T.C.
İSTANBUL ÜNİVERSİTESİ-CERRAHPAŞA
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

GÜVENLİ OLMAYAN SİSTEMLERDE KRİPTO MODÜLLER KULLANARAK ŞİFRELİ ETHERNET HABERLEŞMESİNİN UYGULANMASI

Mevlüt Hulusi GÖNEN

DANIŞMAN
Prof. Dr. Fırat KAÇAR

Anabilim Dalı Adı

Elektrik-Elektronik Mühendisliği Programı

Haziran, 2023

TEZ KABUL VE ONAYI

Mevlüt Hulusi GÖNEN tarafından, **Prof. Dr. Fırat KAÇAR** danışmanlığında hazırlanan **"GÜVENLİ OLMAYAN SİSTEMLERDE KRİPTO MODÜLLER KULLANARAK ŞİFRELİ ETHERNET HABERLEŞMESİNİN UYGULANMASI"** başlıklı bu çalışma, jürimiz tarafından **06/06/2023** tarihinde yapılan sınav sonucunda **oy birliği** ile başarılı bulunarak **Yüksek Lisans Tezi** olarak kabul edilmiştir.

Tez Jürisi

		İmza	**Sonuç**
DANIŞMAN	Prof. Dr. Fırat KAÇAR İstanbul Üniversitesi-Cerrahpaşa Elektrik Elektronik Mühendisliği Anabilim Dalı		☒ Kabul ☐ Ret
ÜYE	Doç. Dr. Muhammed Ali Aydın İstanbul Üniversitesi-Cerrahpaşa Bilgisayar Mühendisliği Anabilim Dalı		☒ Kabul ☐ Ret
ÜYE	Doç. Dr. Yunus Babacan Erzincan Binali Yıldırım Üniversitesi Elektrik Elektronik Mühendisliği Anabilim Dalı		☒ Kabul ☐ Ret

BEYAN

Bu tez çalışmasının kendi çalışmam olduğunu, tezin planlanmasından yazımına kadar bütün aşamalarda etik dışı davranışımın olmadığını, bu tezdeki bütün bilgileri akademik ve bilimsel etik kuralları 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 ve her türlü hukuki sorumluluğu aldığımı kabul ederim.

Mevlüt Hulusi GÖNEN

(İmza)

Annem/Babam/Eşim/Hocam Fırat KAÇAR' a / Aileme ithaf ediyorum…

BÜTÇE DESTEKLERİ

GÜVENLİ OLMAYAN SİSTEMLERDE KRİPTO MODÜLLER KULLANARAK ŞİFRELİ ETHERNET HABERLEŞMESİNİN UYGULANMASI

Bu tez çalışması için herhangi bir kurumdan bütçe desteği alınmamıştır.

TEŞEKKÜR

Başta tüm eğitim hayatım boyunca maddi ve manevi hiçbir desteği esirgemeyen ve beni sürekli teşvik eden babam Prof. Dr. Sait GÖNEN olmak üzere, yine eğitim hayatımda gerekli koşulları hep sağlayan annem Zeliha GÖNEN' e, yüksek öğrenim ve iş hayatımda yanımda olan ve beni sürekli motive eden kıymetli eşim Dr. Busenur GÖNEN' e ve özellikle yüksek lisans programının başından beri eğitimimde büyük katkısı bulunan ve her türlü desteği sağlayan çok değerli danışman hocam Sayın Prof. Dr. Fırat KAÇAR' a sonsuz teşekkürlerimi sunarım.

Haziran 2023 Mevlüt Hulusi GÖNEN

İÇİNDEKİLER

Sayfa No

1. GİRİŞ 1
2. KAVRAMSAL ÇERÇEVE 3
2.1. KRİPTOGRAFİ 3
2.1.1. SİMETRİK KRİPTOGRAFİ 3
2.1.2. ASİMETRİK KRİPTOGRAFİ 4
2.1.3. BLOK ŞİFRELEME 5
2.1.4. AKIŞ ŞİFRELEME 5
2.2. GÖMÜLÜ SİSTEMLER VE MİKRODENETLEYİCİLER 6
2.3. BAGLEBONE BLUE 6
2.4. DEBIAN İŞLETİM SİSTEM 8
2.5. OPENSSL KÜTÜPHANESİ 8
2.6. SUNUCU VE İSTEMCİ HABERLEŞMESİ 9
2.7. TCP/IP 10
2.8. SERİ HABERLEŞME 12
2.9. YAZILIM GELİŞTİRME İÇİN TOOLCHAIN 13
2.10. C ve C ++ KOD DERLEME SÜREÇLERİ 14
2.10.1. ÖNİŞLEMCİ 14
2.10.2. AYRIŞTIRMA 15
2.10.3. SEMANTİK ANALİZ 15
2.10.4. OPTİMİZASYON 15
2.10.5. KOD OLUŞTURMA 15
2.10.6. BİRLEŞTİRME 15
2.10.7. BAĞLANTI KURMA 15
3. YÖNTEM 17
3.1. ÇALIŞMA SENARYOSUNUN BELİRLENMESİ 17
3.2. GELİŞTİRME ORTAMININ OLUŞTURULMASI 18
3.2.1. VS CODE KURULUMU 18
3.2.2. MINICOM KURULUMU 19

3.2.3. GIT KURULUMU 20
3.2.4. CMAKE KURULUMU 20
3.2.5. GNU MAKE KURULUMU 21
3.2.6. OPENSSL KURULUMU 21
3.2.7. LIBTHREAD KURULUMU 22
3.3. KRİPTO MODÜLÜN BELİRLENMESİ 22
3.3.1. GÖMÜLÜ PLATFORM SEÇİMİ 22
3.3.2. KULLANILACAK HABERLEŞME ARAYÜZLERİNİN BELİRLENMESİ 23
3.4. TEST DÜZENEĞİNİN OLUŞTURULMASI 24
3.4.1. SİSTEME GENEL BAKIŞ 24
3.4.2. GÖMÜLÜ SİSTEMİN HAZIR HALE GETİRİLMESİ 25
3.4.3. TEST ORTAMININ FİZİKSEL OLARAK OLUŞTURULMASI 25
3.5. GÜVENLİ HABERLEŞME UYGULAMASI YAZILIM TASARIMI 26
3.5.1. SOKET PROGRAMLAMA YAZILIM TASARIMI 29
3.5.2. SERİ PORT TEST UYGULAMASININ GELİŞTİRİLMESİ 33
3.5.3. KRİPTO MODÜLLERİN YAZILIM TASARIMI 36
4. BULGULAR 43
4.1. ŞİFRELİ HABERLEŞMENİN SİSTEM ÜZERİNDE GERÇEKLEŞTİRİLMESİ 43
4.2. KRİPTOLU HABERLEŞME SIRASINDA VERİ BOYUTUNDAKİ DEĞİŞİM 43
4.3. ŞİFRELİ HABERLEŞMENİN İLETİM HIZINA ETKİSİ 46
4.4. VERİ BOYUTUNUN ŞİFRELEME SÜRESİNE OLAN ETKİSİ 46
4.5. ANAHTAR BOYUTUNUN ŞİFRELEME SÜRESİNE ETKİSİ 47
4.6. AES 256-BIT ŞİFRELEME ALGORİTMASININ GÜVENİRLİĞİ 48
4.7. SİSTEMİN MALİYET TABLOSU 49
5. TARTIŞMA 50
6. SONUÇ VE ÖNERİLER 53

ŞEKİL LİSTESİ

Sayfa No

Şekil 2.1: Simetrik Şifreleme İşlemi 3
Şekil 2.2: Asimetrik Şifreleme İşlemi 4
Şekil 2.3: BeagleBone Blue Görseli 7
Şekil 2.4: OpenSSL Kütüphanesinin Mimarisi 9
Şekil 2.5: Sunucu ve İstemci Bağlantısı 10
Şekil 2.6: OSI Katmanları 12
Şekil 2.7: UART Haberleşme Paket Yapısı 13
Şekil 3.1: Cihazlar Arası RS232 Haberleşmesi 17
Şekil 3.2: Güvenli Kablosuz Haberleşme 18
Şekil 3.3: VS Code İndirme Adresi 19
Şekil 3.4: VS Code İndirme Komutu 19
Şekil 3.5: Minicom İndirme Komutu 19
Şekil 3.6: Git Kurulum Komutu 20
Şekil 3.7: CMAKE Kurulum Komutu 20
Şekil 3.8: Make Kurulum Komutu 21
Şekil 3.9: OpenSSL Geliştirme Kütüphanesi Kurulumu 22
Şekil 3.10: Libthread Kütüphanesi Kurulumu 22
Şekil 3.11: BeagleBone Blue Claud Ide 23
Şekil 3.12: Şifreli Haberleşme Tasarımı 24
Şekil 3.13: BeagleBone Blue Lokal Ağa Bağlanma Komutları 25
Şekil 3.14: Test Düzeneği 26
Şekil 3.15: GHU Akış Diyagramı 28

Şekil 3.16: Minicom Çalıştırma Komutu 28
Şekil 3.17: Soket Uygulaması 29
Şekil 3.18: Sunucu Programı Yazılım Blok Gösterimi 29
Şekil 3.19: İstemci Programı Yazılım Blok Gösterimi 30
Şekil 3.20: Kripto Cihazların Lokal Ağ Bağlantı Testi 30
Şekil 3.21: SSH Bağlantısı Komutu 31
Şekil 3.22: İstemci Hedef IP Tanımlama 32
Şekil 3.23: Make Derleme Komutu 32
Şekil 3.24: Cmake Derleme Komutu 32
Şekil 3.25: Sunucu ve İstemci Bağlantısı Kurulumu 33
Şekil 3.26: Bağlı Cihaz Listeleme Komutu 34
Şekil 3.27: RealTerm Yapılandırması 35
Şekil 3.28: UART Haberleşme Testi 35
Şekil 3.29: AES Şifreleme Akış Diyagramı 37
Şekil 3.30: AES Şifre Çözme Akış Diyagramı 38
Şekil 3.31: OpenSSL Anahtar Oluşturma Komutu 39
Şekil 3.32: OpenSSL Kütüphaneleri 39
Şekil 3.33: İstemciye Gönderilen UART Test Paketi 40
Şekil 3.34: İstemci Uygulamasının Okuduğu UART Test Verisi 41
Şekil 3.35: Sunucuya Gönderilen UART Test Paketi 41
Şekil 3.36: Sunucu Uygulamasının Okuduğu UART Test Paketi 42
Şekil 4.1: Bir Bayt Verinin Boyutu 44
Şekil 4.2: İki Bayt Verinin AES 256-Bit Şifreleme Sonucu Büyülük Değişimi 44
Şekil 4.3: 21 Bayt Verinin AES 256-Bit Şifreleme Sonucu Büyüklük Değişimi 45

TABLO LİSTESİ

Sayfa No

Tablo 4.1: Farklı Büyüklükteki Verilerin AES 256-Bit Şifreleme Sonucu Büyüklükleri.....45

Tablo 4.2: Farklı Boyutlardaki Verilerin Şifrelenme Süreleri.....46

Tablo 4.3: Farklı Boyutlardaki Verilerin Şifre Çözülme Süreleri.....47

Tablo 4.4: Faklı Anahtar Boyutları İle Şifreleme Süreleri.....48

Tablo 4.5: Maliyet Tablosu.....49

Tablo 5.1: Performans Karşılaştırması.....52

SİMGE VE KISALTMA LİSTESİ

Simgeler **Açıklama**

ms : Milisaniye
s : Saniye
usd : United State Dollar

Kısaltmalar **Açıklama**

AST : Abstract Syntax Tree
AES : Advanced Encryption Standard
BB : BeagleBone Blue
CBC : Chipper Block Cryptography
DES : Data Encyption Standard
DSA : Digital Signature Algorithm
DNS : Domain Name System
ECC : Elliptic Curve Cryptography
FTP : File Transfer Protocol
GHU : Güvenli Haberleşme Uygulaması
HTTP : Hyper Text Transfer Protocol
IV : Initial Vector
I2C : Inter-Integrated Circuit
IP : Internet Protocol
IPSec : Internet Protocol Security
UART : Universal Asynchronous Receiver Transmitter
USB : Universal Serial Bus
PRNG : Pseudo Random Number Generator
RS232 : Recommended Standard-232
RSA : Rivest-Shamir-Adleman
SHA : Secure Hash Algorithm
SSH : Secure Shell
TCP : Transmission Control Protocol

TLS : Transistor-Transistor Logic

ÖZET

YÜKSEK LİSANS TEZİ

GÜVENLİ OLMAYAN SİSTEMLERDE KRİPTO MODÜLLER KULLANARAK ŞİFRELİ ETHERNET HABERLEŞMESİNİN UYGULANMASI

Mevlüt Hulusi GÖNEN

İstanbul Üniversitesi-Cerrahpaşa

Lisansüstü Eğitim Enstitüsü

Elektrik-Elektronik Mühendisliği Anabilim Dalı

Elektrik-Elektronik Mühendisliği Programı

Danışman : Prof. Dr. Fırat KAÇAR

II. Danışman : Unvan Ad SOYAD

Bilgi güvenliği tarihten günümüze gelen ve gelecekte de olması muhtemel olan temel bir ihtiyaçtır. Günümüzde bilgi güvenliği ihtiyacının endüstriye iz düşümü kapsamında olan veri iletiminin güvenlik gereksinimi, kripto modüller kullanarak güvenli haberleşmenin sağlanması için yaptığımız çalışmaya da ilham kaynağı olmuştur. Çalışmanın amacı, endüstride kullanılmakta olan mevcut sistemlere mümkün olan en az müdahale ile bir kripto köprü kurularak veri iletim güvenliğini sağlamaktır. Çalışma kapsamında güvensiz bir haberleşme örneği seçilmiş ve haberleşme güvenliğini sağlamak için gömülü sistemler ile oluşturulan kripto modüller kullanılmıştır. Veri iletiminin şifreli olmasını sağlayacak olan Güvenli Haberleşme Uygulamasının yazılım geliştirmesi yapılmış ve güvenli haberleşme için yapılan bir dizi şifreleme ve şifre çözme işlemleri test ortamında gerçekleştirilerek sonuçları incelenmiştir.

Mayıs 2023 , 77 sayfa.

Anahtar kelimeler: Kripto, Şifreleme, Şifre Çözme , Gömülü Sistem, Güvenli Haberleşme

ABSTRACT

M.Sc. THESIS

IMPLEMENTATION of ENCRYPTED ETHERNET COMMUNICATION by USING CRYPTO MODULES in UNSECURE SYSTEMS

Mevlüt Hulusi GÖNEN

İstanbul University-Cerrahpaşa

Institute of Graduate Studies

Department of Electrical and Electronics Engineering

Electrical and Electronics Engineering Program

Supervisor : Prof. Dr. Fırat KAÇAR

Co-Supervisor: Academic Title Name SURNAME

Information security is a fundamental need that has existed throughout history and is likely to exist in the future. Today, the security requirement of data transmission, which is within the scope of the projection of the need for information security to the industry, has inspired our study to provide secure communication using crypto modules. The aim of the study is to provide data transmission security by establishing a crypto bridge with the least possible intervention to the existing systems used in the industry. Within the scope of the study, an insecure communication example was selected and crypto modules created with embedded systems were used to ensure communication security. The software development of the Secure Communication Application, which will ensure that the data transmission is encrypted, has been made and a series of encryption and decryption operations for secure communication are performed in a test environment and the results are analyzed.

May 2023, 77 pages.

Keywords: Crypto, Encryption, Decryption, Embedded System, Secure Communication

1. GİRİŞ

Tarihe bakıldığında, çok eski zamanlardan beri insanlığın şifrelemeye ihtiyaç duyduğu görülecektir. Hükümdarlar, gizli mesajları açık bir şekilde iletmek yerine, anlaşılmış bir şifre veya çerçeve kullanarak iletmeyi tercih etmişlerdir. Bunun nedeni, rakiplerin bu bilgiyi yol boyunca elde etme olasılığıdır. Bu sebeple bilgiler haberciden bile gizli tutulduğu durumlar olmuştur[1]. Şifreleme talebi tarih boyunca giderek artmıştır. Sezar şifresi ilk olarak M.Ö. 100'lerde ortaya çıktı[2]. Bu şifreleme türünde, her harfin tekrar tekrar kaydırılmasıyla şifreleme oluşturulmuştur. Sezar Sistem bilgisine sahip herhangi biri verilerin şifresini çözebilir ve düz metni elde edebilirdi. Daha sonra, şifreleme yöntemleri çeşitlendirilerek birçok farklı teknik kullanılmıştır ve bu teknikler günümüze kadar geliştirilerek gelmiştir[3].

Günümüz dünyasında da benzer şekilde bilgi güvenliği her geçen gün önemini artırmaktadır ve şu anda endüstrinin birçok alanının yanı sıra, kimlik doğrulama, şifreleme, banka kartları, kablosuz telefonlar, e-ticaret, erişim kontrolü (araç kilit sistemleri, kısıtlı alanlar) ve ödeme yöntemleri gibi alanlarda siber güvenlik hayati önem taşımaktadır[2]. Genellikle endüstride faydaya ve fonksiyonelliğe odaklanılırken güvenlik hususları göz ardı edilmekte ve bu durum bazı güvenlik zafiyetlerine yol açabilmektedir[4]. Bir tesisin sermayesini, personelini ve fikri mülkiyetini korumak o tesis için kritik öneme sahiptir. Bu sebeple ilgili tesis için katma değeri bulunan bilgiler, cihazlar ve bu cihazların birbirleriyle ve varsa dışarıyla olan iletişimleri güvenli olmalıdır[5]. Endüstri ve otomasyonda güvenli iletişime ihtiyaç duyulmasının başlıca nedenlerinden biri kritik verilere istenmeyen erişimi önlemektir. Ticari sırlar, fikri mülkiyet ve kişisel olarak tanımlanabilir bilgiler, endüstriyel sistemlerin ve ekipmanların sıklıkla depoladığı ve aktardığı değerli ve gizli bilgilerin örnekleridir. Bilgisayar korsanları veya siber suçlular bu bilgileri ele geçirirse, hırsızlık, sabotaj veya gasp gibi kötü amaçlar için kullanılabilir[6]. Güvenli iletişim, cihazlar arasında gönderilen verilerin bütünlüğünü sağlamak için de gereklidir. Endüstriyel ortamlarda, yanlış veya değiştirilmiş veriler hasarlı ürünler veya verimsiz prosedürler gibi önemli sorunlara yol açabilir.

Ayrıca, şifreli iletişim, verilerin kaybolmasını, bozulmasını veya gerektiğinde erişilememesini önleyerek veri kullanılabilirliğini sağlamaya yardımcı olabilir[7]. Bu kapsamda güvenlik, sistem entegrasyonu, mobilite, enerji verimliliği ve donanım güvenilirliği gereksinimlerinin

artmasıyla birlikte, kriptolojik donanımların teknolojiye daha da fazla nüfuz edeceği öngörülmektedir[8]. Bu çerçevede sistemlerin güvenliğinin sağlanması veya artırılması amacıyla spesifik kripto modüller ve özelleştirilmiş yazılımlar kullanılmaktadır. Bahse konu olan kripto modüller düşük maliyetlerle mikro işlemciler ve mikro denetleyiciler kullanılarak gömülü sistemlerce sağlanabilmektedir. Bu gömülü sistemler, karmaşık savunma sistemlerinden ev aletlerine kadar çeşitli uygulamalarda kullanılmaktadır[9].

