MARMARA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ARAÇSAL AĞLAR İÇİN BLOKZİNCİR VE AKÜMÜLATÖR TABANLI SERTİFİKA YÖNETİM MEKANİZMASI

AYLİN AYDIN

YÜKSEK LİSANS TEZİ
Bilgi Güvenliği
Mühendisliği Programı

DANIŞMAN
Dr.Öğr.Üyesi Hüseyin YÜCE

EŞ-DANIŞMAN
Dr. Muhammed Ali BİNGÖL

İSTANBUL, 2021

Teşekkür

Yüksek lisans tez sürecimde bana yardımcı olan, değerli bilgi ve tecrübeleriyle bana yol gösteren tez danışmanım Dr. Öğr. Üyesi Hüseyin Yüce'ye ve eş danışmanım Dr. Muhammed Ali Bingöl'e teşekkürlerimi sunarım.

Her zaman olduğu gibi tez sürecimde de yanımda olup bana destek veren aileme ve arkadaşlarıma sonsuz teşekkür ederim.

Aynı zamanda yüksek lisans yapma imkanı sağlayan ve çalışmamı tamamlamama izin veren TÜBİTAK BİLGEM'e teşekkür ediyorum.

Araçsal Ağlar için Blokzincir ve Akümülatör Tabanlı Sertifika Yönetim Mekanizması

Özet

Akıllı ulaşım sistemlerinin temeli olan Araçsal Ağlar veya Tasarsız Araçsal Ağların (Vehicular Ad Hoc Networks - VANETs) amacı yol güvenliğini artırmak ve trafik akışının verimliliğini iyileştirmektir. Araçtan araca ya da araçtan altyapıya bağlantı kurarak zamanında ve doğru iletişim sağlamaya çalışır. VANET sistemleri, yol güvenliğini ve insan hayatını tehdit edebilecek kötü amaçlı araçların veri manipülasyonuna karşı savunmasızdır. Bu yüzden en temel gereksinim olan kimlik denetimi kontrolü ilkesi için kullanılan sertifikaların iptal durumunun kontrolü, doğrulama işlemi için en önemli gerekliliklerden biridir. Değişmez, erişilebilir ve güvenilir yapısı nedeniyle, şeffaf ve güvenli bir sertifika iptal mekanizması sağlamak için yakın zamanda blokzincir tabanlı çözümler önerilmiştir. Bu çalışmalar çoğunlukla bir sertifikanın iptal edilmediğini kanıtlamak için sıralanmış Merkle ağacının kullanılmasını önermektedir. Ancak, bu yaklaşım depolama, doğrulama ve güncelleme maliyetleri açısından verimli değildir. Hem geleneksel SİL hem de sıralı Merkle ağacında yaşanılan problemlerden dolayı, VANET'de RSA tabanlı dinamik evrensel akümülatör kullanılarak iptal edilen sertifikaların tek bir kısa değerde toplanıp blokzincirde tutulmasıyla ilgili verimli bir sertifika iptal mekanizması öneriyoruz. Böylelikle, bir araç sabit boyutlu üye olmayan kanıtlar kullanarak diğer üyenin bilgilerini açığa çıkarmadan sertifika durumunun geçerliliğini verimli bir şekilde kanıtlar ve sistemdeki ek yük önemli ölçüde azalır.

Anahtar Sözcükler: Blokzincir, Araçsal Ağlar, Kriptografik Akümülatör, Sertifika durum bilgisi

Blockchain and Accumulator Based Certificate Management Mechanism for Vehicular Ad Hoc Networks

Abstract

Vehicular Ad Hoc Network (VANET) is considered a substantial approach for intelligent transportation systems that promises safe driving, efficient traffic flow and real-time information exchange between vehicles and infrastructure. VANET systems are vulnerable to data manipulation by malicious vehicles which could threaten the road safety and human life. Therefore, certificate revocation status checking is one of the most important processes for verification of user authentication that should be performed securely and efficiently. Due to its immutable, accessible and reliable structure, blockchain based solutions are recently proposed to provide a transparent and secure certificate revocation mechanism. These works are mostly based on sorted Merkle tree for exclusion proof of revoked certificate in the blockchain. However, this approach is not efficient in terms of storage, validation and updating costs. We propose a blockchain and accumulator based certificate revocation mechanism to enhance sorted Merkle tree and traditional certificate revocation list (CRL) approaches. In this work, revoked certificates are accumulated in the blockchain as a succinct value by utilizing the dynamic universal accumulator and the certificate is validated by non-member witnesses. A vehicle efficiently proves the validity of certificate status by using short fixed-size non-membership proofs without violating the other member's privacy, thus the overall overhead is reduced significantly.

Key Words: Blockchain, Vehicular Ad Hoc Network, Cryptographic Accumulator, Revocation

İçindekiler

Teşekkür i

Özet ii

Abstract iii

Şekil Listesi vi

Tablo Listesi vii

Kısaltmalar viii

1 Giriş 1
- 1.1 Çalışmanın Katkıları . . . 4
- 1.2 Organizasyon . . . 5

2 Literatür Araştırması 7

3 Ön Çalışmalar 14
- 3.1 Blokzincir Teknolojisi . . . 14
  - 3.1.1 Blokzincir Çalışma Prensibi ve Mimarisi . . . 16
    - 3.1.1.1 Temel Bileşenleri . . . 18
    - 3.1.1.2 Uzlaşma Yöntemleri . . . 20
      - İş Kanıtı (Proof of Work - PoW) . . . 20
      - Varlık Kanıtı (Proof of Stake - PoS) . . . 21
      - Otorite Kanıtı (Proof of Authority - PoA) . . . 21
  - 3.1.2 Blokzincir Ağ Türleri . . . 22
  - 3.1.3 Blokzincir Teknolojisinin Kullanım Alanları . . . 24
- 3.2 VANET Teknolojisi . . . 25
  - 3.2.1 VANET Mimarisi . . . 26
    - 3.2.1.1 VANET Bileşenleri . . . 27
    - 3.2.1.2 VANET İletişim Tipleri . . . 28
    - 3.2.1.3 Kullanılan Kablosuz Teknolojiler . . . 29
  - 3.2.2 VANET'in Karakteristikleri . . . 30
  - 3.2.3 VANET'in Güvenlik Gereksinimleri . . . 31
  - 3.2.4 Yönlendirme Protokolleri . . . 32
  - 3.2.5 VANET Uygulamaları . . . 32
    - 3.2.5.1 Güvenlik Uygulamaları . . . 33
    - 3.2.5.2 Kullanıcı Uygulamaları . . . 34

3.3 Kriptografik Akümülatörler 34
3.3.1 Akümülatör Yapısının Gelişimi ve Türleri 35
3.3.2 Evrensel Akümülatör 37
3.3.3 Taahhüt Şeması ile Evrensel Akümülatör 40
3.3.4 Uygulama Alanları 41

4 Sistem Tasarımı ve Bulgular **42**
4.1 Önerilen Sistem 42
4.2 Sistem Modeli 43
4.3 RSA Tabanlı Dinamik Evrensel Akümülatörün Uyarlanması 45
4.4 Performans Değerlendirmesi 49

5 Sonuçlar **53**

A Java Kodu **56**

Kaynaklar **61**

Şekil Listesi

3.1 Blokzincirin yıllar içerisindeki gelişimi. 15
3.2 Merkezi, merkezi olmayan, dağıtık ağ yapısı [1]. 17
3.3 Blokzincir mimarisi [2]. 18
3.4 Blokların zincirlenmiş hali ve blok başlıkları [3]. 19
3.5 Merkle ağacı örneği. 20
3.6 Blokzincirde çatallanma örneği [4]. 21
3.7 Açık İzinsiz, Açık İzinli, Konsorsiyum (Özel İzinli) ve Özel izinsiz blokzincir çeşitleri [5]. 23
3.8 Tasarsız ağların sınıflandırılması [6]. 25
3.9 VANET mimarisi [7]. 27
3.10 Sistem bileşenleri [7]. 28
3.11 Araçsal ağların iletişim örneği [8]. 29
3.12 VANET uygulamalarının sınıflandırılması. 33
3.13 Sıralı merkle ağacı. 37
3.14 Akümülatör kullanım alanları örnekleri. 41

4.1 Önerilen sistem için blok diyagramı. 43
4.2 Önerilen sistem modeli. 44
4.3 Üye olmayan kanıtların boyutunun iptal edilen sertifika sayısına göre karşılaştırılması. 51

Tablo Listesi

3.1 Açık blokzincir, özel blokzincir ve konsorsiyum blokzincir çeşitlerinin kıyaslanması [4]. 22
3.2 Blokzincir teknolojisinin sınflandırılması [5]. 23

4.1 RSA tabanlı dinamik evrensel akümülatör algoritmalarında kullanılan gösterim. 45
4.2 VANET'de sertifika iptal durumu kontrolü için önerdiğimiz akümülatör yaklaşımının uyarlanması. 46
4.3 SİL, Sıralı Merkle ağacı ve RSA akümülatörün kıyaslanması. 52

Kısaltmalar

ITS	**I**ntelligent **T**ransportation **S**ystem (Akıllı Ulaşım Sistemleri)
VANET	**V**ehicular **A**d **H**oc **N**etwork (Araçsal Ağ - Tasarsız Araçsal Ağ)
MANET	**M**obile **A**d **H**oc **N**etwork (Mobil Tasarsız Ağ)
V2V	**V**ehicle **t**o **V**ehicle (Araçtan Araca)
V2I	**V**ehicle **t**o **I**nfrastructure (Araçtan Altyapıya)
DSRC	**D**edicated **S**hort **R**ange **C**ommunications (Özel Kısa Menzilli İletişim)
RSU	**R**oad **S**ide **U**nit (Yol Kenarı Ünitesi)
OBU	**O**n **B**oard **U**nit (Yerleşik Ünite)
AU	**A**pplication **U**nit (Uygulama Ünitesi)
AAA	**A**çık **A**nahtar **A**ltyapısı
SM	**S**ertifika **M**akamı
SİL	**S**ertifika **İ**ptal **L**istesi
OCSP	**O**nline **C**ertificate **S**tatus **P**rotocol (Çevrimiçi Sertifika Durum Protokolü)
LMT	**L**exicographic **M**erkle **T**ree (Sözlükbilimsel Merkle Ağacı)
LEA	**L**aw **E**nforcement **A**uthority (Emniyet Otoritesi)
BST	**B**inary **S**earch **T**ree (İkili Arama Ağacı)
P2P	**P**eer **t**o **P**eer (Eşler Arası)
IoT	**I**nternet **o**f **T**hings (Nesnelerin İnterneti)
PoW	**P**roof **o**f **W**ork (İş Kanıtı)
PoS	**P**roof **o**f **S**take (Varlık Kanıtı)
PoA	**P**roof **o**f **A**uthority (Otorite Kanıtı)
GPS	**G**lobal **P**ositioning **S**ystem (Küresel Konumlama Sistemi)
NES	**N**itelikli **E**lektronik **S**ertifika
PDA	**P**ersonal **D**igital **A**ssistant (Kişisel Dijital Asistan)
RFID	**R**adio **F**requency **ID**entification (Radyo Frekansı ile Tanımlama)

Bölüm 1

Giriş

Ulaşım insanlık tarihi açısından önemli bir ihtiyaç olmuştur. Zaman içerisinde insanlık ihtiyaçlarının değişimiyle ulaşım araçları gelişmiştir. İnsanlar için ulaşımda çeşitli ve çok sayıda araç kullanımı kolaylık oluştururken, bu olumlu durum giderek önemli bir problem haline gelmiştir. İnsan nüfusunun ve buna bağlı olarak araç kullanımının artması ile bu durum, trafik yoğunluğuna ve her geçen gün daha çok kazaya neden olmaktadır. Hem araçların, sürücülerin ve yayaların güvenliğini sağlamak hem de ulaşımın verimliliğini iyileştirmek için son yıllarda akıllı ulaşım sistemlerinin (Intelligent Transportation System - ITS) temeli olarak Araçsal Ağlar (Vehicular Ad Hoc Networks - VANETs) önerilmiştir [9].

Yaklaşık son yirmi yılda, kablosuz ağ teknolojilerinin gelişmesiyle araştırmacılar, iletişim ağlarındaki araçların iletişim sistemlerini tasarlamıştır. VANET olarak isimlendirilen tasarsız ağlar, mobil tasarsız ağları (MANETs) ilkesini uygulayarak ortaya çıkmıştır ve kablosuz mobil iletişim alanında bazı araştırmacılar için odak noktası olmuştur. VANET'de araçtan araca (V2V) ve araçtan altyapıya (V2I) iletişim olmak üzere iki çeşit haberleşme tipi bulunmaktadır. Birbirleri ile kendi aralarında kablosuz iletişim kurulabilmesi için Özel Kısa Menzil (Dedicated Short Range - DSRC) isimli iletişim protokolü kullanılır. Bu sayede araçlar arasında yol bilgisi (yol çalışma yerleri, hız limiti), uyarı mesajları (trafik yoğunluğu, trafik kazaları, acil durum senaryoları) gibi parametreler paylaşılarak,

araç ve yol durumları hakkında bilgi verilmiş olmaktadır. Bu yüzden VANET'de verimli, güvenli bir sistemin tasarlanmasının ve geliştirilmesinin, araba kazalarının engellenmesinde, yol güvenliğinin sağlanmasında ve yaşanılan trafik sıkışmalarının azaltılmasında önemli bir rolü bulunmaktadır. Bu kapsamda VANET'in hedefi, sürücülere zamanında ve sahte olmayan bilgileri sağlayarak yol güvenliğini artırmak ve trafik akışını daha da kolaylaştırarak verimliliğini iyileştirmektir. Aksi takdirde, iletilen sahte mesajlar sadece ulaşımdaki trafiğin akışını etkilemeyecek aynı zamanda insan hayatını tehdit ederek kazalara sebep olabilecektir [9–11].

İlk olarak Blokzincir teknolojisi, 2008 yılında Satoshi Nakomoto takma ismi tarafından "Bitcoin: Eşler Arası Elektronik Nakit Sistemi" isimli yayınlanan bir makale ile duyuruldu [12]. Verinin tek bir merkezde tutulması yerine dağıtık yapısı ile eş zamanlı erişim, doğrulama ve değiştirilemez veri kayıtları sağlayarak, blokzincirde bulunan bilgilerin şeffaflığı ve güvenliği sağlanmaktadır. Bu nedenle sahip olduğu özelliklerden dolayı blokzincirin popülerliği günümüzde artmaya devam etmiştir. Uzlaşma mekanizmalarıyla, ağdaki düğüm tarafından blok oluşturulur ve diğer düğümler tarafından doğrulanarak blokzincire eklenir. Blokzincir ağında yapılan değişiklikler, senkronize bir şekilde tüm düğümlere yansıtılır [13]. Bununla birlikte geleneksel merkeziyetçi yaklaşım nedeniyle, araç ağları arasında yapılan bilgi alışverişinin doğruluğu ve bütünlüğü hakkında güvenlik endişeleri bulunmaktadır. Ayrıca merkezi olan güven yönetiminde, araçlar gönderilen verileri oylayarak bir güven değeri oluştururlar ancak doğrulama sürecinde güven tam olarak sağlanamaz. Bu nedenle kimlik denetiminin sağlanması, mesaj bütünlüğünün korunması, inkar edemezlik ilkeleri sistemin temel gereksinimlerindendir. Verilerin takip edilebilmesi ve mesaj bütünlüğünün korunabilmesi için blokzincir dağıtık defter teknolojisinin erişilebilirlik, değişmezlik, güvenlik, esneklik gibi özelliklerinden faydalanılmaktadır. VANET'de hareket halinde olan araçlar ağda birer düğüm olarak kullanılmaktadır [14]. Blokzincirde işlemlerin güvenli bir şekilde gerçekleşmesi için kullanılan açık anahtar kriptolojisi ile hesap adresleri açık anahtar kullanılarak oluşturulup, yapılan işlemler özel anahtarla imzalanır. Akıllı araçlarda da bu güvenlik gereksinimlerini karşılayabilmek için açık anahtar alt yapısı teknolojisi kullanılır [15].

Yaygın olarak kullanılan açık anahtar altyapısı (AAA) ile güvenlik alt yapısını sağlamak için her aracın açık/özel anahtar çifti bulunur. AAA teknolojisi: Dijital sertifikaların oluşturulması, yönetilmesi, iptal edilebilmesi için gerekli ilkelerden meydana gelmektedir. Bu sistemde temel olan işlemler, sertifikalandırma ve doğrulamadır. Sertifikalar

bir varlığı tanımlar ve bu kimliği açık anahtarla ilişkilendirerek, açık anahtarın sahibini kanıtlar. Sertifikaların geçerlilik süreleri bulunmaktadır ve sertifikayı veren sertifika makamının (SM) dijital imzasını bulundurur. Sertifikaları üretmek, gerektiğinde iptal etmek veya yenilemek SM' nin görevlerindendir. Diğer bir işlem de sertifikaların geçerliliğinin doğrulanmasıdır. Bazı belirtilen sebeplerden dolayı elektronik sertifikalar, sertifika otoriteleri tarafından iptal edilebilmektedir. Sertifika sahibinin sertifikada bulunan bilgilerinin değişmesi, özel anahtarının çalınması/kaybedilmesi/ifşa olması, sertifika geçerlilik süresinin sona ermesi gibi çeşitli nedenlerle sertifikalar iptal edilebilir. Sertifika otoriteleri tarafından belirli aralıklarla yayınlanan sertifika iptal listesi (SİL) ve/veya çevrimiçi sertifika durum protokolü (OCSP) istek mesajları ile anlık sorgulanarak sertifikaların iptal durumunda olup olmadığı öğrenilmektedir. Belirli periyotlarla çevrimdışı olarak üretilen SİL dosyalarına göre, çevrimiçi OCSP sunucusu ile bilgiler daha güncel elde edilmektedir ancak öte yandan OCSP sunucusunun kimlik ispatlamak için tüm OCSP cevaplarını imzalaması ile sunucunun işlem yükü artmaktadır. SİL dosyaları da elektronik imzalıdır ve iptal edilen sertifikaların seri numaralarını içermektedir. Eğer SİL içerisinde sertifika seri numarası varsa sertifikanın iptal edilip geçersiz olduğunu yoksa sertifikanın geçerli olduğunu göstermektedir. Ancak belirtilen nedenler sonucunda iptal edilen sertifikalar, SİL dosyasının boyutunu artırmaktadır ve bellekte büyük boyutta yer tutmasına neden olmaktadır. Ayrıca ihtiyaç olmadığı halde iptal edilen diğer sertifika seri numaraları da görünmüş olmaktadır. Dolayısıyla mimari olarak zayıf kabul edilen bu yaklaşım, VANET gibi büyük ölçekli ağ için uygun bulunmamaktadır [15–17].

Bu nedenle çok fazla yüke neden olmadan, VANET için esas bir zorunluluk olan sertifika doğrulama sürecinde kullanılan sertifikaların, verimli bir şekilde geçerli olup olmadığının kontrolünün sağlanması için blokzincir ve akümülatör tabanlı verimli çalışan bir sertifika iptal şeması tasarlamayı amaçlamaktayız. Bu çalışmada, SM tarafından bilgileri doğrulanan her araca sertifika dağıtılır ve RSA tabanlı dinamik evrensel akümülatör özellikleri kullanılarak VANET sisteminde iptal edilen sertifikaların seri numaralarının bir değer kümesini temsil eden sabit boyutlu bir özet değeri şeklinde blokzincirde tutulması önerilmektedir. İlk olarak kriptografik akümülatörler 1993 yılında Josh Benaloh ve Michael de Mare tarafından tanıtıldı [18]. Tek yönlü fonksiyon olma özelliğine sahip olan özet fonksiyonları gibi benzer nitelikte olup, yarı değişmeli (quasi commutative) olarak isimlendirilen ek bir özelliğe sahip olarak tanımlanmıştır. Daha sonra, Camenisch ve Lysyanskaya tarafından verimli bir şekilde akümülatöre eleman ekleme ya da silme

yapmayı sağlayan dinamik akümülatör sunuldu [19]. Ancak var olan bu metotla, bir değerin sadece belirtilen kümenin üyesi olduğu kanıtlanmaktadır. Bu yüzden Li, Li ve Xue tarafından dinamik evrensel akümülatör öne sürülerek, verimli bir biçimde hem üye hem de ayrıca üye olmayan kanıtların oluşturulması sağlandı [20]. Kısacası dinamik evrensel akümülatör, bir değerin verilen bir küme setinde üye olup olmadığını kanıtlayan, alan ve zaman açısından verimli olan güvenli bir veri yapısıdır. Üye/üye olmayan tanık ile sahip olunan akümülatör değeri verimli bir şekilde doğrulanmaktadır. Tüm VANET yapısını değiştirmeden temel olarak önerdiğimiz sertifika iptal şeması, güçlü RSA güvenlik varsayımı altında kabul edilerek dinamik evrensel akümülatör yardımıyla geçersiz sertifika seri numaralarını içeren eleman setinin toplanmasına izin veren sabit uzunlukta bir akümülatör değeri sağlayıp blokzincirde tutularak, sertifika iptali mekanizmasında hafif bir çözüm önerisi olarak VANET'de uygulanabilir [21, 22].

1.1 Çalışmanın Katkıları

VANET teknolojisinde, ağdaki araçların arasında bağlantı kurulmadan ve veri alışverişi yapılmadan önce sistemin güvenilirliği açısından, kimlik doğrulama işleminin yapılması büyük bir önem taşımaktadır. Bu doğrultuda, SM tarafından araçlara verilen sertifikaların denetlenmesi gerekmektedir. Dolayısıyla, sertifika doğrulamanın bir ihtiyaç olduğu VANET sistemlerinde, sertifikaların iptal edilip edilmediğinin bilgisine güvenli, rahat ve hızlı bir şekilde erişilmesi gerekmektedir. Blokzincir sisteminin beraberinde getirdiği kolaylık ve güvenilirlikle birlikte erişilebilirlik, veri bütünlüğü ve güven duygusu sağlanmaktadır. Doğrulama, depolama açısından da daha ucuz bir maliyetin sağlanması ve verimliliğin artırılması, RSA tabanlı dinamik evrensel akümülatör fonksiyonu ile gerçekleştirilmektedir.

Bu nedenle, VANET için yapılan bu çalışmada iptal edilen sertifikaların dinamik evrensel akümülatör ile seri numaralarını simgeleyen sabit uzunlukta bir akümülatör değeri elde edilip, bu akümülatör değerinin blokzincirde tutulmasıyla ilgili bir sertifika iptal şeması önerilmektedir. Çalışma özetle aşağıdaki katkılara sahiptir:

- *Verimli üye olmadığını kanıtlama:* İlk olarak VANET sisteminde sertifikanın iptal durumunu öğrenmek için kullanılan yöntemlere alternatif olarak blokzincir yapısında

RSA tabanlı dinamik evrensel akümülatör fonksiyonunun özelliklerinden yararlanılarak, sertifikanın iptal edilmediği kanıtlanabilecektir. Yani, alternatif bir çözüm yolu olarak dinamik evrensel akümülatör metodunun katkısı ile üye olmayan tanık sunularak, iptal listesinde sertifika seri numarasının olmadığı ispatlanabilecektir. Böylelikle, iptal edilen sertifika seri numaralarından oluşan kümenin tüm elemanları, tek bir kısa değerde toplanarak bu akümülatör değerinin blokzincire yazılmasıyla VANET'de sertifika doğrulama adımı sağlanmış olacaktır.

– *Verimli depolama alanı:* Dinamik evrensel akümülatörün, sabit boyutlu dahil/-dahil etmeme kanıtları olduğu için verimli bir şekilde dinamik olarak ekleme/silme yapabilirler ve üye/üye olmayan kanıtlar sunabilmektedir. Bu özelliği sayesinde depolama yükü azaltılmış olur.

– *Verimli güncelleme:* Ayrıca tek yönlü akümülatörlerin yarı değişmeli özelliği sayesinde, değer kümesinde bulunan iptal edilen sertifikaların seri numaralarının sırası önemli değildir. Dolayısıyla bu değer kümesine ekleme/silme işlemi yapılmasının maliyeti sabit olup, değer kümesinin sıralanışına ve boyutuna bağlı olmayacaktır. Böylelikle büyük ölçekli bir ağ olan VANET için bazı nedenlerden dolayı iptal edilen sertifikaların kümeye eklenmesinde verimlilik sağlanıp korunmuş olacaktır.

