DÜZCE ÜNİVERSİTESİ | LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

BLOKZİNCİR TEKNOLOJİSİNDE İŞBİRLİĞİNE DAYALI AKILLI SÖZLEŞME MODELİNİN GELİŞTİRİLMESİ VE UYGULANMASI

Tunahan TİMUÇİN

DOKTORA TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

DANIŞMAN
DOÇ. DR. SERDAR BİROĞUL

DÜZCE, 2024

T.C.
DÜZCE ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

BLOKZİNCİR TEKNOLOJİSİNDE İŞBİRLİĞİNE DAYALI AKILLI SÖZLEŞME MODELİNİN GELİŞTİRİLMESİ VE UYGULANMASI

Tunahan TİMUÇİN tarafından hazırlanan tez çalışması aşağıdaki jüri tarafından Düzce Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda **DOKTORA TEZİ** olarak kabul edilmiştir.

Tez Danışmanı

Doç. Dr. Serdar BİROĞUL
Düzce Üniversitesi

Jüri Üyeleri

Doç. Dr. Serdar BİROĞUL
Düzce Üniversitesi _____

Prof. Dr. Uğur GÜVENÇ
Düzce Üniversitesi _____

Doç. Dr. Utku KÖSE
Süleyman Demirel Üniversitesi _____

Prof. Dr. Ali ÇALHAN
Düzce Üniversitesi _____

Doç. Dr. Hakan GÜNDÜZ
Kocaeli Üniversitesi _____

Tez Savunma Tarihi: 26/03/2024

BEYAN

Bu tez çalışmasının kendi çalışmam olduğunu, tezin planlanmasından yazımına kadar bütün aşamalarda etik dışı davranışımın olmadığını, bu tezdeki bütün bilgileri akademik ve etik kurallar içinde elde ettiğimi, bu tez çalışmasıyla elde edilmeyen bütün bilgi ve yorumlara kaynak gösterdiğimi ve bu kaynakları da kaynaklar listesine aldığımı, yine bu tezin çalışılması ve yazımı sırasında patent ve telif haklarını ihlal edici bir davranışımın olmadığını beyan ederim.

26 Mart 2024

Tunahan TİMUÇİN

TEŞEKKÜR

Doktora öğrenimimde ve bu tezin hazırlanmasında gösterdiği her türlü destek ve yardımdan dolayı çok değerli hocam Doç. Dr. Serdar BİROĞUL'a, en içten dileklerimle teşekkür ederim.

Bu çalışma boyunca yardımlarını ve desteklerini esirgemeyen sevgili eşim Ezgi KARA TİMUÇİN'e, aileme ve çalışma arkadaşlarıma sonsuz teşekkürlerimi sunarım.

26 Mart 2024 **Tunahan TİMUÇİN**

İÇİNDEKİLER

Sayfa No

ŞEKİL LİSTESİ x
ÇİZELGE LİSTESİ xi
KISALTMALAR xii
ÖZET xiii
ABSTRACT xiv
EXTENDED ABSTRACT xv
1. GİRİŞ 1
1.1. TEZ ÇALIŞMASININ AMACI VE KONUSU 3
1.2. TEZ ÇALIŞMASININ KATKILARI VE BENZERSİZ YÖNLERİ 7
2. BLOKZİNCİR VE AKILLI SÖZLEŞME KAVRAMLARI: LİTERATÜR TARAMASI 8
2.1. BLOKZİNCİR TEKNOLOJİSİ NASIL ÇALIŞIR? 8
2.1.1. Transaction (İşlem) Oluşturma 9
2.1.2. Transaction (İşlem) Onayı 10
2.1.3. Bloğun Zincire Eklenmesi 11
2.1.4. Blokzincir Teknolojisi Literatür Taraması 13
2.2. AKILLI SÖZLEŞMELER NASIL ÇALIŞIR? 17
2.2.1. Akıllı Sözleşme Platformlarının Karşılaştırılması 19
2.2.2. Akıllı Sözleşmeler Literatür Taraması 20
3. MATERYAL VE METOT 24
3.1. AKILLI SÖZLEŞME NEDİR? NEDEN KULLANILIR? 24
3.2. AKILLI SÖZLEŞME YAZIMINDA KULLANILAN ARAÇLAR 25
3.2.1. Metamask 25
3.2.2. NodeJs 28
3.2.3. Ganache 30
3.2.4. Ethereum Cüzdanları 33
3.2.5. Truffle 34
3.2.6. EthereumJs – TestRpc 36
3.2.7. Web3 Provider 37
3.2.8. Remix Ide 38

3.3. AKILLI SÖZLEŞMELER İÇİN ÜCRETSİZ TEST AĞLARI 39
3.3.1. Ethereum Test Ağları 39
3.3.1.1. *Ropsten Test Ağı* 39
3.3.1.2. *Rinkby Test Ağı* 39
3.3.1.3. *Kovan Test Ağı* 39
3.3.1.4. *Goerli Test Ağı* 40
3.3.1.5. *Casper Test Ağı* 40
3.3.2. Avalanche Test Ağları 41
3.3.2.1. *Fuji (P-Chain, X-Chain ve C-Chain) Test Ağları* 41
3.3.2.2. *Islander Test Ağı* 41
3.3.3. Cardano Test Ağları 41
3.3.3.1. *Alonzo Test Ağı* 41
3.3.4. Solana Test Ağları 42
3.3.4.1. *Devnet* 42
3.4. AKILLI SÖZLEŞMELERİN KODLANMASI İÇİN GELİŞTİRİLMİŞ PROGRAMLAMA DİLLERİ 42
3.4.1. Solidity 42
3.4.2. Vyper 43
3.4.3. Simplicity 43
3.4.4. Liquidity 43
3.4.5. Cadence 43
3.4.6. Move 44
3.4.7. Rholang 44
3.4.8. Marlowe 44
3.4.9. Scilla 44
3.4.10. Plutus 45
3.5. TRUFFLE, SOLIDITY VE JAVASCRIPT İLE BASİT AKILLI SÖZLEŞME OLUŞTURMA 45
3.6. SOLIDITY İLE YAZILMIŞ AKILLI SÖZLEŞME KODUNUN JSON KODUNA DÖNÜŞTÜRÜLMESİ 49
3.7. AKILLI SÖZLEŞME KODUNUN JAVASCRIPT İLE KULLANILMASI 52
3.8. 3 ADET TEMEL SEVİYE AKILLI SÖZLEŞME UYGULAMASI 54
3.8.1. Uygulama 1: Merkeziyetsiz Dosya Raporlama Sistemi 54
3.8.2. Uygulama 2: Yapılacaklar Listesi (ToDo) 56
3.8.3. Uygulama 3: Fund Raising (Fon (Bağış) Toplama) Uygulaması 57

3.9. AKILLI SÖZLEŞMELERİN GAS ÜCRETİ, ÇALIŞMA ZAMANI VE BLOKZİNCİR AĞINA DAĞITIMLARI BAKIMINDAN DEĞERLENDİRİLMESİ 59
3.9.1. En Popüler Akıllı Sözleşme Dağıtım Zinciri Ağları..... 59
3.9.1.1. *Bazı Popüler Akıllı Sözleşme Dağıtım Zinciri Ağları ve Önemleri* 60
3.9.2. Bir akıllı sözleşme dağıtım zinciri ağı seçerken göz önünde bulundurulması gereken faktörler nelerdir? 64
3.9.2.1. *Akıllı Sözleşme Dağıtım Zincir Ağlarının, Güvenlik Yönünden Kıyaslaması* 65
3.9.2.2. *Akıllı Sözleşme Dağıtım Zinciri Ağlarının, Ölçeklenebilirlik Açısından Karşılaştırması* 66
3.9.2.3. *Akıllı Sözleşme Dağıtım Zinciri Ağlarının Maliyet Açısından Karşılaştırması* 67
3.9.2.4. *Geliştirici Topluluğu Açısından Mevcut Tüm Akıllı Sözleşme Dağıtım Zinciri Ağlarının Karşılaştırması* 77
3.9.2.5. *Ekosistem Açısından Mevcut Tüm Akıllı Sözleşme Dağıtım Zinciri Ağlarının Karşılaştırması* 78
3.9.2.6. *Akıllı Sözleşme Dağıtım Zinciri Ağlarının, Solidity Veya Rust Programlama Dilini Destekleme Durumları* 80
4. İŞBİRLİKÇİ AKILLI SÖZLEŞMELER MODELİ..... 82
4.1. BASİT AKILLI SÖZLEŞME YAPISI..... 82
4.2. AKILLI SÖZLEŞMELERE ERİŞİM YÖNTEMLERİ 83
4.2.1. Yükseltilebilir(Upgradeable) akıllı sözleşmeler 83
4.2.2. İşlev imzasına göre harici işlev çağrısı..... 84
4.2.3. Interface(Arayüz) 85
4.3. İŞBİRLİKÇİ AKILLI SÖZLEŞMELER MODELİ(İASM - COLLABORATIVE SMART CONTRACT(COSC) MODEL) 86
4.3.1. CoSC Çerçevesine Genel Bakış 86
4.3.2. Akıllı Sözleşme Kayıt Prosedürü..... 87
4.3.3. Dağıtılan akıllı sözleşmelerle etkileşim..... 87
4.3.4. Modelin Avantajları 89
4.4. İŞBİRLİKÇİ AKILLI SÖZLEŞME MODELİNİN EV ALIM-SATIM ANLAŞMASINA UYGULANMASI 90
4.4.1. dApp Uygulama Sonuçları 92
4.5. İŞBİRLİKÇİ AKILLI SÖZLEŞME MODELİNİN ARAÇ ALIM-SATIM ANLAŞMASINA UYGULANMASI 95
4.5.1. Tek Bir Sözleşme Olarak Araç Alım-Satım Akıllı Sözleşmesi..... 96

4.5.2. İşbirlikçi Araç Alım-Satım Akıllı Sözleşmesi 97
4.5.3. Geleneksel Ve İşbirlikçi Akıllı Sözleşmenin Blokzincir Ağına Dağıtım Maliyetlerinin Karşılaştırılması 100
5. SONUÇLAR.....103
6. KAYNAKLAR.....107
ÖZGEÇMİŞ.....116

ŞEKİL LİSTESİ

Sayfa No

Şekil 1.1. Ev Alım-Satım Süreci 5
Şekil 2.1. Blokzincir Akış Şeması 8
Şekil 2.2. Kripto Para Gönderimi 10
Şekil 2.3. Merkle Ağacı Gösterimi 11
Şekil 2.4. SHA256 Eşitlik Gösterimi 12
Şekil 2.5. Blokzincir Ağı 13
Şekil 2.6. Akıllı Sözleşme Nasıl Çalışır? 18
Şekil 2.7. Alıcı ve tedarikçi arasındaki akıllı sözleşme örneği [**42**] 18
Şekil 3.1. Metamask Hesap Oluşturma 26
Şekil 3.2. Nodejs kurulumu 30
Şekil 3.3. Ganache: Kişisel Blokzincir 31
Şekil 3.4. Ethereum Cüzdanı 34
Şekil 3.5. Web3.js Akıllı Sözleşme Etkileşimi 38
Şekil 3.6. Örnek Basit bir Akıllı Sözleşme Kodu (NumberStore) 47
Şekil 3.7. Migrations.sol dosyası kod içeriği 47
Şekil 3.8. Konfigürasyon Dosyası. 48
Şekil 3.9. 1_initial_migration.js kodu 49
Şekil 3.10. Örnek Basit bir Akıllı Sözleşme Kodu (NumberStore) 50
Şekil 3.11. Solidity Kodunun Deploy edilmesi 50
Şekil 3.12. Solidity kod dosyalarından dönüştürülen JSON dosyaları 51
Şekil 3.13. Test için temin edilen adresler 52
Şekil 3.14. Test edilen proje için ödenmesi gereken eth miktarları 52
Şekil 3.15. Akıllı Sözleşmeye javascript ile veri girişi 53
Şekil 3.16. Blokzincire eklenen bilgilerin görüntülenmesi 54
Şekil 3.17. Merkeziyetsiz Dosya Raporlama Sistemi 55
Şekil 3.18. Remix IDE üzerinde Uygulama Arayüzü 55
Şekil 3.19. Yapılacaklar Listesi (ToDo) Akıllı Sözleşme Kodu 56
Şekil 3.20. Remix IDE uygulama arayüzü 57

Şekil 3.21. FundRaising Akıllı Sözleşme Kodu 58
Şekil 3.22. Remix IDE uygulama arayüzü 59
Şekil 3.23. Basit Kod Örneği 76
Şekil 4.1. Akıllı Sözleşme Kodu Mimarisi Örneği 83
Şekil 4.2. Abstract Contract (Interface) Yapısı 85
Şekil 4.3. İşbirlikçi Akıllı Sözleşmeler(Collaborative Smart Contracts) 86
Şekil 4.4. Blokzincir Ağına Dağıtılan Akıllı Sözleşmeyi Çağırma 87
Şekil 4.5. Akıllı Sözleşme Kayıt İşlemi 88
Şekil 4.6. Akıllı Sözleşme etkileşim prosedürü 88
Şekil 4.7. Merkeziyetsiz ev alım-satım uygulaması (dApp) 91
Şekil 4.8. MainContract.sol 92
Şekil 4.9. Municipality Sözleşmesi Arayüzü 93
Şekil 4.10. Bank Sözleşmesi Arayüzü 94
Şekil 4.11. Ana Sözleşme Arayüzü (Call the Contracts) 95
Şekil 4.12. Araç satım sözleşmesinde araç detayları fonksiyonu. 96
Şekil 4.13. Araç satım sözleşmesi uygulama arayüzü 97
Şekil 4.14. İşbirlikçi Sözleşme Dosyaları 98
Şekil 4.15. İşbirlikçi sözleşme etkileşim durumları. 99
Şekil 4.16. İşbirlikçi sözleşme veri girişi. 99

ÇİZELGE LİSTESİ

Sayfa No

Çizelge 2.1. Akıllı Sözleşmeler ve Blokzincir oluşturma platformlarının karşılaştırılması 19
Çizelge 3.1. Zincir Ağlarının Güvenlik Durumları 65
Çizelge 3.2. Zinciri Ağlarının Ölçeklenebilirlik Durumları 66
Çizelge 3.3. Zincir Ağlarının Maliyet Durumları 67
Çizelge 3.4. Örnek Sözleşmeye göre Zincir Ağlarının Maliyet Karşılaştırması 77
Çizelge 3.5. Zincir Ağlarının Maliyet Durumları 78
Çizelge 3.6. Zincir Ağlarının Ekosistem Durumları 79
Çizelge 3.7. Zincir Ağlarının Programlama Dillerine Destek Durumları 80
Çizelge 4.1. Geleneksel ve İşbirlikçi akıllı sözleşmelerin karşılaştırılması 90
Çizelge 4.2. Geleneksel ve İşbirlikçi Akıllı Sözleşmelerin maliyet karşılaştırması 100
Çizelge 4.3. Durum 1 101
Çizelge 4.4. Durum 2 102

KISALTMALAR

SC	Smart Contracts(Akıllı Sözleşmeler-AS)
COSC	Collaborative Smart Contracts(İşbirlikçi Akıllı Sözleşmeler-İAS)
EVM	Ethereum Virtual Machine(Ethereum Sanal Makinesi)
IOT	Internet of Things(Nesnelerin İnterneti)
AI	Artificial Intelligence(Yapay Zeka)

ÖZET

BLOKZİNCİR TEKNOLOJİSİNDE İŞBİRLİĞİNE DAYALI AKILLI SÖZLEŞME MODELİNİN GELİŞTİRİLMESİ VE UYGULANMASI

Tunahan TİMUÇİN
Düzce Üniversitesi
Lisansüstü Eğitim Enstitüsü, Bilgisayar Mühendisliği Anabilim Dalı
Doktora Tezi
Danışman: Doç. Dr. Serdar BİROĞUL
Mart 2024, 115 sayfa

Blokzincir teknolojisi, tüm şifreli işlemleri en kısa tanımlama ile izlemek ve güvence altına almak için geliştirilmiş dağıtık bir veri kayıt sistemidir. Blockchain teknolojisi 2008 yılında adından söz ettirmiştir. Bu teknolojinin ortaya çıkışı, çeşitli alanlarda yenilikçi çözümler için fırsatlar yaratmaktadır. Bu çözümlerden biri, sözleşme hüküm ve koşullarının yürütülmesini otomatikleştirebilen akıllı sözleşmeler (AS – Smart Contracts SC) kavramıdır. Kripto para dünyasında akıllı sözleşme, Blokzincir üzerinde çalışan bir uygulama veya program olarak tanımlanabilir. Belirli kurallara uyması gereken dijital anlaşmalar olarak çalışırlar. Bu kurallar bilgisayar kodları tarafından önceden belirlenir ve daha sonra tüm ağdaki sunucular tarafından kopyalanır ve uygulanırlar. Bu tezde, akıllı sözleşmelerin ve işbirlikçi karar vermenin faydalarını birleştiren yeni bir yaklaşım olan İşbirlikçi Akıllı Sözleşmeleri (İAS, Collaborative Smart Contrats CoSC) tanıtıyoruz. CoSC, daha şeffaf ve adil bir sistem sağlayarak, birden çok tarafın sözleşme yürütme sürecine katılmasına izin vermektedir. Bu tez çalışmasında uygulama örnekleri olarak Araç ve Gayrimenkul alım satım senaryoları işbirlikçi akıllı kontratlar ile gerçekleştirilmiştir. Önerilen yaklaşımımız, CoSC için güvenli ve merkezi olmayan bir ortam oluşturmak için bir konsorsiyum blokzinciri ağı kullanır. CoSC modeli, alıcı, satıcı ve işleme dâhil olan diğer ilgili taraflar arasındaki iletişimi, işbirliğini ve karar vermeyi kolaylaştırmaktadır. CoSC modelinin performansı yürütme süresi, faydalar ve güvenlik açısından değerlendirilmiştir. Tez çalışmasında, CoSC modelinin, blokzincir ağının daha verimli kullanımının sağlanması, kod karmaşasının önüne geçilmesi, modülerlik, yeni teknolojilere entegrasyonun kolaylaşması gibi bir çok fayda sağladığı görülmüştür. Ek olarak, tez çalışmasında, CoSC modeli değerlendirmesinde akıllı sözleşmelerin ağa dağıtılma süresi olarak ortalama %20+ başarı sağlanmıştır. Sonuçlar, CoSC'nin karmaşık, çok taraflı işlemler için umut verici bir çözüm olduğunu ve emlak ve araç alım-satım sektörü başta olmak üzere, tüm sektörlerde verimliliği, şeffaflığı ve güvenilirliği artırabileceğini göstermektedir.

Anahtar sözcükler: Blokzincir Teknolojisi, Akıllı Sözleşmeler, Gayrimenkul Satış Platformu, Araç Alım Satım Platformu.

ABSTRACT

DEVELOPMENT AND IMPLEMENTATION OF COLLABORATIVE SMART CONTRACT MODEL IN BLOCKCHAIN TECHNOLOGY

Tunahan TİMUÇİN
Düzce University
Graduate School, Department of Computer Engineering
Doctoral Thesis
Supervisor: Assoc. Prof. Serdar BİROĞUL
March 2024, 115 pages

Blockchain technology is a distributed data recording system developed to track and secure all encrypted transactions with the shortest identification. Blockchain technology made a name for itself in 2008. The emergence of this technology creates opportunities for innovative solutions in various fields. One of these solutions is the concept of smart contracts (SC), which can automate the execution of contract terms and conditions. In the cryptocurrency world, a smart contract can be defined as an application or program that runs on Blockchain. They operate as digital agreements that must follow certain rules. These rules are predetermined by computer codes and are then copied and enforced by servers throughout the network. In this thesis, we introduce Collaborative Smart Contracts (CoSC), a new approach that combines the benefits of smart contracts and collaborative decision-making. CoSC allows multiple parties to participate in the contract execution process, providing a more transparent and fair system. In this thesis study, Vehicle and Real Estate buying and selling scenarios were implemented with collaborative smart contracts as application examples. Our proposed approach uses a consortium blockchain network to create a secure and decentralized environment for CoSC. The CoSC model facilitates communication, collaboration and decision-making between the buyer, seller and other interested parties involved in the transaction. The performance of the CoSC model was evaluated in terms of execution time, benefits, and security. In the thesis study, it has been seen that the CoSC model provides many benefits such as ensuring more efficient use of the blockchain network, preventing code confusion, modularity, and facilitating integration with new technologies. In addition, in the evaluation of the CoSC model, an average of 20%+ success was achieved in terms of the time it took to deploy smart contracts to the network. The results show that CoSC is a promising solution for complex, multi-party transactions and can increase efficiency, transparency and reliability in all sectors, especially the real estate and vehicle buying and selling sectors.

Keywords: Blockchain Technology, Smart Contracts, Real Estate Sales Platform, Cars for Sale Platform.

EXTENDED ABSTRACT

DEVELOPMENT AND IMPLEMENTATION OF COLLABORATIVE SMART CONTRACT MODEL IN BLOCKCHAIN TECHNOLOGY

Tunahan TİMUÇİN
Düzce University
Graduate School, Department of Computer Engineering
Doctoral Thesis
Supervisor: Assoc. Prof. Serdar BİROĞUL
March 2024, 115 pages

1. INTRODUCTION

In this section, firstly the concepts of centralized web and distributed web are discussed. While centralized web represents a traditional internet structure where information is distributed from central servers located in large data centers, distributed web refers to a more secure internet model where information is shared between devices of many users rather than central servers.

Then, the working principle and importance of blockchain technology are explained. Blockchain functions as a distributed data recording system and securely records all transactions. The first popular use of blockchain technology began with the introduction of Bitcoin in 2008 by Satoshi Nakamoto.

How the blockchain technology works is discussed in detail. The steps for creating, approving and adding transactions to the blockchain in the blockchain are explained. While the transactions are added to the chain in blocks, the process of finding the proof of work number of the miners who perform this transaction is explained. Additionally, information is given about the verification process of transactions and the addition of blocks to the chain.

The interaction and working principle of smart contracts with the blockchain network were examined. Smart contracts work by placing coded conditions and assets on the blockchain, and they run automatically when certain triggers are activated. This can be used in many fields, especially finance, logistics, real estate and IoT.

Finally, a table comparing different smart contract platforms is presented. Platforms like Ethereum are prominent in designing and deploying smart contracts.

2. MATERIAL AND METHODS

Smart contracts are technologies that work automatically through computer programs and enforce contract terms. These contracts operate on a decentralized blockchain network and offer key features such as automation, trust, security and decentralization. This technology can be used in many industries such as finance, supply chain management, real estate, voting systems and more. However, one should be careful about security and the codes should be inspected meticulously, because security vulnerabilities can have serious consequences.

In order to develop a smart contract or to bring innovation to the smart contract, it is necessary to know smart contracts in all details.

Therefore, in this thesis, what you need to know about smart contracts is explained in detail. These; Tools Used in Smart Contract Writing, Free Test Networks for Smart Contracts, Developed Programming Languages for Coding Smart Contracts. Additionally, Simple Smart Contract Created with Truffle, Solidity and Javascript, Conversion of this Smart Contract Code written in Solidity to JSON Code is shown. Finally, the coding of 3 Smart Contract Applications and their results are shown.

After introducing the smart contract in full detail, the stages of creation of the Collaborative Smart Contracts Model, which is the subject of the thesis, and the Interface method used are introduced and applied to the real world problem of a house buying-selling contract.

3. RESULTS AND DISCUSSIONS

The Collaborative Smart Contracts model proposed in the thesis has been applied to real-world house purchase-sale contracts.

As a result of the advantages of this model; By accessing the contract placed on the blockchain network, its functions were used, code confusion was prevented thanks to the model, the cost was reduced by uploading the contracts in parts (the fee cost increases as the line of code increases when a contract is uploaded to the network), Modularity was provided, manageability was facilitated as each contract was responsible for itself and thus updated. Integration with technological developments has become easier.

The development of the model continues. Artificial intelligence will play a major role in new developments.

4. CONCLUSION AND OUTLOOK

Smart Contracts are an innovative technology that tries to establish its existence in the scientific world. That's why the number of developers and smart contract platforms is increasing day by day. In this thesis, a solution has been found to a problem that is lacking in the literature in terms of smart contracts. In the current situation, instead of reusing contracts distributed on the blockchain network, each contract coder can distribute its own code with the same content to the network. The collaborative model, which was developed to prevent this situation and to use the network more efficiently in all aspects, is expected to solve this problem.

In the future, when the stable integration of artificial intelligence with smart contracts is ensured, developments in the field of smart contracts are expected to accelerate further.

1. GİRİŞ

Arama motorlarında (ör. Chrome, Firefox veya Opera) yer alan bilgilerin çoğu, büyük bir veri merkezindeki soğuk bir odada bulunan bir sunucudan gelmektedir. Sıklıkla istemciler denilen birçok küçük bilgisayar ve telefon, ihtiyaç duydukları bilgi parçalarını indirmek için büyük merkezi sunucuya bağlanmaktadırlar. Modern web bu şekilde çalışmaktadır. İsminden de anlaşıldığı gibi, tüm veriler merkezi bir yerde bulunduğundan dolayı, bu sisteme Merkezi Web denmektedir.

Bununla birlikte, bazı gruplar, verileri depolayan, yöneten ve hatta sahip olan büyük bir varlığa sahip olmanın çok rahat olmadığını ve genellikle güvenli olmadığını düşünmeye başlamışlardır. Bunun yerine, herkesin bir miktar veriye sahip olabileceği ve ağdaki diğer herkesle paylaşabileceği bir ağ kurmaya karar verilmiştir. Bu nedenle, ihtiyaç duydukları bilgi parçalarını indirmek için tek bir sunucuya bağlanan bilgisayarlar ve telefonlar yerine, ihtiyaç duydukları daha küçük bilgi parçalarına sahip olan birçok kişiye bağlanmanın daha güvenli olduğu fikri değer kazanmıştır. Ve bilgiyi aldıktan sonra, aynı bilgiye ihtiyaç duyan başkalarıyla da paylaşma sistemi ortaya çıkmıştır. Bu sisteme Dağıtılmış Web denmektedir.

Dağıtılmış web'in birçok ünlü uygulaması vardır. En çok bilinen iki örnek Napster ve Torrent'tir. Napster, şarkıları kullanıcıları arasında paylaşmak için özel olarak oluşturulmuş dağıtılmış bir web'dir. Şarkıları depolayan merkezi bir sunucusu yoktur ve kullanıcılar istedikleri ve sahip oldukları şarkıları indirip yüklemektedirler. Benzer şekilde, Torrent, kullanıcıları arasında her tür dosyayı paylaşmak için özel olarak oluşturulmuş dağıtılmış bir web'dir.

Blokzincir, esasen Napster veya Torrent'e benzeyen, farklı bir amaca hizmet etmek için oluşturulmuş dağıtılmış bir webdir. Blokzincir, bir madeni para veya bir para birimi değildir. Blokzincir teknolojisi en kısa tanımlama ile şifreli tüm işlemlerin takibi ve güvenliğini sağlamak üzere geliştirilmiş dağıtık yapıda olan bir veri kayıt sistemidir. Eşler arası yönteme dayanan Blokzincir ağı, herhangi bir merkezi otorite ya da üçüncü taraf aracılara gereksinim duymayan ve tüm işlemlerin zaman damgalı olduğu dağıtık sistemli

bir veri tabanıdır. Tüm bu özellikleri sayesinde, blokzincir teknolojisi, verileri bir zincir gibi birbirine bağlamaktadır. Çeşitli şifre algoritmaları ile birbirine bağlanan veriler, birçok sunucuda dağıtık olarak paylaşılmaktadır. Blokzincir teknolojisi 2008 yılında Satoshi Nakomoto'nun bir dijital para birimi olan Bitcoin kripto parasını anlattığı makalesiyle adını duyurmuştur.

Blokzincir, internet gibi yıkıcı bir devrimdir. Bu devrim internette yeni bir para birimi ekonomisi ile başlamıştır. Merkezi bir otorite tarafından değil, ağ kullanıcıları arasındaki otomatik mutabakatla verilen ve desteklenen Bitcoin aracılığıyla gerçekleştirilmiştir. Blokzincir teknolojisi yeni bir yapı veya model değildir. Ancak Bitcoin adında bir kripto para birimi ortaya çıkana kadar pek çok kişi tarafından bilinmemiştir. 2008 mali krizini tetikleyen Lehman Brothers'ın iflası aynı yılın Eylül ayında gerçekleşmiştir [**1**]. Bu olaydan birkaç ay sonra Blokzincir, 2008-2009 yılları arasında Bitcoin ile birlikte Satoshi Nakamoto tarafından icat edilmiştir. Satoshi Nakamoto'nun bir kişi mi yoksa bir grup mu olduğu hala bilinmemektedir. Bu teknolojinin düzgün çalıştığından emin olduktan sonra insanlara bağışta bulunduğunu ve diğer teknolojiler üzerinde çalışmaya odaklanacağını belirttiği bir e-postanın ardından ortadan kaybolmuştur [**2**]. 2009 yılından itibaren bitcoin ve bitcoin varyasyonu adı verilen altcoinler üretilmiştir. Alt para birimleri, aynı genel yaklaşımla ancak farklı optimizasyonlar ve ince ayarlarla değişiklik yapan alternatif para birimleridir. Ödemeler, merkezi olmayan takas, madeni para kazançları ve harcamaları, dijital varlık transferi, akıllı sözleşme düzenleme ve yürütme için teknolojik bir altyapı görevi gören Blokzincir teknolojisi, Web'in hiç sahip olmadığı kesintisiz yerleşik ekonomik katmanı olma özelliğini kazanmasını sağlamaktadır. Daha da ileri giderek, bir merkeziyetsiz modu olarak Blokzincir teknolojisi, tüm insan faaliyetlerini geniş çapta yeniden yapılandırma potansiyeline sahip bir sonraki büyük yıkıcı teknoloji ve birinci sınıf bilgi işlem paradigması olma yeteneğine sahiptir. Blockchain teknolojisi bir ağaç gibi dallanan ve büyüyen, yeni bir sektöre öncülük eden, her dalda yeni olan bir teknolojidir.

Blokzincir teknolojisinin asıl gelişimi ise akıllı kontratların Blokzincir teknolojisinde kullanılması ile gerçekleşmiştir. Kripto para dünyasında bir akıllı kontrat, blokzinciri kullanan bir uygulama veya programdır. Genel olarak, belirli kurallara uymak zorunda olan dijital anlaşmalar olarak çalışır. Bilgisayar kodları önceden belirlenir ve tüm ağdaki sunucular tarafından kopyalanarak bu kuralları uygulamaya koyulur. Akıllı Kontratlar,

Nick Szabo tarafından ilk olarak 1990'larda tanıtılmıştır [**3**]. Nick Szabo, o zamanlarda akıllı sözleşmeleri, bilgisayar ağlarını güvenli hale getiren, protokoller ve ara yüzleri entegre ederek bilgisayar ağlarını resmileştiren bir teknoloji olarak tanımlamıştır. Akıllı kontratlar, blokzincir ağıyla birleştikten sonra, güven sorununu ortadan kaldıran protokoller yaratılmasına olanak sunmaktadır. Bu durum da insan hatalarını en aza indirgemiş ve insanların birbirine güven ihtiyacı duymadan taahhütler verebileceği bir platform ortaya çıkmıştır. Ethereum'un yaratıcısı ve kurucu ortağı Vitalik Buterin, Bitcoin protokolünün uzun yıllardır akıllı kontratları desteklemesine rağmen bunları popüler hale getirdi. Akıllı kontratlarda tabir olarak "akıllı" kelimesinin kullanılması el ile yapılmasının önüne geçilerek dijital olarak gerçekleşmesinden gelmektedir.

1.1. TEZ ÇALIŞMASININ AMACI VE KONUSU

Bu tez çalışmasının iki öncelikli amacı bulunmaktadır. Bunlardan ilki, akıllı sözleşmelerin eksik olan bir yönü tespit edilmiş ve bu eksiklik giderilmiştir. İkinci olarak ise geliştirilen bu yöntem gerçek dünya problemleri üzerinde uygulanmıştır.

Öncelikle bu tez çalışmasında, blokzincir akıllı sözleşmelerinin eksik bir yönüne odaklanılmıştır. Bu eksiklik, blokzincir ağında mevcut olan bir sözleşme kodunun, aynı içerikle veya aynı işlevleri yerine getirecek şekilde yeniden kodlanıp ağda dağıtılmasıdır. Bu sebepten ötürü blokzincir ağı verimsiz kullanılmakta, kod karmaşası ve maliyetlerin artması ile sonuçlanmaktadır. Bu sorunun veya eksikliğin çözümü için bu tez çalışmasında, blokzincir ağındaki mevcut dağıtılmış akıllı sözleşmelerin birbiri ile iletişimini ve etkileşimini sağlayan İşbirlikçi Akıllı Sözleşmeler Modeli (İASM, Collaborative Smart Contracts Model –CoSC-M) literatüre kazandırılmıştır. Bu durum tezin teknik olarak amacını oluşturmuştur.

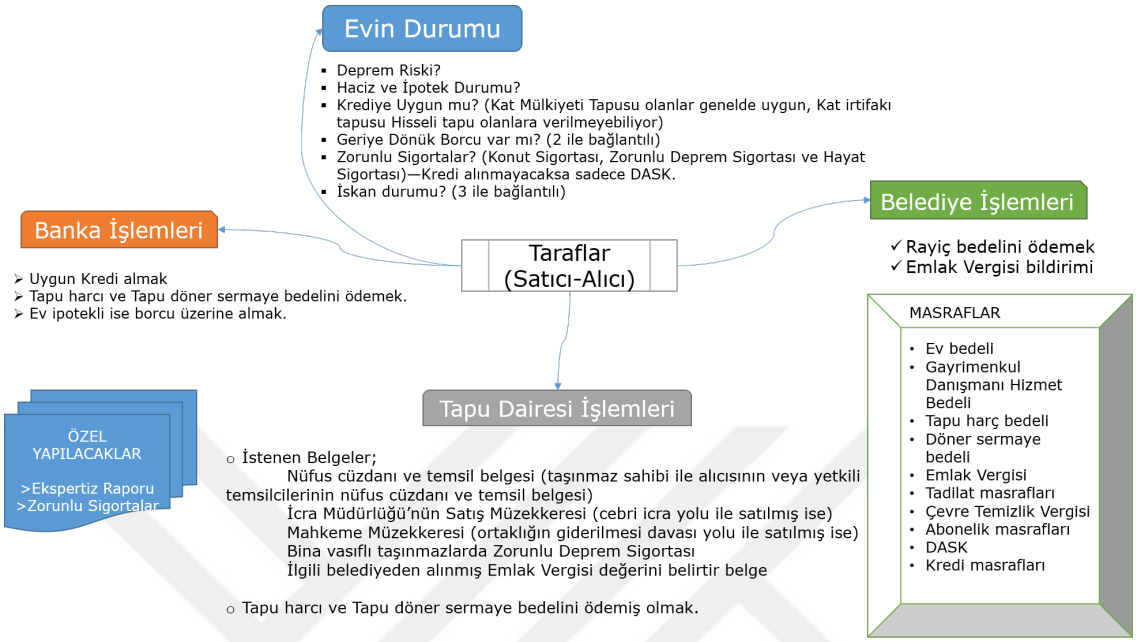
Geliştirilen bu modelin test edilebilmesi için ise gerçek bir dünya problemi olan ev alım-satım sözleşmesi oluşturulmuştur. Sözleşmenin oluşturulabilmesi için öncelikle, ev alım-satım sözleşmesinin nasıl ve hangi koşullarda oluşturulduğu araştırılmış ve aşağıda verilmiştir. Bu araştırma sonucu ise tezin uygulama olarak konusunu oluşturmuştur.

"Tapu devri" terimi, bir taşınmazın veya arazinin bir başkasına devredilmesini ifade eder. Gayrimenkullerin tapu siciline tescil edilmesi, herkes tarafından bilinen bir süreçtir. Miras,

mahkeme kararı, işgal, kamulaştırma ve kanunda belirtilen diğer durumlarda, mülkiyet tescilden önce elde edilir. Bununla birlikte, malikin bu durumlarda tasarruf edebilmesi, mülkiyetin tapu kütüğüne tescil edilmiş olmasına bağlıdır.

Gayrimenkul alım-satım işlemi yapılırken yapılan işlemlerin tam olarak bilinmesi gerekmektedir. Bu süreçlerde evrak fazlalığı, üçüncü şahısların işin içine karışması, güvenirlik sorunları, dolandırıcılığın önüne geçilememesi gibi sorunlarla karşılaşılmaktadır. Bu sebeplerden dolayı gayrimenkul alım satım işlemlerinin taraflarca üzerinde anlaşılan şartlar kapsamında dijital ortamda yapılması gerekmektedir. Bu sistemin tarafların güvenlikten şüphe duymayacağı bir platforma taşınması bekleniyor. Bu sistem için en uygun yöntemin blokzincir üzerine kurulu, üçüncü şahısları ortadan kaldıran ve merkezi bir sunucuya ihtiyaç duymayan akıllı sözleşmeler ile oluşturulması olduğu görülmektedir. Birçok alanda yeri olan akıllı sözleşmeler, gayrimenkul alım satım işlemlerinde de araştırmacıların ilgisini çekmiştir. Şekil 8, bir satın alma ve satış sözleşmesini açıklamaktadır. Gayrimenkul alım satımı ile ilgili literatürde çalışmalar mevcuttur. Sahai ve Pandey, Hindistan'da dolandırıcılığı önlemek için tapu kayıt sistemini akıllı sözleşmelere entegre etmek için çalışmışlardır [**4**]. Panda ve ark. dolandırıcılık ve kara para aklama olaylarını önlemek için akıllı sözleşmeler kullanarak next.js ve react tool'ları ile gayrimenkulü web'e aktarmışlardır [**5**]. Soner ve ark., ağdaki güvensizlik sorununu ortadan kaldırmak ve e-devlet sistemine katkı sağlamak için tapu sisteminin otomatize edilmesi gerekliliğini ortaya koymuştur [**6**]. Önerdikleri modelde, Shinde ve ark. bir arazi satın alındığında, arazi satın alımının basılı bir kopyasının hükümet yetkilisi tarafından alıcıya verilmesini sağlayan akıllı sözleşmeye dayalı bir sistem geliştirmişlerdir [**7**]. Tapu işlemlerinin akıllı sözleşmeler ile yapılması gerekliliği konusunda da farklı çalışmalar mevcuttur [**8**, **9**, **10**]. Tüm bu çalışmalar, akıllı sözleşmelerin geliştirilmesi ve literatürde yaygınlaşması açısından oldukça faydalıdır. Ancak akıllı sözleşmelerin farklı modellerle geliştirilmesi gerekmektedir. Literatürdeki mevcut çalışmalar tamamen sözleşme bazında yürütülmektedir. Bu tezde, önerilen model, blockchain ağında kodlanan ve dağıtılan akıllı sözleşmelerin etkileşimini sağlamak için geliştirilmiştir. Bu tezde, bu yapı, işlevselliğine uygun olarak işbirlikçi akıllı sözleşme yapısı olarak adlandırılmıştır.

Tescille mülkiyetin kazanılması geçerli bir sözleşme ve resmi şekillerin tamamlanması ile gerçekleşir. Bu sözleşme ve resmi şekillerin tamamlanması için birçok kurumdan işlem yapılmasını gerektirir. Şekil 1.1'de ev alım satım sürecinin genel bir şeması verilmiştir.



Şekil 1.1. Ev Alım-Satım Süreci

Şekil 1.1'deki süreçlere bakılarak, bir ev alım-satımı yapılırken;

- Kredi kullanmak, evin ipotek borcunu ödemek gibi işlemler için bir Banka Akıllı Sözleşmesi
- Belediyeden rayiç bedeli ve Emlak vergisi işlemleri için bir Belediye Akıllı Sözleşmesi
- Ev devir işlemlerinin yapılması için bir Tapu Dairesi Akıllı Sözleşmesi
- İki taraf arasındaki sorgulama ve anlaşma işlemleri için de bir Taraflar(Main) Akıllı Sözleşmesi gereklidir.

Tezde İşbirlikçi akıllı sözleşmelerin uygulandığı diğer bir alan ise Araç alım-satım sözleşmesidir.

Günümüzde otomobil sahibi olmak, günlük hayatı kolaylaştıran unsurların başında gelmektedir. Öte yandan araç sahibi olmak bir çeşit birikim yöntemi olarak da görülmektedir. Her iki durumda da satın alınacak aracın ihtiyaçlara cevap vermesi, güvenli olması ve bütçeye uygunluğu önemli konulardır. Bu nedenle araç alırken bilinmesi önemli

olan kimi konuların, satın alma işlemi yapılmadan önce gözden geçirilmesi kritik bir öneme sahiptir.