Bilgi güvenliği ihtiyacı her geçen gün artarken, endüstride işletmelerin mevcut kurulu düzenleri ve prosedürleri bulunmaktadır ve söz konusu şirketler için sıfırdan daha güvenli sistemlerin kurulumu zaman ve kaynak açısından maliyetli olacaktır. Bu çalışmada Mümkün olan en az değişiklik ve en az maliyet ile sistemlerin güvenliğinin sağlanmasına odaklanılmıştır.

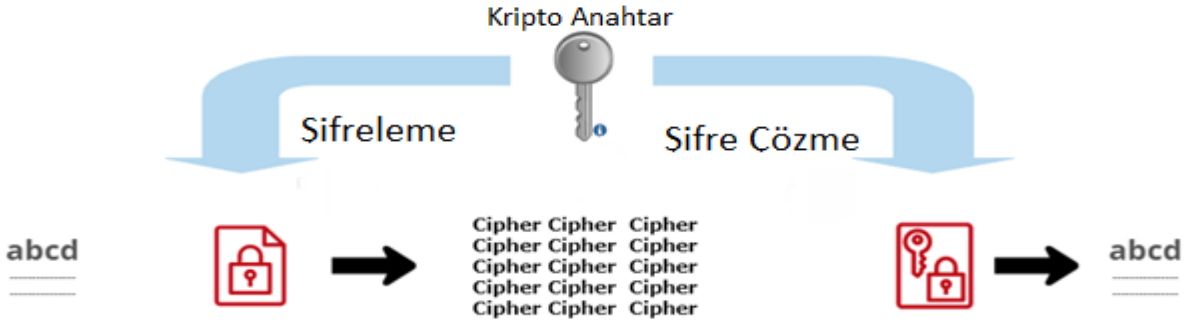
2. KAVRAMSAL ÇERÇEVE

2.1. KRİPTOGRAFİ

Kriptografi, üçüncü tarafların varlığında güvenli bir şekilde iletişim kurma uygulamasıdır. İletişimi ve veri korumasını güvence altına almak için matematiksel algoritmaların ve protokollerin kullanılmasını gerektirir. Kriptografinin amacı, bilgiyi yalnızca yetkili kişilerin anlayabileceği okunamaz bir biçime dönüştürerek gizliliğini, bütünlüğünü ve geçerliliğini sağlamaktır. Kriptografinin, iki kişi arasındaki iletişimin sağlanmasından veri tabanlarında tutulan hassas verilerin korunmasına kadar çok sayıda uygulaması vardır. Şifreleme, şifre çözme, dijital imzalar ve hash fonksiyonları kriptografik prosedürlerin örnekleridir. Kriptografi, modern iletişimde hassas bilgilerin güvenliğini ve gizliliğini korumak için kritik öneme sahiptir[10].

2.1.1. SİMETRİK KRİPTOGRAFİ

Simetrik kriptografi, iletişimin aynı gizli anahtar kullanılarak şifrelendiği ve şifresinin çözüldüğü bir şifreleme türüdür. Başka bir deyişle, Şekil 2.1' deki gibi düz metin iletişimini şifreli metne dönüştürmek için kullanılan anahtar, aynı şekilde şifreli metni tekrar düz metne dönüştürmek için de kullanılır. Buna karşılık, asimetrik kriptografi de şifreleme ve şifre çözme için ayrı birer anahtar kullanır. Simetrik kriptografi de gizli anahtar gizli tutulmalıdır çünkü anahtara erişim sağlayan herkes bu anahtarla şifrelenmiş mesajları deşifre edebilir[11].

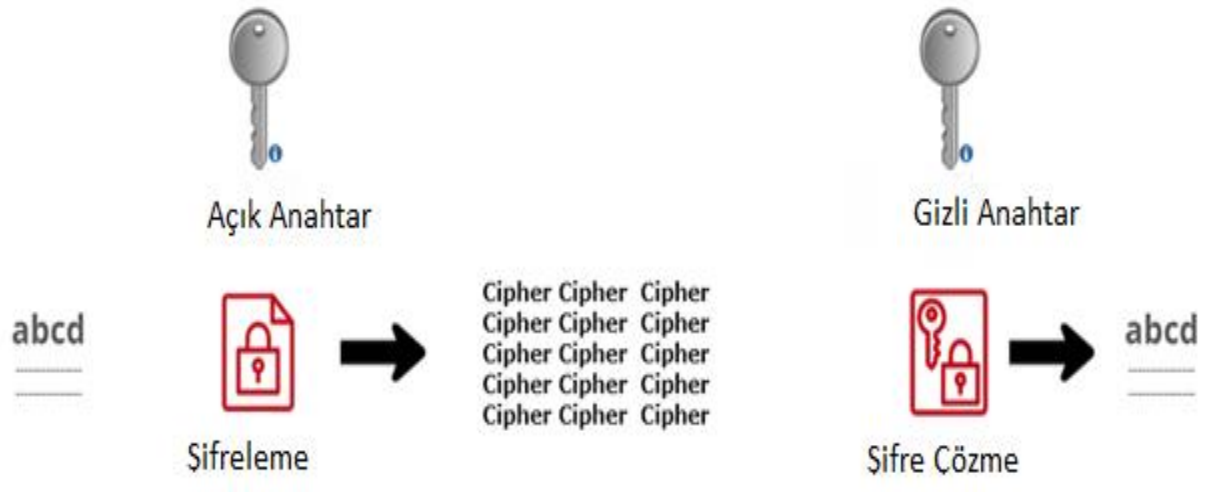


Şekil 2.1: Simetrik Şifreleme İşlemi

Secure Socket Layer (SSL)/ Secure Socket Layer (SSL)/Secure Shell (SSH) ve Internet Protocol Security (IPSec) gibi güvenli iletişim protokollerinde ve AES (Advanced Encryption Standard), Data Encryption Standard (DES) ve 3DES gibi şifreleme tekniklerinde sıklıkla simetrik kriptografi kullanılır. Genel olarak asimetrik kriptografiden daha hızlı ve daha verimlidir, ancak hem göndericinin hem de alıcının aynı gizli anahtara erişmesini gerektirir, bu da bazı durumlarda güvenli bir şekilde sürdürülmesi zor olabilir[12].

2.1.2. ASİMETRİK KRİPTOGRAFİ

Genellikle açık anahtarlı kriptografi olarak bilinen asimetrik kriptografi, bir çift anahtar kullanarak verileri şifreleyen ve şifrelerini çözen bir kriptografik tekniktir. Hem şifreleme hem de şifre çözme için aynı anahtarı kullanan simetrik kriptografinin aksine, Şekil 2.2' deki gibi asimetrik kriptografi verileri şifrelemek için bir anahtar ve şifresini çözmek için farklı ancak matematiksel olarak ilişkili bir anahtar kullanır. Açık anahtar ve özel anahtar asimetrik kriptografide kullanılan iki anahtardır. Mesajları şifrelemek için kullanılan açık anahtara herkesin erişimi vardır. Öte yandan özel anahtar yalnızca anahtar çiftinin sahibi tarafından bilinir ve mesajların şifresini çözmek için kullanılır[13].



Şekil 2.2: Asimetrik Şifreleme İşlemi

Asimetrik kriptografinin birçok şekli vardır, ancak en çok kullanılan ikisi Rivest-Shamir-Adleman (RSA) ve Elliptic Curve Cryptography (ECC)'dir. RSA 1970'lerde oluşturulan popüler bir açık anahtar şifreleme yöntemidir. Büyük tamsayıları asal çarpanlara ayırmanın ne kadar zor olduğuna dayanır. RSA, güvenli e-posta, SSL/TLS ve dijital imzalar gibi birçok uygulamada kullanılmaktadır[14].

ECC, sonlu alanlar üzerinde eliptik eğrilerin cebirsel yapısına dayanan daha yeni bir açık anahtar şifreleme algoritmasıdır. Aynı güvenlik seviyesi için ECC, RSA' dan daha verimlidir ve daha küçük anahtar boyutları gerektirir. ECC güvenli iletişim, dijital imzalar ve kimlik doğrulama gibi çeşitli uygulamalarda kullanılır[12].

Asimetrik kriptografi ayrıca Digital Signature Algorithm (DSA), Diffie-Hellman anahtar değişimi ve ElGamal şifrelemesini de içerir[10].

2.1.3. BLOK ŞİFRELEME

Blok şifreleme, bir mesajın sabit uzunlukta bloklara ayrıldığı ve her bloğun farklı bir şifreleme yöntemi kullanılarak bağımsız olarak şifrelendiği bir şifreleme türüdür. Advanced Encryption Standard (AES), 128, 192 veya 256 bitlik sabit blok boyutlarıyla en yaygın kullanılan blok şifreleme yöntemidir[15].

Düz metin belirli sayıda bloğa ayrılır ve bunların her biri daha sonra farklı bir şifreleme tekniği kullanılarak şifrelenir. Şifreli metni oluşturmak için şifrelenmiş bloklar kesin bir sırayla birleştirilir. Şifreleme işlemi tersine çevrilebilir, yani şifreli metin aynı şifreleme tekniği ve anahtar kullanılarak tekrar düz metne dönüştürülebilir. Blok şifreleme genellikle güvensiz ağlar üzerinden gönderilen hassas verilerin gizliliğini ve bütünlüğünü sağlamak için kullanılır. Ayrıca şifrelenmiş sabit diskler veya bulut depolama hizmetleri gibi veri güvenliği için de kullanılır[16].

2.1.4. AKIŞ ŞİFRELEME

Akış şifreleme, sürekli bir veri akışı ile çalışan bir şifreleme algoritmasıdır. Düz metin her seferinde bir bit veya bayt olarak işlenir ve bu şifreleme yönteminde uygun şifreli metni üretmek için bir anahtar dizisi ile birleştirilir. Bir sözde rasgele sayı üreteci Pseudo Random Number Generator (PRNG), girdi olarak gizli bir anahtar kullanarak anahtar akışını üretir[12].

Modern akış şifreleme algoritmaları, iyi performansı korurken daha yüksek güvenlik seviyeleri sağlamak üzere tasarlanmıştır. Kriptografi topluluğu, çeşitli uygulamalar için güvenliklerini ve uygulanabilirliklerini sağlamak için bu algoritmaları sık sık yoğun inceleme ve testlere tabi tutmuştur [17].

2.2. GÖMÜLÜ SİSTEMLER VE MİKRODENETLEYİCİLER

Mikrodenetleyici, belirli bir işlevi veya sistemi kontrol etmek için tasarlanmış tek çipli bir bilgisayardır. Bir işlemci, bellek ve zamanlayıcılar, seri portlar ve analog-dijital dönüştürücüler gibi giriş/çıkış çevre birimlerini içerir. Mikrodenetleyiciler, daha büyük bir sistemin veya ürünün bir parçası olarak belirli işlevleri yürütmek üzere oluşturulmuş bilgisayar sistemleri olan gömülü sistemlerde sıklıkla kullanılır[18].

Gömülü bir sistem, belirli bir işlevi yerine getirmek için yazılımla birlikte çalışan sensörler, aktüatörler ve ekranlar gibi donanım bileşenlerinden oluşur. Ev otomasyon sistemleri, tıbbi cihazlar, otomobil sistemleri ve endüstriyel kontrol sistemlerinin tümü gömülü sistemlere örnektir[9].

Kompakt, düşük güçlü ve uygun maliyetli olmaları nedeniyle mikrodenetleyiciler genellikle gömülü sistemlerde kullanılır. Ayrıca belirli görevleri yapmak üzere programlanabilirler. Sensörleri ve aktüatörleri izlemek ve kontrol etmek, verileri işlemek ve depolamak ve diğer cihazlar ve sistemlerle iletişim kurmak için sıklıkla kullanılırlar.

Bir mikrodenetleyicinin programlanması genellikle C veya assembly dili gibi düşük seviyeli bir dilde kod oluşturulmasını ve ardından özel bir yazılım geliştirme ortamı kullanılarak kodun derlenip mikrodenetleyiciye yüklenmesini gerektirir. Gömülü sistem geliştirme, sıklıkla donanım ve yazılım tasarımının yanı sıra sistemin spesifikasyonlarını ve gereksinimlerini karşıladığını garanti etmek için test ve doğrulamayı birleştirir[19].

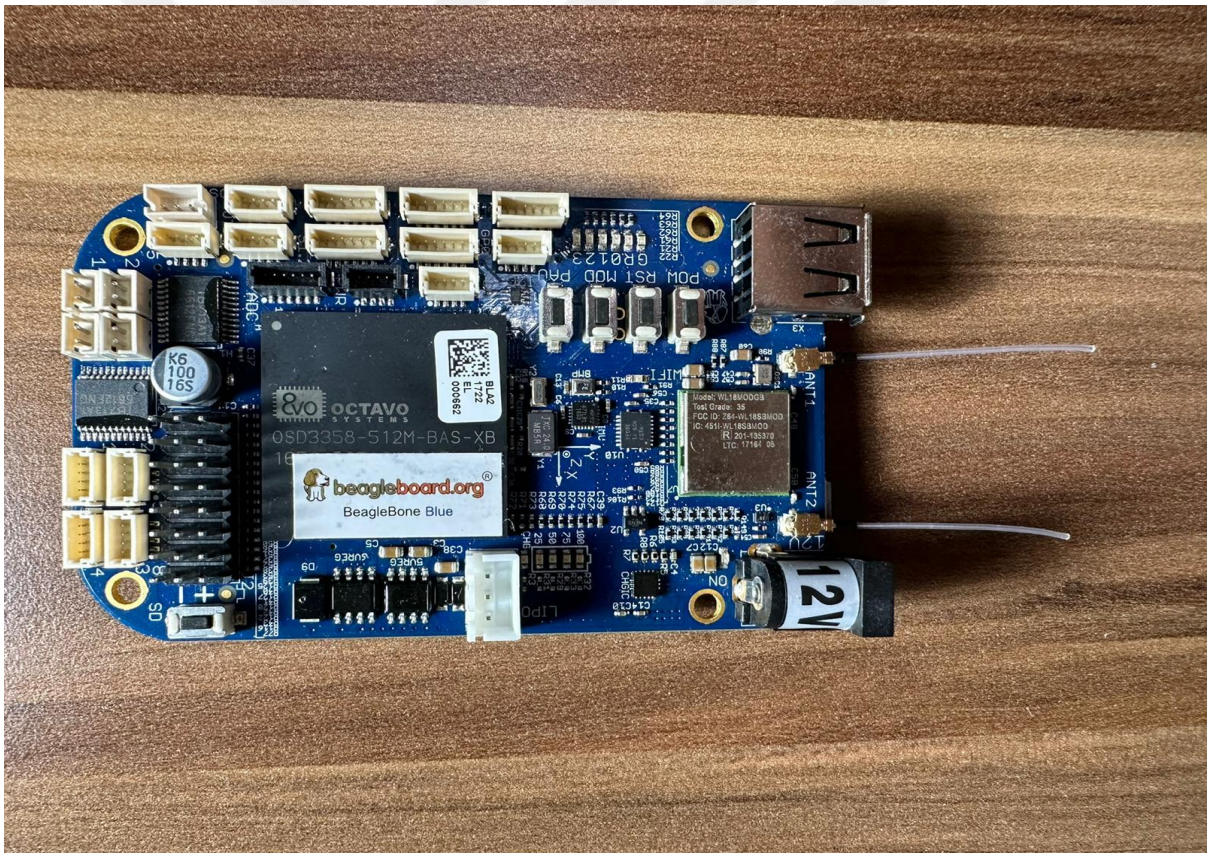
2.3. BAGLEBONE BLUE

BeagleBone Blue (BB) adı verilen tek kartlı bilgisayar düşük maliyetli, açık kaynaklı olarak robotik ve kontrol uygulamaları için üretilmiştir. BeagleBoard serisine aittir ve 1 GHz hızında çalışan ARM Cortex-A8 çekirdekli bir Texas Instruments AM335x işlemci tarafından desteklenmektedir.

Motor kontrolü, sensörler, kablosuz iletişim ve farklı I/O arayüzleri için donanım arayüzlerinin tümü BeagleBone Blue'ya dahildir. Basınç, sıcaklık ve yükseklik ölçümleri için dahili sensörlere de sahip olduğundan drone ve robotik uygulamalarda kullanım için uygundur.[1]

BeagleBone Blue C, C++, Python ve JavaScript gibi programlama dillerini desteklemekle birlikte Debian ve Ubuntu gibi Linux tabanlı işletim sistemlerinde çalışır. Ayrıca, uzaktan kontrol ve kart izleme için entegre bir web tabanlı arayüz içerir, bu da ek donanıma ihtiyaç duymadan projeleri tasarlamayı ve test etmeyi kolaylaştırır.[2]

Genel olarak BeagleBone Blue, güçlü, uyarlanabilir ve uygun fiyatlı tek kartlı bir bilgisayardır ve özellikle robotik ve kontrol uygulamaları için yaratılmıştır. BeagleBone gömülü sistemi Şekil 2.3' deki gibidir.



Şekil 2.3: BeagleBone Blue Görseli

[1] https://ardupilot.org/copter/docs/common-beagle-bone-blue.html/, Mayıs 2023
[2] https://beagleboard.org/blue/, Mayıs 2023

2.4. DEBIAN İŞLETİM SİSTEM

Linux çekirdeğini temel alan Debian, ücretsiz ve açık kaynaklı bir işletim sistemidir. 1993 yılında ilk olarak kullanıma sunulmuştur ve şu anda yazılımı oluşturmak ve geliştirmek için işbirliği yapan bir grup programcı ve kullanıcıdan oluşan topluluk tarafından sürdürülmektedir.

Kullanıcıların yazılım paketlerini kolayca yüklemelerini, güncellemelerini ve kaldırmalarını sağlayan paket yönetim sistemi, Debian'ın temel özelliklerinden biridir. Debian, birçoğu açık kaynaklı ve ücretsiz olan geniş bir yazılım paketi kütüphanesi içerir.