– *Güvenlik:* Güçlü RSA varsayımı olduğu baz alınarak oluşturulan dinamik evrensel akümülatör ile güvenli bir VANET yapısı sağlanmış olacaktır. Elde edilen akümülatör değerinin blokzincirde değiştirilemez bir şekilde şeffaf bir kayıt olarak tutulması ile beraber de veri bütünlüğü korunacağından VANET teknolojisi için güvenilir bir ortam sunulmuş olacaktır. Böylece hem güncel hem de geçmişe yönelik bir tutarsızlığa da müsaade edilmemiş olacaktır.

– *Erişilebilirlik:* VANET'de araçlar birbirlerini doğrulamak istediklerinde, blokzincirin dağıtık defter teknolojisi yapısına sahip olmasından dolayı her zaman yüksek erişilebilirlik imkanı bulunmaktadır ve sertifikalarının iptal edilmediğini kanıtlayabilirler. Geçmişe yönelik kayıtlara da erişilebilmektedir.

1.2 Organizasyon

Tezin bundan sonraki bölümleri aşağıdaki gibi düzenlenmiştir. İlk olarak VANET teknolojisinde iptal mekanizmasıyla ilgili ilişkili çalışmalardan bahsedilmektedir. Bölüm 3'te tez

kapsamında yapılan ön çalışmalardan bahsedilmektedir. Blokzincir teknolojisi, VANET teknolojisi ile kriptografik akümülatörlerden bahsedilerek VANET için önerdiğimiz yapı tanımlanmıştır. Bölüm 4'de VANET'de kullanılmasını önerdiğimiz sertifika iptal şemasından bahsedilmektedir. Ayrıca geleneksel SİL, Merkle ağacı ve önerdiğimiz akümülatör modeli arasında performans değerlendirmesi yapılmaktadır. Son olarak, Bölüm 5'de çalışmamızın sonucundan ve gelecek çalışmalardan bahsedilerek sonlandırılmaktadır.

Bölüm 2

Literatür Araştırması

Bu bölümde VANET teknolojisi için tasarlanan kimlik doğrulama ve iptal mekanizması ile ilgili mevcutta var olan çalışmalardan bahsedilmektedir.

Zhenglin Liu ve diğerleri [9] tarafından önerilen VANET için blokzincir tabanlı mahremiyeti koruyan güven modeli çalışmasında, verimli kimlik doğrulama mekanizması için sözlükbilimsel (lexicographic) bir Merkle ağacının (LMT) özelliklerinden faydalanılmaktadır. Yaşanılan bir anlaşmazlık durumu karşısında müdahale edilebilmesi için gerçek kimliği doğrulanan araç sahiplerine SM tarafından sertifika verme ya da iptal etme iznini veren, araçların davranışlarını kontrol eden emniyet otoritesi (Law Enforcement Authority - LEA) tarafından gerçek kimlik sahibi bilgisi ile ilişkili açık anahtarlar kayda geçirilmektedir. Ancak SM tarafından bir araca sertifika verildiğinde ya da sertifikası iptal edildiğinde, aracın gerçek kimlik sahibi ile ilişkili olan açık anahtarı arasındaki bağ ortadan kaldırılarak araca ait bilgiler açığa çıkarılmadan açık anahtarlar takma ad (pseudonym) olarak sistemde kullanılarak kimlik doğrulama sürecinde kullanılmaktadır. Tasarımda sunulan sıralı Merkle ağacı veri yapısı kullanılarak iptal edilen bu açık anahtarlar blokzincirde tutulmaktadır. Merkle ağacı ya da hash ağacı olarak bilinen veri yapısı, verilerin güvenli bir şekilde doğrulanmasını sağlar ve veri bütünlüğünü korur. Özet fonksiyonlarını kullanarak her yaprak olmayan düğüm (non-leaf node) alt düğümlerinin (child node) kriptografik özet değerini bulundurmaktadır. Sıralanmış bir Merkle ağacında da ikili bir arama ağacı (Binary Search Tree - BST) yapısının tek bir düğümünde alfabetik, sayısal gibi sıralar halinde olarak yeniden gruplandırılmaktadır. Logaritmik zaman ve alanda (OlogN), sıralı Merkle ağacı kullanılarak açık anahtarın

olmadığı ve dolayısıyla iptal edilmediği kanıtlanmaktadır. Bu da doğrulamanın o kadar da ucuz olmadığını göstermektedir. Öte yandan iptal edilen açık anahtarlar sıralı bir şekilde Merkle ağacı veri yapısında tutulduğundan, her yeni iptal edilen açık anahtarlar için sürekli sıra değişecek ve kök (root) de değişecektir. Ağacı yeniden oluşturmadan güncelleme yapılamayacaktır. Bu yüzden hem iptal edilen açık anahtarların sayısına ve sırasına bağlı kalacaktır hem de üye olmayan kanıtlar kolay bir şekilde kümelenemeyecektir. Buna ek olarak, açık anahtarın iptal edilmediğinin kanıtlanması için bazı yaprak düğümlerin (leaf node) açığa çıkması gerekmektedir çünkü iptal edilen açık anahtarlar sıralanmış bir şekilde dizildiği için hangi aralıkta olduğunun belirlenmesi gerekir ve ardışık iki düğümün yan yana bulunduğu gösterilerek bu aralıkta olmadığının yani iptal edilmediğinin kanıtlanması gerekmektedir.

Zhenglin Liu ve diğerleri [23] tarafından önerilen VANET'ler için blokzincir tabanlı gizlilik korumalı kimlik doğrulama olarak sunulan bir başka çalışmada da geleneksel blokzincir yaklaşımını genişletmek için Merkle Patricia ağacı isimli veri yapısı benimsenmiştir. Alan ve hesaplama bakımından elverişli bulunmayan SİL yerine kullanılması için tasarlanmıştır. SM tarafından yayınlanan veya iptal edilen sertifikalarla ilgili işlemler blokzincirde kronolojik Merkle ağacında tutularak şeffaflık sağlanmaktadır. Merkle Patricia ağacı da Ethereum için kriptografik olarak doğrulanmış bir veri yapısı sağlayan Patricia ve Merkle ağacının bir kombinasyonudur. Merkle Patricia ağacının yaprak düğümü, sertifika ve kimliğiyle ilişkili şifrelenmiş bağlantıyı, açık anahtarını içermektedir. Açık anahtarın özeti yaprak düğümünün anahtarıdır, sertifikası da değerine karşılık gelerek anahtar-değer çifti oluşturulmaktadır. Sertifika Merkle Patricia ağacında bulunuyorsa ve imzası doğrulanmışsa açık anahtarının geçerli olduğunu göstermektedir yani iptal edilmediği kanıtlanmaktadır. Sertifika iptali durumunda SM açık anahtarla ilgili yaprak düğümünü siler. Arama, ekleme, silme işlemi logaritmik maliyete sahiptir.

Adnan Shahid Khan ve diğerleri [11] tarafından önerilen VANET'teki saldırıları önlemek için güven tabanlı blokzincir mimarisinin kullanılmasını öne sürdüğü çalışmada da iptal edilen açık anahtarlar blokzincirde bulunmaktadır. Öncelikle sertifikanın geçerlilik süresi kontrol edilir, sertifika süresi dolmamışsa daha sonra sertifikanın blokzincirde bulunup bulunmadığına ve iptal edilen sertifikalar için de ayrı ve sıralı olarak tutulan blokzincirdeki Merkle ağaç yapısına bakılarak doğrulama işlemi gerçekleştirilir. Blokzincirde kontrol edilerek açık anahtarının iptal edilmediği kanıtlanarak diğer araca açık anahtarının geçerli olduğu sunulmaktadır. Ancak bu tasarımda tavsiye edilen iptal mekanizmasında

da Merkle ağacının bahsedilen dezavantajlarından dolayı hem doğrulama ve güncelleme işlemleri verimsiz olacaktır hem de geçerliliğinin kanıtlanmasına ihtiyaç duyulmadığı halde bazı düğümlerin ifşa olması gerekecektir.

Arwa Alrawais ve diğerleri [15] tarafından önerilen VANET için tasarlanan verimli bir iptal şeması çalışmasında, sis bilişim (fog computing) mimarisi ile ağdaki bant genişliğinin tüketimini azaltmayı amaçlayan ve sertifika iptal bilgileri için Merkle ağaç veri yapısını kullanan bir yaklaşım sunulmuştur. Sis bilişim [24] mimarisinin; gecikme süresinin azaltılması, yaygın coğrafi dağılımın sağlanması, konumlandırılması, çok fazla sayıda düğümün bulunması, heterojenlik gibi özellikleri bulunmaktadır ve bulut bilişim modelini genişleterek ağın en uç noktasına kadar yakınlaştırır. Bu çalışmada da sis bilişimin bu özelliklerinden faydalanılarak yollar coğrafi bölgelere ayrılmıştır. Sis teknolojinin esasında sis düğümleri bulunmaktadır. Güvenli bir ağ geçidi gibi davranan ve coğrafi olarak dağınık olan sis düğümleri aracılığıyla bölge dahilinde olan araçlara yeni üretilen sertifikalar iletilir ya da iptal bilgileri paylaşılır. SM, iptal edilen sertifikalar için imzalanmış bir şekilde Merkle ağaç yapısı oluşturur. Ağacın her bir yaprağında iptal edilen sertifikaların seri numaraları bulunmaktadır. Araçların sertifikasının geçerli olup olmadığının kanıtlanması için öncelikle kökün imzasının SM'nin açık anahtarı ile doğrulanması gerekmektedir. Daha sonra iptal edilen sertifikanın seri numarasının ve ağacın diğer düğümlerinin özet değeri hesaplanır ve kök ile aynı olup olmadığı kontrol edilir. Kontrol edilen sonuç doğru ise sertifikanın iptal edilmiş olduğunu göstermektedir. Merkle ağaçlarda doğrulama yolu logaritmiktir (OlogN) ve verimlilik ağaç boyuna bağlı bulunmaktadır ancak bu çalışmada yolların coğrafi bölgelere ayrılması sebebi ile sis (fog) içerisinde bulunan araçların sayısı kısıtlı olduğu için ağacın boyunun etkisinin azaltıldığı düşünülmektedir. Her ne kadar bu durum biraz daha avantajlı bir yöntem olsa da doğrulama maliyeti yine de çok da ucuz olmayacaktır. Merkle ağacının sahip olduğu dezavantajlarından bahsedildiği üzere güncelleme işlemi de verimsiz görünmektedir ve doğrulama yolu boyunca bazı düğümlerin açığa çıkması gerekmektedir. Sis teknolojisinin beraberinde güvenlik ve gizlilik ile ilgili gelen sorunlardan ve iyileştirmelerden kısaca ilişkili çalışmalarda bahsedilse de tam olarak bu konuya değinilmemiştir.

Carlos Ganan ve diğerleri [25] tarafından önerilen VANET için mahremiyet koruma sağlayan iptal mekanizması çalışmasında, evrensel tek yönlü akümülatör kullanımı önerilmektedir. SM, iptal bilgilerini akümülatör kullanarak tek bir değerde sunar ve daha sonra herhangi bir araç sertifikasındaki benzersiz değere tanıklık ederek sertifikasının

durumunun geçersiz olmadığını kanıtlayabilir. SM bu verileri güvenli bir kablo hattı ile RSU'ya gönderir. Kişisel bir bilgi paylaşılmadan geçerliliği kontrol edilmesi gereken araç, yollara yerleştirilen RSU ile temasa geçer. Tanık değeri RSU'dan istenilmektedir ve güncelleme işlemi de RSU tarafından yapılmaktadır. Sertifikanın durumunun geçerliliğinin denetlenmesi için hesaplanan akümülatör değeri ve karşılık gelen tanık değeri herhangi bir mobil depodan (repository) indirilebilir. Bu mobil depolar, RSU'nun sınırları dahilinde irtibat sağlanarak akümülatörün güncellenmiş bir kopyası ile tanık değerini güncellemek için ihtiyaç olan yardımcı (auxiliary) bilgilerin indirilmesinde görevlidir. Kümeleme (clustering) algoritması kullanılarak araçlara mobil bir depo gibi olmasına müsaade edilir. Ortaya konulan bu iptal mekanizmasındaki bahsedilen evrensel tek yönlü akümülatörün yaklaşımında, Nguyen'nin çalışması [26] baz alınarak eliptik eğriler üzerindeki bilinear çiftleri kullanılır çünkü bilinear-map akümülatörü [27] neredeyse birçok durumda performansının diğer akümülatörlere göre daha verimli olabileceği düşünülmektedir. Üye olmayan doğrulamanın güvenliği için q-strong Diffie-Hellman varsayımı kabul edilmiştir. Güvenilir üçüncü bir taraf aracılığı ile akümülatör yönetiminin sağlanması bu çalışmada yeterli görüldüğünden dinamik evrensel akümülatör yapısı için zayıf bir akümülatör tercih edilmiştir. Güvenlik mimarisi için ise AAA mimari çeşitlerinden biri olan birleşik (mesh) mimari modeli esas alınarak çapraz bir güven ağı kurulur ve eşler arası (P2P) SM ilişkisi bulunmaktadır. Böylece, her kullanıcı güveneceği makamı kendi seçebilir ve tek nokta hatasını ortadan kaldırmaktadır. Verimli bir çalışmadır ancak birleşik mimari [28], sisteme yeni eklenen kullanıcıları kolay bir şekilde kapsamaya izin verse de yönetim maliyeti yüksektir ve VANET gibi büyük ölçekli bir ağda kurulan güvenin sayısı, SM' lerin sayısı ile doğru orantılı bir şekilde arttığından dolayı ölçeklenebilirlik problemleri getirebilecektir. Ayrıca yapılan bu çalışmada erişilebilirliğin, şeffaflığın sağlanması ve bilinear-map akümülatörünün diğer akümülatörlere göre gerçek hayatta bir uygulamaya implemente edilebilmesi daha külfetli görünmektedir.

Carlos Ganan ve diğerleri [29] tarafından VANET için verimli ve gizlilik bilinçli bir iptal mekanizması (EPA) olarak sunulan bir başka çalışmada da araçların yüksek boyutlu iptal listelerini indirmek yerine sertifikanın iptal edilmediğini kanıtlamak için tek yönlü özet fonksiyonu ile oluşturulan Merkle ağaçının kullanılmasından bahsedilmektedir. Bahsedildiği üzere yaprak düğümlerinde verilerin özet değeri bulunmaktadır ve bu özet değerleri birleştirilerek hesaplanan özet değeri kökte tutulmaktadır. SM genişletilmiş SİL olarak da adlandırılan sertifika iptal listesini oluşturur ve ağacın kök değerinin özet değeri

hesaplanır. RSU'lara güvenli bir şekilde genişletilmiş SİL bilgisi ulaştırılarak Merkle ağacını yeniden oluşturabilirler. İptal edilmiş takma adlar kümesini ağaç veri yapısında bulundurarak RSU'lar ile iletişim kurularak sertifikanın iptal edilmediğini kanıtlayabilmektedir. Aynı zamanda her araç kendi sertifikasının iptal edilmediğini kanıtlamakla yükümlüdür. Çalışmada iptal edilen sertifikaları temsil eden Merkle ağacı yerine EPA ağacı olarak da adlandırılmaktadır ve bu EPA ağacında yapraklar sıralıdır. Ayrıca izlenebilirliği engellemek için anonimlik protokolleri uygulanmaktadır. Ancak daha önce de bahsedildiği üzere Merkle ağaç yapısının verimlilik açısından getirdiği bazı dezavantajlarından dolayı bu çalışma da etkili bulunamamıştır.

C. Brijilalruban ve diğerleri [30] tarafından VANET için küme tabanlı güvenli iletişim sağlanarak sunulan sertifika iptal şeması çalışmasında (SCCR), iptal edilmiş kimlik kümesi evrensel akümülatör kullanılarak tek bir değerde temsil edilmektedir. Akümülatör üye olmayan tanığı ve tanık düğümlerini kontrol eder. Üye olmayan tanık ile iptal edilen sertifika doğrulanabilmektedir. Kümenin herhangi üyesi de üye olmayan tanık kullanarak akümülatör değerinde birikmeyen sertifikanın kimliğini doğrulayabilmektedir. Sertifika doğrulama esnasında bilinear çiftleri kullanılmaktadır. SM bu bilgileri imzalayarak RSUlara iletir ve RSUlar da mobil depolar olarak da adlandırılan küme başlıklarına (Cluster Head- CH) aktarır. Araçlar kümeler halinde gruplandırılmaktadır ve bu kümeler için güvenilir düğümler baz alınarak bilgileri toplamayı sağlayan CH seçilir. Her CH'nin işlevinde, bir grup aktif düğümün akümülatör değerinde biriktirilen (accumulated) ölçüsünün ve iptal edilen düğümlerin kimliklerinin ölçüsünün saklanması bulunmaktadır. SM her akümülatör değerini güncellediğinde kopyasını saklayan CH'ye bildirilir. Küme üyelerinden herhangi birinin sertifikasıyla ilgili bir bilgiye gereksinim duyulursa bunu CH isteyebilir. CH'den alınan bilgi sonucu sonrası geçerli sertifika işaret edilirse, mesajların güvenli bir şekilde aktarılması, sertifikaların şifrelenmesi ve şifresinin çözülebileceği simetrik şifreleme algoritması kullanılarak sağlanabileceği önerilmektedir.

Khaled Rabieh ve diğerleri [31] tarafından önerilen VANET için takma ad (pseudonym) kullanılarak ölçeklenebilir bir iptal şeması (SPRV) olarak sunulan çalışmalarında sertifika iptali kontrolünün yapılması için sabit boyutlu bir trapdoor kullanılması öne sürülmektedir. İptal mekanizması ile birlikte grup imzası dahil herhangi bir tipte bulunan imzalar bu çalışmada desteklenmektedir. Takma ad üretilirken trapdoor özet algoritmaları olarak da bilinen chamelon özet fonksiyonları kullanılmaktadır. Chamelon özet fonksiyonları tek

yönlü özet fonksiyonu olup açık ve gizli anahtarla ilişkilidir. Çakışmaya dayanıklı (collision resistance) olan özet algoritmalarının trapdoor (gizli anahtar) bilgisi ile çakışmalar kolaylıkla hesaplanabilmektedir. Yani, chamelon özet fonksiyonları gizli bilgi olmadan çakışmaları bulmanın neredeyse imkansız olduğu tipik özet fonksiyonları gibidir. Dijital imzalarla birlikte kullanılarak da inkar edilemezliği sağlamaktadır. Ayrıca bir araç başka bir aracın yanlış bilgiler paylaşarak kötü niyetli bir saldırgan olduğundan şüphelendiği zaman o araç için iptal talebi oluşturabilmektedir. Bu iptal talebi kötü niyetli şüphelinin iptal edilecek takma adını, neden iptal edilmesi gerektiğini, talep eden aracın sertifikasını, imzasını ve zaman damgasını içermektedir. Herhangi bir araç yayınlanan bu iptal talebini doğrulayabilir ve iptal isteğini yapan aracın imzasını kontrol edebilir. Daha sonra herhangi bir araç bu şüpheli aracı iptal etmeyi kabul ederse ve iptal isteğinin doğru ve güvenilir olduğunu ikna edebilmek için çoklu imza kullanılarak iptal talebine eklenebilir. İptal talebinin imzalanması belirli bir miktara ulaştığında RSUlar aracılığıyla iptal otoritesine gönderilir ve doğrulanabilmektedir. Bu çalışmada da güven ilişkisini kurmak, erişilebilirliği, bütünlüğü sağlamak daha elverişsiz gözükmektedir. Oysaki blokzincir zaten yapısı gereği şeffaf, değişmez, yüksek erişilebilir, güvenilir bir ortamdır.

Amirreza Sarencheh ve diğerleri [32] tarafından önerilen VANET için mahremiyeti koruyan, kimlik doğrulama ile verimli bir iptal şemasını (RAPP) ele alan çalışmada sözde (pseudo) kimlik tabanlı imzadan bahsetmektedir. VANET ağında bulunan kötü niyetli saldırganların herhangi bir zararlı davranışlarını engellemek ve mesaj bütünlüğünü koruyabilmek için yollarda bulunan RSUlar hizmet alanları içerisindeki iptal edilmeyen kullanıcılara geçerli tokenlar dağıtmaktadır. Tokenlar, RSU tarafından OBU'lara verilmektedir. Her RSU iptal listesini periyodik bir şekilde güvenilir otoriteden alır ve hizmet alanı içerisindeki araçların bu listede olup olmadığını kontrol etmektedir. Araçlar mesajlarını hafif (lightweight) simetrik algoritma ile şifrelerler. Tokenı bulunan ağdaki her araç başka bir araç tarafından gönderilen mesajın şifresini çözerek mesaja erişebilir. Kötü niyetli bir saldırgan tespit edildiğinde ya da şüphelenildiğinde araçlar RSUlar aracılığı ile iptal listesini yöneten güvenilir makama bu bilgiyi iletirler. İptal edilme talebi önceden belirlenmiş belirli bir eşik değerini (threshold) geçiyorsa güvenilir makam tarafından iptal listesine eklenilmektedir ve RSU'lara bildirilmektedir. Yapılan performans testi çalışmasına bakıldığında maliyeti çok da ucuz gözükmemektedir ve bahsedilen bütünlük, erişilebilirlik özellikleri kolay bir şekilde sağlanamamaktadır.

IEEE 1609.2 [33] standardı olarak önerilen araç ortamlarında kablosuz erişim için test amaçlı olarak oluşturulan standart çalışmasında ve güvenlik hizmetleri konusunda iptal hizmeti için SİL dosyalarından faydalanılarak sertifika durumunun kontrolü gerçekleştirilmektedir. Her araç için sertifika ömrü kısa tutularak SM tarafından sertifikalar dağıtılmaktadır. SM tarafından belirli aralıklarla yayınlanan SİL dosyaları yerel olarak indirilip kullanıldığı için kullanıcıların gizliliğini sağlamaktadır. Ancak bazı nedenlerden dolayı iptal edilen her sertifika ile SİL dosyasının boyutu artmaktadır ve bellekte büyük boyutta yer kaplamaktadır. Dolayısıyla büyük ölçekli araç ortamları için elverişli bulunmamaktadır.

Kenneth Laberteaux ve diğerleri [34] tarafından sunulan VANET için hafif bir sertifika iptal mekanizmasının tasarım ve analizi çalışmasında her araç için ilişkili olacak gizli iptal anahtarı ile gizliliğin korunup sertifika yönetiminin sağlanması ve SİL'in boyutunu sıkıştırmak için bloom filtreleme tekniğinin kullanılması önerilmektedir. Aracın iptal edilmesi gerektiği durumda sadece aracın iptal anahtarı SİL dosyasına eklenmektedir ve boyutu artmaktadır. Bloom filtreleme yöntemi, sabit hesaplama maliyeti (0(1)) olan ve yanlış pozitif sonuçlar gösterebilecek olan bir tekniktir. SİL dosyasında sertifika aramak, depolamak için kullanılmaktadır. Ayrıca yeni iptal edilen sertifika ile tüm SİL dosyasını güncellemek yerine delta SİL gibi ağ yükünü azaltan, SİL dosyası yayınlandıktan sonra iptal edilenleri listeleyen bir mekanizma önerilmektedir.

