplus h(K_i)\right) = c_{e_2}^i$$

… … … …

$$h\left(c_{e_{n-1}} \oplus h(K_i)\right) = c_{e_{n-1}}^i$$

$$h\left(c_{e_n} \oplus h(K_i)\right) = c_{e_n}^i$$

$$h\left(c_{e_1}^i \oplus c_{e_2}^i\right) = c_{et} \tag{3.2}$$

$$h\left(c_{et} \oplus c_{e_3}^i\right) = c_{et}$$

… … … …

$$h\left(c_{et} \oplus c_{e_{n-1}}^{i}\right) = c_{et}$$

$$h\left(c_{et} \oplus c_{e_n}^{i}\right) = c_{et}$$

$$c_{fi} = \begin{cases} h(c_{et} \,||\, h(K_i)) & c_{fs}\ is\ null \\ h\left(c_{fs} \oplus h\left(c_{et} \,||\, h(K_i)\right)\right) & c_{fs}\ is\ not\ null \end{cases}$$

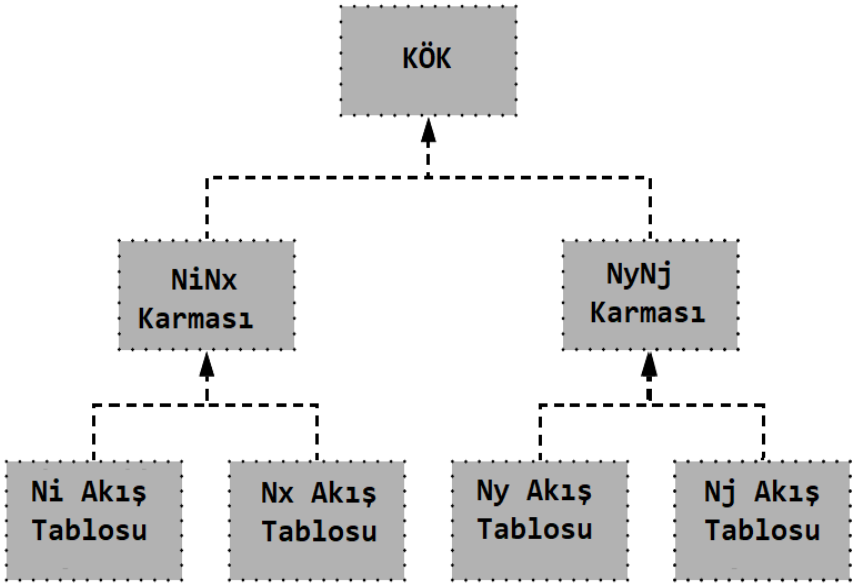
$$c_{fs} = c_{et}$$

SDNC, yoldaki her düğüm için her akış kuralındaki imzayı (c_e) ayrı olarak hesaplar. Denklem 3.1'de, N_i için her akış kuralının her imzası hesaplanır. c_e, N_i'nin gizli anahtarıyla işlenir (XOR işlemi). Denklem 3.2'de, her akış kuralı imzası, diğer akış kuralı imzasıyla sıralı olarak işlenir ve N_i'ye gönderilmesi için genel imza oluşturulur (c_{fi}). Son olarak Akış Tablosu imzası, N_i'den N_j'ye kadar tüm düğümler için ayrı olarak oluşturulur. Burada değinilmesi gereken önemli bir husus bulunmaktadır. N_i için son imza değeri olan c_{fi}, bu değer ilk kez oluşturulursa nihai değer olacaktır. Bu değer ilk kez oluşturulmamışsa (c_{fs} değeri null değilse) c_{fi} ile yeniden hash edilir ve elde edilen değer c_{fi} değeri ile eşitlenir. Her durumda c_{et} değeri c_{fs} değerine eşitlenir ve bellekte saklanır. Buradaki temel amaç, Akış Tablosu imzasının önceki değerini her düğüm için bellekte saklayarak tekrarlama saldırısı gibi saldırıları önlemektir. Buradaki bir diğer önemli nokta ise Akış Tablosu'nun içindeki PNToken alanıdır. Bu alan yoldaki düğüm noktalarına güvenli bir iletişim sağlamayı amaçlamaktadır. Veri göndermesi beklenen komşu düğümün gizli anahtarının karma değerine eşittir. Böylelikle hangi düğümün kendisine veri göndereceğinden emin olur ve taklit düğümden veri aktarımı engellenir. PNToken hesaplaması Denklem 3.3'te gösterilmiştir. PNToken, verileri gönderen düğümün gizli anahtarının karma koduna eşittir.

$$PNToken = h(K_{source\ node}) \tag{3.3}$$

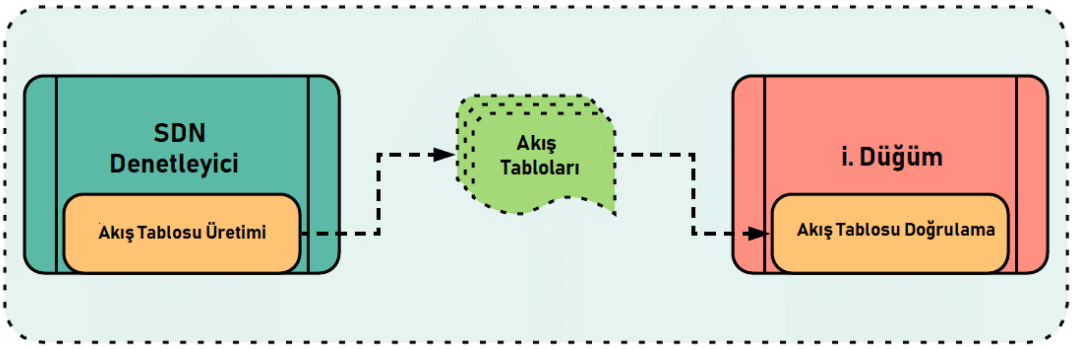
PNToken değeri hesaplandıktan sonra akış kuralına eklenir. Bu akış kurallarını içeren Akış Tablosu daha sonra güvenli bir bağlantı kullanılarak ilgili düğüme gönderilir. Bu çalışmada, ağ üzerinde meydana gelen rota değişikliklerinin değiştirilememesi için Akış Tablosu imzasının hash değeri Blokzincir ağında

tutulacaktır. Bu yaklaşım sayesinde SDNC tarafından oluşturulan Akış Tabloları Blokzincir yaklaşımı kullanılarak bir blok yapı haline getirilmiştir. Rotadaki her düğüm için oluşturulan Akış Tablosu imzaları, Merkle Ağacı algoritmasına göre bir bloğa dönüştürülür. Şekil 3.8.'de rota için oluşturulan Akış Tablosu imzalarının bloklara nasıl dönüştürüldüğünü göstermektedir. Bu bloklar SDNC tarafından saklanacaktır ve gerektiğinde internet üzerinden başka bir Blokzincir ağına yüklenebilir.



Şekil 3.8. Akış Tabloları Merkle Ağacı

Şekil 3.9.'da gösterildiği üzere her düğüm, kendisine gelen Akış Tablosu'nu bir saldırgan düğüm tarafından değil SDNC tarafından teslim edildiğini doğrulamalıdır. Bunun için düğüm Denklem 3.1 ve 3.2'yi kullanarak c_{fi} değerini elde eder. Daha sonra c_{fi} ile alınan Akış Tablosu imzasını karşılaştırır. Bu hesaplama ilk kez yapılıyorsa, c_{fi} doğrudan karşılaştırılır, aksi takdirde c_{fi} önceden hesaplanan ve bellekte saklanan c_{fs} değeri ile yeniden karma fonksiyonuna tabi tutulur. Daha sonra elde edilen sonuç, alınan imza değeri ile karşılaştırılır.

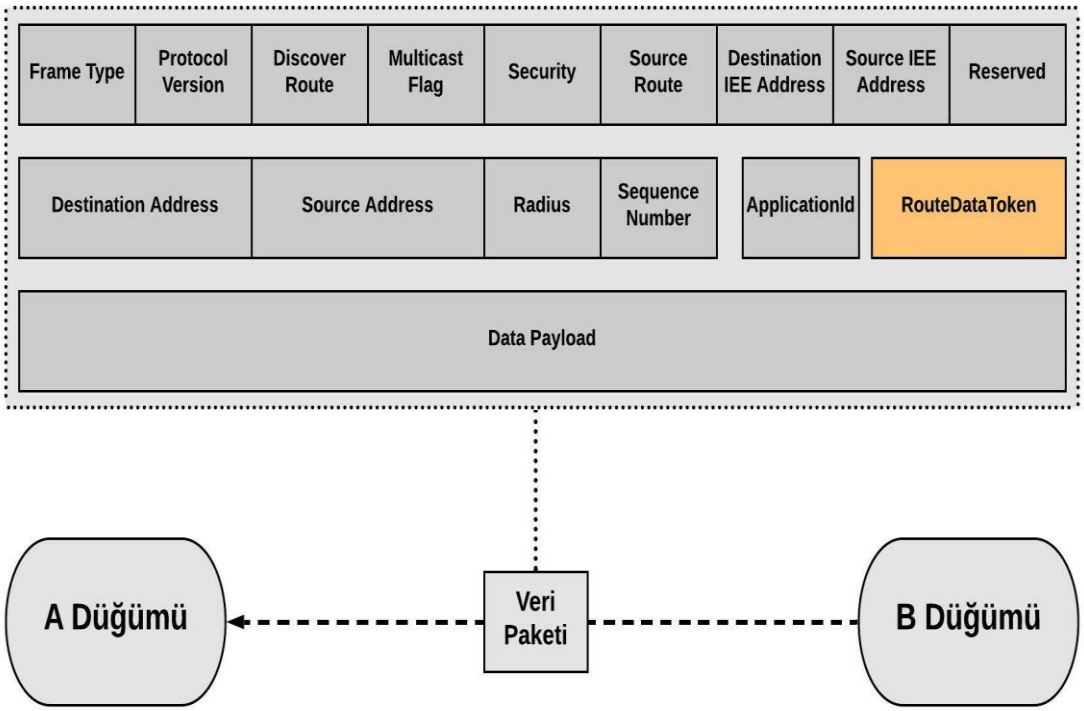


Şekil 3.9. YTA tabanlı KAA'da Akış Tablosu akışı

Her düğüm için üretilen c_{fi} değerleri farklıdır. Her düğümün komşularının sayısı az olduğundan, karma değer üretme işlemlerinin sayısının daha düşük olması beklenir. Bu nedenle düğüm performansı etkilenmeyecek ve kayda değer ek işletim gücü olmayacaktır. Başlık 3.2.1.1.'de Akış Tablosu güvenlik modeli detaylı açıklanmıştır. Tez çalışması kapsamında önerilen algoritma ile her Akış Tablosu'nun değişmez bir imzaya sahip olduğu görülmektedir. Böylelikle ağ içerisinde dolaşan verilerin doğru yolda olması sağlanmıştır.

3.2.1.2. Veri güvenliği

Taklitçi saldırganlar kendilerini güvenli bir düğüm olarak gösterebilir ve kendilerini bir veri akışına dahil edebilirler. Veri akışını güvence altına alabilmek için Şekil 3.10.'da da gösterildiği üzere önerilen güvenlik modeli tarafından IEEE 802.15 paketine 'RouteDataToken' alanı eklenmiştir. Bu alan, veri yolundaki her düğümden gelen verilerin doğrulanmasına izin verir. Yoldaki her düğüm, bir imza değerini ve sağlama toplamını kendisine gelen verilerle karşılaştırır. Bu karşılaştırmanın temel amacı, yol üzerinde taşınan verilerin manipüle edilip edilmediğini belirlemektir. Bu alanın nasıl doğrulandığı Şekil 3.11.'de bir algoritma olarak gösterilmektedir.



Şekil 3.10. Önerilen model için veri paketi yapısı

Örnek olarak; N_i 'nin kaynak düğüm ve N_j'nin hedef düğüm olduğunu düşünelim. N_i, N_i'ye veri aktarımı için SDNC'den yol bilgisi talep eder. SDNC, Akış Tablosu'nu oluşturur ve ilgili düğümlere gönderir. Örnek yol $N_i \rightarrow N_x \rightarrow N_y \rightarrow N_j$ olarak belirlenmiştir.

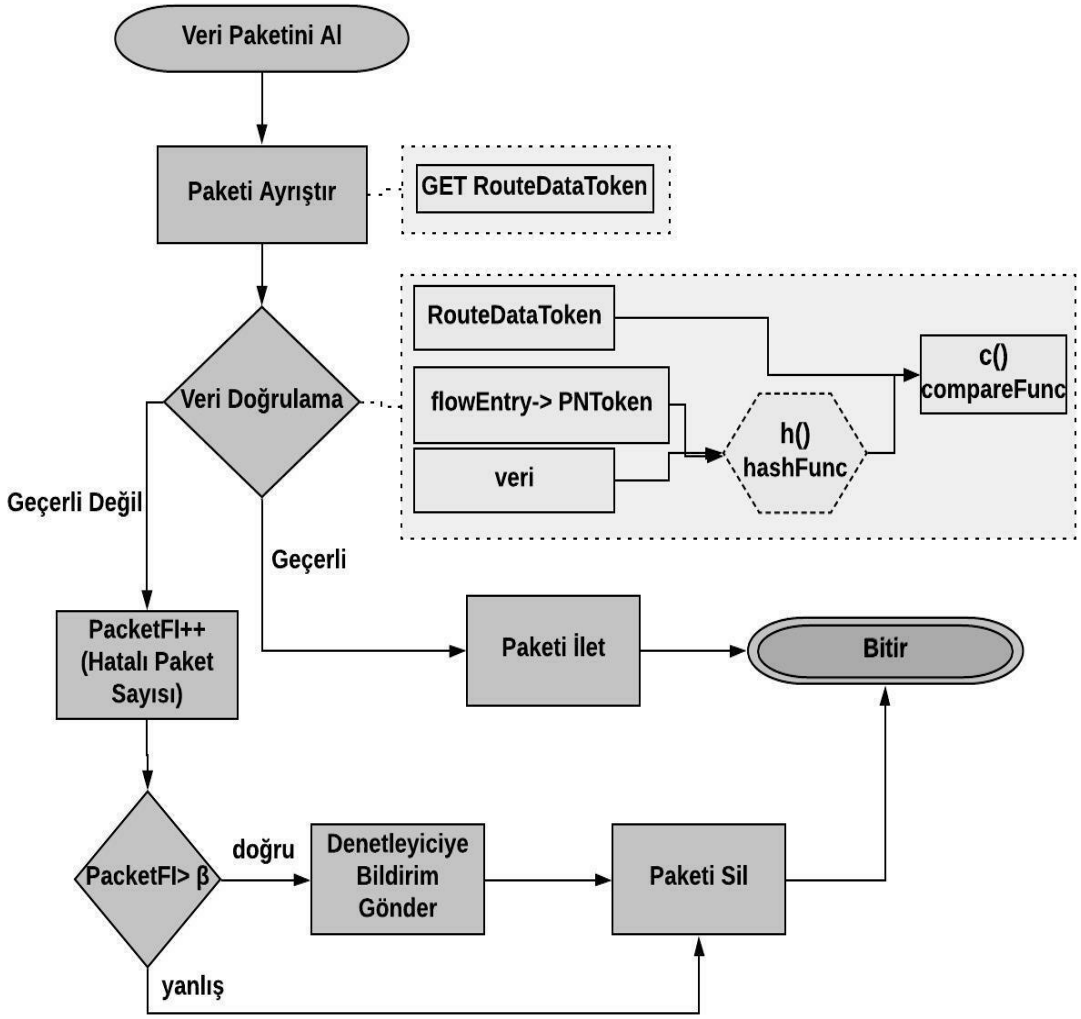
$$h(data_i) = c_d^i \tag{3.4}$$

$$c_d^i = \begin{cases} h\left(c_d^i \oplus h(K_i)\right) & c_{ds}^i \; is \; null \\ h\left(c_{ds}^i \oplus h\left(c_d^i \oplus h(K_i)\right)\right) & c_{ds}^i \; is \; not \; null \end{cases}$$

$$RouteDataToken = c_{ds}^i = c_d^i$$

Şekil 3.11. verinin düğüme geldikten sonra nasıl doğrulandığını göstermektedir. N_x, N_i'den veriyi aldıktan sonra veri çerçevesinin içeriğini çıkarır. Doğrulama için RouteDataToken (c_r^i) ve $h(K_i)$ parametrelerini kullanır. $h(K_i)$, N_x belleğinde depolanan akış kuralındaki PNToken'e eşittir. N_x bu parametrelerle Denklem 3.4'ü kullanır ve c_d^i değerini elde eder. Elde edilen sonucu (c_r^i) ile karşılaştırır. Sonuç

doğruysa, paketteki verilerin doğru olduğunu, doğru düğümden ve doğru yoldan geldiğini teyit eder. Sonuç yanlışsa paket düşürülür.