İster 2. el ister sıfır araç satın almak isteyen herkesin önceden bilmesi önemi olan bazı konular vardır. Yatırım amacıyla veya günlük kullanım için araç satın almaya karar verildiğinde mutlaka gerekli araştırmaların ve kontrollerin önceden yapılması gerekmektedir. Araç satın alırken bazı unsurları gözden geçirmek ileride karşılaşılma ihtimali olan sorunların önüne geçmektedir.

Araç almak isteyen tüketiciler, daha ucuza alabildikleri için 2.el araçlara yönelmektedirler. Alım satım sürecinde gerekli evrakların kotrolünün yapılması ve düzenlenmesi gerekmektedir. Bu yüzden araç alacak kişilerin araca ait trafik kaydını, aracın resmi evraklarını kontrol etmesi, araç alım satımını mutlaka noter eşliğinde yapması gerekmektedir. 2.el araç satışı öncesinde araçla ilgili; trafik cezası, son yapılan muayene bilgisi, motorlu taşıtlar vergisi(mtv)'nin ödenmiş olması kontrol edilmelidir. Çünkü bu bilgilerde sıkıntı olması araç satışına engel durumlardır.

Taraflar gerekli işlemleri noter huzurunda tamamlar ve aracın devir işlemi gerçekleştirilir. Noter huzurunda gerçekleştirilen işlemler şu şekildedir:

* Öncelikle satıcı, araca ait tüm belgeleri notere teslim eder.
* Araçla ilgili satışa engel durumlar kontrol edilir.
* Noter hem alıcı hem satıcıdan, T.C.kimlik numaralarının yazılı olduğu kimlik belgesi talep eder.
* Yetkilinin çıkardığı noter satış sözleşmesine, hem alıcı hem satıcı, ad, soyad, telefon numarası yazarak imzalar.
* İmzalar atıldıktan sonra, alıcı vezneye hem noter bedelini hem araç bedelini öder. Alıcı, işlemler tamamlanınca aracın geçici ruhsatını almış olur.

2.el araç satışı aynı zamanda devir işlemine bir örnektir. Önce araç satışı yapılır daha sonra devir işlemi yapılır. Devir işlemi yapılırken; ilişik kesme belgesi (vergi dairesinden alınan), zorunlu trafik sigortası, noter satış belgesi, plaka, tescil ve trafik belgesi ve vergi numarası bilgileri ibraz edilmelidir.

İşbirlikçi akıllı sözleşmenin araç alım-satımına uygulanması için tüm şart ve koşullar belirlenmiştir. 'Materyal ve Metot' ve 'Sonuç' bölümlerinde, sözleşmenin kodlama aşamaları, teknik detaylar ve sonuçlarına yer verilmiştir.

1.2. TEZ ÇALIŞMASININ KATKILARI VE BENZERSİZ YÖNLERİ

Bu tez çalışması, literatürde, akıllı sözleşmeler alanında önemli bir eksikliği çözüme ulaştırarak katkılar sunmaktadır. Mevcut literatür tarandığında, blokzincir akıllı sözleşmelerinin eksik bir yönü tespit edilmiştir. Bu eksiklik, blokzincir ağında mevcut olan bir sözleşme kodunun, aynı içerikle veya aynı işlevleri yerine getirecek şekilde yeniden kodlanıp ağda dağıtılmasıdır.

Bu sebepten ötürü blokzincir ağı verimsiz kullanılmakta, kod karmaşası ve maliyetlerin artması ile sonuçlanmaktadır. Bu sorunun veya eksikliğin çözümü için bu tez çalışmasında, blokzincir ağındaki mevcut dağıtılmış akıllı sözleşmelerin birbiri ile iletişimini ve etkileşimini sağlayan İşbirlikçi Akıllı Sözleşmeler Modeli (İASM, Collaborative Smart Contracts Model –CoSC-M) literatüre kazandırılmıştır.

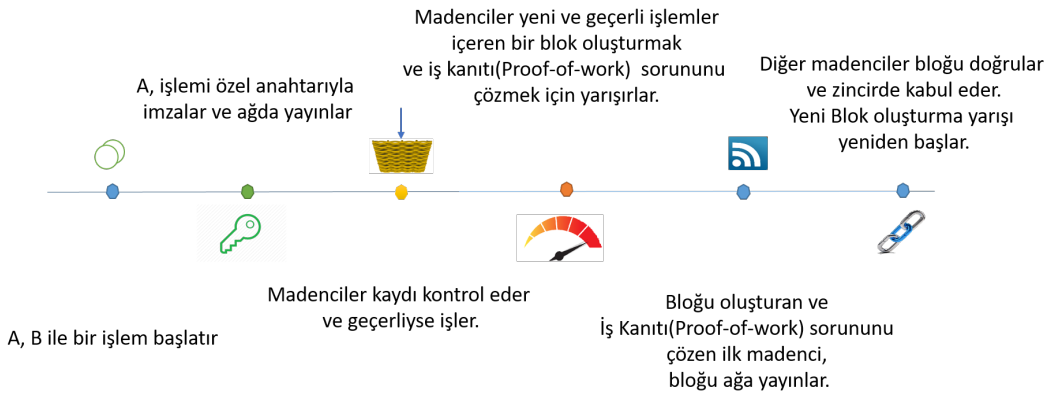
Literatüre kazandırılan İşbirlikçi Akıllı Sözleşmeler Modeli'nin, bu alandaki benzersiz katkıları şu şekildedir:

- ✓ Blokzincir ağında dağıtılmış olan bir sözleşmeye erişme ve işlevlerini kullanma
- ✓ Kod karmaşıklığının önlenmesi
- ✓ Modülerlik sağlamak
- ✓ Yönetilebilirliğin kolaylaştırılması
- ✓ Güncel teknolojik gelişmelerle entegrasyonu kolaylaştırmak
- ✓ Bazı durumlarda maliyetin düşürülmesi

2. BLOKZİNCİR VE AKILLI SÖZLEŞME KAVRAMLARI: LİTERATÜR TARAMASI

2.1. BLOKZİNCİR TEKNOLOJİSİ NASIL ÇALIŞIR?

Blokzincir'in çalışma prensibi ilk başta oldukça karmaşık görünebilir ancak teknolojik bir işlemler bütünü yerine bir veri defteri, bir kayıt defteri olarak düşünülürse daha makul bir yaklaşım elde edilmektedir. Şekil 2.1, bir Blokzincir ağına blok ekleme adımlarını göstermektedir. Blokzincir'in çalışma prensibi şu şekildedir; Bir Blokzincir ağına bir işlem ulaşır. Bu işlem ilk kez gerçekleştiriliyor olabilir veya mevcut bir işlemi düzenlemek istenebilir. Bu işlem geldiğinde, Blokzincir'in tüm bloklarının doğrulanması gerekmektedir. Düğümlerin çoğu, doğrulama için algoritmalar çalıştırır. Blokzincir'deki düğümlerin yarısından fazlası işlemdeki imza ve tarihin geçerliliğini kabul ederse asıl işlem kabul edilir ve zincir bloğa eklenir. Zincir benzeri yapısından dolayı Blokzincir adı verilmektedir. Belirli bir zamanda kaydedilen, her biri kendi imzasına sahip olan veri blokları, sırayla dizilir ve bir Blokzincir oluşturur. Bu yapıdaki ilk kayıt başlangıç bloğu olduğundan, bu bloğa özellikle Genesis adı verilmektedir. Düğümlerin yarısından fazlası işlemin zincir bloğa eklenmesine veya değişiklik yapılmasına katılmaz ise düğüm reddedilir ve zincir bloğa eklenmez. Blokzincir, bu dağıtılmış model sayesinde dağıtılmış bir defter olarak çalışır. Bu model sayesinde üçüncü şahıs ihtiyacı ortadan kalkmaktadır. Blokzincir çalışma şeması üç ana başlık altında açıklanabilmektedir.



Şekil 2.1. Blokzincir Akış Şeması

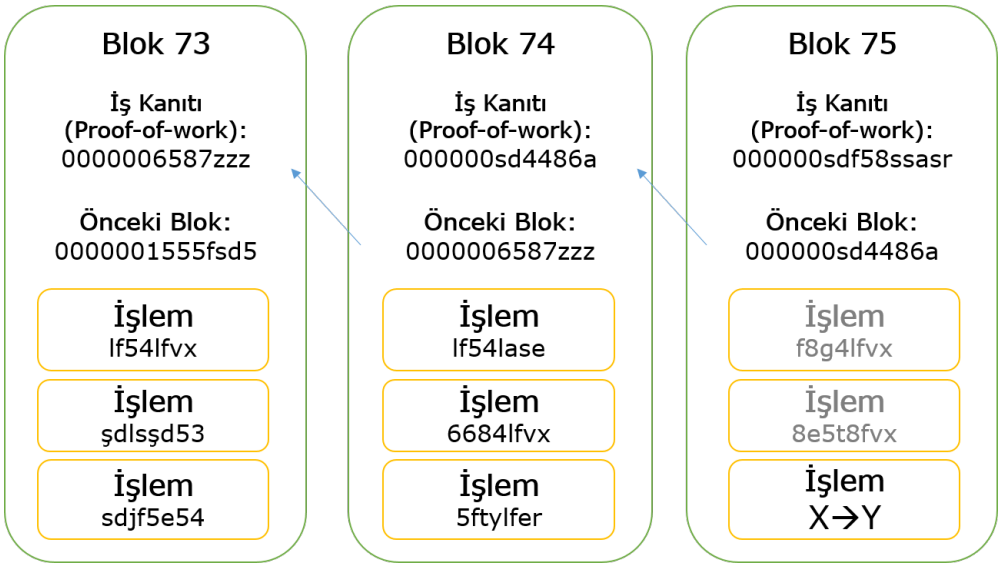
2.1.1. Transaction (İşlem) Oluşturma

Transaction (İşlem), daha küçük parçalara ayrılamayan en küçük işlem yığını olarak adlandırılsa da, bir grup işlemin art arda gerçekleşmesine ve işlemlerin toplu olarak değerlendirilmesine olanak tanımaktadır. "Transaction" prensip olarak ya hep ya hiç temelinde çalışmaktadır. Dolayısıyla, işlemlerden biri başarısız olduğunda, hiçbir işlem gerçekleşmiş sayılmaz. Ancak tüm işlemler başarılı olduğunda "Transaction", içinde meydana gelen tüm veri değişikliklerinin onaylandığı anlamına gelir. Dijital para birimine sahip olmak ve gönderme, alma gibi işlemleri sağlamak için öncelikle bir cüzdan uygulamasına ihtiyaç duyulmaktadır. Bu cüzdan uygulaması indirildiğinde, kişiye açık ve gizli olmak üzere iki adres verilmektedir. Genel adres kullanıcı adı düşünülebilir. Bir kişinin Dijital para göndermek için bu genel adrese ihtiyacı vardır. Gizli adres sadece hesap sahibinin bilmesi gereken bir şifre gibi düşünülebilir. Bir kişi dijital para göndermek istediğinde ise bu adresi kullanmaktadır.

Örneğin; X kişisinin, Y kişisine 2 Bitcoin göndermek istediği varsayılsın. Bunun için cüzdan uygulamasına X'in genel adresini ve gönderilecek Bitcoin miktarını girmesi yeterli olmaktadır. Öncelikle, bu cüzdan X için bir işlem oluşturmaktadır.

Bireylerin Bitcoin bakiyeleri Bitcoin blokzincirinde tutulmaz. Kişilerin sahip olduğu Bitcoin sayısı, daha önce adreslerine gönderilen Bitcoinlerin toplamına göre belirlenmektedir. Bu durumda cüzdan uygulaması, daha önce zincir üzerinde Y'ye gönderilen Bitcoin'leri bulmak için bir arama yapar. Bu araştırma Bitcoin X'in göndermek istediği miktara ulaşana kadar devam etmektedir.

Şekil 2.2'de gösterilen örnekte olduğu gibi, cüzdan uygulaması daha önce X'in açık anahtarına gönderilen işlemleri 2 Bitcoin'e ulaşana kadar bulmaktadır. Ancak işlemler toplandığında 3.0 Bitcoin yaptığından dolayı, 2 Bitcoin'i Y'ye, kalan 1.0 Bitcoin'i X'in kendi genel adresine göndermek için bir işlem oluşturmaktadır. Bu işlemi oluşturmak için son aşama, imzalama aşamasıdır. Bu aşamada, bu işlemin X dışında olduğunun kanıtlanması gerekmektedir. Bunun için dijital imza adı verilen kriptografik bir yöntem kullanılır. Cüzdan uygulaması, X'in gizli anahtarını ve bu işlemi belirli bir yöntem üzerinden geçirerek bir imza oluşturur. Bu imza, gelecekte Blokzincir



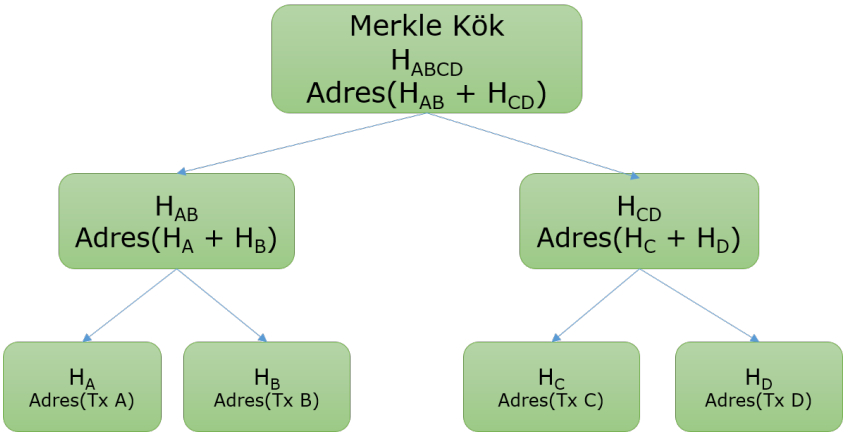
Şekil 2.2. Kripto Para Gönderimi

ağındaki kişiler tarafından bu işlemin aslında X tarafından oluşturulduğunu kanıtlamak için kullanılmaktadır. Son olarak, oluşturulan bu işlemi onaylamak için cüzdan uygulaması Blokzincir ağına gönderilir.

2.1.2. Transaction (İşlem) Onayı

Blokzincir ağı üzerindeki transaction (işlem), blokzincir ağına yazmadan önce kripto para madencileri tarafından doğruluğu kontrol edilmektedir. Bunun için 2 kritere bakılmaktadır. İlk olarak transaction'ın içinde referans edilen Bitcoin'lerin daha önce kullanılıp kullanılmadığı kontrol edilir. Bu kısım merkle kök ağacı yöntemi ile hızlandırılmıştır. Merkle ağaç ikili bir yapıdır ve Merkle kök onun kök düğümüdür. Verinin küçük parçaları, bu ağaç yapısı tarafından sıralı olarak adreslenir. Her bir bileşen, veri adresleme fonksiyonunu kullanarak kendi adreslerini oluşturur. Merkle ağaç yaprakları, ağaç veri yapısında adresleme (hash) fonksiyonunu kullanarak her iki bileşenin adreslerini birleştirerek kendi adreslerini elde ederler. Tüm ağaçtaki yapraklar döngü kök düğüme erişinceye kadar devam eder. Şekil 2.3'te merkle ağacı görünümü verilmiştir.

İkinci olarak da, transaction'ın içindeki imzanın doğru olup olmadığı kontrol edilir. Bu, bir fonksiyon içerisinde gönderenin imzasını, transaction'ı ve açık adresini koyarak bulunmaktadır. Bu denklem genel olarak Denklem (2.1)'deki gibi kullanılmaktadır.



Şekil 2.3. Merkle Ağacı Gösterimi

$$1 = ?v(message, public\_key, signature) \tag{2.1}$$

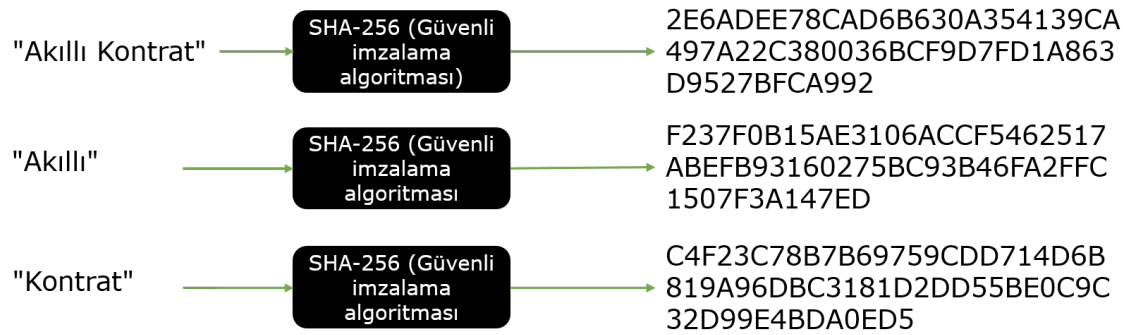
Bu imzadan dönen sonucun "true" olması durumunda, gerçekleşen transaction, onay verilenler havuzuna dâhil edilir. Son adım olarak ise bu transaction zincire dâhil edilir.

2.1.3. Bloğun Zincire Eklenmesi

Blok, bir metin dosyası olarak düşünülebilir. Bu metin dosyasının içinde blok numarası, İş Kanıtı (proof of work) numarası, bir önceki bloğun İş Kanıtı (proof of work) numarası ve son olarak onaylanan transaction'ların tutulduğu bir kayıt defteri gibidir.

Tek bir Blokzincirin, ağ üzerindeki tüm düğümlerde aynı özelliklere sahip olacak şekilde tutulması nedeniyle, ağa bir blok eklenmesi gerektiğinde, bu bloğun tüm düğümlere yayılması ve doğrulanması gerekmektedir. Ve bu işlem oldukça zordur. Aksi durumda, blok ekleme işi kolay olmuş olsa, herkes tarafından zincire aynı anda blok eklenmesi ve bunun sonucu olarak zincirde ağı bozacak dallanmalar gerçekleşmesi mümkün hale gelmektedir. Bu tip sorunların engellenmesi için, Blokzincirin kurucusu olan Satoshi Nakamoto tarafından 'Proof of Work' ismiyle bilinen yeni bir model geliştirilmiştir. Bu yöntem bir çeşit bilmeceye benzetilmektedir. Zincire blok eklenmesi için blokzincir madencileri tarafından bu bilmecenin çözülmesi beklenmektedir. Bu bilmeceyi ilk çözen blokzincir madencisi hem bloğu zincire eklemekte hem de bunun karşılığında bir miktar kripto para ödülü kazanmaktadır. İş Kanıtı (Proof-of-work) mantığından önce SHA256 algoritmasının

çalışma prensibinden bahsetmek gereklidir. SHA256, belirli bir algoritmadan geçen bir dosyanın 256 bit uzunlukta bir özetini oluşturur ve geri dönüşü mümkün değildir. Örneğin bir roman veya tek bir cümle olsa bile, bu dosyanın 256 haneli tahmin edilemez bir hash özeti her zaman ortaya çıkacaktır. Şekil 2.4 örnek sonuçları göstermektedir.

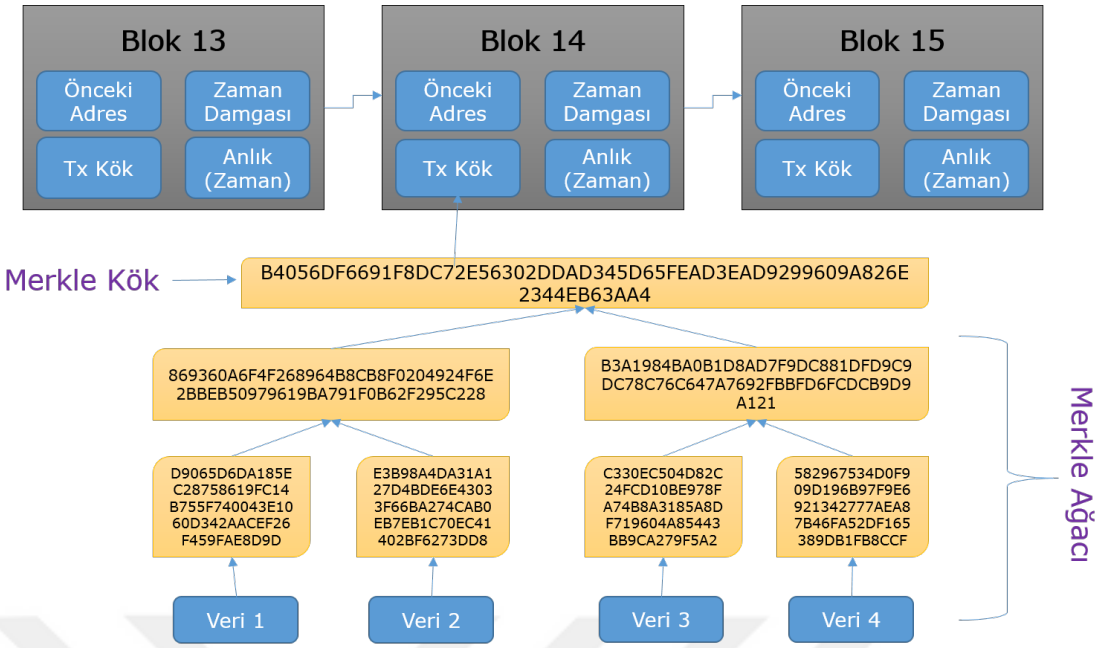


Şekil 2.4. SHA256 Eşitlik Gösterimi

İş Kanıtı (Proof-of-work) yönteminde ise blok içerisinde, içinde İş Kanıtı (Proof-of-work) anlık değeri bulunmaktadır ve bu kısım sürekli SHA-256 algoritmasından geçirilmektedir. Bu işlem sistem tarafından istenilen, örneğin 'ilk 20 hanesi sıfır olan bir hash bul' koşulunu sağlayana kadar devam etmektedir. Bu sıfır olan hane sayısı daha az veya fazla da olabilmektedir. Bu İş Kanıtı (Proof-of-work) numarasını ilk bulan kişi cevabını ve eklemek istediği bloğu ağa yaymaktadır. SHA256 algoritması ile birlikte en yaygın kullanılan algoritmalar; SHA-1, SHA-2 ve MD5 algoritmalarıdır Tüm bu Blokzincir tanımları ve anlatımı göz önüne alınarak Şekil 2.5'te belirtildiği gibi genel bir Blokzincir yapısı belirlenmiştir.

Şekil 2.5'te, blok 13 önceki bloğun Hash'ini temsil eder ve oluşturma zaman damgasından oluşur, işlem kökü Merkle kökü olarak da bilinir ve anlık(nonce) doğrulanması gereken algoritma sayacıdır. İşlemlerle ilgili tüm gizli bilgiler tekrar karıştırılır ve bu hash 14.blokta yakalanır. Önceki bloğun Hash'i, zaman damgası, Merkle kökü ve nonce oluşur.

Şekil 2.5'te yer alan veri 1, veri 2, veri 3 ve veri 4 bölümlerinde tüm veriler eşleştirilmiş ve birden çok Hash bulunur. Böylece, veriler son bir Hash'e ulaşana kadar yukarı yönde ilerlemeye devam eder. Bu bloklar hareket etmeye devam eder ve tüm Merkle ağacı bloklarda yakalanan işlemlerden oluşur. Blokzincir'i bu kadar güçlü, sağlam, değişime dayanıklı, entegre ve değişmez kılan yapı bu şekilde işlemektedir.



Şekil 2.5. Blokzincir Ağı

2.1.4. Blokzincir Teknolojisi Literatür Taraması

Günümüzün gelişen en önemli ve geleceğe yön verebilecek teknolojilerinden birisi olan Blokzincir teknolojisi üzerine yapılan çalışma ve makaleler gün geçtikçe artmaktadır. Chen, R. Y., Blokzincir madenciliğindeki izlenebilirlik zinciri ihtiyaçlarını karşılamak için, çıkarım mekanizmasını kullanan büyük işlem verileriyle fikir birliği kararı için yeni bir yaklaşım gerektiğini ortaya koymuş, ayrıca ANN, Takagi-Sugeno Bulanık bilişsel haritaları kullanan bir yaklaşım önermiştir[**11**]. Lei ve ark., VCS (Araç İletişim Sistemi) ağlarında içeriden gelebilecek saldırıların önlenmesi için yeni bir sertifika iptal şeması önermişlerdir. Önerilen programda Blokzincir konsepti tanıtılıp, bir Blokzincir yapısı tanıtılmıştır. Önerilen Blokzincir yapısı, PKI'nın (Açık Anahtar Altyapısı) takma setlerin sahipliğini izlemeye devam etmesini ve CRL'yi (Sertifika İptal Listesi) etkin bir şekilde dağıtmasını sağlamaktadır. Etkili bir kötü niyetli davranış raporlama mekanizması geliştirilmiş ve CRL dağıtımının bir kısmı, sertifika iptalinin genel gider iletisini en aza indirgemek için grup anahtarı yönetim şemasıyla birleştirilmiştir [**12**]. Dorri ve ark., Blokzincir teknolojisinin değişmez doğası, önceden depolanmış bir bloğun değiştirilmesini veya kaldırılmasını imkansız kıldığını ve böylece Blokzincir güvenliğini artırdığını ifade etmişlerdir. Bununla birlikte, Blokzincir kullanıcıları için özellikle Nesnelerin İnterneti (IoT) gibi büyük ölçekli ağlarda depolama, gizlilik ve maliyet zorluklarının arttığını ifade etmişlerdir. Makalelerinde, daha önce depolanmış bir işlemi kaldırmak, işlemlerini

özetleyerek kullanıcılarını ve Hizmet Sağlayıcıları (SP) güçlendiren bir Bellek Optimize Edilmiş ve Esnek Blokzincir (MOF-BC) önermişlerdir ve önerdikleri sistemin bellek tüketimini %25 azalttığını ortaya koymuşlardır [**13**]. Zhong ve ark., güvenlik, mahremiyet, düşük güç tüketimi, esneklik, zincir dışı ödemeler ve çevrimdışı ödemeler dahil altı özelliği karşılayan Blokzincir tabanlı bir SVLP (güvenli çok yönlü ışık ödeme) sistemi sunmuşlardır. Mevcut kriptografik ilkelere, yani tek yönlü işlev ve dijital imza ve blok zinciri sistemine dayanan genel bir SVLP sistemi yapısı önermişlerdir. Ayrıca, bir elliptic curve digital signature algorithm (ECDSA) şeması ve bir karma işlevi kullanarak SVLP sistemini uygulamışlardır [**14**].

Skowroński, R. çalışmasında, yeni bir Blokzincir destekli siber-fiziksel sistemler ailesi analiz etmiş, örnek bir açık ticaret, kendi kendini dengeleyen SmartGrid mimarisini açıklamıştır [**15**]. Xu, X ve ark., çalışmalarında, Blokzincir tabanlı uygulamalar için yazılım tasarım yaklaşımlarını gerçek bir dünya projesine uygulamak için, originChain yazılımını üretmişler ve originChain'in nasıl tasarlandığını ve uygulandığını göstermişlerdir [**16**]. Yang, M. ve ark., çalışan mekânlarının kaçınılmaz olarak ödeme işlemi sırasında açıklandığı mekansal kalabalığı toplama sistemindeki mevcut gizlilik sorunlarını incelemişler ve gizliliğin ihlallerini önlemek için, yeni bir Blokzincir bazlı gizliliği koruyan kalabalığa dayalı bir sistem önermişlerdir [**17**].

Viriyasitavat ve Hoonsopon, makalelerinde, Blokzincir'i tanımlamışlar ve iş süreçleri açısından Blokzincir'in özelliklerini araştırmışlardır. Ardından, özelliklere dayanarak, Blokzincir döneminde iş süreçlerinin inovasyonu için kilit teknolojilerden ve kılavuzlardan oluşan bir mimari önermişlerdir. Önerilen mimarinin 2 temel unsur vardır. Birincisi, onay sözleşmesiyle ilgili zaman tutarsızlığı sorunu ikincisi ise fikir birliği yapmaktan sorumlu düğümler hakkında ortaya çıkabilecek önyargı sorunudur. Bu sorunları aşmak için akıllı sözleşmelerle PBFT (Practical Byzantine False Tolerance) kullanmışlardır [**18**]. Lu, Y, Blokzincir teknolojisini, üç önemli blok zinciri temelli konuyu (IoT, güvenlik ve veri) ve sektördeki blok zincir uygulamalarını incelemiş ve potansiyel zorlukları ve gelecekteki yönelimlerini ortaya koymuşlardır [**19**]. Cao ve ark., dış kaynaklı EHR'lerin (Electronic Health Records) Blokzincir teknolojisi (Blokzincir temelli para birimleri, örneğin, Ethereum) kullanarak yasadışı değişikliklerden korumak için güvenli bir bulut destekli e-sağlık sistemi önermişlerdir [**20**]. Dubovitskaya ve ark., birincil bakım, tıbbi

veri araştırması ve bağlantılı sağlık gibi farklı sağlık ortamlarında uygulanmak üzere blokzincir teknolojisi uygulaması senaryoları önermişlerdir [**21**]. Xu veark. gibi sosyal açıdan faydalı faaliyetlerde bulunan araştırmacılar da vardır. Onlar, sakinleri ödüllendiren bir Blokzincir-Enabled Sosyal Kredi Sistemine (BLESS) dayanan güvenilir ve güvenli topluluklar oluşturma çalışması yapmışlardır [**22**]. Bununla birlikte Samaniego ve ark. gibi Blokzincir'i farklı alanlarda uygulayanlar da vardır. Samaniego ve ark. çalışmalarında bilgi bloklarını okuyarak ve yazarak birçok akıllı nesneyi iletmek için izin tabanlı blokzincir protokolü olan multichain'i kullanmışlar ve IoT entegrasyonuna katkıda bulunmuşlardır [**23**]. Wang ve ark. ise nesnelerin internetine blokzincir teknolojisinin eklenmesi için araştırma makalesi yazmışlardır [**24**].

Lone ve Mir, araştırmalarında, adli zincir: dijital adli gözaltı zincirine bütünlük ve kurcalama direnci getiren bir blokzincir tabanlı dijital adli gözaltı zinciri önermişler ve bununla birlikte hyperledger composer'da kavram ispatı'nı yaparak blokzincir teknolojisini adli alana taşımışlardır [**25**]. Millard gibi blokzincir 'in kanun düzenlemelerine uygunluğunu inceleyen araştırmacılar da vardır [**26**]. Buocz ve ark. makalelerinde, dağıtılmış ağların sorumluluk verme konusundaki mevcut yasal mekanizmalara nasıl meydan okuduğunu göstermek için kripto para birimi bitcoin ve genel veri koruma yönetmeliği (GDPR) örneği kullanmışlardır [**27**]. Queiroz ve Wamba, blokzincir, tedarik zinciri ve ağ teorisi ve aynı zamanda teknoloji kabul modelleri (TAM'ler) üzerine ortaya çıkan literatürden yola çıkarak, klasik birleşik kabul ve teknoloji kullanım teorisinin (UTAUT) biraz değiştirilmiş versiyonuna dayanan bir model geliştirmişlerdir [**28**].

Blokzincir uygulamaları yaygın bir şekilde konuşlandırılırken, pek çok konu henüz ele alınmamıştır [**29**]. Ancak literatürde yer alan ve blokzincir konusunda ele alınmış makaleler diğer alanların gelişimine büyük katkı sağlamaktadır. Örneğin, ağ uygulamalarının blokzincir kimlik doğrulaması üzerine çalışma yapan mohsin ve ark. bu konuda taksonomi, sınıflandırma, yetenekler, açık zorluklar, motivasyonlar, tavsiyeler ve gelecek yönergeleri ile ilgili bilgileri paylaşmışlardır [**30**]. Greenwood ve ark. ise blokzincir'i inşaat sektöründe uygulayarak gelişiminde katkıda bulunmuşlar, kavramsal modeller ve pratik kullanım örnekleri paylaşmışlardır [**31**]. Blokzincir son yıllarda dünya çapında dikkat çekmektedir. Kamu blok zincirlerinin sayısının artmasıyla, kamu blok zincirlerinin değerlendirilmesi anlamlı hale gelmektedir [**32**]. Bitcoin'in altında yatan teknoloji olan blokzincir, kuruluşlar

üzerinde stratejik etkileri olduğu düşünülen, gelişmekte olan bir finansal teknolojidir (FinTech). Du ve ark., çalışmalarının deneme aşamasının Blokzincir uygulamasından türetilmiş olsa da, yapay zekâ ve sanal gerçeklik gibi kullanım durumları bulunmayan yeni ortaya çıkan dijital teknolojilere uygulanabileceğini belirtmişlerdir [**33**]. Blokzincir tabanlı P2P mikro-grid'leri bütünsel olarak keşfetmek ve kavramsallaştırmak ve kurumsal araştırmalar ve akademik araştırmalar için pratik sonuçlar sunmak gereklidir [**34**]. IoT ağı büyük miktarda veri üretir. Bu, bu ağların izlenmesi ve kontrolünün ve paketlerin IoT ağından sunucuya aktarılmasının iletişimin çökmesine neden olabileceği anlamına gelir. Öte yandan, veritabanlarında depolanan büyük miktarda veri nedeniyle, IoT ağının izlenmesi, yüksek derecede verimliliğe sahip olmak için çok güçlü sunuculara ihtiyaç duymaktadır [**35**].

García-Magariño ve ark., yaptıkları çalışmada, gözetimi desteklemek için İHA ağlarındaki güvenliği ihlal edilmiş İHA'ları tespit etmek için bir güvenlik mekanizması sunmuştur. Bu mekanizma Blokzincir prensiplerinden ilham almakta ve bir güven politikası kullanmaktadır. Onlar çalışmalarında özetle, kötü niyetli saldırıya uğramış İHA'ların davranışları hakkında farklı seçenekler ele almıştır. Daha somut olarak, tehlikeye girmiş bir İHA'nın (1) gerçek gözlemleri atlamak ya da (2) gerçek alarmların ciddiye alınmaması için sistemi itibarsızlaştırmak üzere yanlış alarmlarla gürültü oluşturmak için iki seçeneği olduğunu belirtmişlerdir [**36**].

Araçların İnterneti'nin (IoV) avantajlarından biri, örneğin araç mesajlarını gerçek zamanlı olarak paylaşabilme yeteneği nedeniyle gelişmiş trafik güvenliği ve verimliliğidir şeklinde görüş belirten Zhang ve ark., çalışmalarını sistemin verimliliğini artırmak için Ethereum'a uyarlamışlardır [**37**]. Sağlık alanında literatür taraması yapacak olan araştırmacılar anahtar kelimeler olarak "Blokzincir" AND (" eHealth " OR "EHR" OR "electronic health records" OR "medicine ")olarak "Blokzincir" AND ("Network") gibi farklı arama kriterleri kullanmışlardır [**38**]. Ancak sağlık alanında çalışmalar hala yetersizdir. Yine Blokzincir teknolojisinin yeni teknolojilerden birisi olan bulut teknolojisi üzerinde de uygulamaları görülmektedir. Bulut tabanlı veri paylaşım sisteminde, bulut hizmet sağlayıcıları depolanan kayıtlara tüm erişimleri bilebilir. Ancak, kayıt sahibi depolanan kayda erişim günlüklerini bilmek istiyorsa, denetlemesini ister. Bulut servis sağlayıcısının güvenliği ihlal edilirse, kayıt sahibi yanlış sonuç alır. Noh ve ark., çalışmalarında, bulut depolamasında Blokzincir

tabanlı güvenli veri paylaşım sistemi önermişlerdir. Bir Blokzincir teknolojisinin özellikleri nedeniyle, kayıt sahiplerinin tam güvenilir üçüncü tarafın katılımcısı olmadan bulut sunucusundaki tıbbi kayıtlarına kimlerin erişebileceğini kontrol etmesini sağlayabilmeyi amaçlamışlardır [**39**]. Hyperledger tanıtımında, Bir Blokzincir ağının performansı 4 temel faktörden etkilendiği belirtilmiştir. Bunlar; Okuma Gecikmesi, Okuma Hacmi, İşlem Gecikmesi, İşlem Hacmidir [**40**]. Blokzincir framework'ü Hyperledger fabric, çok katı GDPR işlemleri için gerekli olanlar bile, veri işleme için herhangi bir yazılımla birlikte kullanılabilecek birçok avantaja sahiptir. Hyperledger fabric ölçeklenebilirliği, modülerlik, tüm düzeylerde kimlik yönetimi, gizlilik ve gizlilik özellikleri, dağıtılmış ve güvenli doğanın içsel blok zincirinin üzerine inşa edilmiştir [**41**].

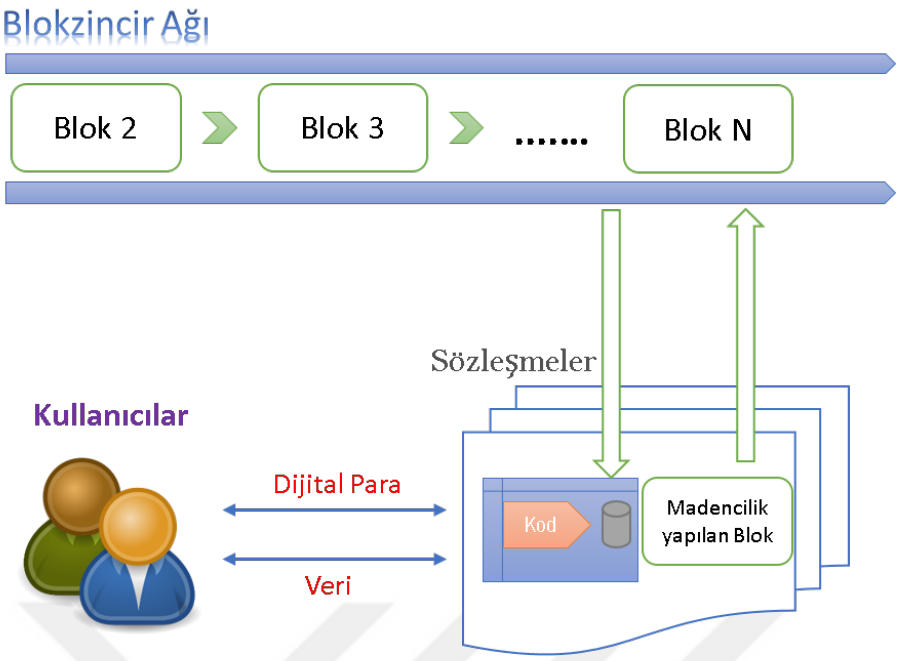
2.2. AKILLI SÖZLEŞMELER NASIL ÇALIŞIR?

Akıllı Sözleşmelerin ana ilkesi otomat makinalarının çalışma ilkesi ile benzerlik göstermektedir. İlk önce sözleşme koşulları ile birlikte varlıklar kodlanır. Kodlanan koşul ve varlıklar blok zincirinin bir bloğuna yerleştirilir. Bu blok, blokzincir ağındaki düğümler arasında birçok kez kopyalanır ve dağıtılır. Sözleşmede belirlenmiş olan tetikleyici aktifleştikten sonra koşullar gerçekleşir. Program sözleşme maddelerinin uygulanmasını otomatik olarak kontrol eder.

Bir Akıllı Sözleşme Oluşturmak; Dijital İmzalar, Kontrat Konusu, Merkezi Olmayan Bir Platform ve Anlaşma Koşulları gerektirir. Akıllı Sözleşmeler; Seçimler, Lojistik, Yönetim, Banka Sistemi, Sigorta, Emlak, IoT gibi farklı alanlarda kullanılmaktadır. Örneğin, A'nın B'ye para gönderdiği her durumda, paranın yarısının otomatik olarak C'ye gönderildiği bir sözleşme olduğu varsayılsın. Bu durum, akıllı bir sözleşme haline getirilip belgelendirilmektedir. Kod, A'dan B'ye para gönderdiğinde otomatik olarak tetiklenir ve paranın yarısı C'ye gönderilir. Bu işlem istenilen şekilde karmaşıklaştırılabilir ve faydalı hale getirilebilir.

Dağıtık defterler, akıllı sözleşmeleri ve yapılan tüm işlemleri çok güvenli ve gizli tutmaktadır.

Şekil 2.6, Akıllı Sözleşmelerin Blokzincir ağı ile etkileşimini göstermektedir.



Şekil 2.6. Akıllı Sözleşme Nasıl Çalışır?

Bitcoin, basit akıllı sözleşmelerin ilk örneğidir çünkü ilk kripto paradır. Şu anda, Ethereum akıllı sözleşmeleri bitcoin'den farklıdır. Geliştiriciler, Ethereum'un kod yapısını kullanarak çok çeşitli amaçlara hizmet edebilecek akıllı sözleşmeler oluşturabilmektedirler.