Bölüm 3

Ön Çalışmalar

3.1 Blokzincir Teknolojisi

En basit tanımıyla blokzinciri blokların birbirlerine kriptografik olarak bağlanarak oluşturdukları merkezi olmayan dağıtık veri kayıt sistemi olarak tanımlayabiliriz. Ağa katılan düğümler işlemleri takip edebilir. Blokzincir ağında gerçekleşen işlemlerin saklandığı blokların oluşturulması, işlemlerin doğruluğu zaman damgası içeren algoritmalarla ağa katılan madencilerin uzlaşma mekanizmasıyla sağlanır. Temelde blokzincir teknolojisi P2P, uzlaşma mekanizması, dağıtık veri tabanları ve kriptografik tekniklere dayanmaktadır. Defterdeki her işlem tutarlıdır, değiştirilemez ve asla silinemezdir. Aynı zamanda güvenli, esnek, değiştirilemez ve şeffaf bir veri yapısıdır. Üçüncü bir tarafa ihtiyaç duyulmadan dağıtık defter teknolojisi özelliği sayesinde veri sahteciliği yapılmasını engeller. Ayrıca blokzincirde işlemlerin güvenli bir şekilde gerçekleşmesi için kullanılan açık anahtar kriptolojisi ile hesap adresleri açık anahtar kullanılarak oluşturulup, yapılan işlemler özel anahtarla imzalanır. Bu nedenle blokzincir dağıtık defter teknolojisi özellikleri ile yüksek güvenlik sağlayarak bu sisteme duyulan güveni doğrulayıp haklı göstermektedir. Genel olarak blokzincir teknolojisinin sahip olduğu avantajları aşağıdaki gibi sıralayabiliriz [4]:

– Dağıtık defter olması nedeniyle blokzincir ağındaki verilerin bir kopyası ağdaki düğümlere senkronize bir şekilde çoğaltılır ve kaydedilir. Böylece hem yüksek erişilebilirlik sağlanarak yapılan işlemler takip edilebilir hem de verilere zarar verilmesi, veri kaybı engellenmiş olur. Tek bir hata noktası yoktur.

– Üçüncü taraflara ihtiyaç duyulmadan tek bir merkezi otoriteye güven gerektirmeden dijital imzalar ve yapılan doğrulamalarla paydaşlar arasında güven sağlanmış olmaktadır.

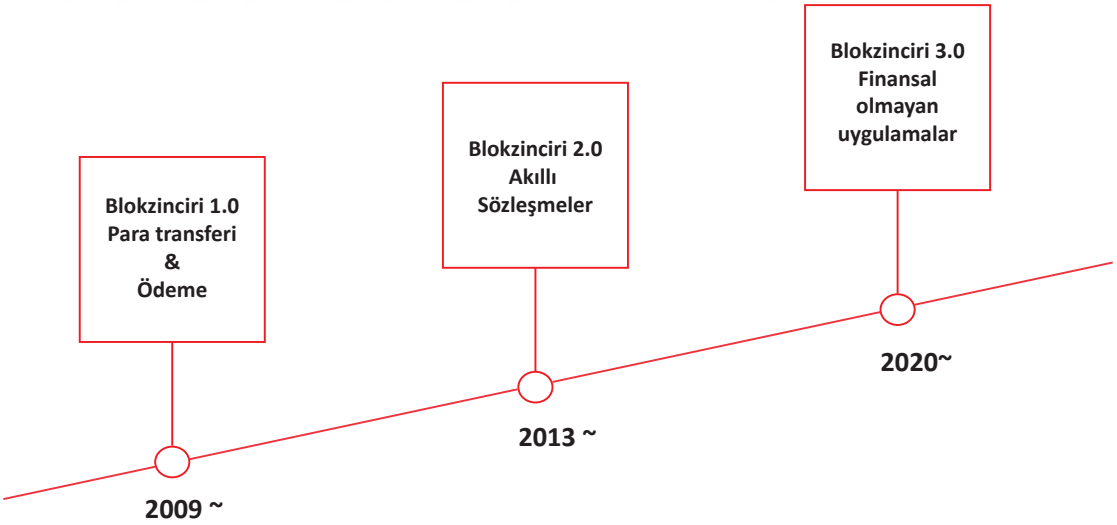
– Deftere kaydedilen veriler kalıcıdır, değişiklik yapılmaz ve silinemez. Veri bütünlüğü korunur.

– İşlemler kronolojik sırada kaydedildiği için tüm bloklar zaman damgası içermektedir.

– Geçmişe yönelik kayıtlar da dahil olmak üzere yetki verilen tüm işlemler görülebilir. Dolayısıyla blokzincirin en önemli özelliği olan şeffaflık sağlanır.

– İşletmeler akıllı sözleşmelerle bazı faaliyetleri önceden ayarlayıp gerekli şartlar sağlandığında tetiklenmesini sağlayarak otomatikleştirebilir.

Kripto paralarda kullanılarak popüler olmaya başlayan blokzincir teknolojisi Blokzinciri 1.0, Blokzinciri 2.0 ve günümüzde Blokzinciri 3.0 olmak üzere üç aşamada geliştirilerek aşağıdaki Şekil 3.1'teki gibi bir gelişim göstermiştir.



Şekil 3.1: Blokzincirin yıllar içerisindeki gelişimi.

Blokzinciri 1.0: Dijital para versiyonudur. Bitcoin gibi kripto paraların kullanılarak para transferi-ödeme işlemlerinin yapılmasını ifade etmektedir.

Blokzinciri 2.0: Dijital ekonomi versiyonudur. Blokzinciri 1.0'da sağlanan işlemlere ek olarak birçok farklı finansal uygulamaları içermektedir. Blokzincir üzerinde akıllı sözleşme olarak adlandırılan dijital sözleşmeler kurularak belirli faaliyetler otomatik olarak hayata geçirilir.

Blokzinciri 3.0: Dijital toplum versiyonudur. Blokzinciri 1.0 ve Blokzinciri 2.0 dışında; bilim, sağlık, eğitim, ulaşım, sanat gibi konuları da içermektedir. Gelecek için daha başarılı çalışmalar sunan blokzincir teknolojisinin kayda değer uygulamalarından biri akıllı ulaşım, akıllı çevre sistemleri, akıllı yönetim gibi anlayışları kapsayan akıllı şehir uygulamalarıdır. Bunun için son zamanlarda sıklıkla duyulan kavram olan nesnelerin interneti (IoT) ile makinelerin, araçların haberleşmesi için blokzinciri teknolojisinin özelliklerinden yararlanmak mümkündür.

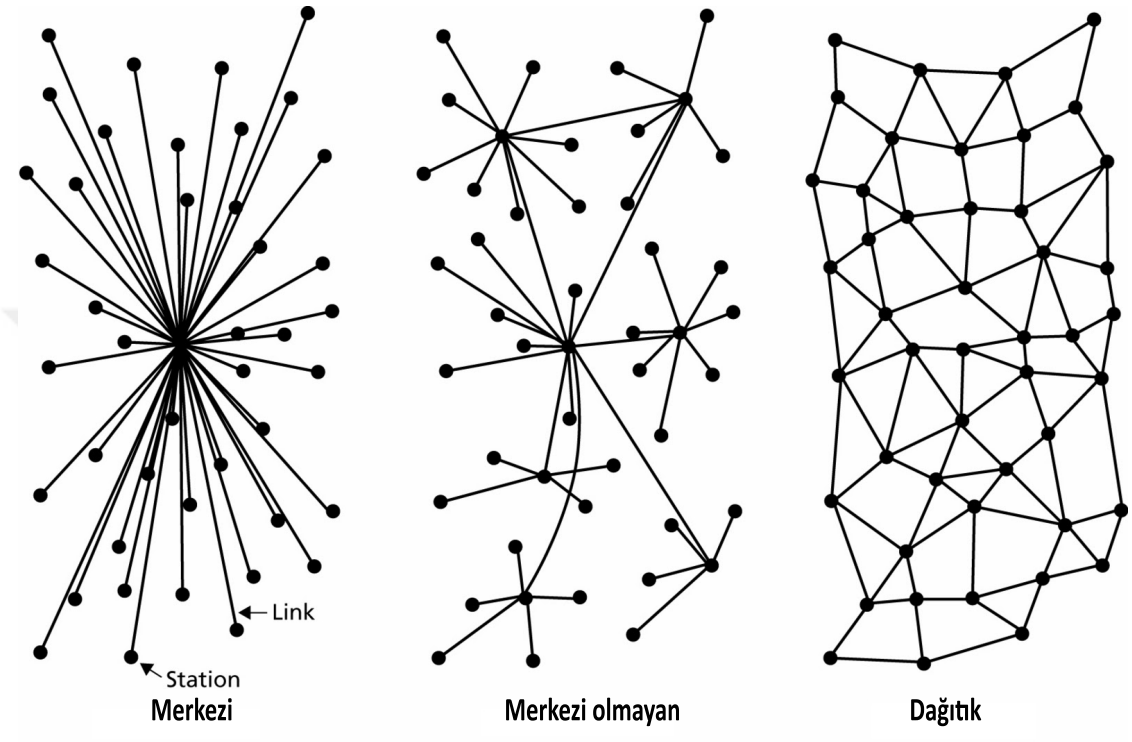
3.1.1 Blokzincir Çalışma Prensibi ve Mimarisi

Öncelikle blokzincirde kullanılan P2P ağ yapısından bahsetmek gerekirse merkezi bir sunucuya gereksinim olmadan tek nokta hatasını ortadan kaldırarak blokzincir ağındaki katılımcılar arasında dolaysız sağlanan veri paylaşımıyla meydana gelen dağıtık ağ yapısıdır. P2P ağlardaki düğümler eşittir. Blokzincirde tutulan veriler ağdaki eşler arasında kopyalanarak çoğaltılır ve değişiklik yapılamaz bir şekildedir. Tek bir merkezi nokta kullanılmadığı için verilerin zarar görmesi, kaybolması gibi riskler azalmaktadır. Bu nedenle kullanıcılar için güvenlik ve gizlilik riskleri daha az taşımaktadır.

Merkezi bir sistemde ise tüm kullanıcılar merkezi bir ağ sahibine veya sunucuya bağlanmaktadır. Ağ sahipleri, diğer kullanıcıların erişebileceği verileri ve kullanıcı bilgilerini depolamaktadır. Kurulumu kolaydır ve kolayca geliştirilebilir ancak tek bir merkezi nokta olduğu için verilerin zarar görme, kaybolma riski daha yüksektir. Bu nedenle kullanıcılar için güvenlik ve gizlilik riskleri daha fazla taşımaktadır. Merkezi yapıların sahip olduğu bu dezavantajlara göre dağıtık defter teknolojisi olan blokzincir daha ölçeklenebilir ve hataya dayanıklıdır.

Adından da anlaşılacağı üzere merkezi olmayan sistemlerin de tek bir merkezi sahibi yoktur. Bunun yerine, her biri genellikle kullanıcıların erişebileceği kaynakların bir kopyasını depolayan birden çok merkezi sahip kullanırlar.

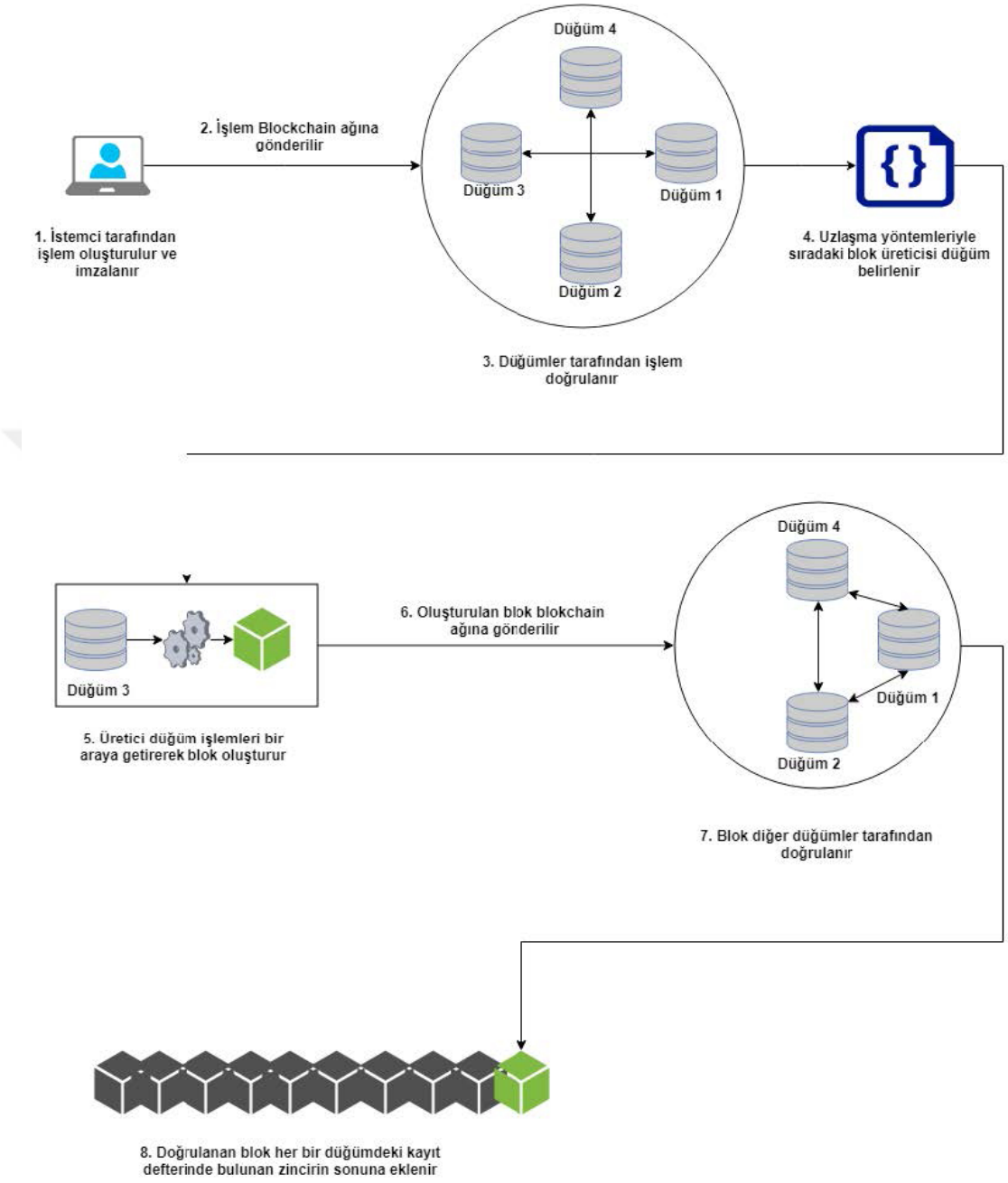
Dağıtılmış bir sistem, merkezi olmayan bir sisteme benzemektedir, çünkü tek bir merkezi sahibi yoktur. Ancak bir adım daha ileri giderek merkezileşmeyi ortadan kaldırır. Dağıtılmış bir sistemde kullanıcılar verilere eşit erişime sahiptir, ancak gerektiğinde kullanıcı ayrıcalıkları etkinleştirilebilir. Geniş, dağıtılmış bir sistemin en iyi örneği internetin kendisidir. Bahsedilen üç farklı ağ aşağıdaki Şekil 3.2'deki yapı gibi gösterilmektedir.



Şekil 3.2: Merkezi, merkezi olmayan, dağıtık ağ yapısı [1].

Blokzincir çalışma prensibi birçok insan tarafından karmaşık olarak görülse de daha önce de birçok alanda kullanılmış ve yapısında bazı temel bileşenler bulunmaktadır. Bunlar, parça parça ele alınarak daha anlaşılabilir olup blokzincir altyapısı kullanımını amaçlayan uygulamaların hayata geçirilmesini kolaylaştıracaktır.

Açık anahtar kriptolojisi kullanılarak açık anahtarla oluşturulan hesap adresleri ile blokzincirde yapılan işlemler özel anahtarla imzalanır ve ağa yayınlanır. Ağdaki düğümler, yayınlanan işlemi değerlendirmek ve doğrulamak için bazı denetimler yapar. Uzlaşma mekanizmalarıyla bloğun hangi düğüm tarafından yayınlanacağı belirlenir. Üretilen blok diğer düğümler tarafından da bir fikir birliğine varılıp doğrulanır ve zincire yeni bir blok eklenir. Bu kapsamda blokzincir teknolojisinin çalışma mekanizması Şekil 3.3'de gösterildiği gibidir [2].

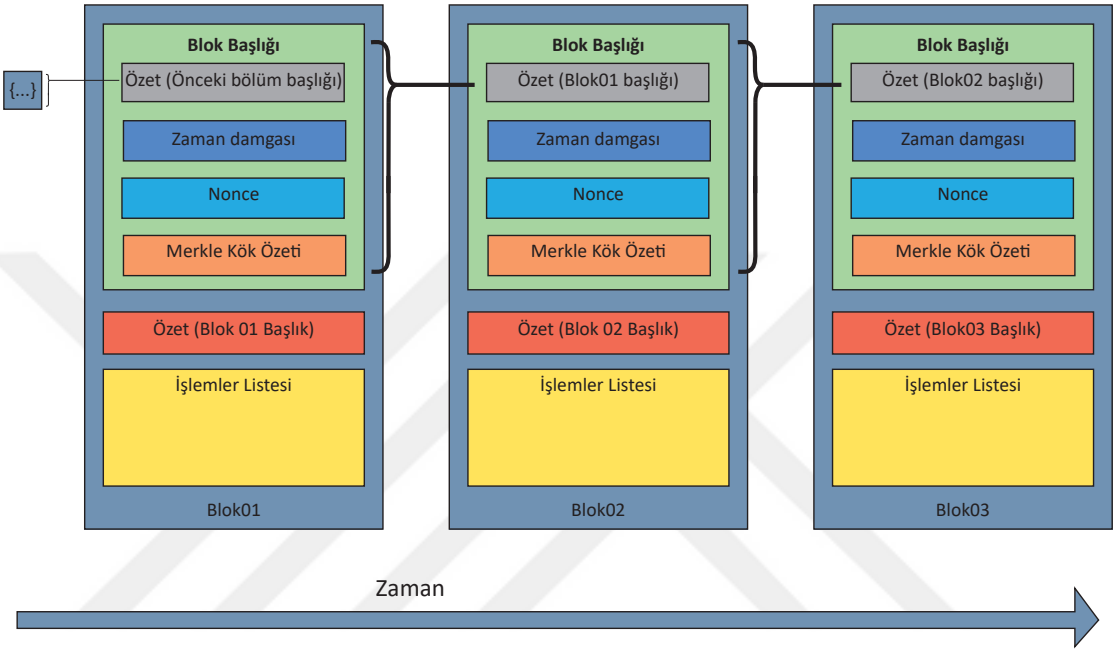


Şekil 3.3: Blokzincir mimarisi [2].

3.1.1.1 Temel Bileşenleri

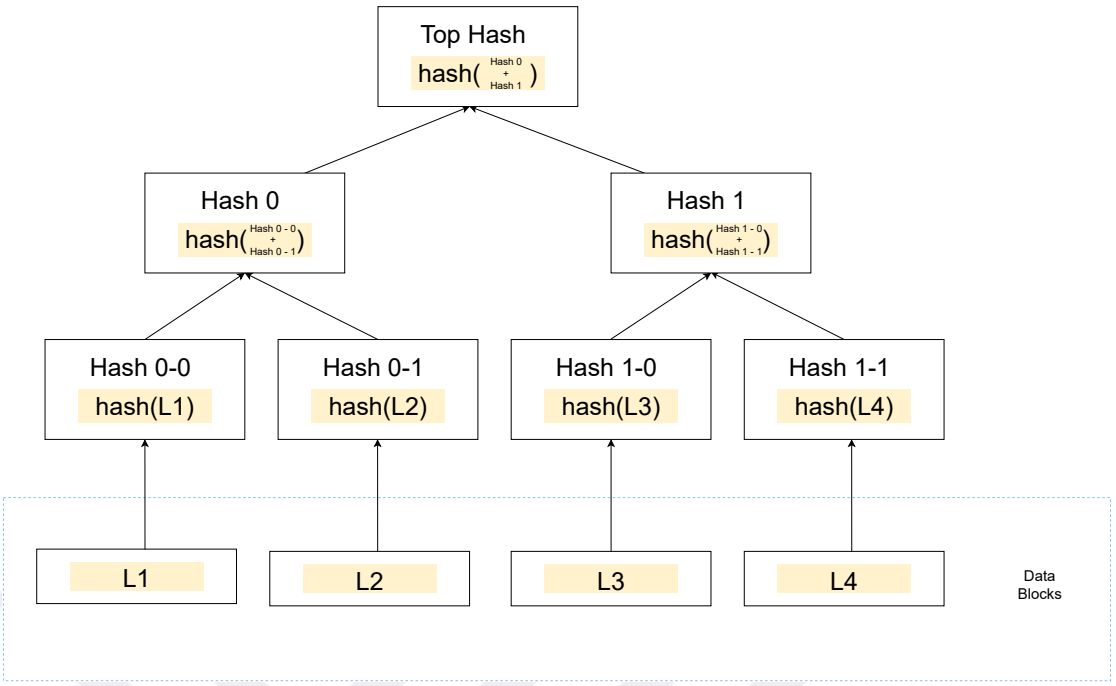
Blokzincirin temel bileşenleri; açık anahtar kriptolojisi, bloklar, hesap adresleri, düğümler, işlemler, defterler, özet fonksiyonları, uzlaşma yöntemleridir [3]. Hesap adresleri kredi kartlarında bulunan IBAN numarası olarak varsayılabilir ve blokzincir ağına katılan

düğümün kimliği niteliğindedir. Özet fonksiyonları herhangi bir uzunlukta olan verileri sabit uzunlukta bir veriye dönüştürür ve veride tek bir nokta bile değişse hesaplanan özet değeri de değişmektedir. Blokzinciri yapısında bloklar birbirine bağlıdır ve bir önceki bloğun sahip olduğu özet değerini tutar. Böylelikle zincir halindeki blokların değişikliğe uğraması önlenir. Blokların zincirlenmiş hali ve blok başlıkları aşağıdaki Şekil 3.4'te gösterilmiştir.



Şekil 3.4: Blokların zincirlenmiş hali ve blok başlıkları [3].

Her blok gövde ve başlık olmak üzere 2 kısımdan oluşur. Blok başlığında bloğa ait işlemlerin bulunduğu Merkle kök özeti, rasgele bir sayı olan nonce değeri, bloğun oluşturulma zamanını içeren zaman damgası ve bir önceki bloğun özet değeri vardır. Gövdesinde de ağda gerçekleşen işlemler ve kendi blok başlığının özet değeri vardır. Ayrıca blokta algoritmanın zorluk derecesi, bloğun versiyon numarasını da bulunmaktadır. Merkle ağacı ya da hash ağacı olarak bilinen veri yapısı, verilerin güvenli bir şekilde doğrulanmasını sağlar ve veri bütünlüğünü korur. Özet fonksiyonlarını kullanarak her yaprak olmayan düğüm alt düğümlerinin kriptografik özet değerini bulundurmaktadır. Merkle kök özeti de ağacın tepesinde bulunan özet değeridir. Aşağıdaki Şekil 3.5'te örnek Merkle ağacı gösterilmektedir.

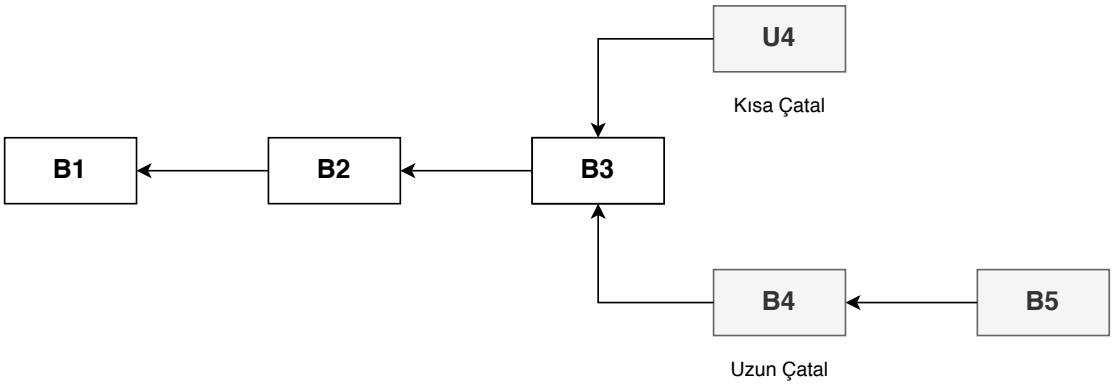


Şekil 3.5: Merkle ağacı örneği.

3.1.1.2 Uzlaşma Yöntemleri

Blokzincirin sonuna eklenecek olan blok için ağdaki madencilerin çoğunluğunun fikir birliği sağlaması gerekmektedir. Fikir birliği sağlandıktan sonra blok eklenmektedir. Bunun için blokzincirde bazı uzlaşma yöntemleri bulunmaktadır.

