T.C.
DÜZCE ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

BLOKZİNCİR YÖNTEMİYLE SUNUCU-İSTEMCİ ARASI GÜVENLİ WEB HABERLEŞMESİ

DURDU ÖZDEN ONAR

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

DANIŞMAN
PROF. DR. RESUL KARA

DÜZCE, 2021

T.C.
DÜZCE ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

BLOKZİNCİR YÖNTEMİYLE SUNUCU-İSTEMCİ ARASI GÜVENLİ WEB HABERLEŞMESİ

Durdu ÖZDEN ONAR tarafından hazırlanan tez çalışması aşağıdaki jüri tarafından Düzce Üniversitesi Lisansüstü Eğitim Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Tez Danışmanı

Prof. Dr. Resul KARA

Düzce Üniversitesi

Jüri Üyeleri

Prof. Dr. Resul KARA
Düzce Üniversitesi _____

Prof. Dr. Pakize ERDOĞMUŞ
Düzce Üniversitesi _____

Dr. Öğr. Üyesi Nihan KAZAK ÇERÇEVİK
Bilecik Şeyh Edebali Üniversitesi _____

Tez Savunma Tarihi: 27/08/2021

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.

27 Ağustos 2021

Durdu ÖZDEN ONAR

TEŞEKKÜR

Yüksek lisans öğrenimimde ve bu tezin hazırlanmasında gösterdiği her türlü destek ve yardımdan dolayı çok değerli hocam Prof. Dr. Resul KARA'ya en içten dileklerimle teşekkür ederim.

Bu çalışma boyunca yardımlarını ve desteklerini esirgemeyen sevgili eşim Mehmet ONAR'a, aileme ve çalışma arkadaşlarıma sonsuz teşekkürlerimi sunarım.

27 Ağustos 2021 **Durdu ÖZDEN ONAR**

İÇİNDEKİLER

ŞEKİL LİSTESİ vii

ÇİZELGE LİSTESİ viii

KISALTMALAR ix

ÖZET x

ABSTRACT xi

1. GİRİŞ 1

2. GÜVENLİ WEB SUNUCU İSTEMCİ HABERLEŞMESİ 4

2.1. HTTP NEDİR ? 4
2.2. HTTP METODLARI 5
2.2.1. GET 5
2.2.2. POST 6
2.2.3. PUT 6
2.2.4. HEAD 6
2.2.5. DELETE 7
2.2.6. OPTIONS 7
2.3. HTTP DURUM KODLARI 7
2.4. SSL/TLS YÖNTEMİ 8
2.5. SSL TEKNOLOJİSİNİN ÇALIŞMA MEKANİZMASI 9
2.5.1. SSL Sertifika 10
2.5.2. Doğrulama Türüne Göre SSL Sertifika Çeşitleri 11
2.5.2.1. *DV (Domain Validation) SSL* 11
2.5.2.2. *OV (Organizational Validation) SSL* 11
2.5.2.3. *EV (Extended Validation) SSL* 11
2.6. ŞİFRELEME YÖNTEMLERİ 11
2.6.1. Simetrik Şifreleme 11
2.6.2. Simetrik Şifreleme Algoritmaları 12
2.6.2.1. *Doğrusal Şifreleme (Affine Cipher)* 13
2.6.2.2. *Hill Şifreleme Yöntemi* 13
2.6.2.3. *Sezar Şifreleme* 14
2.6.2.4. *Permutasyon Şifreleme* 14
2.6.3. Asimetrik Şifreleme 15
2.6.4. Rivest-Shamir-Adleman(RSA) Şifreleme Algoritması 16
2.7. PEER TO PEER (EŞE EŞ) AĞLAR 16

3. BLOKZİNCİR TEKNOLOJİSİ 18

3.1. BLOK YAPISI 19
3.2. BLOKCHAIN ÖZELLİKLERİ 20
3.2.1. Dağıtık Veri Tabanı Mimarisi 20
3.2.2. Şeffaflık 20
3.2.3. Güvenlik 21
3.2.4. Uçlar Arası İletişim 21

4. BLOKZİNCİR İLE GÜVENLİ WEB SUNUCU-İSTEMCİ HABERLEŞMESİ 22

4.1. BLOKZİNCİRİ KULLANARAK GÜVENLİ HTTP BAĞLANTISI 22
4.2. GÜVENLİ İLETİŞİM İÇİN BLOKZİNCİRİ ALT YAPISI 23
4.3. SSLCHAIN 23
4.4. SSLCHAIN BLOK İÇERİĞİ 25
4.5. SSLCHAIN ÇALIŞMA YÖNTEMİ 29
5. BULGULAR 35
6. SONUÇLAR VE ÖNERİLER 39
7. KAYNAKLAR 40
8. EKLER 42
8.1. EK 1: HTTP DURUM KODLARI 42
ÖZGEÇMİŞ 44

ŞEKİL LİSTESİ

Sayfa No

Şekil 2.1. GET metodu ile basit HTTP isteği 6
Şekil 2.2. Sertifika ile anahtar paylaşımı. 10
Şekil 2.3. Sertifika doğrulama zinciri. 11
Şekil 2.4. Substution algoritması anahtarı. 12
Şekil 4.1. SSLChain çalışma yapısı. 24
Şekil 4.2. SSLChain blok yapısı. 26
Şekil 4.3. Blokzincirinde oturum zincir yapısı. 28
Şekil 4.4. Oluşan ilk blok..... 30
Şekil 4.5. Sunucuya gönderilen ilk istek..... 31
Şekil 4.6. Key paylaşımı. 32
Şekil 4.7. İstemcinin key blok paylaşımı. 33
Şekil 4.8. SSLChain blokzinciri. 34
Şekil 5.1. Şifreli HTTP response içeriği 35
Şekil 5.2. Şifresiz request içeriği. 36

ÇİZELGE LİSTESİ

Sayfa No

Çizelge 5.3. 1 sunucu ve 2 istemciden oluşan ağ gecikmeleri..... 37
Çizelge 5.4. 1 sunucu ve 3 istemciden oluşan ağ gecikmeleri..... 37
Çizelge 5.5. 1 sunucu ve 4 istemciden oluşan ağ gecikmeleri..... 37

KISALTMALAR

AES	Gelişmiş Şifreleme Standardı – (Advanced Encryption Standard)
FTP	Dosya Transfer Protokolü – (File Transfer Protocol)
HTTP	Hiper Metin Transferi Protokolü – (Hyper Text Transfer Protocol)
HTTPS	Güvenli Metin Aktarım Protokolü – (Hyper Text Transfer Protocol Secure)
IP	İnternet Protokolü – (Internet Protocol)
NNTP	Ağ Haberleri Aktarım Protokolü – (Network News Transfer Protocol)
RSA	Şifreleme Yöntemi
SMTP	Elektronik Posta Gönderme Protokolü – (Simple E-posta Transfer Protocol)
SSH	Güvenli Kabuk – (Secure Shell)
SSL	Güvenli Giriş Katmanı – (Secure Sockets Layer)
TLS	Taşıma Katmanı Güvenliği – (Transport Layer Security)
URL	Tekdüzen Kaynak Bulucu – (Uniform Resource Loader)

ÖZET

BLOKZİNCİR YÖNTEMİYLE SUNUCU-İSTEMCİ ARASI GÜVENLİ WEB HABERLEŞMESİ

Durdu ÖZDEN ONAR
Düzce Üniversitesi
Lisansüstü Eğitim Enstitüsü, Bilgisayar Mühendisliği Anabilim Dalı
Yüksek Lisans Tezi
Danışman: Prof. Dr. Resul KARA
Ağustos 2021, 43 sayfa

Blokzincir adını bitcoin ile duyurmuş olsa da yapabilecekleri sadece kripto para ile sınırlı kalmayıp birçok teknolojide kullanılabilir. Blokzinciri, sürekli büyüyen bir veri yapısı olup dağıtık bir mimariye sahiptir. İşlemler değiştirilemez, yapılan işlemleri ve bir önceki bloğun adresi bloklara yazılır. Blokzinciri merkezi bir sistem olmayıp ağa katılan tüm katılımcıların zincirde bir halka olarak yer almasını sağlar. Bu çalışmada; günümüzde web sayfalarında istemci ve sunucu arasında iletişim kurmada kullanılan SSL/TLS yerine yeni bir yöntem olan blokzincir kullanılarak güvenli bağlantı kurulması önerilmiştir. Sunucu-istemci arasındaki anahtar paylaşımının yönetilmesinde blokzincirinin görev alacağı şekilde tasarım yapılmıştır. Adına SSLChain denilen yöntem kullanılarak istemci-sunucu arasında güvenli veri alışverişi gerçekleştirildiği deneysel çalışma ile gösterilmiştir.

Anahtar Kelimeler: Blokzinciri, Dağıtık mimari, Güvenlik, SSL, SSLChain.

ABSTRACT

CLIENT-SERVER SECURE WEB COMMUNICATION WITH BLOCKCHAIN METHOD

Durdu ÖZDEN ONAR
Düzce University
Institute of Graduate Studies, Department of Computer Engineering
Master's Thesis
Supervisor: Prof. Dr. Resul KARA
August 2021, 43 pages

Although the term blockchain has been heard with bitcoin, it can be used in many technologies, not just crypto money. Blockchain has a distributed architecture and is a constantly growing data structure. Data blocks cannot be changed, each block has its own data and the previous block's address data. Blockchain does not have a centralized system, it ensures that all participants to participate in the network are included in the chain as a link. In this paper; Secure connection has been proposed using a new method, blockchain can use instead of SSL/TLS, which is used to communicate between client and server on web pages. It has been designed in such a way that the blockchain will take part in the management of the key sharing between the server and the client. It has been demonstrated by an experimental study that secure data exchange between the client and the server is carried out using the method called SSLChain.

Keywords: Blockchain, Distributed architecture, Security, SSL, SSLChain.

1. GİRİŞ

İletişim kurma güdüsü insanlığın var oluşundan bu yana insanlar ile birlikte nesiller boyunca bizi takip etmiştir. Kayda alınan tarihsel olaylardan da anlaşılacağı üzere insanlığın bir süre sesli iletişim kuramadan sadece el kol ve yüz ifadeleri ile iletişim kurduklarını belirlense de bir süre sesli iletişim ardından sesli ve yazılı iletişimi kullandığı anlaşılmıştır. İnsan yapısında iletişim kurma içgüdüsünü taşıdığı gibi mahremiyeti de birlikte taşımıştır. Mahremiyet kelimesi farklı zamanlarda farklı anlamları üzerine yüklese de temelinde gizliliği benimser. "Mahremiyet" kelimesi kendisine akademik metinlerde "kişinin kendisine ilişkin bir alanın olması, kendi başına kalabilmesi, yalnız kendisinin bildiği ve sadece kendi istediği kişilerle paylaştığı özel bilgi ya da niteliklerine ilişkin doğal, insani bir hak" gibi tanımlar bulmuştur [1]. İnsanların en temel manada iletişimlerini mahremiyet altına alma konusunu değerlendirecek olursak sözle iletişim kuran iki kişinin iletişimlerini 3. kişilerden gizlemek için 3. kişilerin duyamayacağı bir şekilde konuşmaları gerekir. Bununla ilgili iletişimdeki kişiler çeşitli yöntemler geliştirebilir. Temel amaç iletişim kuran kişiler arasında kesintisiz bir aktarım gerçekleştirilmesini sağlamalarıdır. Aktarılan bilgilerin yetkisiz kişilerce erişilmesini engellemek aktarılan verilerin tamamının doğru şekilde anlaşılmasını sağlamak ve aktarılanların değiştirilmeden karşıya ulaştırılmasıdır [2]. Yöntemi nasıl olursa olsun iletişim sırasında gönderilen bilgi gizli, bütünlüğünü yitirmemiş ve erişilebilir olmalıdır. En ilkel yöntemden en gelişmiş yönteme kadar hepsinde bu kurallar geçerlidir.

1960'lı yıllarda ortaya çıkan ve 1990'lı yılların sonunda günümüze kadar aktif bir şekilde kullanılan internet ile birlikte iletişim teknikleri de dijitalleşmiş ve internet bir iletişim aracı haline gelmiştir [3]. İnternette iletişim ise belirli kurallara bağlanmış ve protokoller geliştirilmiştir [4]. Günümüzde bu protokolleri bilinçli ya da bilinçsiz olarak çok aktif bir şeklinde kullanılıyor. Bunların en başında gelen ise Hyper Text Transfer Protocol (HTTP) protokolüdür. HTTP, internette sunucular ve son kullanıcılar arasında bilgilerin nasıl aktarılacağına dair kurallar ve yöntemleri düzenleyen uygulama katmanında çalışan bir iletişim protokolüdür. Web sitesi görüntülemek ve üzerinde çeşitli işlemler yapmak için kullanılır [5]. İstemci sunucu arası metin transferi protokolüdür. İletişim sırasında herhangi bir şifreleme söz konusu değildir. Oluşan bu açığın kapatılması için Hyper Text

Transfer Protocol Secure (HTTPS) protokolü geliştirilmiş ve iletişim şifreli hale getirilmiştir.

2018 yılında Uluslararası IEEE veri madenciliği konferansında yayınlanan 'Certificate Transparency Using Blockchain' makalesinde IBM'in Hyperledger Fabric blockchain platformunda herhangi bir sertifika otoritesinden sertifika onayı almaksızın blockzincirinde sertifikayı üretmiştir [6].

Teknolojinin gelişmesi ile iletişim yöntemleri de gelişti. Günümüzde en popüler haberleşme aracı kuşkusuz internettir. İnternet, birçok bilgisayar sisteminin birbirine bağlı olduğu, dünya çapında yaygın olan ve sürekli büyüyen bir iletişim ağıdır. İnternet, insanların her geçen gün gittikçe artan "üretilen bilgiyi saklama/paylaşma ve ona kolayca ulaşma" istekleri sonrasında ortaya çıkmış bir teknolojidir. Bu teknoloji yardımıyla pek çok alandaki bilgilere insanlar kolay, ucuz, hızlı ve güvenli bir şekilde erişebilmektedir. İnternetin temelleri ARPANET (Gelişmiş Araştırma Projeleri Dairesi Ağı) tarafından atıldı. ARPANET üzerinden ilk mesaj 29 Ekim 1969'da saat 22:30'da gönderildi. Bu mesaj Kaliforniya Üniversitesi (UCLA) Profesörü Leonard Kleinrock gözetiminde UCLA'da yazılım üzerine eğitim gören Charley Kline tarafından UCLA'daki bir bilgisayardan Stanfort Üniversitesi'ndeki bir bilgisayara gönderildi. O zamanlar küçük bir grubun kullandığı internet günümüzde 4 milyardan fazla cihazı birbirine bağlayarak çok daha yaygın kullanıma sahip bir iletişim aracı olmuştur.

ARPA'nın ortaya çıkış amacında yüksek düzeyde güvenlik gerektiren ulusal endişeye yanıt vermekti [7].

İnternet, insanların sadece kişisel bilgisayarlarını kullanarak kurdukları iletişimin yanında otonom cihazlarında birbiri ile haberleşmesini sağlamaktadır. Mutlak insan kontrolünde olmasa da bazı akıllı cihazlarda bu büyük haberleşme ağı üzerinden bilgi alışverişinde bulunmaktadır. Bu büyük ve mükemmel bir ağın en büyük sorunu güvenliktir. İletişim sırasında bizi en çok endişeye sevk eden konu bilgilerimizin yetkisiz kişilerce okunup, değiştirilip, saptırmalara ve değişik zafiyetlere yol açmasıydı. İnternette iletişim ile ilgili birçok önlemler alınmaya çalışılmış ve akademik çalışmalar yapılmıştır.