Akıllı sözleşmelerin bir dezavantajı, ücretsiz olmamasıdır. Bu işlemler için düşük de olsa bir ücret ödenmektedir. Bir diğeri de, akıllı sözleşmeler henüz "kusursuz" değildir. Şu anda hack'lenmesi çok zor olsa da, kodları hazırlayan kişiler tarafından yapılabilecek bazı hatalar ciddi sorunlara neden olabilmektedir. Alıcı ve tedarikçi arasındaki bir akıllı sözleşmenin nasıl işlediği, Şekil 2.7'de gösterilmektedir.

Şekil 2.7. Alıcı ve tedarikçi arasındaki akıllı sözleşme örneği [**42**]

2.2.1. Akıllı Sözleşme Platformlarının Karşılaştırılması

Akıllı Sözleşme Platformları; akıllı sözleşmelerin tasarlanmasını, geliştirilmesini ve dağıtımını üstlenen blokzincir tabanlı platformlardır. Bu platformların performansı, blokzincir teknolojisinin gerçek dünya uygulamalarıyla etkileşim kurmasını ve başarılı bir şekilde ticarileştirilmesini sağlamaktadır.

Akıllı sözleşmeler, düzgün çalışabilmeleri için, belirli uygun ortam ve platformlara ihtiyaç duymaktadırlar. Akıllı sözleşmeler, her kullanıcının, özel şifreli kodlar yardımıyla imza oluşturulabilmesi için, açık anahtar şifreleme özelliğini desteklemelidir. Bu özellik mevcut kripto paraların çoğunluğunda kullanılan sistem ile aynıdır. İkinci olarak ise, merkezi olmayan, açık ve sözleşme tarafları tarafından güven sorunu teşkil etmeyecek bir veri tabanına ihtiyaç vardır. Buna ek olarak, sözleşmenin sağlıklı işlemesi için, ortamın kendisinin de merkezi olmayan özellik ile donanmış olması gerekmektedir. Akıllı sözleşmeler çeşitli blokzincir ağlarında yürütülebilmektedir. Bu ağlardan en önemli ve verimlisi Ethereum Blokzincir ağıdır. Son olarak ise, akıllı sözleşmelerde kullanılan dijital verilerin tamamen güvenilir olması, sağlıklı yürütülmesi açısından önemlidir. Bunun sağlanması için ise günümüzde çoğu yazılımda otomatik olarak uygulanabilen güvenli bağlantılar (HTTPS, SSL Güvenlik Sertifikaları vb.) kullanılmaktadır.

Çizelge 2.1' de 6 ayrı akıllı sözleşme ortamı verilmiş ve 5 farklı özellikleri karşılaştırılmıştır.

Çizelge 2.1. Akıllı Sözleşmeler ve Blokzincir oluşturma platformlarının karşılaştırılması

#	Yürütme Ortamı	Consensus Mekanizması	Dil	Kripto Para	Uygulamalar
Ethereum	EVM	Proof-of-work	Solidity	Ether	Merkezi Olmayan Borsalar
EOS	Web Asmbl	BFT-DPOS	C++	EOS Token	Kâr paylaşımları
Tron	Tron VM	Tron (DPOS)	Solidity	Tronix	Oyun
Steller	Docker	Steller Cons. Protocol	C++, GO	Lumen	Petrol Ticareti
Hyperledger Fabric	Docker	Custom	Javascript, Java, GO	-	Tedarik zinciri
NEM	JVM	Proof of Importance	NEM,Java	XEM	CryptoC.

2.2.2. Akıllı Sözleşmeler Literatür Taraması

Blokzincir teknolojisi üzerine yapılan çalışma ve makaleler gün geçtikçe artmakta ve katkı sağladığı alanlar da ilgi çekmeye başlamaktadır. 1970'li yıllarda ortaya çıkmasına rağmen Blokzincir ile beraber asıl gelişmesini gösteren Akıllı Kontratlar da günümüzde önemli değere erişmiştir. Özellikle son birkaç yılda akıllı kontratlar ile ilgili çalışmalar yoğunlaşmıştır. Akıllı sözleşme teknolojisi geleneksel endüstri ve iş süreçlerini yeniden şekillendirmektedir. Blok zincirlere gömülü olan akıllı sözleşmeler, genel tabirle, bir sözleşmenin şartlarının güvenilir bir üçüncü tarafın müdahalesi olmadan otomatik olarak uygulanmasını sağlamaktadır. Sonuç olarak, akıllı sözleşmeler yönetimi azaltabilir ve hizmet maliyetlerinden tasarruf edebilir, iş süreçlerinin verimliliğini artırabilir ve riskleri azaltabilir.

Akıllı sözleşmeler, iş süreçlerinde yeni inovasyon dalgasına yön vermeyi vaat etse de, ele alınması gereken bazı zorluklar vardır. Zheng ve ark., akıllı kontratlarda bu zorlukların neler olduğunu, kullanılan platformları ve bu zorlukların nasıl aşılabileceğini açıklayan bir çalışma ile literatüre katkı sağlamışlardır [**42**]. Bu çalışmalardan birisini Wang ve ark., gerçekleştirmiştir. Onlar veri gizliliği koruma mekanizmaları, çeşitli siber saldırılarla karşı karşıya kalacak kadar sağlam olmadığını belirtip, bu zorlukların üstesinden gelmek için, izin verilen Blokzincir Hyperledger Fabric üzerinden akıllı sözleşmelere dayanan yeni bir finansal kredi yönetim sistemi olan Blokzincir (LoC) kredisi önermişlerdir [**43**]. Akıllı sözleşmelerin güvenliğini sağlamak amacıyla birçok çalışmalar yapılmaya devam etmektedir [**44**, **45**, **46**, **47**, **48**, **49**].

Geliştiricinin merkezi olmayan ve güvenli Blokzincir uygulamasını dağıtmasına yardımcı olabilecek akıllı sözleşme, günümüzde modern Nesnelerin Interneti (IoT) ekosistemi için en umut verici teknolojilerden biridir [**50**, **51**, **52**]. Bununla birlikte, Ethereum akıllı sözleşmesinde dış IoT ortamıyla iletişim kurma yeteneği yoktur. Akıllı sözleşmelerin zincir dışı verileri getirmesini sağlamak için Liu ve ark., makalelerinde, Blokzincir özellikli IoT ortamı için uygun maliyetli ve esnek bir veri taşıyıcı mimarisi önermişlerdir [**53**]. Fan ve ark., çalışmalarında, Ethereum'da yeni bir merkezi olmayan denetim akıllı sözleşmesi önermişlerdir. TPA(üçüncü bölüm denetçisi)'yı tasarlanmış akıllı bir sözleşme ile değiştirerek, herhangi birinin yarı dürüst TPA hakkında endişelenmeden

Ethereum'dan denetim sonucunu alabileceği merkezi olmayan bir denetim şeması (Dredas) önerilmiştir [**54**]. Carvalho [**55**] gibi akıllı sözleşmeleri oyun alanına taşıyan araştırmacılar da bulunmaktadır. Akıllı sözleşmeler ile ilgili çeşitli uygulamalar geliştiren çalışmalar bulunmaktadır [**56**, **57**, **58**, **59**, **60**, **61**, **62**, **63**, **64**]. Akıllı sözleşmeleri ve kalite değerlendirme modellerini birleştiren çalışmalarında Yu ve ark., meyve suyu üretimi için akıllı bir kalite izleme sistemi sunmuşlardır [**65**]. Wang ve ark., çalışmalarında, kitle kaynak kullanımı yoluyla yeni bir merkezi olmayan bilgi grafiği oluşturma yöntemi önermişler ve kitle kaynak kullanımının iş mantığı, şeffaflığı, bütünlüğü ve denetlenebilirliği garanti etmek için blok zinciri destekli akıllı sözleşmeleri uygulamışlardır [**66**]. Akıllı sözleşmelerin oluşturulma amaçlarından birisi de üçüncü parti denetçilerin ortadan kaldırılmasıdır.

Bu nedenler üçüncü parti denetçilerin yerini almak için Wang ve ark., sistemlerinde, genel bulut depolama denetimi ve Blokzincir tabanlı adil ödeme akıllı sözleşmesi tasarlayarak yukarıda belirtilen dezavantajı ele almayı amaçlamışlardır. Onlar, veri sahibinin ve bulut hizmet sağlayıcısının (CSP), Blokzincir tabanlı bir akıllı sözleşme ile yürütülmesini sağlamışlardır [**67**]. Chang ve ark.', çalışmalarında, dijital para birimi kullanan alternatif bir ödeme yönteminin uygulanabilir olduğu ve ödeme sağlama süresini azaltabileceği bulmuşlardır. Onlara göre gerçekten de değer yaratma, mevcut iş süreçlerine bir Blokzincir tabanlı çerçeve ekleyerek elde edilebilmektedir. Ayrıca, onların çalışmaları, işletme yöneticilerine akıllı sözleşme perspektifinden daha iyi bir olay değerlendirme modeli sunmakla kalmayıp aynı zamanda operasyon yönetimi açısından daha iyi verimlilik sağlamaktadır [**68**].

Mhaisen ve ark, akıllı şebekelerde dağıtılmış Pil Enerjisi Depolama Sistemlerini (BESS) kontrol etmek için akıllı sözleşmelerin kullanılmasını ve potansiyel siber saldırılara karşı güvenli çalışma ortamının sağlanması için çalışmalar yapmışlardır. Onlar, önerdikleri kontrol yaklaşımı ile iletişimde çoğaltılan yürütme ve durum mutabakatı özelliklerini ve kriptografik teknikleri kullanarak BESS'lerin güvenliğinin artmasının sağlanmasını amaçlamışlardır [**69**]. Çoğu teknolojide olduğu gibi, akıllı sözleşmelerle ilişkili potansiyel güvenlik açıkları, güvenlik tehditleri ve diğer çeşitli sorunlar vardır. Çeşitli iş mantıklarının yanı sıra platform güvenlik sınırlamaları ve açıkları nedeniyle güvenli akıllı sözleşmeler yazmak son derece zor olabilir. Bu güvenlik açıklarını azaltmak için son zamanlarda resmi

yöntemler savunulmuştur. Singh ve ark., literatürde az bulunan blok zincirler üzerindeki tüm akıllı sözleşmeyle ilgili platformlar üzerinde mevcut resmileştirme araştırması hakkında ilk kez bir çalışma sunmayı amaçlamışlardır [**70**]. Bulut bilişimin ortaya çıkmasıyla birlikte, dış kaynak kullanımı bir hesaplama haline gelmiştir. Bulut tarafından gerçekleştirilen hesaplamalara daha fazla güven duyan bir müşteri, döndürülen sonuçların doğruluğunu doğrulayabilmelidir. Reddy ve ark., Hesaplama ve doğrulamalar bulut tabanlı bir sistem tarafından 2 farklı sistemle yapıldığını açıklayıp, Akıllı sözleşmelerdeki son gelişmelerden hareketle, akıllı sözleşmeleri kullanarak farklı doğrulanabilir bilgi işlem teknikleri için protokoller önermişlerdir. Ayrıca gelecekteki çalışmalarda, esas olarak dış kaynaklı bir sorunun giriş ve çıkışlarının gizliliğini sağlayacak akıllı sözleşmeler kullanarak doğrulanabilir hesaplamalar için adil protokollerin geliştirilmesine odaklanacaklarını belirtmişlerdir [**71**].

Akıllı sözleşmelerin network alanında kullanımlarından birisini Unal ve ark., gerçekleştirmişlerdir. Onlar, 5G ağlarındaki akıllı sözleşmeler için politika uyum kontrolü ve doğrulaması için bir yaklaşım sunmuşlardır [**72**]. de Graaf gibi, programcıların daha iyi akıllı sözleşmeler oluşturmak için avukatlarla birlikte çalışmasını ve yasa koyucunun güvenilir üçüncü taraflarca akıllı sözleşmeler kodunu denetlemeye ve akıllı sözleşmeleri ıslak mürekkep imzalarıyla yazılı sözleşmelerle otomatik olarak eşitlemeye ilişkin yasalara odaklandığını savunan araştırmacılar da vardır [**73**]. Farklı bir alan olarak, Li ve ark., akıllı sözleşme ve blokzincir ile yenilenebilir enerji üretimi ve konut, ticari ve endüstriyel sektörlerden heterojen son kullanıcıları içeren dağıtılmış bir enerji sisteminin tasarımını ve yönetimini incelemişlerdir [**74**].

KIYAK ve ark., blokzincir teknolojisinin sağlık alanına yansımaları konusunda yazılmış hiçbir Türkçe yayının olmamasının ve bu teknolojinin temel işleyişini bile ele alan Türkçe yayın sayısının az olması nedeniyle akıllı kontratlarda sağlık alanına bir bakış getirmişlerdir. Makalelerinde Blokzincir teknolojisinin ve akıllı kontratların nasıl çalıştığı, anlamak için kodlama bilgisi gerektirmeyecek şekilde ifade edildikten sonra sağlık alanındaki üç uygulama örneğine yer verilerek bu konudaki Türkçe literatüre katkıda bulunmayı amaçlanmışlardır [**75**]. Staifi ve Belguidoum, akıllı ev sistemleri üzerine çalışmışlar ve akıllı sözleşmelere dayalı yaşlı sağlık durumuna duyarlı bir akıllı ev sistemi önermişlerdir [**76**]. Chu ve ark., Blokzincir teknolojisinin ne olduğu, Akıllı kontratların

yapısı ve işleyişi ve gelecekte akıllı kontratların durumu hakkında çalışma yapmışlardır [**77**]. Akıllı konteynerler aracılığıyla gönderilen ürünleri içeren verimli tedarik zinciri yönetimi için bir Blokzincir tabanlı çözüm öneren Hasan ve ark., önerdikleri çözüm ile, gönderen ve alıcı arasındaki etkileşimleri yönetmek için ethereum blok zincirindeki akıllı sözleşmelerin özelliklerini kullanmışlardır [**78**]. Blokzincir'in ortaya çıkmasıyla, akıllı sözleşmeler, işlemlere ekledikleri yüksek özelleştirilebilirlik nedeniyle en çok aranan teknolojilerden biri haline gelmiştir. Macrinici ve ark., makalelerinde, Blokzincir teknolojisi içindeki akıllı sözleşmelerin bilgi birikimine katkıda bulunmayı amaçlamışlardır. Sistematik bir haritalama çalışmasına dayanarak, sorunları ve ilgili çözümleri hakkında geniş bir perspektif ile birlikte, alandaki araştırma trendlerini sunmuşlar ve en iyi yayın kaynakları, kanallar, yöntemler ve yaklaşımlar ile gruplandırılmış, tanımlanan 64 makaleyi derlemişlerdir [**79**]. Akıllı sözleşme için girdi filtresi tabanlı güvenli framework sunan wang ve ark., uygulamanın değerlendirmesine dayanarak, bu framework'ün kabul edilebilir bir gaz tüketimi ve performans kaybı altında giriş koruma işlevini etkili bir şekilde sağlayabildiğini göstermişlerdir [**80**].

3. MATERYAL VE METOT

3.1. AKILLI SÖZLEŞME NEDİR? NEDEN KULLANILIR?

Akıllı sözleşme, iki veya daha fazla taraf arasındaki bir anlaşmanın veya sözleşmenin şartlarını otomatik olarak uygulayan ve kolaylaştıran, kendi kendini yürüten bir bilgisayar programıdır. Merkezi olmayan ve dağıtılmış bir defter teknolojisi olan bir blockchain ağı üzerinde çalışır. Akıllı sözleşmeler kavramı ilk olarak 1990'ların başında bilgisayar bilimcisi Nick Szabo tarafından önerilmiştir.

Akıllı bir sözleşmenin temel özellikleri şunları içerir:

- **Otomasyon:** Akıllı sözleşmeler, belirli koşullar karşılandığında önceden tanımlanmış eylemlerin yürütülmesini otomatik hale getirir. Sözleşmenin uygulanmasını denetlemek için aracılara veya arabuluculara olan ihtiyacı ortadan kaldırarak maliyetleri ve olası insan hatalarını azaltırlar.
- **Güven:** Akıllı sözleşmelerin kodu ve yürütülmesi, tüm işlemlerin şeffaf ve değişmez bir kaydını sağlayan bir blok zincirine kaydedilir. Bu özellik, blok zincirinde konuşlandırıldıktan sonra sözleşmenin şartlarını manipüle etmek zorlaştığından taraflar arasındaki güveni artırır.
- **Güvenlik:** Akıllı sözleşmeler, güvenlik ve bütünlük sağlamak için kriptografik teknikler kullanır. Blok zincirine yerleştirildikten sonra, yetkisiz erişime karşı korunurlar ve bu da onları oldukça güvenli hale getirir.
- **Merkeziyetsizlik:** Akıllı sözleşmeler, merkezi olmayan bir bilgisayar ağı (blok zinciri) üzerinde çalışır ve merkezi bir otoriteye olan ihtiyacı ortadan kaldırır. Bu ademi merkeziyetçilik, tek hata noktası riskini azaltır ve sistemi daha dirençli hale getirir.

Akıllı sözleşmeler, aşağıdakiler gibi çeşitli endüstrilerde ve senaryolarda kullanılabilir:

- ***Finansal Hizmetler:***Süreçleri kolaylaştırmak ve gecikmeleri azaltmak için ödemeler, krediler ve sigorta talepleri dahil olmak üzere finansal işlemlerin otomatikleştirilmesini sağlamak.
- ***Tedarik Zinciri Yönetimi:***Bilgi akışını otomatikleştirerek ve ürün menşeilerini doğrulayarak malları izleme ve tedarik zincirinde şeffaflığı sağlamak.
- ***Emlak:***Emlakçılar gibi aracılara ihtiyaç duymadan mülk transferleri ve kiralama sözleşmeleri sürecini kolaylaştırmak.
- ***Oylama Sistemleri:***Seçimler veya karar verme süreçleri için kurcalamaya dayanıklı ve şeffaf oylama sistemleri oluşturmak.
- ***Oyun ve Eğlence:***Oyun içi varlıkları ve işlemleri yönetmek için yerleşik akıllı sözleşmelere sahip merkezi olmayan uygulamalar (DApp'ler) ve oyunlar geliştirmek.
- ***Fikri Mülkiyet:***Fikri mülkiyet haklarını yönetme ve telif haklarını yaratıcılara otomatik olarak dağıtmak.

Akıllı sözleşmelerin kullanımı, artan verimlilik, düşük maliyetler, gelişmiş güvenlik ve taraflar arasında artan güven dâhil olmak üzere çok sayıda avantaj sunar. Bununla birlikte, akıllı sözleşme kodunun kapsamlı bir şekilde denetlendiğinden ve güvenli olduğundan emin olmak önemlidir. Çünkü koddaki herhangi bir hata veya güvenlik açığı, sözleşmelerin kendi kendini yürütme doğası nedeniyle önemli sonuçlar doğurabilmektedir.

3.2. AKILLI SÖZLEŞME YAZIMINDA KULLANILAN ARAÇLAR

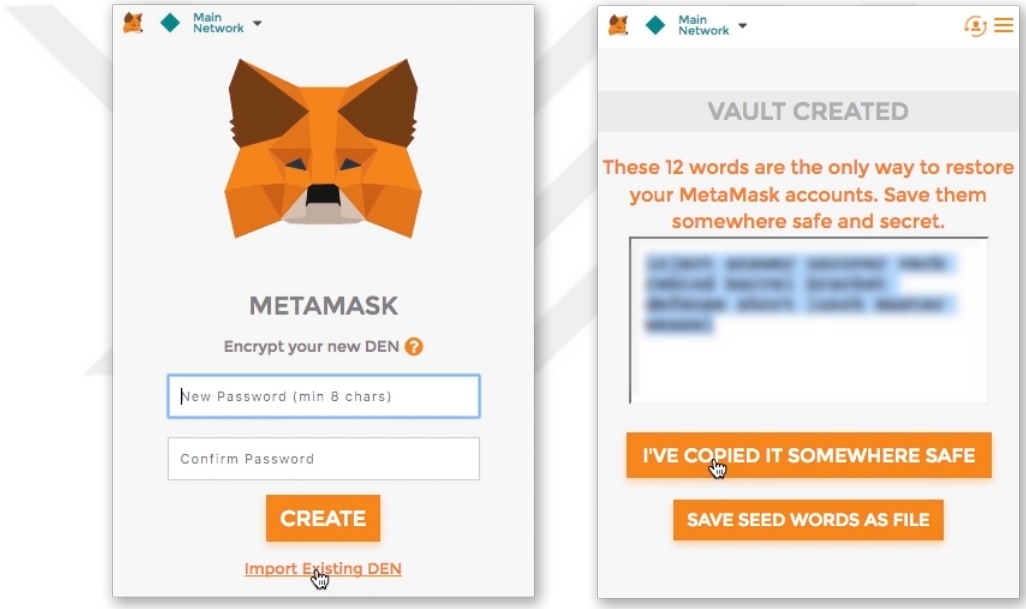
3.2.1. Metamask

Metamask, web sitesinde de tanıtıldığı gibi [**IEEEexample:81**], tarayıcılar aracılığıyla yarının dağıtılmış web sitelerinin ziyaret edilmesini sağlayan bir köprüdür. Bu, normal tarayıcılar ile Ethereum blok zinciri arasında bir köprü görevi gördüğü anlamına gelir. Tam bir Ethereum düğümü çalıştırmadan, doğrudan tarayıcı ile Ethereum dApp'leri çalıştırılmasına olanak tanır.

Metamask, bir Ethereum Düğümü olarak Ethereum ağının bir parçası olmadan dApp'leri çalıştırılmasına izin veren bir tarayıcı uzantısıdır. Ethereum ayrıca kullanıcılarının akıllı sözleşmeler adı verilen işlem yönergeleri yazmasına izin verir. MetaMask, yalnızca Ethereum kripto para birimlerinin anahtarlarını saklamak için kullanılabilir. MetaMask,

Etherleri (Ethereum para birimi) içeren Ethereum cüzdanını yönetir ve bir dApp aracılığıyla Eterlerin gönderilip alınmasını sağlar. Metamask, birçok tarayıcıda çalışabilen basit bir tarayıcı uzantısıdır. Yüklemek için, metamask web sitesi ziyaret edilip, uzantı olarak yüklenebilmektedir. Uzantı yüklendiğinde tarayıcının sağ üst kısmında 3D Tilki görüntüsü belirecektir. Üzerine tıklandığında, kullanım şartlarını kabul etmek gerekir.

Ether gönderip almak ve ayrıca dApp'leri çalıştırmak için yeni bir Ethereum hesabı oluşturabilmektedir. Bir hesap oluşturduktan sonra, parolası unutulan hesabı geri almak için kullanılabilecek 12 kelimelik bir liste verilmektedir. (Şekil 3.1). Hesaba giriş yapmak için bu listenin sırasıyla girilmesi istenmektedir.



Şekil 3.1. Metamask Hesap Oluşturma

Metamask ile gerçek paraya mal olan gerçek Eterleri kullanan ana ağı kullanılabilmekte veya Ropsten Test Network gibi test ağlarında bazı dApp'leri denenebilmektedir.

Avantajları

1. *Açık Kaynak:*Bu, tüm MetaMask kodunun çevrimiçi, ücretsiz ve erişime açık olduğu anlamına gelir. Açık kaynaklı yazılım topluluk tarafından gözden geçirilebilir ve güncellenebilir, bu da sürekli olarak geliştirilebileceği anlamına gelir.

2. *HD ayarları:* Hiyerarşik belirleyici ayarlar, kullanıcıların hesaplarını yedeklemelerine yardımcı olur. Bunu, kullanıcıya bir kelime listesi vererek yaparlar.
3. *Yerleşik madeni para satın alma:*MetaMask, kullanıcıların kripto para birimi satın alabileceği iki borsayla doğrudan bağlantı kurar. Kullanıcılar, Ether veya ERC-20 jetonları, Ether ve ShapeShift satın almak için Coinbase'i seçebilirler.
4. *Müşteri desteği:* MetaMask, Ethereum ağına olabildiğince çok kişinin dâhil olmasını istemektedir. Ana sayfasında bir video tanıtımı ve ayrıntılı bir destek sayfası bulunmaktadır.
5. *Basit arayüz:*Bir kez kurulduktan sonra, MetaMask'ın kullanımı çok basittir. Tüm özellikleri net bir şekilde düzenlenmiştir. Bu nedenle yeni başlayanlar için bile para göndermek ve almak kolaydır.
6. *Yerel anahtar saklama:*Birkaç cüzdan sağlayıcısı anahtarları sunucularında saklar. MetaMask anahtarları kullanıcının kendi tarayıcısında saklanır. Bu, kullanıcıya hem genel hem de özel anahtarları üzerinde daha fazla yetki vermektedir.
7. *Topluluk:* MetaMask, Ethereum topluluğunun önemli bir parçasıdır. Bir milyondan fazla aktif kullanıcısı bulunmaktadır.

Dezavantajları

1. *Tarayıcı erişimi:* Metamask, bilgilerinizin hiçbirine erişemez. Ancak yüklendiği tarayıcı erişebilmektedir. Tarayıcı özel kodlara erişemez, ancak uygulamanın ne zaman ve nasıl kullanıldığı hakkında bilgi toplayabilir. Mozilla ve Google, kripto topluluğunda çok popüler değildir. Birçok kripto kullanıcısı, bu şirketlerin kendileri hakkında bilgi toplamasına izin vermekten rahatsızlık duymaktadır. Bu, bazı potansiyel kullanıcıların Metamask cüzdanını denemesini engelleyebilmektedir.
2. *Çevrimiçi:* Çevrimiçi cüzdanların avantajları ve dezavantajları vardır. Ana dezavantajlardan biri güvenliktir. Çevrimiçi depolanan herhangi bir bilgi, çevrimdışı depolanan bilgilerden çok bilgisayar korsanları tarafından risk altındadır. Metamask kendi başına yeterli güvenliği sağlamaz.

3.2.2. NodeJs

Chrome'un V8 JavaScript motorunda bulunan Node.js, açık kaynaklı bir sunucu tarafı çalışma zamanı ortamı olarak hizmet verir. JavaScript kullanarak yüksek düzeyde ölçeklenebilir sunucu tarafı uygulamaları oluşturmak için platformlar arası çalışma zamanı ve olay odaklı, engellemeyen (asenkron) bir G/Ç ortamı sağlar. Node.js, REST API sunucusu, web uygulaması, komut satırı uygulaması, gerçek zamanlı sohbet uygulaması ve diğerleri gibi çeşitli uygulama türlerini oluşturmak için kullanılabilir. Bununla birlikte, esas olarak PHP, Java veya ASP.NET gibi web sunucuları gibi ağ uygulamaları geliştirmek için kullanılır.

Node.js ne için kullanılır?

- Sosyal Medya Ağı için Arka Uç
- Tek Sayfalı Uygulama (SPA) Geliştirme
- Sohbet robotları
- Veri Akışı
- IoT Uygulama Geliştirme

Bazı popüler Node.js Örnek Olayları

* Netflix
* LinkedIn
* Uber

Node.js'i özel yapan nedir?

– Ölçeklenebilirlik
– Yüksek performans
– Büyük Topluluk Desteği
– Geliştirme Hızını Artırma
– Uygulama Geliştirme Özgürlüğü
– Tek Programlama Dili
– Fullstack ve MEANstack'e Katkı

– Zengin Ekosistem

Node.js'nin kullanılmaması gereken yerler

✘ Arka Uçta İlişkisel Veritabanı ile Sunucu Tarafı Web Uygulaması
✘ Ağır Sunucu Tarafı İşleme
✘ Yoğun CPU Hesaplamaları
✘ Araçlarda Kusur

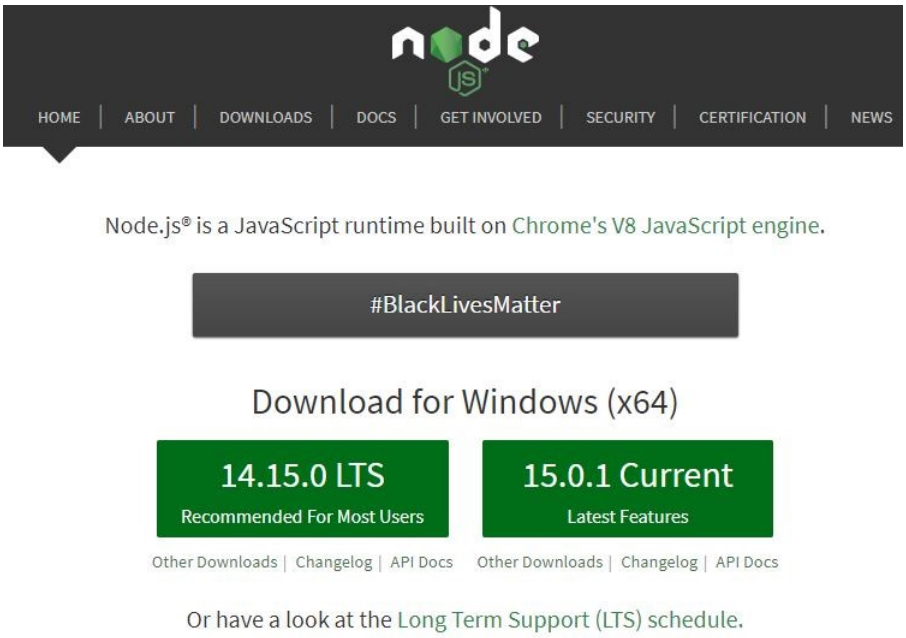
Avantajları

✓ Node.js, MIT lisansı altında açık kaynaklı bir çerçevedir. (MIT lisansı, Massachusetts Institute of Technology (MIT) kaynaklı ücretsiz bir yazılım lisansıdır.)
✓ Tüm sunucu tarafı uygulamasını oluşturmak için JavaScript kullanır.
✓ Çıplak minimum modülleri içeren hafif bir framework'tür. Bir uygulamanın ihtiyacına göre diğer modüller dâhil edilebilir.
✓ Varsayılan olarak asenkrondur. Bu nedenle diğer çerçevelerden daha hızlı performans gösterir.
✓ Windows, MAC veya Linux üzerinde çalışan çapraz platform framework'üdür.

Node. js geliştirme ortamı Windows, Mac, Linux ve Solaris'te kurulabilir. Herhangi bir platformda bir Node.js uygulaması geliştirmek için aşağıdaki araçlar / SDK gereklidir.

➢ Node.js
➢ Node Package Manager (NPM)
➢ IDE (Integrated Development Environment) veya TextEditor

Tüm gerekli araçlar için Node. js resmi web sitesini ziyaret edilebilmektedir [**IEEEexample:82**]. İşletim Sistemine göre işletim sistemini otomatik olarak algılar ve indirme bağlantısını görüntüler. Örneğin, 64 bit Windows işletim sistemi için aşağıdaki indirme bağlantısını gösterecektir. Şekil 3.2, Node.js resmi web sitesine ait ekran görüntüsünü göstermektedir.

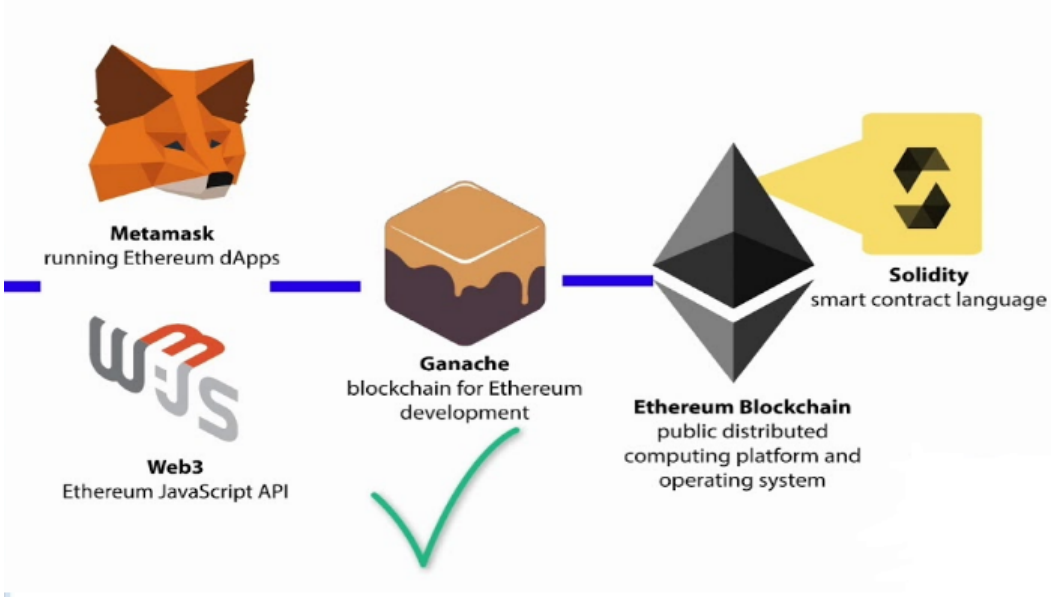


Şekil 3.2. Nodejs kurulumu

3.2.3. Ganache

Web sitelerinde de belirtildiği gibi [**IEEEexample:83**], Ganache, hızlı Ethereum ve Corda dağıtılmış uygulama geliştirme için kişisel bir blok zinciridir. Ganache tüm geliştirme döngüsü boyunca kullanılabilmektedir; dApp'ların güvenli ve belirleyici bir ortamda geliştirilmesi, dağıtması ve test edilmesini sağlar. Ethereum akıllı sözleşmeleri, Ethereum blok zincirindeki işlemler bağlamında yürütülen programlardır. Ethereum Ganache, kullanıcıların blok zinciri ortamlarını yerel olarak yeniden oluşturmalarına ve akıllı sözleşmeleri test etmelerine olanak tanıyan bir dizi geliştirici aracı olan Truffle Suite'in bir parçasını oluşturur. Ethereum blok zincirinde akıllı sözleşme yürütme, diğer yazılım türlerinden çok farklıdır. Her şeyden önce, zincir üzerindeki bağlam dış dünyayla iletişim kurmayı zorlaştırmaktadır. Blok zinciri ile iletişim kurmanın tek yolu işlemler olduğundan, bir konsola yazmak gibi basit giriş ve çıkış işlemleri mümkün değildir. İkinci olarak, blok zincirinin işlemsel doğası, akıllı bir sözleşmeyle tüm durum değiştiren etkileşimlerin doğası gereği asenkron olduğu anlamına gelir. Bu, işlemler gönderildiğinde, işlem bir bloğa dâhil edilerek onaylanana kadar etkilerin görünmeyeceği anlamına gelir. Son olarak, blok zinciri ortamı, çalıştırılabilecek koda, çoğunlukla her işlemle ilişkili maliyetle ilgili bazı özel kısıtlamalar getirir. Programcılar, blok gaz limiti veya belirli işlevlerin gaz payı dâhilinde kaç işlemin güvenli bir şekilde yürütülebileceği gibi faktörleri göz önünde bulundurmalıdır.

Diğer zorluklar arasında rastgele sayılar üretmenin zorluğu, işlem sıralaması bağımlılıkları ve tüm blok zinciri işlemlerinin genel doğası sayılabilir. Kısacası, akıllı sözleşmelerin programlanması zordur. Ek olarak, bir kez konuşlandırıldıktan sonra akıllı sözleşmeler değiştirilemez ve her dağıtımın ilişkili bir maliyeti vardır. Bu nedenle, işleri ilk seferde doğru yapmak, genellikle yalnızca kritik altyapılardaki veya havacılıktaki kontrol yazılımı gibi yüksek riskli uygulamalardaki yazılımlarla ilişkili bir kritikliğe sahiptir. Üretime geçmeden önce akıllı sözleşmelerin hatalarını ayıklamak ve test etmek için, geliştiricilerin, dağıtım maliyetleri ve işlem gecikmelerinin ek rahatsızlığı olmadan, yerel olarak blok zinciri ortamlarını yeniden oluşturmalarına izin vermek çok önemlidir. Ethereum için bir dizi geliştirici aracı olan Truffle Suite, bu amaç için tasarlanmış bir araç olan Ethereum Ganache'yi içermektedir. Şekil 3.3, Ganache tool'unun solidity dili, blokzincir ağı, metamask ve web3 ile etkileşimini göstermektedir.



Şekil 3.3. Ganache: Kişisel Blokzincir

Ethereum Ganache, geliştirme ve test için tasarlanmış yerel bir bellek içi blok zinciridir. Test Ether ile finanse edilen bir dizi hesabın kullanılabilirliği de dâhil olmak üzere gerçek bir Ethereum ağının özelliklerini simüle eder. Etheruem Ganache, kullanıcı ara yüzlü grafiksel uygulama ve komut satırı sürümü olmak üzere iki sürüm de mevcuttur. İlki, projenin web sitesinden birkaç platform için indirilebilirken, ikincisi aşağıdaki komutla NPM paket yöneticisi kullanılarak yüklenebilmektedir:

- npm install -g ganache-cli

Grafik sürümün öğesine tıklanarak veya ganache-cli çalıştırılarak başlatıldığında, her iki sürüm de yerel ana bilgisayarın bir bağlantı noktasından gerçek bir Ethereum düğümüne bağlanıldığı gibi erişilebilen bir RPC arayüzü sunar. Bu, akıllı sözleşmelerin kolayca uygulanmasına ve en kolay şekilde Truffle Suite'te bulunan diğer akıllı sözleşme etkileşimi ve test araçlarıyla çeşitli şekillerde test edilmesine olanak tanır. Böyle bir kurulumla geliştiriciler, hesap oluşturma ve finanse etme konusunda endişelenmeye gerek kalmadan ve anında blok zinciri yanıtıyla akıllı sözleşmelerini ve ilgili uygulamalarını kolayca test edebilirler.

Kullanımı:

Ethereum Ganache'yi kurulumu ve kullanımı basit bir işlemdir. Grafik kullanıcı arayüzünü başlattıktan sonra, kullanıcı hızlı başlangıç veya özelleştirilmiş çalışma alanı arasında seçim yapabilir. İlki, kullanılacak Ethereum Sanal Makinesi (EVM), gas fiyatı (gas=ethereum'un en küçük para birimi), gas limiti ve yerel ana bilgisayara sunulacak RPC arayüzü gibi parametreler için bazı temel varsayılan değerlerle kullanıma hazır bir bellek içi blokzinciri kurar. Ayrıca on adet finanse edilmiş ve kilidi açılmış hesap hazırlar. Test sırasında işlemler bir işlemler sayfasında incelenebilirken, akıllı sözleşmelerin durumu sözleşmeler sayfasından izlenebilmektedir. Test blok zincirinde yayınlanan tüm olayları kaydeden bir olay sayfası da vardır.

Gelişmiş Özellikleri:

Temel kurulum, çoğu akıllı sözleşmeyi dağıtmak için yeterlidir ve geliştiriciler hiçbir zaman hızlı başlangıç çalışma alanının ötesine geçmemektedir. Bununla birlikte, bazen belirli bir EVM sürümü veya gas fiyatı gibi belirli blokzinciri koşullarını simüle etmek gerekli olabilmektedir. Ayrıca, RPC arabirimini farklı bir bağlantı noktasında başlatmak, daha fazla hesap oluşturmak veya anımsatıcı bir çekirdek ifadeden belirli anahtarlar ve hesaplar oluşturmak gerekli olabilmektedir.

Bu işlevler, özel çalışma alanı tarafından kolaylaştırılmaktadır. Komut satırı sürümünü kullanarak, çalışan bir ağı çatallamak bile mümkündür. Yani bir kullanıcı belirli bir blok numarasında tam durumu yeniden oluşturabilmektedir. Örneğin bunun bir kullanımı, test için mevcut Ethereum ana ağını kopyalamaktır.

Özetle, Ganache, zincirin nasıl çalıştığını kontrol ederken testlerin çalıştırılması, komutların yürütülmesi ve durumun incelenmesi için özel bir Ethereum blokzinciri oluşturulmasına olanak tanır. Ana zincir üzerinde yapılacak tüm eylemleri için maliyet olmadan gerçekleştirme yeteneği verir. Birçok geliştirici, geliştirme sırasında akıllı sözleşmelerini test etmek için bu özelliği kullanmaktadır. Gelişmiş madencilik kontrolleri ve yerleşik bir blok gezgini gibi kullanışlı araçlar sağlar.

3.2.4. Ethereum Cüzdanları

Ethereum cüzdanları, Ethereum hesabı ile etkileşim kurmaya izin veren uygulamalardır. Bankasız bir internet bankacılığı uygulaması gibi düşünülebilmektedir. Cüzdan, bakiyenin okunmasına, işlemlerin gönderilmesine ve uygulamalara bağlanmaya izin verir. Para göndermek ve ETH yönetmek için bir cüzdana ihtiyaç vardır. Cüzdan yalnızca Ethereum hesabını yönetmek için bir araçtır. Bu, cüzdan sağlayıcıları istenilen zaman değiştirebileceği anlamına gelir. Birçok cüzdan, tek bir uygulamadan birkaç Ethereum hesabı yönetilmesine de izin verir.