İş Kanıtı (Proof of Work - PoW) İsminde de anlaşılacağı üzere blok eklemek için emek gerektiren bir uzlaşma yöntemidir. PoW, Bitcoin kripto parasının kullandığı bir yöntemdir. Hesaplanması kolay olmayan kriptografik bilmecelere dayanır. Bu bilmeceleri çözmeye çalışan ağdaki düğüme madenci denir. Belirlenen özet değerini elde eden ilk düğüm blokzincirin sonuna blok ekleme hakkına sahip olur. Eğer aynı anda birden fazla madenci aynı özet değerini aynı anda elde ederse ve ağa gönderirse Şekil 3.6'daki gibi bir çatallanma gerçekleşir. Bu durumda çatalların birinde yeni bir blok üretilene kadar iki çatal da bulunmaya devam edecektir. Hangisine yeni bir blok eklenirse uzun çatalda madenciler çalışmalarına devam ederler. Matematiksel çözümlerin zorluğundan dolayı çok fazla bilgisayar gücüne gereksinimi vardır ve çok zaman harcanmaktadır [4].



Şekil 3.6: Blokzincirde çatallanma örneği [4].

Varlık Kanıtı (Proof of Stake - PoS) Çok fazla elektrik harcayan PoW algoritmasına alternatif olarak PoS uzlaşması ortaya konulmuştur. Bu uzlaşma yönteminde ağdaki madenciler arasında fazla varlık hissesi olanlar blok ekleme hakkı elde ederler. Bu yüzden kripto paralarının miktarını ispatlaması gerekmektedir. PoW'daki gibi bir kriptografik bilmece çözme gerektirmediği için işlem gücü fazla değildir. Ancak en fazla hissesi olan düğüm hep baskın olabileceğinden dolayı bir sonraki bloğu yaymak için varlıkların oluşturulma zamanı gibi bilgilere bakılarak çözümler sunulmuştur. Bir başka alternatif çözüm seçeneklerinden biri de düğümlerin yatırdığı dijital paralara süre sınırı eklenerek düğümler pasif hale getirilir. Belirli bir süre geçtikten sonra blok oluşturma hakkı verilip madenci aktif hale getirilir [4].

Otorite Kanıtı (Proof of Authority - PoA) Diğer uzlaşma yöntemlerine göre birkaç önemli avantajı bulunmaktadır. PoW ve PoS yöntemlerine göre belli sayıda blok doğrulayıcısı olduğu için daha ölçeklenebilirdir. Dijital kimliklerle bloklar onaylanır. Dolayısıyla kripto para miktarı yerine kendi itibarlarını kullanırlar. Yeni blokların üretildiği zaman aralığı tahmin edilebilir. PoW ve PoS mutabakatı için bu süre değişir. Yüksek işlem oranı vardır. Bloklar, yetkili ağ düğümleri tarafından belirlenen zaman aralığında bir sırada oluşturulur. Bu da, işlemlerin onaylanma hızını artırır. PoA'nın kullanıldığı bir örnek olarak Microsoft Azure verilebilir. Özel ağlara alternatif bir çözüm olarak sunularak madencilik yapılmasına ihtiyaç olmaz.

3.1.2 Blokzincir Ağ Türleri

Farklı ihtiyaçlara göre blokzincir ağ türleri açık, özel veya konsorsiyum kategorilerinde uygulanabilir [4]. Bu üç farklı blokzincir ağ türlerinin kıyaslanması aşağıdaki Tablo 3.1'deki gibidir.

Açık Blokzincir: Herhangi bir erişim sınırlaması yoktur. Ağa herkes katılabilir. Bu yüzden tamamen şeffaftır ve blokzincirde verilerin tutarlılığını ve güvenliğini artırmaktadır. Ancak ağda çok fazla düğüm olduğu için verimlilik biraz daha düşüktür.

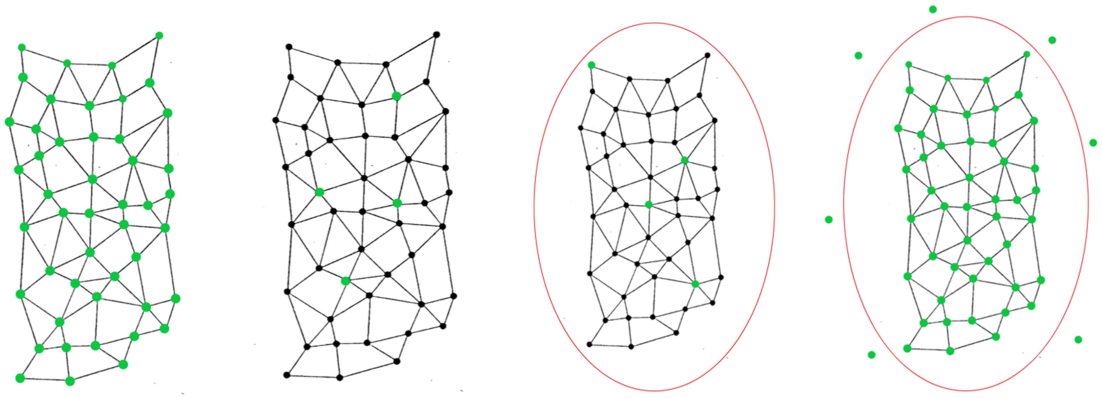
Özel Blokzincir: Sadece belirli bir kuruluştaki kişi ya da gruplar ağa dahil olabilmektedir. Ağa katılmak için erişim izni olması gerektiğinden izne tabidir. Ağa katılacak olan yeni kişi ya da gruplara erişim hakkı blokzinciri yönetenler tarafından verilerek gerçekleşmektedir.

Konsorsiyum Blokzincir: Yalnızca önceden seçilen düğümler uzlaşmada bulunmaktadır. Yani, bir kurumda önceden belirlenen bir grup blok doğrulamada bulunabilir. Kimlerin ağa dahil olup olamayacağına bu gruplar karar vermektedir. Kısmen hem özel hem de izin gerektiren bir blokzincir ağıdır.

Tablo 3.1: Açık blokzincir, özel blokzincir ve konsorsiyum blokzincir çeşitlerinin kıyaslanması [4].

	Açık Blokzincir	Konsorsiyum Blokzincir	Özel Blokzincir
Merkeziyetçilik	Hayır	Kısmen	Evet
Okuma İzni	Herkes	Herkes/Sınırlı Kullanıcı	Herkes/ Sınırlı Kullanıcı)
Ortak Karar Verme	Bütün Madenciler	Seçilen Düğümler	Tek Organizasyon
Verimlilik	Düşük	Yüksek	Yüksek
Uzlaşma Süreci	İzinsiz	İzinli	İzinli

Ayrıca bu üç kategorideki blokzincir ağ çeşitleri incelendiğinde aşağıda Şekil 3.7'de bulunan görseldeki gibi ağa katılmak için izin gerektiren ve izin gerektirmeyen blokzincir olarak sınıflandırılmaktadır. İzne tabi blokzincir ağında sadece yetki verilen katılımcılar ile bloklar oluşturulur. Sınırlı bir uzlaşma süreci ile kontrol edilmektedir. İzne gereksinim olmayan ağlarda ise bütün katılımcılar blok oluşturabilir.



Şekil 3.7: Açık İzinsiz, Açık İzinli, Konsorsiyum (Özel İzinli) ve Özel izinsiz blokzincir çeşitleri [5].

Bu haliyle tekrar ele alınan blokzincir yapısında, açık/özel ve izinli/izinsiz olmak üzere iki temel özellik bulundurduğu sonucuna ulaşılabilmektedir. Açık/özel ekseni ile hangi kullanıcıların blokzinciri görüntüleyebileceği ya da ağa işlem göndereceği belirlenmektedir. İzinli/izinsiz ekseni ile fikir birliğine hangi kullanıcıların katılacağı belirlenmektedir. Bu kapsamdaki iki eksen Tablo 3.2'deki blokzincir teknolojisinin sınflandırılmasına genel bir bakış sunmak için kullanılmaktadır [5].

Tablo 3.2: Blokzincir teknolojisinin sınflandırılması [5].

	İzinsiz	İzinli
Açık	- Verileri kimlerin okuyabileceği konusunda kısıtlama yok. - Kimlerin işlem veya program gönderebileceği konusunda herhangi bir kısıtlama yoktur. - Uzlaşma mekanizmasına kimlerin katılabileceğine dair herhangi bir kısıtlama yoktur.	- Verileri kimlerin okuyabileceği konusunda kısıtlama yok - Kimlerin işlem veya program gönderebileceği konusunda herhangi bir kısıtlama yoktur - Uzlaşma mekanizmasına kimlerin katılabileceğine ilişkin kısıtlama vardır.
Özel	- Kimlerin veri okuyabileceğine ilişkin kısıtlamalar vardır. - Kimlerin işlem veya program gönderebileceğine ilişkin kısıtlamalar vardır. - Uzlaşma sürecine kimlerin katılabileceğine dair herhangi bir kısıtlama yoktur.	- Kimlerin veri okuyabileceğine ilişkin kısıtlamalar vardır. - Kimlerin işlem veya program gönderebileceğine ilişkin kısıtlamalar vardır. - Uzlaşma sürecine kimlerin katılabileceğine dair bir kısıtlama vardır.

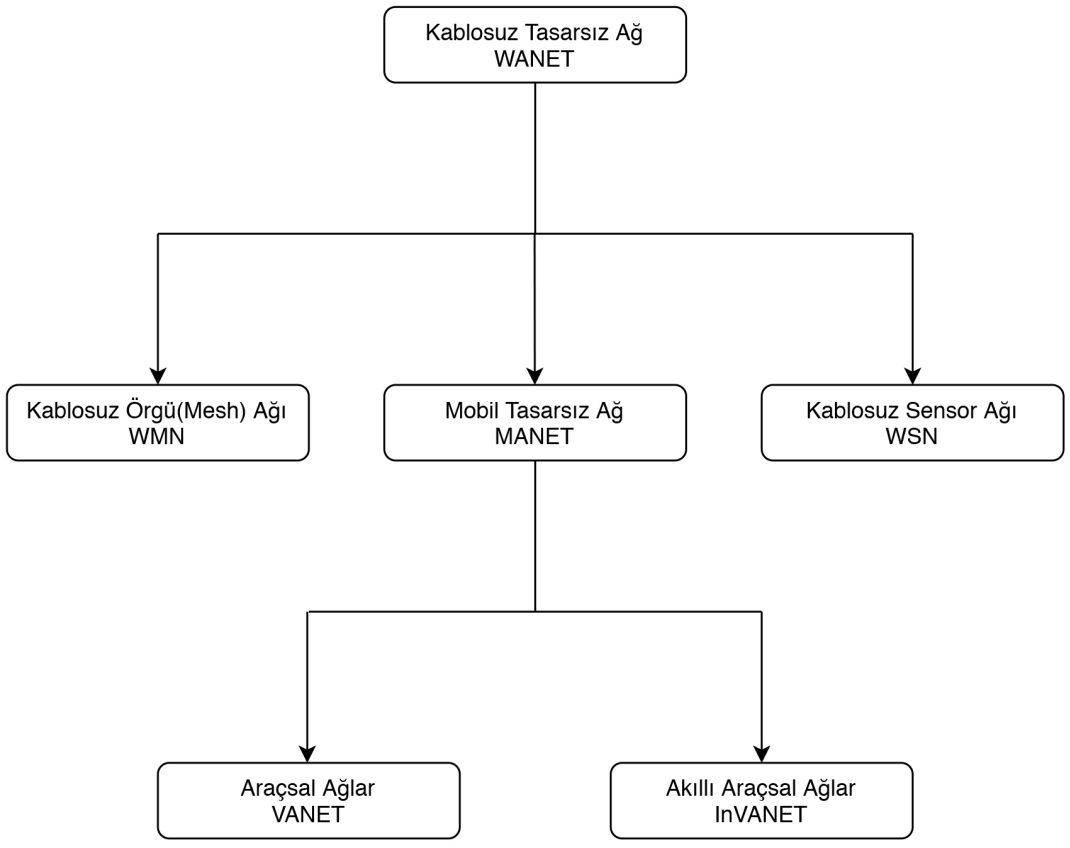
3.1.3 Blokzincir Teknolojisinin Kullanım Alanları

Birçok sektör son zamanlarda popüler olan blokzincir dağıtık defter teknolojisinin sağladığı avantajlardan yararlanabilmek için çalışmalar yürütmektedir ve uygulamaya geçirmişlerdir. Başta sadece Bitcoin gibi kripto paralarda yaygın olarak kullanılmaktaydı. Ancak finans sektöründeki kurumlar kullanım alanlarını genişletmek için araştırmalar yapmaktadır. Bu araştırmalar para transferleri, ödeme işlemleri, dijital kimlik yönetimi gibi konularla ilgilidir. Dolayısıyla blokzincir günümüzde artık sadece kripto para uygulamalarını kapsamayıp birçok farklı sektörlerde ihtiyaçları gidermek için kullanılması planlanmıştır.

Kamu sektöründe blokzincirin kullanım alanları bulunmaktadır. Bu kullanım alanları oylama, akıllı sözleşmeler, dijital pasaport, dijital kimlik gibi konularla ilgilidir. Artık birçok ülke bununla ilgili çalışmalar yapıp blokzincir altyapısını kullanmak için yatırım yapmaya başlamıştır. Kamu sektörü kuruluşları ihtiyaçlarına göre blokzincir ağ türüne karar vererek uygulamalarını blokzincir yapısına uygun olacak şekilde uyarlayabilir. Blokzincir altyapısından faydalanmak isteyen bir başka kamusal sektör de savunma alanıdır. Diğer bir sektör olan sağlık sektöründe de hastalara ait verilerin mahremiyetinin korunması yaşanılan sıkıntılardan birisidir. Blokzincir ağdaki düğümlerin yetkilerine göre erişim sağlayıp güvenilir bir ortam sağlamaktadır. Bu yüzden blokzincirin kullanım alanlarından birisidir. Blokzincir üzerinde akıllı sözleşme olarak adlandırılan dijital sözleşmeler kurularak belirli faaliyetler otomatik olarak hayata geçirilebildiği için birçok sektörü etkileyen en önemli kullanım alanlarından birisidir. Bunun için Ethereum en çok tercih edilendir. Merkezi bir makama gereksinim olmadan ortaklar arasında belirlenen koşullara göre sözleşmeler tanımlanır ve buna göre planlanan eylemler otomatikleştirilir. Blokzincir 2.0 ile gelen akıllı sözleşmeler blokzincir teknolojisinin sahip olduğu kayda değer özelliklerinden birisidir [35]. Blokzincir 3.0 ile akıllı ulaşım, akıllı çevre sistemleri, akıllı yönetim, şehir uygulamaları gibi IoT tabanlı konularda da blokzinciri teknolojisinin özelliklerinden yararlanmak mümkündür.

3.2 VANET Teknolojisi

Tasarsız araç ağlarında bulunan "tasarsız (ad hoc)" terimi latincede "amaca özel" anlamına gelmektedir ve özel bir probleme yönelik geçici bir çözümü anlatmak için kullanılır. Tasarsız ağlar da kablosuz ağları ifade etmektedir ve tasarsız ağlarında kablosuz bağlantılar yoluyla bağlanan düğümler bulunmaktadır. Şekil 3.8'de gösterildiği gibi VANET üç kategoride bulunan kablosuz tasarsız ağlarında MANET'in bir alt sınıfı olarak bulunmaktadır [6]. MANET herhangi bir altyapıya gereksinim duymadan gönderen ve alıcı arasındaki bağlantıyı kuran bir kablosuz ağ çeşididir. MANET'in özelleştirilmiş hali olarak düşünülen VANET teknolojisi araçları mobil düğüm olarak kullanmaktadır. Yakındaki araçlar arasında ve araçlar ile yol kenarı üniteleri arasında haberleşmeyi sağlamaktadır [36].



Şekil 3.8: Tasarsız ağların sınıflandırılması [6].

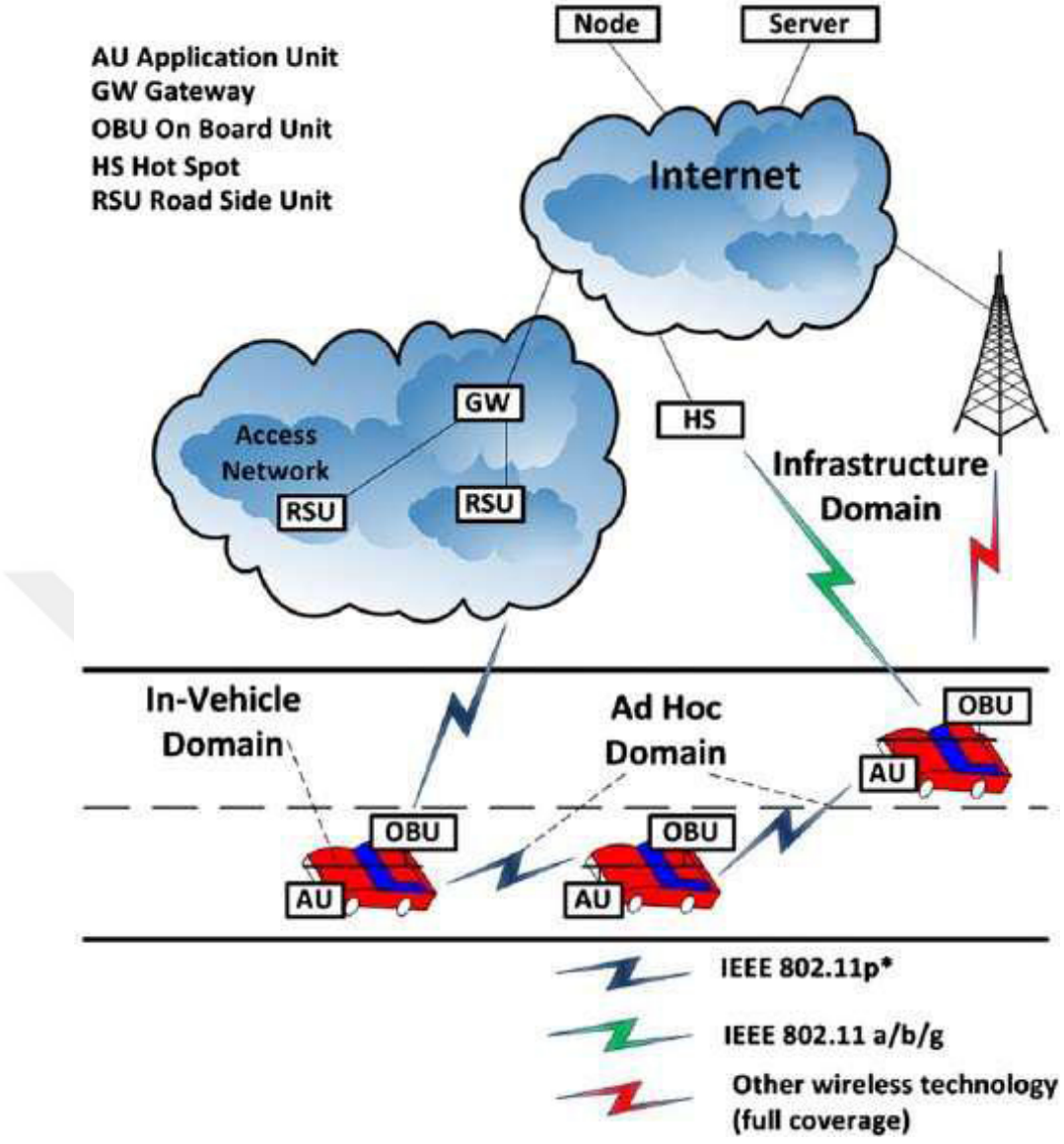
VANET'ler akıllı ulaşım sistemlerinin temelidir. Bunun için akıllı ulaşım ağları olarak da adlandırılmaktadır. Temel amacı yol güvenliğini sağlayarak trafik kazalarını azaltmak, yolcuların rahatlığını sağlamak ve trafiğin verimliliğini iyileştirmektir. Birbirleri ile

kendi aralarında kablosuz iletişim kurulabilmesi için DSRC protokolü kullanılır ve yol durumu hakkında bilgiler verilir. VANET ağında yapılan mesaj alışverişi aşağıdaki gibi gerçekleşebilir [6]:

- *Acil durum ve uyarı mesajları:* Kaza, acil durum araçlarının geçişi gibi önemli ve kritik olaylarda acil durum mesajları iletilir ya da yol çalışmaları, trafik sıkışıklığı gibi durumlarda uyarı mesajları iletilebilir.
- *Kişisel mesajlar:* Bir sürücü ve araçtaki diğer yolcular arasında kişisel mesaj alışverişi gerçekleşebilir.
- *Güvenlik mesajları:* Aracın hızı, durumu, konumu gibi mesajlar iletilebilir.
- *Bilgi ve eğlence mesajları:* Bu mesajlar, yol kenarı ünitesinde bulunan ve / veya yoldaki diğer araçlar tarafından sunulan servislerle gerçekleşir. Yakın civarlarında bulunan yemek noktaları, benzin dolum istasyonları gibi konular için mesajlar iletilebilir.

3.2.1 VANET Mimarisi

Bu bölümde ilk olarak VANET sisteminin temel bileşenleri, iletişim tipleri ve sonrasında kullanılan kablosuz teknolojilerden bahsedilerek Şekil 3.9'de gösterilen VANET sistem mimarisi açıklanmaktadır [7].



Şekil 3.9: VANET mimarisi [7].

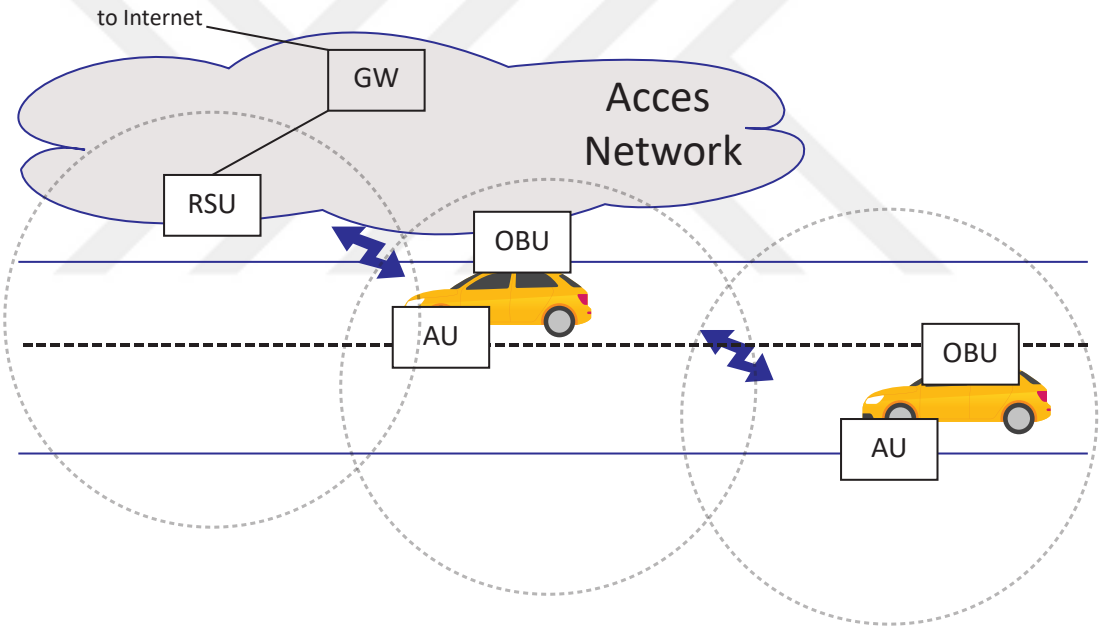
3.2.1.1 VANET Bileşenleri

VANET mimarisinin ana sistem bileşenleri Şekil 3.10'de gösterildiği gibi yerleşik ünite (OBU), uygulama ünitesi (AU) ve yol kenarı ünitesi (RSU) olmak üzere üç tanedir [7].

– *Yerleşik Ünite - On Board Unit (OBU):* Kablosuz iletişim ile bilgi alışverişi sağlamak için araçta kurulu bulunan bir dalga cihazıdır. Bir OBU, RSU ya da diğer araçlardaki OBU ile mesaj alışverişi yapar. OBUlar küresel konumlama sistemi (GPS) ile coğrafi konum, hız gibi bilgilerin elde edilmesini, güvenilir mesaj aktarımının yapılmasını, kablosuz radyo erişimini, ağın yoğun olup olmadığının kontrolünün yapılması gibi işlemleri sağlamaktadır.