Debian, çeşitli dağıtımlarının yanı sıra GNU/Linux, Hurd ve kFreeBSD sürümleriyle de tanınır. Her dağıtım farklı yazılım paketleri ve araçları ile birlikte gelir ve belirli bir donanım mimarisi için uyarlanmıştır. Ayrıca iş istasyonları ve sunucularda yaygın olarak kullanılmaktadır ve Ubuntu ve Linux Mint de dahil olmak üzere bir dizi tanınmış Linux varyantı için temel oluşturmuştur.[3]

2.5. OPENSSL KÜTÜPHANESİ

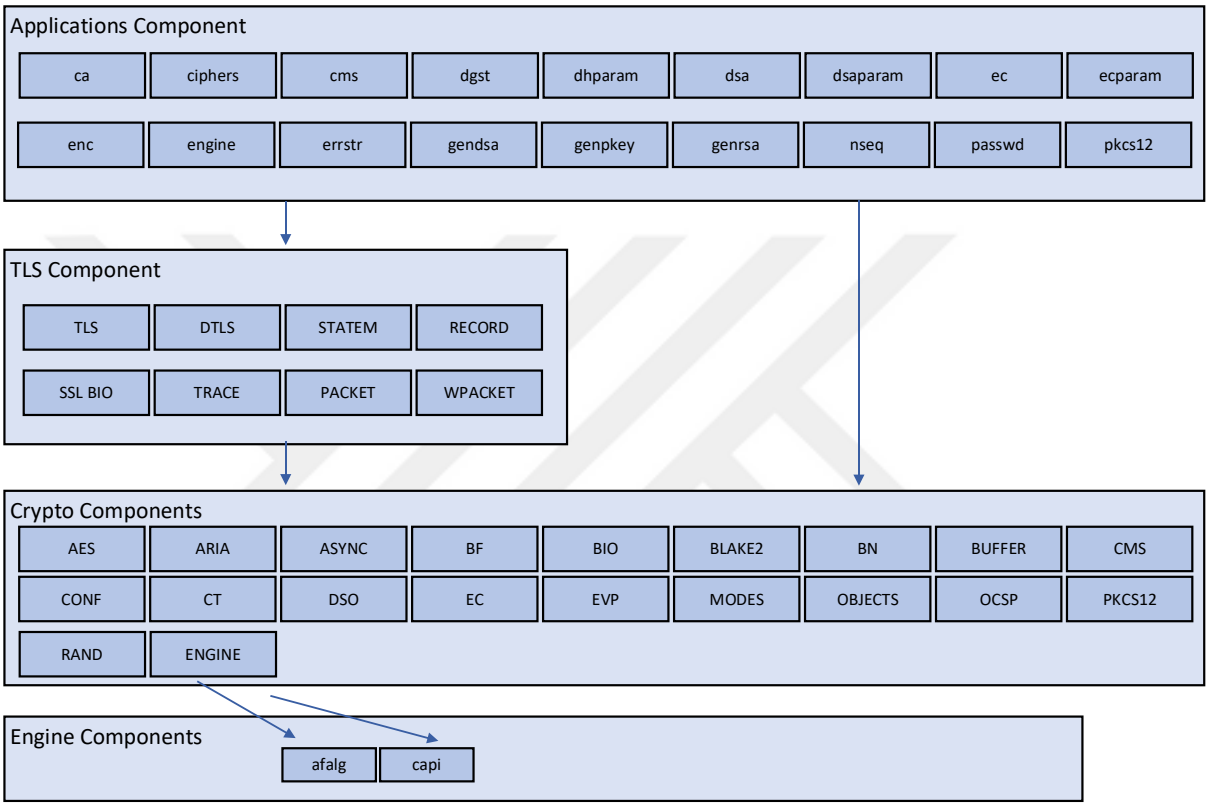
Bilgisayar ağları üzerinden güvenli iletişim, OpenSSL olarak bilinen ve yaygın olarak kullanılan açık kaynak kodlu kriptografi kütüphanesi tarafından desteklenmektedir. Güvenli İnternet bağlantısı için gerekli olan güvenli yuva katmanı SSL ve taşıma katmanı Güvenliği TLS protokolleri esas olarak bu kütüphane kullanılarak uygulanmaktadır. Şifreleme, şifre çözme, dijital imzaların üretimi ve doğrulanması, anahtar değişimi ve diğer kriptografik faaliyetler için geniş bir işlev yelpazesi sunan OpenSSL'yi oluşturmak için C programlama dili kullanılır [20].

OpenSSL'in bir parçası olan komut satırı yardımcı programı, kullanıcıların kriptografik anahtarlar oluşturma, kendinden imzalı sertifikalar üretme ve dijital imzaları kontrol etme dahil olmak üzere bir dizi kriptografik işlemi gerçekleştirmelerini sağlar. AES, RSA ve Secure Hash Algorithm (SHA) OpenSSL tarafından desteklenen kriptografik algoritmalardan sadece birkaçıdır[21].

Uygulamalarda güvenli iletişim kurmak için sağlam ve güvenilir bir kütüphane olan OpenSSL, geliştiriciler ve kuruluşlar tarafından yaygın olarak benimsenmiştir [22]. OpenSSL

[3] https://www.debian.org/intro/people/, Mayıs 2023

kütüphanesinin katmanlı mimari yapısı Şekil 2.4'de gösterilmiştir. Kripto işlemlerin yapıldığı yazılım projelerinde OpenSSL kütüphanesi projeye dahil edilerek, ihtiyaca göre uygulama katmanından gerekli olan kod bloğu seçilip implementasyon içerisinde kullanılır. Şekil 2.4: OpenSSL Kütüphanesinin Mimarisi

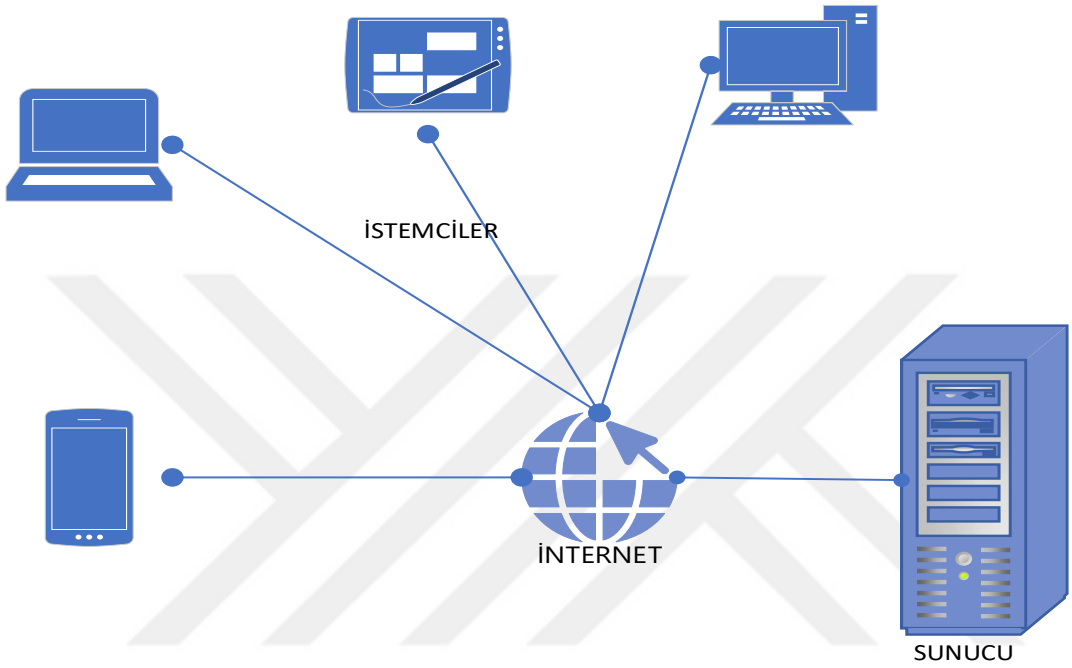


Şekil 2.4: OpenSSL Kütüphanesinin Mimarisi

2.6. SUNUCU VE İSTEMCİ HABERLEŞMESİ

Bir veya daha fazla istemci programa veya cihaza hizmet sunan bir bilgisayar programı veya cihaz, sunucu-istemci iletişiminde sunucu olarak bilinir. Bir sunucudan hizmet talep eden bir bilgisayar programı veya başka bir cihaz ise istemci olarak bilinir. Sunucu ve istemci bağlantısı Şekil 2.5' de gösterilmiştir.

Sunucu, belirli bir bağlantı noktasında gelen istemci isteklerini izler ve istenen verileri sağlayarak veya istenen eylemi gerçekleştirerek istemci isteklerine tepki verir. Sunucu, istemcinin isteğine, istemcinin talep ettiği bilgiyi veya eylemi sağlayarak yanıt verir [23].



Şekil 2.5: Sunucu ve İstemci Bağlantısı

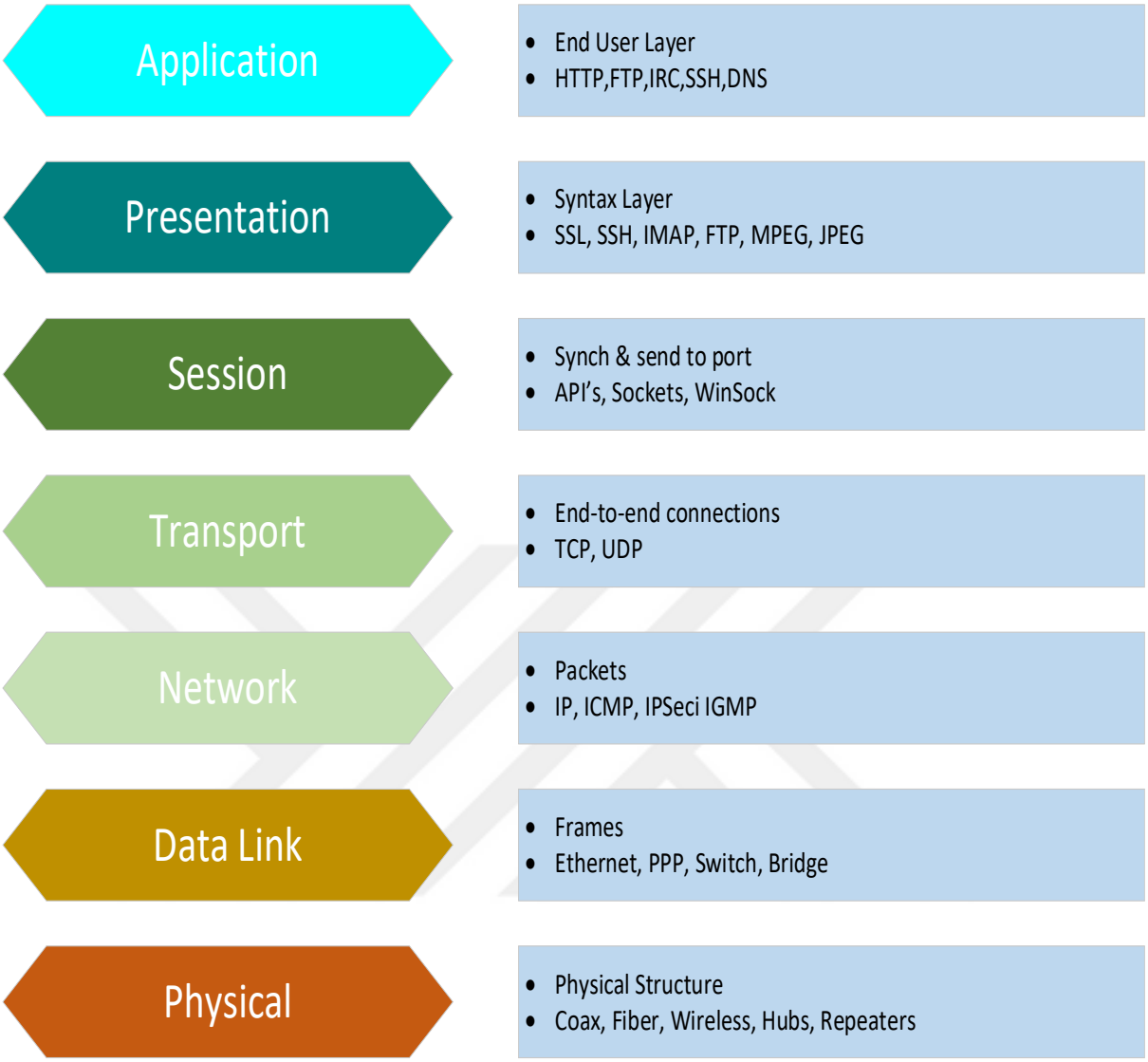
Sunucu ve istemci arasında asenkron veya senkron iletişim mümkündür. Senkron iletişimde istemci yoluna devam etmeden önce sunucunun yanıtını beklerken, asenkron iletişimde istemci isteği gönderir ve hemen çalışmaya devam eder. Web taraması, e-posta, dosya paylaşımı, çevrimiçi oyunlar ve daha fazlası dahil olmak üzere birçok farklı uygulama, sunucu-istemci iletişimine büyük ölçüde güvenmektedir. Bir ağa bağlı çok sayıda cihaz arasında kaynak ve hizmet paylaşımını verimli ve etkili hale getirir[24].

2.7. TCP/IP

Transmission Control Protocol (TCP)/Internet Protocol (IP) olarak da bilinen İletim Kontrol Protokolü/İnternet Protokolü, internet üzerinden cihazdan cihaza iletişim için kullanılan bir iletişim protokoller grubudur. İnternet, veri paketlerinin bilgisayarlar arasında güvenilir ve etkili bir şekilde taşınmasından da sorumlu olan TCP/IP üzerine inşa edilmiştir. TCP'nin

sorumluluğu verilerin küçük paketlere bölünmesi ve ağ üzerinden güvenilir bir şekilde iletilmesi işlemleridir. Bunun yanında hata tespiti ve düzeltmesini sağlar, akışı kontrol eder ve iki cihaz arasında bir bağlantı oluşturur[25].

Öte yandan IP, veri paketlerini çeşitli ağ düğümleri arasında yönlendirmekten de sorumludur. Her cihaza ayrı bir IP adresi verilir ve bu adres daha sonra paketlerin hedeflerine ulaşmak için izleyecekleri en hızlı yolu hesaplamak için kullanılır. TCP ve IP'nin birlikte sağladığı güçlü ve güvenilir iletişim mekanizması sayesinde bilgisayarlar İnternet üzerinden birbirlerine bağlanabilirler. Dosya aktarım protokolleri, e-posta istemcileri ve web tarayıcıları gibi çok çeşitli programlar tarafından kullanılırlar. İnternet üzerinden belirli hizmetleri sunmak için kullanılan diğer protokoller de Hyper Text Transfer Protocol (HTTP), File Transfer Protocol (FTP) ve Domain Name System (DNS) dahil olmak üzere TCP/IP protokol ailesinin bir parçasıdır[26]. Bahse konu bağlantı ve protokolleri kapsayan OSI katmanları Şekil 2.6’da olduğu gibi gösterilmiştir.



Şekil 2.6: OSI Katmanları

2.8. SERİ HABERLEŞME

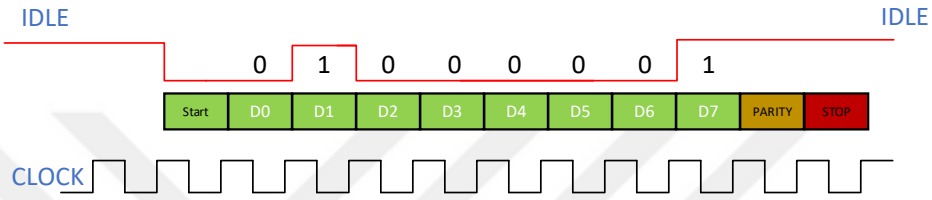
Seri haberleşme, cihazlar arasında tek bir kablo veya kanal aracılığıyla verileri bit bit iletmek için kullanılan bir tekniktir. Tüm bitleri bir kerede göndermek yerine, seri iletişimde veriler sırayla, her seferinde bir bit olacak şekilde iletilir[27].

Asenkron veya senkron seri iletişim mümkündür. Senkron iletişimde veriler belirli bir hızda gönderilir ve verici ile alıcı arasında bir saat sinyali ile senkronize edilir. Asenkron iletişimde herhangi bir saat sinyali kullanılmaz ve veriler rastgele zamanlarda gönderilir.

Seri iletişim telekomünikasyon, bilgisayar ağı ve diğer çeşitli alanlarda yaygın olarak kullanılmaktadır. Sıklıkla kullanılan seri iletişim protokolleri arasında Recommended

Standard-232 (RS-232), Serial Peripheral Interface(SPI), Inter-Integrated Circuit (I2C) ve Universal Asynchronous Receiver Transmitter (UART) bulunmaktadır. Seri haberleşmede tarafların iletilen mesajları anlamlandırabilmesi için bir paket yapısı kullanılmaktadır ve bu paket yapısı Şekil 2.7' de gösterilmiştir.

Seri iletişimin çok sayıda avantajı arasında kullanım kolaylığı, düşük maliyet ve uzak mesafeler arasında veri iletimi kapasitesi yer alır. Paralel iletişime kıyasla nispeten düşük veri taşıma hızı ve verici ile alıcı arasında senkronizasyon gerekliliği dezavantajlarından bazılarıdır[28].



Şekil 2.7: UART Haberleşme Paket Yapısı

RS232 protokolü ile 1 bayt büyüklüğündeki bir verinin iletim hızı, iletişimin yapılandırıldığı baud hızına bağlıdır. Baud hızı saniyedeki sinyal değişimlerinin sayısını belirler. RS232'de her bir veri baytı tipik olarak bir başlangıç biti, 8 veri biti, isteğe bağlı eşlik biti ve durdurma bit veya bitleri ile iletilir. Bu, paritenin kullanılıp kullanılmadığına bağlı olarak bayt başına toplam 10 veya 11 bit anlamına gelir [29].

Bir baytlık verinin iletim süresini hesaplamak için ideal koşullarda ve iletim hızını etkileyebilecek ek ayarlar eve yapılandırmalar hesaba katılmadığında denklem(2.1)' deki g,b, hesaplanabilmektedir.

$$\textit{İletim Süresi} = \textit{Bayt başına bit sayısı} / \textit{Baud Hızı} \tag{2.1}$$

2.9. YAZILIM GELİŞTİRME İÇİN TOOLCHAIN

Yazılım geliştirme bağlamında toolchain, belirli bir görevi veya bir dizi faaliyeti gerçekleştirmek için birlikte çalışan bir yazılım araçları grubudur. Bu teknolojiler genellikle birbirine bağlıdır, yani geliştirme sürecini hızlandırmaya yardımcı olmak için iletişim kurabilir ve veri paylaşabilirler.

Tipik bir yazılım araç zinciri, kaynak kodunu düzenlemek (örnek olarak Git), kod üretmek (örnek olarak GCC), kodu test etmek ve kodu dağıtmak için araçlar içerebilir. Kullanılan araçlar projeye ve geliştirme ortamına bağlı olarak değişebilir.

İyi tasarlanmış bir toolchain, geliştiricilerin daha etkili çalışmasını ve daha yüksek kaliteli kod üretmesini kolaylaştırabilir. Tüm ekip üyeleri aynı araçları ve teknikleri kullandığından, geliştirme sürecinde tutarlılık ve güvenilirliğin sağlanmasına da yardımcı olabilir[30].

2.10. C ve C ++ KOD DERLEME SÜREÇLERİ

Kod derlenmesi, insan tarafından okunabilir kaynak kodunun bir bilgisayarın çalıştırabileceği yürütülebilir bir makine koduna dönüştürülmesi işlemidir. Derleme esnasında derleyicinin sahip olması gereken iki önemli özellik sırası ile verimlilik ve doğruluktur. Bu özellikler aynı zamanda hedef platformun mimarisi ile ilgili özelliklerdir. Buna ek olarak derleyicinin derleme yaparken bellek ve zamanı verimli kullanması beklenmektedir.

2.10.1. ÖNİŞLEMCİ

Bu aşamada ön işlemci kaynak kodu okur ve makro genişletme, başlık dosyası ekleme ve yorum kaldırma gibi görevleri yerine getirir.