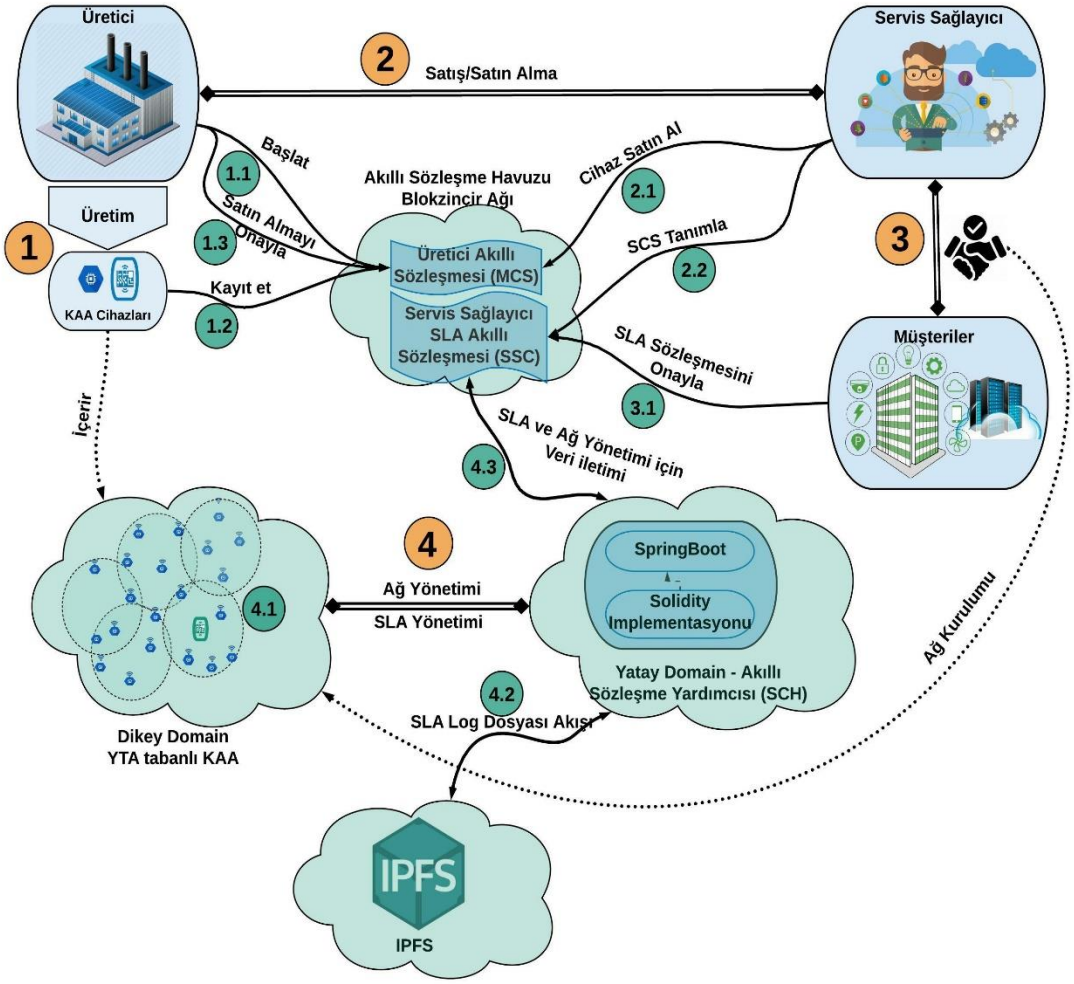


Şekil 3.11. Düğüm üzerinde veri doğrulama algoritması

N_x, paketleri sonraki düğüme (N_y) iletmek için aynı işlemleri tekrarlar. İlk hesaplamadan farklı olarak c_d^x sonraki hesaplamalarla bellekte saklanan c_d^x ile yeniden karma fonksiyonuna tabi tutulur. Daha sonra nihai sonuç c_d^x olarak belleğe yazılır. Bu durum N_x için gelen ilk paket olmadığı anlamına gelimektedir. Bu değerin (c_d^x) amacı, aynı paketi tekrar tekrar gönderen saldırganların başarısız olmasını sağlamaktır.

3.2.2. Akıllı Sözleşmeler kullanılarak SLA yönetiminin güvenli hale getirilmesi

Önerilen modelde üretici, servis sağlayıcı ve müşteri olmak üzere üç taraf yer almaktadır. Bu model Kablosuz Algılayıcı Ağ'ını oluşturacak tüm düğümlerin üretiminden satışına kadar olan tüm süreci kapsamaktadır. Üretici özel anahtarlar barındıran cihazlar üretir ve her üretilen cihaz üreticinin belirlediği bir Akıllı Sözleşme'ye aktarılır ve bu anahtarlar Blokzincir ağında saklanır. Servis sağlayıcı bu cihazları satın alır ve bu cihazlar ile müşteriye özel kurulumlar gerçekleştirir. Servis sağlayıcı her müşteriye özel SLA şartları belirler ve bu şartlar iki taraf tarafından kabul edildiğinde Akıllı Sözleşme'ye uyarlanarak Blokzincir ağına yüklenir. Akıllı Sözleşme başlatıldıktan sonra ağ içerisinde veriler aktarılırken taahhüt edilen ağ metriklerine uyulup uyulmadığına bakılır ve ihlal durumu söz konusu ise Akıllı Sözleşme servisine bildirilir. Akıllı Sözleşme servisi ise isteği alarak Akıllı Sözleşme'nin anlayacağı bir hale çevirir ve Akıllı Sözleşme'deki ilgili metoda iletir. Akıllı Sözleşme ihlal olup olmadığını ve gelen verilerin doğru veri olup olmadığını üreticinin daha önce bildirdiği özel anahtarlar üzerinden teyit eder. Önerilen modelde süreç 4 aşamaya ayrılmıştır ve aşamalar Şekil 3.12. üzerinde gösterilmiştir.



Şekil 3.12. YTA tabanlı KAA'da önerilen SLA yönetim modeli

3.2.2.1. Aşama 1 - üretici aşaması

Bu aşama önerilen model için önemli bir yapıtaşıdır. Üretici, cihazların güvenilir olmasını sağlamalıdır. Bunun için önerilen modelde, üretici her cihaza özel bir gizli anahtar üretmelidir. Ürettiği anahtarlar, üretilen cihazların içerisinde kurcalanmaya karşı güvenli bir bellek bölümünde tutulmalıdır. Üretici her cihaza özel ürettiği bu anahtarları Blokzincir ağında saklamalıdır (Şekil 3.12. Aşama 1.2). Üretici bunu gerçekleştirmek için kendisine özel daha önce oluşturulmuş ve yüklenmiş bir Akıllı Sözleşme olan Üretici Akıllı Sözleşmesi (Manufacturer Smart Contract - MSC) sayesinde gerçekleştirir. Bu Akıllı Sözleşmeler her üreticiye özel oluşturulur ve sadece üretici tarafından başlatılır (Şekil 3.12. Aşama 1.1). Üretici, cihazın üretim tarihini, genel anahtarı (cihaz seri numarası) ve cihaz için özel olarak üretilen özel

anahtarı MSC'de saklar. Böylelikle cihazı satın alan veya kiralayan herkes MSC üzerinden cihazın orijinal olup olmadığını kolaylıkla teyit edebilecektir. Üretilen cihazlar MSC'ye kaydedildikten sonra satışa hazır hale gelir.

3.2.2.2. Aşama 2 - servis sağlayıcı aşaması

Servis sağlayıcı üretilen cihazları (KAA düğümleri) anlaşmalı olduğu üreticiden satın alır. Satın alma işlemi ise sadece üretici tarafından daha önce oluşturulan MSC üzerinden gerçekleşir. Servis sağlayıcı almak istediği cihaz marka ve sayısını Akıllı Sözleşme üzerinden belirtir. Üretici servis sağlayıcının talebini onaylar (Şekil 3.12. Aşama 1.3) ve servis sağlayıcının satın aldığı cihazlara ait gizli anahtarları MSC üzerinde servis sağlayıcıya atanmış özel bir listeye alır (Şekil 3.12. Aşama 2.1). Buradaki asıl amaç, üreticinin sattığı cihazları kullanacak müşteriyi ve servis sağlayıcıyı güvence altına almaktır.

Önerilen modelde müşterinin talep ettiği SLA şartları servis sağlayıcı tarafından Akıllı Sözleşme'ye uyarlanmış ve tanımlanmıştır (Şekil 3.12. Aşama 2.2). Bu şartlar çerçevesinde Servis Sağlayıcı SLA Akıllı Sözleşmesi (Service Provider SLA Smart Contract - SSC) adı verilen bir Akıllı Sözleşme oluşturur. Servis sağlayıcı SSC'yi oluştururken üreticinin belirlediği bir Akıllı Sözleşme arayüzünü (interface) ezerek (override) oluşturmalıdır. Yani bir SSC oluşturmak için muhakkak üreticinin belirlediği bir şablon üzerinden geliştirme yapılmalıdır. Yazılım dünyasında nesneye dayalı programlama da bu yaklaşım arayüz (interface) ve sanal sınıf (abstract class) olarak karşımıza çıkmaktadır. Bu yaklaşım sayesinde geliştirici belirlenen bazı yapısal yaklaşımların dışına çıkamamaktadır. Bu sayede önerilen modelde, geliştirilen bir SSC dayatılan arayüz sayesinde aşağıdaki şartlara sahip olmalıdır.

- MSC ye bağlanarak veri doğrulama yapacak bir metoda sahip olacaktır.
- 6 parametre almalı
- byte tipinde 1 parametre dönmeli

Bu şartlara göre oluşturduğumuz Solidity kodu Şekil 3.13. üzerinde gösterilmektedir.

```
pragma solidity ^0.5.1;
contract MSC_SCCI {
  function verifyControlCode (uint globalKey, bytes32 contentHash, bytes32
  pckCreated, bytes32 pckArrived, bytes32 controlCode) public returns (bytes1);
}
contract SSC is MSC_SCCI {
   function verifyControlCode (uint globalKey, bytes32 contentHash, bytes32
   pckCreated, bytes32 pckArrived, bytes32 controlCode) public returns (bytes1) {
       MSC msc = new MSC(mscAddress);
        ……...
    }
……
}
```

Şekil 3.13. Solidity kodu

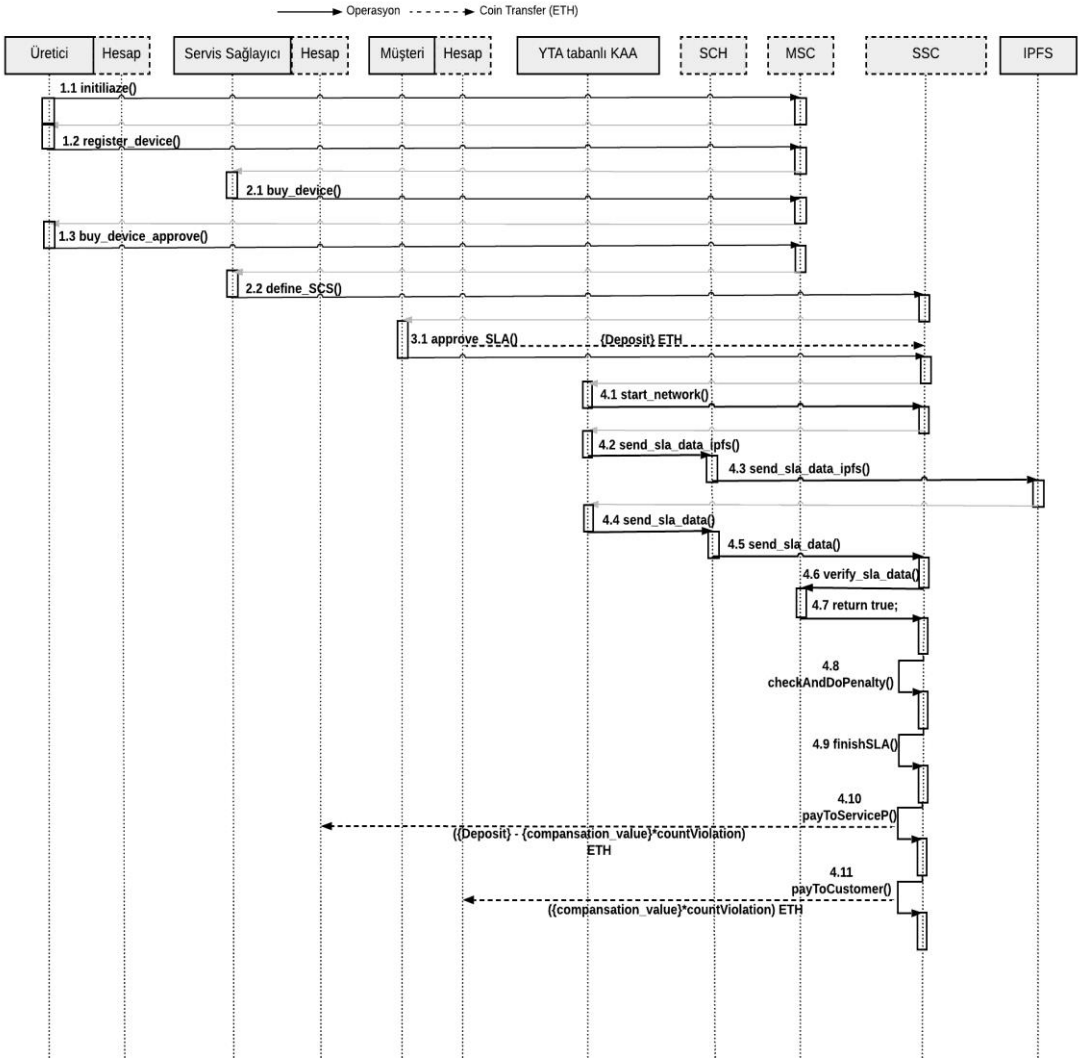
Belirttiğimiz bu şartlar, önerilen modele özel bir standart olarak kabul görmüştür. Bu standartlar mukabilinde SLA şartları ile birlikte Akıllı Sözleşme oluşturulur ve yüklenir. Ayrıca önerilen modelde servis sağlayıcı kablosuz ağ üzerinden gelen verilerin Akıllı Sözleşme'ye ulaşması için Akıllı Sözleşme Yardımcısı (SCH) diye isimlendirdiğimiz bir ara servis katmanını oluşturmalı ve yönetmelidir. Önerilen modelde, kablosuz ağ ve SCH altyapı kurulumunu servis sağlayıcı üstlenmiştir.