İnternete bağlı birimlerin, birbirleriyle iletişim sağlamak amacıyla kullandıkları benzersiz numaralara İnternet Protokolü (IP) adresi adı verilmektedir [8]. Ağ katmanında çalışır ve paketleri IP adreslerine göre bir makineden diğerine iletir. Veri alışverişini sağlar. Mevcut internet oluşturulduğu anda güvenlik söz konusu değildi sistemi güvenli hale getirme

konusu da belirli zahmetler gerektiriyordu [9]. İnternet ağı büyüdükçe güvenliği de daha zahmetli hale gelmiştir. İletişimde güvenliğin sağlanması için kapsamlı protokoller geliştirilmiştir. İletişim kuran kişilerin kendi mahremiyetlerini korumaları için bilinçlendirme çalışmaları yapıldı. İnternet Protokolü'nde (IP), tasarım olarak bir kimlik doğrulaması bulunmamaktadır ve iletişim kuran taraflar kolaylıkla kandırılabilirler. Bu yüzden hassas verilerin kullanıldığı haberleşmelerde taraflar arasında kimlik doğrulaması sağlanmadan iletişime geçilmemelidir [12]. Mesajlaşma bankacılık ve finans belirli üst düzey güvenlik gerektiren uygulamalar da kullanıcılar kimlik doğrulaması aşamalarını geçerek kendilerini kanıtlarlar. İnternet Protokolü'nde bulunan bu açıklığı giderebilmek için SSL, TLS ve SSH gibi yaygın kullanılan internet güvenlik sistemleri kullanılmıştır. İnternette güvenli iletişimin hedefi, bilginin güvenilirliğini ve bütünlüğünü korumaktır [10].

Bu çalışmada HTTP'nin güvenli versiyonu olan HTTPS'e alternatif bir sistem önerilmiştir. Ağda sadece istemci sunucunun paydaş olmadığı genel ağdaki paydaşların da iletişimdeki şifrelemeye katıldığı blokzinciri teknolojisi kullanılarak güvenli HTTPS bağlantısına alternatif güvenli bir blokzinciri güvenliği önerilmiştir.

2. GÜVENLİ WEB SUNUCU İSTEMCİ HABERLEŞMESİ

2.1. HTTP NEDİR?

HTTP, Hypertext Transfer Protokol'ün kısaltmasıdır ve istemci ile web sunucusu arasında iletişim için kullanılan bir protokolün adıdır. HTTP, 1990 yılından beri Geniş Dünya Ağı (Worl Wide Web - WWW) üzerinde küresel bilgi girişimi ve erişimi için kullanılmaktadır [11]. Sunucuda bulunan verileri metin, görüntü, ses ya da video biçiminde aktarır. HTTP istemciden gelen bir istek ile çalışır. İstemci istekte bulunur buna karşılık sunucudan cevap verilir. Sunucu cevapları 80 numaralı porttan verir. Güvenli bağlantı için bu port 443'dür. HTTP'nin ilk sürümü olan HTTP/0.9 internet üzerinden ham verinin taşınması amaçlı, basit bir iletişim kuralıydı. Bu versiyonu günümüzde kullanılan 1.1 versiyonuna göre oldukça kısıtlı özelliklere sahipti. Bunlar;

- İstek yapısı tek satırdan oluşmaktadır. (GET/basit-html-sayfasi.html)
- Sadece GET metodu desteklenmektedir.
- Cevap tipi sadece hiper metin formatındadır.
- Bağlantı isteğe karşı verilen cevaptan sonra hemen kapanır.
- HTTP başlıklarını desteklemez. (hata kodları, içerik tipi)

RFC-1945 dahilinde tanımlanan HTTP/1.0, yeni özellikleri ile önceki sürümünün yeteneklerini genişletmiştir. Bunlar; taşınan verinin meta-bilgilerini ve istek/cevap semantiği düzenleyicilerini içeren ve MIME tipleri taşıyan mesajların taşınabilmesi gibi. HTTP/1.0 önceki versiyonlarına göre aşağıdaki özellikleri ile yenilenmiştir. MIME tipler farklı dosya uzantılarının tespit edilebilmesi için her dosya türüne tanımlanmış kimlik bilgileridir.

- Hem istek hem de yanıt hakkında zengin meta veriler içeren başlık alanları (HTTP versiyon numarası, durum kodu, içerik türü(json,xml))
- Sunucu cevapları: Hiper metinle sınırlı değil (İçerik Türü başlığı, düz HTML dosyaları dışındaki dosyaları iletme yeteneği sağladı. Örneğin komut dosyaları, stil sayfaları, medya)
- GET, HEAD, POST metotlarını desteklemektedir.
- Bağlantı isteğe karşı verilen cevaptan sonra hemen kapanır.

HTTP/1.1 olarak bilinen bu çalışmanın yapıldığı tarih itibari ile en yaygın kullanılan sürüm, iletişim kuralının güvenilir bir biçimde uygulanmasında ihtiyaç duyulan dizisel gereksinimleri içermekte ve 1.0 sürümüne sahip iletişim kuralından daha güvenli olarak görülmektedir.

- Sunulan kritik performans optimizasyonları ve özellik iyileştirmeleri- kalıcı ve ardışık bağlantılar, yığınlanmış aktarımlar, sıkıştırma/açma, içerik müzakereleri, sanal barındırma (birden çok etki alanını barındıran tek bir IP adresine sahip bir sunucu), önbellek desteği ekleyerek daha hızlı yanıt ve büyük bant genişliği tasarrufu.
- GET, HEAD, POS, PUT, DELETE, TRACE, OPTIONS metotlarını destekler.
- Bağlantı isteğe karşı verilen cevaptan sonra hemen kapatılmaz.

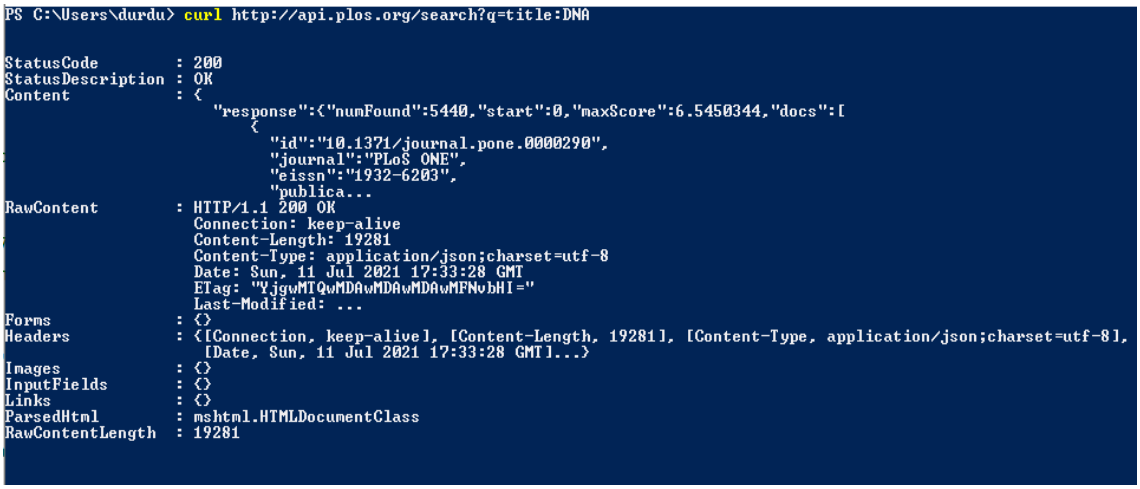
Birtakım uygulamalar, basit bir bilgi alışverişinden çok daha fazlasına ihtiyaç duyar. Bu ihtiyaçlar son kullanıcı ara yüzünün güncellenmesi ve etkileşimli olarak bilgi girişi gibi işlevleri de gerektirmektedir. HTTP, bir isteğin temel amacının ne olduğunu anlatan bir takım üstbilgi kullanımına izin vermektedir. Bir tek biçimli kaynak tanımlayıcısı, yer belirleyici ya da kaynak ismi tarafından sağlanan kaynağa, bir yöntemin uygulanışını bildiren bir dizi kural üzerine kurulmuştur. Gönderiler, Çok Amaçlı İnternet Posta Uzantıları tarafından tanımlandığı ve internet postasında kullanılan benzer bir biçimde aktarılmaktadır.

HTTP aynı zamanda, SMTP, NNTP ve FTP iletişim kurallarını destekleyen bir iletişim kuralıdır. HTTP sayesinde günümüzde internet tarafından sağlanan verilere erişmekte kolaylaşır.

2.2. HTTP METODLARI

2.2.1. GET

GET metodu sunucudaki servislere erişmek için kullanılan bir HTTP istek yöntemidir. GET metodu kullanılarak sunucudan URL üzerindeki yönlendirmeler veya koşullar ile veriler alınır. GET metodu kullanılırken uygulanacak olan koşullar URL ile gönderileceğinden hassas içerikli veriler bu yöntemde gönderilirken dikkat edilmelidir. GET metoduna örnek Şekil 2.1'de verildiği gibidir.



Şekil 2.1. GET metodu ile basit HTTP isteği.

GET metodu ile sunucuya gönderilecek olan veriler URL ve HTTP header ile gönderileceğinden metin gibi bir HTTP body gönderilemez. Yukarıdaki gibi bir istek ağlarda rahatlıkla izleneceğinden, hassas verilerin koşul olarak ekleneceği bir istemci sunucu iletişiminde tercih edilmemesi gereken bir yöntemdir.

2.2.2. POST

POST metodu istemciden sunucuya veri aktarmak için kullanılır. GET metoduna göre güvenli bir gönderim şeklidir. POST yöntemi, işlenecek verileri sunucuya gönderir. HTML formunu desteklemek için genellikle POST yöntemi kullanılır.

2.2.3. PUT

PUT metodu, POST metodu gibi istemciden sunucuya data göndermek için kullanılır. Ancak PUT metodunun tercih edilmesinin sebebi genellikle, kaynaktan mevcut olan verinin güncellenmesi içindir.

2.2.4. HEAD

HEAD metodu GET metoduna göre benzerlikler gösterir. GET metodunda sunucudan büyük miktarda veri transfer ederken HEAD metodunda ise daha küçük bilgileri sunucudan almak için kullanılır. Bu bilgiler servise ait bilgiler ya da ayar bilgiler olabilir. HEAD metodunun daha küçük veriler için kullanılması onu GET metodundan daha hızlı yapar.

2.2.5. DELETE

DELETE metodu istenirse sunucu yapılandırılması ile kapatılabilir, yani istemciden gönderilen DELETE metodu isteklerine metodun desteklenmediğine dair istemciye cevap gönderilebilir. DELETE metodu kaynaktan kayıt silmek için kullanılır.

2.2.6. OPTIONS

OPTIONS metodu, istekte bulunulan temel URL'in (sunucunun) desteklediği HTTP metotlarını listeler.

2.3. HTTP DURUM KODLARI

HTTP protokolü üzerinde iletişim istemci ve sunucu arasında gerçekleştiğinden iletişim sırasında istemcinin sürekli bilgilendirilmesi gerekmektedir. Bu bilgilendirmeler sunucudan gelen cevap içeriğinin yanı sıra iletişimde hataya sebep olan durumun kullanıcıya uygulama sunucusu tarafından iletilir. Sunucudan istemciye gönderilen ve RFC7231 dokümanın da tanımlanmış 3 haneli kodlar her zaman bir hata olduğunun göstergesi değildir. Bu mesajlar sunucudan istemciyi sürekli bilgilendirme için kullanılır.

HTTP durum kodları uygulamada kullanılacak mesajların seçiminde önemli bir rol almaktadır. Sunucudan gelen cevaba göre gönderilen isteğin durumu hakkında bilgi sahibi olabiliriz. Örneğin bir uygulamanın giriş ekranında kullanıcı adı ve şifresinin form içerisinde sunucuya gönderildikten sonra dönen cevabın yanı sıra HTTP durum koduna göre yeniden giriş ekranına ya da giriş ekranından yetki aldıktan sonra ilgili sayfaya yönlendirilmesini sağlar.

HTTP durum kodları, sunucudan gelen cevaba göre 5 gruba ayrılmıştır. Her grup durumu daha iyi özetleyen özel kodlara sahiptir. Bu kodlar temel olarak Bilgilendirici, Başarılı, Yönlendirme, İstemci Hatası, Sunucu Hatası olarak gruplandırılmıştır. RFC7231'de tanımlanmış bütün durum kodları EK-1'deki tabloda listelenmiştir.

1xx (Bilgilendirici)

1xx ile başlayan HTTP durum kodları HTTP/1.0 de tanımlanmamış sadece HTTP/1.1 versiyonunda tanımlanmıştır. İsteğin sunucuya iletildiğini işlemin başladığı hakkında bilgi mesajı içeren kodlardır. Burada gelen cevaplar isteğin başarılı olduğu anlamına gelmemelidir. Burada sadece sunucuya isteğin iletildiği bilgisi içerir.

2xx (Başarı)

2xx ile başlayan HTTP durum kodları genellikle sunucudan başarılı bir cevap döndüğüne ilişkin mesajları içeren kodlardır. Burada sunucuya gelen isteğin kurallara uygun olarak geldiğini ve sunucu tarafından isteğin anlaşıldığını ve sunucu tarafından işlemin tamamlanarak istemciye gönderildiğinin mesajlarını içerir.

3xx (Yönlendirme)

3xx ile başlayan HTTP durum kodları kaynağın geçici/kalıcı ya da herhangi bir sebepten dolayı kaynağın taşındığını bu nedenle yönlendirmenin olduğuna ilişkin sunucu tarafından gönderilen durum kodlarıdır.

4xx (İstemci Hatası)

4xx ile başlayan durum kodları istemciden kaynaklanan sorunlardan dolayı kaynağa erişilemediğine ilişkin sunucudan gönderilen kodlardır. Genellikle hata mesajlarını içerir. Bu grupta çok fazla hata durumu söz konusudur. Bunlar genellikle olmayan bir kaynağa erişme, olan kaynağa yetkisiz erişme, kaynağa istenmeyen bir metot ile erişme ya da gönderilen isteğin kurallara uymayan bir formatta gönderilmesi ile ortaya çıkar.

5xx (Sunucu Hatası)

Bu grupta hatalar uygulamadan ya da uygulama sunucusundan kaynaklanan hatalardır. Uygulama içinde kontrolsüz cevaplar sonucu verilen HTTP 500 durum kodu en bilinen koddur. Buradaki hatalar sunucu iç iletişim hatalarından kaynaklanabilir.

2.4. SSL/TLS YÖNTEMİ

Netscape tarafından 1994 yılında çıkarılan Secure Sockets Layer (SSL) diğer ismi ile Transport Layer Security (TLS) güvensiz ortamda verinin sunucu ve istemci arasında şifreli/güvenli bir şekilde haberleşmesini sağlamak için geliştirilen bir protokoldür [12]. Önceden bir şifreleme algoritması ile şifrelenerek ve sadece gönderilen alıcının çözebileceği, uygulama katmanı ile taşıma katmanı arasında bulunan verinin kriptografik işlemlerden geçerek istemciye ulaşmasını sağlar. Veriyi şifrelemek için genel anahtar veya diğer adıyla açık anahtar (public key), şifrelenen veriyi çözmek için özel anahtar veya diğer adıyla gizli anahtar (private key) kullanılır.

HTTP protokolündeki iletişimi SSL/TLS ile şifrelendiği için protokolün yeni adı HTTPS olmuştur. İstemci bir sunucuya erişmek istediğinde ilk olarak alan adının sertifikasını alır ve ulaşmak istediği sunucunun asıl gitmek istediği yer olduğu kesinleşmiş olur. İnternet üzerindeki iletişim bir mektuplaşmaya benzetilirse; gelen bir postanın doğru göndericiden geldiğinin bir kesinliği olamaz ya da gönderilen mektubun yolda üçüncü kişiler tarafından okunmayacağının bir garantisi olamaz. Bunun sebebi, içeriğinin herkesin anlayacağı bir şekilde gönderilmiş olmasıdır. İnternet ortamındaki web sayfalarının transferi için de aynı durum söz konusudur.