– *Uygulama Ünitesi - Application Unit (AU):* OBU'nun haberleşme kapasitesini kullanır ve OBU'ya kablolu ya da kablosuz bağlantı ile bağlanarak ağa bağlanır. Güvenlik uygulamaları için özel bir cihaz olabilir ya da kişisel dijital asistan (PDA) gibi bir cihaz olabilir. OBU ile tek bir fiziksel ünitede bulunabilir. OBU ile arasındaki fark mantıksaldır.

– *Yol Kenarı Ünitesi - Road Side Unit (RSU):* Yol kenarı boyunca, kavşak, park alanları gibi yerlere sabitlenmiş ünitelerdir. Güvenilir otoritelere ve uygulama sunucularına bağlanabilir. RSU'lar OBU'lara internet erişimi sağlayabilirler ve OBU'ların haberleşme kapasitesini büyütebilirler. Diğer RSU'lar ile haberleşerek mesajları yönlendirebilirler. Ayrıca güvenlik uygulamaları çalıştırarak bilgi kaynağı olarak davranabilir.



Şekil 3.10: Sistem bileşenleri [7].

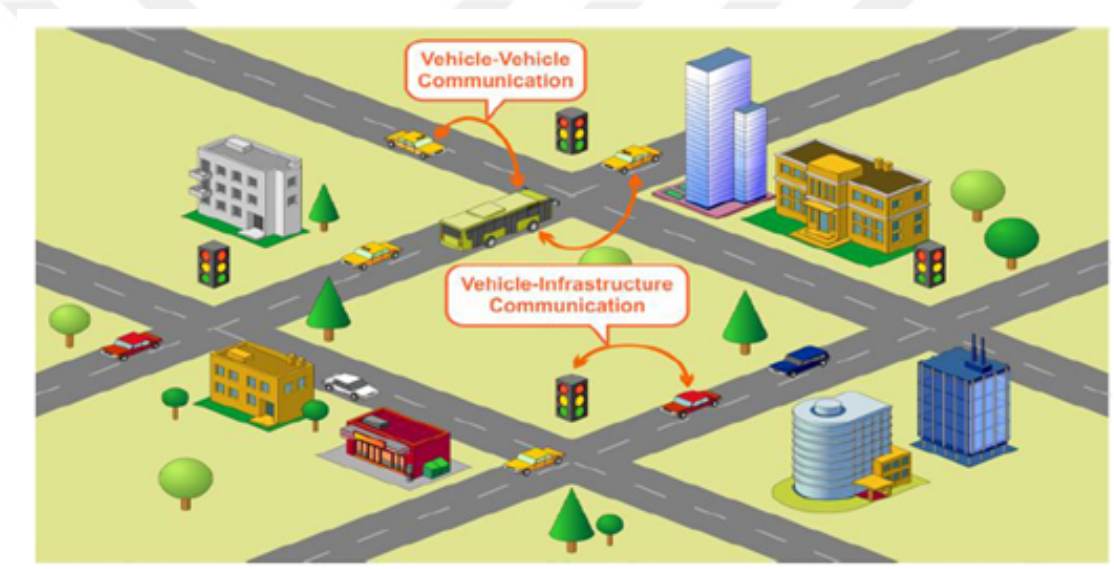
3.2.1.2 VANET İletişim Tipleri

VANET'de araçtan araca (V2V) ve araçtan altyapıya (V2I) iletişim olmak üzere aşağıdaki Şekil 3.11'de gösterildiği gibi temelde iki çeşit iletişim tipi bulunmaktadır [8].

– *V2V İletişimi:* Araçlar birbirleriyle haberleşir. V2V teknolojisi araçların birbirleriyle doğrudan iletişime geçerek hız, konum, fren, şerit değişikliği gibi bilgileri

paylaşabilmelerini sağlamaktadır. Bir araca iletilen mesaj bir başka araca da yönlendirilebilir. Bu sayede trafik kazası, yol durumu gibi konular hakkında bilgilendirerek olası trafik kazalarının önüne geçilmiş olmaktadır. Araçlar için multi hop iletişim sağlanmaktadır.

– *V2I İletişimi:* Araçlar çevrede bulunan altyapı araçlarıyla haberleşir ve haberleşme çift yönlüdür. Altyapı araçları, RFID okuyucuları ve kameralar, trafik ışıkları, yol kenarı üniteleri gibi araç gereçler olabilir. V2I teknolojisinde araçlar yol kenarındaki bu donanımlarla bağlantı kurarak kaza, yol koşulları, trafik yoğunluğu gibi çeşitli bilgiler iletilebilmektedir. V2V iletişimine benzer şekilde V2I teknolojisinde iletişim kurulabilmesi için DSRC protokolü kullanılmaktadır.



Şekil 3.11: Araçsal ağların iletişim örneği [8].

3.2.1.3 Kullanılan Kablosuz Teknolojiler

VANET'de kullanılan kablosuz teknolojiler hücresel ağ, WLAN/Wi-Fi, WiMAX, DSRC/WAWE gibi teknolojilerdir. Mobil ağ olarak da adlandırılan hücresel ağlarda mobil iletişim için küresel sistem (GSM) iletişimi en eski hücresel sistem standardıdır ve hücresel sistemin mevcut alt yapısı kullanılmaktadır. Günümüzde GSM teknolojisinin zaman içerisinde gelişmesi ile 2G/3G/4G şeklinde artan nesiller ile daha yüksek veri hızları sağlanmaktadır. Wi-Fi veya kablosuz yerel alan ağı (WLAN) hareketli araçlar için internet erişimi sağlayarak V2V / V2I iletişimini etkinleştirmek için kullanılmaktadır. WiMAX (mikrodalga erişimi için dünya çapında birlikte çalışabilirlik) teknolojisi

de yüksek hizmet kalitesi ve güvenilir iletişim standartları sunar ve multimedya, video gibi kullanımlar gerektiren uygulamalar için uygundur. WiMAX kablosuz teknolojisi ile de V2V ve V2I iletişim desteklenebilmektedir. Bir diğer kablosuz teknoloji olan DSRC'de V2V ve V2I haberleşmede kullanılmaktadır. DSRC'nin amacı trafik, kaza, yol durumu bilgileri gibi mesajlar iletilirken iletişim sırasındaki gecikmeyi olabildiğince minimuma düşürmek ve yüksek veri aktarımını sağlayabilmektir [6].

3.2.2 VANET'in Karakteristikleri

VANET ile MANET arasında benzerlikler bulunmasına rağmen VANET'in aşağıda belirtildiği gibi kendine özgü bazı karakteristik özellikleri bulunmaktadır [37]:

- *Yüksek hareketlilik:* VANET'in en önemli özelliğidir ve MANET'e göre oldukça yüksek hareketliliğe sahiptir. Ağdaki düğümler farklı yönlerde yüksek hızlarla hareket etmektedirler ve bu da ağdaki karmaşıklığı (mesh) azaltır. Ayrıca düğümler MANET'deki gibi açık alanda serbest hareket etmezler çünkü VANET'de yol güvenliğini sağlamak için diğer hareketli araçlara cevap verme kısıtlılığı bulunmaktadır.

- *Yol güvenliğini sağlama, yolcuların rahatlığını sağlama, trafik verimliliğini iyileştirme:* VANET sisteminde araçlar birbirleri ile doğrudan iletişime geçerken bazı uygulamaların kullanımına izin vererek sürücülere olası kaza durumu gibi acil ihtiyaçlarda uyarılar vermektedir ve ayrıca trafik sıkışıklığı, hava durumu gibi bilgiler edinerek sürücünün rahatlığını sağlayabilmektedir.

- *Büyük ölçekli ağa sahip olması:* Özellikle nüfusun yoğun olduğu alanlarda ağ ölçeği de büyük olmaktadır.

- *Ağ topolojisindeki hızlı değişiklikler:* Araçların yüksek hızla hareket etmesi nedeniyle, topoloji değişiklikleri sık gerçekleşir ve V2V / V2I iletişiminde kısa süreli bağlantıya neden olur.

- *Gerçek zamanlı, zamana duyarlı veri alışverişi:* Sürücülere zamanında sahte olmayan bilgileri sağlayarak yol güvenliğini artırmak ve trafik akışını kolaylaştırarak verimliliğini iyileştirmek için veri alışverişi zamana karşı duyarlı olup gerçek zamanlı gerçekleşir.

– *Yüksek hesaplama kapasitesi:* VANET ağındaki düğümlerde GPS, gelişmiş anten teknolojisi, büyük bir bellek kapasitesi, sensör gibi kaynaklar bulunmaktadır. Bu yüzden bu tür kaynaklar düğümün hesaplama kapasitesini arttırır ve güvenilir kablosuz iletişim kurulmasını sağlayarak doğru bilgi edinilir.

3.2.3 VANET'in Güvenlik Gereksinimleri

VANET teknolojisinde güvenli ve doğru ortamın kurulması hayati önem taşımaktadır bu yüzden aşağıda belirtildiği gibi güvenlik ve mahremiyetin korunması ile ilgili gereksinimlere ihtiyaç duyulmaktadır [37].

– *Kimlik doğrulama:* Ağdaki düğümlerin kimliğinin doğrulanması son derece önemlidir ve her aracın kimlik doğrulama yapması zorunludur. Araçlar dijital sertifikası ile birlikte özel anahtarıyla mesajları imzalar çünkü kötü niyetli bir saldırgan bir başkası adına sahte mesajlar iletebilir ve sisteme zarar verebilir. Bu yüzden kimliği doğrulanmayan araçların mesaj göndermelerine izin verilmemelidir.

– *Erişilebilirlik:* Mesaj göndermek ve almak için gerçek zamanlı olarak her zaman VANET'in erişilebilir olması gerekmektedir. Bazı uygulamalarda gecikmelerin yaşanması önemli sorunlara yol açabilir ve ayrıca gerçek zamanlı isteklerin karşılanması sistemde DDOS ataklara yol açabilir.

– *Mahremiyet:* Sürücülerin gerçek kimlik bilgileri açığa çıkmadan veri alışverişi gerçekleştirilmelidir.

– *Bütünlük:* Ağ üzerinden gönderilen mesajların içeriklerinin değiştirilmesine izin verilmemelidir, tutarlı olmalıdır çünkü sahte mesajlar gerçek zamanda can ve mal kaybına neden olabilmektedir.

– *İnkar edilemezlik:* Ağa yapılan herhangi bir saldırı ile yanlış yönlendirme yapan araçların verdiği olumsuzlukları reddedememesi gerekmektedir.

– *Yetki:* Ağa katılan her aracın neleri yapıp yapamayacağının belirlenmesi gerekmektedir.

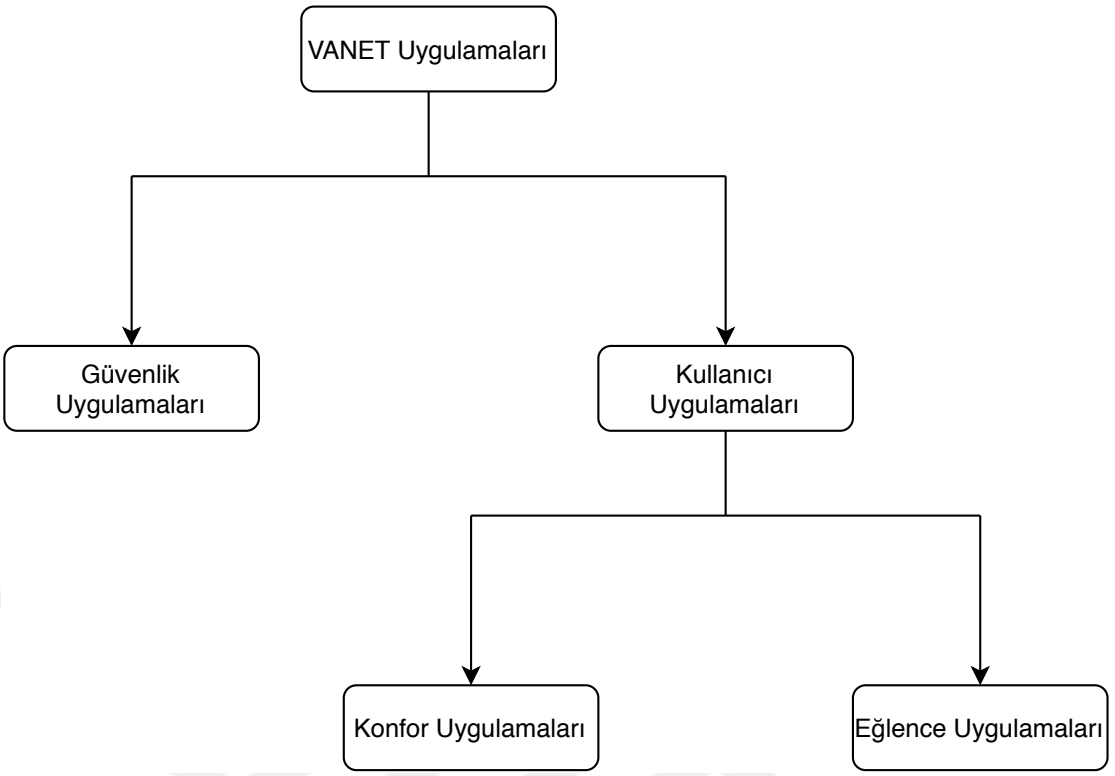
3.2.4 Yönlendirme Protokolleri

VANET teknolojisi ile mesaj alışverişinde yaşanılan gecikmelerin azaltılması hedeflenmektedir ve ağdaki katılımcıların arasında en uygun yola ulaşılması yönlendirme protokolleri ile gerçekleşmektedir. Ancak VANET'de araçlar sürekli hareket halinde olduğu için bazı yönlendirme protokolleri geliştirilmiştir. Bu yönlendirme protokolleri aşağıdaki gibi beş kategoride sınıflandırılmıştır [38]:

- *Topoloji bazlı yönlendirme:* Bağlantı bilgileri ile gelen paketler bir sonraki düğüme gönderilir. Bağlantı durumuna göre paketler gönderilir. Proaktif, reaktif ve hibrit yönlendirme olmak üzere üçe ayrılmaktadır.
- *Konum bazlı yönlendirme:* Coğrafi protokoller olarak da bilinmektedir. Konum bazlı yönlendirme yapılırken düğümün ve yakınındaki düğümlerin konumları GPS veya dijital haritalardan alınmaktadır.
- *Küme tabanlı yönlendirme:* Küçük araç grupları ile bir ağ mimarisi oluşturmayı hedeflemektedir. Kümenin boyutu, bir kümedeki araç sayısına veya araçların coğrafi konumuna bağlı olabilen yönlendirme algoritmasına bağlıdır.
- *Geo-cast yönlendirme:* Konum bazlı birçok noktaya yayın yönlendirmesi yapmaktadır. Bir düğümün bitişiğindeki düğümlere paketler iletmektedir.
- *Yayın bazlı yönlendirme:* Acil durum uyarılarını, trafik sıkışıklığı gibi mesajları iletmek için kullanılır.

3.2.5 VANET Uygulamaları

VANET uygulamaları amaçlarına göre Şekil 3.12'deki gibi kullanıcı ve güvenlik uygulamaları olmak üzere iki temel başlık altında gruplanabilir. Güvenlik uygulamaları yol güvenliğini artırmaya yönelik geliştirilen uygulamalardır. Kullanıcı uygulamaları da konfor ve eğlence uygulamalarını kapsamaktadır [7].



Şekil 3.12: VANET uygulamalarının sınıflandırılması.

3.2.5.1 Güvenlik Uygulamaları

Temel amacı yol güvenliğini sağlayıp can ve mal kaybını olabildiğince en aza indirgemektir. Bunun için geliştirilen güvenlik uygulamaları ile tehlike içeren alanlara yaklaşan araca daha önceden haber verilerek kazalar önlenmek istenmektedir. Kötü niyetli birisinin olması durumunda da insan hayatını tehdit etmeyecek şekilde bu kategorideki uygulamaların güvenliğinin sağlanması gerekmektedir. Aşağıdaki uygulamalar örnek olarak gösterilebilmektedir:

- Tehlikeli yollar için keskin viraj gibi uyarılar veren trafik işareti uygulamaları
- Tehlikeli kavşak, kör nokta, şerit ihlali, adaptive cruise kontrolü gibi yol durumlarında olası çarpışmaları engelleyecek uygulamalar
- Araç arıza tespiti için uygulamalar
- Kaza öncesi ve sonrası uyarı veren uygulamalar

3.2.5.2 Kullanıcı Uygulamaları

Temel amacı yolcu ile sürücünün konforunu artırmak ve trafik verimliliğini iyileştirmektir. Sürücülere ve yolculara hava durumu ve trafik bilgileri paylaşılabilir. Bununla birlikte ayrıca araca en yakın yemek yenilebilecek yerleri, dinlenme tesislerini, benzin istasyonu bilgilerini verebilirler. Yolcular çevrimiçi oyunlar oynayabilir, internete erişebilir ve hatta ofis işlerini bile yapabilirler. Aşağıdaki uygulamalar örnek olarak gösterilebilmektedir:

- Elektronik ödemeler için uygulamalar
- Trafik sıkışıklığı bildirimi yapan uygulamalar
- Park alanı vs. bulmak için bildirim yapan uygulamalar
- Araç bakımı gerektiğinde bildirim yapan uygulamalar
- İnternet, oyun vs. sağlayan uygulamalar

3.3 Kriptografik Akümülatörler

1993 yılında akümülatörler ilk kez Josh Benaloh ve Michael de Mare [18] tarafından ortaya atıldı. Tek yönlü fonksiyon olma özelliğine sahip olan özet fonksiyonları gibi benzer nitelikte olup, yarı değişmeli olarak isimlendirilen ek ve ayırıcı bir özelliği ile öne çıkarılarak tanımlanmıştır [21]. Özet fonksiyonları herhangi bir uzunlukta olan verileri sabit uzunlukta bir veriye dönüştürür ve veride tek bir nokta bile değişse hesaplanan özet değeri de değişmektedir. Özet fonksiyonları çakışmaya karşı dayanıklı olup aynı özet değerine sahip farklı verileri bulma ihtimali çok düşüktür ve tek yönlü olup, özet değerinden veriler elde edilemez.

Tek yönlü özet fonksiyonlarında çoğunlukla tek bir parametre ile fonksiyon tanımlaması yapılmaktadır. Ancak akümülatör yapısında özet fonksiyonları iki parametre alarak benzer boyutta sonuç oluşturan fonksiyonlar olarak biraz farklı bir şekilde kullanılmaktadır. Dolayısıyla sahip olunan bu farklılık ile tek yönlü akümülatörlerin sahip olduğu yarı değişmeli özelliğin tanımlamasının yapılması sağlanmaktadır [18]. Bu doğrultuda;

$\forall$ x $\in$ X ve y $\in$ Y için h: X x Y $\rightarrow$ X olarak tanımlanan h özet fonksiyonunun yarı değişmeli özelliğinin aşağıdaki gibi ifade edilmesine izin verilmektedir.

$$z = h(h(x, y_1), y_2) = h(h(x, y_2), y_1) \tag{3.1}$$

Bu çalışmadaki akümülatör yapısı modüler üs almayı kullanmaktadır ve yaygın olarak kullanılan RSA varsayımı ile $e_n(x, y) = x^y mod n$ şeklinde yazılabilmektedir ve bu durum yarı değişken özelliğe sahip olduğunu açık bir biçimde göstermektedir.

Dolayısıyla $h(h(x, y_1), y_2) = h(h(x, y_2), y_1)$ eşitliği $x^{y_1 y_2} = x^{y_2 y_1}$ olarak ifade edilebilmektedir. İlk değeri ya da seed değeri olarak bilinen x$\in$ X ve kripto sistemin kullanıcıları ile ilişkili olan $y_1, y_2, y_3, ...y_m \in Y$ olarak belirtilen değerler kümesinin biriktirilmiş özet değeri (accumulated hash value) aşağıda belirtilen ifade ile hesaplanabilmektedir ve y_i değerler kümesinin sırası değişse bile biriktirilmiş özet değeri değişmemektedir. Dolayısıyla akümülatör değeri y_i değerler kümesinin sırasına bağlı bulunmamaktadır.

$$z = h(h(h(...h(h(h(x, y_1), y_2), y_3), ..., y_m - 2), y_m - 1), y_m) \tag{3.2}$$

Ayrıca h olarak gösterilen tek yönlü özet fonksiyonlarında $x \in X$ ve $y \in Y$ verildiğinde $y' \in$ Y için $h(x, y) = h(x', y')$ eşitliğinde $x' \in X$ bulmak zordur. Bir kullanıcının belirli bir y_j'si varsa ve $i \neq j$ koşuluyla birlikte y_i kümesinde tüm elemanlar toplanarak y_j'ye karşılık gelen bir tanık değeri bulunmaktadır. Kullanıcı ihtiyacı olduğunda $h(w_j, y_j) = z$ ifadesini kontrol ederek sabit bir akümülatör değeri doğrulanıp bu kümede üye olduğunu kanıtlayabilmektedir. Bu nedenle akümülatörler bir başka elemanın bilgisini ortaya çıkarmadan ve her yeni dahil olan üyenin sırasına bakılmadan değerler kümesinde üye olduğunu kanıtlayabilecek etkili bir mekanizmadır.

3.3.1 Akümülatör Yapısının Gelişimi ve Türleri

Akümülatör çalışması Benaloh ve De Mare tarafından öne sürüldükten sonra tek yönlü özelliğinin yeterli olmamasından dolayı 1997 yılında Baric ve Pfitzmann [39] tarafından güçlü RSA varsayımı ile çakışmaya karşı dayanıklı bir akümülatör yapısı ortaya koyuldu. 2002 yılında Camenisch ve Lysyanskaya [19] tarafından yapılan çalışmada dinamik akümülatör olarak adlandırılan akümülatör yapısı ile verimli bir şekilde akümülatöre eleman ekleme ya da silme işlemi yapılabilmektedir. 2007 yılında Li, Li ve Xue [20]

tarafından öne sürülen çalışmada da dinamik evrensel akümülatör yapısı ile verimli bir biçimde hem üye hem de üye olmayan kanıtların oluşturulması sağlandı.

Kriptografik akümülatör yapısının gelişimi ile birlikte birçok akümülatör şemaları bulunmaktadır ama temelde RSA tabanlı akümülatör, bilinear tabanlı akümülatör, Merkle ağacı gibi akümülatör yapıları bulunmaktadır.

RSA akümülatörün [39] yapısı RSA modülü (güçlü iki asal olan p ve q için N = p x q) ile modüler üs alma esasına dayanmaktadır. Asal sayılardan oluşan X = $x_1, x_2, x_3...x_m$ değerler kümesinin üs biçiminde yazılarak ve $g \in QR_N$ yani N'in ikinci dereceden kare kalanı (quadratic residue) olarak temsil edilerek akümülatör değeri aşağıdaki gibi hesaplanmaktadır:

$$A = g^{x_1 x_2 x_3 ... x_m} mod N \tag{3.3}$$

Bununla birlikte x_i'ye karşılık gelen tanık değeri için ise x_i değeri dahil edilmeyerek aşağıdaki gibi hesaplanmaktadır:

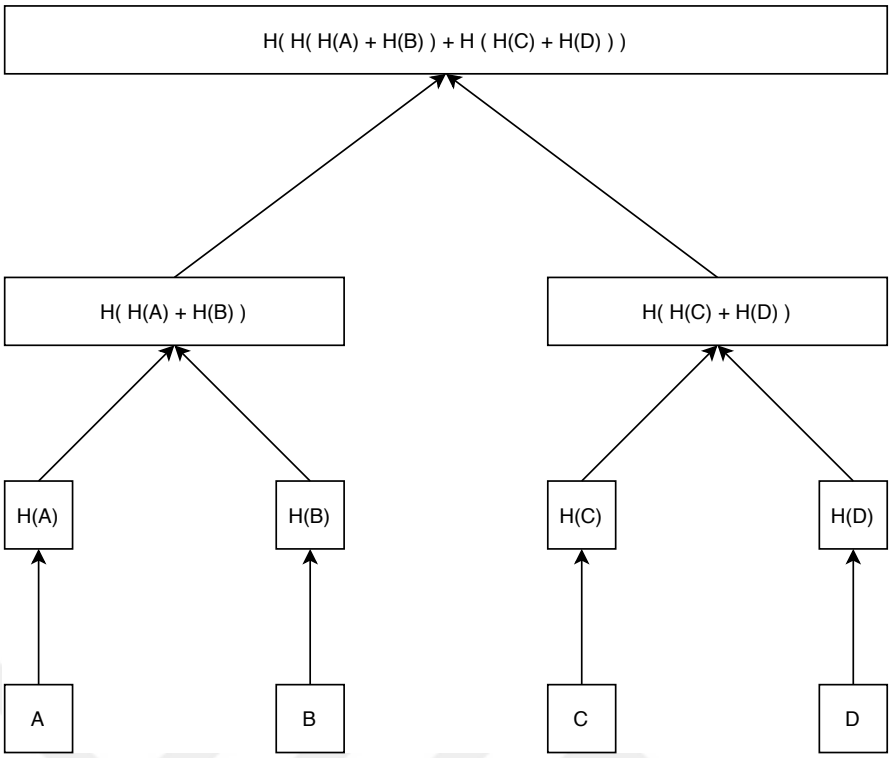
$$w_i = g^{x_1 x_2 x_3 ... x_{i-1} x_{i+1} x_{i+2} ... x_m} mod N \tag{3.4}$$

Bu durumda üyelik testi ${w_i}^{x_i} = A$ eşitliği sağlanarak doğrulanmaktadır.

