TOBB EKONOMİ VE TEKNOLOJİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BLOKZİNCİR TABANLI GİZLİLİK KORUMALI İSTATİSTİKSEL HESAPLAMALAR

YÜKSEK LİSANS TEZİ

Zeynep Delal MUTLU

Bilgisayar Mühendisliği Anabilim Dalı

Tez Danışmanı: Prof. Dr. Ali Aydın SELÇUK

NİSAN 2023

ÖZET

Yüksek Lisans Tezi

BLOKZİNCİR TABANLI GİZLİLİK KORUMALI

İSTATİSTİKSEL HESAPLAMALAR

Zeynep Delal MUTLU

TOBB Ekonomi ve Teknoloji Üniveritesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Prof. Dr. Ali Aydın SELÇUK

Tarih: Nisan 2023

Bu çalışmada, üçüncü taraflarca sağlanan gizli veriler üzerinde istatistiksel hesaplamalara izin vermek için akıllı sözleşmeler ve homomorfik kriptografi kullanan blokzincir tabanlı bir uygulama mimarisi önerilmektedir. Blokzincir kullanımı, bütünlük ve hata toleransı gibi güvenlik özelliklerini sağlarken; homomorfik şifreleme ise verilerin gizliliğini korur. Bu temel güvenlik prensiplerinin dışında, uygulama mimarisi geliştirilirken veri aktarımı ve veri depolama aşamalarında ayrıca güvenlik sağlayabilecek hassasiyette yaklaşımlar kurgulanmıştır. Yapılan literatür taramasında, bu tez kapsamındaki konulara benzer konulara rastlanmışsa da; bu tez çalışması Ethereum akıllı sözleşmesinde doğrusal regresyon analizi yapması açısından diğer çalışmalardan ayrışmaktadır. Çalışma kapsamında Ethereum blokzincirine akıllı sözleşme geliştirilmiş, bu akıllı sözleşmeye veri gönderen ve verileri okuyan kullanıcılar için ayrı ayrı kullanıcı betikleri yazılmıştır. Çalışmanın örnek senaryosu doğrusal regresyon analizi üzerinden geliştirilmiştir. Doğrusal regresyon analizi ile üçüncü taraflardan toplanan veriler şifrelenerek akıllı sözleşmeye şifreli bir biçimde gönderilir. Akıllı sözleşmeye gelen şifreli veriler ilgili değişkenlerde homomorfik toplama işlemi ile toplanarak kaydedilir. Daha sonra akıllı sözleşmeden şifreli verileri okumak isteyen kullanıcı bu değerleri okur ve işlemlerini gerçekleştirir. Çalışmanın

ilerleyen bölümlerinde önerilen sistemin tasarımı, uygulaması ve farklı homomorfik anahtar uzunluklarına dayalı sonuçları sunulmaktadır. Uygulamanın sayısal sonuçları, akıllı sözleşmeye gönderilen farklı anahtar uzunluğuna sahip işlemlerin ne kadar zaman ve gaz ücreti tükettiğini göstermektedir. Farklı anahtar uzunluğundaki verilerin gaz ücreti ve zaman tüketimlerindeki benzerlikler, farklılıklar ve bunların sebepleri tartışılmıştır. Ayrıca değerlendirmeler kısmında, önerilen sistemin getirdiği güvenlik iyileştirmelerinin neler olduğuna da ışık tutulmaktadır. Önerilen mimari, tüm istatistiksel hesaplamalarda kullanılabilecek genel bir bakış açısı kapsamında geliştirilmiştir. İstenildiği takdirde uygulama adımları tekrarlanarak farklı istatistiksel hesaplamalar için de sistem tasarımı yapılabilir. Çalışma, akıllı bir sözleşme aracılığıyla homomorfik hesaplamalara sahip blokzincir tabanlı veri paylaşım mekanizmasının uygulanabilir olduğunu ve verilerin yetkisiz kullanıcılardan korunmasında iyileştirmeler yapılabildiğini göstermektedir.

Anahtar Kelimeler: Blokzincir, Ethereum, Doğrusal regresyon, Homomorfik kriptoloji.

ABSTRACT

Master of Science

BLOCKCHAIN BASED PRIVACY PRESERVING

STATISTICAL CALCULATIONS

Zeynep Delal MUTLU

TOBB University of Economics and Technology
Institute of Natural and Applied Sciences
Computer Engineering

Supervisor: Prof. Dr. Ali Aydın SELÇUK

Date: April 2023

In the present study, a blockchain-based application architecture is proposed that uses smart contracts and homomorphic cryptography to allow statistical calculations on confidential data provided by third parties. Blockchain systems provide security features such as integrity and fault tolerance, while homomorphic encryption preserves data confidentiality. In addition to these basic security principles of blockchain, approaches that can improve security and safety during data transfer and storage were also designed. Although similar topics have been encountered in the literature, this thesis differs from other studies since it performs linear regression analysis in an Ethereum smart contract. In the scope of the study, one smart contract was developed for the Ethereum blockchain, and user scripts were written to send data to the smart contract and read data from it. The sample scenario of this study was developed through linear regression analysis. Encrypted data are collected from third parties and sent to the smart contract through linear regression analysis. Encrypted data received by the smart contract was recorded by homomorphically adding them to the relevant variables. Later, a user who carries authority to read encrypted data from the smart contract reads these values and performs their calculations.In the later sections of the study, the design, implementation, and results based on different homomorphic

encryption key-lengths of the proposed system are presented. The numerical results of the application show how much time and gas fees are consumed by transactions with different key-lengths sent to the smart contract. Similarities and differences in the gas fee and time consumption of data with varied key lengths and their reasons are examined. In addition, in the evaluation section, light is shed on the security improvements brought about by the proposed system. The proposed architecture was developed within a general perspective which is applicable to all other statistical calculations. If desired, the application steps can be repeated to be applied to different statistical calculations. The study demonstrates that a blockchain-based data-sharing mechanism with homomorphic calculations can be implemented through a smart contract, and that improvements can be made in protecting data from unauthorized users.

Keywords: Blockchain, Ethereum, Linear Regression, Homomorphic cryptography.

TEŞEKKÜR

Çalışmalarım esnasında destekleri ve fikirleri ile yol gösteren saygıdeğer tez eş danışmanım Yeşem Kurt Peker'e ve tez danışmanım Ali Aydın Selçuk'a; çalışmalarımı rahat bir ortamda yapmam için emek veren ve hayatım boyunca desteklerini esirgemeyen sevgili aileme teşekkürlerimi bir borç bilirim.

İÇİNDEKİLER

Sayfa

ŞEKİL LİSTESİ

Sayfa

ÇİZELGE LİSTESİ

Sayfa

KISALTMALAR

ABI **:** Uygulama İkili Arayüzü **(**Application Binary Interface)
BGN **:** Boneh-Goh-Nissim
FHE **:** Tamamen Homomorfik Şifreleme (Fully Homomorphic Encryption)
IoT **:** Nesnelerin Interneti (Internet of Things)
P2P **:** Eşler arası (Peer to Peer)
PHE **:** Kısmen Homomorfik Şifreleme (Partially Homomorphic Encryption)
PoS **:** Proof-of-Stake
PoW **:** Proof-of-Work
SWHE **:** Biraz Homomorfik Şifreleme (Somewhat Homomorphic Encryption)

SEMBOL LİSTESİ

Bu çalışmada kullanılmış olan simgeler açıklamaları ile birlikte aşağıda sunulmuştur.

Simgeler	**Açıklama**
p	Rassal asal sayı
q	Rassal asal sayı
n	Paillier açık anahtarı
g	Paillier açık anahtarı
λ	Paillier gizli anahtarı
μ	Paillier gizli anahtarı
m	Açık metin
c	Şifreli metin
x	Veri noktası
y	Veri noktası
n(N)	Veri sayısı
xy	Üretilen veri
x^2	Üretilen veri
E	Şifreleme
D	Şifre Çözme

RESİM LİSTESİ

Sayfa

1. GİRİŞ

Günümüz dünyasının teknoloji açısından hızlı ilerlemesi, teknolojik araçların ve çözümlerinin de hızlı bir şekilde gelişmesine hatta değişmesine yol açmaktadır. Dolayısıyla gelişen teknoloji dünyasında kullanılan veri sayısı ve veri iletişimi de paralel bir şekilde artmış ve değişime uğramıştır. Blokzincir teknolojisi ise bu veri iletişimi kapsamında çığır açan buluşlardan biri olarak değerlendirilmektedir.

Blockzincir, kriptografik özet fonksiyon (hash function) değerleri ile birbirine zincirlenmiş bloklardan oluşan ve merkezi olmayan bir dijital defterdir. İlk olarak 2008 yılında, Bitcoin [1] için bir işlem defteri olarak tanıtılan blokzincir teknolojisi, yalnızca finansta değil aynı zamanda sağlık, tedarik yönetimi, nesnelerin interneti (IoT) ve diğer alanlarda da uygulama alanı bulmuştur. Kurcalamaya karşı direnç (tamper resistance) ve hata toleransı (fault tolerance), blokzincirinde akıllı sözleşmeler sayesinde kod yürütme yeteneği ile birleştiğinde, blok zinciri teknolojisi, bütünlük (integrity), yüksek kullanılabilirlik (availability) ve veri işlemlerinin denetlenebilirliğini (auditability) gerektiren uygulamalar için çok çekici bir çözüm haline gelmiştir [2].

Blokzinciri teknoloji yapısına dahil olmayan bir güvenlik özelliği gizliliktir (confidentiality) [3]. Dışarıdan veya akıllı sözleşmenin faaliyeti olarak blokzincirine gönderilen veriler, herkesin görüntülemesine açıktır. Şifrelenmedikçe, blokzincirinde paylaşılan verilerin gizlilik gereksinimleri karşılanamaz. Bu bilgiden hareketle, blokzincirinde verilerin gizliliğini konu edinen bir çalışma alanı oluşmak durumunda kalmıştır.

Bu çalışmanın da konusu olan; blokzincir üzerindeki verilerin gizliliği için kullanılması önerilen şifreleme yöntemlerinden biri de homomorfik şifrelemedir [4]. Homomorfik şifreleme, şifrelenmiş veriler (şifreli metin) üzerinde şifresini çözmeye gerek kalmadan hesaplamaların yapılmasına izin verir. Bu tez kapsamında, üçüncü şahıslar (third party) tarafından şifrelenerek blokzincirine gönderilen veriler üzerinde istatistiksel hesaplamalar yapmak için homomorfik şifreleme kullanan blokzincir tabanlı veri paylaşım prototipi önerilmektedir. Önerilen sistemin homomorfik

şifreleme kullanmayan sistemlere göre en büyük avantaji veri gizliliğinin sadece veri iletimi ve veri depolaması fazlarında değil aynı zamanda verilerin işlenmesi sırasında da korunmasıdır.

Çalışmaya bir bağlam vermek için, dağıtılmış uç cihazların sensörlerden veri topladığı bir senaryo varsayılmaktadır. Bunun için seçilen senaryoda, şehirlerin farklı noktalarına yerleştirilen sensörlerden toplanan verilerle, hava kirliliği ile trafikteki araç sayısının arasındaki ilişkinin boyutlarının ölçülmesi amaçlanmıştır. Projenin Ethereum ağında geliştirilmesi sayesinde, bu deneye global ölçekte katkıda bulunmak mümkündür. Ölçüm yapılmak istenen yerlere sensörlerin yerleştirilmesi ve Ethereum ağına göndermek için belli yazılımlar kullanılması yeterlidir. Eğer özel bir blokzincir ağı kullanılmış olsaydı, bu büyüklükteki bir projede blokzincir konfigürasyonları için ciddi zaman ve emek harcanması gerekebilirdi. Ethereum blokzincirinin sağladığı en büyük kolaylık açık (public) bir ağ olmasıdır.

Kısacası bu çalışmada; blokzincirnin sağlamış olduğu güvenlik önlemlerinden yararlanılmış, gloabal bir blokzinciri kullanılması sayesinde olabildiğince fazla noktadan veri almasına imkan verilmiş, blokzincirinin güvenlik anlamında yetersiz kaldığı veri gizliliği problemi homomorfik şifreleme ile çözülmüş ve ayrıca uygulama geliştirme aşamasında farklı yaklaşımlar geliştirilerek veri korunması maksimum seviyeye çıkarılmıştır.