3.2.2.3. Aşama 3 - müşteri aşaması

Müşteri hizmet talep eden taraftır. Bu yüzden genellikle kendi ihtiyacına en iyi cevap verebilecek servis sağlayıcı ile anlaşır. Müşteri, servis sağlayıcı ile SLA şartlarında hemfikir olduktan sonra servis sağlayıcının oluşturduğu SSC yi onaylar (Şekil 3.12. Aşama 3.1) ve SLA şartlarında daha önce belirlenen depozito miktarı servis sağlayıcının hesabına aktarılır. Daha sonra servis sağlayıcı kurulumu gerçekleştirir.

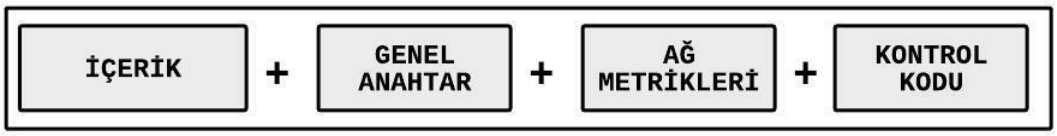
3.2.2.4. Aşama 4 - production aşaması

Production sürecinin detayları Şekil 3.14. akış diyagramı üzerinde de ayrıca gösterilmiştir. Bu aşamada KAA üzerinden gelen veriler SCH'ye TCP portları üzerinden aktarılmaktadır. Kablosuz ağ dışına veri iletimi ise ED üzerinden aktarılmaktadır.



Şekil 3.14. Önerilen SLA yönetim modeli sequence diyagramı

Kablosuz ağ üzerinden veri iletimi başlaması için SDNC veri iletimi için gerekli olan yolu belirler ve ağı başlatır (Şekil 3.12. Aşama 4.1). Bu yol belirlenmeden veri iletimi başlamamaktadır. Veri ED'ye geldikten sonra ED, SCH'ye veriyi iletmek için veri ile birlikte bazı parametreleri de iletmek zorundadır (Şekil 3.12. Aşama 4.3). Bu parametreler; genel anahtar, ağ metrikleri ve ED'nin korumalı belleğinde saklı özel anahtar ile üretilmiş bir kontrol kodudur. Sonuç olarak SCH'ye gönderilecek veri çerçevesi Şekil 3.15.'de gösterilmektedir.



Şekil 3.15. Önerilen SLA yönetimi için geliştirilen veri çerçevesi

Kontrol kodu, ED'nin korumalı hafızasında depolanan özel anahtar ile verilerin ve ağ ölçümlerinin belirli işlemlere tabi tutulmasıyla üretilir. Önerilen modelde, veri iletiminde güvenilirliği sağlamak için kontrol kodu üretimi yalnızca düğümde ve MSC'de gerçekleşir. Ek olarak, kontrol kodunun nasıl üretildiğini yalnızca üretici belirler ve bunu kimseyle paylaşmaz. Bu konuda üreticiler kod üretim metodolojisi bakımından esnek bırakılmışlardır. Denklem 3.5'de görüldüğü gibi kontrol kodu üretimi için aşağıda notasyonları gösterilen parametreler birbirleriyle ve sırayla XOR fonksiyonuna tabi tutulur. Elde edilen sonuç ise keccak256 fonksiyonundan geçirilerek kontrol kodu elde edilir.

Tablo 3.2. Önerilen SLA yönetimi için notasyon bilgileri

Sembol	Notasyon
kh	Keccak256 fonksiyonu
h_d	Verinin karma değeri
h_{dc}	Verinin oluşturulma zamanının karma değeri
h_{da}	Verinin iletilme zamanının karma değeri
h_{const}	Sabit değer karma değeri
h_s	Düğümün özel anahtarının karma değeri
c	Kontrol kodu

$$kh(h_s \oplus h_d \oplus h_{dc} \oplus h_{da} \oplus h_{const}) = c \tag{3.5}$$

SCH daha sonra gelen her ağ ölçümlerini IPFS'ye kaydeder (Şekil 3.12. Aşama 4.2). SCH ayrıca ihlal kontrolü için gelen verileri SSC'ye iletir. SSC, gelen her veriye ait genel anahtarı, ağ ölçümlerini ve kontrol kodu parametrelerini MSC'ye gönderir. Bunun nedeni SSC'nin cihazların özel anahtarlarını bilmemesidir. Başlık 3.2.2.1.'de

daha önce üreticinin tüm özel anahtarlarının üretim aşamasında, genel anahtarlarına referansta bulunularak MSC'ye kaydettiği belirtilmiştir. Sonuç olarak SSC kesinlikle MSC üzerinden veri doğrulaması yapmalıdır. MSC kontrol kodu doğrulama mekanizması Şekil 3.16.'da gösterilmektedir. MSC kontrol kodu doğrulama işlevi ücretsiz bir işlevdir. Yani 'gas' tüketimine gerek yoktur. Bu konuda önerilen modelde, servis sağlayıcı veya müşterinin bu işlem için herhangi bir ücret ödemesi gerekmemektedir.

```
Input:   uint globalKey, bytes32 contentHash,  bytes32 pckCreated,  bytes32
         pckArrived,  bytes32 controlCode
Output:  bytes1
1:       mapping(uint => WSNKey) globalKeys;
2:       foreach i in GlobalKeys
3:         if(GlobalKeys[i].globalKey == globalKey)
             if( keccak256(abi.encodePacked(GlobalKeys[i].secretKey ^
4:           contentHash ^   pckCreated ^ pckArrived ^ globalKeys[i].const))
             == controlCode)
5:               return bytes1(0x01);
6:           end if
7:         endif
8:       end foreach
9:       return bytes1(0x00);
```

Şekil 3.16. Kontrol kodu doğrulama algoritması (MSC)

Kontrol kodu doğrulama mekanizmasında, genel anahtar parametresi ilk olarak MSC'de bulunan bir liste içerisinde aranır ve eşleşme yapıldığında eşleşen liste öğesinin özel anahtarı elde edilir. Bu özel anahtar ve gelen parametreler keccak256 fonksiyonuna tabi tutulur ve elde edilen sonuç ile kontrol kodu karşılaştırılır. Sonuç aynıysa, SSC'ye olumlu cevabı döndürür. Sonuç aynı değilse olumsuz cevabı döndürür. SSC, yanıta bağlı olarak doğrulama veya hata sayacını da birer birer artırır. Bu nedenle, IPFS kayıt aşamasında herhangi bir kötü niyetli davranış olsa bile, SSC üzerinden yapılan toplam veri doğrulama sayısı, aylık raporlama sonunda IPFS dosya sistemi ile karşılaştırılır. Bir fark varsa, SLA'nın iptali gerçekleşir.

SCC veri kontrol mekanizması, Şekil 3.17. üzerinde gösterilmiştir. Algoritma şu şekilde çalışır; MSC veri doğrulama işlevi, gelen parametrelerin kullanılmasıyla çağrılır. Dönüş değeri olumlu ise devam eder ve slaId parametresi sayesinde mevcut

SLA nesnesi SLA Map listesinden bulunur. Karşılık gelen SLA nesnesi için önceden tanımlanmış kabul edilebilir gecikme süresi elde edilir. Bu süre ile, gelen veri oluşturma ve veri varış süreleri arasındaki fark karşılaştırılır. Büyükse, 'calculatePenalties()' olarak adlandırılan Hesaplama Cezaları fonksiyonunda işletilir. Mevcut tazminat için ihlal süresi kabul edilebilir süreyi aşmış ise ceza işlemleri gerçekleştirilir ve SSC bakiyesinden müşteriye ceza ücreti aktarılır. SLA ömrü sona erdiğinde, ceza işlemlerinden sonra SSC bakiyesinde kalan tutar hizmet sağlayıcının hesabına gönderilir. Ceza işlemlerinin toplam tutarı da müşteri hesabına gönderilmiş olur.

```
Input:  uint slaId, uint globalKey, bytes32 contentHash, uint pckCreated,  uint
        pckArrived,  bytes32 controlCode
Output: bytes1
 1:  WSN private wsn;
 2:  Address mscContractAddress;
 3:  mapping(uint => SLA) slaMaps;
 4:  MSC msc = new MSC(mscContractAddress)
 5:  if(msc.verifyControlCode(globalKey, contentHash, bytes32(pckCreated),
     bytes32( pckArrived), controlCode ==  bytes1(0x01))
 6:      if(wsn.slaMaps[slaId].compensation.currentDataLatency > (pckArrived -
         pckCreated))
 7:          calculateAndDoPenalties(slaId, (pckArrived - pckCreated))
 8:          return bytes1(0x10)
 9:      else
10:          wsn.slaMaps[slaId].compensation.invalidCodeCount++
11:          return bytes1(0x11)
12:      end if
13:  else
14:      return bytes1(0x00)
15:  end if
```

Şekil 3.17. SLA verisi doğrulama algoritması (SSC)

BÖLÜM 4. DENEYSEL SONUÇLAR VE TARTIŞMA

4.1. Giriş

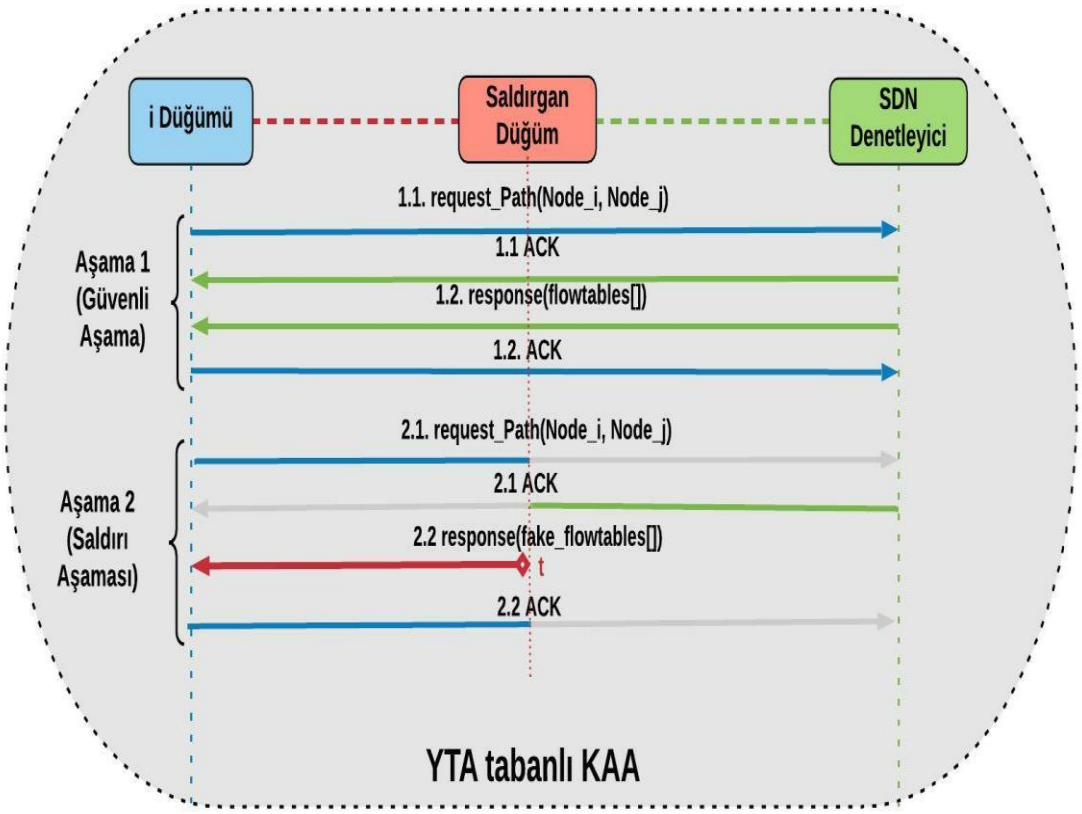
Önerilen modellerinin güvenliğini sınamak, uygulanabilirliğini göstermek ve tartışılabilir sonuçlar elde etmek için aşağıdaki senaryolar uygulanmıştır. Akış Tablosu ve veri güvenliğini sağlayan önerilen model için bir Kara Delik saldırı senaryosunun benzetimi yapılmıştır. Önerilen güvenli SLA yönetimi için ise örnek girdi parametreler üzerinden örnek bir senaryo belirlenmiştir.

4.2. Kara Delik Saldırı Senaryosu

4.2.1. Saldırı modeli

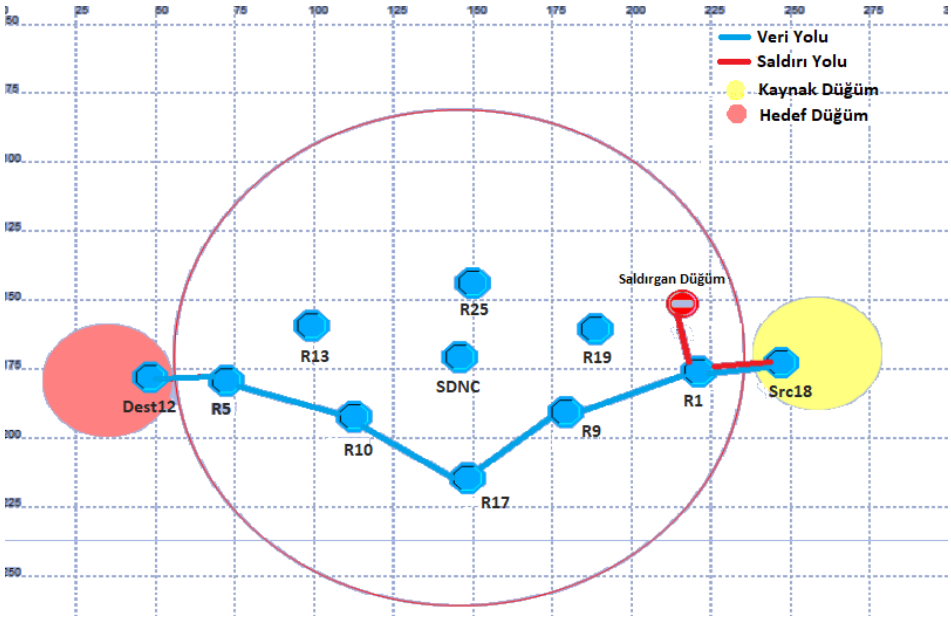
KAA'da bir Kara Delik saldırısının ana fikri, bir saldırgan düğümünün kendisini en kısa yola sahip bir düğüm olarak tanıtmasıdır. Böylelikle saldırgan düğüm tüm veri trafiğini toplar. Önerilen modelde, KAA'daki şifreli iletişimin tehlikeye atıldığı varsayılmıştır. Önceki çalışmalar, ağdaki düğümlerin sınırlı kapasiteye sahip olması nedeniyle hafif şifreleme yöntemlerinin KAA için çok daha uygun olduğunu göstermektedir. Ancak, basit şifreleme algoritmaları gibi saldırganlar için kolay bir hedeftir. Ek olarak, ağdaki Akış Tablosu'nun gönderilmesi de karmaşık olmayan bir prosedürdür. Bu gerçekler dikkate alındığında, bir saldırgan şifrelenmiş iletişimi devralabilir ve SDN'nin etkin olduğu KAA'daki düğümlere yanlış yol bilgisi enjekte edebilir. Böylece saldırgan kaynak düğüm tarafından aktarılan tüm paketleri emebilir ve bu saldırı diğer düğümler tarafından belirli bir süre boyunca hiç algılanmayabilir. Saldırgan ayrıca ağdaki trafiği dinleyebilir ve tekrar saldırısı kullanarak veri paketleri gönderebilir. Bu şekilde ağ trafiğini bozabilirler.