Wallet, Türkçe'de "cüzdan" anlamına gelir. Kripto paraların (Bitcoin, altcoin vb.) saklaması için kullanılmaktadır. Bu kartları kullanmanın bir amacı, tüm varlıkları borsalarda tutma riskini düşürmektir. Borsalar merkeziyetsiz değildir ve sadece bir internet sitesidir. Bu nedenle, kişisel cüzdan kullanmak artık daha önemlidir. Airdroplara katılmak için kişisel cüzdanların olması gerekir ve en yaygın cüzdan Ethereum (ETH)'dir. Kişisel Eth cüzdanlar ERC20'yi desteklemelidir. Aksi takdirde kaybolur ve elde edilebilir değildir. (Airdroplara katılırken borsa cüzdanları veya blokzincir cüzdanları kullanılmamalıdır.) ERC20, Ethereum'un kök cüzdanı olarak hizmet verir ve Ethereum tabanlı tüm coinleri destekler. Şekil 3.4, Ethereum cüzdanına ait temsili bir görüntü verilmiştir.

Ethereum da bitcoin gibi Blockchain mantığında üretime geçmiş ve ayrı bir yazılım mantığı sayesinde merkezsiz platformlar oluşturmuştur. Myetherwallet sitesine giriş yaptıktan sonra şifre belirlenir. Bu şifreyi girdikten sonra private key denen şifre verilmektedir. Şifre kaydedildikten sonra nasıl bir cüzdan istenildiği sorulmaktadır. Private key ile giriş yapılan bir cüzdan seçilmesi gerekmektedir. Bu şifre ile tüm sitelerden yatırımlara ulaşılmış olur. Ethereum cüzdanı girdikten sonra adres verilmektedir. Privete key sayesinde alınanlar ya da gönderilenler site üzerine kaydolmaz böylelikle tehlikeyi en aza indirir. Güvenlik

Şekil 3.4. Ethereum Cüzdanı

sebebi ile ethereum cüzdan sürekli oturum sonlandırır. Bu güvenlik için oldukça başarılı bir durumdur.

3.2.5. Truffle

Tim Coulter, 2015 yılında Ethereum ve ConsenSys ile blokzincir uygulamaları oluşturmaya çalışırken Truffle'ı oluşturmuştur. Kendi yaşamını kolaylaştırmak ve ilerlemesine yardımcı olmak için bir dizi senaryo tasarlamaya başlamıştır. Bunun da bir sonucu olarak Truffle Suite ortaya çıkmıştır.

Geliştiriciler, Truffle adlı bir araç paketini kullanarak herhangi bir blokzincir üzerinde sürdürülebilir profesyonel uygulamalar oluşturabilmektedirler. Geliştiricilerin Ethereum Blokzincirinin anlayabileceği şekilde akıllı kontratlar ve uygulamalar inşa etmesine izin veren bir ortam Ethereum Sanal Makinesi olarak düşünülebilir.

Ethereum için akıllı sözleşmeler genellikle Solidity programlama dili kullanılarak geliştirilir. Böyle bir geliştirme görevi için gereken tek şey bir metin editörü ve bir Solidity derleyicisidir. Bununla birlikte, akıllı sözleşmeler diğer programlardan farklıdır, çünkü blok zincirlerine yerleştirilmeleri amaçlanmıştır. Akıllı sözleşmeleri verimli bir şekilde test etmek ve bunları bir Ethereum ağına yerleştirmek, ek araç desteği olmadan şaşırtıcı derecede zor olabilmektedir. Ethereum Truffle framework'ü, akıllı sözleşmelerin test edilmesini ve akıllı sözleşmelerin bir Ethereum ağına yerleştirilmesini çok daha basit bir süreç haline getirmek için tasarlanmıştır.

Basit sözleşmeler, Remix IDE gibi tarayıcı tabanlı bir geliştirme ortamı kullanılarak test edilip dağıtılabilse de, bu birden çok sözleşmeden oluşan ve harici kitaplıklara bağımlılıkları olan daha büyük projeler için külfetli hale gelir. Böyle bir senaryoda, kod derlemeyi, test etmeyi ve devreye almayı entegre eden ve ayrıca iş akışlarının otomatikleştirilmesine izin veren bir geliştirme çerçevesi kullanılmalıdır. Truffle Suite, birlikte böyle bir geliştirme çerçevesi sağlayan üç bileşenden oluşur:

Ethereum Truffle Framework, derleme, test ve dağıtımı entegre eden gerçek geliştirme araç zinciridir.

Ganache, gerçek test ağları kurmaya veya uzak bir ağa konuşlandırmaya gerek kalmadan blockchain ağlarını ve canlı test sözleşmelerini simüle etmek için kullanılabilen, grafiksel bir kullanıcı arayüzüne sahip yerel olarak konuşlandırılmış bir blokzincir simülatörüdür.

Drizzle, geliştirilen akıllı sözleşmelere bağlanabilen web uygulamaları oluşturmak için bileşenler sağlamayı amaçlayan bir ön uç kitaplıkları koleksiyonudur. Ethereum Truffle, bu konfigürasyonda, dağıtım betiklerinde ve testlerde JavaScript'e dayanmaktadır. Bununla birlikte, testler doğrudan Solidity'de de yazılabilir.

Ethereum Truffle Yükleme

Ethereum Truffle Framework'ü kurmak çok kolaydır. Çoğu JavaScript tabanlı araçta olduğu gibi, npm paket yöneticisi kullanılır:

➛ npm install -g truffle

Framework kurulduktan sonra, Ganache blok zinciri simülatörünü indirip kurmak da mantıklıdır. Bu, grafik kullanıcı ara yüzüne sahip bir uygulama olduğu için, resmi Ganache web sitesi, en popüler platformlar için yükleyiciler sağlar.

Avantajları:

✓ Uygulamaları geliştirmek, entegre etmek ve üretmek için bir ortam sağlar.
✓ Akıllı kontratların testini otomatikleştirir.
✓ Geliştirme, entegrasyon ve üretim ortamları tekrar kullanılabilir.
✓ Geliştiricilerin kendi dillerinde kodlama yapabilmelerini sağlar.
✓ Genel ve özel ağlar için ağ yönetimi vardır.

3.2.6. EthereumJs – TestRpc

testrpc, test ve geliştirme için Node.js tabanlı bir Ethereum istemcisidir. Tam müşteri davranışını simüle etmek ve Ethereum uygulamaları geliştirmeyi çok daha hızlı hale getirmek için ethereumjs kullanır. Ayrıca, tüm popüler RPC işlevlerini ve özelliklerini (etkinlikler gibi) içerir ve geliştirmeyi devamlı hale getirmek için belirleyici olarak çalıştırılabilir.

testrpc, Javascript ile yazılır ve npm aracılığıyla bir Node paketi olarak dağıtılır.

- npm install -g ethereumjs-testrpc

komutuyla yüklenebilmektedir.

- $ testrpc <options>

testrpc, bir çok kullanım seçeneği ile ön plana çıkmaktadır.

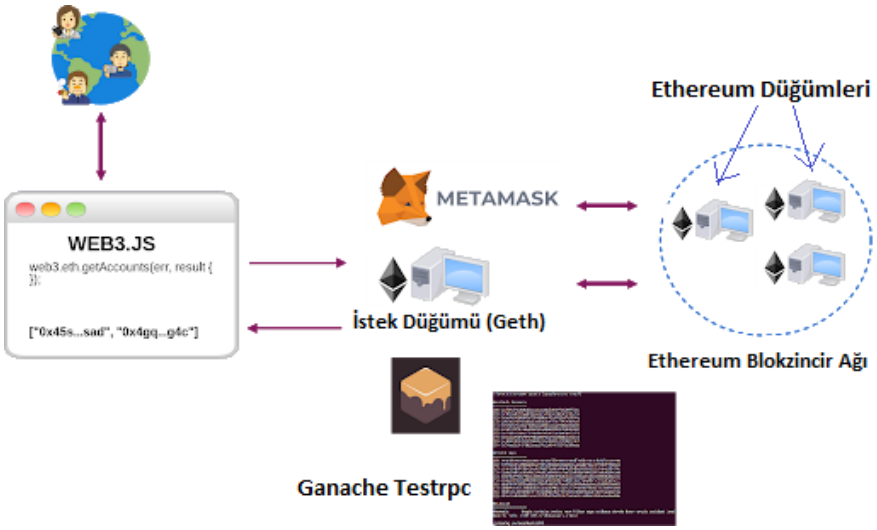
- ***-a veya –accounts:*** Başlangıçta oluşturulacak hesapların sayısını belirtir.
- ***-b veya –blocktime:***Otomatik madencilik için saniye cinsinden blok süresini belirtir. Varsayılan 0'dır ve otomatik madencilik yoktur.
- ***-d veya –deterministic:***Önceden tanımlanmış bir anımsatıcıya dayalı deterministik adresler oluşturur.
- ***-n veya –secure:***Varsayılan olarak mevcut hesapları kilitler (üçüncü taraf işlem imzalama için iyidir)
- ***-m veya –mnemonic:***İlk adresleri oluşturmak için belirli bir HD cüzdan anımsatıcı kullanır.
- ***-p veya –port:*** Dinlenecek bağlantı noktası numarasıdır. 8545 varsayılan değerdir.
- ***-h veya –hostname:***Dinlenecek ana bilgisayar adıdır. Düğümün server.listen () varsayılan değeridir.
- ***-s veya –seed:***Kullanılacak HD cüzdan anımsatıcısını oluşturmak için rastgele verileri kullanır.
- ***-g or –gasPrice:*** Özel bir Gas Fiyatı kullanır (varsayılan olarak 20000000000)
- ***-l veya –gasLimit:***Özel bir Gas Sınırı kullanır (varsayılan olarak 90000)

- ***-f veya –fork:***// localhost: 8545.
- ***-i veya –networkId:*** TestRPC'nin kendisini tanımlamak için kullanacağı ağ kimliğini belirtir (varsayılan olarak geçerli saat veya yapılandırılmışsa çatallı blok zincirinin ağ kimliği)
- ***–db:***Zincir veritabanını kaydetmek için bir dizine giden yolu belirtir. Bir veritabanı zaten mevcutsa, TestRPC yeni bir tane oluşturmak yerine bu zincir başlatılır.
- ***–debug:***Hata ayıklama
- ***–mem:*** Çıktı TestRPC bellek kullanım istatistikleridir. Bu, normal çıktının yerini alır.

3.2.7. Web3 Provider

web3.js, HTTP, IPC veya WebSocket kullanarak yerel veya uzak bir ethereum düğümü ile etkileşim kurmaya izin veren bir kitaplıklar koleksiyonudur. Tasarım gereği web3, bir Ethereum blok zincirinde sözleşme iletişimini yönetme ve işlem yapma yetenekleriyle çalışan bir blokzinciri istemcisi veya hafif düğüm olan bir web3 sağlayıcısı arar. Bu sağlayıcı, tercih edilen eklentiye veya cihaza (MetaMask, Cipher, tam bir go-ethereum düğümü vb.) bağlı olarak değişir.

Geliştirme amacıyla web3, test için hangisi seçilirse seçilsin, Ganache veya Truffle Develop blokzinciri ile iletişim kuracaktır. Dolayısıyla, dapp'i halka açık bir test ağında (Ropsten, Rinkeby, vb.) test etmek ve ardından bir ana ağ başlatmak söz konusu olduğunda, web3 sağlayıcının sözleşmelere erişmek için aynı ağı çalıştırması gerekir. Metamask, Ganache ve Truffle Develop gibi yerel olarak çalışan EVM blok zincirlerine bağlanmayı kolaylaştırır. Şekil 3.5, Web3.js ile Akıllı Sözleşme etkileşimini göstermektedir.



Şekil 3.5. Web3.js Akıllı Sözleşme Etkileşimi

3.2.8. Remix Ide

IDE, entegre geliştirme ortamı anlamına gelir ve programcıların kod yazma, derleme, çalıştırma ve hata ayıklama gibi yazılım geliştirmeyle ilgili farklı görevleri yürütmelerine yardımcı olmak için tasarlanmış bir dizi araca sahip bir uygulamadır. Remix, solidity kodunu yazmak, derlemek ve hata ayıklamak için kullanılan bir solidity IDE'dir. solidity, akıllı sözleşmeler yazmak için üst düzey, sözleşme odaklı bir programlama dilidir. c++, python ve javascript gibi popüler dillerden etkilenmiştir.

Remix IDE, açık kaynaklı bir web ve masaüstü uygulamasıdır. Hızlı bir geliştirme döngüsünü teşvik eder ve sezgisel GUI'lere sahip zengin bir eklenti setine sahiptir. Remix, ethereum öğrenmek ve öğretmek için bir oyun alanı olmanın yanı sıra, sözleşme geliştirme yolculuğunun tamamı için kullanılır.

Remix IDE, bir eklenti mimarisi kullanan geliştirme araçları için bir platform olan remix projesinin bir parçasıdır. remix plugin engine, remix libs ve remix-IDE dahil olmak üzere alt projeleri kapsar. remix IDE, solidity sözleşmelerini doğrudan tarayıcıdan yazmaya yardımcı olan güçlü bir açık kaynaklı araçtır.

Javascript ile yazılmıştır ve hem tarayıcıda hem de tarayıcıda kullanımı destekler. Ancak yerel olarak ve bir masaüstü sürümünde çalışır. Remix IDE, akıllı sözleşmelerin test edilmesi, hata ayıklanması ve devreye alınması ve çok daha fazlası için modüllere sahiptir.

Remix-IDE, ethereum adresinde [**IEEEexample:84**] mevcuttur ve bu belgelerde daha fazla bilgi bulunabilir.

3.3. AKILLI SÖZLEŞMELER İÇİN ÜCRETSİZ TEST AĞLARI

3.3.1. Ethereum Test Ağları

3.3.1.1. *Ropsten Test Ağı*

Ropsten Ethereum ("Ethereum Testnet" olarak da bilinir), ana Ethereum ağı olan Mainnet'te konuşlandırmadan önce blok zinciri geliştirme testine izin veren bir Ethereum test ağıdır.

Ropsten, ilk Ethereum Testnet olan Morden'in halefidir. Kasım 2016'dan beri yayındadır. Ancak Şubat 2017'den Mart 2017'ye kadar saldırı altındaydı. Ağ Geth, Parity, Nethermind ve Hyperledger Besu tarafından desteklenmektedir. Başlangıçta çok stabil olmadığı ve oldukça yavaş olduğu bilinmekle birlikte, hız açısından daha iyi durumda olduğu da görünmekteydi.

*Ethereum ağının merakla beklenen Ethereum Improvement Proposal (EIP) 1559 iyileştirmesini içeren London güncellemesi, Ropsten testnet üzerine eklendi.

3.3.1.2. *Rinkby Test Ağı*

Rinkeby, ana Ethereum ağı olan Mainnet'te konuşlandırmadan önce blok zinciri geliştirme testine izin veren bir Ethereum test ağıdır. Yetki Kanıtı(PoA) test ağı, Nisan 2017'de kuruldu. Clique PoA fikir birliği protokolünü kullanır ve Geth geliştirici ekibi tarafından sürdürülür. Ağ Geth, Nethermind ve Hyperledger Besu tarafından desteklenmektedir.

3.3.1.3. *Kovan Test Ağı*

Kovan, ana Ethereum ağı olan Mainnet'te dağıtımdan önce blok zinciri geliştirme testine izin veren bir Ethereum test ağıdır. Yetki Kanıtı(PoA) test ağı Mart 2017'de kuruldu. AuRa (otorite turu) konsensüs mekanizmasını kullanır ve Parity tarafından desteklenir.

3.3.1.4. *Goerli Test Ağı*

Goerli, ana Ethereum ağı olan Mainnet'te dağıtımdan önce blok zinciri geliştirme testine izin veren bir Ethereum test ağıdır.

Yetki Kanıtı(PoA) test ağı Mart 2019'da kuruldu. Clique fikir birliği mekanizmasını kullanır ve ilk olarak Chainsafe ve Afri Schoedon tarafından önerilmiştir. Ağ, Goerli topluluğu tarafından yönetilir ve Geth, Parity, Nethermind ve Hyperledger Besu tarafından desteklenir.

Goerli'nin artan ağ boyutu, düğüm sayısı ve istemci çeşitliliği nedeniyle Mainnet'i test etmek için Kovan yerine Ethereum Goerli tercih edilebilir.

3.3.1.5. *Casper Test Ağı*

Vitalik Buterin tarafından düzenlenen bir toplantıda Casper ağının test için hazır olduğu açıklandı. Toplantıda Buterin ayrıca, istemciler arasında test sürecinde daha fazla güvenlik sağlanacağını da belirtti. Birçok geliştirici, özellikle birden fazla Ethereum istemcisiyle çalışan programların arkasındaki geliştiricileri bu yeniliğin rahatlatacağı görüşündüler.

Şu anda kullanıma sunulan tek istemciler arası test ağı olan Ropsten, emek kanıtı protokolü olan Proof-of-Work'i kullanmaktadır. Sadece madencilerin olduğu bir ağda Proof-of-Work iyi çalışmaktadır. Ropsten bir test ağı ve madenciler neredeyse yok denecek kadar az olduğu için saldırılara açık bir ağ olarak kabul edilebilmektedir. Rinkeby ve Coven test ağları, bu ataklardan kaçınmak için yetki kanıtı (PoA) bazlı bir protokol kullanmaktadır. Bununla birlikte, çoğu geliştirici bu ağları kullanımı zor olduğu için tercih etmemektedirler.

Casper test ağı şu anda yaygın olarak kullanılmamaktadır, bu nedenle ana Ethereum ağına entegre edilmesi oldukça zordur.

Casper ağının Python programlama diline uyarlanması da düşünülmektedir. Bu sebeple, ağ üzerinde şu anda küçük ölçekli testler yürütülmektedir.

3.3.2. Avalanche Test Ağları

3.3.2.1. *Fuji (P-Chain, X-Chain ve C-Chain) Test Ağları*

AVA platformu, 2019 Nisan ayında testnetini kullanıma açmıştır. İnsanlar dünyanın her yerinden testnet üzerinde çalışmaya ve topluluğa katkıda bulunmaya başladı. Avalanche'ın Birincil Ağı, üç zinciri olan bir alt ağdır: P-Chain, X-Chain ve C-Chain. C-Chain, Avalanche'ın Snowman konsensüs protokolü tarafından desteklenen Ethereum Sanal Makinesinin bir örneğidir. C-Chain RPC, Ethereum standardı RPC çağrılarını kullanarak tipik bir Ethereum istemcisinin yapabileceği her şeyi yapabilir.

Avalanche C-chain üzerinde Ethereum yerine smart contract geliştirmenin iki avantajı vardır: bir saniyede işlem sonlandırma ve daha düşük işlem ücretleri. Ek olarak, işlem ücretleri aynıdır.

3.3.2.2. *Islander Test Ağı*

Islander Testnet, Avalanche ağında başlatılmıştır. Katılımcıların yaklaşmakta olan airdrop'a dahil olmaları için girişi açılmıştır.

3.3.3. Cardano Test Ağları

3.3.3.1. *Alonzo Test Ağı*

Cardano geliştiricileri, akıllı sözleşmelere odaklanan bir test ağı kurmayı planlamaktadırlar. Alonzo ağı, kullanıcıların Cardano'da merkezi olmayan uygulamaları dağıtmasına izin vermektedir.

Mart ayının başlarında Cardano'nun Mary adlı önceki yükseltmesi, kullanıcılara özel tokenler oluşturma fırsatı vererek blokzincirini Ethereum'a yakınlaştırmıştır. Bu, Cardano'yu potansiyel olarak birçok farklı kripto parayı destekleyebilen çok varlıklı bir ağa dönüştürmektedir.

Cardano, Ethereum'un aksine token işlemleri için akıllı sözleşmelere güvenmeyen bir ağdır. Bu, Cardano'ya kıyasla Ethereum'da hızla yükselen transfer ücretlerinin çok daha düşük olabileceği anlamına gelir.

3.3.4. Solana Test Ağları

3.3.4.1. *Devnet*

Devnet, kullanıcı, token sahibi, uygulama geliştiricisi veya doğrulayıcı olarak Solana'yı bir test sürüşüne çıkarmak isteyen herkes için bir oyun alanı olarak hizmet eder.

Geliştirme ağı veya devnet, test ağına benzer şekilde ana ağdan ayrı çalışır. Her blokzinciri protokolü geliştirici tarafından veya her test ağıda kullanılmasa da, bazı protokoller amaçlarına göre farklı ortamlar kullanmaktadırlar. Solana, devnet'in uygulama geliştiricileri, token sahipleri, blokzinciri kullanıcıları veya ağ doğrulayıcıları için bir "oyun alanı" olduğunu iddia etmektedir. Bununla birlikte, Solana blokzinciri test ağı, son sürümlerin ağ performansına, kararlılığına ve ağ doğrulayıcılarının davranışına odaklanan bir stres testine izin vermektedir.

3.4. AKILLI SÖZLEŞMELERİN KODLANMASI İÇİN GELİŞTİRİLMİŞ PROGRAMLAMA DİLLERİ

Bu bölümde, çeşitli blok zinciri platformlarında akıllı sözleşmeleri kodlamak için kullanılan programlama dilleri tanıtılmaktadır.

3.4.1. Solidity

Solidity, akıllı sözleşme teknolojisinin öncüsü olan Ethereum blok zincirinde akıllı sözleşmeler oluşturmak için kullanılan programlama dilidir. Solidity'nin söz dizimi JavaScript'e benzer. Bu da onu daha önce web geliştirme deneyimi olan geliştiriciler için nispeten erişilebilir kılmaktadır. Kalıtım, kütüphaneler ve kullanıcı tanımlı tipler gibi özellikleri destekleyerek karmaşık ve çok yönlü sözleşmelerin oluşturulmasını sağlar. Bununla birlikte, esnekliği nedeniyle Solidity, güvenlik açıklarından payına düşeni almış ve Ethereum topluluğunu güvenlik uygulamalarını geliştirmeye odaklanmaya sevk etmiştir.

3.4.2. Vyper

Vyper, Ethereum akıllı sözleşme geliştirme için Solidity'ye bir alternatif olarak ortaya çıkmıştır. Python benzeri bir sözdizimi kullanarak ve potansiyel güvenlik açıklarına yol açabilecek özellikleri ortadan kaldırarak Solidity ile ilişkili bazı güvenlik endişelerini gidermektedir. Vyper'ın tasarım felsefesi, aşırı esneklik yerine güvenlik ve basitliğe öncelik vermektir. Bu da onu akıllı sözleşme kodlamasında yaygın tuzaklardan kaçınmak isteyen geliştiriciler için uygun bir seçenek haline getirmiştir.

3.4.3. Simplicity

Simplicity, Tezos blok zincirinde akıllı sözleşmeler oluşturmak için özel olarak tasarlanmış bir programlama dilidir. Tezos, güvenliği ve resmi doğrulamayı geliştirmeyi amaçlamaktadır ve Simplicity bu hedefle uyumludur. Sözdizimi, potansiyel güvenlik açıklarının yüzey alanını azaltmak için kasıtlı olarak minimalisttir. Açık ve özlü bir kodlama stilini teşvik eden Simplicity, geliştiricilerin sözleşme davranışı hakkında akıl yürütmesini ve doğruluğu doğrulamasını kolaylaştırır.

3.4.4. Liquidity

Tezos blok zinciri için bir başka dil olan Liquidity, basitlik ve güvenliğe odaklanır. OCaml ve JavaScript'ten öğeler ödünç alır ve bu dillere aşina olan geliştiriciler için nispeten erişilebilir hale getirir. Liquidity'nin tasarımı, güvenli kodlama uygulamalarını teşvik eden ve yaygın programlama hatalarını önleyen ve sonuçta daha güvenilir akıllı sözleşmelere yol açan özellikler içerir.

3.4.5. Cadence

Flow blokzinciri, akıllı sözleşmeler için özel olarak tasarlanmış kaynak odaklı bir programlama dili olan Cadence'ı tanıtmıştır. Flow ölçeklenebilirlik ve kullanılabilirliği vurgulamaktadır ve Cadence bu ilkelerle uyumludur. Cadence, kaynak yönetimini zorunlu kılarak ve değiştirilebilir ve değiştirilemez veriler arasında net bir ayrım sağlayarak, diğer dillerde yaygın olan hataları ve güvenlik açıklarını önlemeyi amaçlamaktadır. Özellikle eş zamanlı yürütme gerektiren uygulamalar için uygundur.

3.4.6. Move

Move, Diem (eski adıyla Libra) blok zincirinin programlama dilidir. Diem'in öncelikli odak noktası istikrarlı ve erişilebilir bir dijital para ekosistemi oluşturmaktır. Move bu hedefe ulaşmada çok önemli bir rol oynamaktadır. Move'un benzersiz yaklaşımı, hataları ve güvenlik açıklarını önlemeye yardımcı olan kaynak odaklı programlamayı içerir. Güvenli, esnek ve karmaşık finansal işlemleri gerçekleştirebilecek şekilde tasarlanmıştır.

3.4.7. Rholang

Rholang, RChain blok zincirinde akıllı sözleşmeler yazmak için kullanılan dildir. Rholang'ın tasarımı, eş zamanlı hesaplama için resmi bir model olan pi-calculus'tan etkilenmiştir. Yüksek düzeyde eşzamanlılık gerektiren uygulamalar için özel olarak tasarlanmıştır. Bu da onu merkezi olmayan uygulamalar ve birden fazla etkileşimli bileşene sahip sistemler için çok uygun hale getirmiştir.

3.4.8. Marlowe

Marlowe, Cardano blok zincirinde finansal sözleşmeler oluşturmak için alana özgü bir dildir. Odak noktası, akıllı sözleşmeleri hem programcılar hem de finans uzmanları tarafından anlaşılabilir hale getirmektir. Marlowe'un tasarımı, yaygın finansal senaryolar için üst düzey soyutlamalar içerir ve kullanıcıların derin teknik uzmanlık olmadan karmaşık finansal anlaşmaları ifade etmelerine olanak tanır.

3.4.9. Scilla

Scilla, Zilliqa blok zincirindeki akıllı sözleşmeler için programlama dili olarak hizmet vermektedir. Zilliqa güvenlik ve ölçeklenebilirliğe öncelik verir ve Scilla güçlü güvenlik garantileri sağlayarak ve resmi doğrulamayı kolaylaştırarak bu öncelikleri yansıtır. Scilla'nın tasarımı, geliştiricilerin güvenli ve güvenilir sözleşmeler yazmasına yardımcı olan ve güvenlik açığı riskini en aza indiren özellikler içerir.

3.4.10. Plutus

Plutus, Cardano blok zincirinde komut dosyası yazmak için kullanılır. Hem akıllı sözleşmeler için üst düzey bir komut dosyası dilini (Haskell tabanlı) hem de zincir üzerinde doğrulama için alt düzey bir komut dosyası dilini kapsar. Plutus, geliştiricilere güvenlik ve doğruluğu sağlarken sofistike akıllı sözleşmeler oluşturmak için araçlar sağlamayı amaçlamaktadır.

Hızla gelişen blok zinciri ortamında, akıllı sözleşmeler için programlama dilleri güvenlik, ölçeklenebilirlik ve kullanılabilirlikle ilgili zorlukları ele almak için gelişmeye devam etmektedir. Her dil, kendi blok zinciri platformunun hedeflerine ve tasarım ilkelerine göre uyarlanarak geliştiricilerin farklı karmaşıklık ve işlevsellik derecelerine sahip merkezi olmayan uygulamalar oluşturmasına olanak tanır.

3.5. TRUFFLE, SOLIDITY VE JAVASCRIPT İLE BASİT AKILLI SÖZLEŞME OLUŞTURMA

Akıllı bir sözleşme oluşturmak için çeşitli ortamlar ve programlama dilleri bulunmaktadır. Bunlardan en stabil çalışan ve en çok kullanılanı Ethereum üzerinde koşan akıllı sözleşmelerdir. Ethereum akıllı sözleşmeleri Solidity programlama dili kullanılarak geliştirilirler. En hızlı büyüyen blok zincir programa dillerinden olan Solidity, Ethereum'un çekirdek takımı tarafından akıllı sözleşmeler için tasarlanmıştır. Solidity, deneyimli blokzincir geliştiricileri tarafından karşılaşılan en özel sorunlara bile çözümler sunmaktadır.

Solidity dili;

Avantajları

- ✺ Statik olarak tanımlanmıştır.
- ✺ Çok basit bir şekilde öğrenilebilir.
- ✺ Çok hızlı bir şekilde popüler hale gelmektedir.

Dezavantajları

- ✖ Çok yenidir.

- ✖ Topluluğu henüz çok küçük olduğu için yeterli destek yoktur.
- ✖ Evrensel değildir - Ethereum ekosisteminde daha iyi çalışır.
- ✖ Son derece savunmasızdır.
- ✖ Kod her zaman ayrıntılı bir şekilde test edilmelidir çünkü kritik hata yapmak çok kolaydır.

Solidity dilinin kolay ve eğlenceli bir şekilde anlatımı için geliştirilmiş olan cryptozombies sitesi [**IEEEexample:85**], bir yandan oyun geliştirme deneyimi sağlarken bir yandan da solidity dilinin ayrıntılı biçimde uygulanmasını sağlamaktadır.

Akıllı sözleşmeler için bir diğer gerekli uygulama Nodejs'tir. Nodejs internet üzerinde yer alan javascript ile yazılmış çeşitli projelere npm anahtar sözcüğü ile ulaşabilen bir motordur. V8, Google tarafından geliştirilen bir motordur. Chrome web tarayıcılarında da kullanılan C, C++ ve javascript dilleri kullanılarak kodlanmıştır.

Nodejs motorunu indirdikten sonra kullanılacak olan kütüphanelerin yüklenmesi gerekmektedir. Sırasıyla aşağıdaki komutları Nodejs terminali aracılığıyla girerek gerekli uygulamalar kurulmalıdır.

- ➛ sudo npm i –g npm to update
- ➛ sudo npm install -g node-gyp
- ➛ sudo npm install -g ethereumjs-testrpc
- ➛ sudo npm install -g truffle

Truffle, diğer tüm özelliklerinin yanında en önemli özelliği, içerisinde hazır akıllı sözleşme projelerinin bulunmasıdır. Bu çalışmada, truffle'ın akıllı sözleşme projeleri kullanılacaktır. Truffle Proje Çekme;

Bilgisayar üzerine istenilen isimde bir çalışma alanı oluşturulsun (Örn. Blockchain). Nodejs terminali üzerinde "cd" anahtar sözcüğü ile klasör içine girilmelidir.

- ➛ truffle init

Bu komut, internet üzerinde yazılmış herhangi bir akıllı sözleşme projesini modifiye edilmek üzere, truffle aracılığıyla belirlenen klasör içerisine indirmektedir.

```
pragma solidity >=0.4.22 <0.8.0;

contract NumberStore {

    mapping(string => int) personNumbers;

    function addPersonNumber(string memory name, int personNumber) public {
        if(personNumbers[name]>0){revert();}
        else{personNumbers[name]=personNumber;}
    }
    function getNumber(string memory name) public view returns (int){
        return personNumbers[name];
    }
}
```

Şekil 3.6. Örnek Basit bir Akıllı Sözleşme Kodu (NumberStore)

Bu proje klasörü içerisinde, Şekil 3.6'da da belirtildiği gibi, build, contracts, migrations, test klasörleri ve proje ile ilgili tüm düzenlemeleri içeren bir javascript dosyası gelmektedir.

Bu klasörlerden contracts klasörü içerisinde yer alan migrations.sol dosyasının kodu Şekil 3.7'te gösterilmiştir. Pragma solidity komutu ile uygulamanın çalışabileceği sürüm aralıkları belirlenmektedir.

Sözleşmenin ismi contract anahtar sözcüğüyle belirlenip içeriği yazılmaktadır. Şekil 3.7'de görüldüğü gibi, Migrations isimli sözleşme oluşturulmuştur. Sözleşme owner isimli bir adres ve last_completed_migration değişken isimli integer bir sayıdan oluşmaktadır.

```
pragma solidity >=0.4.22 <0.8.0;

contract Migrations {
    address public owner = msg.sender;
    uint public last_completed_migration;

    modifier restricted() {
        require(
            msg.sender == owner,
            "This function is restricted to the contract's owner"
        );
        _;
    }
    function setCompleted(uint completed) public restricted {
    last_completed_migration = completed;
    }
}
```

Şekil 3.7. Migrations.sol dosyası kod içeriği

Migration.sol dosyası kopyalanarak, bu kopyalanmış olan kod dosyası üzerinde istenilen akıllı sözleşme kodu oluşturulabilmektedir. Şekil 3.6'te belirtildiği gibi örnek olarak gösterilecek olan NumberStore.sol akıllı sözleşme dosyası oluşturulmuştur.

Akıllı sözleşme kodlanmadan önce, sözleşmeye dair tüm konfigürasyonların yapıldığı, truffle-config.js dosyasının düzenlemesinin yapılması gerekmektedir. Şekil 3.8'de örnek bir configürasyon dosyası görüntüsü verilmiştir. Şekilde görüldüğü gibi, kodun blok zincir ağında yayınlandığında gerçekleşecek olan ayarları yapılandırılmaktadır.

Kodun hangi port üzerinden yayınlanacağı, ağ üzerinde tanınmasını sağlayan id numarası, en fazla ödenebilecek Gas miktarı, her bir işlem için ödenecek olan Gas miktarı, işlemlerin gerçekleştirileceği adresler gibi birçok özellik belirlenebilmektedir.

```
networks: {
    advanced: {
        port: 8777,
        network_id: 1342,
        gas: 8500000,
        gasPrice: 20000000000,
        from: <address>,
        websockets: true
    }
    Useful for deploying to a public network.
    NB: It's important to wrap the provider as a function.
    ropsten:{
        .
        .
```

Şekil 3.8. Konfigürasyon Dosyası.

İnternet üzerinden çekilen bu proje ile beraber gelen bir başka dosya ise 1_initial_migration.js dosyasıdır. Şekil 3.9'da verilmiş olan kodun içeriğinde yer alan require() metodu ile değişkene aktarılan solidity kodları, deploy() metodu ile javascript koduna çevrilmeye hazır hale getirilmektedir.

```
const Migrations = artifacts.require("Migrations");
const NumberStore = artifacts.require("NumberStore");

module.exports = function (deployer) {
    deployer.deploy(Migrations);
    deployer.deploy(NumberStore);
};
```

Şekil 3.9. 1_initial_migration.js kodu

3.6. SOLIDITY İLE YAZILMIŞ AKILLI SÖZLEŞME KODUNUN JSON KODUNA DÖNÜŞTÜRÜLMESİ

Örnek olarak oluşturulmuş basit bir akıllı sözleşme kodu Şekil 3.10'de verilmiştir. Kodda yer alan 'pragma' anahtarı yazılan kodun sürüm aralıklarını belirlemek için kullanılmaktadır. 'mapping' anahtar sözcüğü, mantıksal değerler, tam sayılar, adresler ve yapılar gibi değer türlerini yapılandırmak için kullanılır. İki ana bölümden oluşur: bir _KeyType ve bir _ValueType; aşağıdaki sözdiziminde görünürler:

➝ mapping (_KeyType => _ValueType) mapName

_KeyType'ı, istenen bir değeri veya _ValueType'ı döndürmek için bir işlevden geçilecek anahtar olarak düşünülebilir. Varsayılan olarak, bir eşleme başlangıçta boştur, bu nedenle yeni bir _KeyType'ın önce bir _ValueType ile eşlenmesi gerekir.

Şekil 3.10'da örnek olarak, girilen string bir ifadenin karşılığı olan integer değerin döndürülmesi istenmiştir.

If bloğu içerisinde eklenen kişinin birden fazla eklenmemesi için koşullanmıştır. Eklenmişse 'revert' anahtar sözcüğü ile fonksiyondan çıkılması sağlanır. 'revert' anahtarı 'throw' anahtarının yerine solidity dilinde tercih edilmektedir.

- "memory" anahtar kelimesi, kodun nerede çalışacağının belirlenmesi gerektiğinden dolayı string ifadeler için yazılması zorunludur.

- getNumber fonksiyonunda yer alan “view” ifadesi herhangi bir ücret ödemeden test yapılabilmesi için gerekli bir anahtar kelimedir.

```
pragma solidity >=0.4.22 <0.8.0;

contract NumberStore {

    mapping(string => int) personNumbers;

    function addPersonNumber(string memory name, int personNumber) public {
        if(personNumbers[name]>0){revert();}
        else{personNumbers[name]=personNumber;}
    }
    function getNumber(string memory name) public view returns (int){
        return personNumbers[name];
    }
}
```

Şekil 3.10. Örnek Basit bir Akıllı Sözleşme Kodu (NumberStore)

Nodejs terminali aracılığıyla JSON koduna dönüşüm için yazılan kodun compile edilmesi gerekmektedir. Projede başlangıç düzenlemelerinin yer aldığı 1_initial_migrations.js isimli bir javascript kodu bulunması zorunu olmak ile birlikte, kişisel olarak oluşturulan akıllı sözleşme kodlarının deploy edildiği bir 2_deploy_contracts.js kodu da eklenmelidir. Şekil 3.11’de oluşturulmuş olan NumberStore akıllı sözleşme kodu deploy edilmek üzere javascript koduna eklenmiştir.

```
const NumberStore = artifacts.require("NumberStore");

module.exports = function (deployer) {
    deployer.deploy(NumberStore);
};
```

Şekil 3.11. Solidity Kodunun Deploy edilmesi

Nodejs terminal ekranında proje dosyası içerisindeyken aşağıda yer alan komut çalıştırılmalıdır.

```
➜ truffle compile mapName
```

Proje truffle kullanılarak derlendikten sonra, sol. Uzantılı solidity kod dosyaları birer JSON dosyasına dönüştürülmüş olur. JSON koduna dönüştürülen NumberStore örnek projesinin kod içeriği Şekil 3.12'de verilmiştir.

Solidity kodunun tüm özelliklerini içeren JSON kodunda "source" kısmı tamamen solidity de yazılmış olan kodları göstermektedir.


```
"metadata": "{\"compiler\":{\"version\":\"0.7.4+commit.3f05b770\"},\"language\":\"Solidity\",\"output\":{\"
"bytecode": "0x6080604052348015610010576000 80fd5b50336000808060001b815260200190815260200160
"deployedBytecode": "0x608060405234801561001057600080fd5b50600436106100b45760003560e01c0daab
"immutableReferences": {},
"generatedSources": [],
"deployedGeneratedSources": [],
"sourceMap": "84:6047:1:-:0;;;621:69;;;;;;;;;673:10;652:7;:12;660:3;652:12;;;;;;;;;;;;:18;;;:31;; ;;;;;;
"deployedSourceMap": "84:6047:1:-:0;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
"source": "pragma solidity ^0.7.0;\n\nimport \"./NumberStore.sol\";\n\n/**\n * The NumberStore.sol contract
"sourcePath": "/home/sniffnoy/truffle/ens/contracts/NumberStore.sol",
  "ast": {
    "absolutePath": "project:/contracts/NumberStore.sol",
    "exportedSymbols": {
```


Şekil 3.12. Solidity kod dosyalarından dönüştürülen JSON dosyaları

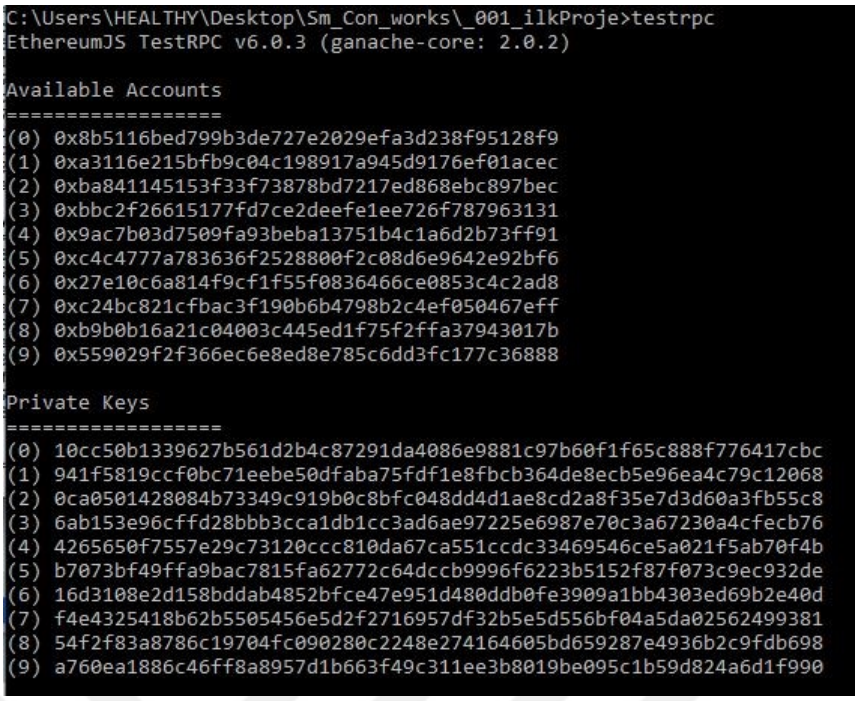
Bu dönüşümlerin amacı, kodu solidity programlama dili bağımlılığından kurtarmaktır. Javascript ve JSON kodlarına dönüşen bu sözleşmeler istenilen dilde kullanılabilir hale getirilmektedir. Bu aşamada kodun diğer dillere açılması için aşağıdaki komutların terminal üzerinden proje içerisinde çalıştırılması gerekmektedir.