Önişlemci komutları, önişlemcinin tanıdığı ve işlediği bir hash (#) sembolü ile başlayan özel komutlardır. Bu yönergeler önişlemciye talimatlar verir ve derleme sürecini düzenler. Burada #include, #define, #ifdef, #ifndef ve yaygın kullanılan diğer direktifler belirtilmiştir. Başlık ekleme aşamasında; derleme için gerekli bildirim ve tanımları içeren başlık dosyalarını dâhil etmek için #include yönergesi kullanılır. Önişlemci tarafından #include yazılmış başlık dosyasının içeriği ile değiştirilir ve bu şekilde dosya içerikleri birleştirilir.

Makrolar sabitleri tanımlamak veya yeniden kullanılabilir kod parçacıkları oluşturmak için kullanılır ve #define komutu ile tanımlanır. Önişlemci, önişleme sırasında makro çağrılarını eşleşen tanımlarıyla değiştirerek genişletir. Bu işlem makro genişletme olarak bilinir.

Koşullu Derleme ise derleme sırasında #ifdef, #ifndef, #if ve diğerleri gibi komutlarla kod bölümlerini koşullu olarak dahil etmek veya etmemek için kullanılır. Belirtilen koşullara bağlı olarak, önişlemci bu kumutları değerlendirir ve ilgili kodu dahil edip etmemeye karar verir.

Bunun yanında önişlemci kaynak koddaki tüm yorumları kaldırır. Yorumlar, açıklayıcı veya tanımlayıcı ifadeler içeren ancak derleyici tarafından hariç bırakılan kısımlardır. Bu yorumlar üretilen kodu daha küçük hale getirmek için kaldırılırlar.

Son olarak satır kontrolü; önişlemci, hata mesajlarının ve ayıklama bilgilerinin kaynak koddaki uygun satırlarla ilişkilendirilmesini gerçekleştiren satır kontrol bilgilerini oluşturur. Dosya adı ve satır numarasını içeren bu bilgiler çoğunlukla hata raporlama ve ayıklama için kullanılır[31].

2.10.2. AYRIŞTIRMA

Bu aşamada önceden işlenmiş kod, bir Abstract Syntax Tree (AST) oluşturmak için derleyici tarafından ayrıştırılır. Parsing aşamasında kod doğrulanır ve programın mantığının yapılandırılmış bir temsili oluşturulur[32].

2.10.3. SEMANTİK ANALİZ

AST derleyici ile anlamsal olarak analiz edilir. Bildirilmemiş değişkenler, tip uyumsuzlukları ve hatalı fonksiyon çağrıları gibi anlamsal hataları tespit eder. Ayrıca tür çıkarımı ve sembol çözümlemesi de kullanır[32].

2.10.4. OPTİMİZASYON

Derleyici, üretilen kodun performansını artırmak için bir dizi optimizasyon işlemi yapabilir. Bunlar, ölü kodun kaldırılması, komutların yeniden düzenlenmesi, sabit katlama, döngü açma ve diğer görevler olarak gerçekleşebilir. Buradaki amaç daha hızlı ve daha küçük yürütülebilir dosyalar elde etmektir[32].

2.10.5. KOD OLUŞTURMA

Optimize edilmiş AST, derleyici tarafından makine kodu oluşturulmasıdır. Hedef mimariye uygun bir şekilde yüksek seviyeli C kodu daha düşük seviyeli makine diline çevrilir[32].

2.10.6. BİRLEŞTİRME

Bu aşamada, assembler assembly dili kodunu alır ve makine kodu talimatları ve verilerinden oluşan nesne koduna çevirir[32].

2.10.7. BAĞLANTI KURMA

Derleme esnasında uygulama harici kütüphaneler gerektiriyorsa veya çok sayıda kaynak dosyası varsa, bağlayıcı üretilen nesne dosyalarını birleştirir ve aralarındaki referansları çözer.

Gerekli olan tüm fonksiyonların ve değişkenlerin uygun bir şekilde eklenmesini sağlayarak son olarak çalıştırılabilir bir dosya elde edilmesini sağlar[32].

3. YÖNTEM

3.1. ÇALIŞMA SENARYOSUNUN BELİRLENMESİ

Çalışma da mevcut durumda kullanılan bir haberleşmenin sisteme mümkün olduğunca müdahale etmeden ve düşük maliyetli olacak şekilde güvenli hale getirilmesine ve bunun sonuçlarının nasıl olacağına odaklanılmıştır.

Bu doğrultuda kripto modüller için yeterli performansa sahip, düşük maliyetli, erişilebilir ve hazır bir gömülü platform seçilmiş ve bu gömülü platform üzerinde çalışacak Güvenli Haberleşme Uygulaması(GHU) geliştirilmiştir.

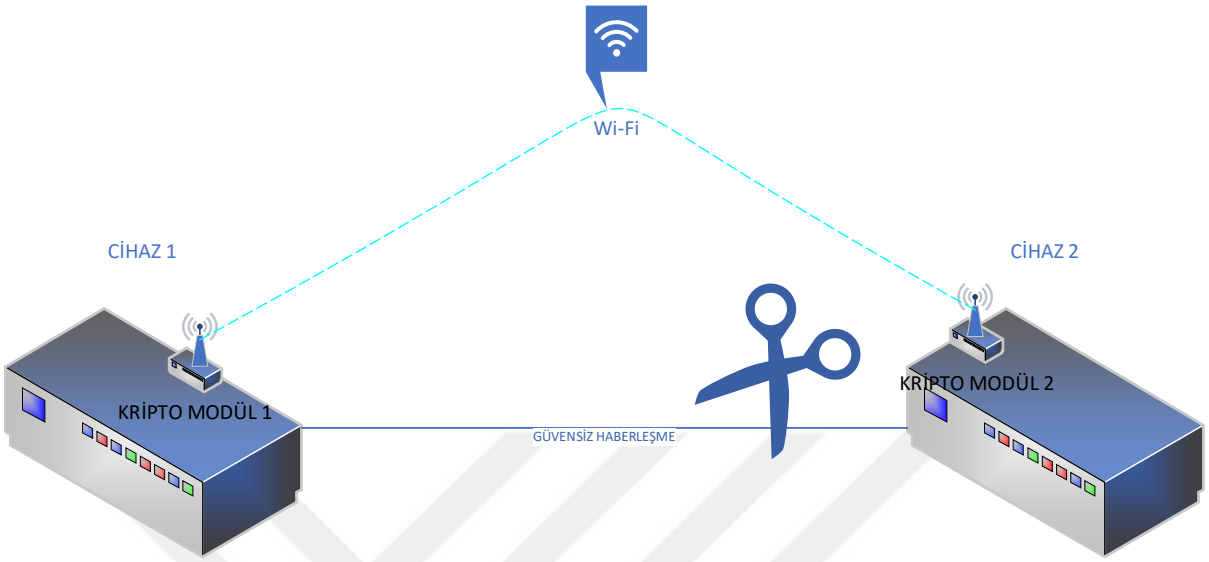
Örnek olarak; endüstride iki tane cihazın Şekil 3.1' deki gibi güvensiz bir haberleşmeye sahip olduğu düşünüldüğünde, Şekil 3.2' deki gibi kripto modüller sisteme dahil edilerek sistemin haberleşmesi güvenli hale getirilebilmektedir.



Şekil 3.1: Cihazlar Arası RS232 Haberleşmesi

Bunun yanında mevcutta olan güvensiz haberleşme RS232 protokolü olarak düşünülmüştür ve iki cihaz arasında kablolu seri haberleşme olduğu varsayılmıştır. Bahsedilen güvensiz haberleşme iki tane gömülü sistem yerleştirilerek güvenli hale getirilmiştir. Çalışma senaryosuna göre Şekil 3.2' deki gibi iki kripto gömülü devrenin birisi cihaz1'e diğeri cihaz2'ye bağlı olacaktır ve bu kripto cihazlar birbirleri ile kablosuz olarak haberleşeceklerdir. Bu

haberleşme çift yönlü olabileceği için kripto cihazlar o anki duruma göre hem verici hem alıcı olarak davranabileceklerdir.



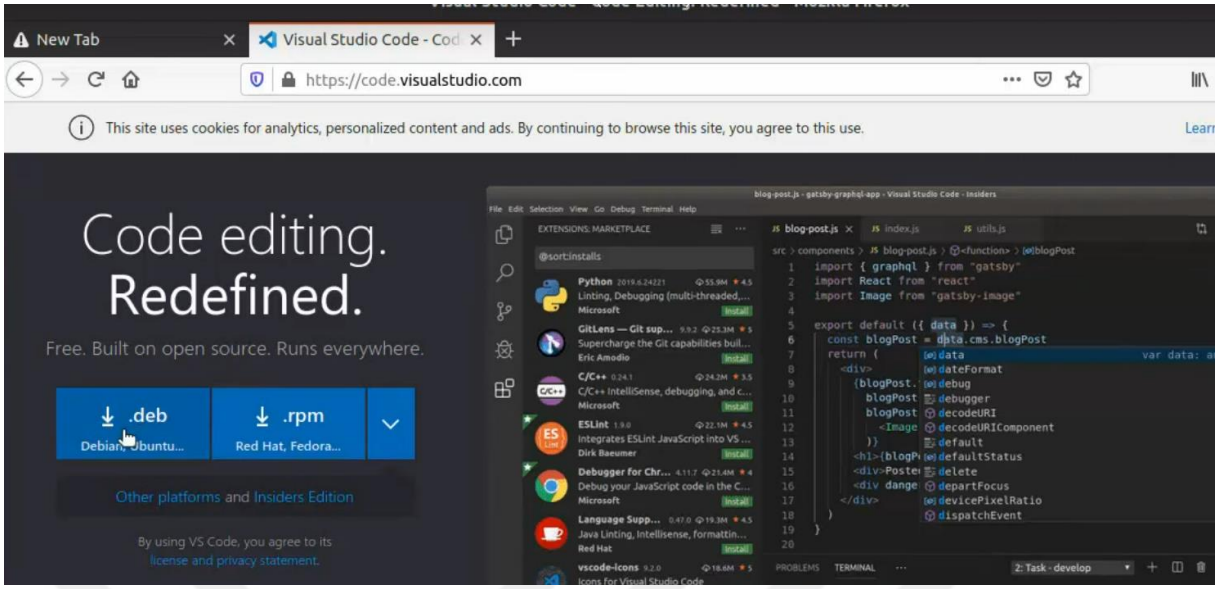
Şekil 3.2: Güvenli Kablosuz Haberleşme

3.2. GELİŞTİRME ORTAMININ OLUŞTURULMASI

Uygulamaların geliştirilmesi için ihtiyaç duyulan ortam oluşturulmuştur. Bu kapsamda Windows 10 işletim sistemi içerisine Oracle VM VirtualBox sanal makinesi kurulmuş ve bu sanal makine de çalışacak şekilde Ubuntu işletim sisteminin 22.04 versiyonu kurulmuştur. Devamında ise geliştirme ortamı için gerekli olan bir dizi araç kurulumu gerçekleştirilmiştir. Kurulan bu araçlar alt başlıklarda anlatılmıştır.

3.2.1. VS CODE KURULUMU

VS Code, Microsoft tarafından geliştirilen ücretsiz ve açık kaynaklı bir kod düzenleyicisidir. Geliştirme ortamında kod editörü olarak kullanılmıştır. Kod düzenlemenin yanında, özelleştirilebilmesi ve birçok eklentisi ile kod geliştirmeyi kolaylaştıran bir araçtır. Örneğin Git entegrasyonu ile toolchain oluşturulmasına da katkı sağlamaktadır. Şekil 3.3'deki gibi visualstudio web sayfasından kurulum dosyası indirildikten sonra Şekil 3.4'deki komut ile yükleme tamamlanmıştır.



Şekil 3.3: VS Code İndirme Adresi

```
ubuntu@ubuntu-VirtualBox:~$ cd Downloads/
ubuntu@ubuntu-VirtualBox:~/Downloads$ ls
code_1.73.1-1667967334_amd64.deb
ubuntu@ubuntu-VirtualBox:~/Downloads$ sudo dpkg -i code_1.73.1-1667967334_amd64.deb
```

Şekil 3.4: VS Code İndirme Komutu

3.2.2. MINICOM KURULUMU

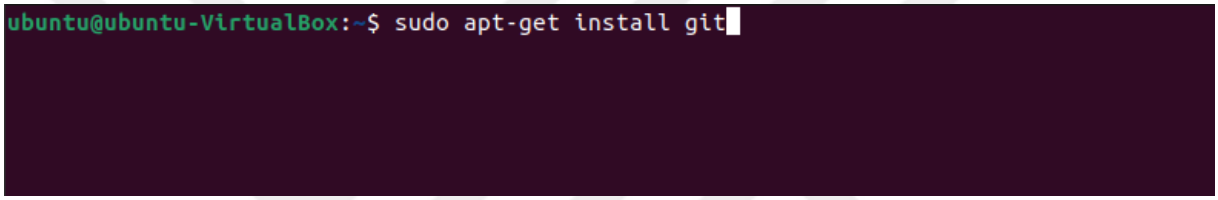
Minicom, veri iletmek ve almak için kullanılan açık kaynaklı bir komut satırı arayüzüdür. Dosya aktarımlarını, seri port yapılandırılmasını ve konut dosyası çalıştırmayı mümkün kılar. Uygulamamızda seri port üzerinden veri göndermek ve veri almak için kullanılmıştır. Ayrıca seri port haberleşme kod bloklarının geliştirilme aşamasında test amaçlı kullanılmıştır. Şekil 3.5'deki komut kullanılarak yüklemesi gerçekleştirilmiştir.

```
ubuntu@ubuntu-VirtualBox:~$ sudo apt-get install minicom
```

Şekil 3.5: Minicom İndirme Komutu

3.2.3. GIT KURULUMU

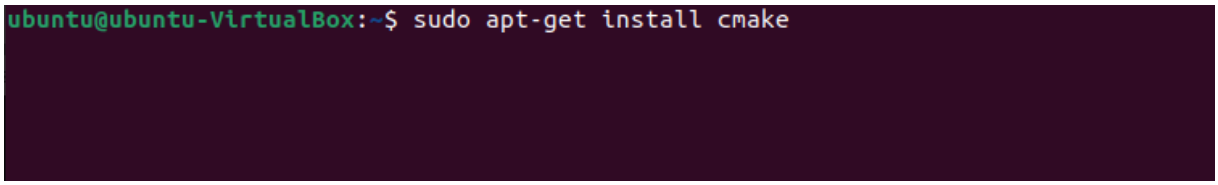
Git, birden fazla geliştiricinin birbirlerinin çalışmalarına müdahale etmeden ortak kod üzerinde geliştirme yapabilmesine ortam oluşturmaktadır. Git, kod üzerinde yapılan tüm değişikliklerin kaydını tutarak; geliştiricilerin kod üzerinde kolayca işbirliği yapmasını, birbirlerinin kodlarını kontrol etmelerini, geçmişe dönük hataların bulunmasını ve hataların düzeltilmesine olanak sağlamaktadır. Git proje ile ilgili tüm dosyaları içinde bulunduran bir repo oluşturur ve bu repo yerel bilgisayara kopyalanabilir, değişiklikler yapılabilir ve sonra ana depoya tekrar gönderilir. GHU geliştirmesinde depolama, versiyon takibi, konfigürasyon takibi ve kod değişikliklerinin takibi için Git versiyon 2.34.1 kullanılmıştır. Git kurulumu Şekil 3.6'daki komut kullanılarak gerçekleştirilmiştir.



Şekil 3.6: Git Kurulum Komutu

3.2.4. CMAKE KURULUMU

Yazılım projelerinin derleme sürecini platformdan bağımsız bir şekilde gerçekleştirmek için kullanılan CMake, yazılım oluşturmaya, test etmeye ve paketlemeye yardımcı olan açık kaynaklı bir build system generator çeşididir. Projenin kaynak dosyalarını ve bağımlılıklarını belirten CMakeList.txt dosyası kullanarak; CMake, Makefiles, Visual Studio ve Xcode gibi çeşitli derleme ortamları için yerel derleme komut dosyaları oluşturur. Toolchain oluşturmak ve platform bağımsız derleme yapabilmek için geliştirme ortamına Şekil 3.7'deki komut ile CMake versiyon 3.22.1 kurulumu gerçekleştirilmiştir.



Şekil 3.7: CMAKE Kurulum Komutu

3.2.5. GNU MAKE KURULUMU

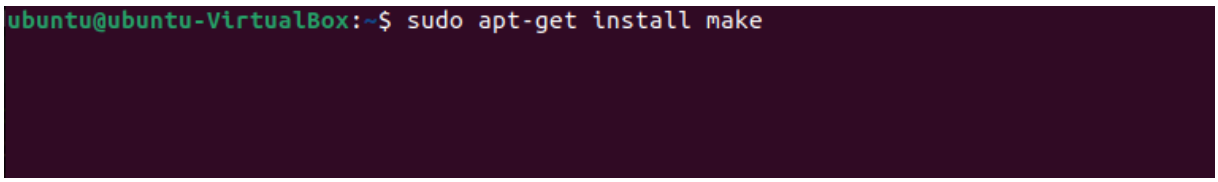
GNU Make, yazılım geliştirmede kaynak koddan yürütülebilir dosyaların ve kütüphanelerin oluşturulmasını otomatikleştirmek için yaygın olarak kullanılan bir derleme otomasyon aracıdır. Make, bir veya daha fazla kaynak dosyadan genellikle yürütülebilir dosya veya kaynak kod olan bir hedefin nasıl oluşturulacağını belirten basit bir metin dosyası olan Makefile'dan bir dizi talimat okur.

Make, kaynak dosyaların ve hedef dosyanın zaman damgalarını kontrol ederek çalışır. Hedef dosya mevcut değilse veya hedefin son oluşturulmasından bu yana kaynak dosyalardan herhangi biri değiştirilmişse, Make hedefi yeniden oluşturmak için Makefile'da belirtilen komutları çalıştırır.

Make ayrıca hedefler arasındaki bağımlılıkları da destekler, böylece bir hedefin oluşturulması başka bir hedefin oluşturulmasını tetikleyebilir. Bu, karmaşık derleme işlemlerinin kolaylıkla otomatikleştirilmesine olanak tanır[33].

GNU Make özgür bir yazılımdır ve GNU Projesinin bir parçasıdır. Unix ve Unix benzeri sistemlerin yanı sıra Windows'ta da yaygın olarak kullanılmaktadır.

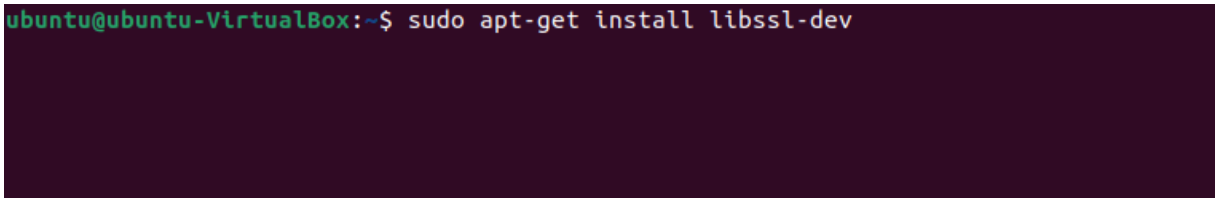
Geliştirme ortamına GNU MAKE versiyon 4.3 Şekil 3.8‘deki komut kullanılarak kurulmuştur.