Şekil 4.1. saldırganın saldırıyı YTA tabanlı KAA'da nasıl gerçekleştirdiğini diyagram üzerinde göstermektedir. Adım 1'de, N_i verileri hedef düğüme aktarabilmek için SDNC'ye bir yol talebi gönderir. Ardından SDNC'den bir Onay Paketi (Acknowledge – ACK) alır. Daha sonra SDNC, Akış Tablosu'nu oluşturur ve bunu N_i'ye gönderir. Sonra N_i'den ACK alır. Şimdiye kadar belirtilen süreç YTA tabanlı KAA'nın doğal akış sürecine göre gerçekleşmiştir. Adım 2'de ise tanımlanan saldırı senaryosu şu şekilde gerçekleşmiştir. Saldırgan düğüm, t zamanında N_i'ye zararlı Akış Tablosu bilgisi gönderir. Bunu başarmak için saldırgan, N_i'nin SDNC ile iletişimini kesebilir veya yüksek enerjili SDNC'den çok daha hızlı yanıt verebilir. Güvenilmeyen modelde, saldırgan düğüm gizli dinleme yaparak SDNC'yi kolayca taklit edebilir. N_i, bu bilginin SDNC tarafından iletildiğini varsayar ve hafızadaki eski akış verilerini ezerek zararlı kuralı üzerine yazabilir. Böylece, saldırgan düğüm zararlı akış kurallarını kurban düğüme enjekte edebilir ve veri paketlerini almayı bekler.



Şekil 4.1. Saldırı Senaryosu Sequence Diyagramı

4.2.2. Saldırının Riverbed Modeler üzerinde modellenmesi



Şekil 4.2. YTA tabanlı KAA üzerinde Kara Delik Saldırı Senaryosu

Simülasyon çalışma süresince, N_{Src18} (Kaynak Düğüm) başlangıçta SDNC'ye verileri N_{Dest12} 'ye (Hedef Düğüm) gönderebilmek için bir yol talebi iletmiştir. SDNC talebi almış ve optimum bir yol bulmuştur. SDNC Akış Tablosu oluşturmuş ve bunu yoldaki tüm düğümlere dağıtmıştır. Rota bilgileri ise şu şekildedir; $N_{Src18} \rightarrow N_{R1} \rightarrow N_{R9} \rightarrow N_{R17} \rightarrow N_{R10} \rightarrow N_{R5} \rightarrow N_{Dest12}$.

Tablo 4.1. SDNC tarafından N_{R1}'e gönderilen akış kuralını göstermektedir. N_{R1}, N_{Src18}'den bir paket aldığında, onu N_{R9}'a iletmesi gerektiği anlamına gelmektedir.

Tablo 4.1. 1 numaralı Akış Tablosu

Kural	**Akış Tablosu 1**			
	Düğüm	***Kaynak***	***Hedef***	***Aksiyon***
1	R1	Src18	R9	İlet

Saldırgan düğüm N_{R1}'i hedeflemiş ve N_{R1}'den geçen paketlerin yolunu değiştirmek istemiştir. Rotayı değiştirmek için, kendisini N_{R1}'e SDNC olarak tanıtmış ve Tablo

4.2.'de görülebileceği gibi sahte bir akış kuralı göndermiştir. Düğümler, YTA tabanlı KAA'daki yol değişikliğinin nedeninin farkında değildir. Bu nedenle gelen sahte rota bilgilerini eskisi ile itaatkâr bir şekilde değiştirmişlerdir. Ardından N_{R1}, paketleri saldırgan düğüme göndermeye başlamıştır.

Tablo 4.2. 2 numaralı Akış Tablosu

Kural	**Akış Tablosu 2 (Sahte)**			
	Düğüm	***Kaynak***	***Hedef***	***Aksiyon***
1	R1	Src18	Saldırgan Düğüm	İlet

Saldırgan, farklı amaçlarla zararlı akış kuralları da gönderebilir. Örneğin, kurbana gönderilen tüm paketleri düşürmek isteyebilir. N_{R1}'den gelen paketler, Tablo 4.3.'de gösterildiği üzere başka bir yere aktarılmadan önce bırakılabilir. Ağdaki düğümler bu anormalliği uzun süre fark etmeyebilir. Bu durum ise Kara Delik saldırılarında en çok kullanılan yöntemlerden biridir.

Tablo 4.3. 3 numaralı Akış Tablosu

Kural	**Akış Tablosu 3 (Sahte)**			
	Düğüm	***Kaynak***	***Hedef***	***Aksiyon***
1	R1	Src18	-	Sil

4.2.3. Sonuçlar ve tartışma

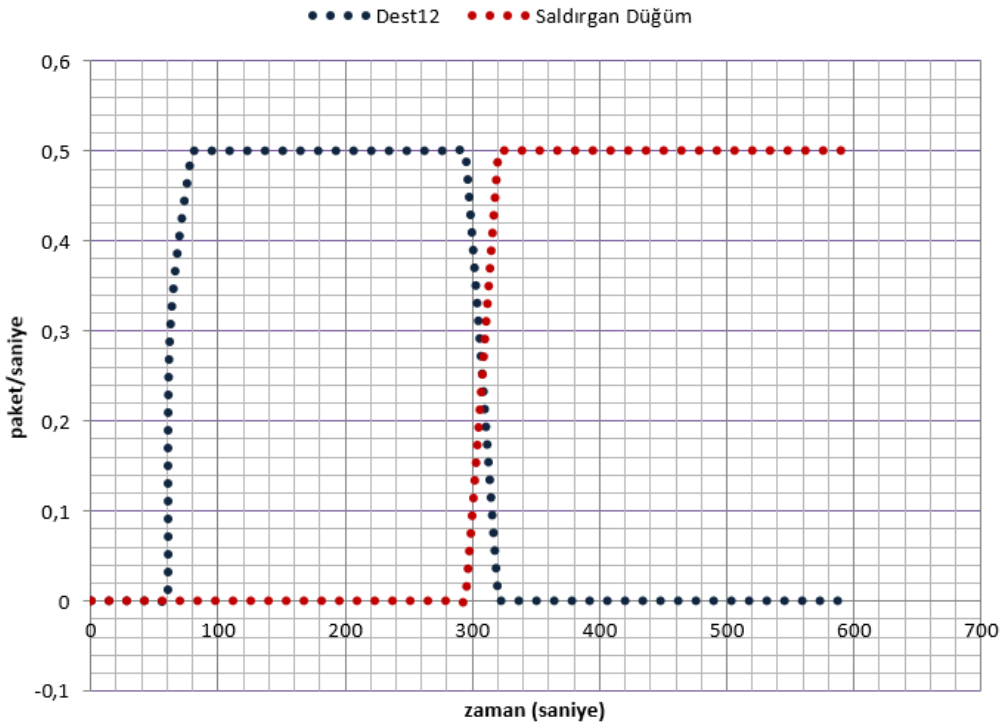
Önerilen Blokzincir-blok tabanlı güvenlik modeli, Riverbed Modeler yazılımı kullanılarak modellenmiş ve simüle edilmiştir. Simülasyon ortamındaki cihazlar için daha gerçekçi bir performans değerlendirmesi için MICAz düğüm çalışma parametreleri kullanılmıştır. Simülasyon sonuçları, 2.50 GHz i5 2450M işlemciye ve 4 GB RAM'e sahip bir bilgisayar kullanılarak elde edilmiştir. Tutarlı performans karşılaştırmaları için tüm senaryolar için benzer çalışma koşulları seçilmiştir. Simülasyon ortamında, bir boş uzay kanalı yayılma modeli kullanılmıştır. Diğer önemli simülasyon parametreleri Tablo 4.4.'de verilmiştir.

Tablo 4.4. Simülasyon parametreleri [1]

	Adı	Değeri
Önerilen model için cihaz ayarları	Veri Hızı (Data Rate)	250 kbps
	ED durumu iletim süresi	25 s
	Başlangıç enerjisi	5 J
	Kanal modeli	Free-space propagation model (LoS)
	Güç eşiği (Power threshold)	−76 dBm (80 mW)
Önerilen model için batarya parametreleri (MICAz mode)	İletim modu (0 dBm)	17.4 mA
	Alma modu (Receive mode)	27.7 mA
	Bekleme modu (Idle mode)	35 µA
	Uyku modu (Sleep mode)	16µA

4.2.4. Paket trafik sonuçları

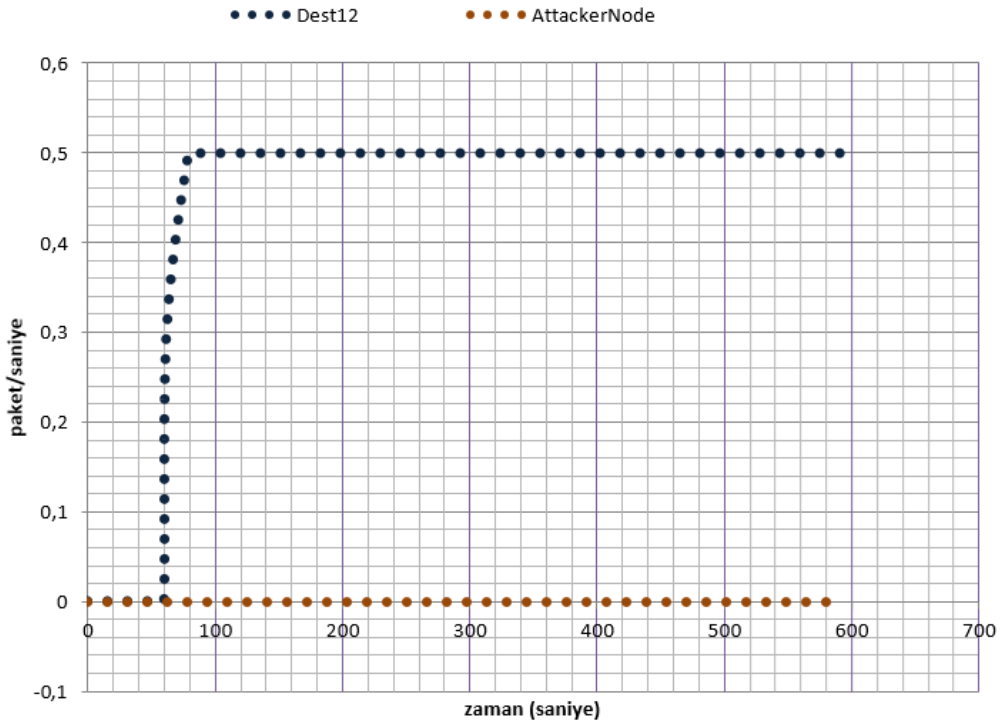
Şekil 4.3. yukarıda bahsedilen Kara Delik saldırı senaryosunun simülasyon sonuçlarını göstermektedir. Saldırgan düğüm, sahte akış kuralını t + 300'de N_{R1}'e göndermiştir. N_{R1}, bu akış kuralını hafızasına yüklemiş ve ardından N_{R1} paketleri N_{R9} yerine saldırgan düğüme göndermiştir. t + 300'den sonra, saldırgan düğümünün saniyede aldığı paket sayısı artmaya başlamıştır. Böylece tüm paketler saldırgan düğüme yönlendirilmiştir. Bu durum ise Kara Delik saldırısının hedefine ulaştığı anlamına gelmektedir. Bu durum, ağda fark edilmeden uzun süre devam edebilir. Simülasyon çalışma süresi boyunca hedefe toplam 5.000 paket gönderilmiştir ve gönderilen tüm paketler hedef düğüm tarafından kabul edilmiştir.



Şekil 4.3. Güvensiz modele gerçekleştirilen saldırıda alınan paket sayıları

Performansı detaylı değerlendirebilmek için önerilen Blokzincir-blok tabanlı güvenlik modeli de ağa uygulanmıştır. Simülasyon sonuçlarına göre Şekil 4.4.'de t + 300'de yapılan benzer saldırı sistemi bozamamıştır. N_{Dest12}, simülasyon çalışma süresinin sonuna kadar paketleri almaya devam etmiştir. Çünkü saldırgan doğru imza değerlerini tahmin edememiştir. Saldırı senaryosunda saldırgan N_{R1}'e göndermek için sahte akış kuralları oluşturmuştur. Ancak bu akış kuralları için doğru imza belirteçlerini oluşturamamıştır. Bu yüzden saldırgan rastgele imza değerleri üretmiştir. Kendisine gelen imza değerinin hatalı olduğunu anlayan N_{R1}, saldırı paketini reddetmiştir. Denklem 3.1.'de gösterildiği gibi, her akış kuralının imza işaretini hesaplamak için düğümün (K_i) gizli anahtarını bilmek ve algoritmanın karmaşıklığını çözmek gerekmektedir. Ayrıca Blokzincir-blok yaklaşımı ile her akış kuralına eklenen zaman damgası alanı sayesinde imza üretimi dinamik hale gelmiştir. Bu nedenle saldırganın edindiği bu değerleri kullanarak tersine mühendislik yapması mümkün gözükmemektedir.

Bu tür bir güvenlik açığı yalnızca ağda trafik kesintisine yol açmakla kalmamakta, aynı zamanda ağdaki diğer zararsız düğümlerin de bunlardan etkilenmesine neden olabilmektedir. Sahte Akış Tablosu bilgisi ile saldırgan atlama noktası olarak yalnızca bir düğüm seçerek tüm trafiği kendisine yönlendirebilir. Böylelikle zararsız düğüm aynı anda birden fazla trafiğe hizmet vereceği için enerjisini çok kısa sürede tüketebilir. Bu şekilde bir saldırgan, ağdaki çoğu düğümün çok kısa bir süre içinde enerjisinin bitmesine de neden olabilmektedir.