- testrrpc
- truffle develop

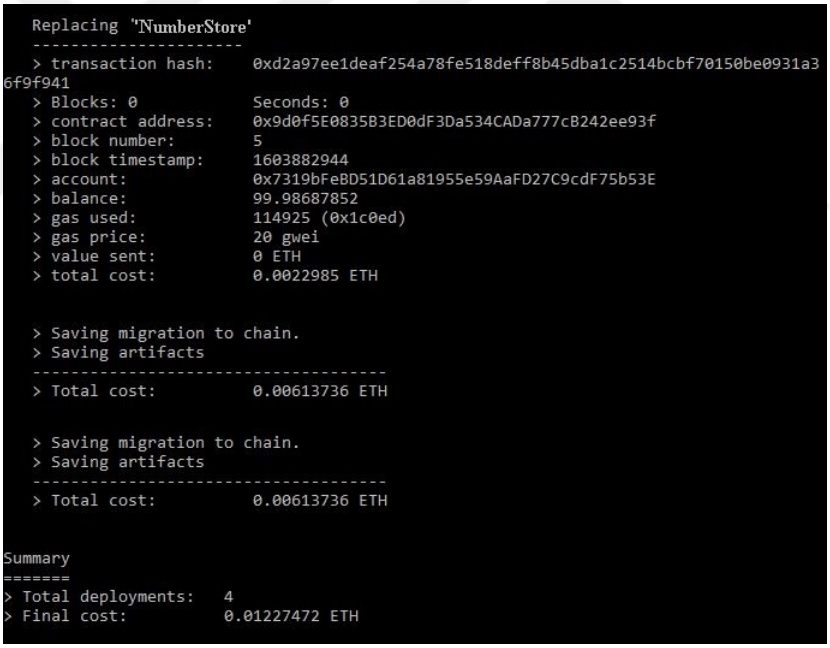
'testrpc' komutu ile kodu doğru çalışmasının sağlaması yapılabilmektedir. Şekil 3.13'te görüldüğü gibi, sistem tarafından yazılan kodun test edilmesi için ganache aracılığıyla 9 adet adres temin edilmiştir. Develop fonksiyonunun içerisine girdikten sonra aşağıdaki komut akıllı sözleşme ile ilgili tüm bilgilerin gösterilmesi ve uygulanması için gereklidir.

- migrate

migrate işlemi sonucu, eğer akıllı sözleşme kullanılacaksa, ödenecek olan fiyatlar (eth cinsinden) hesaplanarak sunulur. Şekil 3.14, örnek projenin gerçek blok zincir ağında yayınlanması durumunda ödenmesi gereken eth miktarlarını göstermektedir.



Şekil 3.13. Test için temin edilen adresler



Şekil 3.14. Test edilen proje için ödenmesi gereken eth miktarları

3.7. AKILLI SÖZLEŞME KODUNUN JAVASCRIPT İLE KULLANILMASI

Migrate komutu çalıştırılan ve gereken ödemeleri hesaplanan akıllı sözleşme kodu, javascript ile ulaşmaya hazır durumdadır. Bu aşamadan sonra javascript kodları kullanılarak istenilen veriye ulaşılıp, istenilen programlama dilinde kullanılabilmektedir. Nodejs

terminali üzerinde, proje içerisinde ve develop komutu üzerinde iken aşağıda yer alan javascript kodu ile akıllı sözleşme kullanılabilmektedir.

- truffle(develop)>var myMessage //bir değişken oluşturulmuştur.
- truffle(develop)>NumberStore.deployed().then(function(deployed) myMessage=deployed;); //Bu fonksiyon ile proje içerisinde yer alan ve erişim engeli bulunmayan tüm fonksiyon ve değişkenlere erişim sağlanmış olunmaktadır.

Belirlenen değişken kullanılarak Şekil 3.15'te blokzincire bir isim eklenmiştir. Bu işlem blokzincirde yer alan akıllı sözleşme kodu aracılığı ile ağa eklenmiştir ve geri alınamaz. Şekilde işlem sonucu gerçekleşen trasaction'lar gösterilmektedir.

- truffle(develop)>myMessage.addPersonNumber("Tunahan TİMUÇİN", 100)

```
truffle(develop)> myMessage.addPersonNumber("Tunahan TİMUÇİN", 100)
{
   tx: '0xb6102344be8cc5553e688981eed6bf2cc4417c25382c06d7c585ea9877ff85bc',
   receipt: {
      transactionHash: '0xb6102344be8cc5553e688981eed0xb6102344be8cc5553e688981eed6bf
      transactionIndex: 0,
      blockHash: '0x31d1a524aec0ca9f7058f358cf83c792d9be33e21a05b43ace2cc73d5f8e579a',
      blockNumber: 6,
      from: '0x7319bfebd51d61a81955e59aafd27c9cdf75b53e',
      to: '0x21fbf3fe02794490bb647c984e4a08adac9120f9',
      gasUsed: 43813,
      cumulativeGasUsed: 43813,
      contractAddress: null,
      logs: [],
      status: true,
      logBloom: '0x00000000000000000000000000000000000000000000000000000000000000(
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
      rawLogs: []
   },
   logs: []
}
```

Şekil 3.15. Akıllı Sözleşmeye javascript ile veri girişi

Örnek projede yer alan bir diğer fonksiyon girilen değerlerin görüntülenmesini sağlayan getNumber fonksiyonudur. Aşağıda yer alan kod parçası aracılığıyla bilgilere ulaşılabilmektedir. Mapping fonksiyonunda belirlendiği gibi "string" bir değer girilip "integer" olan karşılığı listelenmesi beklenmektedir.

- truffle(develop)>myMessage.getNumber.call("Tunahan TİMUÇİN")

İşlem sonucunda veritabanına string ifadenin karşılığı olarak girilmiş olan 100 değeri dönmüştür (Şekil 3.16).

```
truffle(develop)> myMessage.getNumber.call("Tunahan TİMUÇİN")
BN {
    negative: 0,
    words: [ 100, <empty item> ],
    length: 1,
    red: null
}
```

Şekil 3.16. Blokzincire eklenen bilgilerin görüntülenmesi

3.8. 3 ADET TEMEL SEVİYE AKILLI SÖZLEŞME UYGULAMASI

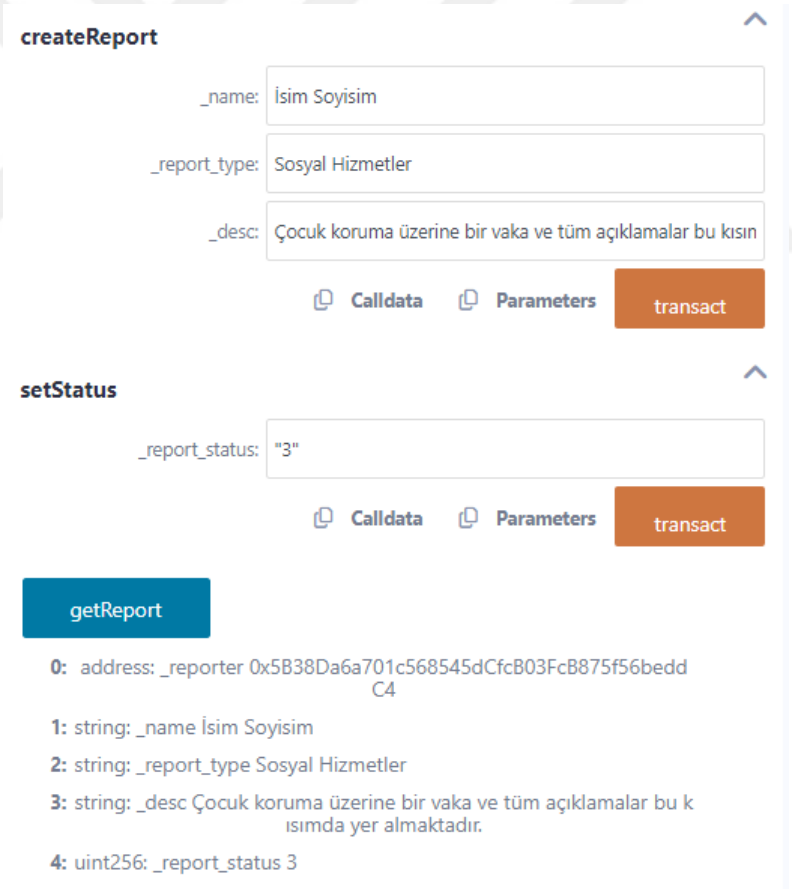
3.8.1. Uygulama 1: Merkeziyetsiz Dosya Raporlama Sistemi

Raporlamanın en temel amacı, tüm işlemlerin sonuçlarını izlemektir. Bu nedenle ele alınması istenen konu aslında çok daha önemlidir. Raporlar, yapılan çalışmaların verimliliği, başarısı ve başarısızlığı gibi birçok soruya yanıt verebilir. Raporlama işlemleri, hukuk bürosunun en temel çarklarından biri olduğu kabul edilmekle birlikte, akıllı sözleşmelerle birlikte sosyal hizmetler, muhasebe gibi tüm alanlara yayılması beklenmektedir. Dosyalar raporlanırken, her rapor şablonunda ana başlık, alt başlık, kolon başlıkları, satırlar gibi bölümlerden oluşur. Bu bölümlerde yer alacak bilgiler doğrudan yazılarak ya da standart ve tanımlı alanlar listelenerek seçilebilir. Raporun başlık bölümünde tanımlanan raporun adı, firma numarası, firma adı, sayfa numarası vb. rapor başlık bilgileri yer alır. Raporda yer alan bir diğer alan ise eklenecek alanın türü ve içeriğidir. Başlığı girilen konuyla alakalı olan içerik description (tanım) alanına eklenebilir. Şekil 3.17'de kodu verilen akıllı sözleşme, başlangıç olarak genel hatlarıyla oluşturulmuş, belirlenecek alana göre yeni özellikler eklenerek geliştirilmeye uygundur. Sözleşmede yer alan raporlamanın; bir ekleyicisi (reporter), Rapor ismi (name), Rapor türü (report_type), Raporun tanımı veya içeriği (desc) ve rapor statüsü veya rapor numarası (report_status) bulunmaktadır.

Şekil 3.18'de görüldüğü gibi ethereum'un web tarayıcıları üzerinde geliştirilen REMIX IDE üzerinde derlenip deploy edilen akıllı sözleşme kodu Şekil 3.18'de görülen şekliyle sonuçlar gösterilmiştir. Raporun oluşturulması için, bir rapor ismi, bir rapor tipi ve raporun içeriğinin girilip createReport tuşuna basılması gerekmektedir. setStatus kısmında raporun statüsünü veya dosya numarasının güncellemesi yapılabilmektedir. Eklenilen raporların görüntülemesi getReport ile yapılabilmektedir.

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.4.22 <0.9.0;

contract ReportingFileSystem{
//Sözleşme ismi

    address reporter;
    string name;
    string report_type;
    string desc;
    uint report_status;

    event createReportEvent(address indexed _reporter,string _name, string _report_type, string _desc);
    // Rapor Dosyasının tanımı

    //Raporun oluşturulduğu fonksiyon. Report status, sıra no olarak düşünülebilir.
    function createReport(string memory _name, string memory _report_type, string memory _desc) public {
        reporter = msg.sender;
        name = _name;
        report_type = _report_type;
        desc = _desc;
        report_status = 1;
        emit createReportEvent(reporter,name,report_type,desc);
    }
```

Şekil 3.17. Merkeziyetsiz Dosya Raporlama Sistemi



Şekil 3.18. Remix IDE üzerinde Uygulama Arayüzü

3.8.2. Uygulama 2: Yapılacaklar Listesi (ToDo)

Günümüzde, insanlar bilimle daha çok ilgilenmeye ve felsefe gibi düşünce alanlarında daha fazla ilerlemeye başlamıştır. Bunun sonucu olarak, yapacaklarının listesini yazmaya ve bunları yazılı olarak yapmaya başlamışlardır. İş yerlerinin motivasyonunu ve verimliliğini artırmak adına ödüllendirme sistemiyle bu listelere uyma zorunluluğu getirmesi ile liste tutmanın önemi daha da artmıştır.

Her bir bireyin bir dizi hedefi vardır. Günlük işler arasında bile çok sayıda iş bulunmaktadır. Hayatta ilerlemek için bu uğraşlardan en az zararla kurtulmak gerekir. Günlük sorumluluklar ve iş sorumlulukları arasında sürekli olarak geçişler yapmak oldukça yorucu ve sıkıcı bir durumdur. Bu noktada listeler kişilere yardımcı olabilir. Listeler verimliliği artırır, ancak listenin nasıl kullanıldığı önemlidir. Yapılması gerekenlerin "büyük resmi" verdiği için yapılacaklar listesi denenmesi gereken bir yöntemdir. Ayrıca kontrol listeleri verimliliği artırır. Bir kontrol listesi düzenli olarak tutmak, tamamlanması gerekenleri hatırlamak için gereken zihinsel yükün azaltılmasına yardımcı olabilir.

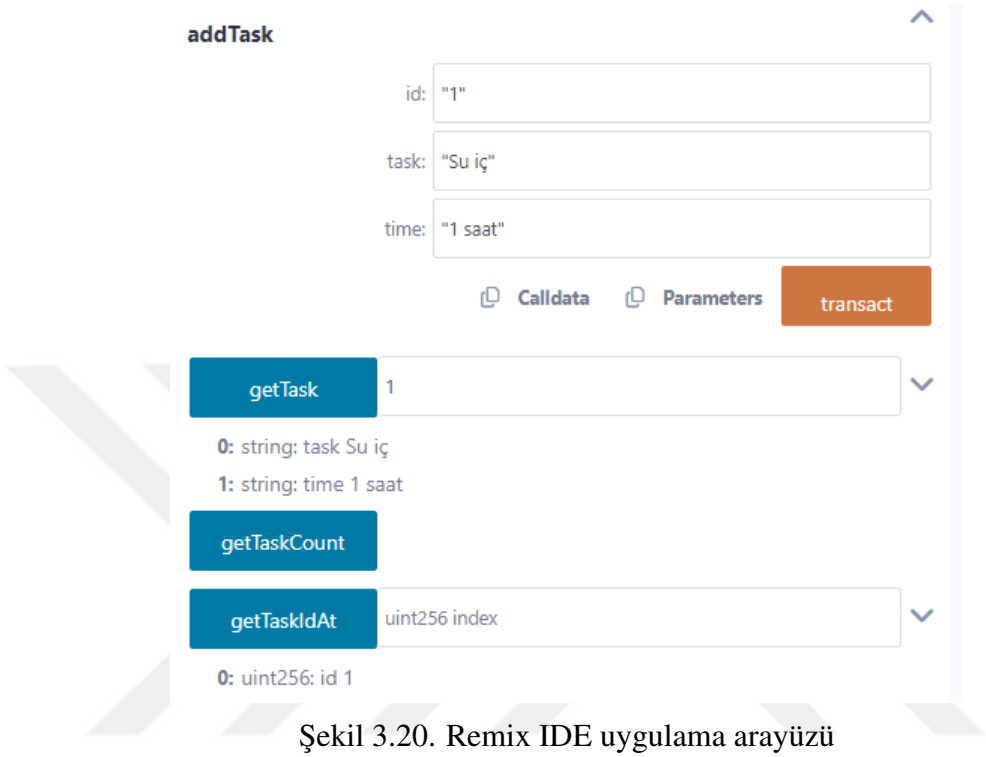
Birçok alanda kendini gösterebilen akıllı sözleşmeler uygulama sahasını da sürekli geliştirmektedir. Şekil 3.19, Yapılacaklar listesinin akıllı sözleşme kodunu göstermektedir.

```
function getTask(uint id) //görev kimliğine göre görev ayrıntısını alma.
        public view
        returns (string memory task, string memory time) {
    Routine storage routine = routines[id];
    return (
        routine.task,
        routine.time);
}

function addTask(uint id, string memory task, string memory time)//görevi eklemek için.
public
        returns (bool successful) {
    routines[id] = Routine({
        task: task,
        time: time
    });
    ids.push(id);
    emit LogTaskAdded(id, task, time);
    return true;
}
```

Şekil 3.19. Yapılacaklar Listesi (ToDo) Akıllı Sözleşme Kodu

Yine Ethereum'un web tarayıcısı üzerinde çalışan derleyicisi olan Remix IDE üzerinde derlenip deploy edilen akıllı sözleşme, yapılacaklar listesinin genel yapısında yer alan

yapılacak görev, ne zaman yapılacağı, ne kadar süreceği ve daha sonra listeleyebilmek için görev id'lerini tutmaktadır. Örnek olarak 2 adet görev ("Kahvaltı Yap!","10:00","30 dk" ve "Su İç!","12:00","1 saat") eklenmiş ve indeksine ve id değerine göre görüntülenmesi sağlanmıştır (Şekil 3.20).



Şekil 3.20. Remix IDE uygulama arayüzü

3.8.3. Uygulama 3: Fund Raising (Fon (Bağış) Toplama) Uygulaması

Fund-Raising, bireyler, işletmeler, hayır kurumları veya devlet kurumlarını dâhil ederek gönüllü mali katkılar arama ve toplama sürecidir.

Fund-Raising, tipik olarak kar amacı gütmeyen kuruluşlar için para toplama çabalarına atıfta bulunsa da, bazen kâr amaçlı işletmeler için yatırımcıların veya diğer sermaye kaynaklarının belirlenmesi ve talep edilmesine atıfta bulunmak için kullanılır. Geleneksel olarak, bağış toplama çoğunlukla yüz yüze bağış toplama, örneğin kapı çalma gibi yollarla bağış istemekten ibaretti. Ancak son yıllarda, çevrimiçi bağış toplama veya tabandan bağış toplamanın yeniden biçimlendirilmiş versiyonu gibi yeni biçimler ortaya çıkmıştır.

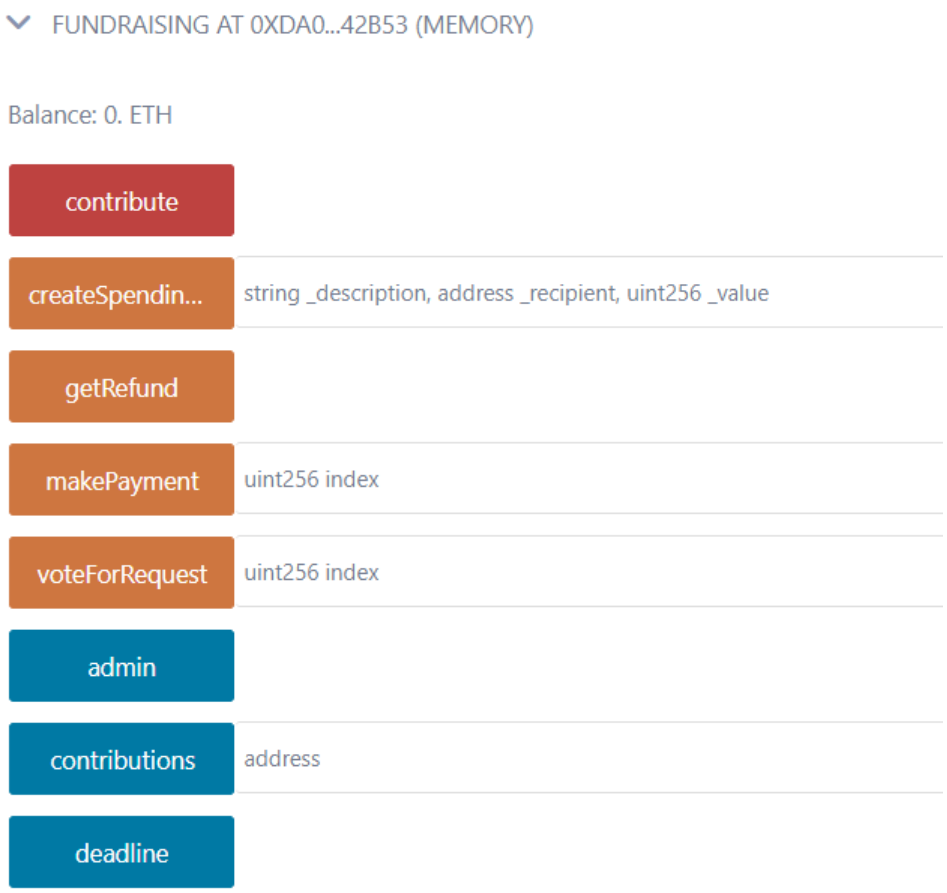
Kaynak yaratmayı organize etmenin en karlı yöntemlerinden biri bir etkinliktir (etkinlik için kaynak yaratma). Genellikle amaç, kâr amacı gütmeyen bir kuruluşun projesini finanse etmeye yetecek kadar para toplamaktır.

Bu, farklı mali ağırlıkta bireysel bağışlar alan büyük kuruluşlar için de geçerlidir. Bu durumda, bağışçıların katkılarının izlenebileceği (ve sıralanabileceği) bir veritabanına sahip olmak çok önemlidir. Böylece onlara destek düzeylerini yansıtan farklı iletişim taktikleri ile hitap edilebilir. Güvene ve sadakate büyük ölçüde güvenen kuruluşlar, taraftar tabanlarıyla iyi bir ilişkiyi sürdürmek için şeffaf ve duyarlı olmalıdır. İşte bu noktada bağış süreçlerindeki güven ve hesap verilebilirlik süreci akıllı sözleşmelerle de sağlanabilir. Bu sayede normal bağış sürecinin aynısı sayısallaştırılmış olur.

```
function createSpendingRequest(string _description, address _recipient, uint _value) public onlyAdmin{

    Request memory newRequest = Request(
        {
            description:_description,
            value:_value,
            recipient:_recipient,
            numberOfVoters:0,
            completed:false
            }
        );

    requests.push(newRequest);
}

function voteForRequest(uint index) public {
    Request storage thisRequest = requests[index];
    require(contributions[msg.sender] > 0);
    require(thisRequest.voters[msg.sender] == false);