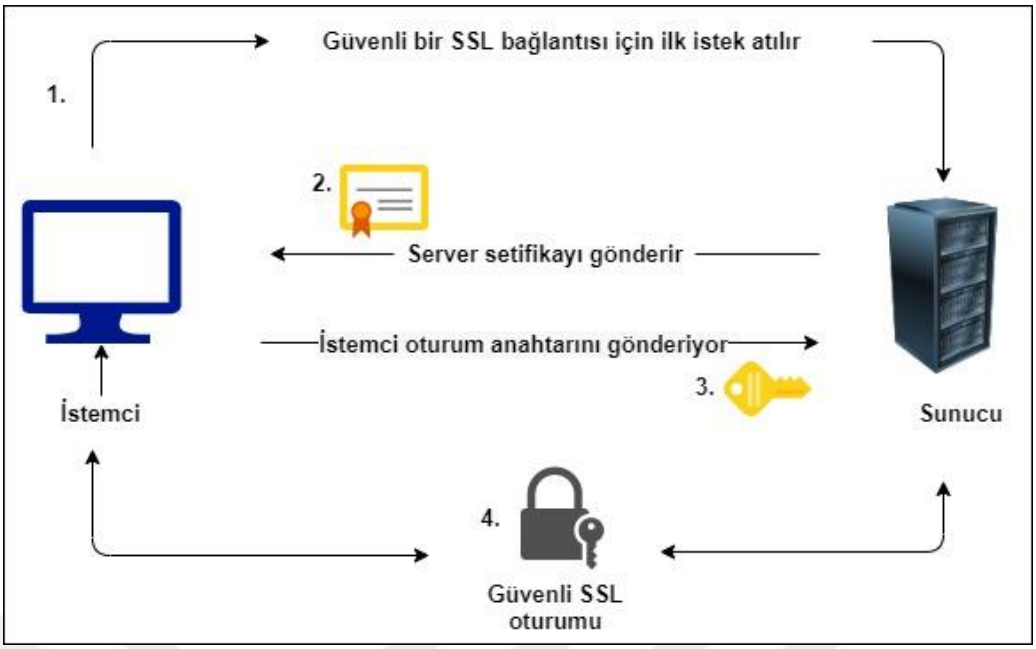
SSL/TLS'in ortaya çıkış motivasyonunda karşılıklı güven eksikliği vardır. İstemci bir istekte bulunmak istediğinde karşındaki kişinin ulaşmak istediği kişi olup olmaması güvensizliği bir sertifika ile aşılmaya çalışılmıştır. Sunucunun isteğe cevap olarak ortak anahtarı ile birlikte sertifika sunar. Fakat sertifikanın da doğrulanması gereklidir. Sertifikanın ispatlanması için bir veya daha fazla sertifika yetkilisinden oluşan güvenilir kuruluşlara ihtiyaç duyulur [13].

SSL sertifikaları, çok sayıda güvenli kuruluş tarafından sağlanmaktadır. SSL 'de anahtar uzunluğu en az 256 bit kullanılarak güvenli istemci-sunucu haberleşmesi gerçekleştirilebilir [14].

2.5. SSL TEKNOLOJİSİNİN ÇALIŞMA MEKANİZMASI

İstemci bağlanmak istediği sunucuya Transmission Control Protocol (TCP) 443 portu üzerinden istek gönderir. Sunucunun yapılandırılmasına göre bu port değiştirilebilir ancak varsayılan port numarası 443'tür. Sunucu, istemcinin bu isteğine karşılık alan adına tanımlı sertifikasını istemciye gönderir. İstemci sertifikayı üreten kurumunun sağladığı servisler ile sertifikayı doğrular. Sunucudan gelen sertifika ile birlikte açık anahtar da alınmış olur. Sertifikanın geçerlilik kontrolleri tamamlandıktan sonra, istemci, oturum için ürettiği iletişimde kullanılacak anahtarı sunucudan gelen açık anahtar ile şifreleyerek sunucuya gönderir. İstemci tarafından oluşturulan bu anahtar artık oturum boyunca iletişimde kullanılır. Sunucu istemciden gelen şifrelenmiş metni kendi gizli anahtarı ile açarak oturumdaki anahtarı elde etmiş olur. Bu aşamadan sonra istemci ve sunucu karşılıklı iletişimlerinde bu anahtarı kullanırlar.

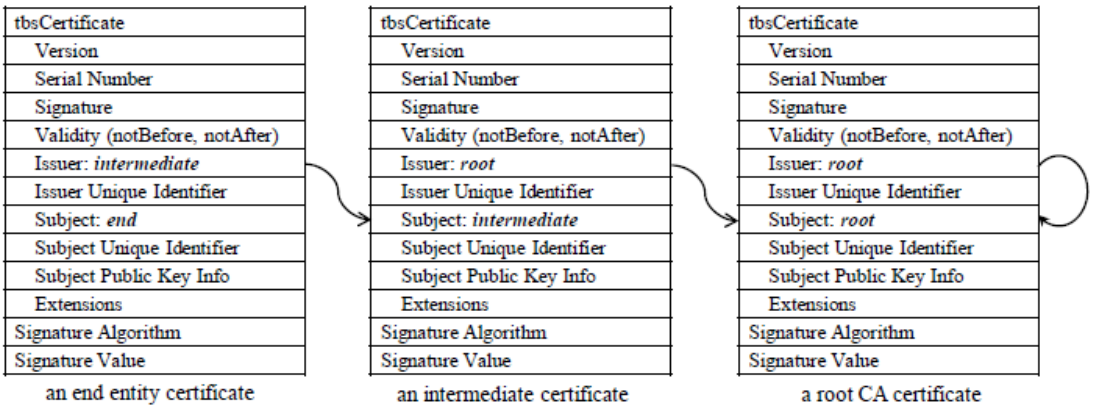
Şekil 2.2'deki resimde gösterildiği gibi, istemci ve sunucu arasında sertifika ile anahtar paylaşımı gösterilmektedir.



Şekil 2.2. Sertifika ile anahtar paylaşımı.

2.5.1. SSL Sertifika

Sertifikalar, internetin güvensiz ortamında veri iletişimi yapılırken sunucunun kimlik doğrulamasının ve ilgili verinin bütünlüğünü sağlayan veri bloğudur. Sunucu istemciye kendisinin doğru sunucu olduğunu ispatlamak için sertifikaları kullanır. İstemci bir sunucu ilk isteği atıp, sunucun üzerinde ilgili alan adı için saklı olan sertifikayı okur. Okunan bu sertifika ile sunucu ve bağlantı hakkında temel bilgiler edinilmiş olur. Bu adım sunucunun güvenirliği hakkında ilk izlenimleri oluşturur. Sertifikalar sunucuların doğrulanması ve güvenli bağlantının oluşturulması için yeterlidir ancak sertifikalarında kendini ispatlaması gerekmektedir. SSL/TLS sertifikaları RFC 5280'da tanımlandığı üzere 3 temel bölümden oluşur; Bunlar sertifika bilgisi, sertifikada açık anahtarın oluşturulduğu algoritma ve açık anahtarın kendisidir [15]. Sertifika bilgisinin içerisinde de Sertifika versiyonu, seri numarası, sertifika sahibi adı, uzantısı ve üst sertifika doğrula bilgisi bulunmaktadır. Sertifikanın kendisini doğrulamada kullanılacak en önemli bilgi sertifikayı imzalayan kuruluş imzası bilgisidir. Sertifikanın kendisini doğrulamak için bir üst sertifika tarafından imzalanır, bu bilgi alt kendisinde bulunur. Kendisini doğrulamayı bir üst sertifikaya dayandırır. Üst sertifika da kendisini başka bir üst sertifikaya dayandırır. En üst seviyedeki sertifikaya kök sertifika denir. Kök sertifikalar köklü kuruluşlar tarafından imzalanıp tarayıcılara ön yüklü eklenmiştir. Doğrulanma türüne göre sertifika tipleri Şekil 2.3'deki gibidir [15].



Şekil 2.3. Sertifika doğrulama zinciri.

2.5.2. Doğrulama Türüne Göre SSL Sertifika Çeşitleri

2.5.2.1. *DV (Domain Validation) SSL*

Doğrulama düzeyi en düşük sertifikadır. Sertifika ile sadece sunucu alan adı doğrulanır. Çok kısa sürede oluşturulabilir.

2.5.2.2. *OV (Organizational Validation) SSL*

Sunucu alan adının yanı sıra alan adına bağlı olarak sertifika sahibi kuruluş hakkında bilgiler içerir. Bu bilgiler sertifikayı imzalayan kuruluş tarafından incelenir ve onaylanır. DV sertifikalara göre çok daha güvenlidir. Sertifikanın oluşturulması imzalayan kuruluşa göre değişkenlik gösterir.

2.5.2.3. *EV (Extended Validation) SSL*

DV ve OV sertifikalara göre çok daha güvenilir olan bu sertifika üretim süreci diğer iki sertifikaya göre daha uzun sürmektedir. Sertifika doğrulayıcı kuruluşlar EV sertifikası talebinde bulunan kuruluşu fiziksel ve hukuki varlığıyla beraber çok kapsamlı kurumsal doğrulamaya tabi tuttuğundan maliyeti yüksek bir sertifikadır [16].

2.6. ŞİFRELEME YÖNTEMLERİ

2.6.1. Simetrik Şifreleme

Verinin güvensiz ortamda güvenli bir şeklide iletilmesinin birçok yöntemi öngörülmüştür. Dijital ortamda verinin güvenli iletilmesi için en fazla uygulanan yöntem verinin şifrelenerek karşıya iletilmesidir. Verinin şifrelenmesi için belirli algoritmalar

kullanılmaktadır. Teknolojinin evirilmesi ile şifreleme yöntemlerinin gelişiminde zamanla değişiklik göstermiştir. Ancak bu gelişim şifreleme yöntemlerinin zamana göre gelişiminin yanı sıra iletim ortamının şekli, alıcı ile anlaşılan protokole ve daha birçok etmene göre değişmektedir. Temelde şifreleme algoritmaları 3 grupta toplanır bunlar; simetrik (gizli anahtarlı), asimetrik (açık anahtarlı) ve data özettir. Verinin özetlenmesi (hash) tam olarak iletim ortamlarında karşılıklı iletişim içinde kullanılmasa da dijital ortamda verinin şifrelenmesi için kullanılan önemli yöntemlerden biridir. Bu tezin kapsamı daha çok iletişimde kullanılan şifreleme ile ilgili olduğundan Hash şifreleme yöntemi üzerinde durulmamıştır.

2.6.2. Simetrik Şifreleme Algoritmaları

Simetrik şifreleme bilinen eski bir şifreleme yöntemidir günümüzde bu yöntemin modern algoritmaları karşımıza çıksa dahi temelde dayandırdığı mantık aynıdır. Simetrik şifreleme, şifreleyen anahtar ile şifrelenmiş metni çözen anahtarın aynı olduğu bir yöntemdir. Simetrik şifreleme yöntemini basit bir örnek ile açıklamak için **Substitution** algoritması irdelenebilir. Bu algoritmadaki temel değer şifrelenecek karakterlerin karşısında başka bir karakterin tutularak şifrelendiği algoritmadır.

Düz metin	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Şifreli metin	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c

Şekil 2.4. Substution algoritması anahtarı.

Bu algoritmadaki anahtar Şekil 2.4'de belirtilmiş tablodur. Ve yine şifreyi çözecek anahtar yine aynıdır. "Textmessage" şifrelenmek istenirse bunun karşılığında şifrelenmiş metin "whawphvvdjh" olacaktır. Simetrik algoritmaların avantajlarının yanında dezavantajları da vardır. Simetrik şifreleme teknikleri birçok modern kriptografi sistemin temeli olmuştur [17].

Kuvvetli Yönleri;

- Algoritmalar olabildiğince hızlıdır.
- Donanımla birlikte kullanılabilir.

Zayıf Yönleri;

- Güvenli anahtar dağıtımı zordur.
- Kapasite sorunu vardır.

- Kimlik doğrulama ve bütünlük ilkeleri hizmetlerini güvenli bir şekilde gerçekleştirmek zordur.

Simetrik algoritmaya bu tez de yer verilmesinin nedeni temel motivasyon olan SSL şifrelemesinde yer almasıdır. HTTP protokolü üzerinden simetrik şifreleme algoritmalarının anahtar dağıtımı sorunu aşılarak avantajları değerlendirilmiştir.

2.6.2.1. *Doğrusal Şifreleme (Affine Cipher)*

Bu şifreleme yöntemindeki hedef doğrunun denklemi olan y=ax+b doğrusal fonksiyonunu kullanarak şifreleme işlemini gerçeklemektir. Buna göre x, şifrelenecek metni (plain text), y şifrelenmiş metni (cipher text) temsil ederek a ve b ikili anahtarını oluşturmaktadır.

Şifresiz Metin: “baba dede”

Güvensiz ortamda iletilecek şifreli metni oluşturmak için; a=3, b=2 anahtarı doğrusal fonksiyonunda şifrelenmiş metin 3x+2 eşitini alındığında ortaya çıkacaktır. B harfi alfabede 2. A harfi 1. D harfi 4 ve E harfi 5. Sırada olduğundan denklem aşağıdaki gibi çözülebilir.

3 x 2 + 2 = 8 Alfabede 8. Harf H

3 x 1 + 2 = 5 Alfabede 5. Harf E

3 x 4 + 2 = 14 Alfabede 14. Harf N

3 x 5 + 2 = 17 Alfabede 17. Harf N

Şifreli Metin: “hehe nqnq”

Şifreli metnin tekrar çözülmesi için ax+b fonksiyonun tersi alınması gerekmektedir.

$R \rightarrow R$, $a, b \in R$ ve $a \neq 0$ için $f(x) = ax + b$ ise $\leftrightarrow f^{-1}(x) = x - b / a$ dır. Burda seçilecek anahtarların tersinin alındığında şifreli metinden şifresiz metin elde edilebilmektedir. Şifrelemede kullanılacak fonksiyon özelleştirilebilir ancak fonksiyonun tersinin alınabilmesi gerekmektedir. Affine Şifreleme Yöntemi, substitution şifreleme yöntemine ile aynıdır sadece harfin karşılığında hangi harfin olacağı bir fonksiyona bağlanmıştır.

2.6.2.2. *Hill Şifreleme Yöntemi*

Hill şifreleme yöntemi blok şifreleme örneğidir. Blok şifrelemenin çalışma şekli ise şu şekildedir; Metni belirli her blokta eşit karakter olacak şekilde gruplamak, her bloğu şifreleyerek şifrelenmiş bloklar oluşturmak ve bu şifreli blokları şifreli metin çıktısı olarak metin akışı haline getirmek. Hill şifreleme yöntemi Lester Hill tarafından 1929

bulunmuş ve yayınlanmıştır. Her blok için Affine Cipher benzeri bir yöntem izlenerek verilen anahtar ile metindeki karakter değeri çarpılır ve elde edilen sonuçlar toplanarak yeni karakter elde edilir.

2.6.2.3. *Sezar Şifreleme*

Sezar şifrelemesi ismini Romalı lider Jül Sezar'dan almış ve ilk defa kendisi tarafından kullanılmış şifreleme tekniğidir. Tarihin yer değiştirme ve harf değiştirme yöntemi kullanılarak oluşturulmuş en ünlü şifreleme tekniği Sezar Şifresi'dir: Bu şifrede, her harf o harften verilen anahtar sayısı kadar sonraki harf kullanılarak yazılır. Örneğin, 3 sayısını anahtar olarak seçildiğinde 3 atlamalı bir yöntem tercih etmiş olunur, Sezar Şifresi'nde "deneme" yerine "ghqhph" yazılır. Şifrelenmiş metinden şifresiz metni elde etmek içinde şifreli karakterler üzerinden anahtar sayısı kadar geri atlama ile mesaj elde edilebilir. Sezar şifreleme ile şifrelenmiş metinlerin çözülmesi oldukça kolaydır. Frekans analizi yöntemi ya da 0 ile 25 (metin uzayı) arası anahtar denemesi ile anahtar kolayca elde edilebilir.

2.6.2.4. *Permutasyon Şifreleme*

Temel olarak Metnin içindeki karakterlerin kendi aralarında yer değiştirmesi ile geliştirilen bir şifreleme yöntemdir. Şifreleme metin bloklara ayrılarak yapılır. Her blokta bulunan karakterler kendi aralarında yer değiştirir. Şifreleme için kullanılacak anahtar seçimi bloklar ile doğrudan ilişkili olmalıdır. Metni her blokta 4 karakter olacak şekilde bloklara ayrılacak ise anahtarda 4 karakter uzunluğunda olmalıdır.

Anahtarımız: 2413

Şifrelenecek metin: "HelloSSL"

Metni bloklara ayrıldığında "HELL", "OSLL" olur burada verilen anahtar 2413 olduğundan bloktaki karakterlerin sırası anahtara göre düzenlenir. Şifrelenmiş metin blokları "ELHL", "SLOL" olacaktır. Bu yöntem ile şifrelenmiş metinden anahtarı elde etmekte kaba kuvvet algoritmaları ile günümüz bilgisayarlarında kolaylıkta elde edilebilir. Anahtara ulaşmak isteyen kişi blok sayısı ikiden başlayarak metinleri kendi arasında yer değiştirerek anahtarı elde edebilir.