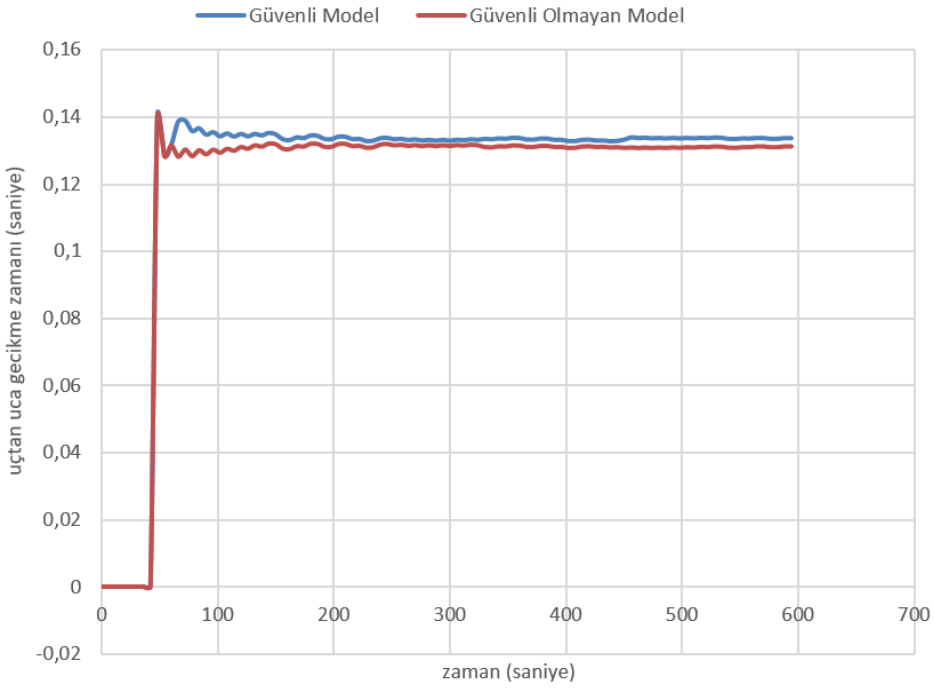


Şekil 4.4. Güvenli modelde gerçekleştirilen saldırıda alınan paket sayıları

Saldırgan farklı bir saldırı yöntemi kullanabilir. Sahte bir Akış Tablosu üretmek yerine, ağı dinleyebilir ve SDNC tarafından N_{R1}'e gönderilen Akış Tablosu'nu yakalayabilir. Bu durumda, elde ettiği Akış Tablosu bilgilerini bir tekrar saldırısı olarak kullanabilir. Ancak önerilen güvenlik modeli sayesinde bu saldırı da başarısız olacaktır. Çünkü saldırgan, N_{R1}'in belleğinde bulunan ve dinamik olarak değişen önceki karma kodu (c_{fs}) değerini tahmin etmek zorunda kalacaktır.

4.2.5. Uçtan uca gecikme sonuçları

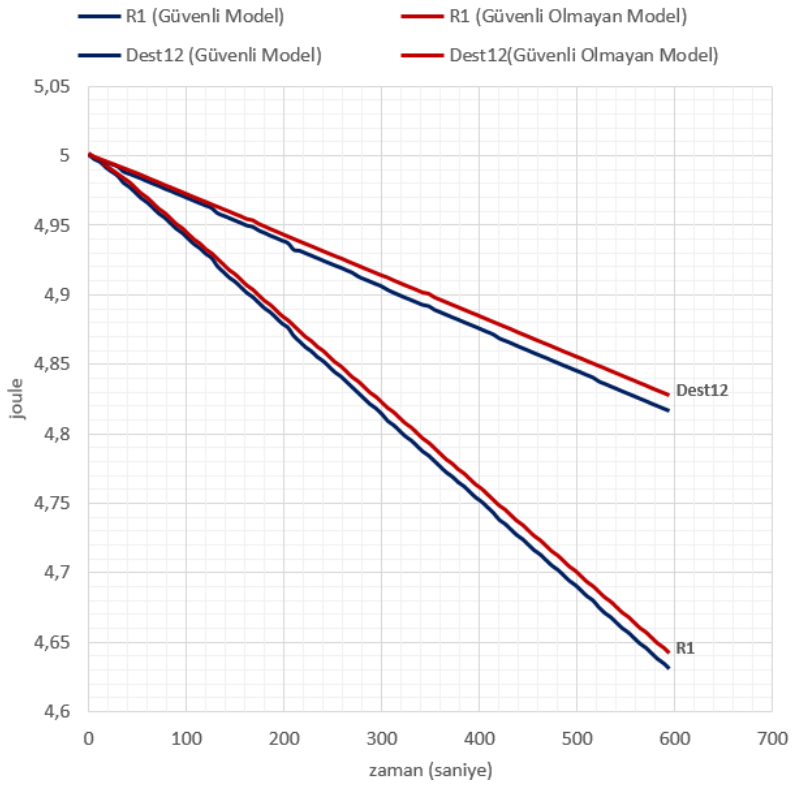
Şekil 4.5. hem güvenilir hem de güvenilmeyen modeller için N_{Src18}'den N_{Dest12}'ye kadar olan paketlerin uçtan uca gecikme sonuçlarını karşılaştırmalı olarak göstermektedir. Önerilen güvenlik modelinde, güvenilmeyen modele kıyasla ortalama 0,002 saniye daha fazla iletim gecikmesi meydana gelmiştir. Bunun nedeni ise önerilen modelde yer alan ek işlem maliyetleridir. Bu fark ise çok küçük olduğundan zamana duyarlı uygulamalarda göz ardı edilebilir.



Şekil 4.5. Src18 ve Dest12 arasında oluşan trafikte uçtan uca gecikme sonuçları

4.2.6. Enerji sonuçları

Şekil 4.6. güvenilir ve güvenilmeyen modeller için N_{Dest12} ve N_{R1}'in karşılaştırmalı zamana dayalı enerji tüketimi sonuçlarını göstermektedir. N_{Dest12} için önerilen güvenlik modelinde ortalama enerji tüketimi, güvenilmeyen modelden 0.002 Joule daha fazla iken bu miktar N_{R1} için 0.01 Joule daha fazladır. Buradaki enerji farkı, önerilen modelin sisteme yüklediği toplam hesaplama yükünü de içermektedir.



Şekil 4.6. R1 ve Dest12 düğümlerinin zamana bağlı kalan enerjileri

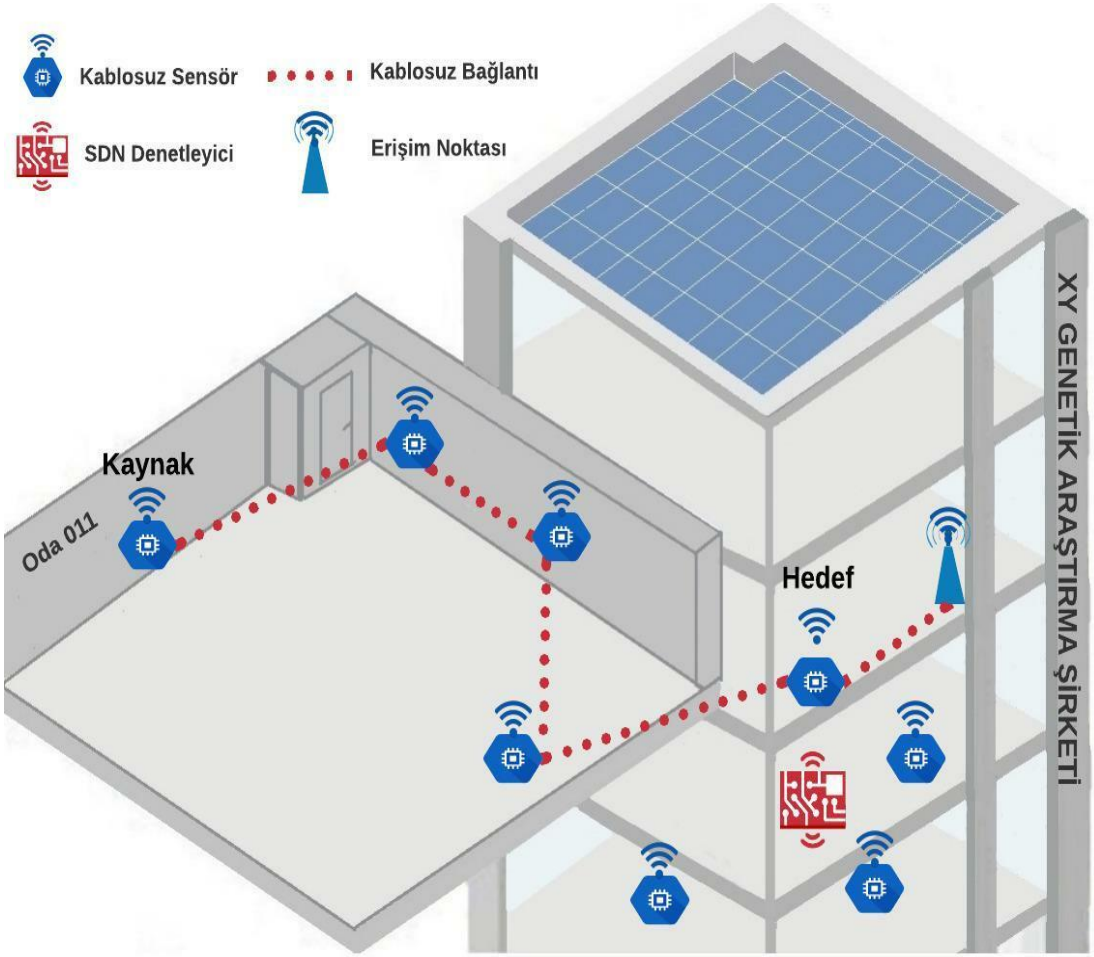
Önerilen modelin ağ üzerindeki etkilerini belirleyebilmek için uçtan uca gecikme ve enerji tüketim oranları gözlemlenmiştir. Sonuçlar, önerilen güvenlik modelinin uçtan uca gecikmeyi % 1,53 artırdığını göstermektedir. Önerilen model ayrıca enerji tüketim oranını hedef düğüm için % 0,4 ve kurban düğüm için ise % 0,2 artırmıştır. Bu değerler oldukça küçük olduğu için önerilen modelin uygulanabilirliği açısından birçok uygulamada ek maliyeti ihmal edilebilir. Ayrıca önerilen model, Ortadaki Adam Saldırısı ve Tekrarlama Saldırılarına karşı güvenlik sağlamaktadır.

4.3. Güvenli SLA Yönetimi İçin Örnek Senaryo

Önerilen güvenli SLA yönetimi modeli için örnek bir senaryo belirlenmiştir. Belirlenen senaryoda müşteri olarak ‘XY‘ adında genetik araştırmalar yapan bir firma seçilmiştir. İş tanımı ise şu şekilde belirlenmiştir;

- Müşteriye ait binada yer alan 011 numaralı araştırma odasında sıcaklık ölçümü büyük önem arz etmektedir. Bu bilgilerin ölçülerek merkezi veri sunucularına kesintisiz veri aktarılması gerekmektedir. Verilerin sürekliliği ve hızlı olması temel kriterdir.
- Gerekli iş tanımı için kurulacak ağ altyapısı ise YTA tabanlı KAA olarak belirlenmiştir.

Belirlenen iş tanımı için gerekli diğer detaylar Tablo 4.5.'de açıklanmıştır. Ayrıca Şekil 4.7. bize servis sağlayıcının XY firması için kurduğu YTA tabanlı KAA fiziksel mimarisini göstermektedir.



Şekil 4.7. YTA tabanlı KAA'da SLA yönetimi

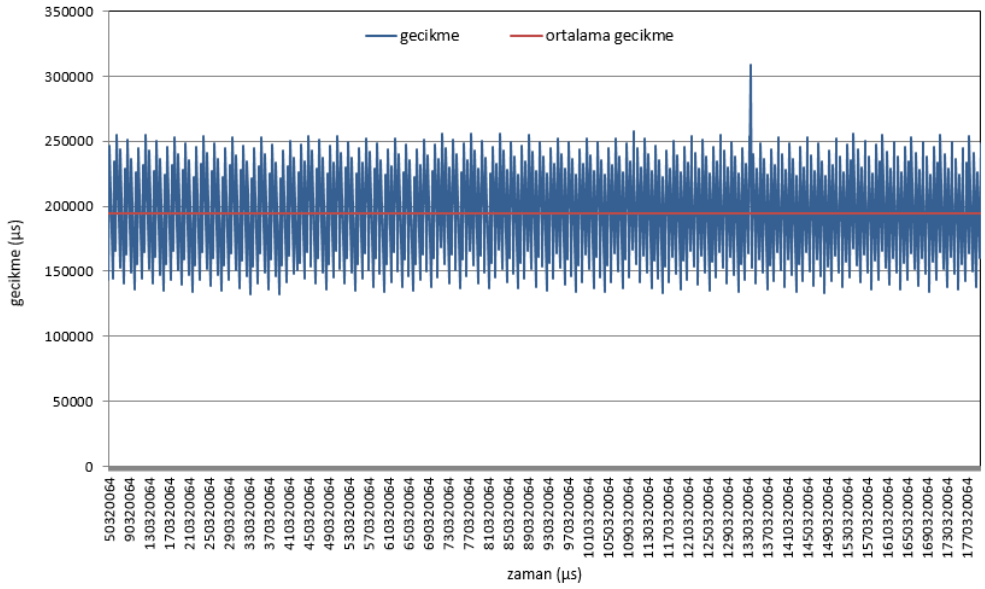
Tablo 4.5. Önerilen senaryo notasyon bilgisi

Parametre	Sabit Değeri	Tanımı
t_s	2 sn	Sensör ölçüm periyodu
c_p	150	Periyod başına ölçüm sayısı
t_c	300 sn (150*2)	Telafi Periyodu
t_{aa}	0.194 sn	Kabul edilebilir ortalama iletim süresi
t_{tad}	< %5	Hedeflenen ortalama iletim gecikme yüzdesi
D_c	30 ETH	Müşteri depozitosu
P_c	0.1 ETH	Her telafi periyodu için gecikme cezası
t_{td}	-	Toplam gecikme süresi
t_t	-	Toplam iletim süresi

XY Şirketi, genetik araştırma yapan bir müşteri şirketidir. Yapılan çalışmalar nedeniyle bina içerisindeki veri aktarım süreleri büyük önem taşımaktadır. Bu nedenle Tablo 4.5.'deki şartlar çerçevesinde müşteri ile ağ altyapısını kuran firma (servis sağlayıcı) arasında Blokzincir güvencesine dayalı bir SLA Akıllı Sözleşme (SSC) oluşturulmuştur. Akıllı Sözleşme'ye göre sensör verileri, belirlenen değerin altına düşmeden merkezi sunucuya iletilmelidir. Bu koşullar müşteri tarafından her araştırma odasında ayrı ayrı olarak belirlenmiş ve belirlediğimiz senaryo için 011 numaralı araştırma odası düşünülmüştür. Bu oda için SLA koşulları şu şekilde tanımlanmıştır;

- 011 araştırma odasındaki sıcaklık değerleri 2 saniyede bir merkez sunucuya iletilmelidir.
- Müşteri sözleşmeyi kabul ederse, 30 ETH (D_c) bir ücret yatırması gerekmektedir.
- Toplam gecikme süreleri, sensörden gelen her 150 veri için hesaplanmalıdır.
- Her verinin sunucuya ulaşması için gereken maksimum süre 0,253 saniye olarak kabul edilmiştir. Bu değeri belirlemek için ağ içi ve dışı ortalama veri iletim süreleri belirlenmiştir. Ağdaki ortalama iletim süresi için bir ön

senaryo testi simülasyonu yapılmıştır. Bu test simülasyonunda, hedef düğüme toplam 900 veri paketi iletilmiştir. Şekil 4.8. bir paketin başlangıç düğümünden hedef düğüme ne kadar süreyle iletildiğinin zamana bağlı olarak grafiğini göstermektedir. Sonuçların ortalama değeri 0.194 sn olarak elde edilmiş ve kabul edilebilir süreyi belirlemek için ortalama süreye %30 sapma payı süresi eklenmiştir.



Şekil 4.8. Son düğüm ile SCH arası veri iletimi

– Kabul edilebilir ortalama iletim gecikme yüzdesi müşteri tarafından %5 olarak belirlenmiştir. Hesaplama detayları Denklem 4.1 üzerinde gösterilmektedir.