    thisRequest.voters[msg.sender] = true;
    thisRequest.numberOfVoters++;
}

function makePayment(uint index) public onlyAdmin {
    Request storage thisRequest = requests[index];
    require(thisRequest.completed == false);
     require(thisRequest.numberOfVoters > totalContributors / 2);//more than 50% voted
    thisRequest.recipient.transfer(thisRequest.value);
    thisRequest.completed = true;
}
```

Şekil 3.21. FundRaising Akıllı Sözleşme Kodu

Şekil 3.21'de de yer alan akıllı sözleşme kodu, fundraising olarak bilinen fon (bağış) veya yatırım toplama organizasyonunun sözleşmeye dökülmüş halidir. Uygulamada, yatırım(bağış) yapılacak bir organizasyon ve bağışçılar bulunmaktadır. Minimum katkının üzerinde bağış yapan tüm kullanıcıların kaydı tutulmaktadır. Ayrıca, bağış yapan kullanıcıların toplam yaptıkları bağış miktarı (ether-wei cinsinden), adrese göre bağış miktarı sorgulama, ulaşılmak istenen miktar, adrese göre talep etme gibi özellikler bulunmaktadır. Uygulamanın remix IDE üzerinde deploy edilmiş hali Şekil 3.22'de verilmiştir.



Şekil 3.22. Remix IDE uygulama arayüzü

3.9. AKILLI SÖZLEŞMELERİN GAS ÜCRETİ, ÇALIŞMA ZAMANI VE BLOKZİNCİR AĞINA DAĞITIMLARI BAKIMINDAN DEĞERLENDİRİLMESİ

3.9.1. En Popüler Akıllı Sözleşme Dağıtım Zinciri Ağları

24 Ekim 2023 itibariyle, en popüler akıllı sözleşme dağıtım zinciri ağlarından bazıları şunlardır: Ethereum, Binance Smart Chain, Polygon, Avalanche, Solana, Fantom, Arbitrum, Optimism, Celo, Harmony, Near, Cosmos.

Bu zincir ağların hepsi Ethereum Sanal Makinesi (Ethereum Virtual Machine - EVM) uyumludur. Yani Solidity'de yazılan akıllı sözleşmeler bu ağlardan herhangi birine dağıtılabilir. Bu, geliştiricilerin ihtiyaçlarına bağlı olarak uygulamalarını farklı ağlara taşımalarını kolaylaştırmaktadır.

Bu zincir ağlara ek olarak, popülerlik kazanan bir dizi başka akıllı sözleşme platformu da vardır. Bu ağlar; Cardano, Polkadot, Algorand, Tezos ve EOS ağlarıdır. Bu platformların kendilerine has özellikleri ve faydaları vardır. Ethereum'a alternatif arayan geliştiricilerin ilgisini çekmektedirler.

Akıllı sözleşme dağıtım ortamının sürekli olarak geliştiğini ve her zaman yeni ağların ortaya çıktığını unutmamak önemlidir. Akıllı kontratınızı dağıtmak için bir ağ seçmeden önce araştırma yapılması önemlidir.

3.9.1.1. *Bazı Popüler Akıllı Sözleşme Dağıtım Zinciri Ağları ve Önemleri*

Aşağıda verilen blokzincir sözleşme geliştirme platformları, 2023 yılında piyasadaki en büyük ve en parlak platformlar arasında yer almaktadırlar.

Ethereum

Ethereum, merkezi bir otoritenin kontrolü olmadan uygulama ve organizasyonlar oluşturmak, varlıkları depolamak, işlem yapmak ve iletişim kurmak için kullanılan bir teknolojidir. Ethereum'u kullanmak için tüm kişisel verilerin teslim edilmesine gerek yoktur - bunların kontrolünü elinizde tutar ve nelerin teslim edileceğini kendiniz düzenlersiniz. Ethereum, ağındaki belirli eylemler için ödeme yapmak için kullanılan kendi kripto para birimi ether'e sahiptir.

2015'te başlatılan Ethereum akıllı sözleşme platformu, Bitcoin yeniliklerine dayanmaktadır, ancak bazı önemli farklılıkları da vardır. Her iki teknoloji de ödeme hizmeti sağlayıcılarının veya bankaların katılımı olmadan dijital para kullanılmasına olanak tanır. Ancak Ethereum platformu programlanabilir olduğundan, ağında merkezi olmayan uygulamalar da oluşturabilir ve dağıtılabilir.

Ethereum'un programlanabilir olması, verileri depolamak veya uygulama yeteneklerini yönetmek için akıllı sözleşme blok zincirini kullanan uygulamaların oluşturulabileceğini anlamını doğurur. Sonuç, her şeyi yapmak için programlanabilen genel amaçlı bir blok zinciridir. Ethereum'un yetenekleri sınırsız olduğundan, ağında büyük ölçekli inovasyona izin verir.

Binance Akıllı Zincir

Binance Smart Chain (BSC) ekosistemine dayalı tüm projeler ve tokenlar, Solidity dilini kullanarak blockchain akıllı sözleşmelerini işler. Binance Coin (BNB), Binance USD (BUSD), PancakeSwap (CAKE), Venus (XVS) ve diğer BEP-20 tokenları akıllı sözleşmeler oluşturmak için Solidity dilini kullanır. Binance Smart Chain, Ethereum'dan sonra en popüler ikinci merkezi olmayan uygulama ekosistemidir. BSC ağı daha iyi bant genişliğine sahiptir ve işlemler daha hızlı ve daha ucuzdur.

Binance Coin (BNB) başlangıçta Ethereum blok zincirinde yayınlanan bir ERC-20 token olarak yaratılmıştır. Daha sonra, Binance geliştiricileri kripto para birimi akıllı sözleşmesini geliştirmiştir. BNB tokenlerinin taşındığı ve bundan sonra bir kripto para birimi haline geldiği kendi ana ağı Binance Chain'i başlatmıştır.

Ardından Binance Chain'in geliştirilmiş bir versiyonu olan Binance Smart Chain ağı başlatılmıştır. Ethereum'a kıyasla daha yüksek verimliliğe sahip olmasına rağmen, BSC ağı aynı dezavantajlara sahiptir ve bu dezavantajlar blok zincirinin popülaritesi arttıkça daha belirgin hale gelmiştir. Örneğin işlemler yüksek yüklerde daha yavaş ve daha pahalı hale gelmiştir.

EOS

EOS, 2018 yılının ortalarında piyasaya sunulmuştur. Bu akıllı sözleşme platformu güçlü bir ekibe sahiptir. CTO Dan Larimer, Bitshares ve Steem gibi büyük ölçekli olanlar da dahil olmak üzere önceki projelerde başarılı bir geçmişe sahiptir. Bununla birlikte, EOS'un ana özelliklerinden birisi, ademi merkeziyetçiliğe odaklanmamasıdır.

TRON

Tron şu anda dağıtık defter teknolojisini kullanan en büyük blok zinciri tabanlı platform olarak kabul edilmektedir. Hizmetin amacı, çevrimiçi işlem yapmanın bir yolu olarak merkezi kanallara olan ihtiyacı azaltmaktır. Tron'un iki ana önceliği vardır. Ademi merkeziyetçilik ve demokratikleşme odaklı bir sistem sağlamak üzere tasarlanmıştır.

Tüm bunlar, ek izinlere ihtiyaç duymadan çalışan protokollerin ve araçların kullanılmasıyla elde edilir. İşlemlerin bütünlüğünü sağlamak için platform, Solidity dilinde oluşturulan akıllı sözleşmeleri kullanır. Bu programlama dili aslen Ethereum ağı için geliştirilmiştir. Tron için programlanan akıllı sözleşmelerin Ethereum ile tamamen uyumlu olduğu anlamına gelir. Buna ek olarak, akıllı sözleşme platformları içinde daha iyi ölçeklenebilirliğe sahiptir.

Stellar

Stellar kendisini "bankacılığın geleceği" olarak sunmakta ve ülkeden ülkeye ve para biriminden para birimine para transferi için merkezi olmayan bir ödeme sistemi olmayı hedeflemektedir. Aynı zamanda Stellar'ın mevcut merkezi finansal yapıların yerini alması planlanmamaktadır. Merkez bankalarıyla ortaklık kurarak ve kripto para birimlerini mevcut finansal ekosisteme entegre ederek ödemeler dünyasında barışçıl bir devrim yaratmayı amaçlamaktadır.

RSK

Bitcoin'e bağlı bir yan zincir olarak tasarlanmış bir akıllı sözleşme geliştirme platformudur. Dünyanın ilk, en güvenli ve güvenilir blok zincirine bağlı olması nedeniyle bu eklenti ona önemli bir avantaj sağlamaktadır.

Artık RSK Infrastructure Framework (RIF) OS adı verilen ve kendi token'ına da sahip olan ikinci bir katman vardır. RIFOS katmanı, geliştiricilere depolama, ödeme kanalları ve iletişim fırsatları da dahil olmak üzere tamamen akıllı sözleşme kripto para biriminin ötesinde özellikler sunar. RSK, Ethereum'un sunduğu her şeyi sunmaktadır ve muhtemelen 100 TPS'ye kadar hızlara ulaşarak ölçeklenebilirlik sorununu da çözebilirmektedir.

Ancak en büyük sorunu şu anda EOS ve Ethereum gibi benzer platformlar kadar tanınmıyor olmasıdır. Bununla birlikte, Bitcoin'in şu anda rakiplerinin üzerinde olduğu göz önüne alındığında, bu durum hızla değişebilmektedir.

Uniswap

Uniswap, ERC-20 tokenlarını takas etmek ve likidite (yield farming) sağlamak için oluşturulmuş bir DeFi-protokolüdür. Ethereum akıllı sözleşme platformuna dayanır ve merkezi olmayan bir borsa (DEX) operasyonları ile otomatik bir piyasa yapıcı (AMM) olarak işlev görür.

Merkezi kripto borsalarının aksine Uniswap, likidite iş adamları tarafından merkezi olmayan platforma sağlanan bir aracıya bağlı değildir. Bu, kullanıcıların aynı anda satışta dijital tokenleri güvenli bir şekilde takas etmelerine ve havuzlara token ekleyerek madencilik likiditesinden gelir elde etmelerine olanak tanır.

Uniswap ile ilgili temel endişe, blok zinciri platformunun Ethereum ekosistemine bağlı olmasıdır. Bu da ana akım blok zinciri ile aynı sorunlarla karşılaşacağı anlamına gelmektedir. Platforma olan talep arttıkça ölçeklendirme sorunları da artacak ve kullanıcıları alternatif desteler aramaya zorlayacaktır.

Polkadot

Polkadot, akıllı sözleşme işlevselliği sağlamak için Solidity dilini kullanan bir kripto para birimi platformudur. Polkadot ekosistemi birden fazla blok zincirini tek bir ağda birleştirerek parachain'ler oluşturur ve platformu ölçeklenebilir hale getirir. Yalnızca ağın bant genişliğini arttırmakla kalmaz, aynı zamanda birbirleriyle etkileşime de girebilirler; buna birlikte çalışabilirlik denir.

Polkadot ağı bölümlere ayrılmıştır ve Bitcoin veya Ethereum gibi diğer blockchain ağlarının aksine parachainler izole edilmemiştir. İşlemleri birbirleriyle paralel olarak işleyebilirler ve bir ağın aşırı yüklenmesi durumunda müşteriler işlem yapmak için diğer parachain'i kullanabilirler.

Ayrıca mimariye yönelik bu yaklaşım, bireysel blok zincirlerinin kimlik yönetimi veya hassas veri depolama gibi belirli görevler için optimize edilmesine olanak tanır. Polkadot tabanlı merkezi olmayan uygulamalar da birbirleriyle etkileşime girebilecek. Bu platform, Ethereum ve diğer benzer ağların ana rakiplerinden biri olarak kabul edilmektedir.

3.9.2. Bir akıllı sözleşme dağıtım zinciri ağı seçerken göz önünde bulundurulması gereken faktörler nelerdir?

Bir akıllı sözleşme dağıtım zinciri ağı seçerken, dikkate alınması gereken bir dizi faktör vardır:

Güvenlik: Ağ iyi bir güvenlik geçmişine sahip olmalı ve kullanıcı verilerini ve fonlarını korumak için güçlü kriptografik algoritmalar kullanmalıdır.

Ölçeklenebilirlik: Ağ, tıkanmadan ya da yavaşlamadan saniyede çok sayıda işlemi idare edebilmelidir.

Maliyet: Ağın işlem ücretleri ve gaz maliyetleri düşük olmalıdır.

Geliştirici topluluğu: Ağ, destek ve kaynak sağlayabilecek geniş ve aktif bir geliştirici topluluğuna sahip olmalıdır.

Ekosistem: Ağ, merkezi olmayan uygulamalar (DApp'ler) ve diğer projelerden oluşan sağlıklı bir ekosisteme sahip olmalıdır.

Bu genel faktörlere ek olarak, aşağıdakileri de göz önünde bulundurulmalıdır:

Akıllı sözleşme programlama dili: Ağ, Solidity veya Rust gibi programlama dillerini desteklemelidir.

Çapraz zincir uyumluluğu: Diğer ağlardaki akıllı sözleşmelerle etkileşime geçilmek istenmesi durumunda, çapraz zincir uyumlu bir ağ seçilmelidir.

Gizlilik: Kullanıcı gizliliğini koruyan akıllı kontratlar kullanılması gerekiyorsa, gizlilik özelliklerini destekleyen bir ağ seçilmelidir.

Tüm bu faktörleri göz önünde bulundurduktan sonra ihtiyaçlara en uygun ağ seçilmelidir. Bu özelliklere göre zincir ağların kıyaslamasının yapılabilmesi için tüm bu alanlarda ağların karşılaştırmaları yapılmış ve tablolarla verilmiştir.

3.9.2.1. *Akıllı Sözleşme Dağıtım Zincir Ağlarının, Güvenlik Yönünden Kıyaslaması*

Bir akıllı sözleşme uygulaması gerçekleştirilirken tespit edilmesi gereken en önemli unsurlardan birisi doğru blokzincir ağının seçilmesidir. Bu blokzincir ağının seçilebilmesi için birçok faktörün bir arada değerlendirilmesi gerekmektedir. Bunlardan ilki, ağların güvenlik yönünden tercihinin yapılmasıdır.

Çizelge 3.1. Zincir Ağlarının Güvenlik Durumları

Zincir Ağ	Kullanılan Kriptografik Algoritmalar
Ethereum	Eliptik eğri kriptografisi (ECC), Keccak-256 karma algoritması, AES-256 şifreleme
Binance Smart Chain	ECC, SHA-256 karma algoritması, AES-256 şifreleme
Polygon	ECC, Keccak-256 karma algoritması, AES-256 şifreleme
Avalanche	ECC, SHA-256 karma algoritması, AES-256 şifreleme
Solana	ECC, SHA-256 karma algoritması, AES-256 şifreleme
Fantom	ECC, SHA-256 karma algoritması, AES-256 şifreleme
Arbitrum	ECC, Keccak-256 karma algoritması, AES-256 şifreleme
Optimism	ECC, Keccak-256 karma algoritması, AES-256 şifreleme
Celo	ECC, SHA-256 karma algoritması, AES-256 şifreleme
Harmony	ECC, SHA-256 karma algoritması, AES-256 şifreleme
Near	ECC, Keccak-256 karma algoritması, AES-256 şifreleme
Cosmos	ECC, SHA-256 karma algoritması, AES-256 şifreleme
Cardano	ECC, SHA-256 karma algoritması, AES-256 şifreleme
Polkadot	ECC, SHA-256 karma algoritması, AES-256 şifreleme
Algorand	ECC, SHA-256 karma algoritması, AES-256 şifreleme
Tezos	ECC, SHA-256 karma algoritması, AES-256 şifreleme
EOS	ECC, SHA-256 karma algoritması, AES-256 şifreleme

Çizelge 3.1'de listelenen tüm ağlar, kullanıcı verilerini ve fonlarını korumak için güçlü kriptografik algoritmalar kullanır. Bununla birlikte, kullanılan belirli algoritmalarda bazı farklılıklar vardır. Örneğin, Ethereum ve Arbitrum hash algoritması olarak Keccak-256 kullanırken Binance Smart Chain ve Polygon SHA-256 kullanır.

Kriptografinin hızla gelişen bir alan olduğunu ve sürekli olarak yeni algoritmalar geliştirildiğini unutmamak önemlidir. Sonuç olarak, en son güvenlik en iyi uygulamaları konusunda güncel kalmayı taahhüt eden bir ağ seçmek önemlidir.

Buna ek olarak, bir akıllı sözleşmenin güvenliğinin yalnızca ağ tarafından kullanılan kriptografik algoritmalara bağlı olmadığına dikkat etmek önemlidir. Aynı zamanda akıllı

sözleşme kodunun kalitesine de bağlıdır. Geliştiriciler her zaman güvenli ve iyi test edilmiş akıllı sözleşmeler yazmaya özen göstermelidir.

3.9.2.2. *Akıllı Sözleşme Dağıtım Zinciri Ağlarının, Ölçeklenebilirlik Açısından Karşılaştırması*

Blokzincir ağının seçilebilmesi için değerlendirilmesi gereken faktörlerden bir diğeri, ölçeklenebilirlik faktörüdür.

Çizelge 3.2. Zinciri Ağlarının Ölçeklenebilirlik Durumları

Zincir Ağ	Saniye başına işlem sayısı (TPS)
Ethereum	15-30 TPS
Binance Smart Chain	250 TPS
Polygon	65.000 TPS
Avalanche	4,500 TPS
Solana	50.000 TPS
Fantom	2.000 TPS
Arbitrum	10.000 TPS
Optimism	10.000 TPS
Celo	100 TPS
Harmony	2,100 TPS
Near	1.000 TPS
Cosmos	1.000 TPS
Cardano	1.700 TPS
Polkadot	1.000 TPS
Algorand	1.000 TPS
Tezos	12 TPS
EOS	4.000 TPS

Çizelge 3.2'de görüldüğü gibi, farklı akıllı sözleşme platformlarının TPS'lerinde geniş bir aralık vardır. Ethereum ve Tezos gibi bazı ağlar nispeten düşük TPS'ye sahiptir. Bu da işlem aktivitesinde ani bir artış olması durumunda tıkanabilecekleri ya da yavaşlayabilecekleri anlamına gelir.

Polygon ve Solana gibi diğer ağlar ise çok daha yüksek TPS değerlerine sahiptir. Bu, tıkanmadan saniyede çok sayıda işlemi gerçekleştirebilecekleri anlamına gelir. Bir ağın TPS'sinin ölçeklenebilirliği etkileyen tek faktör olmadığına dikkat etmek önemlidir. Ağ durumunun boyutu ve blok süresi gibi diğer faktörler de bir rol oynar. Ayrıca, bir ağın TPS'sinin her zaman sabit olmadığına dikkat etmek önemlidir. İşlenmekte olan işlemlerin türü ve ağ yükü gibi bir dizi faktöre bağlı olarak değişebilir.

Genel olarak, en ölçeklenebilir akıllı sözleşme platformları, yüksek TPS'ye sahip olan ve tıkanmadan saniyede çok sayıda işlemi gerçekleştirebilen platformlardır.

3.9.2.3. *Akıllı Sözleşme Dağıtım Zinciri Ağlarının Maliyet Açısından Karşılaştırması*

Blokzincir ağının seçilebilmesi için değerlendirilmesi gereken faktörlerden bir diğeri, maliyet faktörüdür. Maliyet kullanıcılara doğrudan yansıyan bir faktör olduğundan dolayı daha da önem verilmesi ve dikkatli değerlendirilmesi gereken bir faktördür.

Çizelge 3.3. Zincir Ağlarının Maliyet Durumları

Zincir Ağ	Ortalama işlem ücreti	Ortalama gaz maliyeti
Ethereum	$0.20-$10	50-100 Gwei
Binance Smart Chain	$0.01-$0.10	5-10 Gwei
Polygon	$0.001-$0.01	0.1-1 Gwei
Avalanche	$0.01-$0.10	5-10 Gwei
Solana	$0.001-$0.01	1-2 Gwei
Fantom	$0.01-$0.10	5-10 Gwei
Arbitrum	$0.01-$0.10	5-10 Gwei
Optimism	$0.01-$0.10	5-10 Gwei
Celo	$0.01-$0.10	5-10 Gwei
Harmony	$0.001-$0.01	1-2 Gwei
Near	$0.01-$0.10	5-10 Gwei
Cosmos	$0.01-$0.10	5-10 Gwei
Cardano	$0.01-$0.10	5-10 Gwei
Polkadot	$0.01-$0.10	5-10 Gwei
Algorand	$0.01-$0.10	5-10 Gwei
Tezos	$0.01-$0.10	5-10 Gwei
EOS	$0.01-$0.10	5-10 Gwei

*Gwei, işlemlerin ethereum ağı üzerindeki gas fiyatını belirlemek için kullanılan bir birimdir. Ether(ETH), ethereum'un alt birimi olduğu gibi, gwei'de ETH'nin küçük bir alt birimidir (1 ETH(ether)=1,000,000,000 Gwei).

Çizelge 3.3'te görüldüğü gibi, farklı akıllı sözleşme platformlarının işlem ücretleri ve gaz maliyetleri arasında geniş bir aralık vardır. Ethereum gibi bazı ağlar nispeten yüksek ücretlere ve gaz maliyetlerine sahiptir. Bunun nedeni Ethereum'un en popüler akıllı sözleşme platformu olması ve blok alanı için çok fazla talep olmasıdır.

Polygon ve Solana gibi diğer ağların ücretleri ve gaz maliyetleri çok daha düşüktür. Bunun nedeni, bu ağların Ethereum'dan daha az sıkışık olması ve blok alanı için daha az talep

olmasıdır. Bir ağın işlem ücretlerinin ve gaz maliyetlerinin, işlenen işlem türü ve ağ yükü gibi bir dizi faktöre bağlı olarak değişebileceğini unutmamak önemlidir. Ayrıca, Ethereum gibi bazı ağların gelecekte işlem ücretlerini ve gaz maliyetlerini azaltacak ölçeklendirme çözümleri üzerinde çalıştığını belirtmek önemlidir.

Genel olarak, en düşük işlem ücretlerine ve gaz maliyetlerine sahip ağlar, daha az sıkışık olan ve blok alanı için daha düşük talebe sahip olanlardır. Bununla birlikte, akıllı sözleşme dağıtmak için bir ağ seçmeden önce ilgili tüm faktörlerin göz önünde bulundurması önemlidir. İşlem ücretlerini ve gaz maliyetlerini azaltmak için bazı ek ipuçları aşağıda verilmiştir:

Gönderilen işlemin türü göz önünde bulundurulmalıdır. Akıllı sözleşme etkileşimleri gibi bazı işlemlerin ücretleri diğerlerinden daha yüksektir. İşlemin gönderildiği günün saatini göz önünde bulundurulmalıdır. İşlem ücretleri yoğun saatlerde daha yüksek olma eğilimindedir.

Bir katman-2 çözümünü kullanmak düşünülmelidir. Katman-2 çözümleri, birden fazla işlemi bir araya getirip zincir dışında işleyerek işlem ücretlerini ve gaz maliyetlerini azaltmaya yardımcı olabilir.

Akıllı Sözleşmeyi Uygulamanın Maliyeti Ne Kadardır?

Akıllı sözleşmeler, blok zinciri tabanlı sistemlerin yeni paradigmasıdır. Kullanıcıların değer ve bilgileri güvene dayalı olmayan bir şekilde aktarmasına olanak tanıyan, kendi kendini yürüten kod parçalarıdır. Endüstri Araştırmasına göre, küresel akıllı sözleşmeler pazar büyüklüğünün 2028 yılına kadar 1460,3 milyon dolara ulaşacağı tahmin edilmektedir.

Blokzinciri teknolojisinin artan popülaritesi nedeniyle, bu tür geliştirme hizmetlerine olan talep her zamankinden daha yüksektir. Bununla birlikte, işletmeler hala akıllı sözleşmelerin dağıtım maliyetini nasıl tahmin edecekleri konusunda bilgi sahibi değildirler.

Temel soru şudur: "Akıllı bir sözleşme oluşturmanın ve başarıyla uygulamanın maliyeti nedir?"

Akıllı Sözleşme Oluşturma ve Dağıtmanın Adım Adım Süreci

Akıllı sözleşmenizin Ethereum'un blok zinciri teknolojisi üzerine inşa edilmesi isteniyorsa, bu süreç platformun nasıl çalıştığını yeterince iyi bilen bir geliştirici gerektirir. Böylece kural setini takip ederken benzersiz yapılar oluşturulabilir. Ayrıca, tüm sürecin kapsamlı bir şekilde anlaşılması, daha kesin bir akıllı sözleşme maliyet tahmini yapılmasına yardımcı olur.

1. Adım. İş fikrinin tanımlanması. Bu, şirket içinde ya da dışarıdan bir danışmanın yardımıyla yapılabilir. Hangi sorunun çözülmeye çalışıldığı, hedef kitlenin kim olduğu, ne kadar paraları olduğu ve akıllı sözleşmeler platformunda ne kadar zaman harcayacaklarının iyi anlaşılması gerekir.

2. Adım. Ardından, dApp için kullanıcıların ihtiyaç duyduğu tüm temel işlevlere sahip bir token şablonu oluşturulmalıdır (örneğin, hesaplarından token aktarımı).

3. Adım. Bu işlem tamamlandıktan sonra, Solidity Compiler (solc) kullanarak bunu bayt koduna derlemeli ve "sözleşmeler" adlı bir dosya çıkarılmalıdır.

Adım 4. Daha sonra bu sözleşmelerin, gerçek fonların dâhil olduğu ana Ethereum ağına aktarılmadan önce test edilecekleri Ethereum Testnet'e dağıtılması gerekir. Bu noktada, Jest veya Mocha gibi test çerçeveleriyle kolay entegrasyon sağladığı için Truffle kullanılması önerilebilir.

5. Adım. Bir sonraki adım temel kullanıcı arayüzü (UI) bileşenlerinin oluşturulmasını içerir, böylece insanlar JavaScript, C++, Rust veya Python gibi kripto programlama dilleri hakkında hiçbir şey bilmeden bunlarla doğrudan etkileşime girebilir.

Geliştirme Nüansları: Bir Akıllı Sözleşmenin Maliyeti Ne Kadardır?

Akıllı sözleşmeler, herhangi bir blok zinciri geliştirme projesinin önemli bir parçasıdır. Geliştiricilerin merkezi olmayan uygulamalar (dApp'ler) oluşturmasına yardımcı olurlar ve ayrıca kullanıcıların merkezi kuruluşlara ihtiyaç duymadan işlem yapmasına olanak tanırlar. Bununla birlikte, akıllı sözleşmeler ucuz değildir - aslında, genellikle bir dApp'ın bütçesinde önemli bir maliyet faktörüdür. Peki, bir akıllı kontrat geliştirmenin gerçekte maliyeti nedir?

Akıllı sözleşmelerin uygulanmasının maliyetini etkileyen en yaygın faktörlerin ayrıntılı listesi aşağıda verilmiştir.

Proje karmaşıklığı

Akıllı sözleşme ne kadar karmaşık olursa, geliştirme de o kadar pahalı olur. Mühendislerin gerekli tüm özellikleri anlaması zaman ve çaba gerektirir. Böylece başka bir dilde ya da platformda (örneğin Ethereum) zaten inşa edilmiş bir ekosistemde her şeyin düzgün ve güvenli bir şekilde çalıştığından emin olunabilir. Ayrıca, bir adresten diğerine para transferi gibi herhangi bir işlevde çok sayıda hareketli parça varsa, test sırasında bazı gecikmeler beklenebilir çünkü yeterli kaynak mevcut olmayabilir.

Bakım ve teslimat sonrası hizmetler

Bakım maliyetleri çoğunlukla akıllı sözleşmede yapmak istenilen değişikliklerin sayısına bağlıdır. Şu faktörler de dikkate alınır: Akıllı sözleşme ne kadar karmaşık? Bu, iş ihtiyaçlara göre değişebilir. Ancak birden fazla geri arama içeren oldukça karmaşık bir akıllı sözleşme varsa, tüm bu ekstra adımları ve eylemleri gerektirmeyen bir sözleşmeden daha fazla zaman ve para tüketir. İşte bu noktada akıllı sözleşme geliştirme maliyeti birkaç kat artabilir.

Kod tabanı endüstri standartlarına uygun mu? Diğer popüler programlama dilleriyle ne kadar iyi çalışıyor? Çoğu geliştirici kendi ürünlerini oluştururken en iyi uygulamaların farkında olsa da, geliştirme sırasında tüm kodlama standartlarına uyulduğunun ya da yeni özellikler eklendikçe bunların zaman içinde korunduğunun garantisi yoktur. Bu durum, bir kuruluş içindeki diğer sistemlerle entegre edilmeye çalışıldığında ileride sorunlara yol açabilir. Bu tür sorunların piyasaya sürülmeden önce düzeltilmesiyle ilgili ekstra maliyetler ortaya çıkabilir.

Belirtmek gerekir ki kod tabanı endüstri standartlarına uygun olsa bile çok kullanıcı dostu olmayabilir. Bu, belirli işlevlerin ne yaptığını veya birbirleriyle nasıl etkileşime girdiklerini okumanın ve anlamanın zor olduğu anlamına gelebilir.

Projeye dahil olan kişi sayısı

Fikrin karmaşıklığı, onu uygulamak için gereken ekibi oluşturur. Bu nedenle, güvenilir bir hizmet sağlayıcıya danışmadan önce akıllı sözleşme geliştirme maliyetini tam olarak hesaplamak mümkün değildir. Proje üzerinde çalışan ekibin büyüklüğü, uygulanan teknolojiler kadar genel maliyeti de güçlü bir şekilde etkiler.

Bu arada, ne kadar çok kişi olursa, projenin tamamlanması o kadar çok zaman alır ve hata yapma olasılığı da o kadar yüksek olur. Buna ek olarak, akıllı sözleşme üzerinde aynı anda birden fazla ekip üyesi olmayan kişi çalışıyorsa, bu kişilerin birbirleriyle ek koordinasyona ihtiyaç duymaları nedeniyle maliyetlerin artması da muhtemeldir.

Araç kiti ve spesifik teknolojiler

Kullanılan araçlar da akıllı sözleşme geliştirme maliyetinin tahmin edilmesinde önemli bir rol oynar. Blockchain teknolojisi her geçen gün değişmektedir ve bu durum projenin bütçesini etkileyebilir. Örneğin, farklı programlama dilleri kullanan birden fazla geliştiriciyle karmaşık bir fikir üzerinde çalışan büyük bir ekip varsa, bu durum temel maliyeti artıracaktır.

Mevcut teknoloji yığınıyla çalışan bir çözüm aranıyorsa, Ethereum ya da Hyperledger Fabric (Bitcoin tabanlı) çerçevelerinden birini kullanmak uygun olabilir.. Tokenlar ve dijital varlıklar gibi uygulamalar geliştirmek için şablonlar sunulmaktadır. Akıllı sözleşme oluşturma ve dağıtma için gereken bazı teknoloji ve araç örnekleri şunlardır:

Ethereum, blok zinciri tabanlı bir akıllı sözleşme platformudur. Yani Ethereum ağı üzerinde çalışan merkezi olmayan bir sanal makinedir. Aracı olmadan eşler arası işlem yapılmasına olanak tanır. Ayrıca, taraflar arasında şeffaf bir şekilde para veya varlık alışverişini kolaylaştırmak için blok zinciri teknolojisini kullanan, kendi kendini yürüten bilgisayar kodları olan akıllı sözleşmeler de sağlar. Solidity, Ethereum akıllı sözleşmelerini yazmak için kullanılan birkaç dilden biridir (diğerleri Serpent ve LLL'dir). Günümüzde kullanılan en popüler programlama dillerinden biri olan JavaScript'i temel alır.

Web3JS, ConsenSys tarafından oluşturulan ve HTTP üzerinden JSON RPC'ler aracılığıyla yerel veya uzak bir makinede çalışan herhangi bir Ethereum düğümündeki çeşitli bileşenlerle etkileşim kurmak için kullanılabilen açık kaynaklı bir JavaScript

kütüphanesidir. Web3JS'yi akıllı sözleşmelerle etkileşim kurmak, işlem göndermek, blok zincirinden okumak vb. için kullanılanilir.

Truffle, birçok mühendis tarafından akıllı sözleşmeler oluşturmak ve dağıtmak için kullanılan en popüler Ethereum geliştirme çerçevelerinden biridir. Geliştirme, test ve dağıtım aşamalarında kullanılabilecek bir dizi araç sağlar. Truffle'ı kullanan geliştiriciler, Mocha ve Chai kullanarak akıllı sözleşmelerindeki çeşitli işlevler için testler yazabilir. Ayrıca TestRPC veya Ganache gibi Solidity'ye özgü test çerçevelerini de kullanabilirler.

Tüm nüanslar özetlenirse, oluşturma sürecinin yeterince zaman ve para harcadığını, dolayısıyla ucuz olamayacağı söylenebilir. İnternette bulunabilecek temel maliyet 7000$'dan başlayıp 50000$'a kadar çıkmaktadır. Dahası, aşırı karmaşık projeler birkaç kat daha yüksek tahmin edilebilir.

Akıllı Sözleşme Dağıtmanın Maliyetini Hesaplarken Dikkat Edilmesi Gerekenler

Kullanıcıların dağıtım için harcadıkları toplam tutar, temel sözleşmeler için 500$ ile daha karmaşık sözleşmeler için 5000$ arasında değişmektedir. Bu nedenle, "Bir akıllı sözleşmeyi dağıtmanın maliyeti nedir?" sorusunu yanıtlarken, nihai bütçeyi etkileyen faktörlerin listesinden bahsedilmelidir.

Gaz Ücretleri

Gaz ücretleri Ethereum ağına eter olarak ödenir. Bir akıllı sözleşme ile etkileşime girdiklerinde kullanıcının hesabından alınır ve akıllı sözleşmedeki her işlem, işlem veya işlev çağrısı için ücretlendirilir. Akıllı sözleşmeler için gaz maliyetlerini hesaplarken aşağıdaki soruları göz önünde bulundurulmalıdır:

Bu sözleşmenin kaç işlevi var?

Sözleşme daha az fonksiyona sahip birden fazla küçük sözleşmeye bölünebilir mi?

Aynı anda kaç işlem çalıştırılacak?

Her bir işlemin yürütülme süresi ne kadardır?

Bir işlem için kaç gaz ücreti alınacak?

Bu, bir işlemi yürütmek için gereken hesaplama miktarına göre belirlenir. Sözleşme ne kadar karmaşıksa o kadar fazla gaza ihtiyaç duyulacaktır.

Sözleşmenin Oluşturulması

Ethereum'un sarı kağıdına (amaç ve misyonunu açıkladığı arz metni) göre, bir akıllı sözleşmenin oluşturulması için temel maliyet yaklaşık 32000 gazdır. Bununla birlikte, bir akıllı sözleşmeyi dağıtmanın nihai maliyeti şunlara bağlıdır: Dağıtılan kodun boyutu (bayt cinsinden): Akıllı sözleşmenin boyutu, geliştirme ya da dağıtım süreçleri sırasında kaç satır eklenmesi, silinmesi ya da değiştirilmesi gerektiğiyle orantılıdır.

Tek bir çağrı ya da olaydan geçen işlem sayısı: Her işlem bayt kodu boyutunu biraz artırır. Çünkü eklenen, silinen veya değiştirilen her satırla ilişkili bazı meta veriler vardır.

Sözleşme Depolama

Depolama, sözleşmenin blokzincirinde kapladığı alan miktarıdır. Bunun bedeli gaz olarak ödenir ve bu maliyet sözleşmesini oluşturan ya da kullanan kişiye aktarılır. Ethereum'un sarı kağıdına göre, akıllı sözleşme platformu 256 bit için 20000 gas ücret almaktadır.

Depolama maliyetleri iki taraf arasında paylaştırılır: ilk bellek boyutunu ödemek zorunda olan bir sözleşmenin yaratıcısı; ve bu sözleşmeye işlem gönderen veya bu sözleşmeden bilgi okuyan herkesi içeren, onu kullanan herkestir.

Sözleşme Yürütme

Bir akıllı sözleşme yürütüldüğünde, her bir talimatın yürütülmesi için ödeme yapılır. Her bir talimat için gaz ücreti genellikle Wei cinsinden ifade edilir (1 Ether = 1.000.000.000 Wei).

Sözleşme yürütme maliyetleri, mevcut bir akıllı sözleşme içindeki işlevleri veya yöntemleri çağırırken ödenir. Bu, harici kütüphaneler kullanıldığında ya da diğer sözleşmelerle etkileşime girildiğinde de gerçekleşir (örneğin, bir ya da daha fazla taraf eşler arası mesajlaşma hizmeti kullanıyorsa).

Gas, bir blok zinciri ağında gerçekleştirilen her işlem için kullanılır. Yalnızca akıllı sözleşmelerin yürütülmesi için değil, aynı zamanda işlemlerin gönderilmesi ve verilerin zincir üzerinde depolanması için de kullanılır. Bu nedenle, Ethereum'un ağıyla her etkileşim sırasında ne kadar para harcandığını bilmek için mevcut gaz fiyatının takip edilmesi önemlidir.

Akıllı Sözleşme Geliştirme için Hizmet Sağlayıcı Nasıl Seçilir?

Bir akıllı sözleşme geliştirirken, hem iş ihtiyaçlarını anlayan, hem gerekli teknolojiyi bilen hem de kabul edilebilir fiyatlarla uygun çözümler sunan doğru hizmet sağlayıcısını seçmek önemlidir.

Seçilen hizmet sağlayıcının Ethereum tabanlı blok zinciri ağlarıyla çalışma deneyimine sahip olduğundan ve Solidity'yi (Ethereum tarafından kullanılan programlama dili) anladığından emin olunmalıdır. Güvenlik önlemleri sorulmalıdır: özel anahtarların nasıl sakladıklarını, kimlik avı saldırılarını önlemek için ne yaptıklarını vb.

Elbette, gelecekteki geliştirici ekibinin becerisi akıllı sözleşmelerin dağıtım maliyetini etkiler. Ancak aynı zamanda potansiyel risklerin de ortadan kalkması sağlanır. Örneğin, CryptoKitties adlı Ethereum tabanlı bir kripto para biriminde kullanıcıların aynı anda iki kedi satın alabildiği bir hata vardı - ve satın aldılar! Bu, yüksek talep ve yetersiz kapasite nedeniyle tüm ağın çökmesine neden oldu ve o gün 20 milyon doların üzerinde Ether kaybına yol açmıştır (sadece bir kişiden 15 milyon dolar).

Sonuç olarak, Akıllı sözleşmelerin uygulanmasının maliyeti birçok faktöre bağlıdır. Ancak bunlar arasında en önemlisi iş ihtiyacının tam analizidir. Dikkate alınması gereken bir diğer önemli husus ise potansiyel karmaşıklığıdır. Genel bir kural olarak, akıllı sözleşmenizde ne kadar çok kod satırı varsa, bunu oluşturmanın maliyeti de o kadar yüksek olur. Bunun nedeni, geliştiricilerin sıfırdan geliştirmek yerine mevcut araçları ve teknolojileri kullanarak projelerinde tasarruf etmelerinin birçok yolu olmasıdır. Örneğin, tüm kodun baştan yazılması yerine Truffle veya Remix gibi açık kaynaklı bir araç kullanılırsa, bu, yol boyunca her aşamada (ilk planlamadan teste kadar) geliştirme ve dağıtım maliyetlerini önemli ölçüde azaltacaktır.

İşlem Ücreti, Gas Ücreti, ve Gwei

İşlem ücretleri, kullanıcıların işlemlerini bir blok zinciri ağında işlemek için madencilere veya doğrulayıcılara ödedikleri ücretlerdir. İşlem ücretleri genellikle ağın yerel kripto para biriminde ödenir.

Gaz maliyeti, bir işlemin işlenmesi için gereken gaz miktarıdır. Gaz, bir işlemin gerçekleştirilmesi için gereken hesaplama çabasını temsil eden bir ölçü birimidir. Gaz maliyetleri genellikle ağın yerel kripto para biriminin bir değeri olan gwei cinsinden belirtilir.

Gwei ("gwee" olarak telaffuz edilir), Ethereum ağının yerel kripto para birimi olan eterin (ETH) bir değeridir. Bir gwei, bir ETH'nin milyarda birine eşittir. Bir işlemin gaz maliyeti, işlemin türüne ve onu gerçekleştirmek için gereken hesaplama çabasına bağlı olarak değişir. Örneğin, bir akıllı sözleşme etkileşimi tipik olarak basit bir ETH transferinden daha yüksek bir gaz maliyetine sahiptir. Gaz fiyatı kullanıcı tarafından belirlenir ve işlemlerinin ne kadar hızlı işleneceğini belirler. Daha yüksek bir gaz fiyatı, işlemin daha hızlı işleneceği anlamına gelir. Ancak aynı zamanda daha pahalı olacaktır.

İşlem ücretleri, blok zinciri ağındaki işlemleri işleyen ve doğrulayan madencilere veya doğrulayıcılara ödenir. Bu ücretler madencileri ve doğrulayıcıları ağa katılmaya teşvik eder ve ağın güvenli kalmasına yardımcı olur. Gaz maliyetlerinin ağa ve mevcut ağ koşullarına bağlı olarak değişebileceğini unutmamak önemlidir. Örneğin, Ethereum'daki gaz maliyetleri genellikle yoğun saatlerde daha yüksektir.

Farklı Ağlardan Bazı Gaz Ücreti, Gwei Ücreti Ve Toplam İşlem Ücreti Hesaplama Örnekleri

Örnek 1: Ethereum

İşlem: ETH aktarımı
Gaz maliyeti: 21,000 gaz
Gaz fiyatı: 100 gwei
Toplam işlem ücreti: 21.000 gas * 100 gwei = 2.100.000 gwei = 0,0021 ETH

Örnek 2: BNB

İşlem: BNB aktarımı

Gaz maliyeti: 100 gaz

Gaz fiyatı: 5 gwei

Toplam işlem ücreti: 100 gaz * 5 gwei = 500 gwei = 0,0005 BNB

Örnek 3: MATIC

İşlem: MATIC Aktarımı

Gaz maliyeti: 100 gaz

Gaz fiyatı: 1 gwei

Toplam işlem ücreti: 100 gaz * 1 gwei = 100 gwei = 0,0001 MATIC

Görüldüğü gibi, toplam işlem ücreti gaz maliyetinin gaz fiyatıyla çarpılmasıyla hesaplanır. Gaz fiyatı kullanıcı tarafından belirlenir ve işlemlerinin ne kadar hızlı işleneceğini belirler.

Daha yüksek bir gaz fiyatı, işlemin daha hızlı işleneceği anlamına gelir, ancak aynı zamanda daha pahalı olacaktır. Bir işlemin gaz maliyeti, işlemin karmaşıklığına ve ağ koşullarına bağlı olarak değişir. Örneğin, bir akıllı sözleşme etkileşimi, basit bir token transferine kıyasla tipik olarak daha yüksek bir gaz maliyetine sahip olacaktır.

Gaz maliyetlerinin yalnızca tahmini olduğunu unutmamak önemlidir. Bir işlemin gerçek gaz maliyeti, işlemin karmaşıklığına ve mevcut ağ koşullarına bağlı olarak değişebilir.

Solidity

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract SimpleSmartContract {
    constructor() {}

    function setGreeting(string memory _greeting) public {
        greeting = _greeting;
    }

    function getGreeting() public view returns (string memory) {
        return greeting;
    }

    string public greeting;
}
```

Şekil 3.23. Basit Kod Örneği

Şekil 3.23'de farklı ağlardaki dağıtım maliyetlerini karşılaştırmak için kullanabilecek basit bir akıllı sözleşme verilmiştir.

Bu çok basit bir akıllı sözleşmedir, ancak farklı ağlar arasındaki dağıtım maliyetlerindeki farkı göstermek için yeterli olacaktır. Çizelge 3.4'te, akıllı sözleşmenin farklı ağlardaki dağıtım maliyetlerinin bir karşılaştırması verilmiştir.

Çizelge 3.4. Örnek Sözleşmeye göre Zincir Ağlarının Maliyet Karşılaştırması

Zincir Ağ	Gaz maliyeti	Gaz fiyatı	Toplam işlem ücreti	Usd
Ethereum	1 million gas	100 gwei	100,000,000 gwei = 0.1 ETH	$181.67
Binance Smart Chain	10,000 gas	5 gwei	50,000 gwei = 0.0005 BNB	$0.11
Polygon	1,000 gas	1 gwei	1,000 gwei = 0.0001 MATIC	$0.00006

Görüldüğü gibi, bu basit akıllı sözleşmenin dağıtım maliyeti ağa bağlı olarak önemli ölçüde değişmektedir. Dağıtılması en pahalı ağ Ethereum'dur ve onu Binance Smart Chain ve Polygon takip etmektedir.

3.9.2.4. *Geliştirici Topluluğu Açısından Mevcut Tüm Akıllı Sözleşme Dağıtım Zinciri Ağlarının Karşılaştırması*

Blokzincir ağının seçilebilmesi için değerlendirilmesi gereken faktörlerden bir diğeri, geliştirici topluluğudur.

Çizelge 3.5'te görüldüğü gibi, Ethereum en büyük ve en aktif geliştirici topluluğuna sahiptir ve onu Binance Smart Chain, Polygon ve Avalanche takip etmektedir. Bu ağlar aynı zamanda geliştiriciler için en kapsamlı kaynaklara sahiptir.

Listedeki diğer ağlar daha küçük geliştirici topluluklarına sahiptir. Ancak yine de aktiftirler ve büyümektedirler. Bu ağlar ayrıca geliştiriciler için daha az kaynağa sahiptir. Ancak belgelerini ve öğreticilerini geliştirmek için hala çalışmaktadırlar. Bir akıllı kontrat dağıtım zinciri ağı seçerken, geliştirici topluluğunun büyüklüğünü ve etkinliğini göz önünde bulundurmak önemlidir. Büyük ve aktif bir geliştirici topluluğu, destek ve kaynak sağlayabilecek daha fazla kişi olduğu anlamına gelir. Aynı zamanda ağın uzun vadede başarılı olma olasılığının daha yüksek olduğu anlamına gelir.

Çizelge 3.5. Zincir Ağlarının Maliyet Durumları

Zincir Ağ	Geliştirici topluluk büyüklüğü	Aktif geliştirici topluluğu	Mevcut kaynaklar
Ethereum	Çok büyük ve aktif	Evet	Çok büyük ve kapsamlı
Binance Smart Chain	Büyük ve aktif	Evet	Nispeten küçük ama büyüyor
Polygon	Büyük ve aktif	Evet	Orta ölçekli ve büyüyen
Avalanche	Orta ve aktif	Evet	Orta ölçekli ve büyüyen
Solana	Orta ve aktif	Evet	Orta ölçekli ve büyüyen
Fantom	Orta ve aktif	Evet	Orta ölçekli ve büyüyen
Arbitrum	Orta ve aktif	Evet	Orta ölçekli ve büyüyen
Optimism	Orta ve aktif	Evet	Orta ölçekli ve büyüyen
Celo	Küçük ve aktif	Evet	Küçük ve büyüyor
Harmony	Küçük ve aktif	Evet	Küçük ve büyüyor
Near	Küçük ve aktif	Evet	Küçük ve büyüyor
Cosmos	Küçük ve aktif	Evet	Küçük ve büyüyor
Cardano	Küçük ve aktif	Evet	Küçük ve büyüyor
Polkadot	Küçük ve aktif	Evet	Küçük ve büyüyor
Algorand	Küçük ve aktif	Evet	Küçük ve büyüyor
Tezos	Küçük ve aktif	Evet	Orta ölçekli ve büyüyen
EOS	Küçük ve aktif	Evet	Küçük ve büyüyor

Geliştiriciler için mevcut olan kaynakları göz önünde bulundurmak da önemlidir. Kapsamlı kaynaklara sahip bir ağ, geliştiricilerin akıllı sözleşmeler oluşturmasını ve dağıtmasını kolaylaştıracaktır.

Son olarak, özel ihtiyaçların göz önünde bulundurulması önemlidir. Eğer geniş ve aktif bir geliştirici topluluğuna sahip bir ağa akıllı kontrat konuşlandırılması gerekiyorsa Ethereum ya da Binance Smart Chain iyi seçenekler olacaktır. Daha düşük ücretlere sahip bir ağa akıllı kontrat konuşlandırılması gerekiyorsa Polygon ya da Avalanche iyi seçenekler olabilir.

3.9.2.5. *Ekosistem Açısından Mevcut Tüm Akıllı Sözleşme Dağıtım Zinciri Ağlarının Karşılaştırması*

Blokzincir ağının seçilebilmesi için değerlendirilmesi gereken faktörlerden bir diğeri, ekosistem faktörüdür. Sözleşmenin sağlığı ekosistemin sağlığıyla doğru orantılıdır.

Çizelge 3.6. Zincir Ağlarının Ekosistem Durumları

Zincir Ağ	DApp Sayısı	Ekosistem sağlığı
Ethereum	Çok büyük ve çeşitli	Çok sağlıklı
Binance Smart Chain	Büyük ve büyüyor	Sağlıklı
Polygon	Büyük ve büyüyor	Sağlıklı
Avalanche	Orta ve büyüyen	Sağlıklı
Solana	Orta ve büyüyen	Sağlıklı
Fantom	Orta ve büyüyen	Sağlıklı
Arbitrum	Orta ve büyüyen	Sağlıklı
Optimism	Orta ve büyüyen	Sağlıklı
Celo	Küçük ve büyüyor	Orta derecede sağlıklı
Harmony	Küçük ve büyüyor	Orta derecede sağlıklı
Near	Küçük ve büyüyor	Orta derecede sağlıklı
Cosmos	Küçük ve büyüyor	Orta derecede sağlıklı
Cardano	Küçük ve büyüyor	Orta derecede sağlıklı
Polkadot	Küçük ve büyüyor	Orta derecede sağlıklı
Algorand	Küçük ve büyüyor	Orta derecede sağlıklı
Tezos	Küçük ve büyüyor	Orta derecede sağlıklı
EOS	Küçük ve büyüyor	Orta derecede sağlıklı

Çizelge 3.6'da görüldüğü gibi Ethereum, DApp'ler ve diğer projelerden oluşan en büyük ve en çeşitli ekosisteme sahiptir. Bunun nedeni Ethereum'un en eski ve en köklü akıllı sözleşme platformu olmasıdır.

Binance Smart Chain, Polygon ve Avalanche da DApp'ler ve diğer projelerden oluşan büyük ve büyüyen ekosistemlere sahiptir. Bu ağların hepsi nispeten yenidir, ancak Ethereum'dan daha düşük ücretleri ve daha hızlı işlem süreleri nedeniyle son yıllarda popülerlik kazanmışlardır.

Listedeki diğer ağlar daha küçük DApp ve diğer proje ekosistemlerine sahiptir, ancak hala büyümektedirler. Bu ağların hepsi geliştiricileri çekmek ve ekosistemlerini oluşturmak için çalışmaktadır.

Bir akıllı sözleşme dağıtım zinciri ağı seçerken, DApp ve proje ekosisteminin boyutunu ve sağlığını göz önünde bulundurmak önemlidir. Sağlıklı bir ekosistem, aralarından seçim yapılabilecek daha fazla DApp ve proje olduğu anlamına gelir ve aynı zamanda ağın uzun vadede başarılı olma olasılığının daha yüksek olduğu anlamına gelir.

Özel ihtiyaçların göz önünde bulundurulması önemlidir. Eğer büyük ve çeşitli DApp ve proje ekosistemine sahip bir ağa akıllı bir kontrat yerleştirilmesi gerekiyorsa Ethereum iyi bir seçim olacaktır. Daha düşük ücretlere ve daha hızlı işlem sürelerine sahip bir ağa akıllı kontrat yerleştirilmesi gerekiyorsa Binance Smart Chain, Polygon ya da Avalanche iyi seçenekler olacaktır.

3.9.2.6. *Akıllı Sözleşme Dağıtım Zinciri Ağlarının, Solidity Veya Rust Programlama Dilini Destekleme Durumları*

Blokzincir ağının seçilebilmesi için değerlendirilmesi gereken faktörlerden biri ve sonuncusu, programlama dillerini destekleme durumudur. Solidity, rüşt gibi programlama dilleri sözleşme kodlama açısından en çok kullanılan diller olduğundan dolayı, bu dilleri desteklemesi blokzincir ağının gelişimi açısından önemlidir.

Çizelge 3.7. Zincir Ağlarının Programlama Dillerine Destek Durumları

Zincir Ağ	Solidity	Rust
Ethereum	Evet	Evet, ama deneysel
Binance Smart Chain	Evet	Evet, ama deneysel
Polygon	Evet	Evet, ama deneysel
Avalanche	Evet	Evet, ama deneysel
Solana	Hayır	Evet
Fantom	Evet	Evet, ama deneysel
Arbitrum	Evet	Evet, ama deneysel
Optimism	Evet	Evet, ama deneysel
Celo	Evet	Hayır
Harmony	Evet	Hayır
Near	Evet	Hayır
Cosmos	Evet	Hayır
Cardano	Hayır	Evet
Polkadot	Evet	Evet, ama deneysel
Algorand	Hayır	Evet
Tezos	Evet	Hayır
EOS	Evet	Hayır

Çizelge 3.7'de görüldüğü gibi, Solidity en yaygın şekilde desteklenen akıllı sözleşme programlama dilidir. Solana ve Cardano haricinde tüm büyük akıllı sözleşme dağıtım zinciri ağları Solidity'yi desteklemektedir.

Rust daha yeni bir akıllı sözleşme programlama dilidir, ancak popülerlik kazanmaktadır. Bir dizi büyük akıllı sözleşme dağıtım zinciri ağı Rust'ı desteklemektedir. Ancak Rust desteğinin bu ağların çoğunda hala deneysel olduğunu belirtmek önemlidir.

Eğer Solidity ile daha rahat ediliyorsa, aralarından seçim yapabilecek çok çeşitli akıllı kontrat dağıtım zinciri ağları bulunmaktadır. Eğer en çok Rust ile rahat ediliyorsa, o zaman daha sınırlı sayıda seçenek vardır. Ancak Rust desteği hızla artmaktadır.

Bunun mevcut durumun yalnızca bir anlık görüntüsü olduğunu unutmamak önemlidir. Her zaman yeni akıllı kontrat dağıtım zinciri ağları ortaya çıkmakta ve mevcut ağlara sürekli olarak yeni özellikler ve yetenekler eklenmektedir.

4. İŞBİRLİKÇİ AKILLI SÖZLEŞMELER MODELİ

Akıllı sözleşmeler blokzincir teknolojisinin ile entegrasyonunu sağlandığından beri gelişimini hızla devam ettirmektedir. Hızlı ve sürekli olarak yeni alanlarda kendisini gösteren sözleşmelere son zamanlarda, IOT ağlar, Finans, Sağlık, Güvenlik, Oyun, Lojistik alanlarında çokça rastlanmaktadır. Yapılan literatür taramasında, geliştirilen akıllı sözleşmelerin tüm işlem ve metotlarının aynı sözleşme içerisinde oluşturulduğu görülmektedir. Oluşturulan bu sözleşmeler blokzincir ağında deploy edilmektedir. Bu tez çalışmasında önerilen model ise deploy edilen kontratların işbirlikçi yapı ile birbiri ile etkileşime girmesini sağlamaktadır. Önerilen modelin tam olarak anlaşılabilmesi için öncelikle basit bir akıllı sözleşme yapısının nasıl olduğu ve bu sözleşmelere ulaşma yöntemleri anlaşılmalıdır.

4.1. BASİT AKILLI SÖZLEŞME YAPISI

Bir akıllı sözleşmeyi oluşturan çok sayıda parametre vardır. Şekil 4.1'de bir sözleşmenin genel bir yapısı verilmiştir. Her sözleşme Pragma komutuyla başlar. İlk satırda yer alan "pragma solidity Û.4.8;" kodu, sözleşmenin hangi derleyici sürümünde çalışacağını belirler. Solidity sürekli gelişen ve değişen bir dil olduğundan dolayı, derleyicinin doğru seçilmesi önemlidir. Sürüm seçildikten sonra "Contract" anahtar sözcüğüyle sözleşme yazmaya başlanmaktadır. Sözleşme içeriği yapılmak istenen uygulamanın dinamiklerine göre geliştirilmektedir. Struct yapısı diğer nesneye yönelik dillerdeki gibi görev yapmaktadır. Şekilde yer alan örnekte, Client'ın bilgilerini tutup daha sonra kod tekrarını engelleyerek aynı parametrelere sahip farklı özellikte Client'lar geliştirilebilmektedir. Akıllı sözleşmeleri diğerlerinden ayıran özelliklerden bazıları 14.satırdaki fonksiyonda yer alan "Payable" , 8.satırda yer alan "mapping" ve 9.satırda yer alan "address" anahtar kelimeleri örnek gösterilebilmektedir. "payable" anahtar kelimesi o fonksiyonun dijital para alışverişine uygun olduğunu belirtmektedir. Pay fonksiyonuna kontratı kullanarak dijital para transferi yapılabilmektedir. "mapping" ise bir düzen oluşturmak için kullanılmaktadır. Verilen örnekte, adresleri client ile eşleyerek bir arada

tutulması sağlanmaktadır. Diğer bir ifadeyle, mapping işleminin ardından, çağırılan bir adresin hangi client'a ait olduğu bilgisi de sağlanmaktadır. "address" anahtar kelimesi ise blokzincir ağındaki kontratların veya kontratlara dijital para aktaran hesapların adreslerinin tutulması ve bilgilerine ulaşılabilmesi için kullanılmaktadır. Aynı zamanda 11.satırda gösterildiği gibi sözleşmelerin constructor yapısı da oluşturulabilmektedir. Solidity programlama dilinde; internal, external, private ve public olmak üzere 4 adet erişim belirleyici vardır. Public; içeriden ve dışarıdan hem görülüp hem çağrılabilen erişim türü, Private; yalnızca sözleşme tarafından erişime açık olan erişim türü, Internal; yalnızca içeriden veya kalıtım yoluyla oluşturulmuş sözleşmeler tarafından erişilebilen erişim türü, External ise; dışarıdan görülüp erişilebilen ancak içeriden erişilemeyen erişim türüdür. Güvenlik açısından erişim türlerinin doğru belirlenmesi çok önemlidir. Ethereum ağında bir adrese sahip bir sözleşmeye ulaşmak istenirse, ulaşmak istenen fonksiyon veya değişkenin external olarak belirlenmesi gerekmektedir.

```
pragma solidity ^0.4.8;                                          //derleyici sürümü

contract Municipality{                                           //sözleşme başlangıç komutu

    struct Client  {                                             // Struct yapısı örneği
    }

    mapping (address => Client) public clientStructs;            //mapping yapısı kullanım örneği
    address[] public propertyList;                               //array kullanımı

    function Municipality () payable {                           //Yapıcı metot örneği
    }

    function pay() payable external returns (bool success) {     //payable komutu kullanım örneği
    }
}
```

Şekil 4.1. Akıllı Sözleşme Kodu Mimarisi Örneği

Geliştirilen model ve dApp uygulaması solidity programlama dili kullanılarak geliştirilmiştir. Geliştirme ve derleme ortamı olarak ethereum'un remix ide'si kullanılmıştır.