Şekil 3.8: Make Kurulum Komutu

3.2.6. OPENSSL KURULUMU

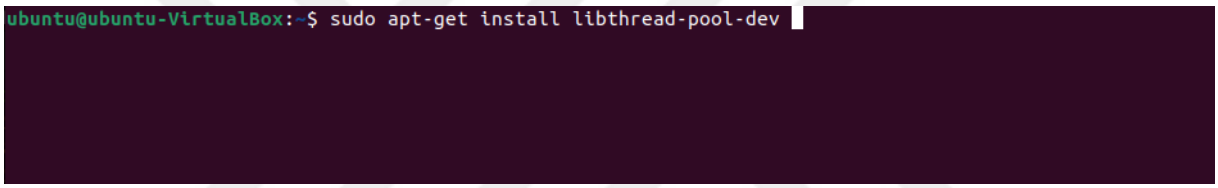
Çalışmada OPENSSL kütüphanesi, şifreleme, şifre çözme, anahtar oluşturma ve anahtar yönetimi işlevleri sağlayan fonksiyonların eriştiği kütüphane olarak kullanılmıştır. Bu kütüphaneler verilerin gizliliğini, bütünlüğünü ve gerçekliğini sağlamaktadır. Geliştirme ortamına OPENSSL kütüphanesi kurulumu Şekil 3.9’daki komut kullanılarak gerçekleştirilmiştir.



Şekil 3.9: OpenSSL Geliştirme Kütüphanesi Kurulumu

3.2.7. LIBTHREAD KURULUMU

Geliştirilen uygulamanın birden çok iş parçacıklı olacağı ön görülmüştür. Örneğin, haberleşmenin çift yönlü olması, veri işlenirken yeni veri gelme ihtimali olması proje de thread'li bir yapı kullanımı gerekli bir hale getirmiştir. Bu doğrultuda libthread kütüphanesi Şekil 3.10'daki komut kullanılarak geliştirme ortamına kurulmuştur.



Şekil 3.10: Libthread Kütüphanesi Kurulumu

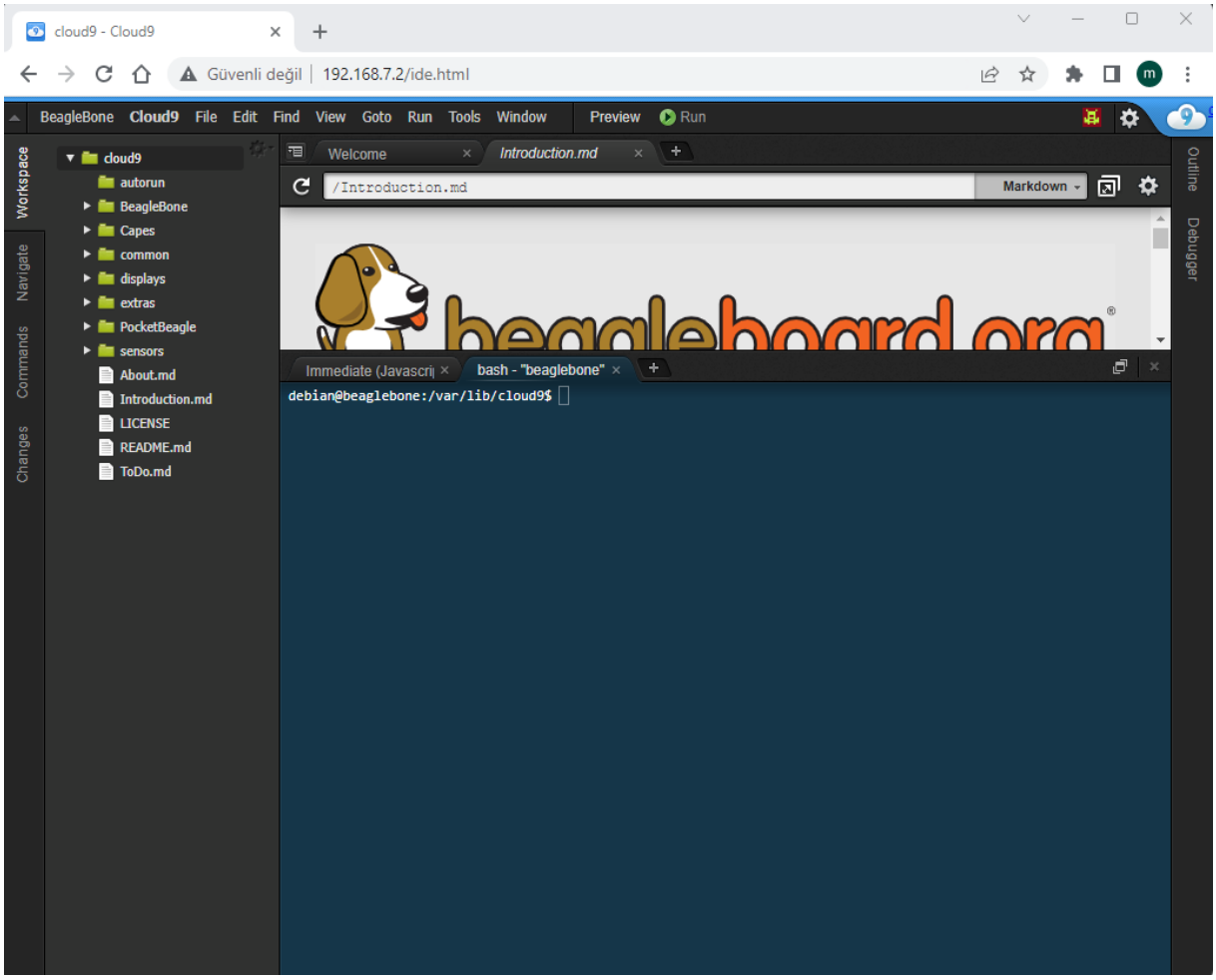
3.3. KRİPTO MODÜLÜN BELİRLENMESİ

Kripto modüller, yeterli performansa sahip, hazır ve ulaşılabilir gömülü sistemler üzerinde gerçekleştirilmesi planlanmıştır. Kripto modüller seçilirken, geliştirilecek projenin farklı alanlarda da kullanılabilmesi için jenerik olmasına, farklı araç ve protokollere uyumlu olmasına özen gösterilmiştir.

3.3.1. GÖMÜLÜ PLATFORM SEÇİMİ

Geliştirilecek olan programların üzerinde çalışabileceği, performanslı ve düşük maliyetli platformlar araştırılmıştır. Bu araştırma esnasında kullanılacak platformun Ethernet haberleşmesi, UART haberleşmesi ve üzerinde Linux tabanlı bir işletim sistemi çalıştırabilecek kapasitede olmasına dikkat edilmiştir. Bütün bunlar düşünüldüğünde Linux uyumlu, geniş bir topluluk tarafından desteklenen, tamamen açık kaynak ve yüksek performanslı bir ürün olan BeagleBone Blue geliştirme kartı tercih edilmiştir. GHU tasarımı ve implementasyonu için kullanılan BB ana mikrodenetleyici olarak TI 1GHz ARM Cortex-A8 işlemciye sahiptir. BB

geliştirme kartı DDR3 512MB RAM'e sahiptir ve üzerinde 4GB eMMC flash belleğe sahiptir[21]. Bunun yanında Şekil 3.11'deki gibi Cloud9 IDE desteği ile platform bağımsız şekilde web sunucu üzerinden erişim ve geliştirme desteği sağlamaktadır.



Şekil 3.11: BeagleBone Blue Claud Ide

3.3.2. KULLANILACAK HABERLEŞME ARAYÜZLERİNİN BELİRLENMESİ

Çalışma esnasında alıcı ve verici olarak bilgisayar kullanılması düşünüldüğü için haberleşme için bilgisayarın USB portlarının kullanılacağı ön görülmüştür. Dolayısı ile cihazlar ile BB arasındaki haberleşmenin UART olmasına karar verilmiştir. Bunun yanında uygulamanın başka fazları olursa veya başka sistemlere uyarlanmak istenirse diye CAN, SPI, I2C veya kablosuz ağ ara yüzlerinin de bulunduğu bir platform tercih edilmiştir. Ayrıca yazılım da diğer haberleşme ara yüzlerinin kullanılmasına elverişli bir şekilde geliştirilmiştir.

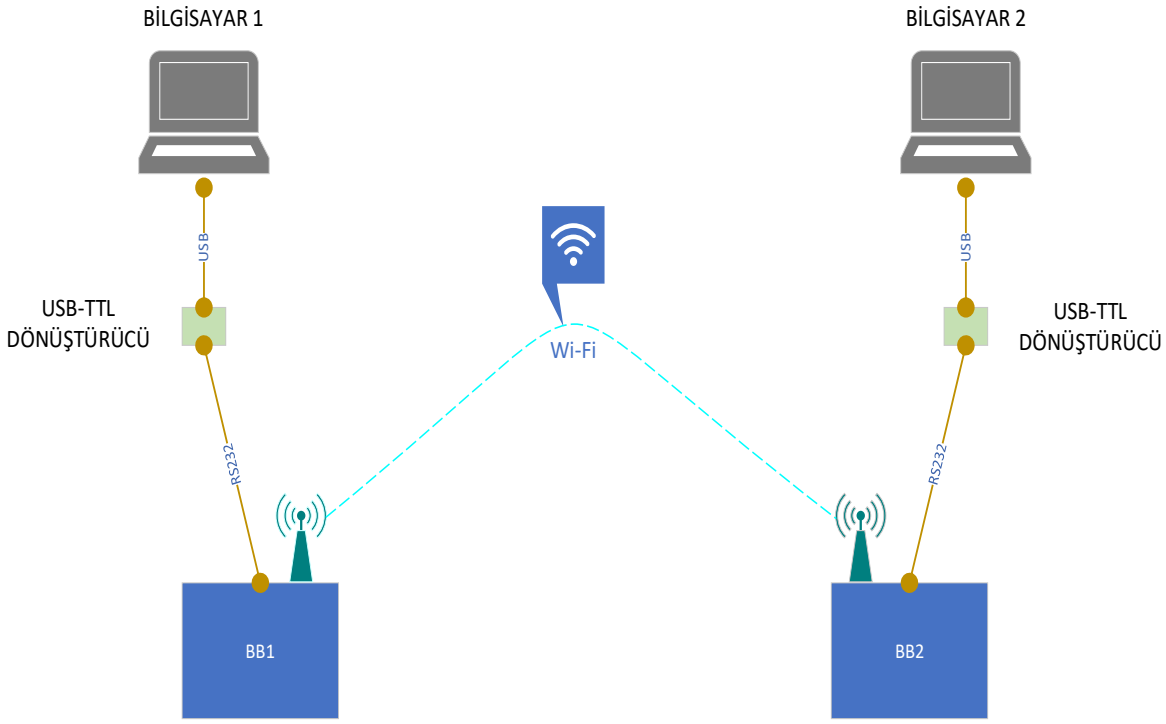
BB'ler arasındaki haberleşme ise gömülü cihazlar üzerinde bulunana kablosuz ağ modülleri üzerinde gerçekleştirilmiştir.

3.4. TEST DÜZENEĞİNİN OLUŞTURULMASI

Test düzeneği oluşturulurken kullanılan araçların, yazılımların, kütüphanelerin ve donanımların açık kaynak ve erişilebilir olmasına özen gösterilmiştir. Ayrıca sistem ilerisi için geliştirilebilir ve başka kullanımlara da entegre edilebilir olarak kurgulanmıştır. Yazılımın doğrulanması, geliştirmenin adım adım devam edebilmesi ve test edilebilmesi için test düzeneğinin iyi kurulmuş olması ve hatasız çalışması büyük önem arz etmektedir.

3.4.1. SİSTEME GENEL BAKIŞ

GHU'nun tasarlandığı, geliştirildiği ve test edildiği sistem Şekil 3.12'deki gibi oluşturulmuştur. Mevcut sistemde, birbirleri ile RS232 üzerinden haberleşen iki cihaz olarak varsayılan iki bilgisayar(Bilgisayar1 ve Bilgisayar2) ve bu haberleşmeyi güvenli hale getirecek GHU yazılımın üzerinde çalıştığı BeagleBone'lar(BB1 ve BB2) kullanılmıştır. Ayrıca bilgisayarın USB portundan alınacak verileri BB'nin UART portuna iletmek için iki tane Şekil 3.12'deki USB-TTL UART dönüştürücü kullanılmıştır.



Şekil 3.12: Şifreli Haberleşme Tasarımı

3.4.2. GÖMÜLÜ SİSTEMİN HAZIR HALE GETİRİLMESİ

BB içerisinde Debian işletim sisteminin masaüstü uygulaması bulunmayan fakat tüm isterleri karşılayabilecek ve 4GB eMMC bulunan BB'ye uyumlu bir sürümü olan versiyon 10.3 kurulmuştur. Debian kurulumu için öncelikle bir mikro SD kart içerisine Debian işetim sistemi imajı yüklenmiş ve daha sonra BB SD kart ile başlatılmıştır. Sonraki işlemde ise imaj BB üzerinde bulunan eMMC flash bellek üzerine aktarılarak BB'nin eMMC üzerinden başlatılabilmesi için gerekli düzenleme yapılmıştır. Ayrıca BB'lerin lokal ağa bağlantısı Şekil 3.13'deki komutlar kullanılarak sağlanmıştır. Geliştirme ortamında oluşturulan olan programın BB'ler üzerinde çalışabilmesi için OPENSSL ve libthread gibi gerekli kütüphaneler kurulmuştur. Böylelikle BB'ler içerisinde işletim sistemi olan ve program çalıştırabilen mini bilgisayarlar haline gelmiştir.

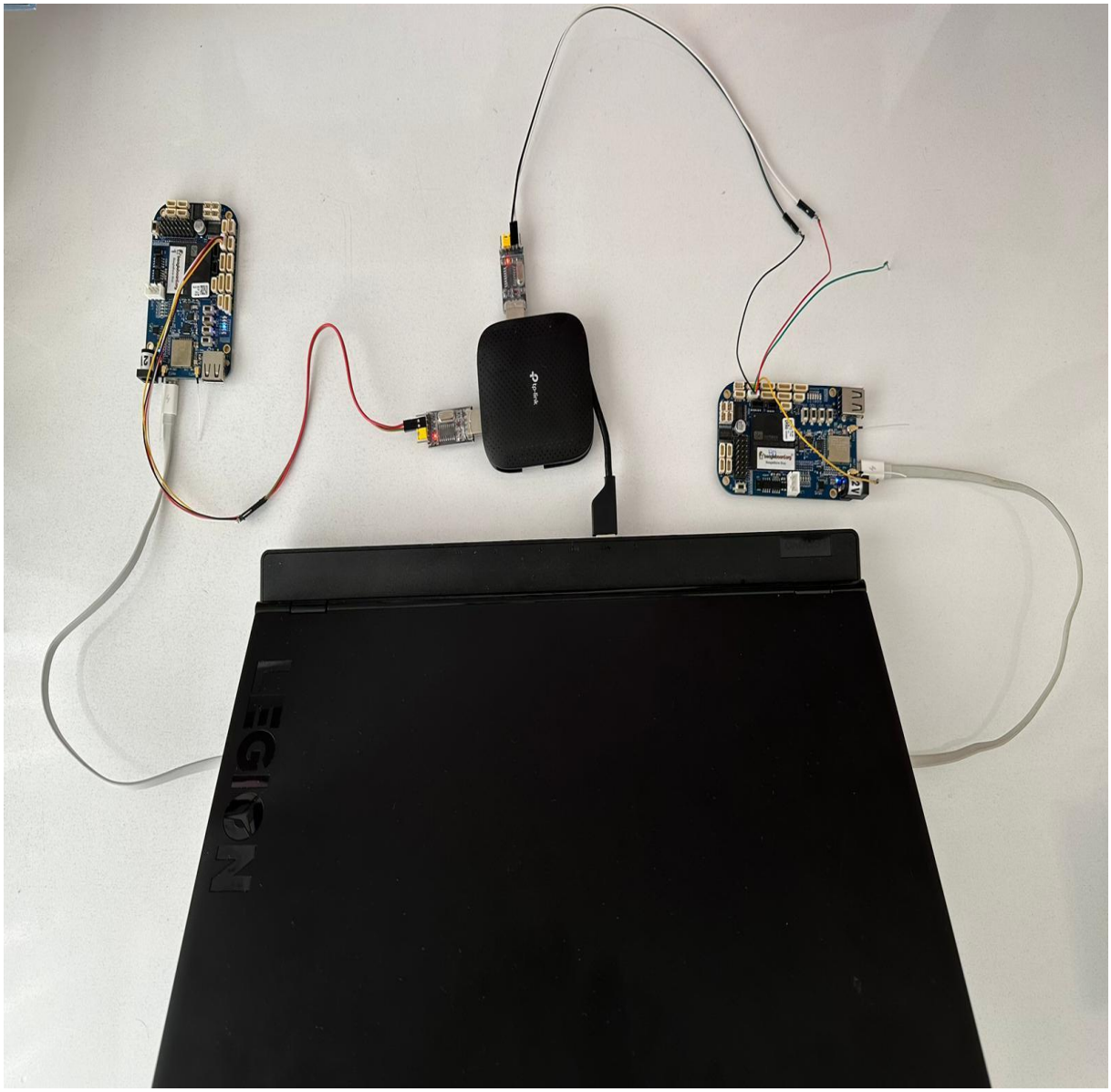
```
root@beaglebone:/var/lib/cloud9# ls
autorun  examples  LauraSand  LICENSE  README.md
root@beaglebone:/var/lib/cloud9# connmanctl
connmanctl> enable wifi
Error wifi: Already enabled
connmanctl> tether wifi disable
Error disabling wifi tethering: Already disabled
connmanctl> scan wifi
Scan completed for wifi
connmanctl> services
*AR arrow-guest          wifi_f45eab4a0141_6172726f772d6775657374_managed_none
*A  ECOMGUEST            wifi_f45eab4a0141_45434f4d4755455354_managed_psk
    Lindsay's iPhone     wifi_f45eab4a0141_4c696e6473617927732069506866f6e65_managed_psk
    beam                 wifi_f45eab4a0141_6265616d_managed_psk
    arrow-employee       wifi_f45eab4a0141_6172726f772d656d706c6f796565_managed_ieee8021x
                         wifi_f45eab4a0141_hidden_managed_psk
    Birch                wifi_f45eab4a0141_4269726368_managed_ieee8021x
    voice                wifi_f45eab4a0141_766f696365_managed_psk
connmanctl> agent on
Agent registered
connmanctl> connect wifi_f45eab4a0141_45434f4d4755455354_managed_psk
Connected wifi_f45eab4a0141_45434f4d4755455354_managed_psk
connmanctl>
```

Şekil 3.13: BeagleBone Blue Lokal Ağa Bağlanma Komutları

3.4.3. TEST ORTAMININ FİZİKSEL OLARAK OLUŞTURULMASI

Test ortamının fiziksel olarak oluşturulması için Şekil 3.14'deki gibi Universal Serial Bus (USB) Transistor-Tansistor Logic(TTL) UART dönüştürücüler şekildeki gibi kablo ve lehim yardımı ile BB'lere bağlanmıştır. RS232 bağlantısı kullanılacağı için BB'lerin beslemesindeki toprak ve verinin gönderilip alınacağı bilgisayarların toprak hattı USB-TTL dönüştürücünün takılacağı USB portundaki toprak hattı aynı olması gerektiği, aksi halde veri iletimi problemli

olacağı için BB'ler USB kabloları ile bilgisayara bağlanarak güç beslemesi sağlanmıştır. USB TTL UART dönüştürücüler ise bilgisayarın farklı portlarına bağlanmıştır. Böylelikle geliştirme ortamı ve test ortamı erişilebilir, kurulumu kolay ve düşük maliyetli olarak kurgulanmıştır. USB dönüştürücüler ayrı ayrı test edilerek doğrulanmıştır. Ayrıca yapılan konektör ve lehimli bağlantılar da tek tek multimetre ile test edilerek doğrulanmıştır.