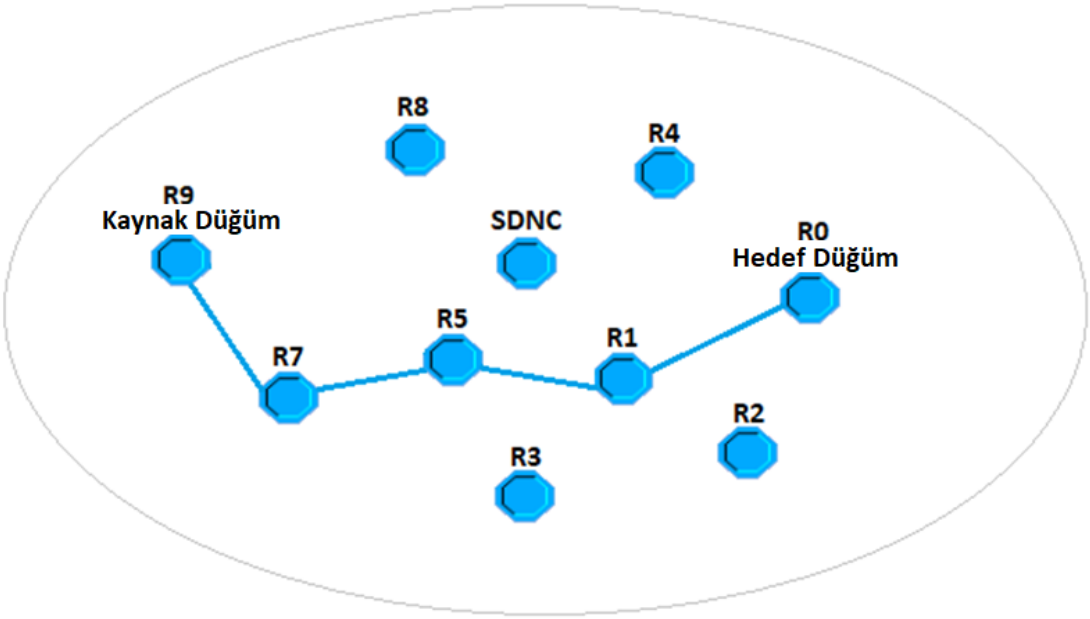
$$t_{td} = t_t - (t_c + (t_{aa} * c_s)) \quad (4.1)$$
$$t_{td} = t_t - (300 + 0{,}253 * 150)$$
$$t_{ad} = \left(\frac{t_{td}}{t_t}\right) * 100$$

– Hedeflenen ortalama iletim gecikme yüzdesi %5'den büyük ise telafi ceza ücreti (P_c) servis sağlayıcıdan kesilir.
– Bu senaryoda veri gecikmesi ana kriter olduğundan, Şekil 4.8.'deki ağ ölçümleri verilerin ağ içi oluşturma zamanı ve verilerin ağ içi varış zamanı olarak belirlenmiştir.

4.3.1. Platform bileşenleri

4.3.1.1. Simülasyon modeli

YTA tabanlı KAA yapısı Riverbed Modeler yazılımı kullanılarak modellenmiş ve simüle edilmiştir. Şekil 4.9.'ta gösterildiği üzere simülasyon senaryosunda 10 adet ED ve 1 adet SDNC cihazı mevcuttur.



Şekil 4.9. Örnek çalışma için simülasyon modeli

Daha gerçekçi bir performans değerlendirmesi için simülasyonda MICAz düğümü enerji tüketimi parametreleri kullanılmıştır. Ek olarak, tutarlı bir performans karşılaştırması elde etmek için tüm senaryolar için benzer çalışma koşulları seçilmiştir. İlgili simülasyon parametreleri Tablo 4.4. üzerinde gösterilmektedir.

4.3.1.2. Akıllı Sözleşme araçları

Önerilen modelde Akıllı Sözleşme geliştirmek için kullandığımız teknolojiler aşağıda listelenmiştir;

- Solidity (Ethereum VM Language): Solidity, Akıllı Sözleşmeler geliştirmek için kullanılan nesne yönelimli bir programlama dilidir [68]. Ethereum başta olmak üzere birçok Blokzincir platformunda kullanılmaktadır. Böylece hesapların davranışı değiştirilebilir [69]. Geliştirilen kod derlendikten sonra EVM üzerinde çalışır.
- Truffle (Akıllı Sözleşme Testi ve Dağıtım Çerçevesi): Truffle, Ethereum Akıllı Sözleşmelerinin geliştirildikten sonra basit ve kolay test edilmesini sağlayan bir geliştirici ortamıdır [70].
- Ganache (Akıllı Sözleşme Yerel VM): Ganache, geliştirilen Akıllı Sözleşmeler için sanal bir Ethereum Blokzincir ağı oluşturur [71]. Ayrıca bu sanal Blokzincir üzerinde yeni hesaplar açarak geliştirilen kod test edilebilmektedir.
- Web3js (Akıllı Sözleşme Java Sınıf Yardımcısı): web3.js, HTTP, IPC veya WebSocket kullanarak yerel veya uzak bir Ethereum ağı ile iletişim kurmanıza izin veren bir program kitaplığıdır [72].
- MetaMask (Blokzincir Hesap Paneli - Google Chrome Eklentisi): MetaMask, Ethereum Blokzincir ile iletişim kurmak için bir tarayıcı uzantısı olarak geliştirilmiş bir kripto para cüzdanıdır [73]. Uyumlu bir tarayıcı sayesinde kullanıcılar kendi hesaplarını yönetebilir ve bu eklenti ile ETH gönderebilir veya alabilir.

4.3.1.3. Akıllı Sözleşme Yardımcısı (SCH)

SCH, Akıllı Sözleşmeler ile YTA tabanlı KAA arasında bir köprü görevi gören Spring Boot [74] tabanlı bir platformdur. YTA tabanlı KAA'dan gelen istekleri ele alır ve bunları Blokzincir ağına iletir. Bu platformu yöneten taraf, platformun sürdürülebilirliğini sağlaması gereken hizmet sağlayıcıdır. Akıllı Sözleşmelerle iletişim sağlamak için SCH üzerinde Web3.js Java kitaplığı kullanılmıştır. Ek olarak, platformdaki çeşitli görevler için çeşitli RESTful POST hizmetleri geliştirilmiştir.

4.3.1.4. Gezegenler Arası Dosya Sistemi (IPFS)

IPFS, genellikle dağıtılmış bir uçtan uca ağda veri depolamaya ve paylaşmaya olanak veren dağıtılmış bir dosya sistemi olarak tanımlanabilir [75]. IPFS sisteminde her dosyaya ve tüm bloklarına benzersiz bir anahtar kodu verilir. Ağdaki düğümler ilgilenilen içeriği ve diğer düğümlerin içeriğiyle ilgili bazı indeksleme bilgilerini depolar. Böylelikle silmeye karşı güvenli bir web ortamı sağlanır.

4.3.2. Sonuçlar ve tartışma

Örnek çalışmanın simülasyon sonuçları, 2.50 GHz i5 2450M işlemciye ve 4 GB RAM'e sahip bir bilgisayar kullanılarak elde edilmiştir. Simülasyon 30 dakika boyunca çalıştırılmış ve hedef düğüme toplam 900 sıcaklık verisi iletilmiştir. Hedef düğüm, her veriyi bazı parametrelerle (genel anahtar, veri oluşturma zamanı, verinin varış zamanı ve kontrol kodu) Akıllı Sözleşme Yardımcısı'na iletilmiştir. Öte yandan Akıllı Sözleşme Yardımcısı, gelen verileri hem IPFS'ye hem de SSC'ye iletmiştir.

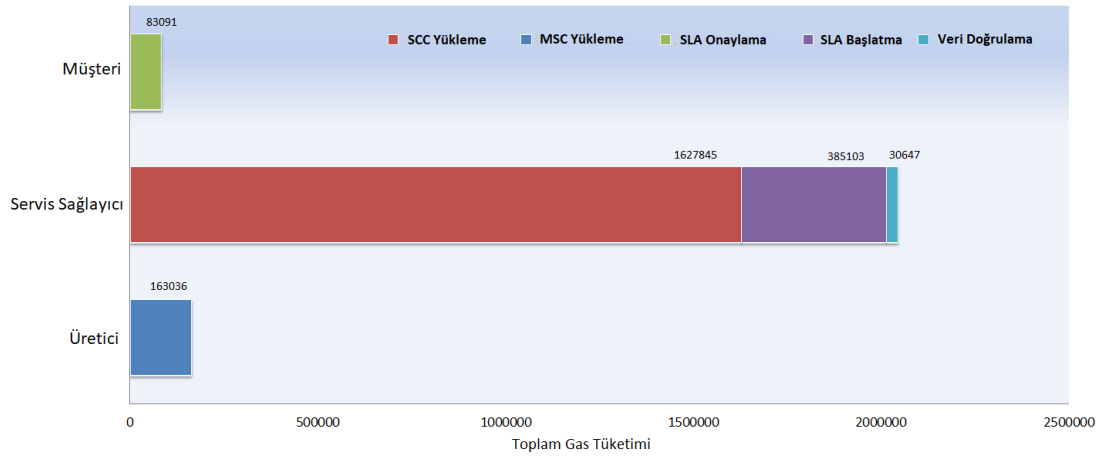
Her telafi döneminde, yani her 5 dakikada bir SCC tarafından gecikme hesabı yapılmıştır. Gecikme belirlenen değerin üzerinde ise ceza kontrolü uygulanmıştır. SCC, gelen verilerin kontrol kodunu MSC'ye ileterek kontrol kodunun doğru veya yanlış olduğunu onaylamıştır. Simülasyon sırasında hatalı kontrol kodu oluşturulmamıştır. Sistem başlatılmadan önce müşteri, servis sağlayıcı ve SCC hesaplarındaki bakiyeler Tablo 4.6.'da gösterilmektedir.

Tablo 4.6. Müşteri, servis sağlayıcı ve Kontrat hesaplarına ait bakiye başlangıç değerleri

Hesap Adı	Başlangıç Değeri
Servis Sağlayıcı	5 ETH
Müşteri	40 ETH
SCC	0,5 ETH

Not: Gas Birim Ücreti = 20 Gwei = ($\boldsymbol{20 * 10^{-9}\ ETH}$)

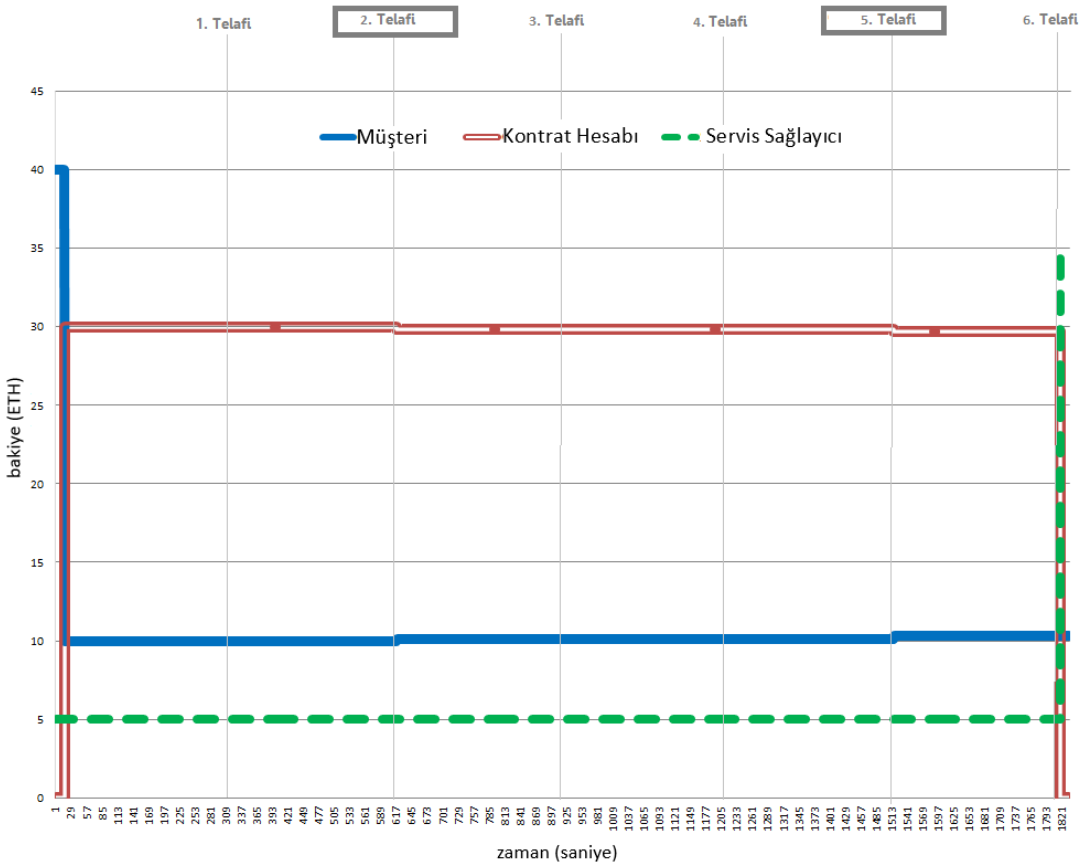
Servis sağlayıcının başlangıç bakiyesi, Akıllı Sözleşme'de SLA'yı başlatmak için gereken işlem ücretini sağlamak için 5 ETH olarak belirlenmiştir. Benzer şekilde, SCC'nin ceza uygulaması için ilk bakiye değeri 0,5 ETH olarak belirlenmiştir. Simülasyon sonuçlarını daha detaylı hale getirmek için depozito ve ceza ücreti gibi ücretler yüksek değerler olarak seçilmiştir. Şekil 4.10.'da, Truffle üzerinden alınan Akıllı Sözleşme işlemlerinin 'gas' tüketim miktarını göstermektedir. Taraflar arasında en çok 'gas' tüketen servis sağlayıcı olmuştur. Bu miktarın yüksek olmasının nedeni, SCC içerisinde SLA yönetimi için çeşitli işlevlerin barındırılması ve bazı kritik verilerin depolanmasıdır. Örneğin, 32 baytlık veriyi depolamak için 20000 gas tüketimi gereklidir [61]. Bu nedenle, SCC'nin yükleme maliyeti yüksek olacaktır. SCC ve MSC dağıtımı için tüketilen toplam gas miktarı 1790881 (0.0358 ETH) olarak hesaplanmıştır. Müşteri SCC'sinde tanımlanan SLA'yı onaylamak için toplam gas tüketimi 83091 iken, hizmet sağlayıcının bu SLA'yı başlatmak için harcadığı gas miktarı 385103 olmuştur. Ayrıca SCC'deki CheckData fonksiyon çağırımında 33974 gas kullanılmıştır. Ayrıca müşteriye ceza ödemesi yapılması için 33974 gas tüketilmesi gerekmiştir. Ethereum, para transferleri için belirli bir gas tüketimi gerektirir. Kullanıcının gas tüketim miktarını yüksek tuttuğu sürece transfer işlemi de şebeke üzerinde o kadar hızlı gerçekleşmektedir. Simülasyonumuzda para transferi için harcanan gas tüketim miktarı 8281 olarak belirlenmiştir.