2005 yılında Nguyen tarafından [26] bilinear tabanlı akümülatör çalışması ortaya konuldu. Bilinear tabanlı akümülatörün yapısı eliptik eğri işlemlerine dayanmaktadır. Akümülatörün güvenliği q-Güçlü Diffie Hellman(q-sDH) varsayımına dayanır ve q biriktirilecek (accumulated) eleman sayısına bir üst sınırdır.

Bilinear çifti olarak iki grup esas alındığında toplamsal gösterim olarak G_1, çarpımsal gösterim olarak G_2 ele alındığında ve G_1'in P ve Q olmak üzere iki üreteci olduğu varsayılırsa e : G_1 x $G_1 \rightarrow G_2$ şeklinde ifade edilmektedir [25]. Grupların asal sırasında olup rasgele seçilen s için akümülatör değeri A = $g^{\prod_1^m (s+e_i)}$ şeklinde hesaplanmaktadır.

Buna ilaveten Merkle ağacı veri yapısı da bir çeşit akümülatördür. Kimlik doğrulama mekanizmalarında genellikle kullanılan Merkle ağacında Şekil 3.13'de gösterildiği gibi kök düğüme ulaşana kadar tek bir özet değeri elde edilir. Ağaç veri yapısı ile üye/üye olmayan kanıtlar ve güncellemeler logaritmik olarak gerçekleşmektedir.



Şekil 3.13: Sıralı merkle ağacı.

3.3.2 Evrensel Akümülatör

Evrensel akümülatörler dinamik akümülatör yapısının genişletilerek üye olmayan tanık desteğinin eklenmesi ile kanıtlar sağlayarak veri kümesinde biriktirilmediğini göstermektedir. Üye olmayan tanık özelliğinin bulunması uygulamaların performansı açısından önemlidir ve verimliliğini artırmaktadır. Örneğin SM'ler milyonlarca kullanıcıya sertifika verebilmektedir ve sisteme katılan yeni kullanıcılar da sürekli artmaktadır. Dolayısıyla bu durum tanık değerinin çoğu zaman güncellenmesini gerektirmektedir. Sürekli artarak geçerli sertifikası olan kullanıcı listesi ile üye tanık değerini sıklıkla güncellemek yerine sık sık olmayan ve bazı nedenlerden dolayı sertifikası iptal edilmesi gereken geçersiz kullanıcı listesi için üye olmayan tanık değeri daha az güncelleneceğinden dolayı sertifika durum kontrolü bu yöntemle yapılmalıdır. Böylelikle hem sertifika iptal listesinin boyutu daha az olacaktır hem de kullanıcı, üye olmayan tanık değeri ile sertifikasının iptal edilmediğini verimli bir şekilde kanıtlayabilecektir. Yapılan bu çalışma [20] ile girdi kümeleri ailesi X_k olarak tanımlanan güvenli akümülatör aşağıdaki özellikleri içeren bir F_k fonksiyon ailesidir:

– *Verimli Üretim:* Rasgele seçilen bir G algoritması bulunmalıdır. 1^k parametresi

alan G algoritmasının çıktısı olarak $G(1^k) \rightarrow$ f $\in F_k$ elde edilmektedir ve ayrıca k uzunluğunda rasgele bir n modülü (iki asal sayı olan p ve q'nun çarpımı) üretilir. Böylelikle G algoritması f için aux_f olarak gösterilen yardımcı (auxiliary) bilgiyi de üretmektedir. aux_f bilgisi n'in çarpanlara ayrılmasıdır.

– *Verimli Değerlendirme:* Polinom zamanda gerçeklenebilen f $\in F_k$ fonksiyonu için girdi (g, x) $\in G_f$ x X_k olup çıktısı h $\in G_f$ değeridir. f fonksiyonunun girdi alanı G_f ve X_k olarak temsil edilmektedir ve X_k akümülatörde biriktirilecek değerler kümesidir.

– *Yarı-Değişmeli - Quasi-Commutative:* Bütün f $\in F_k$ ve g $\in G_f$ değerleri için; $f(f(g, x_1), x_2) = f(f(g, x_2), x_1)$ olarak yazılabilir ve X kümesi $x_1, x_2, x_3...x_m \subset X_k$ olarak belirtilirse aşağıdaki gibi gösterilmektedir.

$$f(g, X) = f(f(...(g, x_1), ...), x_m) = g^{x_1 x_2 x_3 .....x_m} modn \tag{3.5}$$

– *Üye Tanık:* f fonksiyonuna karşılık gelen doğrulama fonksiyonu p_1 olarak ifade edildiğinde $p_1(c, x, c_x) = 1$ olmalıdır. x $\in X_k$ elemanının dahil olmadığı $c_x \in G_f$ değer kümesi x için üye tanıktır. Böylelikle $(c_x)^x =$ c eşitliği sağlandığında kanıtlanmış olmaktadır.

– *Üye Olmayan Tanık:* f fonksiyonuna karşılık gelen doğrulama fonksiyonu p_2 olarak ifade edildiğinde $p_2(c, x, a, d) = 1$ olmalıdır. $\prod_1^m x_i = x_1 x_2 x_3 ..... =$ u olarak ifade edildiğinde f(g, X) $= g^u modn =$ c olarak akümülatör değeri hesaplanmaktadır. Öklid algoritması iki değerin en büyük ortak bölenini (ebob) hesaplar ve genişletilmiş versiyonunda au + bx = ebob(u, x) olarak ifade edilebilmektedir ve Bezout katsayısı olan a, b hesaplanır. ebob(u, x) = 1 olmaktadır çünkü X kümesi asal sayı olmalıdır. Bu nedenle, aşağıdaki formatta eşitlik yazılabilmektedir.

$$c^a = g^{ua} modn = g^{1-bx} modn \tag{3.6}$$

g^{-b} ifadesi kısaca d ile temsil edilerek $c^a = d^x$g modn şeklinde gösterilebilmektedir ve x için (a, d) ikilisi üye olmayan tanıktır.

– Güvenlik: Her olasılıksal polinom zamanlı düşman A için aşağıdaki eşitlik ile $x \in X_k$ için geçerli üye/üye olmayan tanığın bulunmasının mümkün görünmediği söylenmektedir.

$\Pr[\mathrm{f} \leftarrow G(1^k); g \leftarrow RG_f; (x, c_x, a, d, X) \leftarrow A_k(f, G_f, g) : x \in X_k; X \subset X_k; p_1(f(g, X), x, c_x) = 1; p_2(f(g, X), x, a, d) = 1] = \mathrm{neg(k)}$

Hem dinamik yapısı ile ekleme/silme işlemlerini destekleyen hem de üye olmayan kanıt sunan dinamik evrensel akümülatör [20] için ise tanımlanan özellikler aşağıdaki gibidir:

- *Akümülatörün Verimli Güncellenmesi:* X değerler kümesinde bulunmayan yeni bir $\hat{x}$ değeri ekleneceği zaman c olarak gösterilen akümülatör değerinin güncellenmesi gerekmektedir. Yeni akümülatör değeri $\hat{c}$ olarak gösterilerek $\hat{c} = c^{\hat{x}}$ modn şeklinde hesaplanmaktadır. X değerler kümesine eklenen $\hat{x}$ elemanının akümülatörden silinmesi gerektiğinde ise aux bilgisi $\phi(n)$ olan güncellenen yeni akümülatörün değeri $\hat{c} = c^{\hat{x}^{-1} mod \phi(n)}$ modn şeklinde hesaplanmaktadır.
- *Üye Tanığın Verimli Güncellenmesi:* $\mathrm{x} \in X_k$ için üye tanık değerinin w, akümülatör değerinin c olduğu ve yeni bir $\hat{x}$ değerinin eklendiği $\mathrm{x} \neq \hat{x}$ durumu için güncellenmesi gereken yeni üye tanık değeri $\hat{w} = f(w, \hat{x}) = w^{\hat{x}}$ modn şeklinde hesaplanmaktadır ve bu ifade $p_1(\hat{c}, x, \hat{w}) = 1$ eşitliği ile doğrulanmaktadır. $\hat{x}$ elemanının silindiği durumdaysa, W_1 algoritması a ve b diye iki sayı seçer. Böylece yeni üye tanık $ax + b\hat{x} = 1$ eşitliğine göre $\hat{w} = w^b \hat{c}^a$ modn şeklindedir ve doğrulama aşağıdaki ifade ile yapılabilmektedir:

$$\hat{w}^x = (w^b \hat{c}^a)^x = ((w^b \hat{c}^a)^{x\hat{x}})^{1/\hat{x}} = (c^{b\hat{x}} c^{ax})^{1/\hat{x}} = \hat{c}(modn) \tag{3.7}$$

- *Üye Olmayan Tanığın Verimli Güncellenmesi:* x için üye olmayan tanık (a, d) ikilisi olduğu farz edilsin. Yeni bir $\hat{x}$ değerinin eklendiğinde yeni üye olmayan tanık $(\hat{a}, \hat{d})$ aşağıdaki ifade ile hesaplanabilmektedir. W_2 algoritması $\hat{a}_0$ ve $\hat{r}_0$ diye iki sayı seçerse x ve $\hat{x}$ değerleri aralarında asal olacağı için $\hat{a}_0\hat{x}_0 + \hat{r}_0 x = 1$ eşitliği yazılır ve her taraf a ile çarpıldığında $\hat{a}_0 a\hat{x} + \hat{r}_0 ax = a$ denklemi elde edilir. Bu denkleme göre $\hat{a} = \hat{a}_0 a$ modx şeklinde hesaplanmaktadır ve $\hat{a}\hat{x} = a + rx$ olacak şekilde r bulunur. W_2 algoritması $\hat{d} = dc^r$ modn eşitliğini hesaplar ve doğrulama aşağıdaki ifade ile yapılabilmektedir:

$$\hat{c}^{\hat{a}} = c^{\hat{a}\hat{x}} = c^{a+rx} = c^{rx} c^a = c^{rx} d^x g = (dc^r)^x g = \hat{d}^x g(modn) \tag{3.8}$$

$\hat{x}$ üyesi silindiğinde ise yeni üye olmayan tanık olan $(\hat{a}\hat{d})$ çiftinin hesaplanabilmesi için W_2 algoritması r diye bir sayı seçerek $\hat{a} = a\hat{x} - rx$ ve $\hat{d} = d\hat{c}^{-r}$ modn şeklinde hesaplanmaktadır ve aşağıdaki ifade ile doğrulanabilmektedir.

$$\hat{c}^{\hat{a}} = \hat{c}^{\hat{a}\hat{x}-rx} = c^a\hat{c}^{-rx} = d^x g\hat{c}^{-rx} = \hat{d}^x g(modn) \tag{3.9}$$

3.3.3 Taahhüt Şeması ile Evrensel Aküm ülatör

Bir kullanıcı sertifikasının iptal edilmediğini kanıtlayabilmek için sertifika seri numarasını göstererek ve üye olmayan tanık değerini kullanarak akümülatörde birikmediğini kanıtlar ancak böylelikle kullanıcının sertifika seri numarası dolayısıyla kimliği de açığa çıkmaktadır. Bu yüzden evrensel akümülatör [20] çalışmasında anonimliği sağlayabilmek için sıfır bilgi protokollerinin yapıtaşı olan taahhüt (commitment) şeması ile bir değerin akümülatörde üye olmadığı kanıtlanabilmektedir. Buna göre öncelikle kullanıcı sertifika seri numarasını taahhüt eder daha sonra taahhüt edilen seri numarasının iptal listesinde biriktirilmediğini kanıtlar. Taahhüt şemaları bir değerin daha sonra açık edilebilecek şekilde geçici olarak saklı tutulmasını sağlamaktadır ve kullanım amacı aslında kanıtlayıcının sıfır bilgi kanıtını hesaplamasına izin vermektir. Yani temel prensip olarak bir bilgiyi alıcıya vermeden kanıtlayıcının bilgisini ispat etmesi amaçlanmaktadır. Taahhüt ve açığa çıkarma aşaması olmak üzere iki adımdır:

1- Taahhüt aşaması: Gönderici mesajı saklar ve taahhüt eder.

2- Açığa çıkarma aşaması: Taahhüt edilen mesaja dair gönderici kanıtını sunarak alıcıyı ikna etmektedir.

Taahhüt şemalarının gizleme (hiding) ve bağlama (binding) özelliklerini sağlaması gerekmektedir. Yani, kötü niyetli herhangi birisi taahhüt edilen mesajla ilgili bilgi edinememelidir ve gönderici kanıt sunarken bir başka mesaj için alıcıyı ikna edememelidir. Ayrıca taahhüt edildikten sonra gönderilen mesaj değiştirilemezdir. Geliştirilen taahhüt şemasındaki [40, 41] parametreler (n_1, g_1, h_1) şeklindedir ve k_1 uzunluğunda RSA modülünü n_1 , ikinci dereceden kare kalan olan rasgele bir değeri $h_1 \in QR_{n1}$ ve h_1 tarafından oluşturulan gruptaki rasgele değeri g_1 temsil etmektedir. x değeri taahhüt edilmek istenildiğinde rasgele bir r seçilir ve (x, r) = $g_1^x h_1^r$ mod n şeklinde taahhüt hesaplanır. Taahhüt şeması ile evrensel akümülatör çalışmasında [20] taahhüt edilen c_1 değerinin

c akümülatöründe birikmediğini kanıtlamak için sunulan sıfır bilgi ispati protokolünde $c_1, n_1, g_1, h_1, c, n, g, h$ girdileri ve buna ek olarak ispatlayan kişi için x, r, a, d girdileri vardır. a ve d değerleri daha önce de bahsedilen evrensel akümülatör bölümündeki gibi akümülatör c için üye olmayan tanıktır. $c_1 = g_1^x h_1^r \bmod n_1$ eşitliğindeki x değeri c_1'in taahhüt(committed) edilen değeridir ve $c^a = d^x g \bmod n$ eşitliğindeki x değeri c akümülatöründe biriktirilmediğini göstermektedir.

3.3.4 Uygulama Alanları

Akümülatörler birçok uygulamada kullanılmaktadır ve temelde amacı bant genişliği gereksinimini en aza indirmek ve mahremiyeti korumaktır. Bu kapsamda ilk olarak akümülatörler AAA kullanarak veri bütünlüğünü koruyan ve belirli bir verinin belirtilen bir tarihte mevcut olduğunu kanıtlayan zaman damgaları için ve daha sonrasında üyelik testi için önerilmiştir [42]. Kullanıcı ihtiyacı olduğunda tüm üye listesinin açığa çıkmasına gerek kalmadan ilişkili değerin gruba ait üye olup olmadığını kanıtlayabilmektedir. Aynı zamanda akümülatör şemaları sertifika iptal yönetiminde ve artık grup imzalarının, anonim kimlik bilgilerinin iptali için de yaygın olarak kullanılmaktadır. Son zamanlarda, bir başka uygulama alanı olarak da akümülatörler Bitcoin kripto para biriminin uzantısı olan ve sıfır bilgi kanıtı ile gizlilik ve anonimlik sağlayan Zerocoin'de kullanıldı. Buna göre aşağıdaki Şekil 3.14'de kriptografik akümülatörlerin kullanıldığı örnek uygulama alanları gösterilmektedir [43].



Şekil 3.14: Akümülatör kullanım alanları örnekleri.

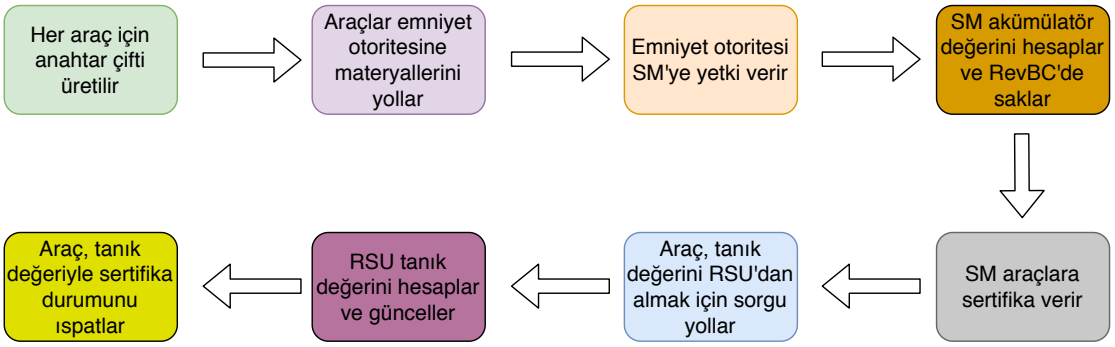
Bölüm 4

Sistem Tasarımı ve Bulgular

Bu bölümde, VANET için esas bir zorunluluk olan kimlik doğrulama sürecinde kullanılan sertifikaların, verimli bir şekilde geçerli olup olmadığının kontrolünün sağlanması için önerdiğimiz blokzincir ve akümülatör tabanlı çalışan sertifika iptal mekanizmasından bahsedilmektedir.

4.1 Önerilen Sistem

Daha önceki bölümde bahsedilen evrensel akümülatör fonksiyonu ile VANET'de araçların birbirlerine sertifikalarının geçerliliğini verimli bir şekilde kanıtlaması için Şekil 4.1'deki gibi adımlar sağlanarak üye olmayan testin yapılmasını önermekteyiz. Gerçek hayatta uygulanabilirliği açısından da daha kolay olacağını düşündüğümüz güvenli RSA varsayımı ile Li, Li ve Xue tarafından öne sürülen [20] dinamik evrensel akümülatör fonksiyonu kullanılarak, iptal edilen sertifikaların tek bir kısa değerde tutularak daha verimli bir sertifika doğrulama süreci gerçekleşecektir ve hesaplanan akümülatör değerinin blokzincire kaydedilerek sistemin güvenilirliği, şeffaflığı, takip edilebilirliği ya da erişilebilirliği kolaylaşacaktır. Ayrıca anonimliği sağlamak için taahhüt şeması kullanılarak kullanıcı önce sertifika seri numarasını taahhüt eder ve daha sonra taahhüt edilen seri numarasının iptal listesinde biriktirilmediğini kanıtlar [20] . Böylelikle bir araç sertifikasının diğer üyenin bilgilerini açığa çıkarmadan grubun üyesi olup olmadığı da kanıtlanabilecektir.



Şekil 4.1: Önerilen sistem için blok diyagramı.

Sistem kurulumunda SM seed değerini oluşturarak ilk akümülatör değerini oluşturur ve sistem ilklendirilirken her araç için anahtar çifti üretilir. Araçlar açık anahtarını, kişisel bilgilerini emniyet otoritesine iletirler. Bilgileri doğrulanan araçlar için emniyet otoritesi SM'nin sertifika üretmesine izin verir ve ağa katılan araç için sertifika üretilir. İptal edilmesi gereken sertifikalar akümülatörde biriktirilerek blokzincire yazılır. Her yeni sertifika iptal edildiğinde SM akümülatör değerini ve iptal blokzincirini güncellemelidir. RSU'lar da üye olmayan tanıkları güncellerler ve araçlara tanık değerlerini iletirler. Araçlar sertifikalarının geçerli olduğunu kanıtlamak istediklerinde sertifikasındaki bu benzersiz değere tanıklık ederler.

4.2 Sistem Modeli

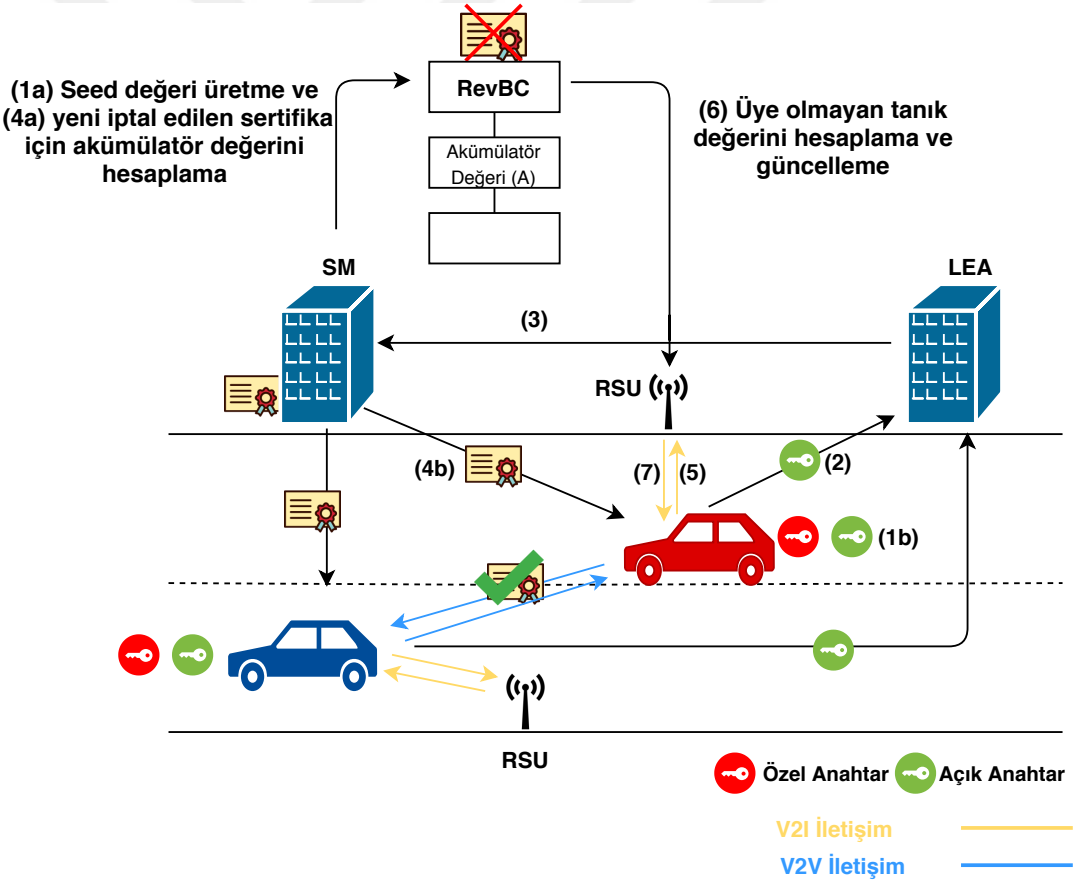
Önerilen sistem modeli Şekil 4.2'de gösterildiği gibidir ve tasarlanan sistemde yer alan varlıklar sertifika makamı, emniyet otoritesi, yol kenarı ünitesi ve araçlardan oluşmaktadır.