2.6.3. Asimetrik Şifreleme

Simetrik şifreleme algoritmalarının zaaflarından biride şifrelemede kullanılan anahtarın mesajı açacak kişiye ulaştırılması için güvenli bir kanala ihtiyacın olmasıdır. Güvensiz ortamda güvenli iletişim kurmak için şifrelemede kullanılan anahtarın güvenli bir kanal aracılığı ile karşıya ulaştırılması gerekmektedir. Anahtarın karşıya ulaşmasında çeşitli problemler ortaya çıkmaktadır. Simetrik şifreleme tekniğinde bulunan anahtar dağıtım problemini çözmek için şifreleme ve çözme işlemlerinin her birisi için ayrı ayrı anahtar kullanma prensibine dayanan bir şifreleme sistemi geliştirilmiştir. Şifreli mesajın çözülmesi için kullanılacak anahtarın karşıya gönderilmesi yerine alıcının mesajı açmak için kendi anahtarını kullanması yöntemine asimetrik şifreleme yöntemi denir. Algoritmanın asimetrik olarak adlandırılmasının sebebi mesajı şifreleyen anahtar ile mesajı çözen anahtarın çapraz olarak farklı olmasıdır [18]. Bu şifreleme yöntemi ilk olarak Diffie ve Hellman'ın açık anahtarlı kriptografi olarak tanımladıkları 1976 yılında yayımlanmış "New Directions in Cryptography" isimli makalesine açık anahtarlı şifreleme olarak yayınlandı. Diffie-Hellman ortak gizli anahtar oluşturma sistemi ayrık logaritma problemini üzerine kurulmuş ve güvenirliği çok büyük asal sayıları seçmeye dayanmaktadır [19], [20]. Asimetrik şifreleme yöntemi ile şifrelenmiş bir metnin çözülmesi simetrik şifreleme yöntemi ile şifrelenmiş bir metnin çözülmesinden çok daha zordur ancak bununla birlikte hız problemi ortaya çıkmaktadır.

Asimetrik şifrelemede iki anahtar söz konusudur. Bunlardan biri açık anahtar diğeri de gizli anahtardır. Gizli anahtar ile açık anahtar aynı anda üretilir ve aralarına matematiksel bir ilişki bulunmaktadır. Açık anahtar güvensiz ortamda yayınlanırken gizli anahtar ise saklı tutulur. Açık anahtardan gizli anahtara ulaşması teknik olarak mümkün olmasına rağmen zorluk oranı çok fazladır.

Diffie-Hellman anahtar değişim protokolü iletişim kuracak iki kullanıcı arasında gizli anahtarların iki taraf içinde elde edilmesidir. İletişimi başlatacak kişi açık ve gizli anahtarını oluşturarak açık anahtarını güvensiz ortamda yayınlar. Alıcı da gönderici gibi göndericinin açık anahtarını kullanacak ortak anahtarı hesaplar ve kendi açık anahtarını göndericiye iletir. Gönderici gizli anahtarı ile alıcıdan gelen ve kendi açık anahtarı ile şifrelenmiş metni çözer, ortak anahtarı hesaplar. Bu iletişimin sonunda alıcıda ve göndericide ortak gizli anahtarları ile gönderilen metni çözebilecektir.

2.6.4. Rivest-Shamir-Adleman(RSA) Şifreleme Algoritması

Ron Rivest, Adi Shamir ve Leonard Adleman tarafından 1978'de bulunmuş ve bulanların soyadının baş harflerini alarak isimlendirilmiştir. Şifreleme zorluğunu ise tam sayıların çarpanlarına ayrılması güçlüğünden alır. Bunu da iki tane çok büyük asal sayının çarpımı ile elde eder bu çarpımın yanında seçtiği bir değerle birlikte yayınlar. Asal sayıları gizli, çarpımlarını ve seçtiği diğer değeri açık anahtar olarak belirler. Ortak anahtar ile şifrelenmiş metinler kendisine ulaştığından gizli anahtarı ile metne erişebilir. Özellikleri temel olarak aşağıdaki gibi listelenmiştir.

- Şifrelemede ve şifrelemiş metni çözmede iki farklı anahtar kullanır.
- Şifreleme zorluğu çarpım için seçilen asal sayıların büyüklüğü ile doğru orantılıdır.
- Dijital imza ve kimlik doğrulamasına olanak sağlar
- Seçilen sayıların büyüklüğü nedeniyle işlemlerde yavaşlık yaşanmasına neden olmaktadır.
- Şifrelenecek olan metinlerin boyutları anahtar uzunluğu ile sınırlı kalmaktadır.

2.7. PEER TO PEER (EŞE EŞ) AĞLAR

Ağ bağlantısı birden fazla bilgisayarın birbirlerine fiziksel olarak bir kablo ile ya da radyo sinyalleri fiziksel bağlantı olmadan bağlanmasıdır. Bu bağlantıya sahip herhangi iki düğüm belirli protokoller kullanarak aralarında veri ve verinin çeşitli formlarını paylaşabilirler. Peer to Peer (eşler arası) ağlar bilinen yaygın kullanılan bir ağ çeşididir. P2P, birden fazla düğümün bir araya gelerek verilerini paylaşıma açması ile oluşmuş bir ağ ekosistemidir. Bu şekilde ağda çok fazla veri oluşur bu veriler eşlere kısmı olarak kopyalandığından ağ ekosisteminin gücü artırılır. Tek bir sunucuya bağlı sistemlerde hizmetler genellikle merkezi sunucular tarafından sağlanır. Merkezi sunucunun ağında oluşabilecek herhangi bir aksaklık ya da sunucunun ağına yapılan bir saldırı bütün bir ağın hizmet akışının kesilmesine sebep olmaktadır. P2P ağında bir düğüm herhangi bir sunucuya bağlı değildir. Bunun yerine her bilgisayarda ağdaki diğer bilgisayarların adresleri bulunur. Ağda hizmet almak isteyen bir bilgisayar elindeki listedeki bilgisayarları dolaşır ve istediği hizmeti bularak işini tamamlar. Böylelikle trafik sadece

bu iki bilgisayar arasında yaşanır ve sunucu gibi tek bir noktada trafik sorunu olmaz. Bu şekilde oluşturulan bir bağlantıda kullanıcıların yaptığı işlemi izlemek oldukça zordur oluşabilecek servis aksamaların sorunu ve çözümü konusunda kullanıcılar kendi hallerine bırakılmıştır.

Napster, P2P uygulamalarının ilk örneklerindendir. Uygulama düğümlere kurulduktan sonra her düğümün diskinde kayıtlı mp3 dosyalarını ağa bağlı diğer düğümlerle paylaşımını sağlayarak zamanın ses getiren uygulamalarından biri olmuştur. Bu uygulama ile her kullanıcı kendi diskindeki mp3 dosyasını paylaşarak bir mp3 havuzu oluşturuyor. Bu havuzdan kullanıcılar istedikleri müzik dosyasını ağdan indirip kullanabiliyordu. Mp3 dosyaları sadece mantıksal bir havuzdaydı merkezi olmayan bu sistemde merkezi bir sunucu sistemi yoktu. Bu şekilde kullanıcılar P2P ağlarının gücünü tanımış oldu.

Eşler arasında oluşturulan bu ağlar günümüzde de kullanımı hızla artmaktadır. Belirli bir merkeze bağlı olmayan bu sistemlerin faydalarının yanında bazı dezavantajları da bulunmaktadır. Bu ağlarda dosyaların denetlenmesine ilişkin herhangi bir sağlayıcı bulunmadığından kullanıcılar kendi terminallerine zarar verecek herhangi bir dosyayı istemeden terminallerinde barındırmış olabilirler.

3. BLOKZİNCİR TEKNOLOJİSİ

Blokzincir; kayıtların bir zaman sırasında bloklar halinde tutulduğu, kayıtların herkes tarafından izlendiği merkezi olmayan ve her bir kaydın (bloğun) önceki kaydı doğruladığı dağıtık veri kaydı yapısıdır. Literatürdeki adı "Blockchain" olup dilimize blokzincir olarak yerleşmiştir. Ekonomide meydana gelen 2008 yılındaki finans krizi sonucu dijital para olarak bilinen Bitcoin ortaya çıkmıştır. Blokzincir ilk kimliği tam olarak belli olmayan Satoshi Nakamoto isimli biri tarafından "Bitcoin: A Peer-to-Peer Electronic Cash System" başlıklı makalede ortaya çıkmıştır [21]. Verilerin her birinin bir önceki verinin doğruluğunu sağlaması ve aralarında bir bağ olması tam olarak Blokzinciri yapısını oluşturmaktadır. Verilerin değiştirilememesinin nedeni her bir bloğun şifrelenerek özetinin sonraki bloğa aktarılmasıdır. Saldırganın araya girip bloğun yapısını değiştirmesi bloğun özetinin de değişmesine neden olacağından hem sonraki blok tarafından doğrulanmamasına hem de dağıtık yapıdaki diğer katılımcılar arasından ayrışacağından saldırganın zayıf kalmasına neden olacaktır. Blokzincirine bu haliyle kronolojik dağıtık bir veritabanıdır demek doğru olacaktır. Bitcoin, bu yapısal yönetim özelliğine sahip blokzincirinin en önemli uygulamalarından biridir. Bitcoin hiçbir merkezi otoriteye bağlı olmadan arada aracı bir banka olmadan katılan herkese açık olan ve tamamen ağ üzerinden yönetilen dağıtık bir yapıdır. Ancak yapabilecekleri bununla sınırlı olmayan, günümüz teknolojileri uygulamalarının geliştirilmesi ve daha önce hiç kullanılmayan alanlara uygulanmasını sağlamıştır. Blokzinciri tamamen güvensiz ortamda bir araya gelen kişiler arasında güveni sağlar. Blokzincir kullanımı için ortamın bu ihtiyacı hissetmesi gerekir. Blokzinciri sanal para yanında, sağlık, eğitim, oy kullanma, tedarik zinciri gibi pek çok alanda kullanılmaktadır. Kullanım alanı şeffaf ve merkezi olmayan yapısının anlaşılmasıyla farklı sektörlerde kullanılarak geniş bir yelpazeye sahip olmuştur [22]. Teknoloji, dağıtılmış bir defter yapısına ve konsensüs sürecine dayanmaktadır.

Blokzinciri temel olarak kayıtlar arasında mutlak bir bağın olduğu ve kopyalarının ağdaki tüm katılımcılara dağıtıldığı bir veri tabanıdır. Merkezi otoriteye ihtiyaç yoktur, tüm katılımcıların doğrulama yapabildiği bir dağıtık defter teknolojisi ortaya çıkar. Bir merkeze bağlı olmayan ağların, ağa saldırı olması durumunda sadece ilgili düğümü

etkisiz hale getirip sistemin çökmesi engellenir [23].

İnternette haberleşmede güvenli bağlantı oluşturmanın bilinen en yaygın yöntemi sertifika kullanarak bağlantı oluşturmaktır. Bir otoriteden sertifika almanın belirli maliyetleri vardır. Dahası sertifikalar her ne kadar güvenli olursa olsun çeşitli saldırılara maruz kalmaktadır. Bu çalışmada herhangi bir elektronik sertifika kullanmadan güvenli bir HTTP bağlantısı oluşturmak için blokzinciri gücünden faydalanılmıştır.

Literatürde blokzincir aracılığı ile güvenli bağlantıyı oluşturacak bir tarayıcı yazılımı ya da eklentisi mevcut değildir. Blokzinciri ile güvenli HTTP bağlantısını yapacak bir tarayıcının olmayışı dışarıdan harici bir uygulama ile yapılmasını zorunlu kılmıştır. Bu uygulama ileride tarayıcılar için ilham oluşturup blokzinciri için uygun tarayıcıların geliştirilmesine olanak sağlayacağı düşünülmektedir.

Blokzinciri oluşması için ortamın bu ihtiyacı hissetmesi gerekir. Düğümler arasında en önemli ihtiyaç güvenlik ihtiyacıdır. Tipik bir blokzinciri sistemi, birbirlerine tam olarak güvenmeyen birden fazla düğümden oluşur [24].

Mevcut blokzinciri sistemlerinin incelendiğinde katılımcılara ilişkin bir sınırlama söz konusu değildir. Yani kullanıcı ağa dahil olup verilerin bir kopyasını alabilir. Her ne kadar verilerin içeriklerine erişip düzenlemese de ağa dahil olmuştur. Ancak "Untangling Blockchain: A Data Processing View of Blockchain Systems" isimli makalede blockchain sistemi iki kategoride anlatmış bunlardan birisi public blockchain diğeri ise private blockchain. Public blockchain bitcoin sisteminde de olduğu gibi düğümlerin katılımına ilişkin bir sınırlamanın olmadığı bir blokzinciri sistemidir. Private blockchain ise katılımcı düğümlere sınırlamaların konulduğu bir sistemdir. "Hyperledger" en bilinen private blockchain ağı oluşturmaya yarayan bir frameworktur [24].

3.1. BLOK YAPISI

Blok zinciri yapısı içerisinde bulunan her bir blok içerisinde "Transactions" adı verilen kayıtların listesini tutar. Bloklar kaba tabir ile veri sayfaları olarak adlandırılabilir. Blokzinciri yapısı kurulduğunda ilk kaydedilen bloğa "Genesis Block" ismi verilir. Bu ilk blok protokolün bir parçası olarak tanımlanır. Zincirdeki her blok kendinden önceki bloğun kriptografik şifresini kendi içerisinde barındırır ve doğruluğunu teyit eder. Önceki blokta olacak bit bazında bir değişiklik bile bloğun hash bilgisini yani "Kriptografik Özet" bilgisini değiştireceğinden zincirin yapısına herhangi bir müdahale zorlaşır. Geçerli

bir blok, mevcut bloktaki işlemlerde şifreleme işlemleri için adreslenebilen bir eşsiz kimlik içerir. Bu işleme madencilik adı verilir [25]. Blokların içeriği iki parçadan oluşur. Bunlar işlemler ve blok başlığıdır. İşlemler bloğun içerinde kayıtlı asıl verilerdir. Blok başlığı ise 3 parçadan oluşmaktadır. Bunlar;

- Önceki bloğun kriptografik özeti
- Zaman bilgisi
- Transactions Merkle Kök değeri

3.2. BLOKCHAIN ÖZELLİKLERİ

Blockchain ile ilgili tercih sebebi olmasının nedenlerine baktığımızda karşımıza çıkacak avantajları 5 başlık altında toplayabiliriz. Bunlar; dağıtık veri tabanı mimarisi, şeffaflık, güvenlik, uçlar arası iletişimdir [26].

3.2.1. Dağıtık Veri Tabanı Mimarisi

Blokzinciri sistemi dağıtık veri tabanı mimarisinin sağlamış olduğu avantajlar ile herhangi bir üçüncü taraf ihtiyacı olmaksızın; dağıtık veri yapısında, ağdaki katılımcıların onaylaması ile birlikte zincirler oluşturulmakta, bu şekilde sistemin merkezi bir otorite ihtiyacı olmaksızın sağlıklı bir şekilde çalışması sağlanmaktadır [27]. Sistemde tüm işlemler, işlemlerin yapıldığı ve tüm kullanıcılara açık olan dağıtık muhasebe defterinde kaydedilir. Geleneksel yaklaşımda (merkezi veri tabanı), veri tabanı bir üçüncü tarafça kontrol edilirken, blokzinciri yaklaşımında veri tabanının kopyası tüm katılımcılara açıktır. Bu sayede verinin değiştirilmesi, bozulması, manipüle edilmesi engellenir. Böylelikle güvenilir, merkezi bir otorite ihtiyacı ortadan kalkar. Herkesin doğrulama yapabildiği dağıtık veri tabanı sisteminde kimseye güvenmeye gerek kalmadan bilgi muhafaza edilir [28].

3.2.2. Şeffaflık

Kayıt defterindeki tüm hareketler kayıt altına alınıp ağdaki herkese dağıtıldığından veri şifrelenmiş olarak tüm düğümlerde mevcuttur. Farklı kullanıcılara ait farklı bloklar yine kullanıcılara ait benzersiz bir anahtar ile erişilebilir. Verileri her ne kadar ağdaki herkeste bulunsa da sadece bloğa veri kaydeden kişi tarafından okunabilir. Bloktaki veriler herkeste olduğundan şeffaflık ilkesine tam uyumludur. Sistemdeki varlığın hangi

kaynaktan çıktığı ve hangi kullanıcıların kullanımına geçtiğini ve nerede olduğunu takip etmek için ideal bir platformdur.