Tez çalışmasının bundan sonraki kısımları şu şekilde organize edilmiştir: Bölüm 1.2'de, blokzincirinde gizliliği koruyan homomorfik şifreleme ile veri paylaşımı hakkındaki literatür gözden geçirilmiştir. Bölüm 2'de, önerilen sistemde kullanılan teknolojilerle ilgili teknik bakımdan detaylı bir arka plan sunulmuştur. Bölüm 3'te, önerilen sistem, kullanılan araçlar ve uygulamaların detayları ayrıntılı olarak açıklanmaktadır. Bölüm 4’te deney ve deneylerin sonuçları paylaşılmaktadır. Değerlendirmeler bölümünde tüm bu deneylerden elde edilen numerik sonuçlar tartışılmaktadır. Çalışmanın sonucu ve ileriki çalışmalara yol göstermesi açısından bu çalışmayı iyileştirme yolları Sonuç ve Öneriler bölümünde yer almaktadır.

1.1 TezinAmacı

Bu çalışmadaki katkı, verilerin işlenmesi sırasında gizlilik sunan bir prototip sunmanın yanında; doğrusal regresyon analizinin açık bir ağ olan Ethereum blokzinciri üzerinde

gerçekleştiriliyor olmasıdır. Çalışmanın farklı anahtar uzunlukları ile uygulanabilirliğini incelemek ve uygulama kapsamında eklenen güvenlik yaklaşımları ile de çalışmaya farklı bir derinlik katmak amaçlanmıştır.

1.2 Literatür Araştırması

Blokzincir teknolojisi üzerinden gizliliği koruyan veri analizi, son yıllarda birçok makalenin çalışma konusu olmaktadır. Araştırmada [5], akıllı sözleşmelerde homomorfik işlemlerin uygulanmasına ilişkin teorik bir bakış açısı geliştirilmiş ve örnek bir mimari sunulmuştur. Bu teorik çalışma, blokzincir sistemlerinde farklı homomorfik kriptosistemlerin uygulanabileceğini ortaya koymaktadır.

Farklı uç birimlerden toplanan büyük miktarda verinin işlenmesi gerektiğinde blockchain sistemlerini kullanan çalışmalar bulunmaktadır. Bu çalışmalardan biri olan makale [6], blokzincir sistemlerinde makine öğrenmesi algoritması hesaplamalarının nasıl yapılabileceğine dair bir bakış açısı sunmaktadır.

IoT uygulamaları için de blokzincir temelinde benzer çalışmalar yapılmıştır. Referans [7], IoT verilerini homomorfik şifreleme ile şifreleyen ve blok zincirine gönderen makalelerden biridir. Bu çalışma, IoT verilerinin homomorfik toplanması üzerine bir çalışmadır. Sistem çapında veri toplama yöntemini de şematize eden bu çalışma, projeyi özel bir Ethereum blok zincirinde uygulamaya geçirmiştir. Bu tezin genel kapsamıyla paralellik gösteren bir çalışmadır. Doğrusal regresyon analizi yapmaması ve uygulamadaki farklılıklar açısından bu tez kapsamından ayrılmaktadır.

Blockchain, veri bütünlüğünü koruma ilkesi nedeniyle elektronik oylama için uygun bir sistem olarak kabul edilmektedir. Ancak blockchain sistemlerinde verilerin mahremiyeti korunmaz. Referans [8], homomorfik şifreleme ile verilerin gizliliğini koruyarak oyların sayılmasını önermektedir.

Çalışma [9]'da, Hyperledger Fabric blockchain sisteminde Paillier homomorfik kriptosistemini kullanarak özel sağlık verilerinin sayısını, ortalamasını, değişkenini ve çarpıklığını güvenli bir şekilde hesaplamak için bir sistem önerilmiştir. Bu sistem, yalnızca konsorsiyumun parçası olan taraflara izin verilen bir konsorsiyum blokzincirinde uygulanmıştır. Yazarlar, blok zincirine yönelik kullanıcı istekleri için bir REST uygulaması geliştirmiştir. Deneysel sonuçlarına göre, istekler makul bir süre

içinde yapılır ve güvenlik ihtiyaçları karşılanır, bu nedenle gerçek hayattaki uygulamalarda uygulanabilir sonucu çıkarmışlardır.

Makale [10]'da güvenli çok taraflı hesaplama(Secure Multiparty Computation) ile çok sayıda veri kullanarak lineer regresyon hesaplaması yapmıştır. Güvenli çok taraflı hesaplama, katılımcılar arasında şifreli veri alışverişine imkan sağlar. Aynı zamanda önerdiği devre ile şifreli veriler üzerinde tıpkı homomorfik şifrelemede olduğu gibi hesaplar yapılmasına da imkan tanır. Ancak uygulama alanının Ethereum ağı kadar geniş bir alanı hedeflenmiyor olması, bu tez çalışmasından bu sistemler yerine açık ağ olan Ethereum araştırmasına odaklanılmasına neden olmuştur.

Makale [11] blokzincir ağlarında tamamen homomorfik şifreleme (Fully Homomorphic Encryption - FHE) kullanılarak blokzincir ağlarında sağlık verilerinin aktarımı ve korunmasıyla ilgili bir mimari model sunulmuştur. Bu modelde ayrıntılı olarak veri paylaşımının hangi sistemler üzerinde aktarılacağı açıklanmıştır; ancak kullanılan algoritmaların veya blokzincirin adı adı net bir şekilde belirtilmemiştir. Bu makale, blokzincir sistemlerinde tamamen homomorif şifrelemenin nasıl olabileceği ile ilgili geniş bir bakış açısı sunmaktadır.

IoT cihazları için yapılmış çalışmalardan biri de makale [12] dir. Bu makalede IoT cihazlarından blokzincrine gönderilen verilere yapılabilecek manipülasyonları engellemek üzerine bir araştırma yürütlmüştür. Makalede herhangi bir spesifik blokzinciri türü kullanılmamış, Paillier homomorfik şifreleme kullanılarak mimarinin nasıl yapılması gerektiğiyle ilgili bir bakış açısı sunulmuştur.

Makale [13]'te, bu tez çalışmasının da konusu olan ve klasik bir makine öğrenmesi algoritması olarak anılan doğrusal regresyon analizi üzerinde çalışma yapılmıştır. Söz konusu makalenin bu tez çalışmasıyla paralellik gösteren noktalarından biri ise Ethereum üzerinde geliştirilen bir uygulama mimarisinin öneriliyor olmasıdır. Ancak bu makalede veri gizliliği homomorfik şifreleme ile sağlanmak yerine, yazarların kendi önerisi olan bir veri karmaşıklaştırma (obfuscation) yöntemi ile önerilmektedir. Verilerin Ethereum ağıyla entegre çalışan bulut platformlarına gönderildiği bir senaryoya sahip olan bu makalede iki farklı gizli anahtar kullanılarak bir mimari kurulmuştur. Makalenin sonuç bölümünde ise bu yapının tercih edilmesinin sebebinin tamamen homomorfik şifrelemenin işlem verimi bakımından düşük olabileceği endişesidir. Bu tez çalışması ise tamamen homomorfik şifreleme yerine kısmen

homomorifk şifrelemeye odaklanmış ve söz konusu makalenin işlem gücü probleminin olabildiğince dışında kalmayı başarmıştır.

Makale [14], [15] ve [16] homomorfik kriptosistem kullanarak doğrusal regresyon analizi yapan diğer makalelerdir. Bu makalelerdeki genel yaklaşım sunuculara gönderilen verilerin gizliliği konusundaki endişeye dayanmaktadır. Doğrusal regresyon analizinin sunucu tabanlı korunması için önerilerde bulunan bu makalelerde herhangi bir blokzincir mimarisi kullanılmamış, blokzincirin sağladığı temel güvenlik prensiplerinden yararlanılmamıştır. Yine de doğrusal regresyon analizi için gerekli olan yaklaşımlarını ve yöntemlerini dile getirmişlerdir.

Bu tez çalışması, makale [9]'dan esinlenerek bir fikir ve çalışma konusu haline getirilmiştir. Tez çalışması için, daha geniş bir kullanıcı yelpazesinin sisteme erişmesine izin vermek için halka açık bir blokzincir ağı olan Ethereum tercih edilmiştir. Ethereum ağı için akıllı sözleşme ve istemci uygulamaları geliştirmek hedeflenmiştir. Makale [9]'dan farklı olarak doğrusal regresyon analizi çalışılmıştır. Elde edilen sonuçlar, veri paylaşımı ve analizi için gereken süre ve maliyet açısından pratik uygulamalar için umut vericidir.

Bu literatür araştırması, doğrusal regresyon istatistiksel hesaplamasının daha önce Ethereum blok zincirinde homomorfik şifreleme kullanılarak hesaplanan bir yöntem olmadığını gösteriyor. Bu nedenle bu tez çalışması, blokzincir sistemlerinde her ne kadar homomorfik kriptosistemler çalışılmışsa da doğrusal regresyon hesaplaması açısından tüm çalışmalardan farklılık göstermektedir. İlgili çalışmalar homomorfik kriptosistemler içermesine rağmen birçok makalede public blokzincir sistemleri yerine özel veya konsorsiyum sistemleri kullanılmaktadır. Bu nedenle bu çalışma, genel (public) Ethereum ağı ve doğrusal regresyon hesaplaması açısından da önceki çalışmalardan farklılık göstermektedir.

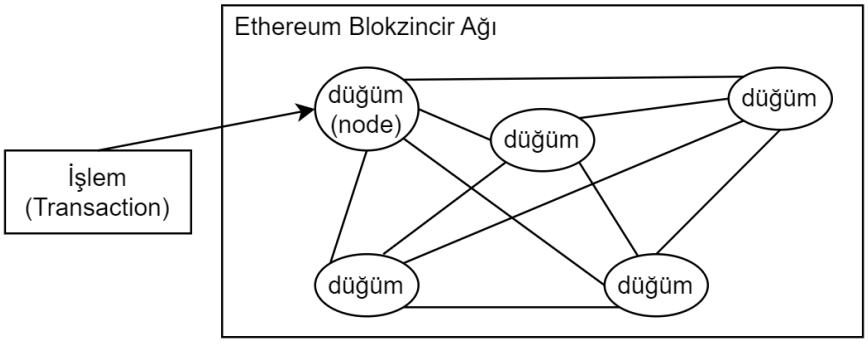
2. ARKA PLAN BİLGİSİ

Bu bölümde, çalışmanın konusu olan Ethereum blokzincir ağı ve homomorfik şifreleme yöntemleri temel alınmış; uygulama olarak ise lineer regresyon istatistiksel hesaplaması incelenmiştir.

2.1 Ethereum Blokzinciri ve Akıllı Sözleşmeler

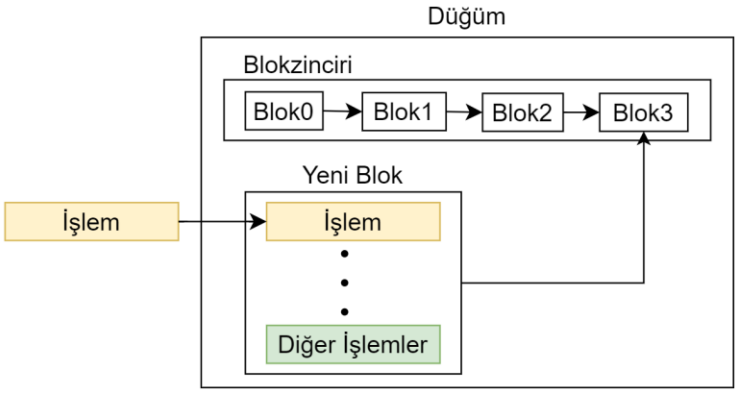
Blokzincir, birbirine zincirlenmiş bloklardan oluşan dijital bir kayıt defteridir. Blokzincirin ademi merkeziyetçilik/merkeziyetsizcilik (decentralization), kalıcılık (persistency), anonimlik (anonymity) ve denetlenebilirlik (auditability) gibi bazı karakteristik özellikleri vardır [2].

Blokzincir yapısı, ademi merkeziyetçilik ilkesine dayanmaktadır, yani herhangi bir merkezi yönetim yoktur. Bu sayede br merkezin yaptığı işleme güvenmek zorunda kalmadan, dağıtık olarak saklanan verinin doğrulanmasından tüm dahil olan birimler sorumludur. Bu da, yapılan işlemleri yanıltmayı ciddi ölçüde zorlaştırır. Blokzincir ağları düğümlerden (node) oluşur. Her düğüm, bir blokzincir istemcisi gibi davranan fiziksel bir makinedir. Blokzincirine gönderilen bir işlem (transaction) öncelikli olarak bir tane düğüme gönderilir. Daha sonra düğümler birbirlerine bu işlemi ileterek tüm ağın işlemden haberdar olmasını sağlar. Ethereum blokzincir ağında tüm düğümler arasında Şekil 2.1'de de gösterildiği üzere düğümler arasında P2P (peer to peer) iletişimi vardır. Ethereum blokzincir ağı, düğümler arasında P2P protokolü ile iletişim kurmasıyla bilinir.