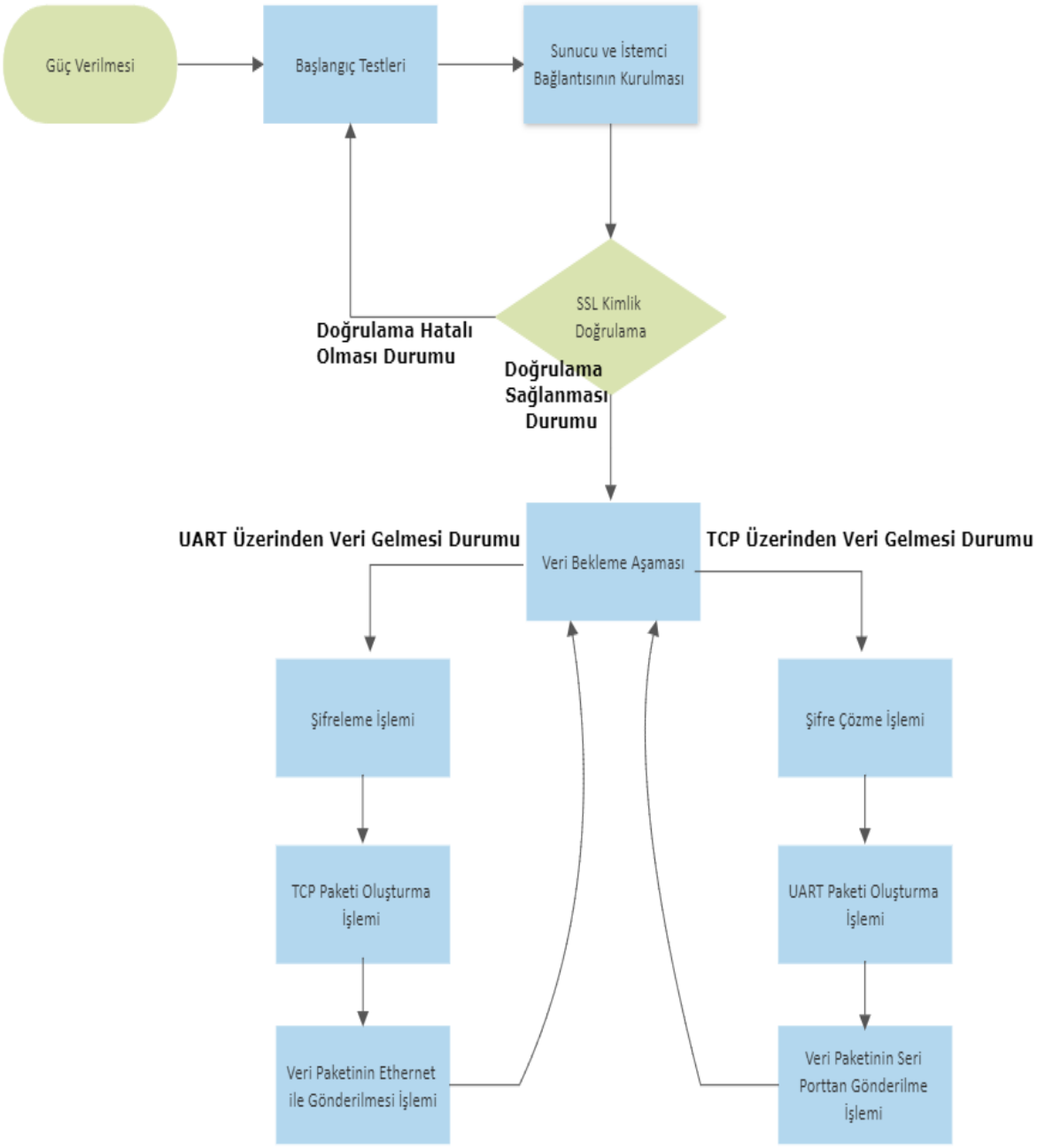
Şekil 3.14: Test Düzeneği

3.5. GÜVENLİ HABERLEŞME UYGULAMASI YAZILIM TASARIMI

GHU yazılım implementasyonu iki ayrı BB üzerinde çalışacak iki aydı uygulama olarak tasarlanmıştır. Birbirine benzer olan bu iki uygulama sunucu ve istemci olarak düşünülmüştür.

İki BB üzerinde de programlar ayağa kalkıp hazır hale geldiğinde aralarında SSL kimlik doğrulama yaparak birbirleri ile sunucu ve istemci bağlantısı kurmaktadırlar. Sunucu ve istemci bağlantısı sağlıklı olarak sağlandıktan sonra ise sistem hazır bir şekilde veri iletimini beklemektedir. BB'ye veri UART kanalından gelirse BB verici gibi davranarak gelen veriyi alır ardından şifreler, paketler ve TCP ile diğer BB'nin hedef IP ve portuna gönderir. Verinin BB'ye TCP ile geldiği durumda ise, program bunun şifreli veri olduğunu anlar ve şifreyi çözer. Ulaşılan veri paketi UART ile gönderilir. İki modül de bu şekilde çalışacağı için şifreli haberleşme çift yönlü olarak sağlanabilecektir. Program akışı Şekil 3.15'deki gibi oluşturulmuştur. Bu akışa göre sisteme enerji verildiğinde öncelikle başlangıç testleri yapılır ve burada bir sorun ile karşılaşılmazsa sunucu istemci bağlantısı yapılmak için SSL kimlik doğrulama işlemi yapılır. Kimlik doğrulama doğru bir şekilde olduğunda BB1 ve BB2 hem UART üzerinden hem de kablosuz bağlantı üzerinden paket beklemeye başlar. UART üzerinden veri paketi gelirse; gelen veri şifrelenerek TCP paketleri olarak diğer BB'ye gönderilir. Kablosuz bağlantı üzerinden veri paketi gelirse; bunun şifreli veri olduğu anlaşılarak şifre çözme işlemi uygulanır ve UART üzerinden hedef cihaza gönderilir. Böylelikle bir cihazdan diğer cihaza olan RS232 haberleşme de kripto modüller araya girerek haberleşmeyi şifreli haberleşmeye çevirip veri güvenliğini sağlamış olur.

Bu kapsamda kod tasarımı parçalara ayrıldığında karşımıza soket programlama kod bloğu, UART haberleşme kod bloğu, şifreleme işlemleri kod bloğu ve bu işlemleri eş zamanlı yapabilen thread'li yapının oluşturulması gibi iş paketleri çıkmaktadır. Bunlar sırası ile önce haberleşmelerin düzgün kurulması ve ardından haberleşmenin güvenli hale getirilmesi şeklinde geliştirilmiştir. Kablosuz haberleşmenin geliştirilmesi sırasında geliştirme ortamı için kullanılan bilgisayar ve BB'ler arasında istemci sunucu bağlantısı kurulacak şekilde uygulamalar çalıştırılmış ve test edilmiştir. UART haberleşmenin geliştirmesi sırasında geliştirme ortamı ile BB'ler minicom uygulaması yardımı ile haberleştirilerek testler yapılmıştır. Bu amaçla minicom uygulaması Şekil 3.16'daki komut yardımı ile ayarlanarak çalıştırılmıştır. Haberleşme arayüzlerinin sorunsuz bir şekilde çalıştığı doğrulandıktan sonra sıra şifreleme ve şifre çözme işlemlerinin yapılmasına gelmiş ve modüler bir şekilde ileride farklı kripto algoritmalarına da uygun entegrasyonların yapılabilirliği gözetilerek OpenSSL kütüphanesi altyapısını kullanan kripto yazılım fonksiyonlarının geliştirmesi tamamlanmıştır. Testlerde ve sonuçlarda kullanılmak üzere bazı ara sonuçlar, alınan, iletilen, şifrelenen ve şifresi çözülen veriler ve bunların boyutları konsol üzerine yazdırılarak incelenmesi sağlanmıştır.



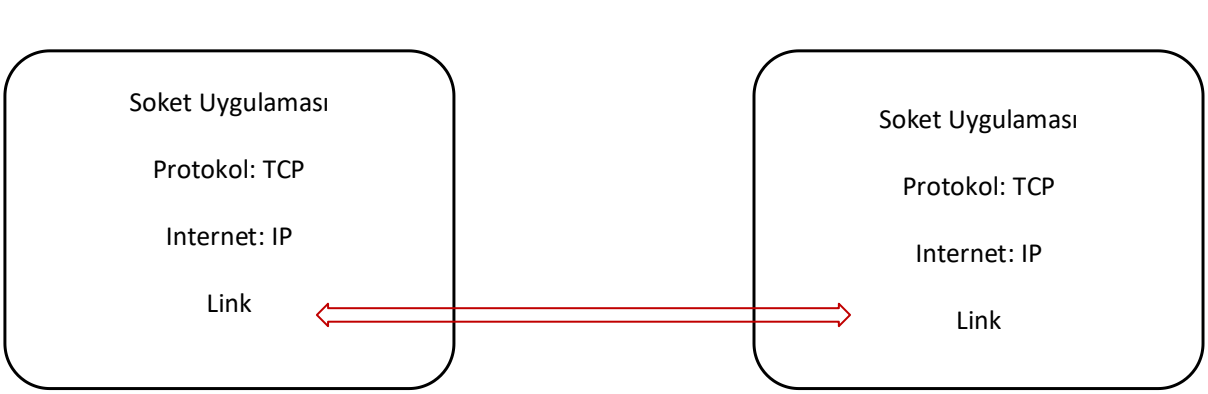
Şekil 3.15: GHU Akış Diyagramı

```
ubuntu@ubuntu-VirtualBox:~$ sudo minicom -c on -b 115200 -D /dev/ttyUSB0
```

Şekil 3.16: Minicom Çalıştırma Komutu

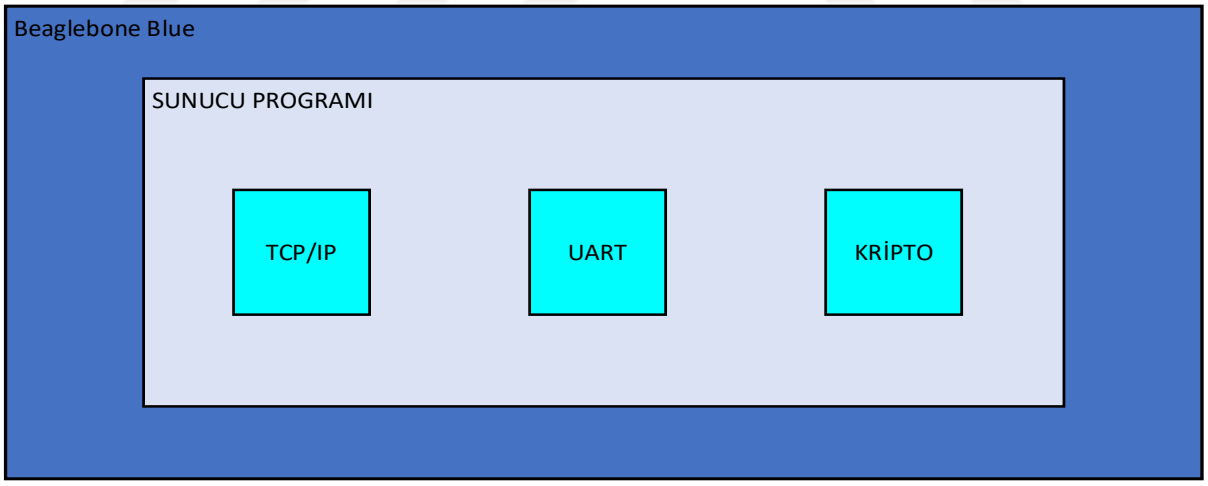
3.5.1. SOKET PROGRAMLAMA YAZILIM TASARIMI

GHU uygulamalarının çalışacağı BB'lerin birbirleri ile sunucu ve istemci olarak Şekil 3.17'deki gibi haberleşmesini sağlayacak yazılım uygulamaları geliştirilmiştir.



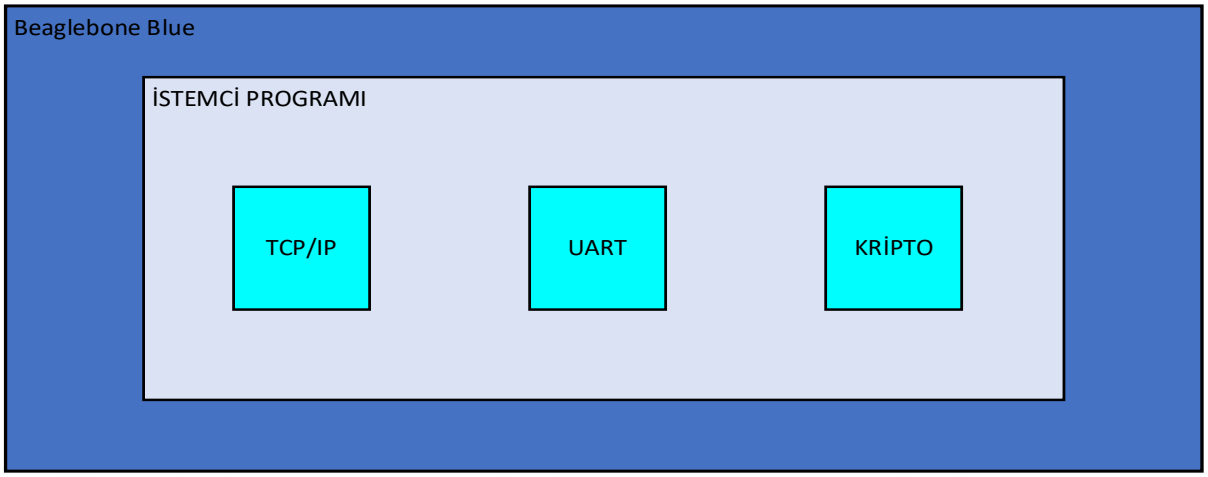
Şekil 3.17: Soket Uygulaması

Sunucu olarak çalışacak BB deki kod blok yapısı TCP/IP bloğu, UART haberleşme bloğu ve kripto işlemlerinin yapıldığı blok olmak üzere Şekil 3.18'deki gibi üç temel bloktan oluşmaktadır.



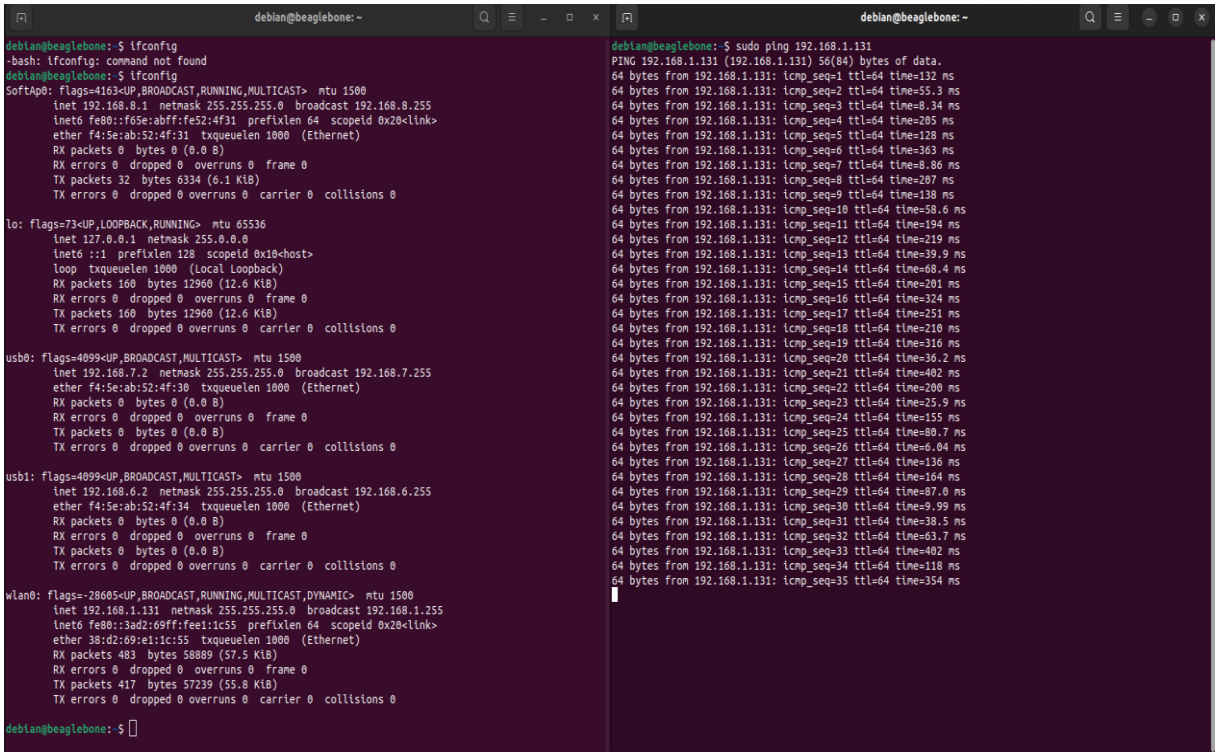
Şekil 3.18: Sunucu Programı Yazılım Blok Gösterimi

İstemci olarak çalışacak uygulamanın kod blok yapısı ise TCP/IP haberleşme bloğu, UART haberleşme bloğu ve kripto işlemlerin yapıldığı blok olmak üzere Şekil 3.19'daki gibi üç temel bloktan oluşmaktadır.