3.2.3. Güvenlik

Blokzincirinde güvenlikle ilişkili olan birçok özellik olsa da iki en önemli kavram mutabakat ve değişmezliktir. Mutabakat, dağıtılmış bir blockchain ağı içindeki düğümlerin ağın gerçek durumu ve işlemlerin geçerliliği üzerinde fikir birliğine varabilmesi demektir. Genellikle, mutabakata varma süreci mutabakat algoritmalarına dayanır.

Diğer yandan değişmezlik blockchainlerin, hali hazırda onaylanmış olan işlemlerin değiştirilmesini engellemedeki becerisine denir. Bu işlemler genellikle kripto para transferlerine ilişkin olsa da dijital verinin parayla ilişkin olmayan diğer türlerinin kayıtlarına yönelik de olabilir.

Mutabakat, değişmezlikle birleşerek blockchain ağlarındaki veri güvenliğinin genel çerçevesini sunar. Mutabakat algoritmaları sistem kurallarına uyulduğunu ve dahil olan tüm partilerin ağın mevcut durumu hakkında fikir birliğine vardığını garantilerken, değişmezlik de geçerli olarak kabul edilen her bir bloğun ardından verinin ve işlem kayıtlarının bütünlüğünü garantiler.

3.2.4. Uçlar Arası İletişim

Blokzincir teknolojisinde ağlar bir merkeze bağlı değildir. Her bir düğüm ağdaki diğer tüm düğümler ile doğrudan ilişkilidir. Merkeze bağımlı bir iletişimde merkez düğüme yapılan bir ağ saldırısı tüm ağı kilitler iletişimi imkânsız hale getirip yapılan işlemleri durdurabilir. Blokzincirinde her bir düğüm diğer tüm düğümler ile doğrudan iletişim halinde olduğundan ağa yapılacak bir saldırı sadece ilgili düğümü etkisiz hale getirir sistemin bütünü çalışabilir halde olacaktır.

4. BLOKZİNCİR İLE GÜVENLİ WEB SUNUCU-İSTEMCİ HABERLEŞMESİ

Blokzinciri temelde bir kayıt defteridir. Para transferinde kayıt defterine alıcı gönderici ilişkisi yazılıyor ve sadece gönderici ve alıcıdan başka kimse tarafından erişilemiyorsa veri iletişiminde de iletişim bir kayıt defterinde tutulacak ise sadece alıcı ve verici arasında okunabilir olur ve gönderici ile verici arasında güven problemi oluşur. HTTP iletişiminde gizlilik ve iletişim kurulan kişinin kendini doğrulama zorunluğu ile blokzinciri arasında ortak bir çalışma anlayışı benimsendiğinden veri iletişiminde blockchain kullanılmıştır.

4.1. BLOKZİNCİRİ KULLANARAK GÜVENLİ HTTP BAĞLANTISI

HTTP bağlantıları istemci sunucu arasındaki iletişimin güvenliğini garanti edemez güvenli bir HTTP bağlantısı için sertifikalara ihtiyaç vardır. Sertifikaların güvenilirliği konusunda internet tarayıcılarında ön yüklü sertifikalar olduğundan sunucunun cevabında gelen sertifikaya güvenilip güvenilmemesi hususunda kullanıcıya bilgi verir. Bir sunucu ile güvenli bağlantı sağlamak için iki tarafın da ortak bir şifreleme algoritmasına ve anahtara ihtiyaç vardır. Sertifika ile gelen açık anahtar tarayıcı tarafından kullanılarak gizli anahtarın sunucuya gönderilmesi ile güvenli bağlantı oluşur. Bu bağlantı güvenli bağlantıdır.

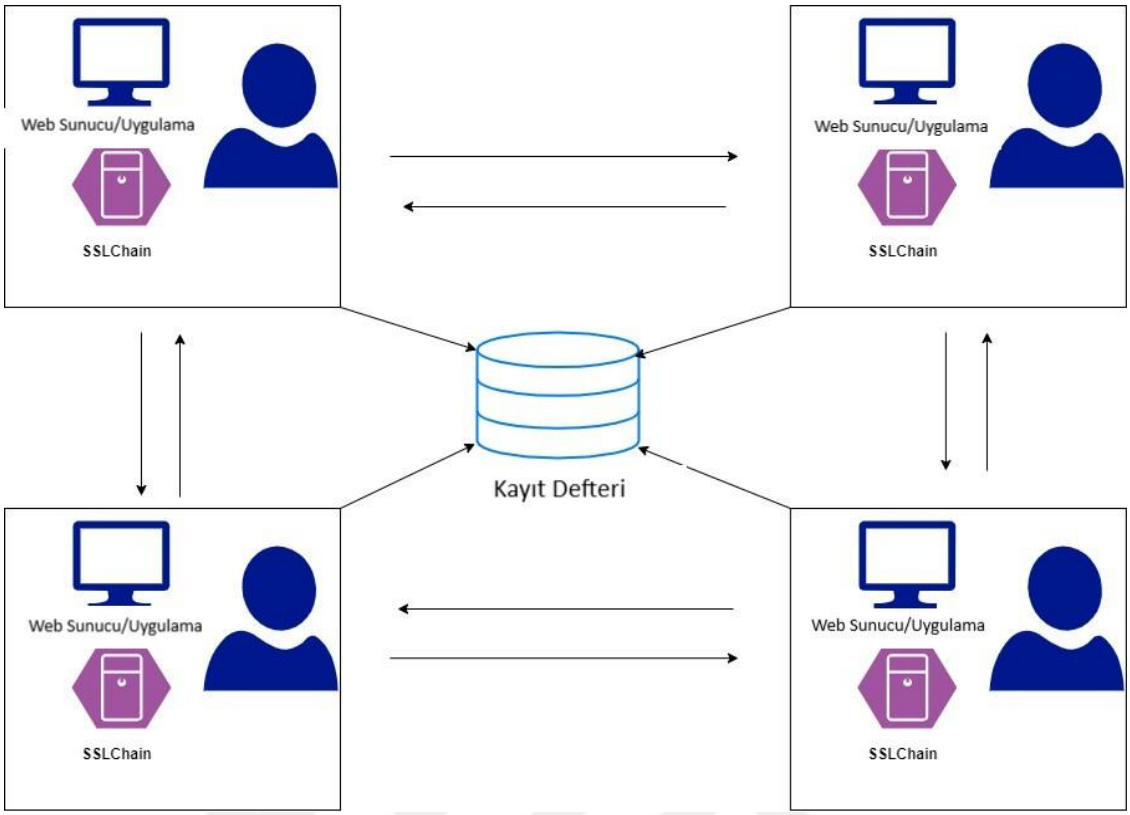
Güvenli bağlantı oluşturmanın bilinen en yaygın yöntemi sertifika kullanarak bağlantı oluşturmaktır. Bir otoriteden sertifika almanın belirli maliyetleri vardır. Dahası sertifikalar her ne kadar güvenli olursa olsun saldırılara maruz kalmaktadır. Bu çalışmada herhangi bir elektronik sertifika kullanmadan güvenli bir HTTP bağlantısı oluşturmak için blockchainin gücünden faydalanılmıştır. Çalışmanın devamında detaylı bir blok yapısı oluşturulmuştur. Zincirin oluşma kuralları ağa dahil olma, iletişimde kullanılacak geliştirme araçları, iletişim alt yapısı oluşturup, örnek bir uygulama ile verileri nasıl şifrenmiş bloklar halinde gönderebildiği incelenmiştir.

4.2. GÜVENLİ İLETİŞİM İÇİN BLOKZİNCİRİ ALT YAPISI

HTTPS bağlantısında iletişimde kullanılacak açık anahtarın sunucu tarafından ilk istekte istemcinin internet tarayıcısına gönderildiği bu sertifika tarayıcı tarafından ön yüklü üst sertifikalarla onaylanmış ise sunucudan gelen sertifika içinde bulunan açık anahtar ile oturum için oluşturduğu gizli anahtarı açık anahtar ile şifreleyip sunucuya gönderir. Bu şekilde oturum boyunca güvenli bağlantı sağlanmış olur. Günümüzde bilinen tüm tarayıcılar kendi yazılım alt yapısı ile bu bağlantıyı kurabilmektedir. Bu çalışmanın yayınladığı tarih itibari ile blockchain aracılığı ile güvenli bağlantıyı oluşturacak bir tarayıcı yazılımı ya da eklentisi mevcut değildir. Blokzinciri ile güvenli HTTP bağlantısını yapacak bir tarayıcının olmayışı dışarıdan harici bir uygulama ile yapılmasını zorunlu kılmıştır. Bu uygulama ileride tarayıcılar için ilham oluşturup blockzinciri için uygun tarayıcıların geliştirilmesine olanak sağlayacak ve eşten eşe bağlantı kurabilecek tarayıcıların gelişmesi için bir fikir sunacaktır. Uygulamanın nasıl çalışacağı, blokların yapısının detayı, doğrulama yapısının detayına ilişkin veriler aşağıda detaylandırılmıştır.

4.3. SSLCHAIN

SSLChain, blokzinciri ile kullanılarak oluşturulacak güvenli bağlantıda blokların yönetiminden sorumlu çekirdek bir uygulama olarak tasarlanmıştır. Uygulama haricen çalıştırılır. Bu uygulama güvenli bağlantıyı sağlayacak bir P2P (Peer to Peer) ağı ile merkezi olmadan çalışacak bir uygulamadır. HTTP protokolü üzerinden güvenli bir iletişim için uygulama ağa katılacak tüm düğümlerde yüklü olması gerekir. Uygulama, mevcut göz atıcıların blockchain ile iletişimi desteklemediğinden geliştirilmiştir. Ancak güvenli bir iletişim için sunucuda çalışacak uygulamanın da blockchain ile güvenli iletişime uyarlanması gerekir. Mevcut sunucu sistemlerinde geliştirilen bir uygulamada uygulamayı geliştiren kişi iletişim güvenliğinden kendini sorumlu hissetmez. Bunun yerine bu görevi sunucuda geliştirilmiş uygulamalar ve tarayıcılar üstlenir. Çalışmaya konu güvenli iletişimi gerçekleştirecek uygun tarayıcı olmadığı gibi güvenli iletişimde sunucuda bu iletişim için bir aksiyon gerçekleştirmeyecektir. Bu görev uygulamayı geliştiren kişinin bu iletişim protokolünün kurallarını bilmesi, uygulamasını bu kurallara uygun bir şekilde geliştirmesi gerekmektedir. İletişimin çekirdeğinde yer alacak SSLChain uygulaması yine HTTP üzerinde hizmet vereceğinden bir web servis olarak geliştirilmiştir.



Şekil 4.1.SSLChain çalışma yapısı.

SSLChain ağa bağlı düğümlerin HTTP üzerinden iletişim kurabilmesi için REST web servis olarak geliştirilmiştir. REST (Representational State Transfer), 2000 yılında Roy Fielding tarafından doktora tezinde tanıtılmış ve tanımlanmıştır. REST, dağıtık sistemler tasarlamak için kullanılan bir mimari tarzıdır [28].

REST, istemci-sunucu arasındaki haberleşmeyi sağlayan HTTP protokolü üzerinden çalışan bir mimaridir. REST, servis yönelimli mimari üzerine oluşturulan yazılımlarda kullanılan bir transfer yöntemidir. İstemci ve sunucu arasında XML ve JSON verilerini taşıyarak uygulamanın haberleşmesini sağlar. REST mimarisini kullanan servislere ise RESTful servis denir.

Bir REST web servis geliştirmek için günümüzde birçok yazılım geliştirme dili mevcuttur. Aynı zamanda bu yazılım dilleri ile geliştirilmiş çok fazla framework bulunmaktadır. SSLChain geliştirilirken Java ile geliştirilmiş Springboot frameworku kullanılmıştır. SSLChainin bir web servis üzerine kurulu olmasının nedeni servislerin hem kendi bilgisayarımızda hem de diğer düğümlere yine web servis üzerinden durum bilgisi gönderebilmesidir. Blokchain ile Güvenli HTTP bağlantısı mimarisinin merkezinde yer alan bu çekirdek uygulama zincirinin oluşturulması, gelen blokların

kontrol edilmesi ve blok içeriklerinin şifrelenmesi gibi önemli işlemleri gerçekleştirecektir. İstemciden sunucuya giden her istek ya da sunucudan istemciye gelen her cevap birer blok olarak gidip gelir. Blokların her biri zincire kaydedilir.

4.4. SSLCHAIN BLOK İÇERİĞİ

Bu çalışma kapsamında oluşturulan blokzinciri ile güvenli HTTP bağlantısı ekosisteminde her düğümde kullanıcının bilgisi dahilinde bilgisayarında çalışacak ve SSLChain sistemine dahil olacak blokların alınmasında ve gönderilmesinden sorumlu bir uygulamadır. Kullanıcılar her ne kadar kendi bilgisayarlarında çalışan uygulamayı bilseler dahi gelen/giden veriler hakkında bilgi sahibi olamayacaktır. Gelen/giden blokların düğümlerde sabit disklere kaydedilmesi öngörülmemiştir. Bunun yerine düğümlerin belleklerinde tutulması planlanmıştır. İletişim kuran iki düğüm oturumu bitirdiği anda bellekten silinir. Blokların sabit diske kaydedip sürekli olarak saklanması iletişim esnasında hassas verilerin olmasından dolayı daha sonra şifrelenmiş blokların içeriklerinin tespit edilmesi ihtimaline karşılık blokların silinmesi öngörülmüştür. Blokların içeriği önemli ölçüde transfer edilen verileri tutacak olsa da istemciye ve sunucuya ait veriler, oturuma ait veriler ve diğer verilere de sahip olacaktır. Çalışmada SSLChain için kullanılan blok yapısı Şekil 4.2’de yer verildiği gibidir.

Şekil 4.2. SSLChain blok yapısı.

Şekil 4.2'de yer alan blok yapısı, bağlantı sırasında istemci ile sunucu arasında transfer edilecek bloğu ifade etmektedir. Blokta verilerin yoğun kısmı Blok Verisi biriminde oluşacaktır. Şekil 4.2'de tanımlanmış alanların detayları aşağıdaki gibidir.

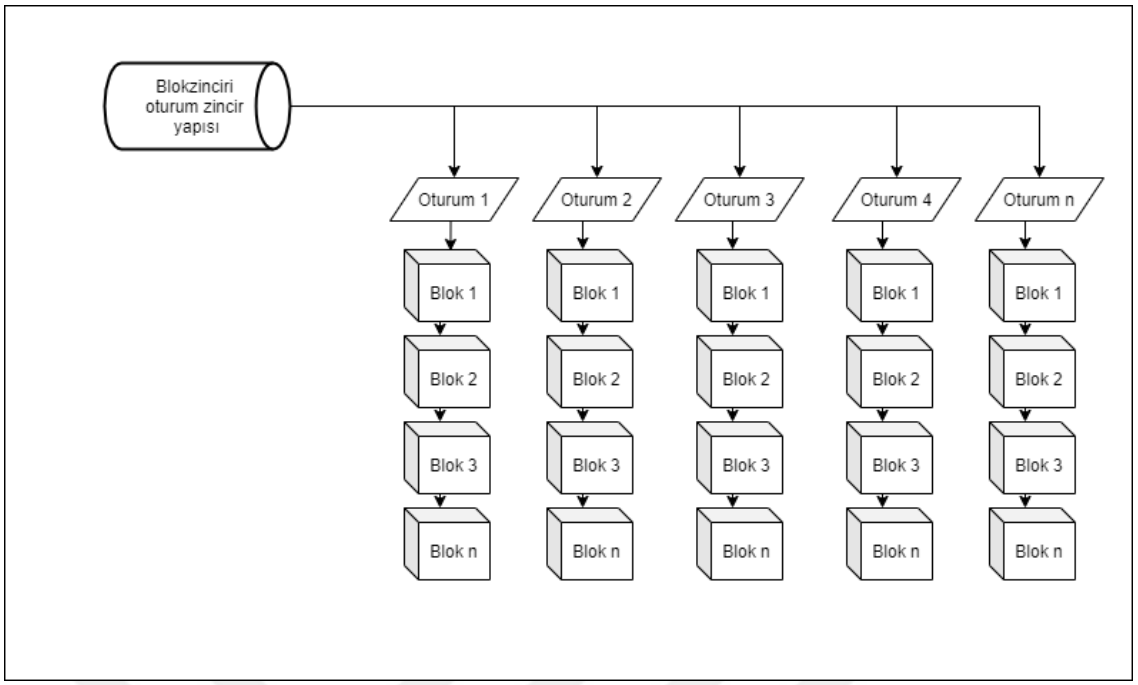
Kaynak Bilgisayar: İstemci-sunucu arasında gerçekleşecek şifreli veri alışverişinde bloğu gönderen kişinin bilgilerinin tutulduğu alandır. Veri iletişiminde bloğun başlık kısmında bulunur. Bu alan için sabit bir düğüm öngörülmemiştir. Kaynak bilgisayar sadece istemci için değildir. İstemciden sunucuya bir istek gerçekleştirmek için blok hazırlandığında "Kaynak Bilgisayar" istemci, sunucu cevap verirken blok oluşturduğunda "Kaynak Bilgisayar" sunucu olmaktadır.