Şekil 4.10.Akıllı Sözleşme işlemleri için tüketilen gas miktarları

Şekil 4.11. hesap bakiyelerinin zamana bağlı değişimini göstermektedir. SLA'nın onayı ile müşteri hesabından 30 ETH SCC hesabına aktarılmıştır. Toplam iletim

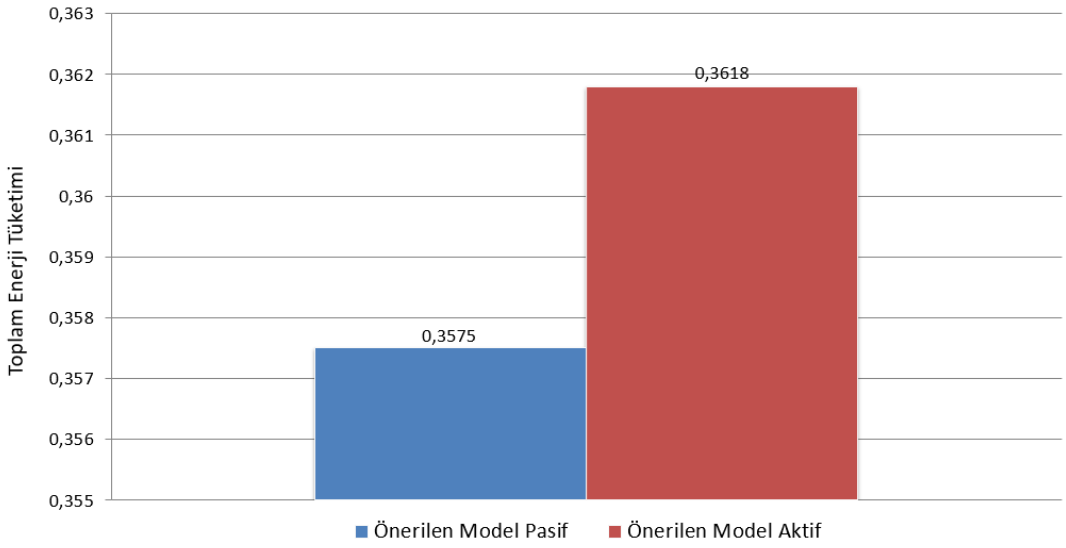
süresi 6 telafi süresi ile kontrol edilmiştir. Telafi zaman aralıklarından 2 ve 5 aralıkları hedeflenen süreyi aşmıştır. Bu nedenle her iki tazminat aralığında da 0.15 ETH SCC hesabından müşterinin hesabına aktarılmıştır. Süre tamamlandığında SLA feshedilmiş ve SCC, kalan bakiyeyi servis sağlayıcıya aktarmıştır.



Şekil 4.11. Müşteri, servis sağlayıcı ve Kontrat hesap bakiyelerinin veri transferi ve veri doğrulama sürecinde zamana bağlı değişimleri

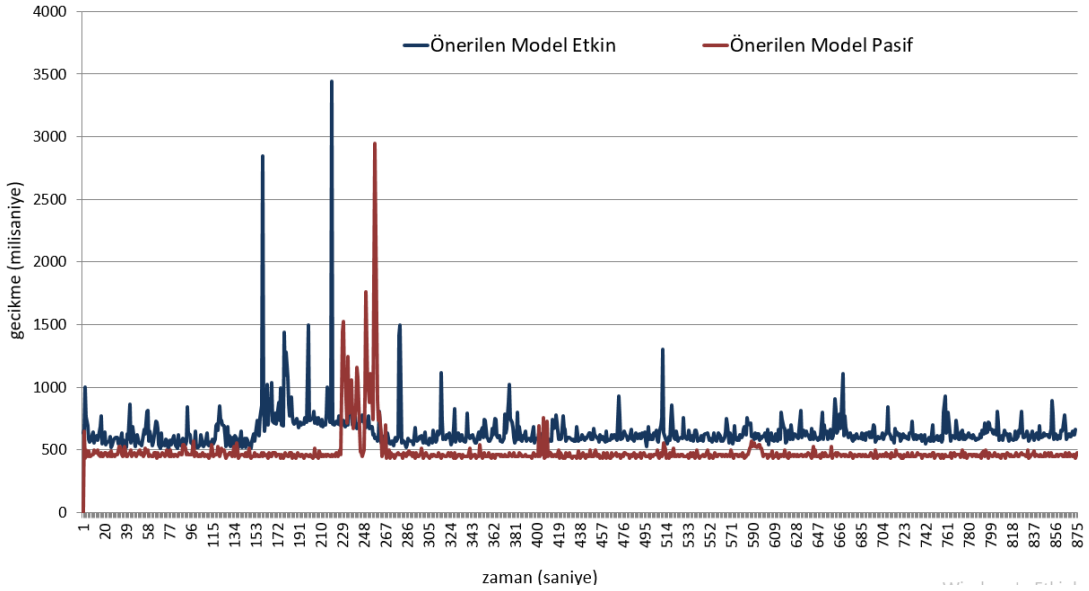
Riverbed Modeler üzerinde elde edilen simülasyon sonuçlarının karşılaştırılabilir olmasını sağlamak için simülasyon iki model üzerinde çalıştırılmıştır. İlk modelde, Riverbed Modeler modelinde herhangi bir değişiklik yapılmamıştır. İkincisinde, önerilen model uygulanmış ve kontrol kodunu oluşturmak için gerekli işlemler ED'nin uygulama katmanında uygulanmıştır. Böylece bu iki modeli kullanarak Şekil 4.12. ve Şekil 4.13.'teki sonuçlar elde edilmiştir. Şekil 4.12. çalışma sırasında iki model arasındaki ortalama enerji tüketim farkını göstermektedir. Enerji

tüketimindeki fark 0,0043 Joule olarak hesaplanmıştır. Buradaki enerji farkı, önerilen modelin sisteme yüklediği toplam hesaplama yükünü göstermektedir.



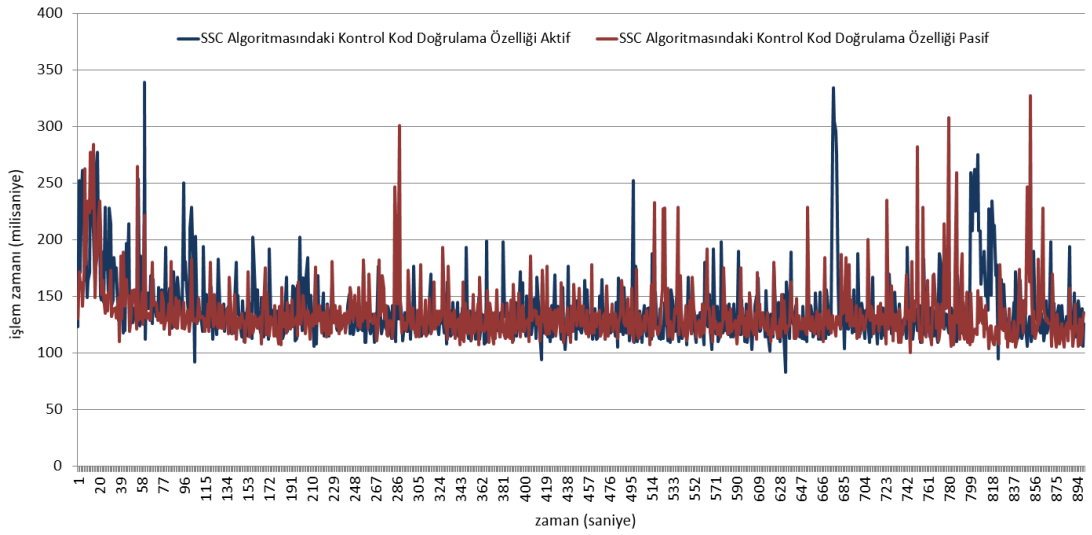
Şekil 4.12. Önerilen modelin aktif/pasif durumuna göre toplam enerji tüketimleri

Şekil 4.13. ED ile SCH arasında aktarılan iki SLA verisi arasında geçen süreyi göstermektedir. Grafikteki veriler ED üzerinde kontrol kodu oluşturma işleminin aktif/pasif olmasına göre elde edilmiştir. Sonuçlara göre, ED'de kontrol kodunu oluşturmak için gereken ek işlem maliyeti nedeniyle SCH'ye veri aktarımı sırasında ortalama 4.561 milisaniyeye varan bir gecikme yaşanmıştır.



Şekil 4.13. Son düğüm üzerindeki kontrol kodu özelliğinin aktif/pasif durumuna göre uç düğüm ile SCH arasındaki veri iletimi gecikmesi

Şekil 4.14. algoritmanın işlem süresini göstermektedir. Yukarıda, SSC'nin aldığı verilerin doğru olduğunu onaylamak için MSC'de bir fonksiyonu çağırdığı belirtilmişti. Bu işlem, devre dışı bırakılmış ve bu eylemi çağırmanın toplam iletim sürelerine ne kadar ek maliyet getireceğini belirlemek için simülasyon yenilenmiştir. Elde edilen bu iki sonuç Şekil 4.14.'te gösterilmektedir. Sonuçlara göre ortalama sürede yaklaşık 3.006 milisaniye gibi göz ardı edilebilecek bir zaman gecikmesi meydana gelmiştir.



Şekil 4.14. SSC algoritmasında kontrol kod doğrulama özelliğinin aktif/pasif durumuna göre SSC algoritmasının işlem zamanı

Sonuçlar ED'den SCH'ye veri aktarımı sırasında % 0,75'lik bir gecikme olduğunu ve ED için enerji tüketim oranını % 1,2 oranında artırdığını göstermektedir.

BÖLÜM 5. SONUÇ VE ÖNERİLER

Bu çalışmada, Blokzincir teknolojileri ve altyapısı kullanılarak YTA tabanlı KAA üzerinde iki farklı çalışma geliştirilmiştir. Birinci çalışmada, Blokzincir-blok yaklaşımı kullanılarak Kara Delik saldırılarına karşı veri bütünlüğünü ve enerji verimliliğini destekleyen deterministik yeni bir güvenlik modeli geliştirilmiştir. Geliştirilen bu güvenlik modeli sayesinde çeşitli saldırı türlerine karşı hem rota hem de veri güvenliği sağlanabilmektedir. Modelde, tüm Akış Tablosu bilgileri Blokzincir-blok yaklaşımı kullanılarak bloklara bölünmüştür. Bu blokların karmaları Merkle Ağacı yapısında saklanmış ve böylece tartışmasız değiştirilemez ve şeffaf bir ağ geçmişi elde edilebilmiştir. Bu yaklaşım sayesinde elde edilen geçmiş, özellikle adli araştırmalarda kanıt olarak kullanılma potansiyeline sahiptir. İkinci modelde ise Blokzincir teknolojisini kullanan Akıllı Sözleşmeler sayesinde YTA tabanlı KAA üzerinde güvenli SLA yönetimi modeli geliştirilmiştir. Önerilen modelde yenilenen işlemlerde ‘gas’ kullanımı minimum seviyede kalması sağlanmıştır. Özellikle önerilen modelin temel noktası olan veri kontrol kodu doğrulama fonksiyonunun ücretsiz olması ve düğümlerde gerçekleşen ek işlemlerin enerji ve performans bakımından kayda değer yük getirmemesi gerçek dünya uygulamalarında kullanabilirlik açısından umut verici olduğu düşünülmektedir. Ayrıca bu çalışma ile birlikte düğümlerde üretilen kontrol kodu mekanizması sayesinde düğüme yeni program yükleme ve kurcalama gibi saldırı vektörlerine karşı etkili olacağı kanaatine varılmıştır. Çünkü korumalı hafızada saklanmış gizli kod sayesinde saldırganın kontrol kodu üretim işlemini taklit etmesi çok zordur.

Blokzincir teknolojisi gün geçtikçe yaygın kullanım oranını artırmış ve birçok alanda çeşitli teknolojilerle etkileşime geçerek yeni mimarilerin ortaya çıkmasına aracı olmuştur. Kablosuz ağların esnek yapısı ve Blokzincir’in etkin gücü gelecekte çeşitli alanlarda kullanım fırsatı sunmaktadır. Önerilen çalışmanın aşağıdaki sektörlerde de kullanım sağlanabileceği düşünülmektedir;

- Ağda oluşan zafiyetler sonucu meydana gelebilecek müşteri taraflı zararlarda (yangın, sistemlerin durdurulması, vb) taraflar arasında meydana gelebilecek anlaşmazlıkların kaldırılması,
- Ağın kiralanması, devredilebilmesi,
- Ağa sensör ekleme (kendi sensörüne sahip bir kullanıcının belirli protokollerin yer aldığı Akıllı Sözleşmeler sayesinde kablosuz ağa katılarak kendi verisini kendi sensörü ile tedarik edebilmesi),
- Ağ kullanım süreleri, veri sayılarının SLA üzerinde belirlenmesi.

KAYNAKLAR

[1] Ç. C. Karakoç Emre, “Black Hole Attack Prevent Scheme using Blockchain-Block Approach in SDN-Enabled WSN,” *Int. J. Ad Hoc Ubiquitous Comput.*, vol. 37, no. 1, pp. 37–49, 2021.

[2] D. Latha and K. Palanivel, “Secure routing through trusted nodes in wireless sensor networks - a survey,” *Int. J. Adv. Res. Comput. Eng. Technol.*, vol. 3, no. 11, pp. 3792–3799, 2014.

[3] A. Karakaya and S. Akleylek, “A survey on security threats and authentication approaches in wireless sensor networks”, *2018 6th International Symposium on Digital Forensic and Security (ISDFS)*, 2018.

[4] A. Perrig, J. Stankovic, and D. Wagner, “Security in wireless sensor networks,” *Commun. ACM*, vol. 47, no. 6, pp. 53–57, 2004.

[5] W. Tang, K. Zhang, J. Ren, Y. Zhang, and X. S. Shen, “Flexible and efficient authenticated key agreement scheme for BANs based on physiological features,” *IEEE Trans. Mob. Comput.*, vol. 18, no. 4, pp. 845–856, 2019.

[6] K. N. Shreenath and V. M. Manasa, “Black hole attack detection in zone based WSN,” *Int. J. Recent Innov. Trends Comput. Commun.*, vol. 5, no. 4, pp. 148–151, 2017.

[7] P. Nayak, V. Bhavani, and B. Lavanya, “Impact of black hole and sink hole attacks on routing protocols for WSN,” *Int. J. Comput. Appl.*, vol. 116, no. 4, 2015, doi: ISSN: 0975-8887.

[8] A. Moh’d, H. Marzi, N. Aslam, W. Phillips, and W. Robertson, “A secure platform of wireless sensor networks”, *The 2nd International Conference on Ambient Systems,* 2011.

[9] Ç. C. Al Hubaishi M. and A. S. A., “A novel energy-aware routing mechanism for SDN-enabled WSAN”, *International Journal of Communication Systems*, 2018.

[10] I. Standard and I. C. Society, *Low-Rate Wireless Personal Area Networks (LR-WPANs)*, vol. Part 15.4. Local and Metropolitan Area Networks, 2011.

[11] F. T. and F. V. G. Gaillard, D. Barthel, “Service Level Agreements for WSN: a WSN Operator’s Point of View,” *2014 IEEE Netw. Oper. Manag. Symp.*, 2014, doi: 10.1109/NOMS.2014.6838261.

[12] S. Otoum, B. Kantarci, and H. T. Mouftah, “Hierarchical trust based, black hole detection in wsn based smart grids,” in *IEEE International Conference on Communications (ICC)*, 2017.