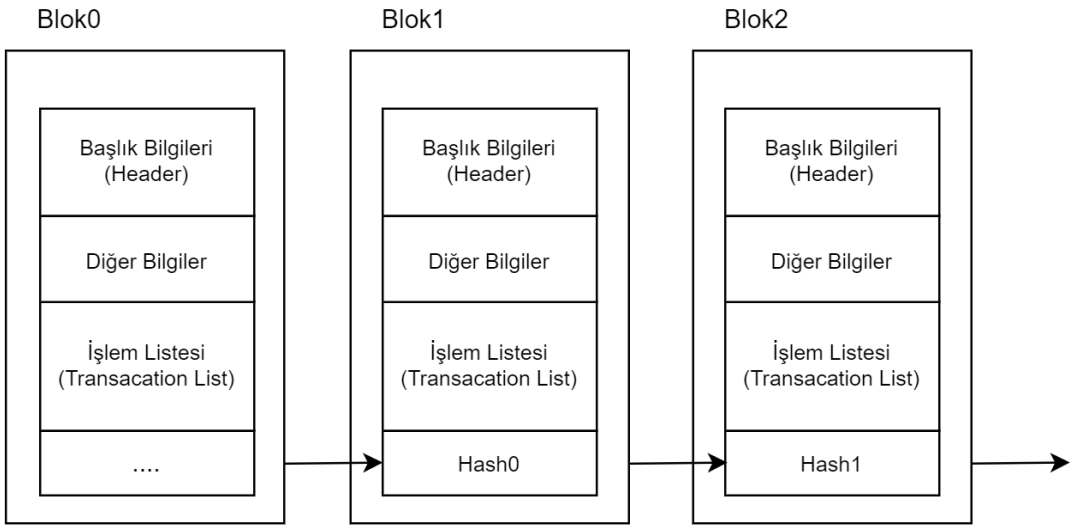
Şekil 2.1: Blokzincirinde Düğümler

Blokzincirinde yapılan tüm işlemlerin kayıtları düğümlerde bloklar halinde tutulur (Şekil 2.2). Düğüme gelen birden fazla işlem bir araya getirilerek bir blok olarak değerlendirilir. Bir işlem blokzincir sistemi tarafından onaylanıyorsa, bu, yapılan işlemin diğer düğümlere de başarılı şekilde kaydedildiği anlamına gelir.



Şekil 2.2: İşlemlerin Düğümlere Kaydedilmesi

Blokzincirine kaydedilen işlemler pratikte değiştirilemez (değiştirilmesi için gereken işlem kapasitesi çok yüksektir); çünkü her blok bir sonraki bloğa kendi özet (hash) değeri ile bağlıdır (Şekil 2.3). Dolayısıyla pratikte değiştirilmesi mümkün olmayan bir yapıya sahiptir. Bir işlem değiştirilirse, o bloğun özet değeri zarar görür, dolayısıyla zincir mimarisi bozulur. Blokzincirindeki bu sağlam yapı kurcalamaya karşı direnç veya kalıcılık olarak adlandırılır. Blokzincirindeki işlemler düğümlerde açık bir şekilde kaydedildiğinden ve değiştirilemediğinden dolayı, düğümlerdeki kayıtlar aracılığıyla tüm işlemler geriye doğru izlenebilir. Bu da denetlenebilirlik sağlar.



Şekil 2.3: Blokzincir Ağlarının Zincir Yapısı

Blokzincir'de tüm işlemler bir adres değeri ile yapılır. Bu durum, blokzincir sistemi üzerinden haberleşmek ya da bilgi alıp göndermek isteyen her kullanıcının isteklerini blokzincire gönderdiği bir hesabı olmasını gerektirir. Bu hesabın bir gizli anahtarı, bir açık anahtarı ve hesap adresi/adresleri vardır. Haberleşmek isteyen her kullanıcı bir ya da birden fazla adres değeri ile işlem yapabilir. Kullanıcının gerçek kimliği ile adres değerleri arasında herhangi bir bağlantı olmadığı için blokzincirinin anonimlik sağladığı ifade edilmektedir.

Ethereum blokzinciri, uygulama geliştirmeye izin veren ve akıllı sözleşme (smart contract) kodlarını çalıştırabilen bir yapıya sahiptir. Akıllı sözleşmeler, içerisinde çağrıldıklarında otomatik olarak çalıştırılan fonksiyonlar barındıran, geliştirmeye açık yazılımlardır. Ethereum teknik incelemesinde, "Akıllı sözleşmeler, değer içeren ve yalnızca belirli koşullar yerine getirildiğinde kilidini açan kriptografik 'kutular' ..." ifadesi beyan edilir [17]. Akıllı sözleşme, blokzincirindeki bir hesap gibi davranır ve çağrıldıklarında otomatik olarak çalıştırılan bir dizi işleve sahiptir.

Bir Ethereum akıllı sözleşmesini kodlamak için farklı yazılım dilleri kullanılabilir. Solidity yazılım dili, akıllı sözleşme geliştirmek konusunda en yaygın kullanım alanına sahiptir. Solidity resmi web sitesi kaynak [18]'de Solidity programlama dili ile ilgili tüm ayrıntılı dokümantasyon sunulmaktadır. Solidity yazılım dili ile yazılan akıllı kontratlar Solidity derleyicisi ile derlendikten sonra iki farklı çıktı oluşturur; ABI (Application Binary Interface) ve bayt kodu (byte code). Bölüm 3.2'de detaylı anlatılan Şekil 3.2'den de anlaşılacağı üzere; ABI sayesinde Ethereum ağına konuşlandırılan akıllı sözleşmelerle iletişim kuracak olan arayüzlere sahip olunur. Bayt kodu çıktısı ise akıllı sözleşmenin blokzincirine yerleştirilen kopyasıdır. Akıllı sözleşmede işlem yapmak demek blokzincirine yerleştirilmiş olan bu bayt kodu ile işlem yapmak demektir.

Ethereum ağında konsensüs mekanizmaları vardır. Ethereum Whitepaper'ının da sunulduğu Ethereum resmi web sitesinde bu konsensus mekanizmalarının detayları anlatılmaktadır [19]. Ethereum ağının güvenliği, konsensüs mekanizmaları sayesinde sağlanır. Konsensüs mekanizmaları, ağdaki tüm düğümlerin bir işlemi onaylamak veya reddetmek için anlaşmalarını sağlar. Ethereum ağı, Proof-of-Work (PoW) ve Proof-of-Stake (PoS) olmak üzere iki farklı konsensüs mekanizması kullanmaktadır.

PoW, blokların doğrulanması aşamasında yüksek matematiksel işlem gücü gerektiren bir çalışma mekanizmasına sahiptir. Bu yöntem, blokların doğrulanması esnasında çok fazla enerji tüketimine sebep olduğu için eleştirilen bir yöntemdir. PoS konsensüs mekanizmasında ise blokzincirindeki hesap sahipleri belli bir miktar ether miktarını "stake" adı verilen depozitolara yatırırlar ve blok doğrulamalarında bu miktar ile doğru orantılı olarak söz sahibi olurlar. Bu konsensüs yöntemi, stake için daha çok ether ayırabilen hesap sahiplerinin blok doğrulamalarında daha fazla söz hakkı olduğu gerekçesiyle güvenlik endişesi barındıran bir yöntemdir.

Ethereum blokzincirinde gerçekleştirilen her işlem için ağa belirli bir gaz (gas) ücreti ödenir. Yine Ethereum resmi web sitesinde gaz harcalamaları ile ilgili tüm detaylı bilgiler mevcuttur [20]. Gaz, Ethereum blokzincirinde işlemlerin gerçekleştirilmesi için gerekli olan hesaplama gücü olarak düşünülebilir. Bir işlem yapılırken, Ether cinsinden bir ücret hesaplanır ve ağa bir ödeme yapılır. Bu bağlamda kullanılan iki terim işlem ücreti ve birim gaz fiyatıdır.

İşlem ücreti/Gaz ücreti (Txn fee/ Gas fee):İşlem ücreti, işlemi blokzincirine eklemek için ödenen toplam ücrettir ve genellikle "ether" adı verilen büyük birimlerle ifade edilir. Bu ücret, işlemin gerçekleşmesi için kullanılan gaz miktarına göre belirlenir. İşlem ücreti iki bileşenden oluşur: gaz miktarı ve gaz birim fiyatı.

Birim Gaz fiyatı (Unit Gas Price): İşlem ücretinin hesaplanmasında kullanılır ve gaz birimi başına maliyettir. Gaz fiyatı, bir işlemdeki her bir gaz birimi için ödenen fiyatı belirtir. Genellikle "gwei" adı verilen küçük birimlerle ifade edilir. En küçük birim olan Wei genellikle gaz hesaplamalarında çok küçük olduğu için, hata yapılmaması için, kullanılmaz. Bunun yerine Gwei (giga wei) birimi gaz fiyatı hesaplamaları için daha uygun görülür. Yine de gaz fiyatları Wei, Gwei ve Ether cinsinden ifade edilebilir. Bu birimler arasındaki ilişki aşağıdaki gibidir:

1 Ether = 1,000,000,000 Gwei ve 1 Gwei = 1,000,000,000 Wei

Ethereum'da ağın kendisi gaz fiyatını belirlemez. Gaz fiyatını belirleyen kişi işlemi başlatan kişi yani işlemi ağa gönderen kullanıcıdır. Gaz fiyatı piyasa koşullarına ve talebe göre değişebilir.

Gaz fiyatı ne kadar yüksek olursa, işlem o kadar hızlı bir şekilde gerçekleşir. Bunun sebebi; madenciler daha yüksek gaz fiyatlarına sahip işlemleri öncelikli olarak işleme alırlar, çünkü daha yüksek ücretler alırlar. Bu nedenle, gaz fiyatı yüksek olan işlemler

daha hızlı bir şekilde işlenir denilmektedir. Yine de tüm işlemlerin gerçekleşmesini sağlamak için bazı mekanizmalarla sistemin devamlılığı kontrol altında tutulmaktadır. Bu sayede gaz fiyatı düşük olan işlemler süresiz şekilde bekletilmemektte, mutlaka işleme alınmaktadır. Gaz fiyatı, ağın yoğunluğuna göre değişebilir. Ağdaki yoğunluk arttıkça, gaz fiyatları da işlemin hızlı işlenmesi için artmaya başlar. Bu da işlem ücretini artırır. Dolayısıyla ağın yoğun zamanlarında gaz birim fiyatı ve işlem ücretleri artar yorumlarına sıkça rastlanmaktadır.

Ethereum ile ilgili teknik detaylara odaklanan Ethereum sarı makalesinde (yellow paper) iki farklı gaz limitinden bahsedilmektedir [21]: işlem gaz limiti ve blok gaz limiti.

İşlem Gaz Limiti: İşlemde kullanılabilecek maksimum gaz miktarını ifade eder. İşlemin gaz limiti, işlemi gönderen tarafından belirlenir. Her işlem, farklı bir gaz limitine sahip olabilir. Bu limit işlemin türüne ve işlemdeki verilerin boyutuna bağlıdır. İşlem ne kadar karmaşık olursa, işlem gaz limiti de o kadar yüksek olacaktır. İşlem için maksimum gaz limitini belirleyen kişi, işlemi gönderen kullanıcıdır. İşlem gaz limitini aşan işlemler yapılmaz. İşlem gerçekleşirken harcanan toplam gaz miktarı, işlem gaz limitiyle aynı ya da daha aşağıda bir değerdir. İşlem gaz limitinin tamamı blokzinciri tarafından kullanılmadığı zaman, artan gaz miktarı işlemin gerçekleştirildiği hesaba iade edilir.

Blok Gaz Limiti: Her bloğun işleme alabileceği maksimum bir gaz limiti vardır. O bloktaki işlemlerin toplam gaz miktarı, bloğun gaz limitini aşamaz. Bir işlemin gaz limiti, bloğun gaz limitini aşıyorsa bu işlem o bloğa kabul edilmez. O yüzden çok yüksek işlem gaz limiti belirtmek, blokların gaz limitini aşabileceği için, bu işlemin uygun bloğa kabul edilmesini beklemesine sebep olabilir. Bu durum da işlemin gerçekleşme süresini uzatacaktır.

2.2 Homomorfik Kriptosistem ve Paillier Şifrelemesi

Homomorfik şifreleme, belirli matematiksel işlemlerin şifrelenmiş veriler üzerinde şifre çözme işlemi yapılmadan uygulanmasına izin verir. Verilerin güvenilmeyen bir ortamda veya verileri görme yetkisi olmayan üçüncü kişiler tarafından işlendiği durumlarda kullanılması için son derece uygun bir kriptosistemdir. Homomorfik işlemlerin çıktısı, işlemlerin sonucunda elde edilen verilerin şifreli halidir.