4.2. AKILLI SÖZLEŞMELERE ERİŞİM YÖNTEMLERİ

4.2.1. Yükseltilebilir(Upgradeable) akıllı sözleşmeler

Ethereum akıllı sözleşmeleri oluşturulurken karşılaşılan en büyük problemlerden birisi, dağıtılan sözleşme ve işlemlerin değişmez olmasıdır. Bunun anlamı, bir sözleşmenin bir ağda dağıtıldıktan sonra kaynak kodunun değiştirilemeyeceğidir.

Bu durum geliştiricileri de baskı altına almaktadır. Çünkü bu durum hiçbir şekilde hata kabul etmemektedir. Kod tamamen sağlıklı yazılırsa problem olmayacaktır. Ancak, akıllı sözleşmeler Zero-day-exploits'e karşı savunmasızdır. Aynı nedenden dolayı, akıllı sözleşmeler yazılım yaşam döngüsü gibi güncellemelere sahip değildirler.

Blokzincir ağında dağıtılan sözleşme kodunu yükseltmek mümkün değildir. Ancak böyle bir ihtiyaç olduğunda, dağıtılan sözleşmelerin kullanılmasına olanak sağlamak için bir Proxy Sözleşme Mimarisi kurmak mümkündür. Proxy Sözleşmesi ile en son dağıtılan sözleşmenin adresi saklanabilmektedir. Bu özellik ise sözleşmenin adresi değiştiğinde yeni sözleşmenin adresini güncellemekte kullanılan bir teknolojidir.

Ancak sözleşmenin içeriğinden bilgi alma özelliğini taşımadığından dolayı yetersizdir.

4.2.2. İşlev imzasına göre harici işlev çağrısı

Ethereum ağının temel bileşenlerinden birisi EVM(Ethereum Sanal Makinesi)'dir. Akıllı sözleşme dillerinin kodunun çalıştırılabilmesi için EVM bayt kodu aracılığıyla derlenmesi gerekmektedir. EVM üzerinde yürütülen bir kod olan EVM bayt kodu, ABI(Uygulama İkili Arayüzü) ile etkileşime girmektedir. Örneğin, javascript dili ile bir sözleşme yazmak istendiğinde, ABI, EVM bayt kod ve javascript arasında iletişim kuran bir arayüz işlevi yürütür. ABI, bir sözleşmede işlevlerin nasıl çağırılacağına, verilerin nasıl alınacağına ilişkin bir standart tanımlayan arabirimdir. EVM, üst düzey akıllı sözleşme dillerininin kodlarını doğrudan yürütemez. Bu kodlar opkodlar (düşük seviyeli makine dilleri) yardımıyla derlenirler ve bayt kodunda kodlanırlar. Blokzincirinde sözleşmeyi dağıtabilmek için, kodun ABI ve Bayt kodunda (bin) derlenmesi gerekmektedir. Halihazırda ağda dağıtılmış olan bir sözleşme ile etkileşime geçebilmek için sözleşmenin işlev imzalarına veya gerçek koduna (ABI) sahip olmak etkileşimin daha sağlıklı ilerlemesi açısından faydalıdır. Ancak bu sözleşmelerin ABI'si bilinmiyorsa 4 yöntem uygulanabilmektedir.

1. Callcode:Kullanımdan kaldırıldı.
2. Call: Başka bir sözleşmenin kodunu yürütmek için kullanılmaktadır.
3. Staticcall:Normal çağrı ile aynıdır. Aradaki tek fark, çağrılan fonksiyonlar herhangi bir şekilde değiştirildiğinde geri dönmektedir.

4. Delegatecall:Bu işlevde de başka bir sözleşme kodu yürütülür ancak aradaki fark, çağırılan sözleşmede storage özelliği kullanılmaktadır.

4.2.3. Interface(Arayüz)

Interface, diğer nesneye yönelik programa dillerinde aldığı görevlere benzer şekilde solidity programlama dilinde de görev yapmaktadır. Bir Interface'in görevi, bir nesnenin çalışması için kullandığı işlevlerin açıklamasını tutmak ve başka bir nesnede belirlenen işlevleri yürütmektir.

Genellikle "Interface" komutuyla bir sözleşmenin en üst satırında bulunurlar. Başka bir sözleşmede yer alan işlevleri çağırmak için çağırmak istenilen fonksiyon ya da değişkenin bir imzasını tutar. Interface'in işlev imzaları noktalı virgül ile bitirilir ve liste şeklindedir. Interface, kod karmaşasının önüne geçerek hem tekrarı hem yükü azaltırlar. Interface ile ilgili bazı kısıtlamalar da vardır. Diğer interface'lerden miras alabilirler ancak diğer sözleşmelerden miras alamazlar. Bir Interface fonksiyonu yalnızca "external" tipinde olabilir. Constructor ve State Variable tanımlayamazlar.

Interface ve Soyut Sözleşmeler birçok yönden benzerdirler ve birbirlerinin yerine kullanılabilmektedirler. Bu çalışmada Interface yerine aynı özelliklere sahip soyut sözleşme yapısı, diğer bir sözleşmeden fonksiyon imzalarını tutmak için kullanılmıştır. Şekil 4.2, uygulamada kullanılan Abstract Contract(Interface) yapısını göstermektedir.

```
pragma solidity ^0.4.8;

contract Municipality
{
    function isSellable() external returns (uint);
}

contract Bank
{
    function bankB() external returns (uint);
}
```

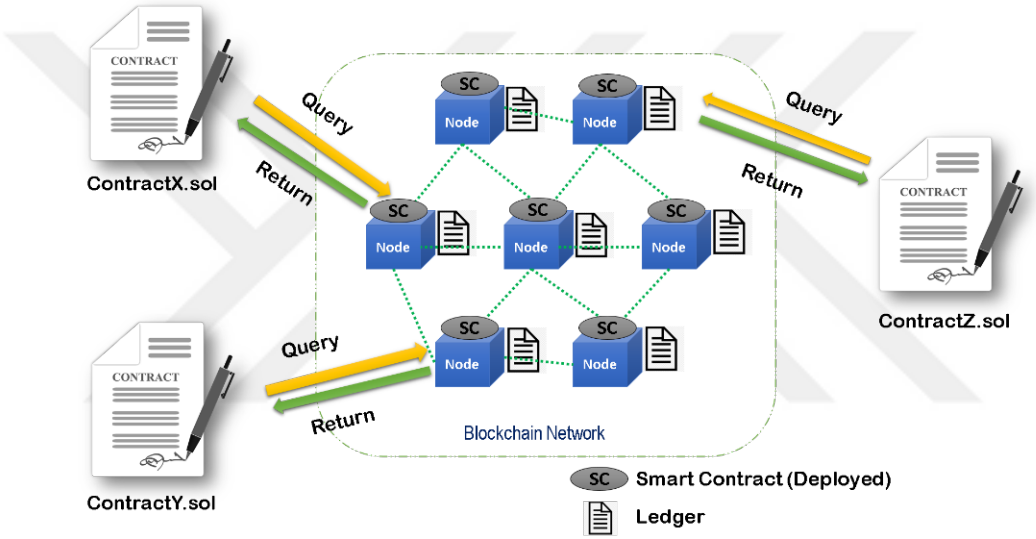
Şekil 4.2. Abstract Contract (Interface) Yapısı

Şekil 4.2'deki örnekte, 2 ayrı sözleşmeden (Municipality, Bank) çağırılmak istenen fonksiyon ve değişkenlerin imzası tutulmaktadır. External yapısı ise sözleşmeye dışarıdan erişim sağlamaktadır. Bu tez çalışmasında, sözleşmelere erişim yöntemlerinden, Interface yapısı kullanılarak, İşbirlikçi Akıllı Sözleşme Modeli oluşturulmuştur.

4.3. İŞBİRLİKÇİ AKILLI SÖZLEŞMELER MODELİ(İASM - COLLABORATIVE SMART CONTRACT(COSC) MODEL)

4.3.1. CoSC Çerçevesine Genel Bakış

Açık kaynak kodlu platformlar kendini geliştirdiği gibi akıllı sözleşmelerin de yenilenip geliştirilmesi gerekmektedir. Bir sözleşmeden Deploy edilmiş ağda adresi bilinen sözleşmeler ile tekrar iletişime geçmek, bu sözleşmelerin fonksiyon ve değişkenlerine ulaşarak sonuçlarını alabilmek için yeni bir modele ihtiyaç duyulmaktadır. Şekil 4.3'te akıllı sözleşmelerin işbirliği içerisinde çalışabilmesi için önerilen model verilmiştir.



Şekil 4.3. İşbirlikçi Akıllı Sözleşmeler(Collaborative Smart Contracts)

Şekil 4.3'teki modelde, blokzincir ağına deploy edilmiş kontratların ağda birer adresi bulunmaktadır. "0x6cd..." şeklinde her sözleşmenin blokzincir ağında bir benzersiz adresi bulunmaktadır. Bu sözleşmelerin içerisinde yer alan fonksiyonlara ve değişkenlere benzersiz adresleri aracılığıyla ulaşılabilmektedir. Şekil 4.4'te benzersiz adresi bulunan sözleşmeye ulaşma ve sözleşmeye ait fonksiyonu çağırma kodu verilmiştir.

Şekil 4.4'te gösterildiği gibi, Deploy edilmiş Municipality sözleşmesinin ağdaki örnek adresi mp nesnesinde tutulmaktadır. Bu nesne ile sözleşmeye ait isSellable değişkenini çağırılmış ve sonucu istenmiştir. Çağrı sonucu gelen veri döndürülmektedir. dApp uygulamasında client'a ait örnek bir adres tutulmuştur. Bu adres parametre olarak municipality sözleşmesindeki fonksiyona gönderilip, bu adrese sahip client'ın bilgileri

```
function getMuniFrom() public returns (uint results){

        address client = 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2;
        Municipality mp = Municipality(0xd9145CCE52D386f254917e481eB44e9943F39138);

        result = mp.isSellable();
        setMuni(results);
        return results;
}
```

Şekil 4.4. Blokzincir Ağına Dağıtılan Akıllı Sözleşmeyi Çağırma

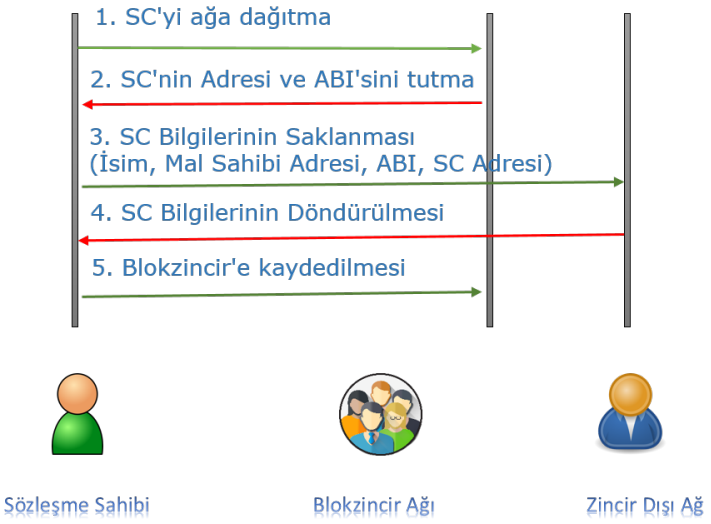
alınabilmektedir.

4.3.2. Akıllı Sözleşme Kayıt Prosedürü

Akıllı sözleşme kayıt prosedürü, akıllı sözleşme sahipleri için bir mekanizma sağlar. Bu mekanizma sayesinde akıllı sözleşme sahipleri, akıllı sözleşmelerini blok zinciri ağında dağıtabilir ve bu sözleşmeyle ilgili tüm bilgileri zincirde tutabilirler. Şekil 4.5'te akıllı sözleşme prosedürü ve sözleşme bilgilerinin kaydedilme aşamaları gösterilmektedir. Sözleşme sahibi akıllı sözleşme için gerekli kodu yazar, bu kodu bir derleme ortamında derler ve blokzinciri ağına dağıtır. Bu sözleşmeyi dağıtırken sözleşme adresini ve ABI (diğer sözleşme bilgilerini) alır. ABI bilgileri bu sözleşmeye erişimi olan kimlikler ve erişim bilgileri gibi bilgileri içerebilir. Alınan bilgiler daha sonra taşınmak üzere zincir içinde saklanır. Son adım olarak bu sözleşme, sahibi tarafından blokzincir ağına kaydedilir. Bu sözleşmenin versiyon bilgisi tutularak daha sonra sözleşme kodunda yapılacak güncelleme ile değişecek olan sözleşme adres bilgisinin güncellenmesi mümkün olur.

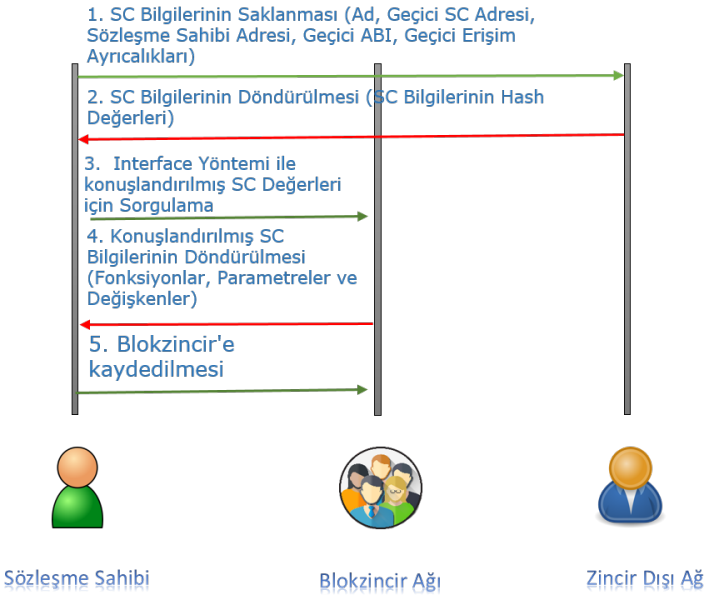
4.3.3. Dağıtılan akıllı sözleşmelerle etkileşim

Blok zinciri ağında konuşlandırılmış bir sözleşme ile etkileşim için bir prosedür olarak tanımlanır. Bu prosedür sayesinde, blok zinciri ağında halihazırda var olan bir sözleşmenin yeniden kullanılmasını sağlar. Bu da aynı kodun ağa yeniden dağıtılmasını engeller. Bu sayede ağın daha etkin kullanımı sağlanır. Bununla birlikte, tüm kodun tek bir sözleşmede yazılması yerine, farklı sözleşmelerin etkileşime girmesi sağlanarak bağımsız hareket



Şekil 4.5. Akıllı Sözleşme Kayıt İşlemi

etmek ve sözleşmeleri yönetmek daha kolay hale getirilir. Bir sözleşmede yapılan değişiklik sadece o sözleşmeyi etkiler. Şekil 4.6, konuşlandırılmış sözleşme ile etkileşim prosedürünü göstermektedir. Yeni oluşturulan sözleşmenin sahipleri, kendi kimlikleri ile etkileşime geçecekleri sözleşmenin bilgilerini talep ederler. Bu talebin blok zinciri ağına ulaşmasıyla birlikte blok zinciri, zincir dışı kayıtlardan bu kimliğin erişim durumunu kaydeder. İlgili kimliğin erişim ayrıcalığı ile talep edilen bilgi ilgili kullanıcıya iletilir. Etkileşimde bulunacağı sözleşme hakkında bilgi edindikten sonra, sözleşme sahipleri, sözleşmelerini kodlarken ilgili sözleşmenin işlevlerini ve değişkenlerini çağırmak için elde edilen bu sözleşme bilgilerini kullanır.



Şekil 4.6. Akıllı Sözleşme etkileşim prosedürü

4.3.4. Modelin Avantajları

Önerilen model akıllı sözleşmelerin gelişimi açısından önemlidir. Mevcut sözleşmeler, sadece ağa yüklenip dijital para transferi yapılabilen sözleşmelerdir. Bu çalışmada ise ağa yüklenmiş olan sözleşmelere ait fonksiyon ve değişkenlerin çağırılması, değişimlerinin takip edilmesi gibi birçok fayda bulundurmaktadır. Modelin öne çıkan avantajları arasında;

- Blokzincir ağı üzerine yerleşmiş bir sözleşmeye ulaşılabilmesi ve fonksiyonlarının kullanılabilmesi
- Kod karmaşasının engellenmesi
- Sözleşmelerin parçalar halinde yüklenerek maliyetin düşürülmesi (Bir sözleşme ağa yüklenirken kod satırı arttıkça fee maliyeti artmaktadır)
- Modülerlik sağlaması
- Yönetilebilirliği kolaylaştırması
- Güncel teknolojik gelişmelere entegrasyonu kolaylaştırması şeklinde birçok fayda sağlaması beklenmektedir.

Çizelge 4.1, Geleneksel Akıllı Sözleşmeler ile İşbirlikçi Akıllı Sözleşmelerin güvenlik, maliyet, uygulama, yükseltilebilirlik ve karmaşıklık gibi birçok farklı açıdan karşılaştırmasını göstermektedir. Bu değerler, bu çalışmada önerilen sistemin, İşbirlikçi Akıllı Sözleşmelerin, gelecek vaat ettiğini ve birçok yönden fayda sağlamaya hazır olduğunu göstermektedir.

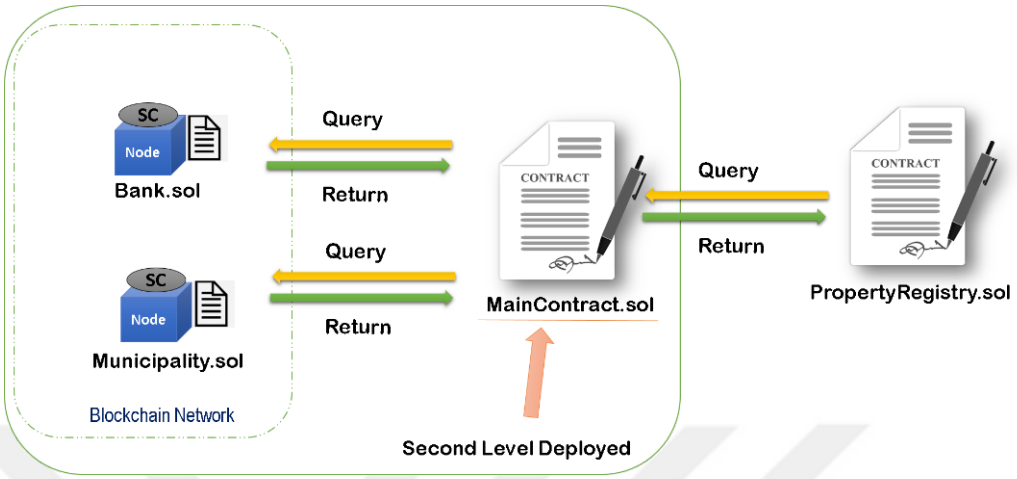
Çizelge 4.1. Geleneksel ve İşbirlikçi akıllı sözleşmelerin karşılaştırılması

Özellik	Geleneksel Akıllı Sözleşmeler	İşbirlikçi Akıllı Sözleşmeler
Tanım	Alıcı ve satıcı arasındaki anlaşmanın şartlarının doğrudan koda yazıldığı ve blokzincir ağında yürütüldüğü kendi kendine yürütülen sözleşme	Ortak bir amaç doğrultusunda birlikte çalışmayı kabul eden birden fazla tarafın yer aldığı ve anlaşma şartlarının yazılı olduğu sözleşme koda dönüştürülür ve bir blokzincir ağında yürütülür
Güvenlik	Güvenlik açıkları kodda veya platformda mevcut olabilir	Güvenlik, işbirlikçi doğası ve blokzincir teknolojisinin kullanımı yoluyla geliştirilmiştir
Maliyet	Daha fazla hesaplama kaynağına duyulan ihtiyaç nedeniyle gaz ücretleri daha yüksek olabilir	Kaynakları paylaşma ve maliyetleri dağıtma olanağı nedeniyle gaz ücretleri daha düşük olabilir
Uygulama	Genellikle basit işlemlerde ve finansal uygulamalarda kullanılır	Karmaşık tedarik zincirleri ve çok taraflı anlaşmalar dahil olmak üzere çok çeşitli uygulamalarda kullanılabilir
Yükseltilebilirlik	Zor	Kolay
Karmaşıklık	Karmaşıklık ve kapsam bakımından sınırlıdır	Daha karmaşık kullanım senaryolarını ve işlemleri yönetebilir

4.4. İŞBİRLİKÇİ AKILLI SÖZLEŞME MODELİNİN EV ALIM-SATIM ANLAŞMASINA UYGULANMASI

Bir gayrimenkul ya da taşınmaz malın el değiştirme işlemine tapu devri adı verilir. Tapu devri için en azından üç kurum ziyaret edilmelidir. Bu çalışmada yer alan örnekte; bir tapu

devri yapılırken belediye, banka ve tapu dairesi için farklı işlevlere sahip akıllı sözleşmeler kodlanmıştır. İşbirlikçi Akıllı Sözleşme Modeli'nin Tapu işlemlerine uyarlanması Şekil 4.7'de sunulmuştur.



Şekil 4.7. Merkeziyetsiz ev alım-satım uygulaması (dApp)

Uygulama birbiriyle işbirliği içerisinde hareket eden 4 adet sözleşmeden oluşmaktadır. İlk olarak "Bank.sol" ve "Municipality.sol" kontratları oluşturulup blokzincir ağına deploy edilmiştir. İkinci aşamada, "MainContract.sol" sözleşmesi, ağda deploy edilmiş iki kontrattan gerekli fonksiyonları çağırarak aldığı bilgileri değişkene atayacak şekilde deploy edilmiştir. Son olarak "PropertyRegistry.sol" sözleşmesi "MainContrat.sol" sözleşmesinden aldığı veriler ve kendi içerisinde yer alan şartların kontrolünü gerçekleştirmektedir. Sonuç Gayrimenkul 'un satılabilir olmasına uygun olursa, owner tarafından tapu devrinin gerçekleşmesine izin verilmektedir. Sonuç uygun çıkmazsa, taraflara neden satışa uygun olmadığı bilgisi döndürülerek anlaşma şartlarının yerine getirilmesi beklenmektedir. Şekil 4.8'de blokzincir ağında yer alan" Bank.sol" ve "Municipality.sol" akıllı sözleşmeleri ile etkileşime geçen "MainContract.sol" sözleşme kodu verilmiştir.

Kodlama yapılırken belirli bir düzende yapılmaktadır;

1. Municipality ve Bank akıllı sözleşmelerinden istenen bilgilerin Abstract Contract (Interface) ile işlev imzaları kodlanır.
2. Daha sonra, fonksiyonlardaki değişiklikleri sadece sözleşme sahibinin yapabileceği belirlenen fonksiyonlara uyarlamak üzere OnlyOwner isimli Modifier kodlanır.

```
contract Municipality { function isSellable() external returns (uint);}

contract Bank { function bankResults() external returns (uint);}

contract calltheDeployedFunction{

uint public ok1;
uint public ok2;
address owner;

modifier OnlyOwner(){
    require(owner==msg.sender,"you are not owner");
    _;
}

    function getMunifrom () public returns (uint resultMuni){

        //address client = 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2;
        Municipality mp = Municipality(0xd9145CCE52D386f254917e481eB44e9943F39138);
        resultMuni=mp.isSellable();
        setMuni(resultMuni);
        return resultMuni;
    }

    function getBankfrom () public returns (uint resultBank){

        //address client2 = 0x78731D3Ca6b7E34aC0F824c42a7cC18A495cabaB;
        Bank b = Bank(0x9ecEA68DE55F316B702f27eE389D10C2EE0dde84);
        resultBank=b.bankResults();
        setBankResult(resultBank);
        return resultBank;
    }
```

Şekil 4.8. MainContract.sol

3. Sözleşme üzerinde diğer iki sözleşme ile etkileşime girebilmek için, fonksiyonlar oluşturulur.
4. Etkileşime girilecek sözleşmelerin blokzincir ağı üzerinde yer alan adresi “address” tipinde değişkene atanır.
5. Municipality mp= Municipality(0xd9145CCE52D386f254917e481eB44e9943F39138) kodu ile sözleşmenin adresi üzerinden veriler mp nesnesine aktarılır.
6. Son olarak mp nesnesi ile ulaşmak istenen fonksiyon veya değişkenlere, parametreli veya parametresiz olarak ulaşılmaktadır.
7. Setter fonksiyonları ise şartların sağlandığına sözleşme sahibinin karar verebilmesi için oluşturulmuştur.

4.4.1. dApp Uygulama Sonuçları

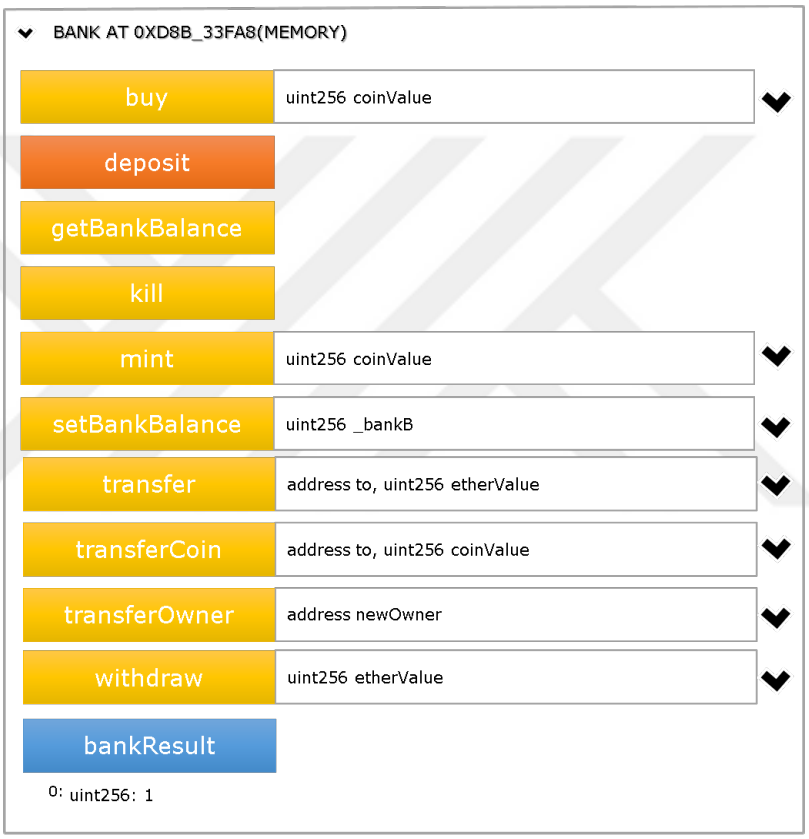
İşbirlikçi Akıllı Sözleşmeler Modeli ile gerçekleştirilmiş olan Ev Alım-Satım Uygulaması, Ethereum kurucuları tarafından sözleşme geliştiricileri için geliştirilen Remix IDE

ortamında geliştirilmiş, derlenmiş ve ethereum ağında test edilmiştir. Bu bölümde, geliştirilen uygulamaya ait arayüzlerin tanıtımlarına yer verilmiştir. Şekil 4.9'da 'Municipality.sol' sözleşmesinin içeriğine ait ekran görüntüsü verilmiştir. Sözleşme içerisinde her bir müşteriye ait ödeme bilgileri tutulmaktadır. Şekilde örnek olarak bir müşteriye ait, sözleşme üzerinde yapılmış 15 wei değerinde ödeme yapılmıştır. Yapılan bu ödeme, Rayiç bedeli ve vergi bedeli toplamı olarak alınmaktadır. Kod içerisinde, müşteriden alınan bedel müşteriye ait bilgilere kaydedilmektedir. isSellable değişkeni sözleşme içerisinde, rayiç bedeli ve vergi bedeline bağlı bir değişkendir. Diğer sözleşmelerden işbirlikçi akıllı sözleşme mantığı ile isSellable değerinin sonucu sorgulandığında, eğer bir eve ait rayiç ve vergi bedelleri sorgusu yapılan müşteri tarafından ödenmiş ise, 1 değeri, ödenmemiş ise 0 değeri döndürülmektedir.

Şekil 4.9. Municipality Sözleşmesi Arayüzü

Şekil 4.10'da 'Bank.sol' sözleşmesinin içeriğine ait ekran görüntüsü verilmiştir. Banka akıllı sözleşmesinde, birçok işlem yer almaktadır. Bunlar arasında; para yatırma, para çekme, para transfer etme, bankada mevcut para ile coin satın alabilme, banka bakiyesini görüntüleme gibi özellikler yer almaktadır. Bu sözleşmede yer alan BankResult değişkeni,

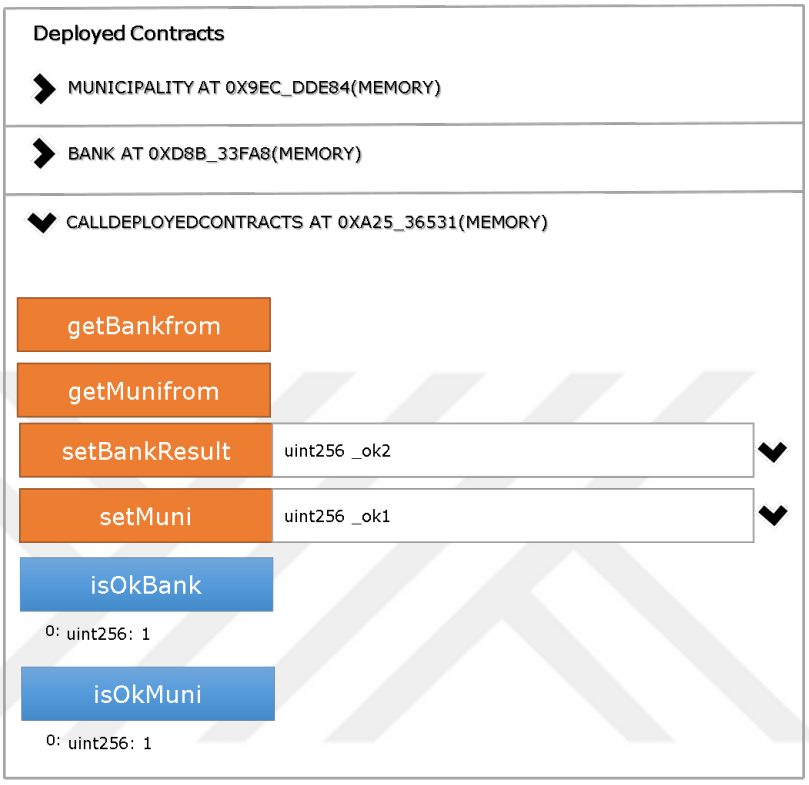
dışarıdan işbirlikçi akıllı sözleşmeler kullanılarak çağırılabilme özelliği taşımaktadır. Diğer sözleşmelerden bankResult değişkeni sonucu sorgulandığında; eğer müşteriye ait bankada yer alan yeterli miktarda kredi çekme varsa ve ilgili ev ile ilgili herhangi bir haciz vs durumu yer almıyorsa bu değişken sonucu 1 döndürülmektedir. Eğer bu değerlerden herhangi birisi olması gereken değeri karşılamıyorsa bankResult değeri 0 döndürülmektedir. Şekil 4.11'de 'MainContract.sol' sözleşmesinin içeriğine ait ekran görüntüsü verilmiştir. Şekil 4.9'da arayüzü verilen sözleşme 'Bank.sol' ve 'Municipality.sol' akıllı sözleşmeleri ile işbirliği içerisinde çalışması için oluşturulmuştur.



Şekil 4.10. Bank Sözleşmesi Arayüzü

Bu sözleşmede, interface metoduyla diğer sözleşmeleri imzaları tutulmaktadır. Bu sayede diğer sözleşmelerin nesneleri aracılığıyla, bu sözleşmelere ait istenilen fonksiyonlar ve değişkenler sorgulanmaktadır. Şekil 4.9'da 'Municipality.sol' sözleşmesine ait isSellable değişkeni sorgulanmış ve isMuniOk değişkenine atanmış, 'Bank.sol' sözleşmesine ait bankResult değişkeni sorgulanıp, isBankOk değişkenine atanmıştır. Eğer, elde edilen bu 2 değişken de 1 sonucunu döndürürse, ev satılabilir durumda olmaktadır. Bu 2 değişkenden 1 tanesinin bile 0 olması durumunda kullanıcıya 0 olan değerin(değerlerin) düzeltilmesi yani ödemesinin yapılması için bilgi mesajı döndürülmektedir. Şekil 4.9 ve Şekil 4.10'da

gösterilen sözleşmelerde istenilen kriterlerin karşılandığı ve buna bağlı olarak sonuçların 1 döndüğü görülmektedir. Bu durumda, Şekil 4.11'de işbirlikçi yapı sayesinde, diğer 2 sözleşmeden sorgulamalar yapılmış ve aynı sonuçlar elde edilmiştir. Bu sonuçlar, İşbirlikçi Akıllı Sözleşme Modelinin sağlıklı bir şekilde çalıştığını göstermektedir.



Şekil 4.11. Ana Sözleşme Arayüzü (Call the Contracts)

4.5. İŞBİRLİKÇİ AKILLI SÖZLEŞME MODELİNİN ARAÇ ALIM-SATIM ANLAŞMASINA UYGULANMASI

Günümüzde otomobil sahibi olmak, günlük hayatı kolaylaştıran unsurların başında gelmektedir. Öte yandan araç sahibi olmak bir çeşit birikim yöntemi olarak da görülmektedir. Her iki durumda da satın alınacak aracın ihtiyaçlara cevap vermesi, güvenli olması ve bütçeye uygunluğu önemli konulardır. Bu nedenle araç alırken bilinmesi önemli olan kimi konuların, satın alma işlemi yapılmadan önce gözden geçirilmesi kritik bir öneme sahiptir.

Bu tezde, araç alım satımının 2 tür versiyonu kodlanmıştır. Öncelikle, araç alım-satım sözleşmesi, klasik olarak bilinen tek bir akıllı sözleşme olarak kodlanmış, daha sonra ise işbirlikçi akıllı sözleşme olarak kodlanmıştır. Bu iki versiyonun sonuçları elde edilmiş, arasındaki avantaj farkları ortaya konmuştur. Son olarak ise, bu iki sözleşmenin blokzincir

ağına yüklenmesi sırasında harcanan gas miktarlarının karşılaştırması yapılarak sonuçları verilmiştir.

4.5.1. Tek Bir Sözleşme Olarak Araç Alım-Satım Akıllı Sözleşmesi

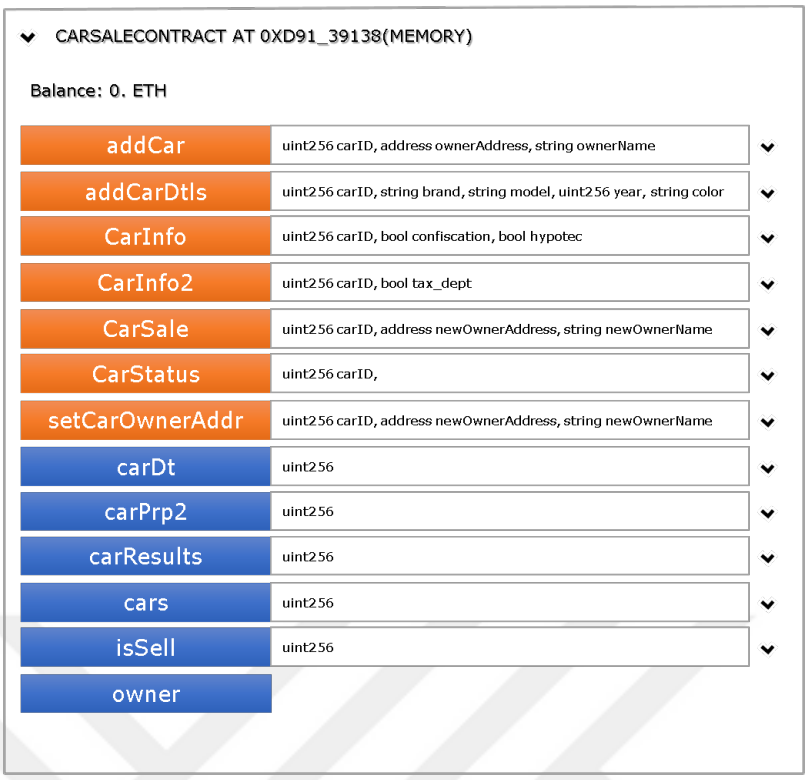
Akıllı sözleşmelerin geleneksel hali olarak bilinen versiyonu, tüm işlemlerin ve tarafların tek bir sözleşme içerisinde bulunduğu versiyonu olarak belirtilmektedir. Bu örnekte, araç alım-satım sözleşmesinin klasik versiyonu gösterilmektedir. Şekil 4.12'de yer alan görüntü, sözleşmenin başlangıç kısmına aittir. Bu kısımda bir aracın temel özellikleri tutulmaktadır.

```
1  // SPDX-License-Identifier: MIT
2  pragma solidity ^0.8.0;
3
4  contract CarSaleContract {
5      address public owner;
6
7      constructor() {
8          owner = msg.sender;
9      }
10
11     modifier onlyOwner() {
12         require(msg.sender == owner, "Only owner can call this function");
13         _;
14     }
15
16     struct Car {
17         address ownerAddress;
18         string ownerName;
19         bool exists;
20     }
21
22     struct CarDetails {
23         string brand;
24         string model;
25         uint year;
26         string color;
27         bool exists2;
28     }
```

Şekil 4.12. Araç satım sözleşmesinde araç detayları fonksiyonu.

Sözleşme kodlandıktan sonra ethereum'un remix ide isimli akıllı sözleşme çalıştırma ortamında derlenmiş ve çalıştırılmıştır. Şekil 4.13'te bu sözleşme'nin (CarSaleContract) çalışan hali verilmiştir. Sözleşme içerisinde, araç ekleme,araç detaylarını ekleme, araca ait vergi, ipotek ve vergi borcu bilgilerini girme, aracın uygun olduğu koşullarda satışını yapma gibi fonksiyonlar bulunmaktadır.

Bu tür sözleşmeler, tüm tarafların aynı sözleşme içerisinde yer alması açısından problemler barındırmaktadır. Çünkü sözleşmede yer alacak her türlü aksilikte, sözleşmenin yeniden kodlanması ve blokzincir ağına yeniden ve yeni bir sözleşme adresi ile dahil edilmesi gerekmektedir. Her yanlış tüm sözleşmeyi etkilemektedir. Blokzincir ağının verimsiz



Şekil 4.13. Araç satım sözleşmesi uygulama arayüzü

kullanımına ve bir sözleşme çöplüğüne dönmesine neden olmaktadır. Bu nedenlerden dolayı bu sözleşmelerin taraflara ayrılıp birbirleri ile iletişim ve etkileşim halinde çalışması elzem bir durumdur. Sonraki bölümde bu sözleşmenin işbirlikçi yapıda nasıl çalıştığına ve avantajlarına yer verilmiştir.

4.5.2. İşbirlikçi Araç Alım-Satım Akıllı Sözleşmesi

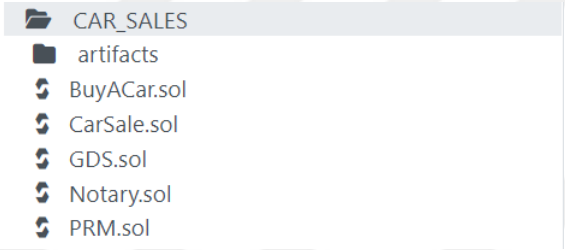
Bu tez çalışmasında, işbirlikçi akıllı sözleşmeler modeli, araç alım satım sözleşmesine uygulanmıştır. Araç alım-satımında, 5 ayrı akıllı sözleşme kodlanmış ve birbiri ile etkileşim halinde çalıştırılmıştır.

Şekil 4.14'te kodlanan 5 akıllı sözleşmenin dosya isimleri yer almaktadır. Kısaca özetlenirse, BuyACar.sol sözleşmesi, araçların temel bilgilerinin ve özelliklerinin tutulduğu sözleşmedir. Araçlara ait, araç sahibinin adresi, ismi, aracın model özellikleri bu sözleşmede tutulmaktadır.

GDS.sol sözleşmesi ise, ingilizcesi 'General Directorate of Security' olan Emniyet Genel Müdürlüğü'nün kısaltmasıdır. Bu sözleşmede, her bir araç için, bu birimdeki yetkili kişi tarafından, haciz ve ipotek bilgisi girilmekte ve satıma uygunluğu sorgulanabilmektedir.

PRM.sol sözleşmesi, ingilizcesi 'President of Revenue Management' olan Gelir İdaresi Başkanlığı'nın kısaltmasıdır. Bu sözleşmede, her bir araç için, bu birimdeki yetkili kişi tarafından, vergi bilgisi girilmekte ve satıma uygunluğu sorgulanabilmektedir. Notary.sol sözleşmesi, adından da anlaşılabileceği gibi, aracın satışa uygun olduğunun noter tarafından onaylandığı sözleşmedir.