Hedef Bilgisayar: İstemci-sunucu arasında şifrelenmiş blok transferinde istek yapılacak ya da cevap verilecek düğümü ifade etmektedir. Bu alan düğümlerin veri transferi sırasında isteğin ya da cevabın kendilerini muhatap alıp almadığı hususunda önemli bir alandır.

Hedef ve kaynak düğüm bilgilerinin tutulduğu bu alanlar için IP, hizmet verdiği port ve diğer HTTP başlık bilgilerine konu bilgilerin tutulması öngörülmüştür.

Bloğun Amacı (İstemci talebi/Sunucu cevabı (Request/Response)): Şifrelenmiş veri transferi sırasında isteğin ya da cevabın niteliğini belirleyen istemcinin ya da sunucunun bu alana göre cevap verme şekillerini değiştiren bir alandır. Standart bir HTTPS bağlantısında bir siteye istek gönderildiğinde sunucudan sertifikanın alınıp sertifikaya göre oturum için bir şifreleme anahtarı üretmesi ile ilgilenilmez. Bu işlemi tarayıcılar gerçekleştirir. Ancak blokzinciri ile güvenli iletişim için bu işlemin kullanıcı tarafından gerçekleştirilmesi gerekir. Yine kullanıcı bu aşamalardan haberdar olmayacak şekilde uygulamalar gerçekleştirilir ancak uygulamada ilk isteklerin diğer isteklerden ayırılması gerekir. İletişim için kullanılacak durum tipleri PublicKeyDemand, PublicKeyReply, HttpRequest, HttpResponse olarak belirlenmiştir. PublicKeyDemand tarayıcıdan sitenin kök dizinine erişim için ilk istek atıldığı zaman sunucu tarafından bu isteğin ilk istek olduğu tespit edildiği anda cevap için oluşturduğu bloğa bu sabiti belirler. İlk istek sırasında herhangi bir anahtar paylaşımı olmadığından istemci de bu duruma göre kendi durumunu belirler. PublicKeyReply; Sunucu, istemciden gelen bloğun durumu PublicKeyDemand ise istemci oturum için sunucudan gelen açık anahtarı kullanarak kendisi bir şifre belirleyip bloğa kaydeder ve bloğun durumunu PublicKeyReply yapar. PublicKeyDemand ve PublicKeyReply standart HTTPS bağlantısında kullanıcının müdahalesi dışında gerçekleşen anahtar paylaşımı istek ve cevaplarına benzemektedir. HttpRequest, HttpResponse SSLChain bağlantısında anahtar paylaşımı yapılıp oturum için ortak anahtar belirlendiği kısımdan sona standart istek cevap durumlarıdır.

Blok İndeksi: Blokzinciri teknolojilerinde tüm bloklar tek bir ana zincir üzerinde ilerlemektedir. Ancak bizim bloklarımız oturum boyunca tutulup daha sonra yok edileceğinden ağdakilerin tamamını tek bir zincire bağlanmaması gerektiği öngörülmüştür. Standart blokzinciri teknolojisinde tek bir zincir üzerinde birbirleri ile tutarlı bloklar her biri önceki ve sonraki zincirler ile kendisini doğrulayacak şeklide inşa edilir. Bizim kullanımımızda blokların uzun süre saklanması kişisel verilerin korunması hususunda oturum sonunda yok edilmesi gerektiği düşünülmüştür. Tüm düğümleri tek bir zincirde toplanmış olsaydı oturum sonrasında bu zincirlerin aradan çıkartılması zincirin doğrulanması konusunda sorunlar oluşturacaktı bunun yerine her oturum için bir zincir ve zincirin adı bloğun indeks kısmında tutulması gerektiği öngörüldü. Bu sayede sunucu birden fazla oturumu yönettiğinden her istemcisi için bir zincir oluşturacak kendisine gelen blokları indeks değerine göre doğru zincire ekleyecektir. Zincirin ve oturumun ilişkisi aşağıdaki Şekil 4.3'de örneklenmiştir.



Şekil 4.3. Blokzincirinde oturum zincir yapısı.

Blok Verisi: İstemci – sunucu arasında gerçekleşecek veri transferi sırasında asıl verinin yani ekranlarda kullanılacak verilerin bloklarda tutulduğu alandır. Buradaki tutulacak veri bir json veri nesnesi olarak transfer edilecektir. Veri json olarak tutulmasının nedeni kullanıcıların kendi objelerine gelen veriyi dönüştürebilmesidir. Burada bloğun durumunda göre de veri içeriği değişmektedir. Bu içerik normal bir HTTP bağlantısında formlarda ya da istek başlıklarında taşınmaktadır. Sunucundan istemciye gelen cevaplarda ise http body içerisinde gelmedir. Yine HTTP bağlantında bu içerik json, xml ya da form verileri olabilir.

Blok Özet Verisi: Standart blokzinciri teknolojilerinde kullanılan blokların benzersizliğini sağlayan alanlarda blokların doğruluğunu teyit etmek için kullanılır. Bu alan oluştururken yine blok içerisinde yer alan block data (blok verisi), previos hash(kendinden önceki blok özet verisi) ve data alanlarının SHA-256 özetleme algoritması ile özetlenmesi ile oluşur.

Kendinden Önceki Blok Özet Verisi: Blokların birbirini bağlanması ve doğrulama için kullanılan alandır. Her bir blok yaratılırken önceki bloğun özet değeri alınarak bloğa eklenir.

Zaman Damgası: Blok oluşturulduğu anda oluşturulan veridir. Blok özeti oluşturulurken karmaşıklığı yükseltmek ve zaman bilgisini tutmak için kullanılmıştır.

4.5. SSLCHAIN ÇALIŞMA YÖNTEMİ

Çalışmamıza konu blokzinciri ile güvenli HTTP bağlantısının çekirdeğini oluşturacak uygulamanın çalışma yöntemi 4 modüle bölünmüştür. Bunlardan ilki çalışan sistem ile ilgili bilgilendirme yapan servistir. http://~/system/info adresinde hizmet veren bu servis çalışan ekosistem ile ilgili genel bilgileri vermesi için kullanılması öngörülmüştür. Bir diğer modül ise uygulamada hem istemcinin hem de sunucunun kullanacağı ortak blok işlemleri için kullanılacak servislerdir. http://~/chain/brodcast/addBlock adresinde çalışan servis zincire blok eklemek için hem istemcinin hem de sunucunun kullanacağı bir servistir. http://~/chain/brodcast/verifyBlock adresinde çalışan servis ise düğümlerin ellerindeki blokları doğrulamak için kullanabileceklerdir. http://~/chain/brodcast/getChain/{index} adresi ise ağda bulunan düğümlerin erişmek istediği blokları almalarını sağlayacak bir servistir. Burada index parametresi bir değişkendir. Düğümler talep ettikleri blokların index bilgisine sahip olmalıdır. Oturum için önceden sahip oldukları index bilgisini kullanarak bloklara erişebilecektir. Buraya kadar olan servisler ortak kullanım için oluşturulmuş servislerdi. Sistemde sadece sunucuların kullanacağı ya da istemcilerin kullanacağı servisler olacaktır. Bunların detayı uygulama tanıtımında yer verilecektir.

SSLChain ile güvenli bir bağlantı oluşturmak için ağa katılmak isteyen tüm düğümler ağın merkezinde çalışacak ve düğümlerin birbiri ile haberdar olmasını sağlayacak uygulamayı çalıştırması gerekir. Uygulama varsayılan olarak 8080 portunda çalışması öngörülmüştür. Düğümler uygulamayı kendi terminallerinde çalıştırdığı anda merkezi kayıt sistemine düğümün IP adresini ve uygulamanın hizmet verdiği port bilgisini gönderiyor olacaktır. Bu şekilde ağdaki diğer düğümler ağa katılan yeni düğümden haberdar olacaktır. SSLChain kullanabilmek için istemci tarafında yapılması gereken sadece uygulamayı çalıştırmak olacaktır. Uygulama üzerinden güvenli bağlantı için yapılması gereken sunucu tarafında hizmet verecek sunucuda yazılması gereklidir.

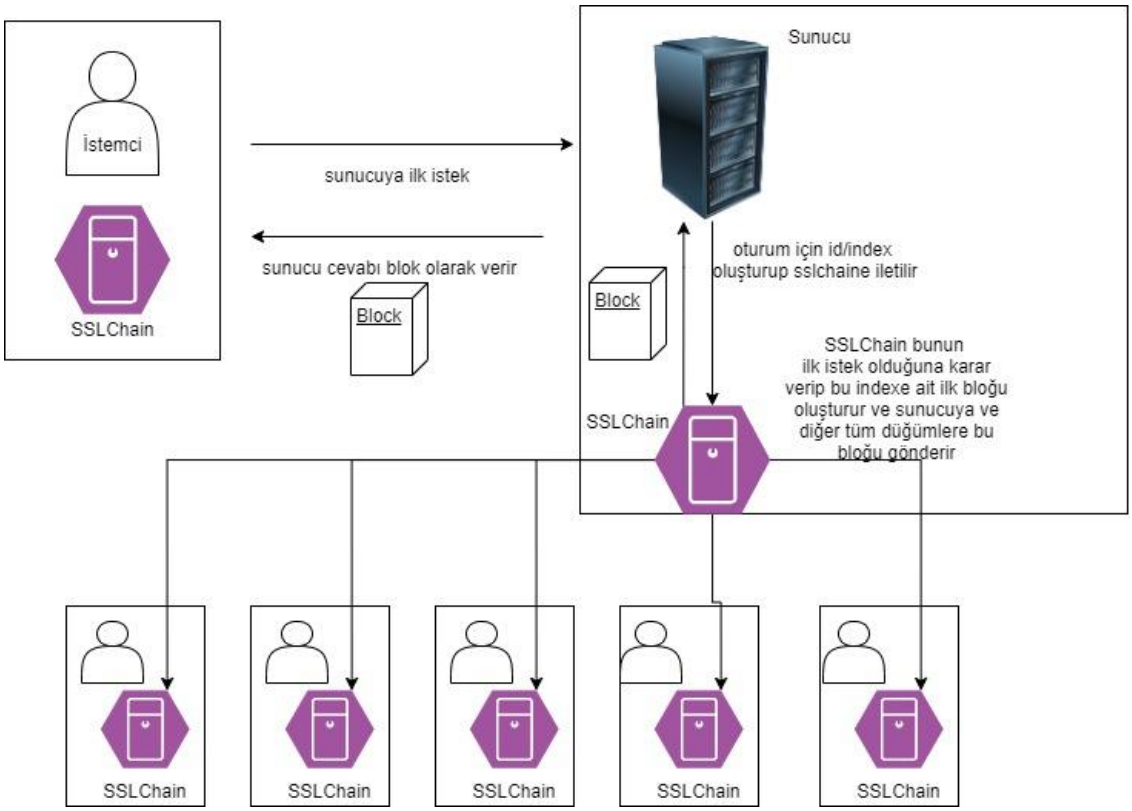
SSLChain ile iletişim için geliştirilmiş bir uygulama hizmet verirken, uygulama sunucuda kök dizin çağrıldığı anda sunucu tarafında bu istemci ile devam eden bir oturum olup olmadığı kontrol etmesi gerekir. Eğer bir oturum yok ise sunucu kendi tarafında http://localhost:8080/chain/request/blockAdd adresine index oluşturarak sadece index verisini gönderir. SSLChain uygulaması bu isteği aldığı anda gelen indexin durum verisini tespit eder. Durum değeri tanımlanmamış ya da anahtar paylaşım talebi ise bu

daha önce bu indexte bir anahtar paylaşımı olmadığı anlamına gelir. Sunucudan, sunucu tarafında çalışan SSLChain uygulamasına gönderilen index için SSLChain uygulaması bir blok oluşturur ve 2048 bit uzunluğunda RSA algoritması ile çalışacak bir anahtar yaratıp bunu bloğun veri kısmına ekler. Bloğun başlık verisini doldurup bloğun durum tipini PublicKeyDemand olarak düzenler. Oluşturduğu bu bloğu ilk olarak kendi bloklarını tuttuğu indexlenmiş zincirlere kaydeder. Eğer index yok ise oluşturur. Kendi zincir sistemine bu bloğu kaydettikten sonra merkezi kayıt sistemindeki tüm düğümlere düğümlerdeki /chain/brodcast/addBlock servislerini kullanarak gönderir. Ağdaki tüm düğümler bu bloğu alır ilgili zincire kaydeder. Daha sonra sunucudan gelen index verisine karşı bloğu sunucuya gönderir. Sunucu da aldığı bu bloğu istemciye olduğu gibi iletir. İlk isteğin sonucunda zincirde oluşan blok Şekil 4.4'deki gibi olacaktır.


```
{
    "index": 580638370,
    "timestamp": 1592754726632,
    "blockHeader": {
        "sourceHost": null,
        "targetHost": null,
        "intentStateType": "PublicKeyDemand"
    },
    "data": "{\"clientPublicKey\":\"MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA
        +h0ckArRpK5fH6nhFerWZ5kWkXwNbtHsEM17EdjmlsBhA8dWXBQJBZsKLm8gaqgfCFfjppMF
        +KaYTwGN4JFLZBsnKG5Yyk618xRYZc2NDVINihQA1q30mCNgbBul9oZTbkonj9AVInLF5Qr
        ",
    "hash": "5a24d53702b6dce59409be01d0434ea2dcc70480c5a5c6345b308679a5f3c81b",
    "previousHash": "-1"
},
```


Şekil 4.4. Oluşan ilk blok.

İlk blok olduğundan önceki bloğun özet bilgisine sahip olmayacaktır. Bloğun veri kısmında sadece public bir key bulunmaktadır. İlk isteğin temsili bir çizimi Şekil 4.5'deki gibidir.



Şekil 4.5. Sunucuya gönderilen ilk istek.