Kaynak [22]'te detaylı bir şekilde ifade edildiği üzere şifreleme şemaları başlıca üç kategoriye ayrılmıştır: kısmen homomorfik şifreleme (Partially Homomorphic Encryption - PHE), biraz homomorfik şifreleme (Somewhat Homomorphic Encryption - SWHE) ve tamamen homomorfik şifreleme (Fully Homomorphic Encryption - FHE). PHE, şifreli metinler üzerinde sadece belli başlı matematiksel işlemlere izin veren bir şifreleme işlemidir. RSA, ElGamal ve Paillier kriptosistemleri bu türe örnektir. Örneğin; RSA sadece matematiksel çarpma işlemine izin verirken Paillier ile sadece homomorfik toplama işlemi yapılabilmektedir. SWHE sisteminde işlem sayısında kısıtlamalar söz konusudur. Boneh-Goh-Nissim (BGN) bir SWHE yöntemine bir örnektir [22]. Bu sistemde şifreli veriler üzerinde sayısız toplama işlemi yapılırken yalnızca bir tane çarpma işlemi yapılabilmektedir. FHE sisteminde ise sayısız sayıda toplama ve çarpma işlemi yapılabilmektedir.

Bu çalışmada, Paillier kısmen homomorfik şifreleme sistemi [23] kullanılmıştır. Yukarıda da bahsedildiği gibi Paillier sistemi, homomorfik toplama özelliği sayesinde verilerin şifresini çözmeye gerek kalmadan şifrelenmiş verileri toplamaya izin verir. Bu özellik şu şekilde formüle edilmiştir:

$$E(a + b) = E(a)E(b) \tag{2.1}$$

Denklem 2.1'de E şifreleme anlamına gelirken a ve b iki düz metni temsil eder. Paillier kriptosistemi, diğer kriptosistemlerde de olduğu gibi anahtar oluşturma, şifreleme ve şifre çözme algoritmalarına sahiptir. Bunun yanında kısmen homomorfik bir kriptosistem olduğu için toplama ve sabit katsayı ile çarpma gibi homomorfik operasyon özellikleri de vardır.

Anahtar Üretimi:

Adım 1: Rastgele büyük asal sayılar p ve q seçilir.

Adım 2: $n = pq$, $\lambda = lcm(p-1, q-1)$ hesaplanır.

Adım 3: Rastgele g sayısı seçilir $g \in Z^*n^2$

Adım 4: $g = L(g\lambda \bmod n^2) - 1 \; mod n$

Açık anahtar : (n, g)

Gizli anahtar : (λ, μ)

Şifreleme:

Adım 1: m is açık metin, $0 \leq m < n$

Adım 2: Rastgele r değeri seçilir, $0 < r < n$

Adım 3: $c = g^m r^n mod n^2$ hesaplanır.

Şifre Çözme:

Adım 1: $c \in Z^* n^2$ (c: şifreli metin)

Adım 2: m değeri hesaplanır, $m = L\,(c\lambda \bmod n^2) \cdot \lambda \bmod n$

Paillier Homomorfik Operasyonlar:

D şifre çözme işlemi için, E şifreleme işlemi için kullanılan sembollerdir.

- Homomorfik Toplama: m_1 ve m_2 açık metinler olmak üzere;

$D\,(\,E\,(\,m_1\,,r_1\,)\,.E\,(\,m_2\,,r_2\,)\,mod\,n^2\,) = (m_1 + m_2\,)\,mod\,n^2$

- Sabit Katsayı ile Homomorfik Çarpma:

1. $D\,(\,E\,(\,m_1\,,r_1\,)\,m_2\,mod\,n^2\,) = (m_1 m_2)\,mod\,n$ denklemi

$D\,(\,E\,(\,m_2\,,r_2\,)\,m_1\,mod\,n^2\,) = (m_1 m_2)\,mod\,n$ olarak da kullanılabilir.

2. k sabit bir katsayı olmak üzere;

$D\,(\,E\,(\,m_1\,,r_1\,)\,k\,mod\,n^2\,) = k m_1\,mod\,n$

Burada dikkat edilmesi gereken husus; n^2 değerinin n anahtar uzunluğunun iki katı kadar olmasıdır. Yani n değeri 256 bit seçiliyorsa, n^2 değeri 512 bit uzunluğunda olmaktadır. Aynı şekilde 256 bitlik iki şifreli metnin homomorfik toplamı yani artimetik çarpımları da yine 512 bit uzunluğunda olmaktadır.

2.3 Doğrusal (Lineer) Regresyon Analizi

Doğrusal regresyon, bilinen veri değerlerini kullanarak bilinmeyen verilerin değerini tahmin eden istatistiksel bir veri analiz tekniğidir. Bağımlı değişkeni, bağımsız değişkenler cinsinden doğrusal bir denklem olarak matematiksel olarak modeller. Doğrusal regresyon parametreleri için formüller şunlardır:

$$y = a_0 + a_1 x \tag{2.2}$$

$$a_0 = \bar{y} - a_1 \bar{x}\,,\ \bar{x} = \frac{\sum x_i}{n}\,,\quad \bar{y} = \frac{\sum y_i}{n} \tag{2.3}$$

$$a_1 = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{n\sum {x_i}^2 - (\sum x_i)^2} \tag{2.4}$$

3. METODOLOJİ

Bu çalışmada, akıllı sözleşmelerdeki şifrelenmiş veriler üzerinde istatistiksel hesaplamalar yapan blokzinciri tabanlı bir veri paylaşım sistemi önerilmektedir. Bu bölümde; önerilen yöntem, sistemin bileşenlerinin birbirleriyle nasıl etkileşime girdiği ve istatistiksel hesaplamaların bu sistem içinde nasıl gerçekleştirildiği detaylı şekilde incelenmektedir. Uygulama olarak seçilen lineer regresyon analizi incelenmiş ve uygulama mimarisi detaylı bir şekilde anlatılmıştır.

3.1 Önerilen Yöntem

Önerilen sistem, üçüncü tarafların, görüntüleme yetkileri olmayan veriler üzerinde belli analizler yapmalarına izin verir. Senaryoda farklı bölgerlerden toplanan sensör bilgileri, ait olduğu uç birime aktarılır. Uçbirim cihazları, sensörlerden gelen verileri toplu biçimde tutar. Uçbirim cihazlarında toplanmış halde bulunan veriler şifrelenerek blokzincirine gönderilir. Blokzincirinde, her bir uçbirimden gelen şifreli toplam değerleri tekrar toplanarak nihai toplam değeri elde edilir. Buradaki önemli nokta, uçbirimden gelen toplam sensör değerleri şifreli olduğu için toplam değer elde edilirken şifreli veriler kendi aralarında homomorfik toplama işlemi ile toplanmıştır. Bu sayede sensör verileri uçbirimlerde, uçbirim değerleri ise blokzinciri içerisinde toplanarak tüm sensör verileri toplanmış olur.

Senaryo gereği üçüncü bir tarafın sensör verileri üzerinde istatistiksel analiz yapmak istediği varsayılmıştır. Tüm toplam değerleri blokzinciri içerisinde homomorfik toplam sonucunda şifreli bir biçimde elde edilmiştir. Bu veri akışı sayesinde üçüncü tarafın istediği toplam değerleri elde edilmiş olur. Kısacası; akıllı sözleşmeler, veri sahiplerinden şifrelenmiş toplu verileri toplar ve bunları homomorfik işlemler kullanarak araştırmacı için hazır hale getirir.

Çalışmanın geri kalanında [9]'daki terminolojiye uyarak uç cihazlara "veri sahipleri" ve üçüncü kişilere "araştırmacı" denmiştir. Sistemdeki bir diğer aktör, blokzincir üzerinde araştırmacılar için akıllı sözleşmeler uygulayan "sözleşme geliştiricisi"dir.

Sözleşme geliştiricisi, geliştirdiği akıllı sözleşmeyi geliştirmek ve blokzincirine yerleştirmekle görevlidir.

Senaryoda yer alan sözleşme geliştiricisi, dört farklı veri sahibi ve araştırmacı için ayrı hesap adreslerinden işlemler gerçekleştirilmiştir.

Çizelge 3.1'de sistemdeki aktörlerin Ethereum test ağında işlem yaptıkları adres değerli verilmiştir.

Çizelge 3.1: Aktörlerin Ethereum Ağındaki Hesap Adresleri

Roller	Hesap Adresleri
Sözleşme Geliştiricisi	0xEDA7F5df47B6Ed8Cd6722CAB8F8C0f49F0E01aB3
Araştırmacı	0x83d9DECbAB1AC5c3FCBd8A6d48a0990c96eC6148
VeriSahibi1	0xA28618aae27019c21Ec1A4dc3984160a2d2EEB8A
VeriSahibi2	0x626638a063d259449b4D5BF22e7443E1a4C99BFb
VeriSahibi3	0x837089Df472A9460F58617C8DB5A90A2bDCC0Ae7
VeriSahibi4	0xfCAb9dCc8e720df275e6CD415C8A5843ab46749f

Çizelge 3.1'de de görüldüğü gibi, sistemdeki tüm roller farklı hesaplar ile Ethereum ağına dahil olmuşlardır.

3.1.1. Doğrusal regresyon yöntem analizi

Denklem-2.3 ve Denklem-2.4 bakarsak, a_0 ve a_1 'in hesaplanması için x, y, n, x^2 ve xy olmak üzere beş farklı değişkenin toplam değerlerine ihtiyaç vardır. Her veri sahibi kendi değişkenlerinin toplamını şifreler, akıllı sözleşmeye gönderir ve Çizelge 3.2'de gösterildiği gibi kaydeder.

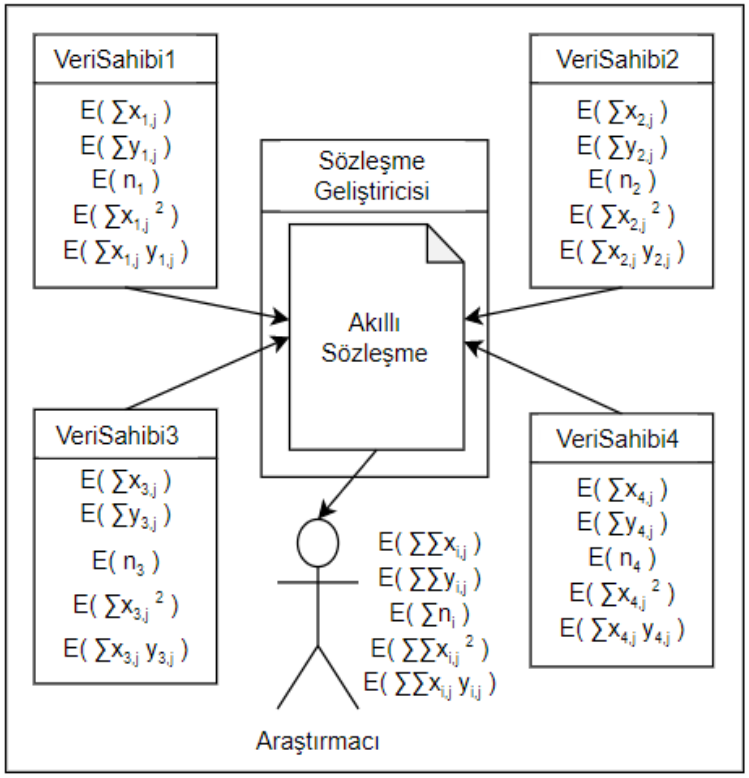
Çizelge 3.2: Blokzincirdeki Toplam Verilerinin Şifreli Hali

Toplam sensör değerleri	x	y	n	x.y	x^2
VeriSahibi1*	E($\sum x_{1,j}$)	E($\sum y_{1,j}$)	E(n_1)	E($\sum x_{1,j}y_{1,j}$)	E($\sum x_{1,j}^2$)
VeriSahibi2*	E($\sum x_{2,j}$)	E($\sum y_{2,j}$)	E(n_2)	E($\sum x_{2,j}y_{2,j}$)	E($\sum x_{2,j}^2$)
VeriSahibi3*	E($\sum x_{3,j}$)	E($\sum y_{3,j}$)	E(n_3)	E($\sum x_{3,j}y_{3,j}$)	E($\sum x_{3,j}^2$)
VeriSahibi4*	E($\sum x_{4,j}$)	E($\sum y_{4,j}$)	E(n_4)	E($\sum x_{4,j}y_{4,j}$)	E($\sum x_{4,j}^2$)
Toplam**	E($\sum\sum x_{i,j}$)	E($\sum\sum y_{i,j}$)	E($\sum n_i$)	E($\sum\sum x_{i,j}y_{i,j}$)	E($\sum\sum x_{i,j}^2$)

*: veri sahiplerinden gelen şifreli toplam sensör verileri

**: lineer regresyon analizinde kullanılması gereken tüm verilerin şifreli toplam değerleri

Çizelge 3.2'de, $x_{i,j}$ i. veri sahibinden gelen j. veri noktasının x koordinatını ve $y_{i,j}$, i. veri sahibinden j. veri noktasının y koordinatını temsil eder. Toplamlar, sonlu sayıda veri noktası ile elde edilir ve veri sahipleri farklı sayıda veri noktasına sahip olabilir. Basitlik adına, toplam ifadelerinin limitleri şemadaki toplamlara dahil edilmemiştir. Önerilen sistemin nasıl çalıştığını göstermek için, dört veri sahibinin ve bir araştırmacının akıllı sözleşme ile nasıl etkileşime girdiği Şekil 3.1'de sunulmaktadır.