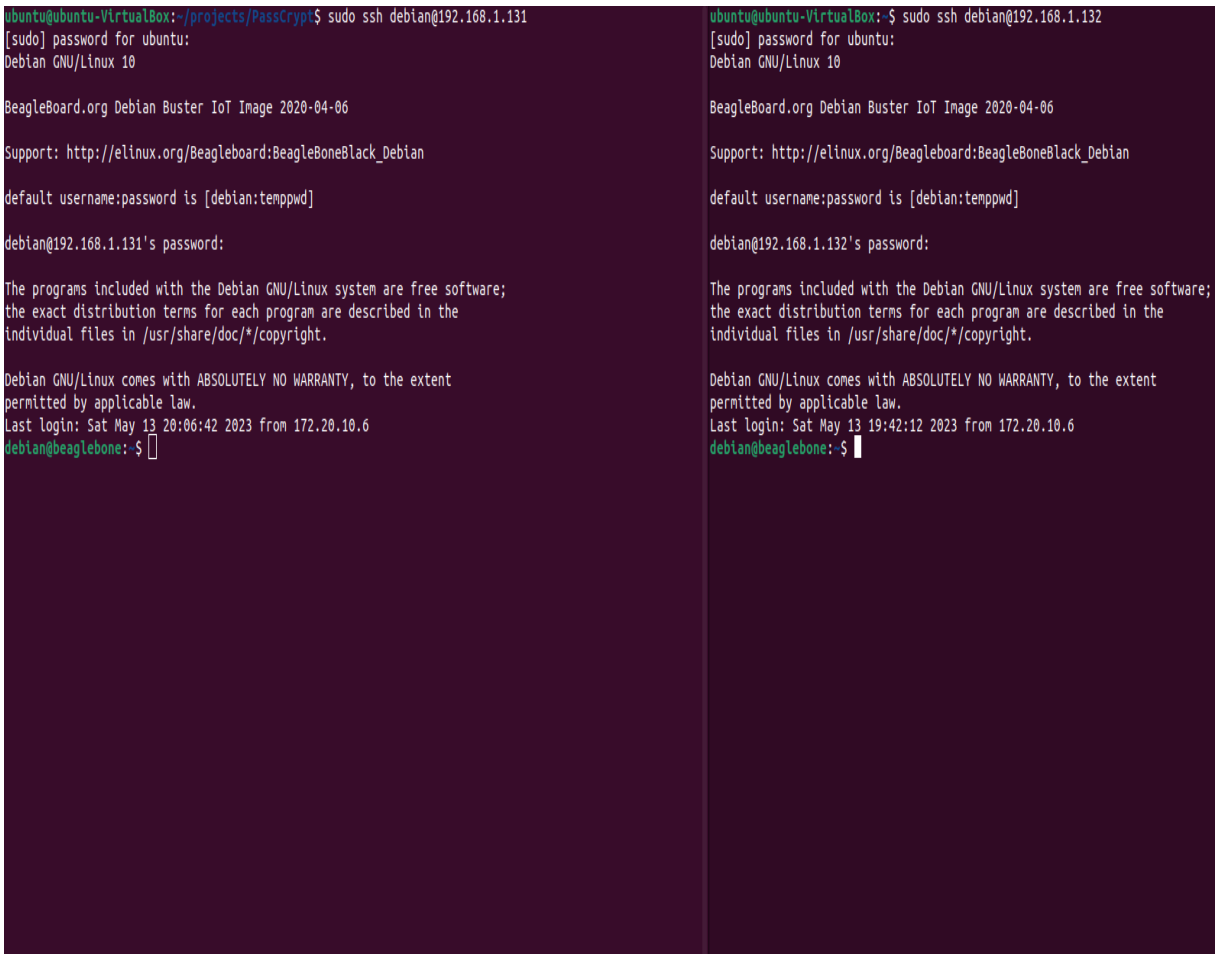
Şekil 3.19: İstemci Programı Yazılım Blok Gösterimi

Öncelikle, geliştirme ortamının olduğu bilgisayar, BB1 ve BB2 aynı yerel ağa bağlanır. Bağlantının sorunsuz olarak yapılabildiğini test etmek için Şekil1deki gibi önce IP'ler sorgulanmış ve BB1 den BB2'ye ping atılarak sonuçlar incelenmiştir.



Şekil 3.20: Kripto Cihazların Lokal Ağ Bağlantı Testi

BB1 ve BB2 arasındaki bağlantı doğrulandıktan sonra, geliştirme ortamında iki ayrı terminal açılarak ssh komutları ile BB1 ve BB2 ye IP adresleri girilerek Şekil 3.21'deki gibi SSH bağlantısı yapılmıştır. Uzaktan yapılan bu bağlantı sayesinde, gerekli root şifreleri girildikten sonra başka bir bilgisayar üzerinden BB'ler kullanılabilmektedir.



Şekil 3.21: SSH Bağlantısı Komutu

Uzaktan yapılan ssh bağlantıları ile bilgisayar üzerinden BB1 ve BB2 ye erişilerek, ilgili gömülü sistemler yönetilmiştir. Geliştirme ortamında tamamlanarak GitHub'a yüklenmiş yazılım proje dosyaları BB1 ve BB2 ye indirilmiştir. İstemci uygulaması içerisinde buluanan hedef IP kısmına, sunucunun IP adresi Şekil 3.22'deki gibi eklenmiştir. Bunun ardından indirilen kod dosyaları BB'ler içerisinde derlenerek yürütülebilir dosyalar oluşturulmuştur. Sunucu kodları BB platformu üzerinde Şekil 3.23'deki gibi MAKE komutu ile derlenmiştir. İstemci kodları ise Şekil 3.24'deki gibi CMAKE komutları ile derlenmiştir.

```
#include "client.h"

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <sys/stat.h>
#include <sys/sendfile.h>
#include <fcntl.h>

#include <iostream>
#include "EVP.h"

#define BUFFER_SIZE        4096

#define IP_ADDR             "192.168.1.132"
#define PORT_NUM            "7777"
```

Şekil 3.22: İstemci Hedef IP Tanımlama

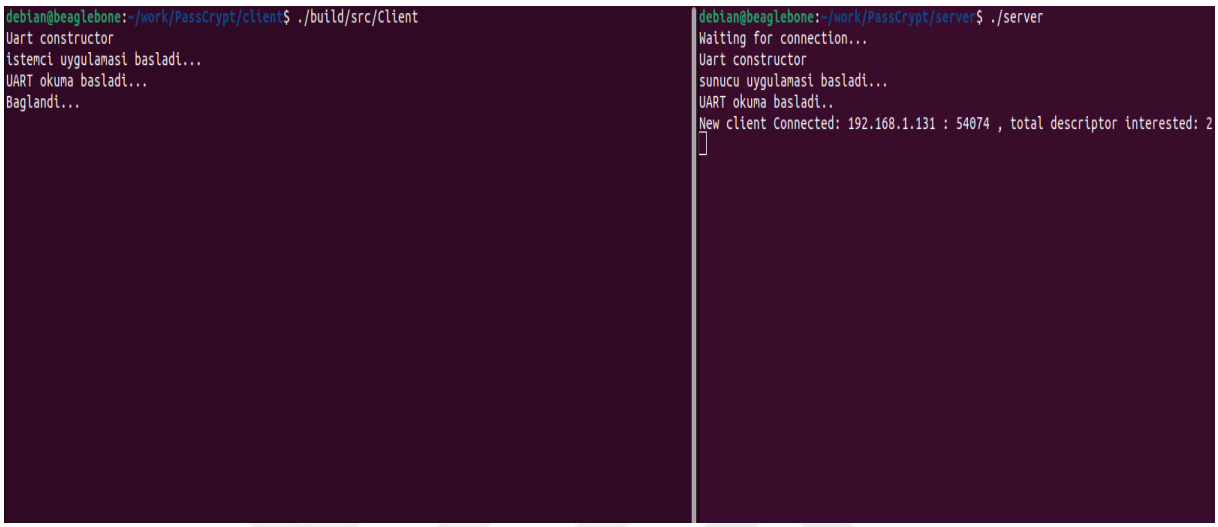
```
ubuntu@ubuntu-VirtualBox:~/projects/PassCrypt/server$ make
g++ -c EVP.cpp -pthread -std=c++17 -Wall -lcrypto -lssl
g++ -c main.cpp -pthread -std=c++17 -Wall -lcrypto -lssl
g++ -c server.cpp -pthread -std=c++17 -Wall -lcrypto -lssl
g++ -c uart.cpp -pthread -std=c++17 -Wall -lcrypto -lssl
g++ -o server EVP.o main.o server.o uart.o -pthread -std=c++17 -Wall -lcrypto -lssl
ubuntu@ubuntu-VirtualBox:~/projects/PassCrypt/server$
```

Şekil 3.23: Make Derleme Komutu

```
debian@beaglebone:~/work/PassCrypt/client$ cmake -S . -B build
-- The C compiler identification is GNU 8.3.0
-- The CXX compiler identification is GNU 8.3.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /home/debian/work/PassCrypt/client/build
debian@beaglebone:~/work/PassCrypt/client$ cmake --build build/
Scanning dependencies of target Client
[ 20%] Building CXX object src/CMakeFiles/Client.dir/EVP.cpp.o
[ 40%] Building CXX object src/CMakeFiles/Client.dir/client.cpp.o
[ 60%] Building CXX object src/CMakeFiles/Client.dir/main.cpp.o
[ 80%] Building CXX object src/CMakeFiles/Client.dir/uart.cpp.o
[100%] Linking CXX executable Client
[100%] Built target Client
debian@beaglebone:~/work/PassCrypt/client$
```

Şekil 3.24: Cmake Derleme Komutu

Öncelikli olarak sunucu ve istemci arasındaki haberleşmenin çift yönlü olarak kurulması gerçekleştirilmiş ve test edilmiş olup kripto işlemler sonraki adımlarda yapılmıştır. BB1 ve BB2 üzerinde çalıştırılan sunucu ve istemci uygulamaları Şekil 3.25'deki gibi çalıştırılarak bağlantı durumları test edilmiş ve konsol üzerine sonuçlar yazdırılmıştır.



Şekil 3.25: Sunucu ve İstemci Bağlantısı Kurulumu

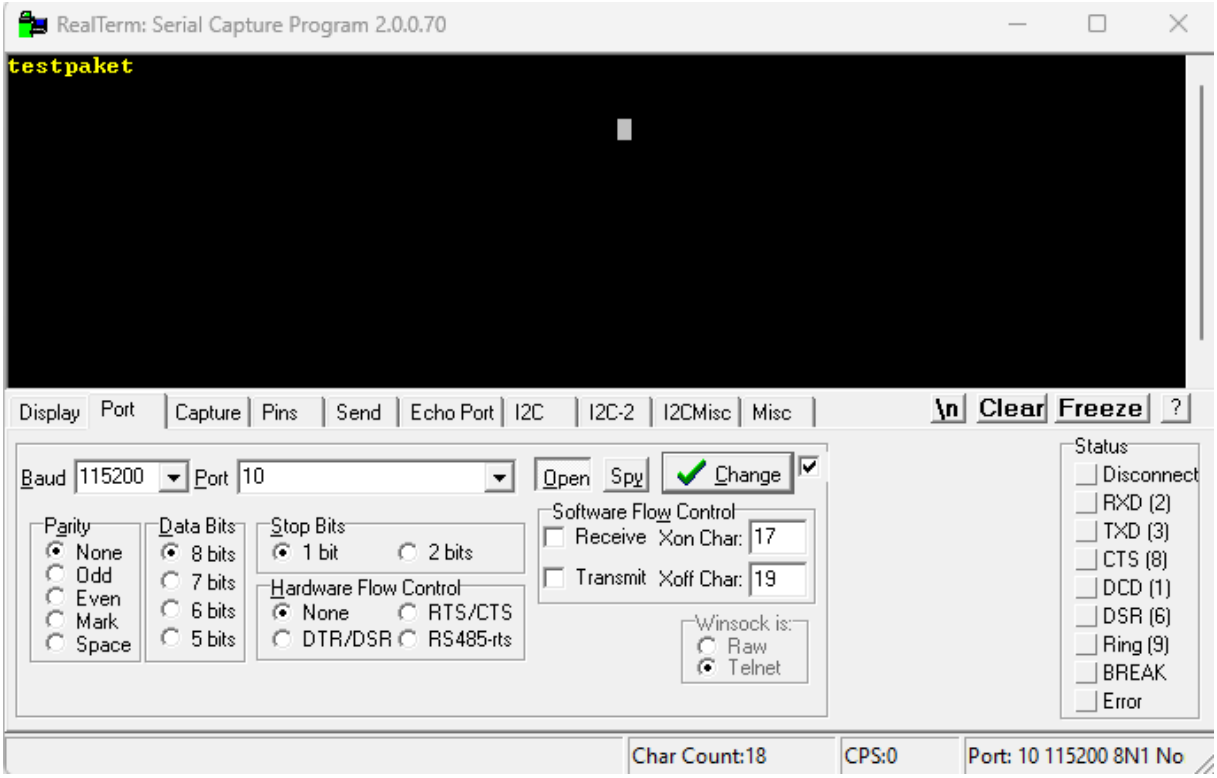
3.5.2. SERİ PORT TEST UYGULAMASININ GELİŞTİRİLMESİ

Sunucu ve istemci uygulamalarının üzerinde çalışacağı BB1 ve BB2 kullanımda kripto köprü işlevi göreceği için mesajı gönderen cihaz ve mesajın gönderildiği cihaz ile de bağlantılı olması gerekmektedir. Mesajın alınacağı ve mesajın iletileceği cihazlar ile olacak bağlantı UART protokolü olarak belirlendiği için hem BB1 hem de BB2 UART üzerinden paket gönderip alma kabiliyetine sahip olmaları gerekmektedir. Dolayısı ile sunucu ve istemci uygulamalarına aynı zamanda çift yönlü UART haberleşme kütüphaneleri eklenmiştir. İki uçtan da eş zamanlı paket gelme ihtimali de öngörülerek yazılım geliştirme de thread'li yap ı kullanılmıştır. Haberleşme de kullanılacak baudrate 115200 bps olarak uygulanmış ve UART için ilk iki bayt paketin uzunluğunu belirtecek şekilde bir paket yapısı oluşturulmuştur. Yazılım içerisinde bu paket yapısına göre UART üzerinden gelen veriler paketlenerek bir ring buffer içerisine yazılmaktadır. Daha sonra bu ring buffer şifreleme bloğunda kullanılmak üzere oluşturulmuştur. UART geliştirmesi testlerini yapabilmek için linux Linux işletim sisteminde minicom uygulaması kullanılmıştır. Minicom üzerinden yapılacak testler için bağlantılar Şekil 3.26'deki komutla listelenmiş ve minicom bağlantısı yapılarak geliştirme ortamı ve BB'ler arasındaki UART haberleşme testleri yapılmıştır.

```
ubuntu@ubuntu-VirtualBox:~$ ll /dev/tty*
crw-rw-rw- 1 root   tty      5,  0 May 20 00:45 /dev/tty
crw--w---- 1 root   tty      4,  0 May 20 00:45 /dev/tty0
crw--w---- 1 root   tty      4,  1 May 20 00:45 /dev/tty1
crw--w---- 1 root   tty      4, 10 May 20 00:45 /dev/tty10
crw--w---- 1 root   tty      4, 11 May 20 00:45 /dev/tty11
crw--w---- 1 root   tty      4, 12 May 20 00:45 /dev/tty12
crw--w---- 1 root   tty      4, 13 May 20 00:45 /dev/tty13
crw--w---- 1 root   tty      4, 14 May 20 00:45 /dev/tty14
crw--w---- 1 root   tty      4, 15 May 20 00:45 /dev/tty15
crw--w---- 1 root   tty      4, 16 May 20 00:45 /dev/tty16
crw--w---- 1 root   tty      4, 17 May 20 00:45 /dev/tty17
crw--w---- 1 root   tty      4, 18 May 20 00:45 /dev/tty18
crw--w---- 1 root   tty      4, 19 May 20 00:45 /dev/tty19
crw--w---- 1 ubuntu tty      4,  2 May 20 00:49 /dev/tty2
crw--w---- 1 root   tty      4, 20 May 20 00:45 /dev/tty20
crw--w---- 1 root   tty      4, 21 May 20 00:45 /dev/tty21
crw--w---- 1 root   tty      4, 22 May 20 00:45 /dev/tty22
crw--w---- 1 root   tty      4, 23 May 20 00:45 /dev/tty23
crw--w---- 1 root   tty      4, 24 May 20 00:45 /dev/tty24
crw--w---- 1 root   tty      4, 25 May 20 00:45 /dev/tty25
crw--w---- 1 root   tty      4, 26 May 20 00:45 /dev/tty26
crw--w---- 1 root   tty      4, 27 May 20 00:45 /dev/tty27
crw--w---- 1 root   tty      4, 28 May 20 00:45 /dev/tty28
crw--w---- 1 root   tty      4, 29 May 20 00:45 /dev/tty29
crw--w---- 1 root   tty      4,  3 May 20 00:45 /dev/tty3
crw--w---- 1 root   tty      4, 30 May 20 00:45 /dev/tty30
crw--w---- 1 root   tty      4, 31 May 20 00:45 /dev/tty31
crw--w---- 1 root   tty      4, 32 May 20 00:45 /dev/tty32
crw--w---- 1 root   tty      4, 33 May 20 00:45 /dev/tty33
crw--w---- 1 root   tty      4, 34 May 20 00:45 /dev/tty34
crw--w---- 1 root   tty      4, 35 May 20 00:45 /dev/tty35
crw--w---- 1 root   tty      4, 36 May 20 00:45 /dev/tty36
crw--w---- 1 root   tty      4, 37 May 20 00:45 /dev/tty37
crw--w---- 1 root   tty      4, 38 May 20 00:45 /dev/tty38
crw--w---- 1 root   tty      4, 39 May 20 00:45 /dev/tty39
crw--w---- 1 root   tty      4,  4 May 20 00:45 /dev/tty4
crw--w---- 1 root   tty      4, 40 May 20 00:45 /dev/tty40
crw--w---- 1 root   tty      4, 41 May 20 00:45 /dev/tty41
crw--w---- 1 root   tty      4, 42 May 20 00:45 /dev/tty42
crw--w---- 1 root   tty      4, 43 May 20 00:45 /dev/tty43
crw--w---- 1 root   tty      4, 44 May 20 00:45 /dev/tty44
crw--w---- 1 root   tty      4, 45 May 20 00:45 /dev/tty45
crw--w---- 1 root   tty      4, 46 May 20 00:45 /dev/tty46
crw--w---- 1 root   tty      4, 47 May 20 00:45 /dev/tty47
crw--w---- 1 root   tty      4, 48 May 20 00:45 /dev/tty48
crw--w---- 1 root   tty      4, 49 May 20 00:45 /dev/tty49
crw--w---- 1 root   tty      4,  5 May 20 00:45 /dev/tty5
crw--w---- 1 root   tty      4, 50 May 20 00:45 /dev/tty50
```

Şekil 3.26: Bağlı Cihaz Listeleme Komutu

Windows işletim sisteminde ise ücretsiz bir seri port haberleşme aracı olan RealTerm uygulaması kullanılmıştır. Port ve baudrate ayarları Şekil 3.27'deki gibi yapılarak "testpaket" veri paketi UART ile gönderilmiştir.



Şekil 3.27: RealTerm Yapılandırması

UART ile gönderilen veri paketinin BB' ye iletildiği Şekil 3.28'deki gibi konsol üzerine yazdırılarak sonuç gösterilmiştir. Bu işlem her iki BB için de gerçekleştirilerek UART haberleşme tarafı test edilmiştir. Ayrıca farklı boyutlarda veriler gönderilerek, sistemin yük testleri de yapılmıştır.

```
debian@beaglebone:~/work/PassCrypt/client$ ./build/src/Client
Uart constructor
UART okuma basladi...
Baglandi...
result = 1
veri alindi = testpaket
```

Şekil 3.28: UART Haberleşme Testi

3.5.3. KRİPTO MODÜLLERİN YAZILIM TASARIMI

Kripto modüllerin tasarlanması süreçlerinde, oldukça yaygın kullanıma sahip ve geliştirmesi geniş topluluklarca desteklenen bir kütüphane olan OPENSSL kütüphanesinden yararlanılmıştır. Yazılan yazılım fonksiyonları OPENSSL kütüphanesi içerisinde bulunan şifreleme ve şifre çözme algoritmalarını kullanacak şekilde oluşturulmuştur.