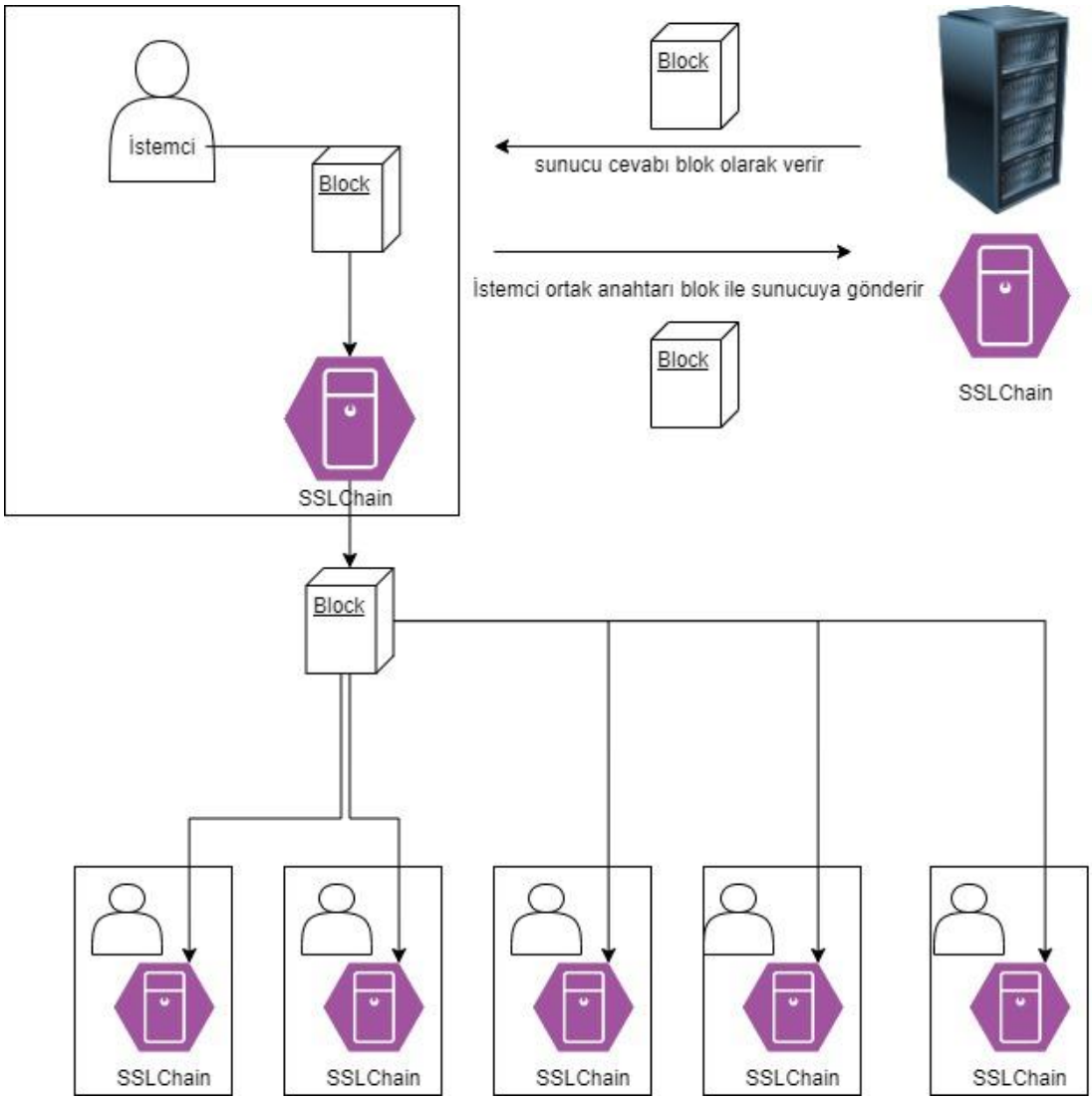
İstemcinin bu bloğu aldığı anda yine tarayıcısı tarafında çalışacak bir script ile bu bloğun ilk blok olduğunu anlayıp istemci tarafında çalışan SSLChain uygulamasında http://localhost:8080/chain/response/checkBlock servisine bu bloğu gönderecektir. Uygulama bu servisten gelen bloğu doğrulamak için merkezi kayıt sistemine bağlanıp buradaki düğümlerin bilgilerini indirecektir. Bu bilgileri aldıktan sonra her düğüme düğümlerde çalışan /chain/brodcast/verifyBlock servisinden bu bloğun doğruluğu teyit edilecektir.

Bloğun doğruluk politikası çalışmanın yapıldığı tarih itibari ile ağdaki tüm düğümlerin en az yarısının bu bloğa, zincirlerindeki bloklar da dahil olmak üzere sahip olması olarak belirlenmiştir. Diğer düğümlerin en az yarısı bu bloğu teyit ettikten sonra bloğun doğru olduğuna dair veri dönecektir. Eğer blok teyit edilmiş ise ve bloğun ilk blok olduğu teyit edilmiş ise istemci tarafında çalışacak SSLChain uygulaması AES şifreleme algoritması ile çalışacak 256 bitlik bir ortak anahtar oluşturacaktır. Bu anahtarı bloğun veri kısmına ekledikten sonra sunucuda gerçekleştiği gibi blok tüm düğümlere ve sunucuya gönderilecektir. Bu iletişimin sonucunda zincire ikinci blok eklenmiş olacaktır. İletişimde bloklar Şekil 4.6'daki gibi bağlanmıştır.

```
[
    {
        "index": 580638370,
        "timestamp": 1592754726632,
        "blockHeader": {
            "sourceHost": null,
            "targetHost": null,
            "intentStateType": "PublicKeyDemand"
        },
        "data": "{\"clientPublicKey\":\"MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAsq8kQTF17N4zcS53Tg05paSYhNtR7lr5hCTCKprjuK
            +h0ckArRpK5fH6nhFerWZ5kWkXwNbtHsEM17EdjmlsBhA8dWXBQJBZsKLm8gaqgfCFfjppMPZg0+v2NiEK5XcDtdFrBu/qtmYkgxB9C4fo4I5ToNn
            +KaYTwGN4JFLZBsnKG5Yyk618xRYZc2NDVINihQA1q30mCNgbBul9oZTbkonj9AVInLF5QrlTxXDkng56l/z6NFiZHo2KyFKq9D71b6gsWcoDX6AKQ
            \",
        "hash": "5a24d53702b6dce59409be01d0434ea2dcc70480c5a5c6345b308679a5f3c81b",
        "previousHash": "-1"
    },
    {
        "index": 580638370,
        "timestamp": 1592754749802,
        "blockHeader": {
            "sourceHost": null,
            "targetHost": null,
            "intentStateType": "PublicKeyReply"
        },
        "data":"{\"commonKey\":\"BZ0eLiOtw+Ih/1pFq5FbMKSndVcMwfjnXwPqQGqLIbDdIqOCk0Fedg0WIngKAcuajrvhorB/nTVmQrt2X7Tbz0C37qNXK
            JUfTA1DB33Ktf4NWzECykZ/2gho/Nx7ubrFL2rSTasFdHqduzEFWggnL2+FOq/iTj11krH7BVhL+UfusO0NfgWqOLndRLx+dQ9I8m/XEyDDByhtr3d
            TnZAojI28fSVqnfGppuFQMBKHAZIN039z/Hcyqh085spYpBIIRDI+1PYDmJg\=\=\"}",
        "hash": "420c4c918a3d75f9e7185c1dd6a42999de3461caf9a71cd9e3ed96d9b3074fe8",
        "previousHash": "5a24d53702b6dce59409be01d0434ea2dcc70480c5a5c6345b308679a5f3c81b"
    }
]
```


Şekil 4.6. Key paylaşımı.

İkinci adım sonrasında her iki tarafta SSLChain ve Sunucu dahil olmak üzere oturum için birer index kaydetmiş ve oturum boyunca bu indexi kullanacaktır.



Şekil 4.7. İstemcinin key blok paylaşımı.

Artık istemci sunucu arasında yapılacak istek cevap işlemi standart HTTP üzerinden gerçekleşecek ancak istek ve cevapta bloklar kullanılacaktır. Yani ağ trafiğini izleyen biri sadece şifrelenmiş blokları görüntüleyebilecektir. Bu sayede iletişim şifrelenmiş olacaktır. İletişimin devamında bloklar Şekil 4.8'deki gibi listelenecektir.


```
[
    {
        "index": 943462082,
        "timestamp": 1592757156630,
        "blockHeader": {
            "sourceHost": null,
            "targetHost": null,
            "intentStateType": "PublicKeyDemand"
        },
        "data": "{\"clientPublicKey\":\"MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAu
iIf8ounplAU03OCdSkpo2KScPCCcAawLTnXWcJUJcvEYZBS3U/Yb5tICah7HZJKJytz+FR1fNn9bW8dqiaONO
cfEI80W6MiY+uIGNoJfbQ/YaV1kBTZWkwfSPCIVWPh/5aRpv0fJZ8lwJMyvGI16pflSmTkn5mzGPNUqSLIsom
```

```
CrdV14AA9ECLutpOVgEfiNRJgxgOPFAJ5yRf9pbmGTL7EyOUBHUuaaGQdlqT5hyypADePjl565QAg3Pxcscvy
kpnaeSVTGA6N3R74k19wFx1e5wMEp7F9nHcQF4cDjVShEcC44UxCpy/kGkkiJ/ZQWCwOTH+JQwJiIFHLYRenA
wIDAQAB\"}",
        "hash": "da2653ec164a3faa76230535df3b7d851c4011bed1df6164176750417950c0ff",
        "previousHash": "-1"
    },
    {
        "index": 943462082,
        "timestamp": 1592757157420,
        "blockHeader": {
            "sourceHost": null,
            "targetHost": null,
            "intentStateType": "PublicKeyReply"
        },
        "data":"{\"commonKey\":\"tgnLsQDkcm+DPigpH3A/8h1YEXF3e/RA1I/L67eKfabktmrsssz7
WIw0x/TC+2/a0veLY7PUUQb5csS9Qx26A1oXa8U5ngu1qKnD1n3lVdEEJbuzyPXpD/EsFUcRgnaFy5FMfEmRI
oIJfnmPJjsp097CEI0gQBFkAcTpeiLrMU5D5yN7J6i7cXU2zvT+CCCZ4EcBNimnZofyevHrece4b/88YMq7YP
Jn4BqXPyudj0is8c6DFb8/pRarLGw381zgJDt8jwBj991iDtmOr4ZaL3BymKqsiP9AKR9Xv5GzNt+6P3rdxOw
q89FdvHaWJbLMmf9Xx81TMj9KAPYZLFlkOg\=\=\"}",
        "hash": "caacaabea581ea8a675a9b7d9c48857c9d30f6c201dd602558fee2448eb12c34",
        "previousHash": "da2653ec164a3faa76230535df3b7d851c4011bed1df6164176750417950
c0ff"
    },
    {
        "index": 943462082,
        "timestamp": 1592757172501,
        "blockHeader": {
            "sourceHost": null,
            "targetHost": null,
            "intentStateType": "HttpResponse"
        },
        "data": "wqJIMabLNkj+UnuP4ZdSUSZlmfwxX8oGp0lhy6C4hzw1V3N18dNlhzbzcQtWiXDZ",
        "hash": "995d7c01baf330e1448c5ca93d23628b207e1f56da08ed1042cbcae10761efc8",
        "previousHash": "caacaabea581ea8a675a9b7d9c48857c9d30f6c201dd602558fee2448eb1
2c34"
    },
    {
        "index": 943462082,
        "timestamp": 1592757231071,
        "blockHeader": {
            "sourceHost": null,
            "targetHost": null,
            "intentStateType": "HttpResponse"
        },
        "data": "wqJIMabLNkj+UnuP4ZdSUZbaOrFwenYY9qXTfiibArBDQdjS68AKT1DqXhBcTlfS",
        "hash": "10452aed45b31c1509d14d9c22fdc9e03045f83d498153901e34cccd056b3021",
        "previousHash": "995d7c01baf330e1448c5ca93d23628b207e1f56da08ed1042cbcae10761
efc8"
    }
]
```

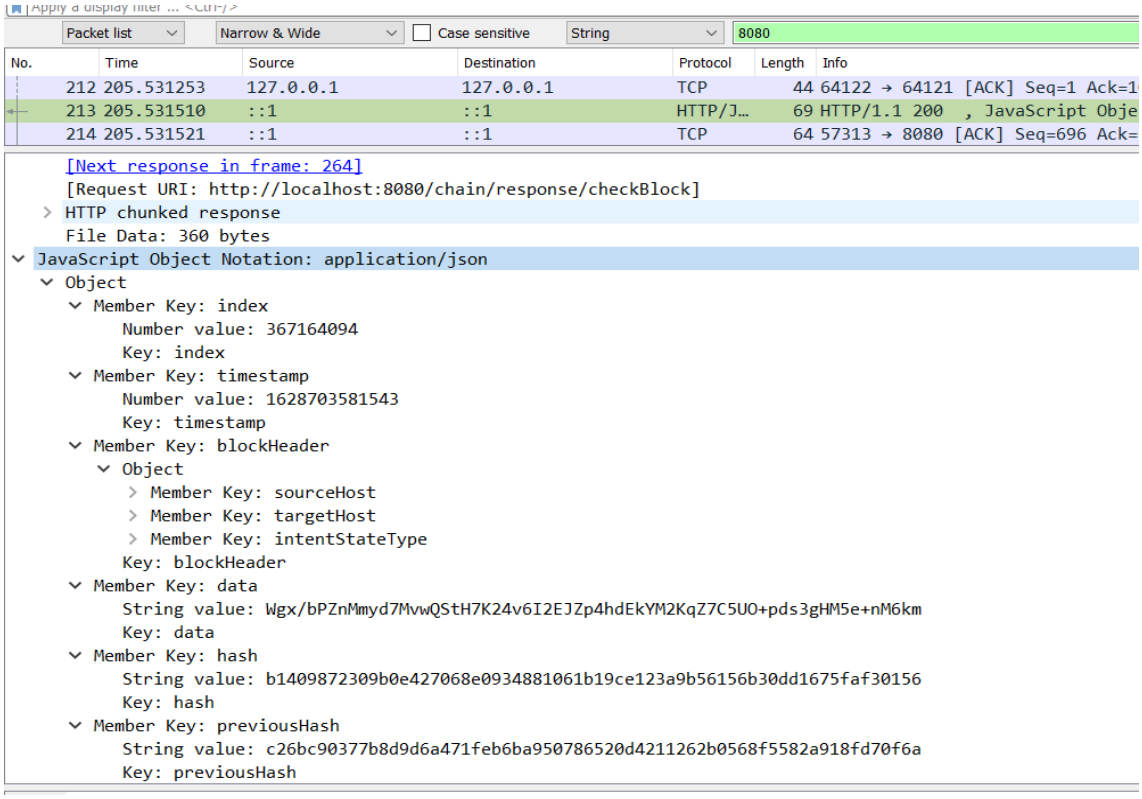

Şekil 4.8. SSLChain blokzinciri.

5. BULGULAR

Önerilen yöntemin örneklenmesi için bir web sunucusu ve 4 istemciden oluşan bir ağ kurulmuştur. Önerilen yöntemdeki güvenli iletişimin sağlanması için sunucuda çalışan uygulama SSLChaine uygun geliştirilmiştir. Uygulama istemciden aldığı iki tane sayıyı güvenli bir şekilde sunucuya gönderilmesi ve cevabında yeniden istemciye güvenli şekilde ulaşmasını hedeflemiştir.

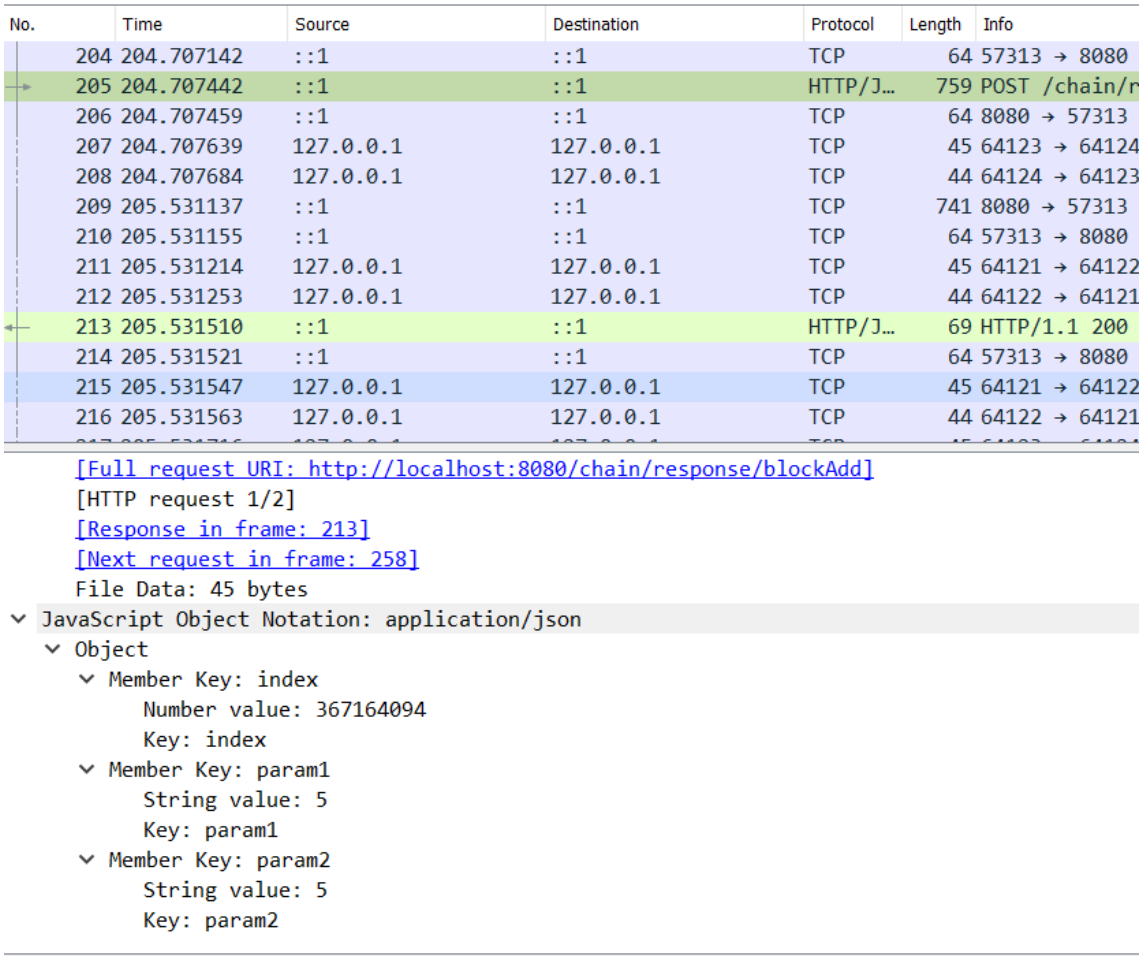
Anahtar paylaşımı sırasında asimetrik şifreleme için AES algoritması 2048 bit anahtar uzunluğuyla kullanılmıştır. Anahtar istemci tarafından ortak anahtarı şifreleyip tekrar sunucuya göndermek için kullanılmıştır. Ortak anahtar için RSA algoritması 256 bit anahtar uzunluğu ile kullanılmıştır.