Şekil 3.1: Blokzincirinde Doğrusal Regresyon Hesaplaması için Gerçekleşen Veri Akışı

Bir veri sahibi tarafından gönderilen şifreli veriler, doğrusal regresyon parametrelerinin hesaplanmasında gerekli olan veri toplamlarını içerir. Denklem 2.3 ve 2.4'te görüldüğü gibi, a_0 ve a_1 parametrelerinin hesaplanması için beş farklı toplama ihtiyaç vardır. Bunlar x'ler, y'ler, n'ler, x^2'ler ve xy'lerin toplamlarıdır.

Her veri sahibi, veri noktalarının sensörlerden aldığı ilgili toplamlarını şifreler ve akıllı sözleşmeye gönderir. Bu şifrelenmiş toplam değerler akıllı sözleşmede bir değişkende homomorfik toplanarak saklanır. Bir diğer ifadeyle her bir şifrelenmiş toplam değer türü, o tür için şifrelenmiş toplamı elde etmek üzere homomorfik olarak birbirleriyle toplanır (yani cebirsel olarak çarpılır).

Örneğin, dört veri sahibi için hepsinin x degerlerinin şifreli toplamı şu şekilde hesaplanır:

$$E\left(\sum_{j=1}^{n_1} x_{1,j}\right) \times E\left(\sum_{j=1}^{n_2} x_{2,j}\right) \times E\left(\sum_{j=1}^{n_3} x_{3,j}\right) \times E\left(\sum_{j=1}^{n_4} x_{4,j}\right) \quad (3.1)$$

Paillier'in homomorfik özelliği, yukarıdaki hesaplamanın sonucunun tüm x'lerin şifrelenmiş toplamı ile aynı olmasını sağlar:

$$E\left(\sum_{j=1}^{n_1} x_{1,j} + \sum_{j=1}^{n_2} x_{2,j} + \sum_{j=1}^{n_3} x_{3,j} + \sum_{j=1}^{n_4} x_{4,j}\right) \quad (3.2)$$

Daha genel bir ifade ile şifreli dataların toplamı şu şekilde ifade edilebilir:

$$E\left(\sum_{1=1}^{k} \sum_{j=1}^{n_i} x_{i,j}\right) = \prod_{i=1}^{k} \left(E\left(\sum_{j=1}^{n_i} x_{i,j}\right)\right) \quad (3.3)$$

Hesaplanan şifreli toplamlar, araştırmacının kullanımına sunulur.

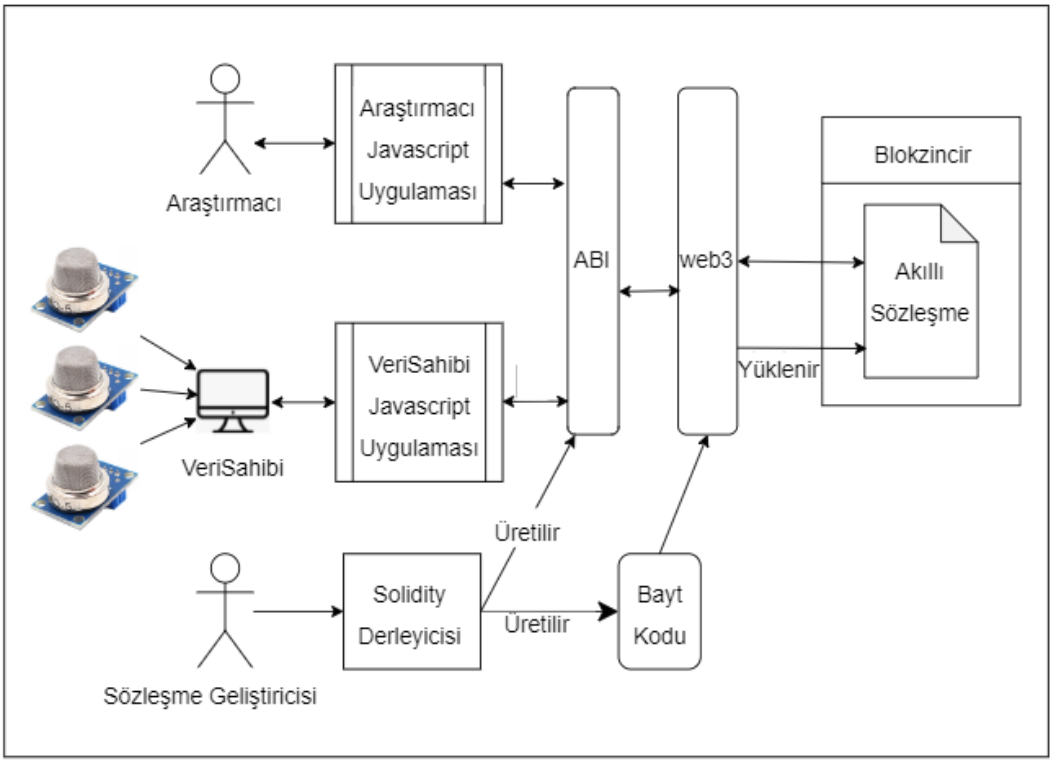
Toplamları aldıktan sonra, araştırmacı kendi özel anahtarını kullanarak toplamların şifresini çözer ve doğrusal regresyon denklemlerini elde etmek için bunları Denklem 2.3 ve 2.4'e yerleştirir.

3.2 Uygulama Geliştirme

Önerilen sistem, Paillier şifreleme sistemi kullanılarak Ethereum test ağı olan Goerli'de uygulanmıştır. Akıllı sözleşme geliştirmek ve sözleşmeyle etkileşim kurmak için kullanılan yazılım araçları şunlardır:

- Web3.0: Web3 olarak da adlandırılır. Blokzincir sistemi ile dışarıdan etkileşim için kullanılan bir kütüphanedir.
- Remix: Akıllı sözleşmeyi geliştirmek ve dağıtmak için çevrimiçi Remix platformu seçilmiştir.
- Metamask: Ethereum blok zincirine bağlanmak için kullanılan elektronik cüzdandır. Akıllı sözleşmeyi blokzincir sistemine yerleştirmek (deploy) için Remix'te metamask hesabı tanıtılmıştır.
- Visual Studio Code: Kullanıcı JavaScript uygulamaları geliştirmek için kullanılmıştır.

Sistemdeki üç aktörün; akıllı sözleşme geliştiricisi, veri sahibi ve araştırmacının, Şekil 3.2'de blokzinciriyle nasıl etkileşime girdiğine dair bir genel bakış sunulmuştur. Sözleşme geliştiricisi, akıllı sözleşmeyi Solidity'de programlar ve derler. Akıllı sözleşmenin derlenmesi, sözleşme için diğer kullanıcıların akıllı sözleşme ile etkileşime girmesine izin veren uygulama arayüzünü (ABI – Application Binary Interface) ve akıllı sözleşmenin blokzincirine gönderilecek kopyası olan bayt kodunu üretir. Üretilen bayt kodu, Web3 kütüphanesi kullanılarak blokzincirine dağıtılır (deploy). Uygulamada, akıllı sözleşmeyle etkileşime girecek kullanıcı uygulamalarının geliştirilmesi için ise JavaScript yazılım dili kullanılmıştır.



Şekil 3.2: Projedeki Aktörlerin Sisteme Entegrasyonu

Bu çalışmada büyük tamsayılarla hesaplamalara ihtiyaç duyduğundan, hem akıllı sözleşme hem de kullanıcı uygulamaları büyük tamsayı kütüphanelerine ihtiyaç duyar. Uygulamada akıllı sözleşmenin geliştirilmesi için BigNumbers.sol kütüphanesi [24], veri sahibi ve araştırmacı uygulamaları için yazılan betiklerde ise BigInteger.js kütüphanesi [25] kullanılmıştır.

3.2.1 Doğrusal regresyon uygulama senaryosu

Gerçekleştirilen senaryoda sensörlerden gelen x ve y değerleri, veri sahipleri olarak adlandırılan uçbirimlerde toplanıyor. Her bir sensörden gelen x ve y değerlerinden yola

çıkarak uçbirimlerde x^2, xy değerleri hesaplanıyor ve n ile ifade edilen veri sayısı bir artıyor. Bir uçbirime bağlı olan tüm sensörlerden gelen veriler uçbirimde toplandıktan sonra şifreleniyor ve Çizelge 3.2'de ifade edilen değerler E($\sum x_{1,j}$), E($\sum y_{1,j}$), E(n_1), E($\sum x_{1,j}\, y_{1,j}$), E($\sum x_{1,j}^2$) akıllı sözleşmeye gönderilmek üzere hesaplanmış oluyor.

Veri sahiplerine gelen sensör verileri için örnek değerlerle oluşturulmuş bir çizelge aşağıda verilmiştir. Elbetteki bu örnek, çok büyük sayı kümeleri ile de denenebilir. Bu çizelgede veri sahiplerindeki sensör verileri ayrı ayrı gösterilmektedir. Bu sensör verilerinin toplam değerleri de 'Toplam_VS' isimlendirmesiyle çizelgede gösterilmiştir. Tüm şifreli 'Toplam_VS' değerleri senaryo gereği akıllı sözleşmeye şifreli şekilde gönderilmektedir. Araştırmacı ise akıllı sözleşmeden tüm veri sahiplerinin toplam değerlerini istediği zaman, Çizelge 3.3'teki 'Toplam' satırındaki değerleri şifreli olarak almaktadır. Daha sonra bu şifreli değerleri kendi elindeki özel anahtar ile çözüp toplam değerleri elde etmektedir.

Çizelge 3.3: Sensörlerden Veri Sahiplerine Gelen Örnek Değerler

	x	y	n	xy	x^2
VeriSahibi1	1 2 3	4 2 5	3	4 4 15	1 4 9
Toplam_VS1	6	11	3	23	14
VeriSahibi2	4 5	2 5	2	6 20	9 16
Toplam_VS2	9	7	2	26	25
VeriSahibi3	6 7	4 6	2	15 36	25 36
Toplam_VS3	13	10	2	51	61
VeriSahibi4	8 9 10	6 5 9	3	54 45 90	64 81 100
Toplam_VS4	27	20	3	80	130
Toplam	55	48	10	180	230

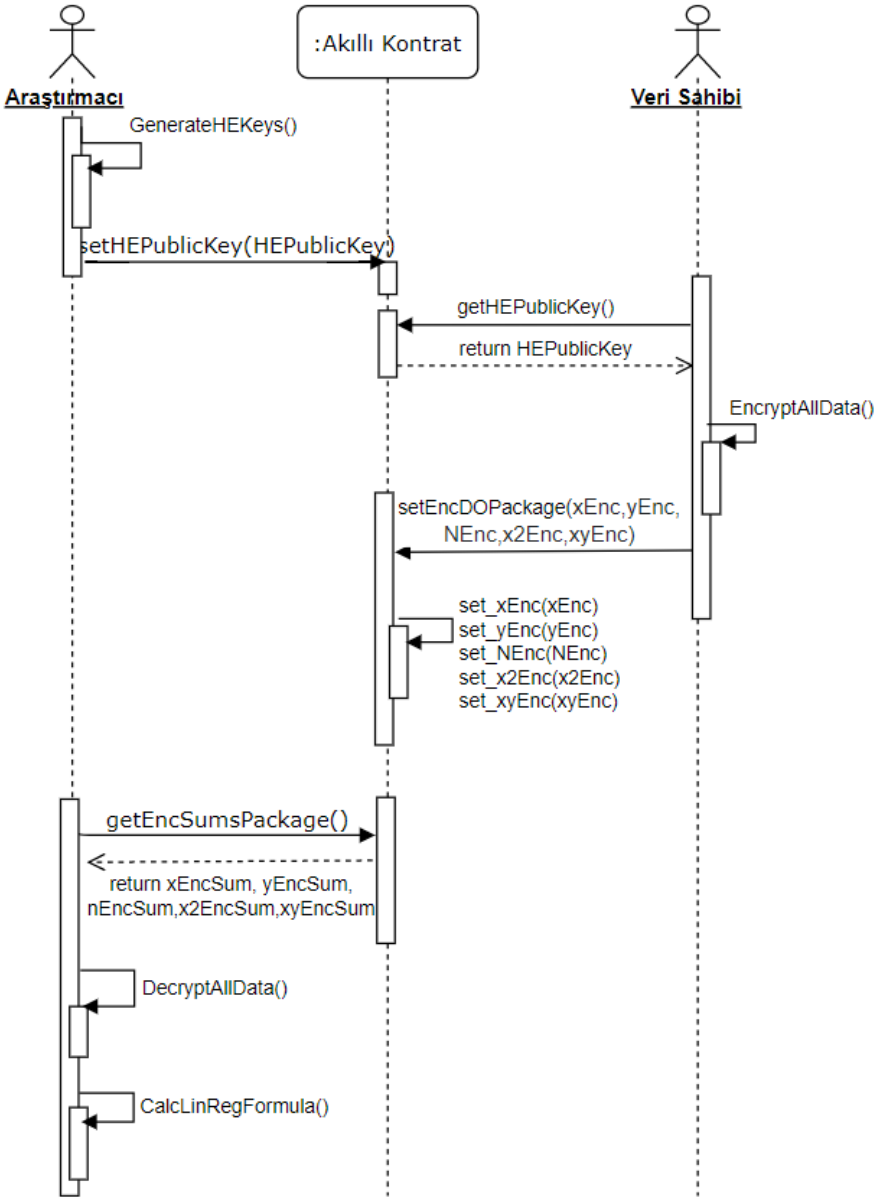
Yapılan deney, Çizelge 3.3'teki değerler kullanılarak gerçekleştirilmiştir. Bu çizelgedeki Toplam-VS değerleri ve Toplam değeri şifreli şekilde blokzincirinde bulunmaktadır.