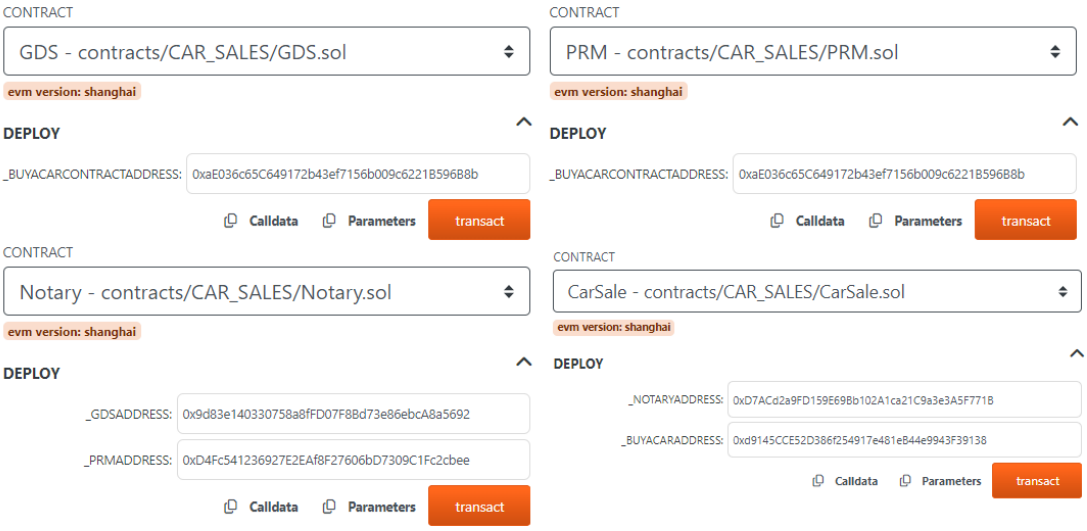
Son olarak, CarSale.sol isimli sözleşme ise, araç satışının yapıldığı, araca ait bilgiler sabitken, sahibi olarak yeni alıcısının adresinin atandığı sözleşmedir.



Şekil 4.14. İşbirlikçi Sözleşme Dosyaları

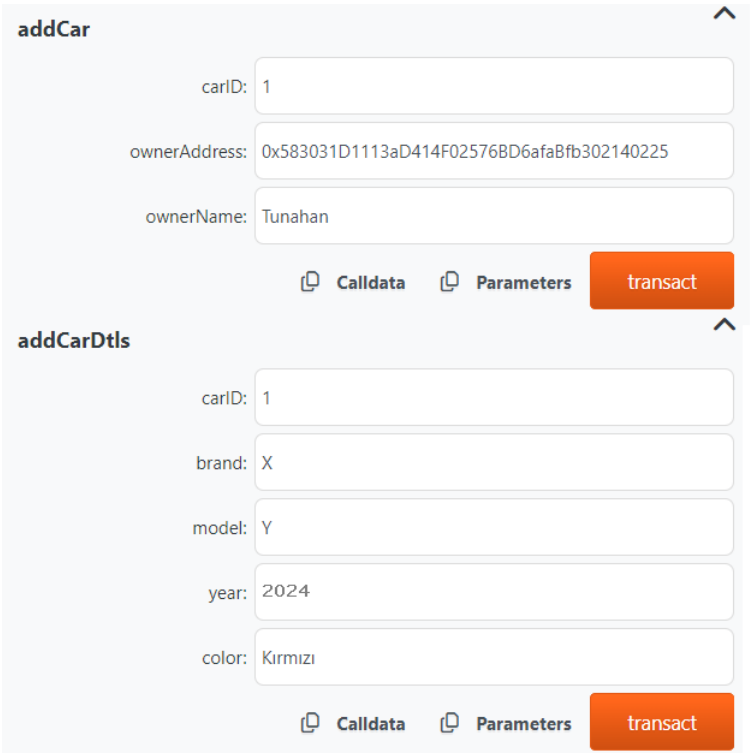
Şekil 4.15'te akıllı sözleşmelerin birbiri ile etkileşime geçmeleri için remix ide tarafından atanmış olan sözleşme adresleri verilmiştir. GDS sözleşmesi ve PRM sözleşmesi, BuyACar sözleşmesi ile etkileşim sağlamaktadır. Notary sözleşmesi hem GDS hem PRM sözleşmeleri ile etkileşime geçerek sorgulamalar yapmaktadır. CarSale sözleşmesi ile noterden gelen onay için Notary sözleşmesi ile, araç satımı gerçekleştikten sonra araç sahiplik bilgilerinin güncellenmesi için ise BuyACar sözleşmesi ile etkileşime girmektedir.

Akıllı sözleşmelerin işbirliği içerisinde çalıştığı bu model, barındırdığı özellikler sayesinde diğerlerinden avantajlı konumdadır. Her bir sözleşme diğer sözleşme ile benzersiz adresi sayesinde iletişime geçmektedir. Bu sözleşmelerden herhangi birinde problem çıkması durumunda, sadece probleme neden olan sözleşme yeniden kodlanıp blokzincir ağına dağıtılmaktadır. Diğer sözleşmelerde yapılacak değişim ise ağa yeniden dağıtılan sözleşmenin adresinin güncellenmesi olmaktadır. Bu sayede tüm sözleşmelerin yeniden blokzincir ağında dağıtılıp ağın verimsiz bir sözleşme çöplüğü olmasının önü kesilmiş olacaktır.



Şekil 4.15. İşbirlikçi sözleşme etkileşim durumları.

Şekil 4.16'da BuyACar sözleşmesine ait fonksiyonlar olan araç ekleme ve aracın detaylarını ekleme fonksiyonları verilmiştir. Bu fonksiyonlarda, örneklerde de görüldüğü gibi öncelikle bir aracı olan kişi kendi adresini ve aracın bilgilerini sözleşmeye kaydetmektedir. Bu sayede, bu aracı almak isteyen kullanıcılar araç sahibinin ve aracın bilgilerini görebilmektedir. Bu adres sayesinde araç satıldığında, sadece yeni kullanıcının adresi araca kaydedilerek alım-satım işlemi tamamlanmaktadır.



Şekil 4.16. İşbirlikçi sözleşme veri girişi.

4.5.3. Geleneksel Ve İşbirlikçi Akıllı Sözleşmenin Blokzincir Ağına Dağıtım Maliyetlerinin Karşılaştırılması

Bu tezde, genel olarak anlatılmak istenen, akıllı sözleşmeler alanında bilinen geleneksel akıllı sözleşmeler ile, bu tez çalışmasında ortaya koyulan işbirlikçi akıllı sözleşmelerin karşılaştırılmasının yapılmasıdır. Bu bölüme kadar uygulama sonuçlarına ve işbirlikçi sözleşmelerin tüm avantajlarına yer verilmiştir.

Bu bölümde ise, akıllı sözleşmelerin geleneksel ve işbirlikçi olarak blokzincir ağına dağıtıldığında harcanan maliyetlerin (gas maliyeti(ethereum)) ethereum sanal makinesinin web sürümü olan remix ide üzerinde karşılaştırmasını sunulmaktadır.

Çizelge 4.2 bu karşılaştırmaya dair sayısal sonuçlara yer verilmiştir.

Çizelge 4.2. Geleneksel ve İşbirlikçi Akıllı Sözleşmelerin maliyet karşılaştırması

#	Geleneksel Akıllı Sözleşme	İşbirlikçi Akıllı Sözleşme				
	CarSale.sol	BuyACar.sol	GDS.sol	PRM.sol	Notary.sol	CarSale.sol
Gas maliyeti	1745144 gas	1022164 gas	668118 gas	644512 gas	783453 gas	743349 gas
İşlem maliyeti	1517940 gas	889062 gas	581134 gas	560601 gas	681457 gas	646572 gas
Yürütme maliyeti	1361560 gas	775610 gas	488560 gas	469541 gas	581071 gas	549786 gas

Öncelikle tabloda yer alan gas maliyeti, işlem maliyeti ve yürütme maliyeti terimlerinin anlaşılması gerekmektedir.

Gas, ethereum blokzincir ağı üzerindeki işlem maliyetini ölçmek için kullanılan bir birimdir. Her işlem bir miktar gas tüketmektedir. Bu miktar ise ağdaki yoğunluk ve işlemin karmaşıklığına göre değişmektedir. Gas tüketen işlemler; veri depolama, akıllı sözleşmenin çalıştırılması gibi işlemlerdir. Gas, ethereum'un ekonomik anlamda oluşturduğu modelin bir parçasıdır ve aynı zamanda ağ güvenliğinin sağlanması için de kullanılmaktadır.

İşlem maliyeti ise, bir işlemin gerçekleştirilmesi için gereken toplam ücret olarak ifade edilmektedir. Bu ücret, gas fiyatı ve işlem sırasında tüketilen toplam gas miktarının çarpımı ile belirlenmektedir.

Yürütme maliyeti ise, bir sözleşmenin yürütülmesi için harcanan maliyet olarak ifade edilmektedir. Yani sözleşme çalıştırıldığında kod ile ilgili maliyetleri kapsar. Bir işlemdeki toplam maliyetin bir parçasıdır. İşlemdeki tüm kod bloklarının yürütülmesi sırasında harcanan veya gereken gas miktarıdır.

Bu durumda tabloda yer alan bilgilere göre, şu çıkarımlar yapılabilmektedir.

1. İşbirlikçi sözleşmeler ayrı ayrı düşünüldüğünde, her birinin ağa dağıtılma maliyeti geleneksel sözleşmenin dağıtım maliyetinden ortalama yarı yarıya düşüktür.

2. İşbirlikçi sözleşme toplu olarak ele alınırsa dağıtım maliyeti, geleneksel sözleşmeye göre yüksek olmaktadır.

*Bir sözleşmenin ağda dağıtım maliyeti kod miktarının uzunluğu ve yapılan işlemlerin karmaşıklığı ile doğru orantılıdır. Bu nedenle, sözleşme parçalara ayrıldığında, kod miktarı arttığından dolayı (Bu duruma her bir sözleşme başında yer alan pragma komutu bile dahildir) maliyet de artmaktadır. 2. madde bu nedenden kaynaklanmaktadır. Her iki sözleşme ağa dağıtıldıktan sonra yapılacak bir işlemin karşılaştırılması aşağıda verilmiştir.

a. Yapılan işlem; CarID:1, araç sahibi adresi:0xd…., araç sahibi ismi:tunahan

Araca ait belirtilen bilgilerin girildiği fonksiyon çalıştırıldıktan sonra Çizelge 4.3'teki sayısal değerler elde edilmiştir.

Çizelge 4.3. Durum 1

#	Geleneksel Akıllı Sözleşme	İşbirlikçi Akıllı Sözleşme
Harcanan Gas Miktarı	107064 gas	104572 gas

b. Yapılan işlem; CarID:1, araç marka:X, araç model:Y, araç yıl: 2022, araç renk: kırmızı

Bir diğer fonksiyonda sahibi eklenen aracın diğer detay bilgileri girilmiştir. Bu fonksiyon çalıştırıldığında harcanan gas miktarları tablo 4.4'teki sonuçlar elde edilmiştir.

Çizelge 4.4. Durum 2

#	Geleneksel Akıllı Sözleşme	İşbirlikçi Akıllı Sözleşme
Harcanan Gas Miktarı	158903 gas	158801 gas

Verilen örneklerde, işbirlikçi akıllı sözleşmeler daha iyi sonuçlar vermiştir. Ancak geleneksel sözleşmenin iyi sonuç verdiği örnekler de bulunmaktadır. Ayrıca iki sözleşme türü farklı sayıda sözleşmeler içerdiğinden dolayı ortaya çıkan sonuçların karşılaştırılması da sağlıklı olmamaktadır. Genel olarak karşılaştırıldığında, bu tezde modeli sunulan işbirlikçi akıllı sözleşmeler modelinin, geleneksel akıllı sözleşmelere göre büyük oranda daha avantajlı olduğu tüm sonuçlara yansımaktadır.

5. SONUÇLAR

Akıllı sözleşmelerin ve blokzincir'in entegrasyonu çeşitli endüstrilerde devrim yaratmıştır. Ethereum'un kurucu ortağı Vitalik Buterin'in belirttiği gibi, "Bitcoin her şeyden önce bir para birimidir; bu, blokzincir'in özel bir uygulamasıdır. Ancak tek uygulama olmaktan uzaktır." 2015 yılında piyasaya sürülen Ethereum, merkezi olmayan uygulamalar (DApp'ler) oluşturmak için merkezi olmayan bir platform sunarak akıllı sözleşmelerin yeteneklerini daha da genişletmiştir. Bu entegrasyon, finansal işlemlerde, tedarik zinciri yönetiminde ve daha fazlasındaki süreçleri kolaylaştırmıştır.

Bu entegrasyonun teknik önemi, güvenliği garanti eden ve tek bir arıza noktasını ortadan kaldıran blokzincirin merkezi olmayan doğasında yatmaktadır. Bu, akıllı sözleşmelerin deterministik ve otomatik olarak yürütülmesiyle birleştiğinde dolandırıcılık ve manipülasyon riskini önemli ölçüde azaltır. Blokzincir'in değişmezliği, bir akıllı sözleşme dağıtıldığında kodunun ve yürütme geçmişinin değişmeden kalmasını sağlar. Bu nitelikler, faaliyetlerinde şeffaflık, verimlilik ve güvenlik arayan sektörler için derin anlamlar taşımaktadır. Akıllı sözleşmeler ile blokzincir teknolojisinin yakınsaması, güvenilir, otomatik ve merkezi olmayan çözümlerle tanımlanan bir teknolojik manzarayı şekillendirmeye devam etmektedir.

Akıllı sözleşmelere ilişkin mevcut literatür, bireysel sözleşme işlevlerine odaklanılmasından kaynaklanan dikkate değer bir eksikliği ortaya koymaktadır. Mevcut söylemin çoğu, akıllı sözleşme yürütmenin temelleri ve bunların belirli alanlarda uygulanması üzerine yoğunlaşma eğilimindedir. Bu eksikliğin nedeni, akıllı sözleşmelerin daha karmaşık, çok adımlı süreçleri yürütmek için nasıl işbirliği yapabileceğinin sınırlı araştırılmasında yatmaktadır. Bu boşluk kısmen blokzincir teknolojisinin göreceli yeniliğinden ve merkezi olmayan uygulamaların gelişen doğasından kaynaklanmaktadır. Bu eksikliği gidermek için bu tez çalışmasında, blokzincir üzerinde dağıtılmış mevcut akıllı sözleşmeler ile etkileşime girebilen, İşbirlikçi Akıllı Sözleşmeler Modeli geliştirilmiştir. Çalışmada, işbirlikçi yönler derinlemesine incelenerek, merkezi olmayan bir çerçeve

içinde etkileşime giren birden fazla akıllı sözleşmenin sinerjik yeteneklerinin daha kapsamlı bir şekilde anlaşılmasına katkıda bulunulmuştur. Odaktaki bu değişim, yalnızca akıllı sözleşmeleri çevreleyen akademik söylemi geliştirmekle kalmayacak, aynı zamanda çeşitli sektörlerdeki pratik uygulamalar için değerli bilgiler sunacaktır.

Basit Akıllı Sözleşmelerin ve İşbirlikçi Akıllı Sözleşmelerin araştırılması, bunların operasyonel çerçeveleri ve etkinlikleri açısından keskin bir zıtlığı ortaya çıkarmıştır. Basit Akıllı Sözleşmeler, basit ve önceden belirlenmiş bir yapıya bağlı kalırken, İşbirlikçi Akıllı Sözleşmeler, dinamik ve işbirliğine dayalı bir yaklaşımı benimsemekte ve uyarlanabilirlik, esneklik ve ölçeklenebilirlik açısından önemli avantajlar sunmaktadır.

Basit Akıllı Sözleşmeler, sürekli değişen modern uygulamalar ortamına uyum sağlamakta zorlanmaktadır. Statik yapıları, karmaşık senaryolara uygulanabilirliklerini sınırlayarak onları eskimeye ve esnekliğe karşı duyarlı hale getirmektedir. Bu katılık, öngörülemeyen koşullara ve gelişen gereksinimlere yanıt verme yeteneklerini engeller ve potansiyel olarak verimsizliklere ve eski işlevselliklere yol açmaktadır.

Buna karşılık, İşbirlikçi Akıllı Sözleşmeler, esnekliği ve uyarlanabilirliği teşvik eden işbirlikçi unsurları birleştiren bir paradigma değişimini bünyesinde barındırmaktadır. Merkezi olmayan mimarileri, paydaşlar arasındaki işbirliğini kolaylaştırarak fikir birliği oluşturmayı ve ortak karar almayı mümkün kılmaktadır. Bu işbirlikçi doğa, İşbirlikçi Akıllı Sözleşmelerin kendilerini optimize etmelerini ve gelişen koşullara uyum sağlamalarını sağlayarak, sürekli alaka ve etkinliklerini garanti altına alır.

Dahası, İşbirlikçi Akıllı Sözleşmeler, kontrol ve sorumluluğu birden fazla katılımcı arasında dağıtarak gelişmiş dayanıklılık göstermektedir. Bu dağıtılmış yönetişim modeli, tek bir merkezi otoriteye olan bağımlılığı azaltarak tek hata noktalarının etkisini azaltır. Sonuç olarak, İşbirlikçi Akıllı Sözleşmeler, kesintiler ve düşmanca saldırılar karşısında daha fazla dayanıklılık sergileyerek geleneksel merkezi sistemleri felce uğratabilecek zorluklara dayanma yeteneği göstermektedir.

Ayrıca İşbirlikçi Akıllı Sözleşmeler, Basit Akıllı Sözleşmeleri sıklıkla rahatsız eden ölçeklenebilirlik sınırlamalarına da gidermektedir. İçsel modülerlikleri, performanstan ödün vermeden genişleyen bir kullanıcı tabanına ve artan işlem hacmine uyum

sağlamalarına olanak tanır. İşbirlikçi Akıllı Sözleşmelerin merkezi olmayan yapısı, hesaplama görevlerini birden fazla düğüme dağıtarak büyük ölçekli işlemleri verimli bir şekilde yürütmelerine olanak tanır. Bu ölçeklenebilirlik, İşbirlikçi Akıllı Sözleşmeler i yüksek verim ve giderek artan sayıda kullanıcı ve işlemi yönetebilme becerisi gerektiren uygulamalar için çok uygun hale getirir.

Sonuç olarak İşbirlikçi Akıllı Sözleşmeler, akıllı sözleşmeler alanında daha karmaşık ve etkili bir paradigma olarak ortaya çıkmaktadır. Uyum sağlama, işbirliği yapma ve ölçeklendirme yetenekleri, onları modern uygulamaların ve gelişen iş ortamlarının karmaşıklıklarına çözüm bulmak için iyi bir konuma getirmektedir. Dinamik ve dayanıklı akıllı sözleşme çözümlerine olan talep arttıkça İşbirlikçi Akıllı Sözleşmeler, dağıtılmış defter teknolojisinin geleceğini şekillendirmede giderek daha belirgin bir rol oynamaya aday görünmektedir.

Akıllı sözleşmelerin geleceği, İşbirlikçi Akıllı Sözleşmelerin örneklediği işbirlikçi ve uyarlanabilir yaklaşımın benimsenmesinde yatmaktadır. Kendi kendini optimize etme, değişen koşullara uyum sağlama ve kesintilere dayanma yetenekleri, onları sürekli gelişen dijital ortam için sağlam ve ölçeklenebilir çözümler oluşturmak için ideal temel olarak konumlandırmaktadır.

Gelecekte işbirlikçi akıllı sözleşmelerin gelişimi, blockchain teknolojisinin, programlama dillerinin ve standartlarının sürekli gelişimine bağlıdır. Daha karmaşık merkezi olmayan uygulamalara olan talep arttıkça, geliştiriciler muhtemelen akıllı sözleşmelerin kusursuz işbirliğini kolaylaştıran çerçeveler ve araçlar oluşturmaya odaklanacaktır. İşbirliğine dayalı işlevler için özel olarak tasarlanmış birlikte çalışabilirlik standartları ve gelişmiş programlama dilleri ortaya çıkabilir ve geliştiricilerin karmaşık, birbirine bağlı akıllı sözleşme ekosistemleri oluşturmasına olanak tanıyabilir.

İşbirlikçi akıllı sözleşmelerin ve yapay zekanın (AI) entegrasyonu, çeşitli endüstrilerde devrim yaratma potansiyeline sahiptir. Yapay zeka, akıllı sözleşmelerdeki karar verme süreçlerini geliştirerek değişen koşullara uyum sağlamalarını ve bunlara akıllıca yanıt vermelerini sağlayabilir. Örneğin, yapay zeka algoritmaları veri girişlerini analiz edebilir, eğilimleri tahmin edebilir ve işbirliğine dayalı akıllı sözleşmelerin gerçek zamanlı olarak yürütülmesini optimize edebilir. Bu sinerji, özellikle finans, tedarik zinciri yönetimi ve

sağlık hizmetleri gibi alanlarda daha uyarlanabilir ve verimli merkezi olmayan sistemlere yol açabilir.

İşbirliğine dayalı akıllı sözleşmeler ile yapay zekanın birleşimi, otonom ve kendi kendini geliştiren sistemlerin önünü açabilir. Akıllı sözleşmeler, işledikleri verilerden dinamik olarak öğrenebilir ve zaman içindeki performanslarını sürekli olarak optimize edebilir. Bu entegrasyon, yalnızca güvenli ve şeffaf değil, aynı zamanda akıllı ve uyarlanabilir karar alma yeteneğine sahip, merkezi olmayan uygulamalara yönelik yeni bir çağ başlatabilir. Ancak bu entegrasyonun tüm potansiyelinden sorumlu bir şekilde yararlanmak için veri gizliliği, güvenlik ve etik hususlarla ilgili zorlukların dikkatle ele alınması gerekir.

6. KAYNAKLAR

[1] T. Laurence, *Blockchain for dummies*. John Wiley & Sons, 2023.

[2] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized Business Review*, p. 21260, 2008.

[3] N. Szabo *et al.*, "Smart contracts," 1994.

[4] A. Sahai and R. Pandey, "Smart contract definition for land registry in blockchain," in *2020 IEEE 9th International conference on communication systems and network technologies (CSNT)*. IEEE, 2020, pp. 230–235.

[5] S. K. Panda, G. B. Mohammad, S. Nandan Mohanty, and S. Sahoo, "Smart contract-based land registry system to reduce frauds and time delay," *Security and Privacy*, vol. 4, no. 5, p. e172, 2021.

[6] S. Soner, R. Litoriya, and P. Pandey, "Exploring blockchain and smart contract technology for reliable and secure land registration and record management," *Wireless Personal Communications*, vol. 121, no. 4, pp. 2495–2509, 2021.

[7] D. Shinde, S. Padekar, S. Raut, A. Wasay, and S. Sambhare, "Land registry using blockchain-a survey of existing systems and proposing a feasible solution," in *2019 5th International Conference On Computing, Communication, Control And Automation (ICCUBEA)*. IEEE, 2019, pp. 1–6.

[8] P. Kumar, G. Dhanush, D. Srivatsa, A. Nithin, and S. Sahisnu, "A buyer and seller's protocol via utilization of smart contracts using blockchain technology," in *Advanced Informatics for Computing Research: Third International Conference, ICAICR 2019, Shimla, India, June 15–16, 2019, Revised Selected Papers, Part I 3*. Springer, 2019, pp. 464–474.

[9] S. M. Joshi and K. Rajeswari, "Efficient and accurate property title retrieval using ethereum blockchain," in *Sustainable Communication Networks and Application: ICSCN 2019*. Springer, 2020, pp. 424–438.

[10] J. Desic and K. Lenac, "Is blockchain technology the future of land registry digitalization?" *Zb. Prav. Fak. Sveuc. Rij.*, vol. 41, p. 609, 2020.

[11] R.-Y. Chen, "A traceability chain algorithm for artificial neural networks using t–s fuzzy cognitive maps in blockchain," *Future Generation Computer Systems*, vol. 80, pp. 198–210, 2018.

[12] A. Lei, Y. Cao, S. Bao, D. Li, P. Asuquo, H. Cruickshank, and Z. Sun, "A blockchain based certificate revocation scheme for vehicular communication systems," *Future Generation Computer Systems*, vol. 110, pp. 892–903, 2020.

[13] A. Dorri, S. S. Kanhere, and R. Jurdak, "Mof-bc: A memory optimized and flexible blockchain for large scale networks," *Future Generation Computer Systems*, vol. 92, pp. 357–373, 2019.

[14] L. Zhong, Q. Wu, J. Xie, J. Li, and B. Qin, "A secure versatile light payment system based on blockchain," *Future Generation Computer Systems*, vol. 93, pp. 327–337, 2019.

[15] R. Skowroński, "The open blockchain-aided multi-agent symbiotic cyber–physical systems," *Future Generation Computer Systems*, vol. 94, pp. 430–443, 2019.

[16] X. Xu, Q. Lu, Y. Liu, L. Zhu, H. Yao, and A. V. Vasilakos, "Designing blockchain-based applications a case study for imported product traceability," *Future Generation Computer Systems*, vol. 92, pp. 399–406, 2019.

[17] M. Yang, T. Zhu, K. Liang, W. Zhou, and R. H. Deng, "A blockchain-based location privacy-preserving crowdsensing system," *Future Generation Computer Systems*, vol. 94, pp. 408–418, 2019.

[18] W. Viriyasitavat and D. Hoonsopon, "Blockchain characteristics and consensus in modern business processes," *Journal of Industrial Information Integration*, vol. 13, pp. 32–39, 2019.

[19] Y. Lu, "The blockchain: State-of-the-art and research challenges," *Journal of Industrial Information Integration*, vol. 15, pp. 80–90, 2019.

[20] S. Cao, G. Zhang, P. Liu, X. Zhang, and F. Neri, "Cloud-assisted secure ehealth systems for tamper-proofing ehr via blockchain," *Information Sciences*, vol. 485, pp. 427–440, 2019.

[21] A. Dubovitskaya, Z. Xu, S. Ryu, M. Schumacher, and F. Wang, "Secure and trustable electronic medical records sharing using blockchain," in *AMIA annual symposium proceedings*, vol. 2017. American Medical Informatics Association, 2017, p. 650.

[22] R. Xu, X. Lin, Q. Dong, and Y. Chen, "Constructing trustworthy and safe communities on a blockchain-enabled social credits system," in *Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, 2018, pp. 449–453.

[23] M. Samaniego and R. Deters, "Internet of smart things-iost: Using blockchain and clips to make things autonomous," in *2017 IEEE international conference on cognitive computing (ICCC)*. IEEE, 2017, pp. 9–16.

[24] X. Wang, X. Zha, W. Ni, R. P. Liu, Y. J. Guo, X. Niu, and K. Zheng, "Survey on blockchain for internet of things," *Computer Communications*, vol. 136, pp. 10–29, 2019.

[25] A. H. Lone and R. N. Mir, "Forensic-chain: Blockchain based digital forensics chain of custody with poc in hyperledger composer," *Digital investigation*, vol. 28, pp. 44–55, 2019.

[26] C. Millard, "Blockchain and law: Incompatible codes?" *Computer Law & Security Review*, vol. 34, no. 4, pp. 843–846, 2018.

[27] T. Buocz, T. Ehrke-Rabel, E. Hödl, and I. Eisenberger, "Bitcoin and the gdpr: Allocating responsibility in distributed networks," *Computer Law & Security Review*, vol. 35, no. 2, pp. 182–198, 2019.

[28] M. M. Queiroz and S. F. Wamba, "Blockchain adoption challenges in supply chain: An empirical investigation of the main drivers in india and the usa," *International Journal of Information Management*, vol. 46, pp. 70–82, 2019.

[29] F. Casino, T. K. Dasaklis, and C. Patsakis, "A systematic literature review of blockchain-based applications: Current status, classification and open issues," *Telematics and informatics*, vol. 36, pp. 55–81, 2019.

[30] A. H. Mohsin, A. Zaidan, B. Zaidan, O. S. Albahri, A. S. Albahri, M. Alsalem, and K. Mohammed, "Blockchain authentication of network applications: Taxonomy, classification, capabilities, open challenges, motivations, recommendations and future directions," *Computer Standards & Interfaces*, vol. 64, pp. 41–60, 2019.

[31] J. Li, D. Greenwood, and M. Kassem, "Blockchain in the built environment and construction industry: A systematic review, conceptual models and practical use cases," *Automation in construction*, vol. 102, pp. 288–307, 2019.

[32] H. Tang, Y. Shi, and P. Dong, "Public blockchain evaluation using entropy and topsis," *Expert Systems with Applications*, vol. 117, pp. 204–210, 2019.

[33] W. D. Du, S. L. Pan, D. E. Leidner, and W. Ying, "Affordances, experimentation and actualization of fintech: A blockchain implementation study," *The Journal of Strategic Information Systems*, vol. 28, no. 1, pp. 50–65, 2019.

[34] A. Ahl, M. Yarime, K. Tanaka, and D. Sagawa, "Review of blockchain-based distributed energy: Implications for institutional development," *Renewable and Sustainable Energy Reviews*, vol. 107, pp. 200–211, 2019.

[35] R. Casado-Vara, P. Chamoso, F. De la Prieta, J. Prieto, and J. M. Corchado, "Non-linear adaptive closed-loop control system for improved efficiency in iot-blockchain management," *Information Fusion*, vol. 49, pp. 227–239, 2019.

[36] I. García-Magariño, R. Lacuesta, M. Rajarajan, and J. Lloret, "Security in networks of unmanned aerial vehicles for surveillance with an agent-based approach inspired by the principles of blockchain," *Ad Hoc Networks*, vol. 86, pp. 72–82, 2019.

[37] L. Zhang, M. Luo, J. Li, M. H. Au, K.-K. R. Choo, T. Chen, and S. Tian, "Blockchain based secure data sharing system for internet of vehicles: A position paper," *Vehicular Communications*, vol. 16, pp. 85–93, 2019.

[38] S. G. Alonso, J. Arambarri, M. López-Coronado, and I. de la Torre Díez, "Proposing new blockchain challenges in ehealth," *Journal of medical systems*, vol. 43, no. 3, p. 64, 2019.

[39] S.-W. Noh, Y. Park, C. Sur, S.-U. Shin, and K.-H. Rhee, "Blockchain-based user-centric records management system," *Int J Control Autom*, vol. 10, no. 11, pp. 133–144, 2017.

[40] "Hyperledger blockchain performance metrics," 2020, [Online]. Available: https://www.hyperledger.org/learn/publications/blockchain-performance-metrics (current November 2020).

[41] P. Hristov and W. Dimitrov, "The blockchain as a backbone of gdpr compliant frameworks," *Calitatea*, vol. 20, no. S1, p. 305, 2019.

[42] Z. Zheng, S. Xie, H.-N. Dai, W. Chen, X. Chen, J. Weng, and M. Imran, "An overview on smart contracts: Challenges, advances and platforms," *Future Generation Computer Systems*, vol. 105, pp. 475–491, 2020.

[43] H. Wang, C. Guo, and S. Cheng, "Loc—a new financial loan management system based on smart contracts," *Future Generation Computer Systems*, vol. 100, pp. 648–655, 2019.

[44] Z. Abou El Houda, A. S. Hafid, and L. Khoukhi, "Cochain-sc: An intra-and inter-domain ddos mitigation scheme based on blockchain using sdn and smart contract," *IEEE Access*, vol. 7, pp. 98 893–98 907, 2019.

[45] Y. Huang, Y. Bian, R. Li, J. L. Zhao, and P. Shi, "Smart contract security: A software lifecycle perspective," *IEEE Access*, vol. 7, pp. 150 184–150 202, 2019.

[46] J.-W. Liao, T.-T. Tsai, C.-K. He, and C.-W. Tien, "Soliaudit: Smart contract vulnerability assessment based on machine learning and fuzz testing," in *2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*. IEEE, 2019, pp. 458–465.

[47] P. Momeni, Y. Wang, and R. Samavi, "Machine learning model for smart contracts security analysis," in *2019 17th International Conference on Privacy, Security and Trust (PST)*. IEEE, 2019, pp. 1–6.

[48] J. Moubarak, M. Chamoun, and E. Filiol, "On distributed ledgers security and illegal uses," *Future Generation Computer Systems*, vol. 113, pp. 183–195, 2020.

[49] P. Qian, Z. Liu, Q. He, R. Zimmermann, and X. Wang, "Towards automated reentrancy detection for smart contracts based on sequential models," *IEEE Access*, vol. 8, pp. 19 685–19 695, 2020.

[50] K. R. Özyilmaz, M. Doğan, and A. Yurdakul, "Idmob: Iot data marketplace on blockchain," in *2018 crypto valley conference on blockchain technology (CVCBT)*. IEEE, 2018, pp. 11–19.

[51] Z. Li, H. Guo, W. M. Wang, Y. Guan, A. V. Barenji, G. Q. Huang, K. S. McFall, and X. Chen, "A blockchain and automl approach for open and automated customer service," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3642–3651, 2019.

[52] Y. Jiang, Y. Zhong, and X. Ge, "Smart contract-based data commodity transactions for industrial internet of things," *IEEE Access*, vol. 7, pp. 180 856–180 866, 2019.

[53] X. Liu, K. Muhammad, J. Lloret, Y.-W. Chen, and S.-M. Yuan, "Elastic and cost-effective data carrier architecture for smart contract in blockchain," *Future Generation Computer Systems*, vol. 100, pp. 590–599, 2019.

[54] K. Fan, Z. Bao, M. Liu, A. V. Vasilakos, and W. Shi, "Dredas: Decentralized, reliable and efficient remote outsourced data auditing scheme with blockchain smart contract for industrial iot," *Future Generation Computer Systems*, vol. 110, pp. 665–674, 2020.

[55] A. Carvalho, "Bringing transparency and trustworthiness to loot boxes with blockchain and smart contracts," *Decision Support Systems*, p. 113508, 2021.

[56] S.-M. Lee, S. Park, and Y. B. Park, "Formal specification technique in smart contract verification," in *2019 International Conference on Platform Technology and Service (PlatCon)*. IEEE, 2019, pp. 1–4.

[57] S. J. Pee, E. S. Kang, J. G. Song, and J. W. Jang, "Blockchain based smart energy trading platform using smart contract," in *2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*. IEEE, 2019, pp. 322–325.

[58] H. R. Hasan and K. Salah, "Combating deepfake videos using blockchain and smart contracts," *Ieee Access*, vol. 7, pp. 41 596–41 606, 2019.

[59] O. Choudhury, N. Rudolph, I. Sylla, N. Fairoza, and A. Das, "Auto-generation of smart contracts from domain-specific ontologies and semantic rules," in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. IEEE, 2018, pp. 963–970.

[60] H. Desai, K. Liu, M. Kantarcioglu, and L. Kagal, "Adjudicating violations in data sharing agreements using smart contracts," in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. IEEE, 2018, pp. 1553–1560.

[61] Y. Kim, D. Pak, and J. Lee, "Scanat: identification of bytecode-only smart contracts with multiple attribute tags," *IEEE Access*, vol. 7, pp. 98 669–98 683, 2019.

[62] W. Xiong and L. Xiong, "Smart contract based data trading mode using blockchain and machine learning," *IEEE Access*, vol. 7, pp. 102 331–102 344, 2019.

[63] S. Zhang and J.-H. Lee, "Smart contract-based secure model for miner registration and block validation," *IEEE Access*, vol. 7, pp. 132 087–132 094, 2019.

[64] W. Zheng, Z. Zheng, X. Chen, K. Dai, P. Li, and R. Chen, "Nutbaas: A blockchain-as-a-service platform," *Ieee Access*, vol. 7, pp. 134 422–134 433, 2019.

[65] B. Yu, P. Zhan, M. Lei, F. Zhou, and P. Wang, "Food quality monitoring system based on smart contracts and evaluation models," *IEEE Access*, vol. 8, pp. 12 479–12 490, 2020.

[66] S. Wang, C. Huang, J. Li, Y. Yuan, and F.-Y. Wang, "Decentralized construction of knowledge graphs for deep recommender systems based on blockchain-powered smart contracts," *IEEE Access*, vol. 7, pp. 136 951–136 961, 2019.

[67] H. Wang, H. Qin, M. Zhao, X. Wei, H. Shen, and W. Susilo, "Blockchain-based fair payment smart contract for public cloud storage auditing," *Information Sciences*, vol. 519, pp. 348–362, 2020.

[68] S. E. Chang, Y.-C. Chen, and M.-F. Lu, "Supply chain re-engineering using blockchain technology: A case of smart contract based tracking process," *Technological Forecasting and Social Change*, vol. 144, pp. 1–11, 2019.

[69] N. Mhaisen, N. Fetais, and A. Massoud, "Secure smart contract-enabled control of battery energy storage systems against cyber-attacks," *Alexandria Engineering Journal*, vol. 58, no. 4, pp. 1291–1300, 2019.

[70] A. Singh, R. M. Parizi, Q. Zhang, K.-K. R. Choo, and A. Dehghantanha, "Blockchain smart contracts formalization: Approaches and challenges to address vulnerabilities," *Computers & Security*, vol. 88, p. 101654, 2020.

[71] M. R. Dorsala, V. Sastry, and S. Chapram, "Fair payments for verifiable cloud services using smart contracts," *Computers & Security*, vol. 90, p. 101712, 2020.

[72] D. Unal, M. Hammoudeh, and M. S. Kiraz, "Policy specification and verification for blockchain and smart contracts in 5g networks," *ICT Express*, vol. 6, no. 1, pp. 43–47, 2020.

[73] T. de Graaf, "From old to new: From internet to smart contracts and from people to smart contracts," *Computer Law & Security Review*, vol. 35, no. 5, p. 105322, 2019.

[74] Y. Li, W. Yang, P. He, C. Chen, and X. Wang, "Design and management of a distributed hybrid energy system through smart contract and blockchain," *Applied Energy*, vol. 248, pp. 390–405, 2019.

[75] Y. S. KIYAK, Ö. COŞKUN, and I. İ. BUDAKOĞLU, "Blokzinciri, akilli kontratlar ve sağlik alanindaki üç uygulama örneği," *Hacettepe Sağlık İdaresi Dergisi*, vol. 22, no. 2, pp. 457–466.

[76] N. Staifi and M. Belguidoum, "Adapted smart home services based on smart contracts and service level agreements," *Concurrency and Computation: Practice and Experience*, p. e6208, 2021.

[77] Y. Chu, J. Ream, and D. S. C. Insights, "Getting smart about smart contracts," 2016.

[78] H. Hasan, E. AlHadhrami, A. AlDhaheri, K. Salah, and R. Jayaraman, "Smart contract-based approach for efficient shipment management," *Computers & Industrial Engineering*, vol. 136, pp. 149–159, 2019.

[79] D. Macrinici, C. Cartofeanu, and S. Gao, "Smart contract applications within blockchain technology: A systematic mapping study," *Telematics and Informatics*, vol. 35, no. 8, pp. 2337–2354, 2018.

[80] Z. Wang, W. Dai, K.-K. R. Choo, H. Jin, and D. Zou, "Fsfc: An input filter-based secure framework for smart contract," *Journal of Network and Computer Applications*, vol. 154, p. 102530, 2020.

[81] Metamask. (2023, Jul.) Track, manage, buy, swap, bridge, and stake your web3 assets in one place. [Online]. Available: `https://metamask.io/`

[82] Nodejs. (2023, Jul.) Open-source, cross-platform javascript runtime environment. [Online]. Available: `https://nodejs.org/`

[83] T. Suite. (2023, Jul.). [Online]. Available: `https://www.trufflesuite.com/docs/ganache/overview`

[84] R. IDE. (2023, Jul.). [Online]. Available: `https://remix.ethereum.org/`

[85] Cryptozombies. (2023, Jul.) Learn to code blockchain dapps by building simple games. [Online]. Available: `https://cryptozombies.io/`

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı : Tunahan TİMUÇİN

Yabancı Dili : İngilizce

ÖĞRENİM DURUMU

Derece	**Alan**	**Okul/Üniversite**	**Mezuniyet Yılı**
Doktora	Bilgisayar Mühendisliği	Düzce Üniversitesi	2024
Y. Lisans	Bilgisayar Mühendisliği	Düzce Üniversitesi	2018
Lisans	Bilgisayar Mühendisliği	Ankara Üniversitesi	2015
Lise	Fen Bilimleri	Yusuf Baysal And. Lis.	2009

YAYINLAR

T. Timuçin and S. Biroğul, "Collaborative Smart Contracts (CoSC): example of real estate purchase and sale (s)," *The Journal of Supercomputing,* pp. 1-20, 2023.

T. Timuçin and S. Biroğul, "A survey: Making "Smart Contracts" really smart," *Transactions on Emerging Telecommunications Technologies,* vol.32(11), e4338,2021.

T. Timuçin and S. Biroğul, "The Evolution of Smart Contract Platforms: A look at Current Trends and Future Directions," *Mugla Journal of Science and Technology,* vol.9(2), pp.46-55, 2023