- *Sertifika Makamı (SM):* LEA denetimi altında sertifika yönetimini sağlayıp VANET ağındaki araçlara sertifikalarını verir ve ihtiyaç duyulduğunda sertifikalarını iptal eder. Şeffaf, değiştirilemez ve yüksek erişilebilirlik imkanları sağlayan izne tabi blokzincir yapısında iptal edilen sertifika listesini tek bir kısa değerde biriktirerek blokzincire kaydeder.
- *Emniyet Otoritesi (LEA):* Aracın davranışlarını denetler. Araç bilgisi ve ilişkili açık anahtarı doğrulayarak SM'nin araçlara sertifika vermesine izin verir. Aynı zamanda gerektiği durumlarda da sertifikanın iptal edilmesine izin verir. Olası

tartışmalar yaşanması durumu için sadece LEA'da aracın açık anahtarı ile ilişkili gerçek kimliği güvenli bir şekilde saklı tutulmaktadır.

- *Yol Kenarı Ünitesi (RSU):* Yollara yerleştirilen sabit ünitelerdir. Sertifikaların iptal edilmediği kanıtlanırken sunulan tanık değerleri kişisel bilgi paylaşılmadan RSU'dan alınmaktadır. RSU iptal blokzincirini (RevBC) alır ve üye olmayan tanık değerini hesaplar. Bu tanık değeri herhangi bir RSU'dan alınabilir. Akümülatöre yeni bir iptal edilen sertifika dahil olduğunda da tanık değerini günceller.

- *Araçlar:* Ağın katılımcılarıdır. Blokzincirinde tutulan akümülatör değeri ve hesaplanan tanık değeri ile sertifika durumunu kanıtlayabilmektedirler. Ayrıca, blokzincirinde kaydedilen tüm işlemleri, geçmiş kayıtları bile izleyebilir ve doğrulayabilmektedir. Böylelikle, otoritelerin yaptıkları işlemleri takip etmiş olurlar.



Şekil 4.2: Önerilen sistem modeli.

4.3 RSA Tabanlı Dinamik Evrensel Akümülatörün Uyarlanması

Evrensel akümülatörler ile verimli bir şekilde üye olmayan test gerçekleştirilmektedir ve üye olmayan tanık özelliğinin bulunması uygulamaların performansı açısından önemlidir. Geçersiz kullanıcı listesi için üye olmayan tanık değeri daha az güncelleneceğinden dolayı sertifika durum kontrolü bu yöntemle yapılmalıdır [20]. İptal edilen sertifikaların seri numaralarının biriktirildiği akümülatör değeri ile üye olmayan tanık değeri sunularak sertifikasının iptal edilmediği kanıtlanır.

Bunun için yeni bir eleman eklendiğinde akümülatörün güncellenmesi, tanık değerinin oluşturulması, tanık değerinin güncellenmesi, doğrulanması gibi gereken işlevler aşağıdaki Tablo 4.2'deki gibi kullanılmaktadır. Kullanılan algoritmalardaki sembollerin tanımları Tablo 4.1'te gösterildiği gibidir. Algoritmada kullanılan // sembolü yorum satırını temsil etmektedir.

Tablo 4.1: RSA tabanlı dinamik evrensel akümülatör algoritmalarında kullanılan gösterim.

Sembol	Tanım
V_i	Araç i
pk_{V_i}	V_i için açık anahtar
sk_{V_i}	V_i için özel anahtar
k	Güvenlik parametresi
A	Akümülatör değeri
A_0	İlk akümülatör değeri
g	Seed değeri
X	Biriktirilen elemanlar kümesi
nw_x	x'in üye olmayan tanık değeri

TABLO 4.2: VANET'de sertifika iptal durumu kontrolü için önerdiğimiz akümülatör yaklaşımının uyarlanması.

Algoritma 1:
anahtarUret()
1. **for** each vehicle V_i **do**
2. generate pk_{V_i}, sk_{V_i}
3. **end for**

Algoritma 2:
kurulum (k)
1. Choose p,q primes and $N \leftarrow pq$
2. $g \leftarrow_R \mathbb{Z}_N$ and $A_0 \leftarrow g$
3. **return** N, A_0

Algoritma 3:
uyeIptaliEkle (A_i, x_i)
1. **if** $x_i \in X$: **return** A_i
2. **else**:
3. Accumulate revoked certificate in X as:
   $X \leftarrow X \bigcup \{x_i\}$
4. Update $A \leftarrow A_i{}^{x_i}$ and update RevBC
5. **return** A

Algoritma 4:
uyeOlmayanTanikOlusturma (A, x)
1. RSU receives RevBC
   $u \leftarrow \prod_{i=1}^{m} x_i$ and $A \leftarrow g^u$
2. **if** $gcd(x, u) == 1$ **then**
   Find **Bézout** coefficients (x, u) // $ax + bu = 1$
3. **if** $A^b == g^{bu}$ **then** $d \leftarrow g^a$
4. **return** $nw_x \leftarrow \{d, b\}$

Algoritma 5:
uyeOlmayanTanikGuncelleme $(A, x, \hat{x})$
1. **if** RevBC is updated **then**
   RSU receives RevBC
2. **if** $gcd(x, \hat{x}) == 1$ and $\hat{x} \neq x$ **then**
   Find **Bézout** coefficients $(x, \hat{x})$ $//a_0x + b_0\hat{x} = 1$
   $a_0bx + b_0b\hat{x} = b$, $\hat{b} \leftarrow b_0b$ $//\hat{b}\hat{x} = b + ax$
3. **if** $A^{\hat{b}\hat{x}} = A^{b+ax}$ **then** $\hat{d} \leftarrow dA^a$
4. **return** $nw_x \leftarrow \{\hat{d}, \hat{b}\}$

Algoritma 6:
uyeOlmayanDogrulama (A, x, nw_x)
1. Download witness from any RSU
   $\{d, b\} \leftarrow nw_x$
2. **if** $(d^x\ A^b) = A_0$ **then**
3. **return** Non-revoked certificate
4. **else**
5. **return** Revoked certificate

Öncelikle ortamın kurulması için iki asal sayı olan p ve q değerleri oluşturularak k güvenlik parametresine denk gelen rastgele bir modül N değerinin oluşturulması gerekmektedir. Tablo 4.2'deki adım 1a için Algoritma 2 kullanılarak modül N değeri oluşturulur ve [0, N-1] aralığında olan rastgele bir sayı oluşturularak akümülatör için bir seed değeri oluşturulmaktadır. Bu çalışmada 112 bitlik güvenlik parametresine denk gelen 2048 bitlik RSA anahtar boyu seçilmiştir. Sistem ilklendirilirken araçların her birinin anahtar çiftine sahip olması gerekmektedir. Adım 1b için kullanılan Algoritma 1 ile araçların anahtar çiftine sahip olduğu gösterilmektedir. Adım 2'de aracın açık anahtarı ile birlikte kimlik bilgileriyle ilgili materyaller güvenli bir kanalla emniyet otoritesine iletilir ve kişisel bilgiler güvenli bir şekilde sadece emniyet otoritesinde saklanmaktadır. Adım 3'te bilgileri doğrulanan araçlar için emniyet otoritesi SM'ye yetki verir. Eğer yeni bir

iptal edilen sertifika varsa SM akümülatör değerini yeniden hesaplayıp RevBC'yi güncellemelidir. Adım 4a için Algoritma 3 uygulanarak akümülatöre yeni iptal edilen eleman dahil edilmektedir. Eğer biriktirilmiş değerlerin içerisinde yeni eklenecek elaman zaten bulunuyorsa akümülatör değeri aynı kalır. Ancak dahil edilecek eleman akümülatörde bulunmadığında ise tek defa kullanılan sayı olan nonce değeri elemana eklenmektedir. Daha sonra güvenlik parametresine göre SHA-256 özet algoritması için blok sayısı oluşturularak elemanın özet değeri alınır ve asallık testi yapılarak eleman asal olana kadar nonce değeri eklenmeye devam edilmektedir. Böylece asal hale getirilen iptal araç sertifikasının değeri akümülatöre dahil edilir ve mevcut akümülatör değerinin bu dahil edilen değer ile üssü alınarak yeni akümülatör değeri hesaplanır. Adım 4b'de SM araçlara sertifikasını vermektedir. Bir araç sertifikasının iptal edilmediğini kanıtlamak için ise karşılık gelen üye olmayan tanık değerinin oluşturulması gerekmektedir. Bunun için adım 5'te kişisel bilgiler sızdırılmadan tanık değerini elde edebilmek için RSU'ya sorgu yollanmaktadır. Algoritma 4 ile adım 6 gerçekleştirilip tanık değeri hesaplanmaktadır. Akümülatörde biriktirilen değerlerin asal sayılardan oluşması gerektiği için genişletilmiş Öklit algoritması ile Bezout katsayıları bulunarak üye olmayan tanıklar oluşturulmaktadır. Bunun için üye olmadığının kanıtlanması gereken elemanın dahil olmadığı diğer asal halde olan üyelerin biriktirilmiş özet değerlerinin çarpımının hesaplanması gerekmektedir. Asal hale getirilen üye olmayan değer ile hesaplanan çarpım sonucunun aralarında asal olması gerektiğinden dolayı Bezout katsayıları bulunur ve buna göre üye olmayan tanıklar oluşturulmaktadır. Benzer bir şekilde akümülatöre yeni bir eleman eklendiğinde akümülatör değerinin güncellenip Algoritma 5'teki gibi üye olmayan tanık değerinin de güncellenmesi gerekmektedir. Adım 6 ile hesaplanan ve güncellenen tanık değeri adım 7'de araçlara gönderilmektedir. Mesaj alışverişi yapılmadan önce aracın sertifikasının geçerli olduğunu kanıtlaması gerekmektedir. Algoritma 6'da RSU'dan indirilen üye olmayan tanık değeri ve blokzincirden alınan akümülatör değeriyle sertifikanın iptal edilmediği kanıtlanmaktadır. Güncel olan akümülatör değerinin b (üye olmayan tanık) ile üssü alınarak mod n'i hesaplanır. Bir diğer üye olmayan tanık olan d değerinin de asal hale getirilen üye olmayan değeri ile üssü alınarak mod n'i hesaplanır. Elde edilen iki değerin çarpımlarının sonucunun mod n'de kalanı akümülatör için oluşturulan ilk seed değerine eşit olduğunda bir elemanın üye olmadığı doğrulanmış olmaktadır. Buna göre Java dilinde yazılan üye olmayan testin kaynak kodunun bir kısmı aşağıda verilmiştir.

```
@Test

public void uyeOlmayanTest() {

RSAAkumulator rsaAkumulator = new RSAAkumulator();

BigInteger uye1 = new RandomUtil().nextBigInteger();
BigInteger uye2 = new RandomUtil().nextBigInteger();
BigInteger uye3 = new RandomUtil().nextBigInteger();

//x:uye olmayan eleman
BigInteger x = new RandomUtil().nextBigInteger();

BigInteger A0 = rsaAkumulator.getSeedDegeri();
BigInteger A1 = rsaAkumulator.uyeIptaliEkle(uye1);
BigInteger A2 = rsaAkumulator.uyeIptaliEkle(uye2);
BigInteger A3 = rsaAkumulator.uyeIptaliEkle(uye3);

Pair<BigInteger> pair = Util.hashToPrime(x);
BigInteger x_nonce = pair.getSecond();

BigInteger d = rsaAkumulator.uyeOlmayanTanikOlusturma(x, x_nonce).getFirst();
BigInteger b = rsaAkumulator.uyeOlmayanTanikOlusturma(x, x_nonce).getSecond();

boolean isValid = RSAAkumulator.uyeOlmayanDogrulama(A0, A3, d, b, x, x_nonce, rsaAkumulator.getN());
Assert.assertTrue(isValid);
}
```

LISTING 4.1: Önerilen sistem modeli için üye olmayan testin birim test kaynak kodu.

```
private final static int RSA_ANAHTAR_BOYU = 2048;
private final static int GUVENLIK_PARAMETRESI = 112;

private BigInteger A; //Akumulator Degeri
private BigInteger g; // Seed Degeri
private BigInteger N; // modul N

RandomUtil random;
//Akumulatorde biriken degerler
Map<BigInteger, BigInteger> biriktirilmisDegerler;

//Kurulum
public RSAAkumulator() {...}

public BigInteger getN() { return N; }
```

```
public BigInteger getSeedDegeri() {return g;}

public BigInteger getNonce(BigInteger x) {return biriktirilmisDegerler.get(x);}

public BigInteger uyeIptaliEkle(BigInteger x) {...}

public Pair<BigInteger> uyeOlmayanTanikOlusturma(BigInteger x, BigInteger x_nonce) {
    if (biriktirilmisDegerler.containsKey(x)) { return null; }
   else {
      {...}
      return new Pair<>(d, b);
   }
}

public static boolean uyeOlmayanDogrulama(BigInteger A0, BigInteger A_N, BigInteger d, BigInteger b,
                                          BigInteger x, BigInteger x_nonce, BigInteger n) {...}
```

LISTING 4.2: Önerilen sistem modeli için üye olmayan testin kaynak kodu.

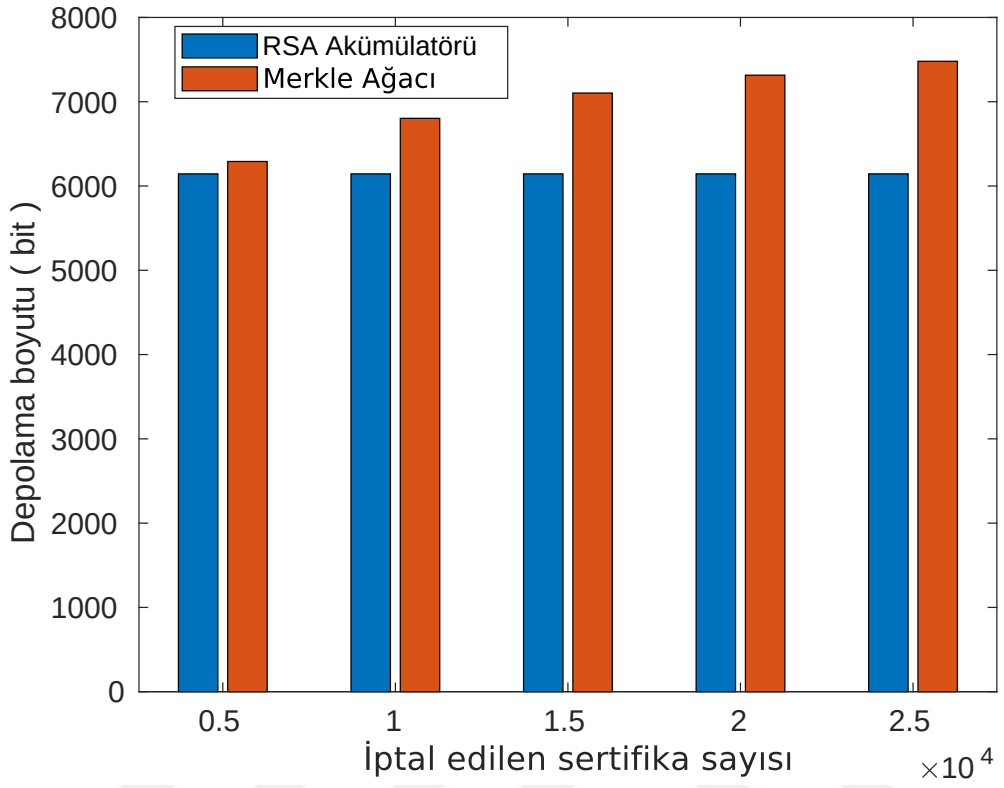
4.4 Performans Değerlendirmesi

Bu bölümde, VANET için önerdiğimiz sertifika iptal şeması diğer blokzincir tabanlı çalışmalarda önerilen sıralı Merkle ağaç veri yapısı ve geleneksel yaklaşım olan SİL dosyası kullanımıyla kıyaslandırılmaktadır ve yapılan analizler sunulmaktadır.

İlk olarak iptal edilen sertifikaların seri numaralarının bulunduğu geleneksel SİL dosyalarının boyutu ve iptal edilen açık anahtarların bulunduğu sıralı Merkle ağacının boyutuyla kıyaslama yapıldığında, iptal edilen her aracın sertifika sayısının artışıyla SİL dosyasının boyutu doğrusal olarak artmaktadır ve bu durumda alan karmaşıklığı O(n)'dir. Dolayısıyla indirilen SİL dosyasının boyutu iptal edilen araçların sertifika sayısına bağlıdır ve bellekte fazla yer kaplamaktadır. Ayrıca boyut arttıkça dosyanın indirilme süresi yavaşladığı gibi dosyanın açılması da yavaşlamaktadır. Örneğin, SM tarafından yayınlanan 237 Megabaytlık (MB) bir SİL dosyası indirildiğinde açılamamaktadır ve sertifika durumu kontrol edilememektedir. İptal listesinde eleman bulunmayan ve anahtar tipi RSA-2048, imzalama özet algoritması SHA-256 olan nitelikli elektronik sertifika (NES) için imzalanmış bir SİL dosyası yaklaşık olarak 525 bayt yer kaplamaktadır. İptal edilen sertifikanın seri numarasını ve iptal tarihini bulunduran SİL listesine eklenen her sertifika, dosyanın boyutunu yaklaşık 20-25 bayt olarak artırmaktadır. Diğer bir taraftan, iptal edilen n

tane aracın açık anahtarları n yapraklı sıralı Merkle ağacında tutulmaktadır. Böylece n + n/2 + n/4 +...+ 4 + 2 + 1 serisi şeklinde olup Merkle ağacındaki toplam düğüm sayısı 2n-1'dir. Dolayısıyla kullandığı alan n tane yaprak düğümlerinin sayısı ile doğrusal olarak artmaktadır ve alan karmaşıklığı O(n)'dir. Bu nedenle Merkle ağacının büyüklüğü de iptal edilen araçların açık anahtar sayısına bağlı bulunmaktadır ve bellekte fazla yer kaplamaktadır. Ayrıca Merkle ağacında üye - üye olmayan kanıtların da sabit boyutlu kanıtı bulunmamaktadır. Büyük ölçekli ağ olan VANET sistemlerinin iptal edilen araç sayısına bağlı olması sertifika yönetiminde bahsedilen problemlere yol açacaktır. Bu yüzden SİL ve Merkle ağacı yöntemlerinin kullanımı elverişli bulunmamaktadır. VANET'de sertifika iptal şemasında kullanılmasını önerdiğimiz dinamik evrensel akümülatör ile iptal edilen sertifikaların seri numaralarının toplandığı akümülatör değeri ve kanıtlar ise sabit boyutludur. Bundan dolayı araçların iptal edilen sertifika sayısına da bağlı bulunmamaktadır. Böylece durumsal olmayan blokzincir modeli ile depolama yükü azaltılmış olmaktadır.

Anahtar yönetimi için NIST tarafından yayınlanan bildiriye göre [44], imzanın güçlü olduğu kullanılan iki algoritmanın zayıflığına göre belirlenmektedir. 2048 bit anahtar kullanan RSA ile kullanılan SHA-256 algoritması 112 bit güvenlik sağlamaktadır çünkü 2048 bit RSA anahtarı yalnızca 112 bit güvenlik sağlar. 128 bit güvenlik gerektiğinde, 3072 bit RSA anahtarının uygun olduğu önerilmektedir. Bu durumda 128 bit güvenlik için [20] 3072 bitlik RSA anahtar boyutunun ve 256 bitlik özet algoritmasının kullanıldığı varsayılsın. Sıralı Merkle ağacında yaprak değerinden küçük ve büyük olan iki yaprağın üyeliğinin kantlanması gerekmektedir. Dinamik evrensel akümülatörde de daha önceki bölümde bahsedilen b ve d değerleri üye olmayan tanıklardır. Bu yüzden iki öğe bulunması gerekmektedir. 4096'dan fazla elemanı olan bir küme için Merkle ağacının üye olmayan kanıtının depolama boyutu $2 \times \lceil logN \rceil$ olarak hesaplanmaktadır. Böylece kümedeki eleman sayısına bağlı olarak logaritmik artan Merkle ağaç yolu ile $\lceil log(4097) \rceil$ = 13 özet elemanlarına gereksinim olacaktır. Her bir özet elemanı 256 bit büyüklükte olacağı için $2 \times 13 \times 256 = 6656$ bit yer kaplayacaktır. Bir RSA akümülatörü üye olmayan kanıtının depolama alanı ise 6.144 bit olacaktır. Yani bu durumda, dinamik evrensel akümülatörün kanıt boyutu sıralı Merkle ağacına göre daha az yer kaplayacaktır. Kıyaslanan kanıtların depolama maliyetleri aşağıdaki Şekil 4.3'de bulunan grafikteki gibidir.



Şekil 4.3: Üye olmayan kanıtların boyutunun iptal edilen sertifika sayısına göre karşılaştırılması.

Doğrulama maliyeti olarak SİL ve sıralı Merkle ağacı ile kıyaslama yapıldığında ise sertifikanın durumu SİL dosyasında araştırıldığında en kötü durum, iptal edilen sertifikaların sayısı ile doğrusal artarak O(n)'dir. Sıralı bir Merkle ağacında ise bir elemanın üye olmadığını yani açık anahtarının iptal edilmediğini kanıtlamak için hangi aralıkta olması gerektiğine göre komşu iki düğümün açığa çıkması gerekmektedir ve bu sayede kök doğrulanmaktadır. Dolayısıyla iptal edilen araçların sertifika sayısına bağlı olarak logaritmik artan üye - üye olmayan kanıt ve doğrulama maliyeti sağlamaktadır. Bu nedenle sıralı Merkle ağacının doğrulama maliyeti O(logn)'dir. Daha önceki bölümde de bahsedildiği üzere VANET'de kullanılması önerilen RSA tabanlı dinamik evrensel akümülatör kullanılarak $d^x A^b = g$ eşitliği ile üye olmayan doğrulama yapılmaktadır ve eleman sayısına bağlı olmayan dinamik evrensel akümülatörün bu durumda üye - üye olmayan kanıt boyutu sabit olup doğrulama maliyeti de sabittir. Sıralı Merkle ağacındaki gibi doğrulama yapılırken diğer elemanların açığa çıkmasını gerektirmez.

Ayrıca iptal edilen yeni bir araç sertifikası eklenileceği zaman güncelleme işlemi için SİL ve sıralı Merkle ağacı ile kıyaslama yapıldığında, SİL'de yapılan güncelleme işlemi doğrusal olup tüm SİL dosyası güncellenir. Sıralı Merkle ağacında ise güncelleme işlemi logaritmiktir çünkü Merkle ağacı sıralı olduğu için her defasında ağacın yeniden inşa edilmesi gerekmektedir. Bu yüzden, sıralı Merkle ağacı için güncelleme işlemi O(logn)'dir. Ancak, VANET gibi büyük ölçekli bir ağda her yeni eklenen eleman için bu durum bir problem oluşturacaktır çünkü üye - üye olmayan kanıtların büyümesiyle birlikte blokzinciri doğrularken pahalı kontroller yapılmasını gerektirecektir. Daha önceki bölümde bahsedildiği üzere VANET'de kullanılmasını önerdiğimiz dinamik evrensel akümülatörde güncelleme işlemi sabit maliyetli olup birikmiş kümenin boyutundan ve sırasından bağımsız bir şekilde yeni eleman kolaylıkla eklenebilmektedir. Akümülatördeki diğer elemanları bilmeden basit bir şekilde ekleme yapılabilmektedir. Bu yüzden akümülatöre yeni eleman eklenmesinin maliyeti O(1)'dir. Tablo 4.3'de SİL ve sıralı Merkle ağacı ile yapılan kıyaslamanın özeti bulunmaktadır.

Tablo 4.3: SİL, Sıralı Merkle ağacı ve RSA akümülatörün kıyaslanması.