Önerilen sistemdeki akışı gösteren bir senaryo diyagramı Şekil 3.3'de verilmiştir. Diyagram, araştırmacı ile bir veri sahibi arasında veri paylaşımını sağlayan halihazırda blokzincirinde konuşlandırılmış bir akıllı sözleşmenin veri aktarım senaryosunu içermektedir. Diyagramda, veriler üzerinde lineer regresyon analizi yapmak isteyen bir araştırmacı ve bu analizde kullanılacak verileri sağlayan bir veri sahibi bulunmaktadır. Senaryoya dahil olan tüm veri sahiplerinin Şekil 3.3'deki veri sahibi gösterimi üzerinden sisteme dahil olacağı düşünülmüştür.

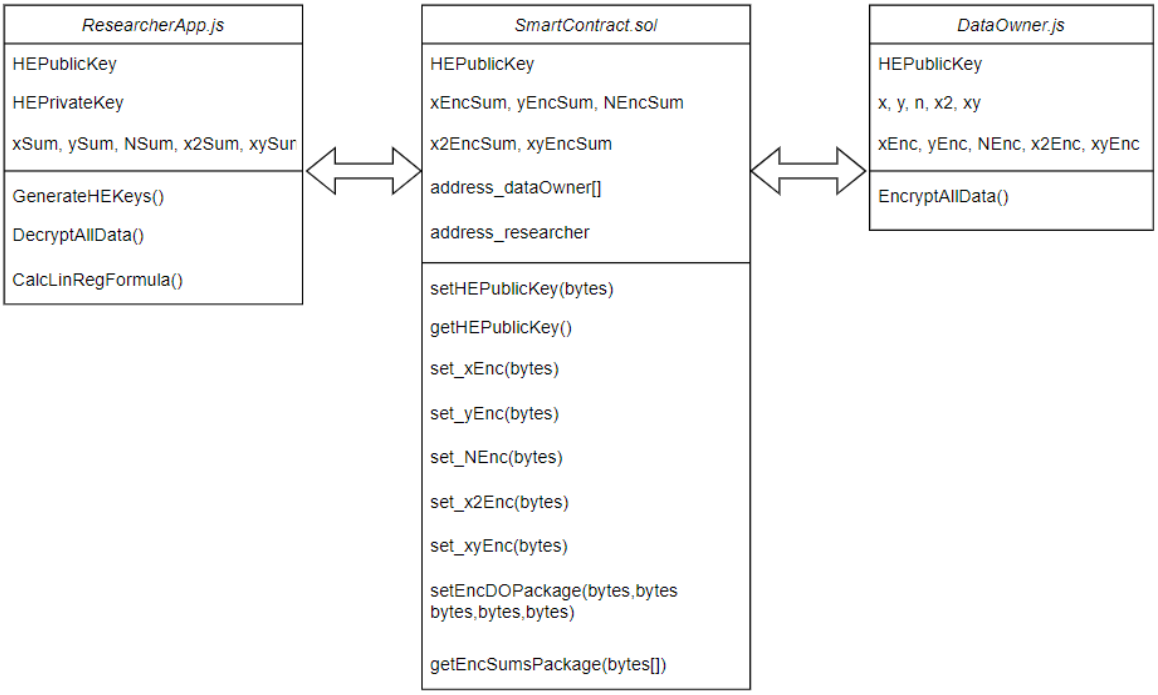
Senaryoda ilk işlem olarak araştırmacı, homomorfik anahtar çiftlerini üretir ve homomorfik açık anahtarını akıllı sözleşmeye kaydeder. Veri sahibi bu açık anahtarı akıllı sözleşmeden alır ve veri değerlerinin toplamını araştırmacının açık anahtarı ile şifreler. Şifreli sonuçları akıllı sözleşmeye gönderir. Senaryoya dahil olan her veri sahibi aynı adımları takip eder ve şifrelenmiş toplamlarını akıllı sözleşmeye gönderir.

Akıllı sözleşme, tüm veri noktalarının şifrelenmiş toplamlarını elde etmek için veri sahiplerinden aldığı şifreli toplamları kendisine geldikçe homomorfik olarak toplar ve bir değişkende tutar. Buradaki önemli nokta, veri sahiplerinden gelen değerlerin şifreli olmasına rağmen akıllı sözleşmeye doğrudan kaydedilmeden toplam değere eklenerek tutulmasıdır. Burada bahsedilen toplama işlemi homomorfik toplamadır ve Paillier sisteminde aslında modüler bir çarpma işlemidir.

Projenin uygulaması için yazılan kodlar Şekil 3.4'te de gösterildiği gibi üç aşamada düşünülebilir; araştırmacının homomorfik anahtar çiftini üretip daha sonra toplam şifreli değerleri çözerek doğrusal regresyon yaptığı Javascript kod parçası, veri sahiplerinin homomorfik anahtarı ile toplam sensör değerlerini şifreleyerek blokzincirine gönderdiği Javascript uygulaması ve akıllı sözleşmenin kendisi.



Şekil 3.3: Senaryodaki Veri Akışı



Şekil 3.4: Araştırmacı, Akıllı Sözleşme ve Veri Sahibi için Sınıf Fonksiyonları

Yazılan uygulamaların Şekil 3.4'te gösterilen detaylı fonksiyonalitesi ve açıklaması Çizelge 3.4, Çizelge 3.5 ve Çizelge 3.6'da açıklanmıştır.

Akıllı sözleşme için geliştirilen uygulamadaki fonksiyonların açıklaması aşağıdaki Çizelge 3.4'te açıklanmıştır. Fonksiyonların detaylı açıklamalarından da anlaşılacağı üzere akıllı sözleşmede, araştırmacının homomorfik şifreleme için ürettiği açık anahtarı kaydetmek için bir fonksiyon mevcuttur. Bu fonksiyonu kullanarak araştırmacı açık anahtarını akıllı sözleşemeye göndererek kaydeder. Akıllı sözleşmedeki bir diğer önemli nokta ise, veri sahiplerinin kendilerindeki şifreli toplam verilerini tek seferde bir paket halinde akıllı sözleşmeye gönderip kaydetmesini sağlayan bir fonksiyon bulundurmasıdır. Bu sayede veri seti içerisinde herhangi bir bozukluk olmadan, tek seferde, birbirleriyle ilişkili olan bu veri değerleri akıllı sözleşmeye veri sahipleri tarafından gönderilebilmektedir. Şifreli veriler akıllı sözleşmeye bir paket halinde geldikten sonra ise her şifreli toplam değerinin kendi ilgili toplam değerine eklenmesi gerekmektedir. Bunun için her şifreli değerin kendi ilgili toplam değerine homomorfik toplama işlemi ile ekleneceği bir fonksiyonu vardır.

Çizelge 3.4: Akıllı Sözleşme Uygulaması Fonksiyonları

setHEPublicKey(bytes)	Araştırmacı, homomorfik şifreleme için kendi açık anahtarını akıllı sözleşmeye gönderir.
bytes getHEPublicKey()	Her veri sahibi, araştırmacının açık anahtarını akıllı sözleşmeden alır.
set_xEnc(bytes)	Veri sahibi, x değerlerinin şifreli toplamını akıllı sözleşmeye gönderir. Veri sahibinden gelen şifreli toplam değeri, akıllı sözleşmede tutulan kümülatif değere (xEncSum) homomorfik toplama işlemi ile eklenir.
set_yEnc(bytes)	Veri sahibi, y değerlerinin şifreli toplamını akıllı sözleşmeye gönderir. Veri sahibinden gelen şifreli toplam değeri, akıllı sözleşmede tutulan kümülatif değere (yEncSum) homomorfik toplama işlemi ile ile eklenir.
set_NEnc(bytes)	Veri sahibi, N (veri sayısı) değerinin şifreli akıllı sözleşmeye gönderir. Veri sahibinden gelen şifreli değer, akıllı sözleşmede tutulan kümülatif değere (NEncSum) homomorfik toplama işlemi ile eklenerek saklanır.
set_x2(bytes)	Veri sahibi, x^2 değerlerinin şifreli toplamını akıllı sözleşmeye gönderir. Veri sahibinden gelen şifreli toplam değeri, akıllı sözleşmede tutulan kümülatif değere (x2EncSum) homomorfik toplama işlemi ile ile eklenir.
set_xy(bytes)	Veri sahibi, xy değerlerinin şifreli toplamını akıllı sözleşmeye gönderir. Veri sahibinden gelen şifreli toplam değeri, akıllı sözleşmede tutulan kümülatif değere (xyEncSum) homomorfik toplama işlemi ile ile eklenir.
setEncDOPackage(bytes, bytes,bytes,bytes,bytes)	Veri sahipleri kendilerindeki veri toplamlarını şifreli bir biçimde tek seferde akıllı sözleşmeye gönderip kaydederler. Bu fonksiyon, her parametrenin kendi homomorfik toplam fonksiyonunu çağırır ve kümülatif şifreli değerin üzerine ekleme yapar.
getEncSumsPackage (bytes[])	Araştırmacı, akıllı sözleşmeden tüm veri sahiplerinin toplam şifreli değerlerini tek seferde alır.

Araştırmacı için geliştirilen uygulamadaki fonksiyonların açıklaması aşağıdaki Çizelge 3.5'te açıklanmıştır.

Çizelge 3.5: Araştırmacı Uygulaması Fonksiyonları

GenerateHEKeys()	Araştırmacı, homomorfik şifreleme için genel ve özel anahtar çiftleri oluşturur.
DecryptAllData()	Araştırmacı tarafında tüm şifrelenmiş verilerin şifresi çözülür.
CalcLinRegFormula()	Araştırmacı tarafında lineer regresyon formülü uygulanır.

Veri Sahibi için geliştirilen uygulamadaki fonksiyonların açıklaması aşağıdaki Çizelge 3.6'da açıklanmıştır.

Çizelge 3.6: Veri Sahibi Uygulaması Fonksiyonları

EncryptAllData()	Tüm veri sahipleri kendi için x, y, n, x^2 ve xy toplamlarını şifreler.

Akıllı sözleşmedeki x değerlerinin homomorfik toplama işlemi için yazılan sözde kod Çizelge 3.7'de verilmiştir. Tüm değişkenlerin toplam değerlerini hesaplamak için aynı algoritmaya başvurulmuştur. Şifrelenmiş toplamlar Çizelge 3.3'teki "Toplam" değerlerine karşılık gelmektedir. Bu değerler, araştırmacının doğrusal regresyon analizi için ihtiyaç duyduğu değerlerdir. Araştırmacı, şifrelenmiş toplamları aldıktan sonra, kendi özel anahtarını kullanarak bu toplamların şifresini çözer ve regresyon değişkenlerini bulmak için Denklem 2.3 ve 2.4'e yerleştirir.