[13] A. Aljumah and T. A. Ahanger, “Futuristic method to detect and prevent black-hole attack in wireless sensor networks,” *IJCSNS Int. J. Comput. Sci. Netw. Secur.*, vol. 17, pp. 194–201, 2017.

[14] S. Sharma and S. Gambhir, “CRCMD&R: cluster and reputation based cooperative malicious node detection & removal scheme in MANETs,” in *11th International Conference on Intelligent Systems and Control (ISCO)*, 2017, pp. 36–340.

[15] P. Dewal, G. S. Narula, and V. Jain, “Detection and prevention of black hole attacks in cluster based wireless sensor networks,” in *3rd International Conference on Computing for Sustainable Global Development (INDIACom),IEEE*, 2016, pp. 3399–3403.

[16] M. U. Farooq, X. Wang, R. Yasrab, and S. Qaisar, “Energy preserving detection model for collaborative black hole attacks in wireless sensor networks, IEEE,” in *12th International Conference on Mobile Ad-Hoc and Sensor Networks*, 2016, pp. 395–399.

[17] J. Kakarla, B. Majhi, and B. R. Babu, “A trust based secured coordination mechanism for WSAN,” in *International Conference on Signal Processing*, 2015, pp. 1–5.

[18] M. M. Ozcelik, E. Irmak, and S. Ozdemir, “A hybrid trust based IDS for WSN,” *Networks, Comput. Commun.*, pp. 1–6, 2017.

[19] G. Kaur, V. K. Jain, and Y. Chaba, “Detection and prevention of black hole attacks in WSN,” in *International Conference on Intelligent, Secure, Dependable Systems in Distributed and Cloud Environments (ISDDC), IEEE*, 2017, pp. 118–126.

[20] A. B. Karuppiah, J. Dalfiah, K. Yuvashri, and S. Rajaram, “An improvised hierarchical black hole detection algorithm in wireless sensor networks,” in *International Conference on Innovation Information in Computing Technologies (ICIICT),IEEE*, 2015, pp. 1–7.

[21] M. Tiwari, K. V Arya, R. Choudhari, and K. S. Choudhary, “Designing intrusion detection to detect black hole and selective forwarding attack in WSN based on local information”, *2009 Fourth International Conference on Computer Sciences and Convergence Information Technology*, 2009.

[22] G. Arulkumaran and R. K. Gnanamurthy, “Fuzzy trust approach for detecting black hole attack in mobile adhoc network,” *Springer Mob. Netw. Appl. Sept.*, vol. 24, no. 2, pp. 1–8, 2017, doi: ISSN: 1572-8153.

[23] M. Singh, R. Das, M. K. Sarkar, K. Majumder, and S. K. Sarkar, “A key-based two-tier trust management filtering scheme for intrusion detection in WSN,” *Proc. Second Int. Conf. Comput. Commun. Technol. Springer, India*, vol. 1, pp. 679–690, 2016.

[24] R. Alattas, “Detecting black-hole attacks in WSNs using multiple base stations and check agents,” *Futur. Technol. Conf. San Fr. CA, USA, IEEE*, vol. 6, pp. 1020–1024, 2016.

[25] A. Kaur, “Detection and isolation of black hole attack in WSN using hybrid technique (received and time delay),” *Int. Conf. Inven. Comput. Technol. (ICICT),* vol. 2, pp. 1–5, 2016, [Online]. Available: Coimbatore,.

[26] Y.-S. Kim, N. Sun, and S.-H. Lee, “Digital Signature Model of Sensor Network Using Hash Chain,”, *2010 The 2nd International Conference on Computer and Automation Engineering (ICCAE),* 2010, doi: ISBN: 978-1-4244-5586-7.

[27] U. Iqbal and S. Intikhab, “Re-keying mechanism for TinySec using ECC and hash chains”, *2017 International Conference on Advanced Computing and Communication Systems (ICACCS-2017)*, 2017.

[28] M. Tariq, M. Khan, and S. Fatima, “Detection of false data in wireless sensor network using hash chain”, *2018 International Conference on Applied and Engineering Mathematics*, 2018.

[29] A. D. L. R. Gomez-Arevalillo and P. Papadimitratos, “Blockchain-based public key infrastructure for inter-domain secure routing,” in *Proceedings of the International Workshop on Open Problems in Network Security (iNetSec)*, 2017, pp. 20–38.

[30] J. Li, G. Liang, and T. Liu, “A novel multi-link integrated factor algorithm considering node trust degree for blockchain-based communication,” *KSII Trans. Internet Inf. Syst.*, vol. 11, no. 8, pp. 3766–3788, 2017.

[31] G. Ramezan and C. Leung, “A blockchain-based contractual routing protocol for the internet of things using smart contracts,” *Wirel. Commun. Mob. Comput.*, 2018, doi: p.4029591.

[32] J. Yang, S. He, Y. Xu, L. Chen, and J. Ren, “A trusted routing scheme using blockchain and reinforcement learning for wireless sensor networks,” *Sensors*, vol. 19, p. 970, 2019, doi: 10.3390/s19040970.

[33] A. Kumar and A. R. Pais, “Blockchain based en-route filtering of false data in wireless sensor networks”, *Conference: 2019 11th International Conference on Communication Systems & Networks (COMSNETS)*, 2019.

[34] S. R. Basnet and S. Shakya, “BSS: blockchain security over software defined network”, *International Conference on Computing*, 2017.

[35] P. K. Sharma, S. Singh, Y.-S. Jeong, and J. H. Park, “DistBlockNet: a distributed blockchains-based secure SDN Architecture for IoT networks,” *IEEE Commun. Mag.*, vol. 55, no. 9, pp. 78–85, 2017.

[36] J. M.-B. João Paulo de Brito Gonçalves, Roberta Lima Gomes, Rodolfo da Silva Villaca, Esteban Municio, “A Quality of Service Compliance System Empowered by Smart Contracts and Oracles,” *2020 IEEE Int. Conf. Blockchain*, 2020.

[37] J. M.-B. João Paulo de Brito Gonçalves, Rodolfo da Silva Villaca, Esteban Municio, “A Service Level Agreement Verification,” *2020 IEEE 11th Int. Conf. Softw. Eng. Serv. Sci.*, 2020, doi: 10.1109/ICSESS49938.2020.9237735.

[38] E. J. Scheid, B. B. Rodrigues, L. Z. Granville, B. Stiller, “Enabling dynamic sla compensation using Blockchain-based Smart Contracts,” in *IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, 2019, pp. 53–61.

[39] R. B. Uriarte, R. de Nicola, K. Kritikos, “Towards distributed sla management with Smart Contracts and Blockchain,” *2018 IEEE Int. Conf. Cloud Comput. Technol. Sci.*, pp. 266–271, 2018.

[40] R. D. N. R. B. Uriarte, H. Zhou, K. Kritikos, Z. Shi, Z. Zhao, “Distributed service-level agreement management with Smart Contracts and Blockchain,” *Concurr. Comput. Pract. Exp. Spec. Issue Pap.*

[41] H. Zhou, X. Ouyang, Z. Ren, J. Su, C. de Laat, Z. Zhao, “Enforcing trustworthy cloud SLA with witnesses: A game theory–based model using Smart Contracts,” *Concurr. Comput. Pract. Exp. Spec. Issue Pap.*, 2019.

[42] P. H. Y. C. Hu, T. T. Lee, D. Chatzopoulos, “Analyzing Smart Contract interactions and contract level state consensus,” *Spec. Issue Spec. Issue Cryptocurrencies Blockchains Distrib. Syst.*, vol. 32, no. 12, 2019.

[43] B. S. E. J. Scheid, “Automatic SLA Compensation based on Smart Contracts”, *No. IFI-2018.02,University of Zurich Department of Informatics*, 2018.

[44] O. Mammela, J. Backman, S. Yrjola, K. Valtanen, “Blockchain network slice broker in 5g: Slice leasing in factory of the future use case,” *2017 Internet Things Bus. Model. Users, Networks. IEEE*, pp. 1–8, 2017.

[45] X. Costa-Perez, L. Zanzi, A. Albanese, V. Sciancalepore, “Ns-bchain: A secure Blockchain framework for network slicing brokerage,” *arXiv Prepr. arXiv2003.07748, 2020*.

[46] X. G. S. Zheng, T. Han , Y. Jiang, “Smart Contract-Based Spectrum Sharing Transactions for Multi-Operators Wireless Communication Networks,” *IEEE Access*, vol. 8, pp. 88547 – 88557, 2020.

[47] O. B. A. S. Omar, “Identity Management in IoT Networks Using Blockchain and Smart Contracts”, *2018 IEEE Confs on Internet of Things, Green Computing and Communications, Cyber, Physical and Social Computing, Smart Data, Blockchain, Computer and Information Technology, Congress on Cybermatics,* 2018, doi: 10.1109/Cybermatics_2018.2018.00187.

[48] S. A.-D. A.-S. M. Shurman, A.Al-Rahman Obeidat, “Blockchain and Smart Contract for IoT”, *2020 11th International Conference on Information and Communication Systems (ICICS),* 2020.

[49] A. P. P. H. D. R. Putra, B.Anggorojati, “Blockchain and smart-contract for scalable access control in Internet of Things”, *2019 International Conference on ICT for Smart Society (ICISS)*, 2019.

[50] S. Y. M. A. B. Ahmadon, “IoT Device Multi-layer Connection Management Mechanism with Blockchain Smart Contracts”, *2020 Zooming Innovation in Consumer Technologies Conference (ZINC),* 2020.

[51] T.-V. Le H.-A. Pham, T.-K. Le, T.-N.-M. Pham, H.-Q.-T. Nguyen, “Enhanced Security of IoT Data Sharing Management by Smart Contracts and Blockchain”, *2019 19th International Symposium on Communications and Information Technologies (ISCIT),* 2019.

[52] J. W. Y. Zhang, S. Kasahara, Y. Shen, X. Jiang, “Smart Contract-Based Access Control for the Internet of Things,” *IEEE Internet Things J.*, vol. 6, no. 2, 2019.

[53] T. T. Y. N. Aung, “Ethereum-based Emergency Service for Smart Home System: Smart Contract Implementation”, *147 International Conference on Advanced Communications Technology(ICACT),* 2019.

[54] T. G. C. Lehnert, G. Engel, “Distributed Ledger and Smart Contract Based Approach for IoT Sensor Applications”, *2020 International Conference on Omni-layer Intelligent Systems (COINS),* Barcelona, Spain, 2020.

[55] L. H. and D. Kim, “LA-Based Sharing Economy Service with Smart Contract for Resource Integrity in the Internet of Things,” *Applied Sciences*, vol. 9, no. 17, 2019.

[56] Ç. C. Al-Shaikhli A. and M. Al-Hubaishi, “WSANFlow: an interface protocol between sdn controller and end devices for SDN-oriented WSAN,” *Wirel. Pers. Commun.*, vol. 101, no. 2, pp. 755–773, 2018.

[57] “IOTLab Sakarya,” *http://www.iotlab.sakarya.edu.tr/Projects/*. http://www.iotlab.sakarya.edu.tr/Projects/. Erişim Tarihi: 03.02.2021.

[58] “Bitcoin Hashing Algorithm.” https://en.bitcoin.it/wiki/Block_hashing_algorithm. Erişim Tarihi: 03.02.2021.

[59] L. Lamport, “Password authentication with insecure communication,” *Comput. Sci. Publ. CACM*, 1981, doi: 10.1145/358790.358797.

[60] “Hash Chain,” *https://en.wikipedia.org/wiki/Hash_chain*. https://en.wikipedia.org/wiki/Hash_chain.

[61] G. W. et Al., “Ethereum: A secure decentralised generalised transaction ledger,” *Ethereum Proj. yellow Pap.*, vol. 151, pp. 1–32, 2014.

[62] S. Nakamoto, “A Peer-to-Peer Electronic Cash System”, 2008, http://www.bitcoin.org/bitcoin.pdf. Erişim Tarihi: 03.02.2021.

[63] V. Buterin, “Ethereum - On Public and Public Blockchains,” *Ethereum - On Public and Public Blockchains*, 2015. https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains. Erişim Tarihi: 03.02.2021.

[64] “Ethereum Wiki.” https://tr.wikipedia.org/wiki/Ethereum. Erişim Tarihi: 08.02.2021.

[65] C. N. G. Iordache, Ad. Paschke , M. Monacu, “Service Level Agreement Characteristics of Monitoring WSN for Water Resource Management (SLAs4Water),” *Stud. Informatics Control*, 2017.

[66] E. E. Khin and T. Phyu, “Impact of black hole attack on aodv routing protocol,” *Int. J. Inf. Technol. Model. Comput.*, vol. 2, no. 2, p. 9, May , doi: 10.5121/ijitmc.2014.2202.

[67] M. Al-Shurman, S.-M. Yoo, and S. Park, “Black hole attack in mobile ad hoc networks,” *Assoc. Comput. Mach. Southeast Conf.*, 2004.

[68] “Solidity,” *https://en.wikipedia.org/wiki/Solidity*. https://en.wikipedia.org/wiki/Solidity. Erişim Tarihi: 03.02.2021.

[69] “Solidity Wiki,” *https://docs.soliditylang.org/en/v0.8.1/*. https://docs.soliditylang.org/en/v0.8.1/. Erişim Tarihi: 03.02.2021.

[70] “Truffle,” *https://www.trufflesuite.com/docs/truffle/overview*. https://www.trufflesuite.com/docs/truffle/overview. Erişim Tarihi: 03.02.2021.

[71] “Ganache,” *https://www.trufflesuite.com/docs/ganache/overview*. https://www.trufflesuite.com/docs/ganache/overview. Erişim Tarihi: 03.02.2021.

[72] “Web3js,” *https://web3js.readthedocs.io/en/v1.3.0/*. https://web3js.readthedocs.io/en/v1.3.0/. Erişim Tarihi: 01.02.2021.

[73] “Metamask Wiki,” *https://en.wikipedia.org/wiki/MetaMask*. https://en.wikipedia.org/wiki/MetaMask. Erişim Tarihi: 01.02.2021.

[74] “Spring Boot,” *https://spring.io/projects/spring-boot*. https://spring.io/projects/spring-boot. Erişim Tarihi: 01.02.2021.

[75] “IPFS,” *https://ipfs.io/*. https://ipfs.io/. Erişim Tarihi: 01.02.2021.

ÖZGEÇMİŞ

Adı Soyadı **:** Emre Karakoç

ÖĞRENİM DURUMU

Derece	Eğitim Birimi	Mezuniyet Yılı
Doktora	Sakarya Üniversitesi / Fen Bilimleri Enstitüsü / Bilgisayar ve Bilişim Mühendisliği	2021
Yüksek Lisans	Sakarya Üniversitesi / Fen Bilimleri Enstitüsü / Bilgisayar ve Bilişim Mühendisliği	2016
Lisans	Sakarya Üniversitesi / Mühendislik Fakültesi / Bilgisayar Mühendisliği	2012

İŞ DENEYİMİ

Yıl	Yer	Görev
2018-Halen	Kamu	Yazılım Uzmanı
2012-2018	Özel Sektör	Yazılım Uzmanı

YABANCI DİL

İngilizce

ESERLER (makale, bildiri, proje vb.)

1. Karakoç E., Çeken C., "Black Hole Attack Prevent Scheme using Blockchain Block Approach in SDN-Enabled WSN," Int. J. Ad Hoc Ubiquitous Comput., vol. 37, no. 1, pp. 37–49, 2021.