Depolama, Doğrulama, Güncelleme Gereksinimleri			
	SİL	**Sıralı Merkle Ağacı**	**RSA Akümülatör**
Üye İptali (Ekleme)	O(N)	O(logN)	O(1)
Üye Tanık Oluşturma	X	O(logN)	O(1)
Üye Olmayan Tanık Oluşturma	X	O(logN)	O(1)
Üye Tanık Güncelleme	X	O(logN)	O(1)
Üye Olmayan Tanık Güncelleme	X	O(logN)	O(1)
Üyeliği Doğrulama Maliyeti	O(N)	O(logN)	O(1)
Üye Olmayanları Doğrulama Maliyeti	X	O(logN)	O(1)
Kanıt Boyutu	O(N)	O(logN)	O(1)
Özellikler			
Dinamik	X	✓	✓
İptal edilen kullanıcı mahremiyeti	X	X	✓
Sıfır Bilgi Ispatı	X	X	✓
Tek Yönlülük	X	✓	✓
Yarı Değişmeli	X	X	✓
Ölçeklenebilirlik	Az	Orta	Yüksek

Bölüm 5

Sonuçlar

VANET sistemlerinde güven yönetimi hala açık bir sorun olup, araçlar arasındaki bilgi alışverişinin güvenilirliği ve bütünlüğü hakkında güvenlik ve gizlilik endişeleri bulunmaktadır. Bu nedenle, ilk olarak VANET sistemlerinin temel gereksinimi olan kimlik doğrulama güvenliğinin sağlanması gerekmektedir. Bu yüzden, VANET ağındaki araçların kimliğinin kanıtlanmasında kullanılan elektronik sertifikaların doğrulanması en önemli gereksinimlerden biridir . Çalışma kapsamında hem blokzincir teknolojisinin hem de kriptografik akümülatör veri yapısının avantajları, kullanım yöntemi ve kullanım alanları hakkında literatür çalışması yapılmıştır. Araştırmaların sonucunda sertifika iptal bilgilerinin kontrol edilmesinde yaşanılan sorunlar üzerinde durularak VANET sistemlerinde sertifikaların iptal yönetimi ele alınmıştır ve mevcut problemler için hem blokzincir teknolojisinin hem de kriptografik akümülatör veri yapısının fayda sağlayabileceği ön görülmüştür. Önerilen sertifika iptal şemamız kimlik doğrulama güvenliği, koşullu anonimlik, sertifika iptal şeffaflığı, verimlilik ve ölçeklenebilirlik özellikleri ile öne çıkmaktadır.

Son yıllarda popülerliği artan blokzincir dağıtık defter teknolojisi yapısı ile şeffaf, değişmez, yüksek erişilebilir ve güvenilir bir ortam sağlanmaktadır. Akıllı ulaşım sistemlerinin temeli olan VANET sistemlerinde de blokzincir teknolojisinin bu özelliklerinden yararlanarak ağdaki işlemlerin şeffaflığından şüphe etmek pek mümkün olmayacaktır ve sertifika iptal bilgilerine de kolayca erişilenebilecektir. Bu nedenle sertifika iptal bilgileri, tasarımımızdaki iptal blokzincirinde(RevBC) tutularak otoritelerin yaptıkları işlemleri değiştirilemez, şeffaf hale getirir ve takip edilebilir kılar.

Herhangi bir anlaşmazlık durumu söz konusu olduğunda ise araçları takip edebilmek adına sistemin koşullu anonim olması gerekir. Bu nedenle tasarımda açık anahtarı ile ilişkili gerçek kimlik bilgisi güvenli bir şekilde sadece emniyet otoritesinde tutulmaktadır.

Ayrıca VANET sistemlerinde sertifika iptal kontrolü için kullanılan sertifika iptal listesi (SİL) ve iptal listesinin tutulduğu Merkle ağacı yaklaşımları yerine, bu çalışmada daha verimli bir çözüm yolu olarak kriptografik akümülatör yapısının kullanılması önerilmektedir. RSA tabanlı dinamik evrensel akümülatör yardımıyla iptal edilen sertifika listesinin tek bir kısa değerle blokzincirde tutulması önerilmiştir. Bu veri yapısı sayesinde, VANET sistemlerinde kısa sabit boyutlu kanıtlar kullanılarak sertifikaların iptal edilmediği verimli bir şekilde kanıtlanır. Dolayısıyla, sisteme ek yük getiren ve doğrulaması zaman alan geleneksel yaklaşımları tek yönlü akümülatör veri yapısı ile değiştirerek sertifika durum kontrol süreci iyileştirilir.

Bir sertifikanın iptal edilmiş sertifika listesine dahil olmadığını kanıtlamak için kullanılmasını önerdiğimiz RSA akümülatör yapısı Java dilinde implemente edilmiştir. Yapılan analizler sonucu SİL dosyasının boyutu iptal edilen sertifika sayısına bağlı olarak doğrusal bir şekilde artmaktadır. Dosya boyutu arttıkça sertifika makamının yayınladığı sertifika iptal listelerinin indirilme süresi de gecikmektedir ve sertifikanın doğrulanması zaman almaktadır. Dolayısıyla, SİL boyutu VANET sistemindeki ağ ölçeklenebilirliği nedeniyle önemli ölçüde büyüyecek ve böylece arama, dağıtım ve depolama bakımından büyük bir ek yük haline gelecektir.

Sertifikanın iptal edilmediğini kanıtlamak için iptal edilen anahtarların saklandığı sıralı Merkle ağacı yapısı ise geleneksel SİL yaklaşımına göre daha verimlidir. Sertifikanın ağaçta bulunmadığını yani iptal edilmediğini ispatlayan kanıtın depolama boyutu $2 \times \lceil logN \rceil$ olarak hesaplanmaktadır . Kriptografik akümülatör yapısı ile kıyaslandığında başlangıç için bu kanıtın depolama boyutunun daha az yer kapladığı görülmektedir ancak daha sonrasında iptal edilen her yeni elemanın sayısına ve sırasına bağlı olarak kanıt boyutlarının akümülatördeki kanıt boyutuna göre daha çok yer işgal ettiği görülmüştür. Ayrıca akümülatör kullanılarak sertifikanın iptal edilmediğini ispatlayan kanıt boyutlarının iptal edilen sertifika sayısına ve sırasına bakmadan sabit boyutlu olduğu görülmüştür. Dolayısıyla her geçen gün artan araç sayısına göre arama, depolama bakımından sıralı Merkle ağacı kullanıldığında da problemler ortaya çıkacaktır.

Sonuç olarak SİL, sıralı Merkle ağacı ve kullanılmasını önerdiğimiz RSA akümülatör yapısı kıyaslandığında; SİL'in depolama, doğrulama ve güncelleme maliyeti O(N)'dir. Sıralı Merkle ağacına ise iptal edilen sertifikanın eklenmesi, iptal edilmediğinin doğrulanması ve kanıt boyutunun artmasının maliyeti O(logN)'dir. Ancak, kullanılmasını önerdiğimiz kriptografik akümülatör yapısının kanıt boyutu, sertifikanın akümülatörde dahil olmadığının kanıtlanması ve akümülatörün güncellenmesinin maliyeti sabit olup O(1)'dir. Böylece akümülatör ile sertifikanın iptal edilmediğini kanıtlamak için kullanılan sabit boyutlu tanık değerleri ile verimliliğin iyileştirildiği görülmektedir.

VANET'de güvenli kimlik doğrulama için tasarladığımız bu sertifika iptal şeması ile hem akademik çalışmalara hem de sektöre katkı sağlayacağı düşünülmektedir. Gelecekte, önerilen bu sistemin gerçek araçlar üzerinde uygulanabilirliği sağlanabilir. Bu durum öncelikle implemente edilmesi daha kolay olan RSA tabanlı akümülatör ile gerçekleştirilebilir. Daha sonra uygulanabilirliği daha külfetli olsa da bilinear-map akümülatör ile gerçekleştirilebilir çünkü performansının diğer akümülatörlere göre bazı durumlarda daha verimli olabileceği düşünülmektedir [27].

Ek A

Java Kodu

Python dilinde yazılan [45] ve üyelik testi için Java dilinde yazılan [46] kaynaklar baz alınarak Java dilinde üye olmayan test implemente edilmiştir.

```
import org.junit.Assert;
import org.junit.Test;
import java.math.BigInteger;

public class TestRSAAkumulator {

  @Test
  public void uyeOlmayanTest() {
    RSAAkumulator rsaAkumulator = new RSAAkumulator();

    BigInteger uye1 = new RandomUtil().nextBigInteger();
    BigInteger uye2 = new RandomUtil().nextBigInteger();
    BigInteger uye3 = new RandomUtil().nextBigInteger();

    // x: uye olmayan eleman
    BigInteger x = new RandomUtil().nextBigInteger();

    BigInteger A1 = rsaAkumulator.uyeIptaliEkle(uye1);
    BigInteger A2 = rsaAkumulator.uyeIptaliEkle(uye2);
    BigInteger A3 = rsaAkumulator.uyeIptaliEkle(uye3);

    Pair<BigInteger> pair = Util.hashToPrime(x);
    BigInteger x_nonce = pair.getSecond();

    BigInteger  d = rsaAkumulator.uyeOlmayanTanikOlusturma(x,  x_nonce).getFirst();
    BigInteger  b = rsaAkumulator.uyeOlmayanTanikOlusturma(x, x_nonce).getSecond();

    BigInteger A0 = rsaAkumulator.getSeedDegeri();

    boolean isValid = RSAAkumulator.uyeOlmayanDogrulama (A0, A3, d, b, x, x_nonce,
rsaAkumulator.getN());
    Assert.assertTrue(isValid);
  }
}
```

```
import java.math.BigInteger;
import java.util.HashMap;
import java.util.Map;

public class RSAAkumulator {

  private final static int RSA_ANAHTAR_BOYU = 2048;
  private final static int GUVENLIK_PARAMETRESI = 112;

  private BigInteger A; //Akumulator Degeri
  private BigInteger g; // Seed Degeri
  private BigInteger N; // modul N

  RandomUtil random;
  Map<BigInteger, BigInteger> biriktirilmisDegerler; //Akumulatorde biriken degerler

  public RSAAkumulator() {
    biriktirilmisDegerler = new HashMap<>();
    Pair<BigInteger> bigIntegerPair = Util.ikiFarkliAsalUret(RSA_ANAHTAR_BOYU / 2);
    BigInteger p = bigIntegerPair.getFirst();
    BigInteger q = bigIntegerPair.getSecond();
    N = p.multiply(q);

    random = new RandomUtil();
    g = random.nextBigInteger(BigInteger.ZERO, N);
    A = g;
  }

  public BigInteger getN() { return N; }

  public BigInteger getSeedDegeri() { return g;  }

  public BigInteger getNonce(BigInteger x) { return biriktirilmisDegerler.get(x); }

  public BigInteger uyeIptaliEkle(BigInteger x) {
    if (biriktirilmisDegerler.containsKey(x))
    {
      return A;
    } else {
      Pair<BigInteger> bigIntegerPair = Util.hashToPrime(x, GUVENLIK_PARAMETRESI);
      BigInteger     hashPrime     = bigIntegerPair.getFirst();
      BigInteger     nonce         = bigIntegerPair.getSecond();

      A = A.modPow(hashPrime, N);
      biriktirilmisDegerler.put(x, nonce);
      return A;
    }
  }
```

```
public Pair<BigInteger> uyeOlmayanTanikOlusturma(BigInteger x, BigInteger x_nonce) {
    if (biriktirilmisDegerler.containsKey(x)) { return null; }
    else {
        BigInteger product = BigInteger.ONE;
        //x'in dahil olmadigi elemanlarin carpimi hesaplanir
        for (BigInteger k : biriktirilmisDegerler.keySet()) {
          BigInteger nonce = biriktirilmisDegerler.get(k);
          product = product.multiply(Util.hashToPrime(k, GUVENLIK_PARAMETRESI,
nonce).getFirst());
        }

        BigInteger prime = Util.hashToPrime(x, GUVENLIK_PARAMETRESI, x_nonce).getFirst();
        Triple<BigInteger> katsayilar = Util.bulBezoutKatsayilari(prime, product);
        BigInteger a = katsayilar.getSecond();
        BigInteger b = katsayilar.getThird();
        BigInteger d;

        if (a.intValue() < 0) {
          BigInteger pozitif_a = a.multiply(new BigInteger("-1"));
          BigInteger inverse_A0 = Util.inversion(g, N);
          d = inverse_A0.modPow(pozitif_a, N);
        } else {
          d = g.modPow(a, N);
        }
      return new Pair<>(d, b);
    }
  }

  public static boolean uyeOlmayanDogrulama(BigInteger A0, BigInteger A_N, BigInteger d,
                                            BigInteger b, BigInteger x,
                                            BigInteger x_nonce, BigInteger n) {
    BigInteger prime = Util.hashToPrime(x, GUVENLIK_PARAMETRESI, x_nonce).getFirst();
    BigInteger ikinci_pow;
    if(b.intValue() < 0){
      BigInteger pozitif_b = b.multiply(new BigInteger("-1"));
      BigInteger inverse_Afinal = Util.inversion(A_N, n);
      ikinci_pow =  inverse_Afinal.modPow(pozitif_b, n);
    }else{
      ikinci_pow = A_N.modPow(b, n);
    }
    return (d.modPow(prime, n).multiply(ikinci_pow)).remainder(n).equals(A0);
  }
}
```

```
import com.google.common.hash.Hashing;
import com.google.common.io.BaseEncoding;

import java.math.BigInteger;
import java.util.Random;

public class Util {
  private static final int ASAL_KESINLIK = 5;

  public static String toHexString(byte[] bytes) {
    return BaseEncoding.base16().lowerCase().encode(bytes);
  }

  public static BigInteger asalUret(int bitUzunlugu) {
    Random random = new java.util.Random();
    return BigInteger.probablePrime(bitUzunlugu, random);
  }

  public static Pair<BigInteger> ikiFarkliAsalUret(int bitUzunlugu) {
    BigInteger birinci = asalUret(bitUzunlugu);
    while (true) {
      BigInteger ikinci = asalUret(bitUzunlugu);
      if (birinci.compareTo(ikinci) != 0) {
        return new Pair<>(birinci, ikinci);
      }
    }
  }

  public static Pair<BigInteger> hashToPrime(BigInteger x, int bitUzunlugu) {
    return hashToPrime(x, bitUzunlugu, BigInteger.ZERO);
  }

  public static Pair<BigInteger> hashToPrime(BigInteger x) {
    return hashToPrime(x, 120, BigInteger.ZERO);
  }

  public static Pair<BigInteger> hashToPrime(BigInteger x, int bitUzunlugu, BigInteger initNonce) {
    BigInteger nonce = initNonce;
    while (true) {
      BigInteger num = hashToLength(x.add(nonce), bitUzunlugu);
      if (num.isProbablePrime(ASAL_KESINLIK)) {
        return new Pair<>(num, nonce);
      }
      nonce = nonce.add(BigInteger.ONE);
    }
  }
```

```
public static BigInteger hashToLength(BigInteger x, int bitUzunlugu) {
    StringBuilder randomHexString = new StringBuilder();
    int blokSayisi = (int) Math.ceil(bitUzunlugu / 256.00);
    for (int i = 0; i < blokSayisi; i++) {
      final BigInteger bigIntI = new BigInteger(Integer.toString(i));
      randomHexString.append(
          toHexString(Hashing.sha256().hashBytes((x.add(bigIntI)).toString(10)
              .getBytes()).asBytes()));
    }
    if (bitUzunlugu % 256 > 0) {
      randomHexString = new StringBuilder(randomHexString.substring((bitUzunlugu % 256) / 4));
    }
    return new BigInteger(randomHexString.toString(), 16);
  }
  public static Triple<BigInteger> bulBezoutKatsayilari(BigInteger a, BigInteger b){
    BigInteger x = BigInteger.ONE, y = BigInteger.ZERO, lastx = BigInteger.ZERO,
     lasty =   BigInteger.ONE, temp, temp1, temp2, temp3;
    if(b.compareTo(a) == 1) {
      temp = a;
      a = b;
      b = temp;
      while (b.intValue() != 0) {
        BigInteger q = a.divide(b);
        temp1 = a.remainder(b);

        a = b;
        b = temp1;

        temp2 = lastx;
        lastx = x.subtract(q.multiply(lastx));
        x = temp2;

        temp3 = lasty;
        lasty = y.subtract(q.multiply(lasty));
        y = temp3;
      }
      temp = x;
      x = y;
      y = temp;
    }else{
      while (b.intValue() != 0) { ... }
    }
    return new Triple<>(a, x, y);
  }
  public static BigInteger inversion(BigInteger b, BigInteger n){
    Triple<BigInteger> triple = bulBezoutKatsayilari(b, n);
    if(triple.getFirst().equals(BigInteger.ONE)) {
      return triple.getSecond().remainder(n);
    }
    return BigInteger.ZERO;
  }
}
```

Kaynaklar

[1] P. Baran, "On distributed communications networks," *IEEE Transactions on Communications Systems*, vol. 12, pp. 1–9, March 1964.

[2] M. Murat, "Blockchain ile güvenli elektronik sağlık sistemi," *İstanbul Teknik Üniversitesi*, 2018.

[3] S. Kardas, "Blokzincir teknolojisi: Uzlaşma protokolleri," *DÜMF Mühendislik Dergisi*, vol. 10, pp. 481–496, 06 2019.

[4] *Bilişim Teknolojileri Dergisi*, vol. 12, pp. 203 – 217, 2019.

[5] D. Meijer, "Consequences of the implementation of blockchain technology," *Delft University of Technology*, 2017.

[6] V. Jindal and P. Bedi, "Vehicular ad-hoc networks: introduction, standards, routing protocols and challenges," 2016.

[7] M. Saggi and R. Sandhu, "A survey of vehicular ad hoc network on attacks & security threats in vanets," 12 2014.

[8] M. Q. Alper Kaan Yıldırım, Ali Boyacı, "Araçsal ağların afet senaryolarındaki performanslarının karşılaştırılması," 2018.

[9] Z. Lu, W. Liu, Q. Wang, G. Qu, and Z. Liu, "A privacy-preserving trust model based on blockchain for vanets," *IEEE Access*, vol. 6, pp. 45 655–45 664, 2018.

[10] G. Khekare and A. Sakhare, "Intelligent traffic system for vanet: A survey," *International Journal of Advanced Computer Research*, vol. 2, pp. 99–102, 12 2012.

[11] A. Khan, K. Balan, Y. Javed, and S. Tarmizi, "Secure trust-based blockchain architecture to prevent attacks in vanet," *Sensors*, vol. 19, p. 4954, 11 2019.

[12] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Cryptography Mailing list at https://metzdowd.com*, 03 2009.

[13] R. Twesige, "Bitcoin a simple explanation of bitcoin and block chain technology january 2015 richard lee twesige," 01 2015.

[14] I. Mohmand and N. Javaid, "Blockchain based mobile nodes in vehicular network," 07 2019.

[15] A. Alrawais, A. Alhothaily, B. Mei, T. Song, and X. Cheng, "An efficient revocation scheme for vehicular ad-hoc networks," *Procedia Computer Science*, vol. 129, pp. 312–318, 01 2018.

[16] B. Öğretmen, "Çevrimiçi sertifika durum protokolü (ocsp)." [Online]. Available: http://www.kamusm.gov.tr/dosyalar/makaleler/Cevrimici%20Sertifika%20Durum%20Protokolu%20%20-%20OCSP.pdf

[17] A. Albarqi, E. Alzaid, F. Ghamdi, S. Asiri, and J. Kar, "Public key infrastructure: A survey," *Journal of Information Security*, vol. 06, pp. 31–37, 01 2015.

[18] J. Benaloh and M. D. Mare, "One-way accumulators: A decentralized alternative to digital signatures (extended abstract)," 1994.

[19] J. Camenisch and A. Lysyanskaya, "Dynamic accumulators and application to efficient revocation of anonymous credentials," pp. 61–76, 08 2002.

[20] J. Li, N. Li, and R. Xue, "Universal accumulators with efficient nonmembership proofs," vol. 4521, pp. 253–269, 01 2007.

[21] N. Fazio and A. Nicolosi, "Cryptographic accumulators: Definitions, constructions and applications," 01 2003.

[22] A. Kumar, P. Lafourcade, and C. Lauradoux, "Performances of cryptographic accumulators," 05 2014.

[23] Z. Lu, Q. Wang, G. Qu, H. Zhang, and Z. Liu, "A blockchain-based privacy-preserving authentication scheme for vanets," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 12, pp. 2792–2801, 2019.

[24] F. Bonomi, R. A. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," 2012.

[25] "Pprem: Privacy preserving revocation mechanism for vehicular ad hoc networks," *Computer Standards & Interfaces*, vol. 36, no. 3, pp. 513 – 523, 2014.

[26] L. Nguyen, "Accumulators from bilinear pairings and applications to id-based ring signatures and group membership revocation." *iACR Cryptology ePrint Archive*, vol. 2005, p. 123, 01 2005.

[27] E. Tremel, "Real-world performance of cryptographic accumulators."

[28] C. Satizábal, R. V. Páez, and J. Forné, "Pki trust relationships: From a hybrid architecture to a hierarchical model," p. 8 pp., 05 2006.

[29] C. Hernandez-Ganan, J. Munoz, O. Esparza, J. Mata, and J. Alins, "Epa: An efficient and privacy-aware revocation mechanism for vehicular ad hoc networks," *Pervasive and Mobile Computing*, vol. 21, 01 2014.

[30] C. Brijilalruban and B. Paramasivan, "Cluster-based secure communication and certificate revocation scheme for vanet," *Computer Journal*, vol. 62, pp. 263–275, 02 2019.

[31] K. Rabeh, M. Pan, Z. Han, and V. Ford, "Srpv: A scalable revocation scheme for pseudonyms-based vehicular ad hoc networks," pp. 1–6, 05 2018.

[32] A. Sarencheh, M. Asaar, M. Salmasizadeh, and M. Aref, "Rapp: An efficient revocation scheme with authentication and privacy preserving for vehicular ad-hoc networks," pp. 359–366, 10 2017.

[33] "ieee trial-use standard for wireless access in vehicular environments - security services for applications and management messages," *iEEE Std 1609.2-2006*, 2006.

[34] J. Haas, Y.-C. Hu, and K. Laberteaux, "Design and analysis of a lightweight certificate revocation mechanism for vanet," pp. 89–98, 01 2009.

[35] E. Ünsal and O. Kocaoğlu, "Blok zinciri teknolojisi: Kullanım alanları, açık noktaları ve gelecek beklentileri," *European Journal of Science and Technology*, pp. 54–64, 08 2018.

[36] W. Liang, Z. Li, H. Zhang, S. Wang, and R. Bie, "Vehicular ad hoc networks: Architectures, research issues, methodologies, challenges, and trends," *international Journal of Distributed Sensor Networks*, vol. 2015, pp. 1–11, 08 2015.

[37] V. R. Ankur Jain, Ratnesh Dubey, "Vehicular ad hoc network- architecture and operations," 09 2016.

[38] E. Dehghan and B. Canberk, "Weighted voting game based relay node managemnet in vanets," vol. 174, pp. 90–100, 05 2017.

[39] N. Barić and B. Pfitzmann, "Collision-free accumulators and fail-stop signature schemes without trees," pp. 480–494, 1997.

[40] i. Damgard and E. Fujisaki, "An integer commitment scheme based on groups with hidden order," 12 2001.

[41] E. Fujisaki and T. Okamoto, "Statistical zero knowledge protocols to prove modular polynomial relations." *iEiCE Transactions on Fundamentals of Electronics Communications and Computer Sciences*, vol. E82A, pp. 16–30, 1997.

[42] D. Derler, C. Hanser, and D. Slamanig, "Revisiting cryptographic accumulators, additional properties and relations to other primitives," *iACR Cryptology ePrint Archive*, vol. 2015, p. 87, 04 2015.

[43] P. Camacho, A. Hevia, M. Kiwi, and R. Opazo, "Strong accumulators from collision-resistant hashing," *international Journal of information Security*, vol. 11, pp. 471–486, 09 2008.

[44] W. B. W. P. M. S. Elaine Barker, William Barker, "Nist sp 800-57," pp. 62–64, 2012.

[45] O. Leiba, https://github.com/oleiba/RSA-accumulator.

[46] L. Huang, https://github.com/RayneHwang/rsa-accumulator-java.

ÖZGEÇMİŞ

Aylin AYDIN, İlk, orta ve lise eğitimini Ordu'da tamamladı. 2016 yılında Anadolu Üniversitesi Bilgisayar Mühendisliği(İngilizce) bölümünü bitirdi. 2013 yılında Erasmus programı kapsamıyla Polonya'nın Cracow University of Technology üniversitesinde değişim öğrencisi olarak yer almıştır. 2018 yılında Marmara Üniversitesi Bilgi Güvenliği Mühendisliği bölümünde yüksek lisans eğitimine başlamıştır. 2016 yılında Volt Enerji ve Bilişim teknolojileri firmasında yazılım mühendisi olarak işe başlamıştır. 18/12/2017 tarihinden itibaren TÜBİTAK Bilgem'de kariyerine devam etmektedir.