Akıllı sözleşmeye gelen şifrelenmiş x değerlerini eklemek için homomorfik toplama işlevinin sözde kodu aşağıdaki gibidir:

Çizelge 3.7: Akıllı Sözleşmedeki ‘x’ Değerleri Homomorfik Toplam Fonksiyonu

```
// n2: Homomorfik açık anahtarın n*n değeridir.
// sigma_xEncBigNum: Tüm veri sahiplerinden gelen kümülatif şifreli x değerini temsil eder.

BigNumber sigma_xEncBigNum = BigNumbers.one();
function set_xEnc(bytes memory xSumEnc) public {
    require(msg.sender == addr_dataOwner);
    BigNumber memory xSumEnc_BigNum = BigNumbers.init(xSumEnc, false, 512);
    BigNumber memory result = BigNumbers.modmul(sigma_xEncBigNum, xSumEnc_BigNum, n2);
    sigma_xEncBigNum = BigNumbers.init(result.val, false, 512);
}
```

Yukarıdaki kod, veri sahiplerinden gelen şifreli değerlerin homomorfik toplama işlemi için yazılmıştır. Diğer homomorfik toplama işlemleri için de benzer kod parçaları kullanılmıştır.

Kod içerisindeki en önemli noktalardan biri, akıllı sözleşmeye gelen şifreli x değerlerinin tek tek saklanmamasıdır. Her veri sahibinden gelen şifreli x değerleri Çizelge 3.7’de yer alan ‘sigma_xEncBigNum’ değişkenine eklenerek işlem görmektedir. Bu sayede veri sahiplerinden gelen veriler şifreli bile olsa, akıllı sözleşmede bu verilere ayrı ayrı erişmek mümkün olmayacaktır.

4. DENEY ve DENEY SONUÇLARI

Bu bölümde, farklı anahtar uzunlukları ile test edilen sistemin sonuçları paylaşılmıştır. Anahtar uzunluğu olarak 256, 512, 1024 ve 2048 bitlik anahtar uzunlukları kullanılmıştır. Test kapsamında iki farklı ölçüm yapılmıştır.

Birinci ölçülen değer (Bölüm 4.1), bir veri sahibinin şifreli toplam sensör değerlerini akıllı sözleşmeye göndermesi ve akıllı sözleşmede bu değerlerin homomorfik toplam işlemine girmesini kapsamaktadır. Test senaryosunda hesaplanan ikinci değer (Bölüm 4.2) ise bu işlemler sırasında harcanan gaz değerleridir.

4.1 Sözleşmedeki Homomorfik Toplama İşlem Süreleri

Bu bölümde, bir Ethereum test ağında farklı anahtar bit uzunluklarına göre homomorfik şifrelenen verilerle yapılan işlemlerinin süre bazında değişimleri incelenmiştir. Kurgulanan bu test senaryosu dört farklı veri sahibinden gelen verileri içermektedir. Her deney toplamda beş kez tekrarlanmıştır. Bir deneyin tamamlanması, Şekil 3.3'deki senaryonun dört veri sahibi için de tamamlanması ve en sonunda araştırmacının doğrusal regresyon hesabını yapabilmesi olarak düşünülmüştür.

Her deneyde dört farklı veri sahibinin sisteme veri gönderim süresi ayrı ayrı ölçülmüş ve ortalama işlem süresi o deneyin ortalama süresi olarak not edilmiştir. Bu test senaryosunun sonucu olarak beş farklı deneyin ortalama veri sahibi işlem sürelerinin ortalaması alınmıştır. Her çizelgenin altında hesaplanan bu ortalama değer, ilgili anahtar uzunluğuna sahip bir işlemde bir veri sahibinin ne kadar sürede verilerini akıllı sözleşmeye gönderdiği ve homomorfik toplamın hesaplanabildiğini göstermektedir. Senaryo gereği tüm veri sahipleri akıllı sözleşmeye eş zamanlı işlem gönderdikleri için, aslında deney toplam sürelerine bakarak yorum yapılmamıştır.

256-bit (Çizelge 4.1), 512-bit (Çizelge 4.2), 1024-bit (Çizelge 4.3) ve 2048-bit (Çizelge 4.4) anahtar uzunluğundaki işlemler için beş adet deney yapılmıştır. Veri sahiplerinin blokzincir ağına gönderdiği şifreli verilerin homomorfik toplama yapılarak ağa kaydedilmesi için geçen zamanlar saniye cinsinden ilgili çizelgelerde

gösterilmiştir. Veri sahiplerinin işlem sürelerinin ortalamasını alarak her anahtar uzunluğu için gereken ortalama işlem süresi hakkında fikir yürütmek mümkündür.

Çizelge 4.1: 256-bit Anahtar Uzunluğu İşlem Süreleri (saniye)

Homomorfik Toplama Deney No	VeriSahibi1	VeriSahibi2	VeriSahibi3	VeriSahibi4	Deneyin Süresi
1	15,710	12,780	14,768	10,817	54,075
2	8,664	10,697	9,686	54,882	83,929
3	18,779	16,717	11,713	5,815	53,024
4	4,732	10,824	33,790	7,672	57,018
5	20,695	6,886	9,698	15,711	52,990
Ortalama İşlem Süresi	13,716	11,581	15,931	18,979	60,207

Tüm veri sahiplerinin işlem sürelerinin ortalaması 256-bit anahtar uzunluğu için;

(13,716 + 11,581 + 15,931+ 18,979) / 4 = 15,052 (15 saniye 52 milisaniye) saniyedir.

Çizelge 4.2: 512-bit Anahtar Uzunluğu İşlem Süreleri (saniye)

Homomorfik Toplama Deney No	VeriSahibi1	VeriSahibi2	VeriSahibi3	VeriSahibi4	Deney Süresi
1	18,794	8,692	10,667	15,782	53,935
2	6,638	5,668	8,753	7,647	28,706
3	24,788	26,802	7,686	26,757	86,033
4	10,899	14,195	12,899	14,659	52,652
5	11,733	12,775	13,768	22,717	60,993
Ortalama İşlem Süresi	14,570	13,626	10,755	17,512	56,464

Tüm veri sahiplerinin işlem sürelerinin ortalaması 512-bit anahtar uzunluğu için;

(14,570 + 13,626 + 10,755+ 17,512) / 4 = 14,116 (14 saniye 116 milisaniye) saniyedir.

Çizelge 4.3: 1024-bit Anahtar Uzunluğu İşlem Süreleri (saniye)

Homomorfik Toplama Deney No	VeriSahibi1	VeriSahibi2	VeriSahibi3	VeriSahibi4	Deney Süresi
1	42,949	9,720	12,740	15,765	81,174
2	7,818	9,743	4,685	23,911	46,157
3	14,792	4,698	35,926	27,828	83,244
4	16,738	17,742	25,125	12,714	72,319
5	11,721	7,734	12,858	8,820	41,133
Ortalama İşlem Süresi	18,804	9,927	18,267	17,808	64,805

Tüm veri sahiplerinin işlem sürelerinin ortalaması 1024-bit anahtar uzunluğu için;

(18,804 + 9,927 + 18,267+ 17,808) / 4 = 16,201 (16 saniye 201 milisaniye) saniyedir.

Çizelge 4.4: 2048-bit Anahtar Uzunluğu İşlem Süreleri (saniye)

Homomorfik Toplama Deney No	VeriSahibi1	VeriSahibi2	VeriSahibi3	VeriSahibi4	Deney Süresi
1	6,675	24,953	13,768	3,673	49,069
2	23,872	14,719	5,693	17,892	62,176
3	15,791	14,733	5,789	14,888	51,201
4	3,881	19,966	39,923	14,759	78,529
5	27,907	4,030	11,733	14,721	58,391
Ortalama İşlem Süresi	15,625	15,680	15,381	13,187	59,873

Tüm veri sahiplerinin işlem sürelerinin ortalaması 2048-bit anahtar uzunluğu için;

(15,625 + 15,680 + 15,381+ 13,187) / 4 = 14,968 (14 saniye 968 milisaniye) saniyedir.

4.2 Harcanan Gaz Ücretleri

Yapılan deneylerde, blokzincirine gönderilen işlemlerin harcadığı toplam gaz ücretleri her veri sahibi ve anahtar uzunluğu için Çizelge 4.5'te gösterilmiştir. Her anahtar uzunluğu için beş adet deney yapılmış, deneylerde dört veri sahibinin yaptığı işlemlerin harcadığı gaz ücretleri paylaşılmıştır.

Harcanan gaz değerleri hassasiyeti virgülden sonraki dört haneye kadar değerlendirmeye alınmıştır. Gaz değerlerinin hassas ölçümlerine, tüm Goerli Ethereum test ağının işlem kayıtlarını tutan Goerli Etherscan sitesinden erişmek mümkündür [26]. Etherscan sitesindeki gösterim biçimine göre değerler Ether para birimi cinsindendir ve işlem ücreti hesabı şu şekildedir:

(Gaz Fiyatı) X (İşlem tarafından kullanılan gaz miktarı) = Txn(işlem)

Bu işlem hesabına göre, işlemlerin harcadığı gaz miktarı ve gaz başına harcanan ether çarpıldığı zaman, bu işlem için ne kadar ether harcandığı bulunur. Çizelge 4.5'te her bir işlem için harcanan gaz ücreti paylaşılmıştır.

Çizelge 4.5: Deneylerdeki İşlemlerin Harcadıkları Gaz Ücretleri

Anahtar	Deney No	VeriSahibi1	VeriSahibi2	VeriSahibi3	VeriSahibi4
256-bit	Deney 1	0,0017	0,0012	0,0012	0,0011
	Deney 2	0,0011	0,0011	0,0011	0,0010
	Deney 3	0,0011	0,0011	0,0011	0,0011
	Deney 4	0,0011	0,0010	0,0010	0,0010
	Deney 5	0,0010	0,0010	0,0010	0,0010
512-bit	Deney 1	0,0027	0,0015	0,0015	0,0014
	Deney 2	0,0015	0,0015	0,0015	0,0015
	Deney 3	0,0015	0,0014	0,0015	0,0015
	Deney 4	0,0016	0,0017	0,0018	0,0018
	Deney 5	0,0018	0,0017	0,0017	0,0017
1024-bit	Deney 1	0,0068	0,0033	0,0033	0,0032
	Deney 2	0,0032	0,0030	0,0029	0,0030
	Deney 3	0,0033	0,0033	0,0030	0,0031
	Deney 4	0,0029	0,0029	0,0028	0,0030
	Deney 5	0,0032	0,0032	0,0030	0,0028
2048-bit	Deney 1	0,0098	0,0040	0,0039	0,0039
	Deney 2	0,0041	0,0040	0,0039	0,0041
	Deney 3	0,0040	0,0040	0,0039	0,0042
	Deney 4	0,0040	0,0039	0,0039	0,0040
	Deney 5	0,0038	0,0040	0,0039	0,0042

Bu gaz ücretlerinin, hangi gaz miktarı ve hangi gaz fiyatının çarpımları sonucu oluştuğu, Goerli Etherscan hesabından ayrıntılı bir şekilde incelenebilir. Çizelgelerde yer alan gaz ücretlerinden birini, örnek teşkil etmesi açısından gaz fiyatı ve gaz miktarı açısından incelemek gerekirse: VeriSahibi4'ün Deney5 te yaptığı 1024-bit anahtar uzunluğu kullanılan işlemdeki harcadığı toplam gaz ücreti 0,0028 Ether'dir.

Öncelikle Goerli Etherscan adresinde ilgili işlemi buluyoruz (Resim 4.1):

Resim 4.1: Goerli Etherscan Üzerinde Görüntülenen İşlem

Resim 4.1'de yer alan sol taraftaki işlem ayrıntıları simgesine tıkladığımızda Resim 4.2'deki arayüzü görüyoruz.

Resim 4.2: Gaz Ücreti Hesabı için Önemli Değerler

İlgili işlem için harcanan gaz ücretini hesaplarken Resim 4.2'de gösterilen kullanılan gaz miktarı ve birim gaz fiyatını çarptığımızda sonuç:

2,840,236.42090342 Gwen ~= 0.00284023 Ether çıkmaktadır.

5. DEĞERLENDİRMELER

Anahtar uzunluğunun blokzincir ağına yapılan isteklerdeki etkisini araştırmak için iki farklı türde sonuç verisine odaklanılmıştır; işlem süresi ve işlemler için harcanan gaz değerleri. İşlem süreleri sırasıyla 256 bit, 512 bit, 1024 bit ve 2048 bit anahtar boyutları için Çizelge 4.1-4.4 tablolarında kayıt edilmiştir. Gaz ücretleri ise her veri sahibinin işlemi sonucunda harcanan gaz ücretleri Ether cinsinden Çizelge 4.5'te kaydedilmiştir. Bu bölümde her iki türdeki sonuçlar değerlendirilmiş ve ayrıca tüm sistemin güvenlik açısından sağladığı avantajlar incelenmiştir.