Oluşturulan bu ağda trafik oluştuğu anda ağın içerisinden izlenmeye başlanmıştır. İstemcilerden sunucuya gönderilen isteklerin içerisindeki form verilerinin ve içeriklerinin şifrelenerek blok şeklinde gönderildiği Wireshark programından görülmüştür. Trafiğin durumu ve şifrelenmiş içerikler Şekil 5.1'deki gibidir.



Şekil 5.1. Şifreli HTTP response içeriği.

Wireshark uygulamasından görüldüğü üzere sunucudan gelen veriler tek bir blok ile data alanının içerisinde şifrelenmiş olarak gelmiştir. Bunun yanı sıra aynı uygulamadan HTTP bağlantısının güvensizliğini göstermek için istemciden sunucuya giden veriler şifrelenmemiştir. Bunun örneği de Şekil 5.2'deki gibidir.



Şekil 5.2. Şifresiz request içeriği.

Şekil 5.2'de görüldüğü üzere güvensiz olan HTTP metodu ile sunucuya giden param1: 5 ve param2: 5 değerleri ağ üzerinden açıkça izlenebilmektedir. Ancak sunucudan gelen verileri ise şifrelenmiş olarak blok içerisinde gelmektedir.

Saldırgan istemcinin haberleşme trafiğini elde edebilmesine rağmen verileri elde edemediği sadece şifrelenmiş blokların yapılarını görebildiği tespit edilmiştir.

Oluşturulan ağda SSLChain ile sertifika ücreti ödemeksizin güvenli bağlantı kurulabilmiştir. Bağlantı sırasında uygulamanın hem avantajı hem de dezavantajı olan husus düğümlerin sayısının artmasıdır. Düğümlerin sayısının artması hem daha fazla düğümden doğrulama yapılması hem de her düğüm ile bilgi alışverişine olanak

sağladığından çok fazla bekleme süresine sebep olmuştur.

Sistemin performans testi için sunucu rolünde bir bilgisayar, istemci olarak çalışacak dört terminalin olduğu bir ağ oluşturulmuştur. Aynı ağ bağlantısında gerçekleştirilen testte ağlar arası gecikme göz ardı edilmiştir. Çalışan uygulama sadece anahtar paylaşımı ve basit bir istek ve cevaptan oluşturulmuştur. Uygulama içinde veri tabanı sorgu gecikmesi olmamıştır. Tomcat uygulama sunucusunun ve kullanılan SpringBoot frameworkunun ortalama gecikme süreleri Çizelge 5.3, Çizelge 5.4 ve Çizelge 5.5'te verilmiştir.

Çizelge 5.3. 1 sunucu ve 2 istemciden oluşan ağ gecikmeleri.

1 sunucu ve 2 istemciden oluşan ağ			
	Açık anahtarın üretilmesi ve düğümlere dağıtılması	Gizli anahtarın sunucuya iletilmesi	İsteklerin cevaplanması
Sunucu	12ms	-	8ms
İstemci-1	-	5ms	9ms
İstemci-2	-	5ms	9ms

Çizelge5.4. 1 sunucu ve 3 istemciden oluşan ağ gecikmeleri.

1 sunucu ve 3 istemciden oluşan ağ			
	Açık anahtarın üretilmesi ve düğümlere dağıtılması	Gizli anahtarın sunucuya iletilmesi	İsteklerin cevaplanması
Sunucu	14ms	-	11ms
İstemci-1	-	5ms	12ms
İstemci-2	-	5ms	12ms
İstemci-3	-	5ms	12ms

Çizelge 5.5. 1 sunucu ve 4 istemciden oluşan ağ gecikmeleri.

1 sunucu ve 4 istemciden oluşan ağ			
	Açık anahtarın üretilmesi ve düğümlere dağıtılması	Gizli anahtarın sunucuya iletilmesi	İsteklerin cevaplanması
Sunucu	16ms	-	12ms
İstemci-1	-	5ms	15ms
İstemci-2	-	5ms	15ms
İstemci-3	-	5ms	15ms
İstemci-4	-	5ms	15ms

Yeni düğümlerin eklenmesi ile gecikmelerin doğrusal artış göstermediği sadece düğüm başına sunucuda her istemci için oluşturulan oturum maliyetinin eklenmiş olduğu görülmüştür. Geniş ölçekli ağlarda her istemciye erişmek, açık anahtarı ve blokları dağıtmak maliyetli olacağından istemci grupları oluşturularak blokları dağıtmak daha sürdürülebilir olacaktır.

6. SONUÇLAR VE ÖNERİLER

Bu çalışmada blokzinciri yönteminin literatürde yer alan uygulama alanlarına yeni bir alan eklenerek web haberleşmesinde sunucu-istemci arasındaki güven ilişkisinin oluşturulması sağlanmıştır. Güvenli web istemci-sunucu haberleşmesi için önerilen blokzinciri teknolojisine dayalı olan ve adına SSLChain denilen yöntem, HTTP bağlantılarının SSL/TLS sertifikaları kullanmadan verilerin güvenli bir şekilde gönderilip alınmasını sağladığı görülmüştür. Bu amacı gerçekleştirmek için sunucu ve istemcilerin kullanması gereken bir uygulama geliştirilerek etkinliği ağ paket yakalama yazılımları ile gösterilmiştir. Önerilen yöntemin kullanımı ile uygulama geliştiriciler ve uygulama sunucular için alternatif bir güvenli el sıkışma yöntemi elde edilmiştir.

Önerilen yöntemi kullanan ağdaki istemci sayısının artışının isteklerin cevaplanma zamanına olan etkisi oluşturulan deneysel ortam kullanılarak incelenmiştir. İstemci sayısının artmasının ağda gecikmelere yol açtığı belirlenmiştir.

Önerilen yöntemin uygulanmasında ağdaki her istemciye erişim zamanı gecikmesine yol açacağından, yüksek düğüm yoğunluğuna sahip ağlarda bu yöntemin kullanımı için istemci ve sunucuyu içerisine alacak alt gruplar oluşturulmasına dayalı gruplama algoritmaları ile birlikte kullanımına yönelik yeni çalışmalar yapılabilir.

7. KAYNAKLAR

[1] S. G. Dedeoğlu, "Özgürlük, mahremiyet, demokrasinin değeri ve bilişim toplumunda maruz kaldığı tehditler," *Journal of Yasar University*, c. 9, sayı 34, ss. 5889–5891, 2014.

[2] M. Işık, *İletişim Bilimine Giriş*, *4*. baskı, Konya, Türkiye: Eğitim Yayınevi, 2018, ss. 150-220.

[3] B. Segal "A short history of Internet protocols at CERN," *CERN Computer Newsletter No.2001-001 in Section*, ss. 1-3, 1995.

[4] A. Shiranzaei ve R. Z. Khan, "Internet protocol versions," *2nd International Conference on Computing for Sustainable Global Development*, Yeni Delhi, Hindistan, 2015, ss. 397-400.

[5] A. Goldberg, R. Buff ve A. Schmitt, "A comparison of HTTP And HTTPS performance," *Computer Measurement Group International Conference*, New York, ABD, 1998.

[6] D. S. V. Madala, M. P. Jhanwar ve A. Chattopadhyay, "Certificate transparency using blockchain," *IEEE International Conference on Data Mining Workshops*, Singapur, ss. 71-80, 2018.

[7] S. J. Lukasik, "Why the Arpanet Was Built," *IEEE Annals of the History of Computing*, c. 33, sayı 3, ss. 4-21, 2011.

[8] J. Postel, "Internet Protocol," *Darpa Internet Program Protocol Specification*, 1981, ss. 1-39.

[9] B. Segal. (2021, 1 Şubat) *A short history of internet protocols at CERN*. [Online]. Erişim: http://ben.home.cern.ch/ben/TCPHIST.html.

[10] M. Baykara, R. Daş ve İ. Karadoğan, "Bilgi güvenliği sistemlerinde kullanılan araçların incelenmesi," *1st International Symposium on Digital Forensics and Security*, Elazığ, Türkiye, 2013, ss.231-239.

[11] I. Stanivuk, V. Belic, T. Samardzic ve D. Simic, "Expanding lua interface to support HTTP/HTTPS protocol," *13th International Conference on Advanced Technologies, Systems and Services in Telecommunications*, 2017, ss. 407-409.

[12] A. Çakmak, "Web güvenliğinde SSL/TLS kriptografik protokolü: açıklıklar, saldırılar ve güvenlik önlemleri," Yüksek lisans tezi, Bilgi Güvenliği Mühendisliği ve Kriptografi, Fen Bilimleri Enstitüsü, İstanbul Şehir Üniversitesi, İstanbul, Türkiye, 2018.

[13] C. Brubaker, S. Jana, B. Ray, S. Khurshid ve V. Shmatikov, "Using frankencerts for automated adversarial testing of certificate validation in SSL/TLS implementations," *2014 IEEE Symposium on Security and Privacy*, Berkeley, A.B.D., 2014, ss. 114-129.

[14] D. Wagner ve B. Schneier, "Analysis of the SSL 3.0 protocol," *The Second USENIX Workshop on Electronic Commerce*, Kaliforniya,A.B.D., 1996.

[15] C. Chen, C. Tian, Z. Duan ve L. Zhao, "RFC- directed differential testing of certificate validaiton in SSL/TLS implementations," *40th International Conference on Software Engineering*, Göteborg, İsveç, 2018, ss. 859-861.

[16] R. Biddle, P. C. Oorschot, A. S. Patrick, J. Sobey ve T. Whalen, "Browser interfaces and extended validation SSL certificates: An empirical study," *Proceedings of the 2009 ACM workshop on Cloud computing security*, Ottawa, Kanada, 2009, ss. 19-25.

[17] C. E. Shannon "Communication Theory of Secrecy Systems," *Communication Theory of Secrecy Systems*, c. 28, sayı 4, ss. 657-715, 1949.

[18] T. Yerlikaya, E. Buluş ve N. Buluş, "Kripto algoritmalarının gelişimi ve önemi," *Akademik Bilişim Konferansları*, 2006, ss. 9-11.

[19] K. Palmgren, "Diffie-Hellman key exchange – A non-mathematician's explanation," *Information Systems Security Association Journal*, ss. 1-9, 2006

[20] W. Diffie ve M. E., "Hellman new directions in cryptography," *IEEE transactions on Information Theory*, c. 22, sayı 6, ss. 644– 654, 1976.

[21] S. Nakamoto. (2020, 8 Ekim) *Bitcoin: A peer-to-peer electronic cash system*. [Online]. Erişim: https://bitcoin.org/bitcoin.pdf.

[22] J. A. Jaoude ve R. G. Saade, "Blockchain applications – usage in different domains," *IEEE Access*, c. 7, ss. 45360-45381, 2019.

[23] E. Karaarslan ve M. F. Akbaş, "Blokzinciri tabanlı siber güvenlik sistemleri," *Uluslararası Bilgi Güvenliği Mühendisliği Dergisi*, c. 3, sayı 2, ss. 16-21, 2017.

[24] T. Dinh, R. Lui, M. Zhang ve B. Ooi, "A data processing view of blockchain system," *IEEE Transactions on Knowledge and Data Engineering*, ss. 1366-1385, 2018.

[25] C. Yalçın, "Blokzinciri mimarisinin nesnelerin internetine uyarlanması ve performans analizleri," Yüksek lisans tezi, Bilgisayar Mühendisliği Anabilim Dalı, Fen Bilimleri Enstitüsü, Süleyman Demirel Üniversitesi, Isparta,Türkiye, 2019.

[26] A. F. Mendi ve A. Çabuk, "Bitcoin'in arkasındaki güç: Blockchain," *GSI Journals Serie C: Advancements in Informatıon Sciences and Technologies*, c. 1, sayı 1, ss. 12-23, 2018.

[27] M. Swan, *Blockchain Blueprint for A New Economy*, Kaliforniya, A.B.D: O'Reilly Media Inc., 2015.

[28] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Doktora tezi, Bilgi ve Bilgisayar Bilimleri, Kaliforniya Üniversitesi, Kaliforniya, A.B.D., 2000.

8. EKLER

8.1. EK 1: HTTP DURUM KODLARI

Durum Kodu	Durum Kodu Açıklması
100	Continue
101	Switching Protocols
102	Processing
103	Early Hints
104-199	Unassigned
200	OK
201	Created
202	Accepted
203	Non-Authoritative Information
204	No Content
205	Reset Content
206	Partial Content
207	Multi-Status
208	Already Reported
209-225	Unassigned
226	IM Used
227-299	Unassigned
300	Multiple Choices
301	Moved Permanently
302	Found
303	See Other
304	Not Modified
305	Use Proxy
306	(Unused)
307	Temporary Redirect
308	Permanent Redirect
309-399	Unassigned
400	Bad Request
401	Unauthorized
402	Payment Required
403	Forbidden
404	Not Found
405	Method Not Allowed
406	Not Acceptable
407	Proxy Authentication Required
408	Request Timeout
409	Conflict

410	Gone
411	Length Required
412	Precondition Failed
413	Payload Too Large
414	URI Too Long
415	Unsupported Media Type
416	Range Not Satisfiable
417	Expectation Failed
418-420	Unassigned
421	Misdirected Request
422	Unprocessable Entity
423	Locked
424	Failed Dependency
425	Too Early
426	Upgrade Required
427	Unassigned
428	Precondition Required
429	Too Many Requests
430	Unassigned
431	Request Header Fields Too Large
432-450	Unassigned
451	Unavailable For Legal Reasons
452-499	Unassigned
500	Internal Server Error
501	Not Implemented
502	Bad Gateway
503	Service Unavailable
504	Gateway Timeout
505	HTTP Version Not Supported
506	Variant Also Negotiates
507	Insufficient Storage
508	Loop Detected
509	Unassigned
510	Not Extended
511	Network Authentication Required
512-599	Unassigned

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı : Durdu ÖZDEN ONAR

Yabancı Dili : İngilizce

ÖĞRENİM DURUMU

Derece	**Alan**	**Okul/Üniversite**	**Mezuniyet Yılı**
Yüksek Lisans	Bilgisayar Mühendisliği	Düzce Üniversitesi	2021
Lisans	Bilgisayar Mühendisliği	Düzce Üniversitesi	2017

YAYINLAR

D. Özden Onar ve R. Kara, "SSLChain: Blokzincir Yöntemiyle Sunucu-İstemci Arası Güvenli Web Haberleşmesi," *Düzce Üniversitesi Bilim ve Teknoloji Dergisi*, c. 9, 2021.