Farklı anahtar uzunluklarının Bölüm 4.1'de yer alan ortalama işlem süreleri:

Anahtar uzunluğu 256-bit iken blokzincir işlemleri 15,052 saniye

Anahtar uzunluğu 512-bit iken blokzincir işlemleri 14,116 saniye

Anahtar uzunluğu 1024-bit iken blokzincir işlemleri 16,201 saniye

Anahtar uzunluğu 2048-bit iken blokzincir işlemleri 14,968 saniye olarak ölçülmüştür.

Ethereum Goerli ağına gönderilen işlemlerin işlem süresi 15-30 saniye arasındadır. Bu saniye değeri, Ethereum ağına gönderilen bir işlemin ağ tarafından işlenme süresidir. Yapılan ölçümler ve ortalama işlem sonuçları da yukarıdaki ortalama değerler dikkate alındığında bu sürenin doğruluğunu kanıtlar niteliktedir. Anahtar uzunlukları arasında ciddi bir fark olmasına rağmen işlem süreleri arasında kayda değer bir fark görülmemiştir. Bu noktada yapılan yorum; veri boyutunun işlem sürelerinin üzerinde bir etkisinin olmadığı yönündedir. Kaynak [27]'deki çalışmanın sonuçları da bulunan bu sonuçları doğrular niteliktedir.

İşlemlerde, anahtar uzunluğuna orantılı olarak gaz ücretlerinde de bir artış yaşandığı Çizelge 4.5'te gösterilmiştir.

Farklı anahtar uzunluklarının Bölüm 4.2'de yer alan ortalama gaz ücretleri:

Anahtar uzunluğu 256-bit iken blokzincir işlemleri 0.0011 Ether

Anahtar uzunluğu 512-bit iken blokzincir işlemleri 0.0016 Ether

Anahtar uzunluğu 1024-bit iken blokzincir işlemleri 0.0032 Ether

Anahtar uzunluğu 2048-bit iken blokzincir işlemleri 0.0041 Ether olarak ölçülmüştür.

Bir işlemin içindeki veri miktarı arttıkça, bu veriyi yönetmek için gereken ara işlemlerin sayısı arttığı için işlemin harcadığı gaz miktarında bir artış yaşanır. Buna paralel olarak, gaz ücretlerinde veri büyüklüğü arttıkça bir artış beklenmektedir. Nitekim Çizelge 4.5'te de görüldüğü gibi işlemlerin içindeki şifreli metinlerin oluşturulduğu anahtar büyüklüğü arttıkça veri miktarına bağlı olarak harcanan gaz değeri de artmıştır. Bu artan değer, işlemin blokzincirinin işlem onayı mekanizmasından ziyade; akıllı sözleşmedeki işlem yükünün artmasına bağlı olarak yaşanan bir işlem artışı olduğu düşünülmüştür. Bunun sonucu olarak; veri miktarı arttıkça işlem onaylanma sürelerinde artış olmamasına rağmen harcanan gaz miktarında düzenli bir artış gözlemlenmiştir.

Bu çalışmanın en önemli noktalarından biri ise, blokzincir işlemlerine getirilen ekstra güvenlik önlemleridir. Getirilen güvenlik önlemlerini ve etkilerini sıralamak gerekirse;

1. Fonksiyonların belirli adreslere özel yazılması ile akıllı sözleşmeye ve akıllı sözleşmedeki ilgili fonksiyonlara sadece izinli adreslerden işlem isteği atılabilabilmesi sağlanmıştır.

2. Uçbirimlerden yani veri sahiplerinden gelen toplam sensör verilerinin şifreli olmasından dolayı blokzinciri asla verilerin açık hallerine erişim sağlayamamaktadır.

3. Homomorfik toplama işlemi sayesinde blokzincirindeki hiçbir şifreli veri blokzinciri içinde çözülmemektedir.

4. Uçbirimlerden şifreli değerler akıllı sözleşmeye geldikçe, akıllı sözleşme içerisindeki kümülatif değişkene eklenerek tutulurlar. Böylece şifreli olsa bile uçbirimlerden gelen veriler ayrı bir şekilde akıllı sözleşme içinde tutulmazlar.

5. Uçbirimlerden gelen beş farklı şifreli veri ayrı ayrı işlemlerle akıllı sözleşmeye kaydedilmez. Tek seferde bir işlemde tüm verilerin gönderilmesiyle akıllı sözleşmeye beş farklı şifreli veri de eklenmiş olur. Böylece birbirlerine bağlı işlem değişkenlerinden birinin kaybolması durumunda oluşacak değer bozukluklarının önüne geçilmiş olunur.

Blokzincir teknolojisinin doğasında bulunan; kurcalamaya karşı direnç (tamper resistance), hata toleransı (fault tolerance), bütünlük (integrity), kullanılabilirlik (availability) ve veri işlemlerinin denetlenebilirliğini (auditability) yanında; geliştirilen bu projeyle birlikte yukarıda beş maddede belirtilen güvenlik önlemleri de sağlanmıştır. Dolayısıyla bu proje, getirdiği güvenlik yaklaşımlarıyla uygulamada avantaj sağlamaktadır.

6. SONUÇ ve ÖNERİLER

Geliştirilen projede, doğrusal regresyon analizi hesaplaması esas alınmıştır. Bu projede aynı yaklaşımla yapay zeka ve makine öğrenimi algoritmaları da dahil olmak üzere diğer veri analizlerini içerecek şekilde genişletilebilir.

Paillier şifreleme sistemi, toplama işlemine izin veren kısmen homomorfik bir sistemdir, dolayısıyla diğer işlemleri içeren daha karmaşık analizlerin gerekli olduğu sistemler için uygun değildir. Bu gibi karmaşık kullanım alanları için, blockchain teknolojisi üzerinden Paillier dışındaki diğer homomorfik algoritmalar da incelenebilir.

Bu çalışmadaki senaryoda, bir araştırmacı için tasarlanmış yalnızca bir akıllı sözleşme vardır. Genişletilmiş bir senaryo olarak sisteme birden fazla araştırmacı eklenebilir. Bu, şu iki yoldan biriyle yapılabilir:

1. Her yeni araştırmacı sisteme dahil olmak istediğinde, o araştırmacıya özel yeni bir akıllı sözleşme devreye alınır: Bunun için araştırmacılardan gelen istekleri kabul eden ve her araştırmacı için yeni bir akıllı sözleşme oluşturan bir yönetici akıllı sözleşmesi oluşturulabilir. Akıllı sözleşme, hangi araştırmacı için oluşturuldu ise o araştırmacının genel anahtarını saklar,
2. Bir akıllı sözleşme mevcuttur. Tüm araştırmacılar bu akıllı sözleşmeye genel anahtarlarını kaydederler.

Uygulamaya bağlı olarak, her birinin faydaları olabilir: İlk yöntem, yönetim görevlerini ayırma avantajına sahiptir. Farklı türde veri ve analizlerle ilgilenen araştırmacılar için kullanılırsa, ilk yöntem onlar için oluşturulan akıllı sözleşmenin kolayca özelleştirilmesine olanak sağlayacaktır. Sistemin verilerle benzer türde analizlerle ilgilenen araştırmacılar için kullanıldığı durumlarda, tüm işlevleri aynı akıllı sözleşmede tutmak, sözleşme ve taleplerin yönetimini kolaylaştırabilir. Bu yöntemde dikkat edilmesi gereken husus, anahtarları yönetmeyi kolaylaştırmak için farklı fonksiyonların eklenmesi gerekliliğidir.

Tüm bu öneriler, belirli konuların derinleşmesi ile daha da detaylandırılabilir.

KAYNAKLAR

[1] **Nakamoto, S.,** (2008).Bitcoin: A peer-to-peer electronic cash system.

[2] **Krichen, M., Ammi, M., Mihoub, A., ve Almutiq, M.,** (2022). Blockchain for modern applications: A survey, *Sensors*, cilt 22, no. 14. p. 5274, 2022.

[3] **Hilary, G.,** (2018). Blockchain: Security and Confidentiality. *SSRN Electronic Journal*.

[4] **Bozduman, H. Ç., ve Afacan E.,** (2020). Simulation of a homomorphic encryption system, *Applied Mathematics and Nonlinear Sciences*, cilt 5, no. 1, pp. 479-484.

[5] **Regueiro, C., Seco, I., Diego, S., Lage, O., ve Etxebarria, L.,** (2021). Privacy-enhancing distributed protocol for data aggregation based on blockchain and homomorphic encryption, *Information Processing & Management*, cilt 58, no. 6, p. 102745.

[6] **Mitra, A., Bera, B., Das, A. K., Jamal S. S., ve I. You, I.,** (2023). Impact on blockchain-based AI/ML-enabled big data analytics for Cognitive Internet of Things Environment, *Computer Communications*, cilt 197, pp. 173-185.

[7] **Loukil, F., Ghedira-Guegan, C., Boukadi, K., veBenharkat, A. N.,** (2021). Privacy-preserving IOT data aggregation based on blockchain and homomorphic encryption, *Sensors*, cilt 21, no. 7, p. 2452.

[8] **Umar, B. U., Olaniyi, O. M., Olajide, D. O., ve Dogo, E. M.,** (2022). Paillier cryptosystem based Chainnode for secure electronic voting, *Frontiers in Blockchain* , cilt 5.

[9] **Ghadamyari, M.,** (2019). Privacy-Preserving Statistical Analysis of Health Data Using Paillier Homomorphic Encryption and Permissioned Blockchain, *Electronic Thesis and Dissertations*.

[10] **Gascon, A., Schoppmann, P., Balle, B., Raykova, M., Doerner, J. Zahur, S., ve Evans, D.,** (2017). Privacy-Preserving Distributed Linear Regression on High-Dimensional Data, *Proceedings on Privacy Enhancing Technologies*, pp. 345-364.

[11] **Silva, F. N., Vanin, L., Policarpo, M., ve Rosa Righi, R.,** (2022). A Blockchain-Based End-to-End Data Protection Model for Personal Health Records Sharing: A Fully Homomorphic Encryption Approach, *Sensors*.

[12] **López Delgado, J. L.,Bermejo, J. A. Á., ve López Ramos J. A.,** (2022). Homomorphic Asymmetric Encryption Applied to the Analysis of IoT Communications, *Wireless Sensor Networks towards the Internet of Things*.

[13] **Zhang, H., Gao, P.,Yu, J.,Lin, J., ve Xiong, N. N.,** (2022). Machine Learning on Cloud With Blockchain: A Secure, Verifiable and Fair Approach to Outsource the Linear Regression, *IEEE Transactions on Network Science and Engineering* , cilt 9, no. 6, pp. 3956 - 3967.

[14] **Chena, B., ve Zhenga, X.,** (2021). Implementing Linear Regression with Homomorphic Encryption, *International Conference on Identification, Information and Knowledge in the internet of Things,*, pp. 324-329.

[15] **V, A., ve Varghese, L.,** (2022). Machine Learning With Homomorphic Encryption For Linear Regression Model,» *Proceedings of the National Conference on Emerging Computer Applications (NCECA)*, cilt 4, no. 1.

[16] **Akavia, A., Shaul, H., Weiss, M., ve Yakhini, Z.,** (2019). Linear-Regression on Packed Encrypted Data in the Two-Server Model, %1 içinde *WAHC'19: Proceedings of the 7th ACM Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, New York.

[17] ethereum.org, [Erişildi: 29 January 2023]. Ethereum Whitepaper, Ethereum Foundation, https://ethereum.org/en/whitepaper/.

[18] **Team, S.,** (2022). Solidity, 2022. https://soliditylang.org/.

[19] Konsensus Mekanizmaları https://ethereum.org/en/developers/docs/consensus-mechanisms/.

[20] ethereum.org, (2022). https://ethereum.org/en/developers/docs/gas/.

[21] **Wood, D. G.,** (2022). Ethereum: A Secure Decentralised Generalised Transaction Ledger, *Berlin Version*.

[22] **Gentry, C.,** (2009).A Fully Homomorphic Encryption Scheme, Standford University.

[23] **Sridokmai, T., ve Prakancharoen, S.,** (2015). The homomorphic other property of Paillier Cryptosystem, *International Conference on Science and Technology*.

[24] **Firoorg,** (2023). Firoorg/solidity-bignumber: Full BigNumber library implementation for solidity, : https://github.com/firoorg/solidity-BigNumber.

[25] **Peterolson,** Peterolson/BigNumber.js: An arbitrary length integer library for Javascript, https://github.com/peterolson/BigInteger.js.

[26] **Ethereum,** (2023). Goerli Testnet Explorer, https://goerli.etherscan.io/.

[27] **Reilly, E.,** (2019). An Ethereum-Based, Integrity-First Communication Protocol for IoT Devices, Massachusetts Institute of Technology.

[28] **Mutlu, Z. D., Kurt Peker, Y., ve Selçuk, A. A.** (2023) Blockchain-based Privacy Preserving Linear Regression, ICONCS, Karabük.