3.5.3.1. AES ŞİFRELEME ALGORİTMASI

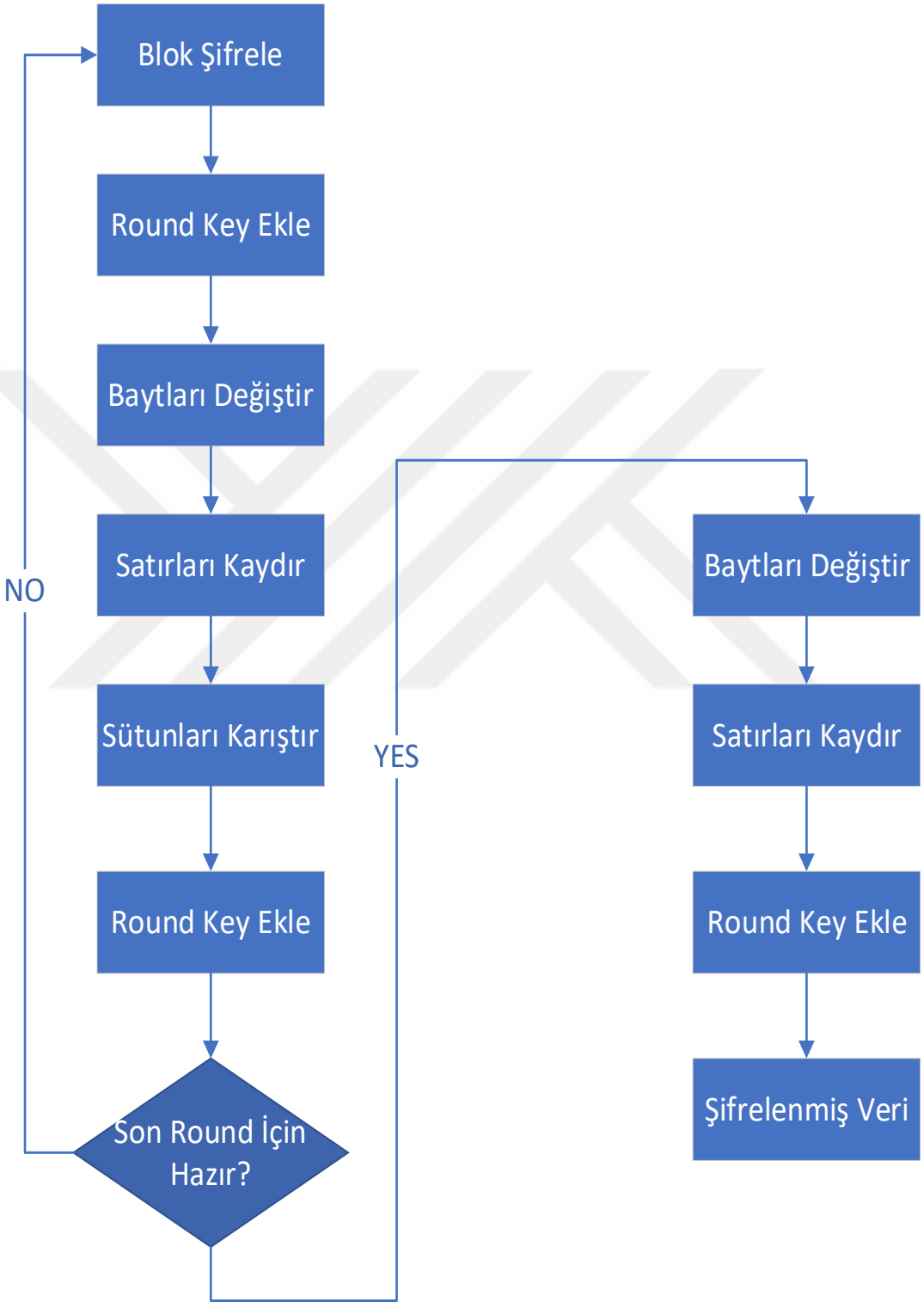
AES 256 (256 bit anahtar boyutuna sahip Gelişmiş Şifreleme Standardı) veri güvenliği ve gizliliği sağlaması bakımından tercih edilmiştir. Önemli bilgileri yetkisiz erişim veya manipülasyondan korumak için kullanılır. Birçok yazılım kütüphanesi, bilgisayar dili, işletim sistemi ve ağ protokolü AES 256-bit şifrelemeyi destekler. Uyumluluğu ve birlikte çalışabilirliği nedeniyle çeşitli platformlar ve cihazlar arasında iletişimi korumak için yaygın olarak tercih edilen bir seçenektir. AES 128-bit ve AES 192-bit ile kıyaslanırsa, AES 256-bit şifreleme daha yüksek bir güvenlik seviyesi sağlamaktadır. 256 bit kripto anahtarının kullanıldığı şifrelemenin çözülmesi doğal olarak çok daha zordur. AES 256-bit şifreleme tekniği böylelikle hassas veri güvenliğini artırır ve saldırılara karşı duyarlılığını azaltır[34].

Devlet kurumları, finans kuruluşları ve en üst düzeyde güvenlik talep eden diğer işletmelerde bu teknik kullanılmaktadır. Ayrıca ABD hükümeti hassas verileri korumak için standart olarak AES 256-bit şifrelemeyi kullanmaktadır. AES 256-bit şifreleme bilinen tüm gerçek dünya saldırılarına karşı güvenli olarak değerlendirilmesinin yanında 256 bitlik anahtar uzunluğu sayesinde çok sayıda potansiyel kombinasyon sunarak kaba siber saldırılar karşısında şifrenin çözülmesini hesaplama açısından oldukça zor hale getirmektedir.

AES'in üzerine inşa edildiği simetrik anahtar şifreleme tekniği, hem şifreleme hem de şifre çözme işlemleri için aynı kripto anahtarı kullanmaktadır. AES 256-bit şifreleme, farklı kriptografik saldırılara karşı dirençli hale getirmek için veri değiştirme, permütasyon ve karıştırma gibi bir dizi matematiksel işlem kullanır[35].

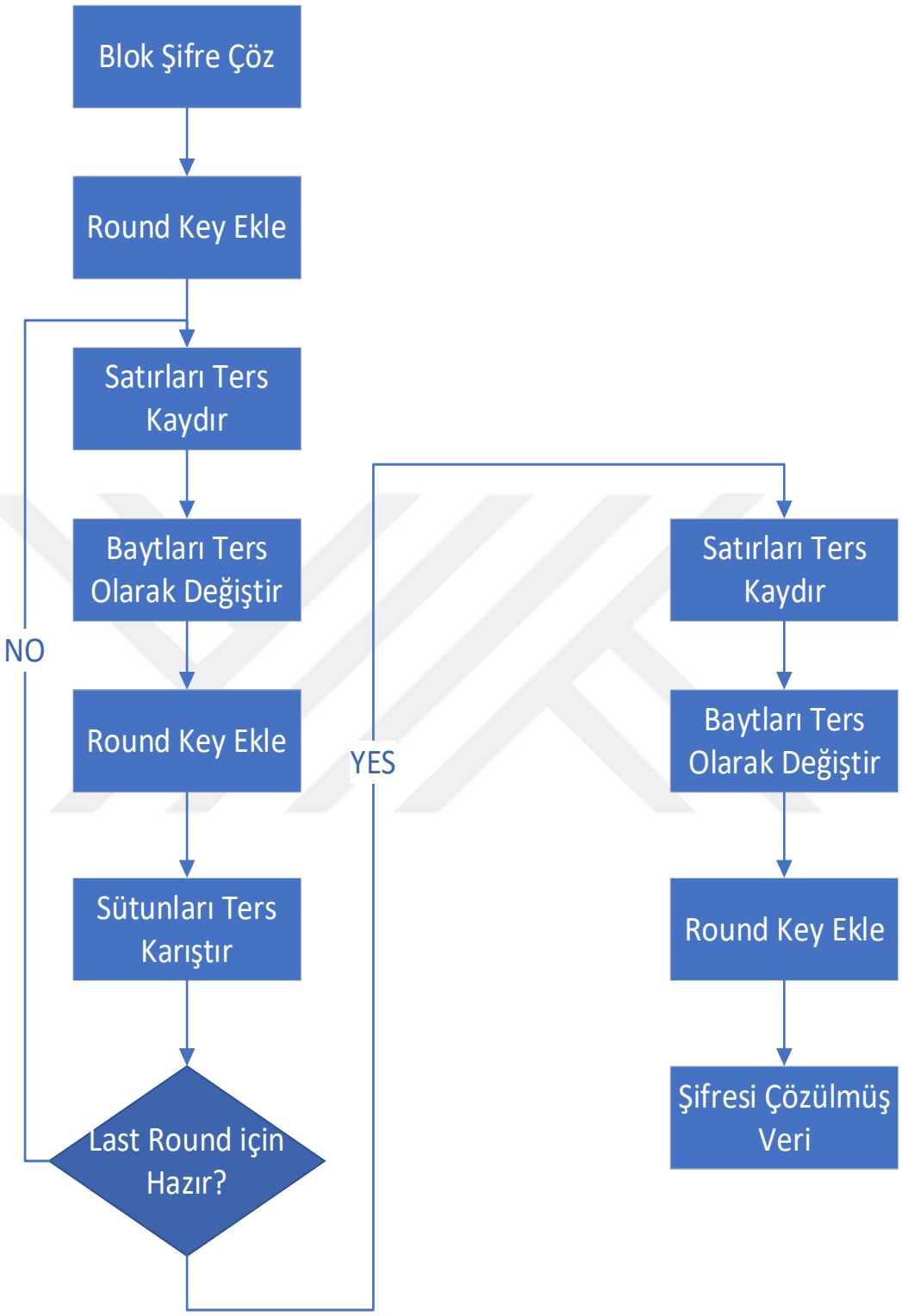
AES 256-bit şifreleme oldukça güçlü ve güvenilir olmasına rağmen, haberleşmenin güvenliğini tek başına sağlayamaz. Şifreli haberleşmenin toplam güvenliğinin sağlanmasında, güvenli anahtar yönetimi, etkin uygulama ve güvenli anahtar iletimi gibi diğer unsurların da etkisi bulunmaktadır. AES şifreleme işleminin akış diyagramı Şekil 3.29'deki gibi gösterilmiştir. Şifrelemeden önce düz metnin ilk bloğunu başlatmak için kullanılan rastgele bir değer olan bir

Initial Vector (IV) Chiper Block Chaining (CBC) modu için gereklidir. Başka bir koruma derecesi sağlamak için IV, şifreli metnin ilk bloğu ile XORlanır.



Şekil 3.29: AES Şifreleme Akış Diyagramı

AES algoritmaları kullanarak şifre çözülmesindeki akış Şekil 3.30'de gösterilmiştir.



Şekil 3.30: AES Şifre Çözme Akış Diyagramı

CPU'lar, GPU'lar ve kriptografik donanımlar dahil olmak üzere modern bilgisayar donanımları, AES 256-bit şifrelemeyi etkin bir şekilde uygular. Bu, performansı belirgin bir şekilde etkilemeden hızlı ve etkili şifreleme ve şifre çözmeyi sağlar.

3.5.3.2. ANAHTAR VE BAŞLANGIÇ VEKTÖRÜ OLUŞTURMA

AES kripto algoritmasında kullanılmak üzere başlangıç vektörü ve 256 bit uzunluğunda anahtar oluşturulmuştur. Anahtar oluşturma işlemi için OPENSSL kütüphanesinde bulunan Şekil 3.31'deki komut kullanılmıştır.

```
ubuntu@ubuntu-VirtualBox:~$ openssl enc -aes-256-cbc -k secret -P -md sha1
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
salt=1055189A2BB45E6C
key=5F896F09A9ED2C495F5843EC2C823FA4366E14FC21B7D5F1AB56FDC104B09F57
iv =53750CB86BDAB4E3768534A50FF2ACDD
ubuntu@ubuntu-VirtualBox:~$
```

Şekil 3.31: OpenSSL Anahtar Oluşturma Komutu

3.5.3.3. OPENSSL KÜTÜPHANESİ ENTEGRASYONU

OPENSS kütüphanesini kullanabilmek için gerekli olan header dosyalar Şekil 3.32'deki gibi eklenilmiştir.

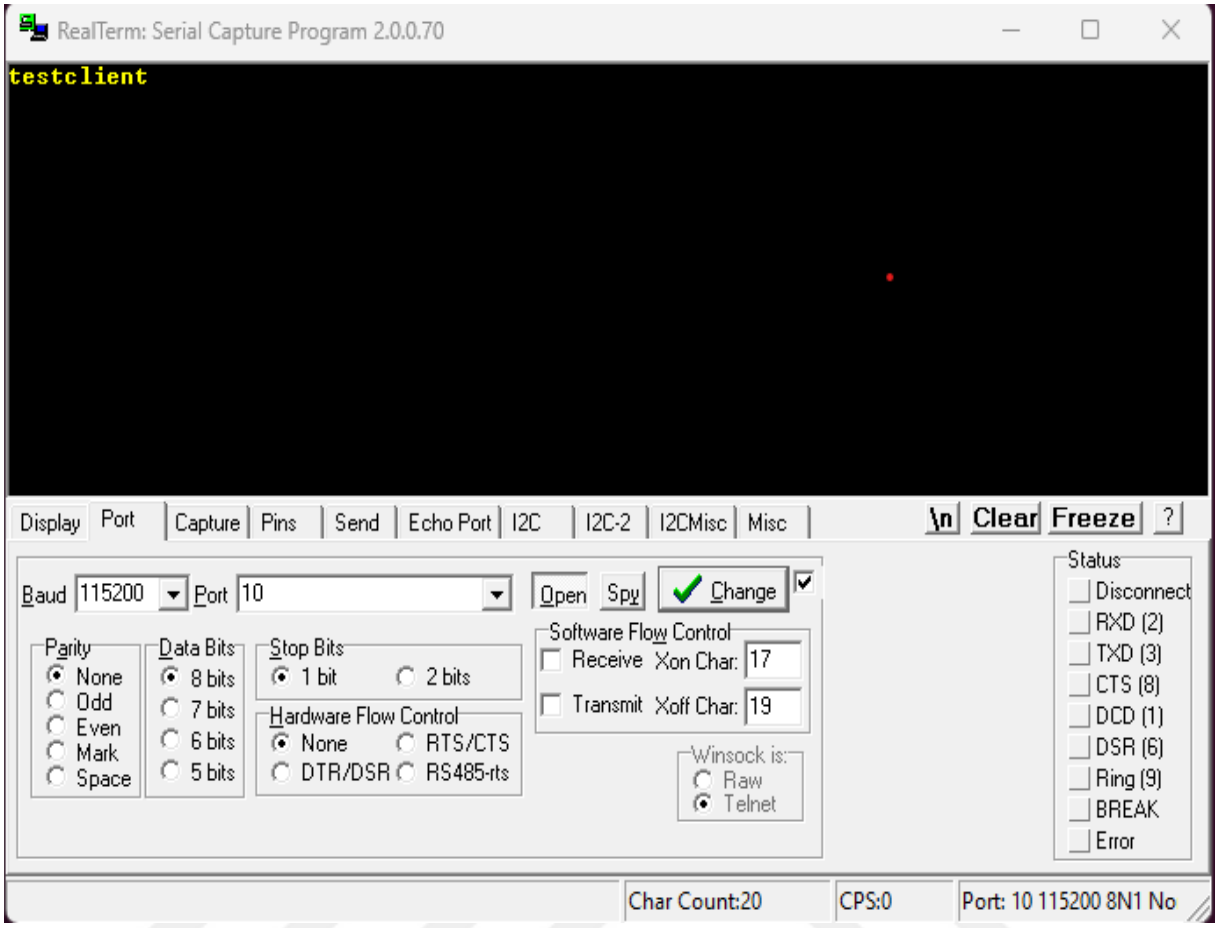
```
1  #include <openssl/conf.h>
2  #include <openssl/evp.h>
3  #include <openssl/err.h>
```

Şekil 3.32: OpenSSL Kütüphaneleri

Daha sonra OPENSSL kütüphanesi içerisinde bulunan AES CBC şifreleme ve şifre çözme algoritmalarını kullanan yazılım fonksiyonları yazılarak blok şifreleme ve şifre çözme işlemleri gerçekleştirilmiştir. Ayrıca simetrik kripto algoritmaları kullanılacağı için üretilen kripto anahtar ve başlangıç vektörü hem istemci yazılımının hem de sunucu yazılımının içerisine eklenilmiştir.

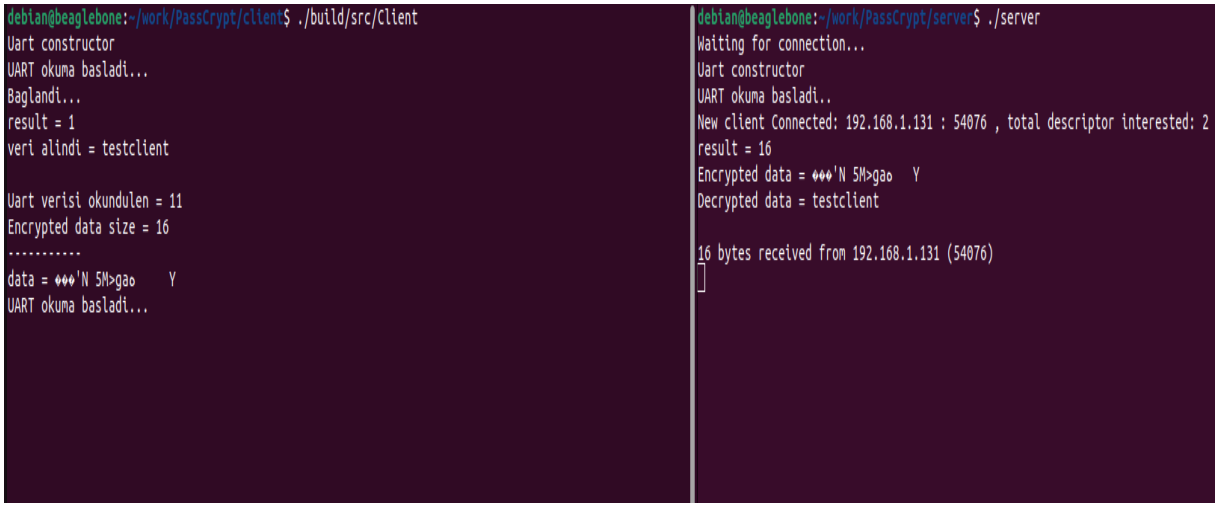
3.5.3.4. ŞİFRELİ HABERLEŞMENİN TEST EDİLMESİ

Oluşturulan kullanım senaryosuna göre BB1 veya BB2 ye UART üzerinden bir veri paketi gönderildiğinde, gelen veri paketi şifrelenerek diğer BB'ye TCP/IP ile gönderilmesi gerekmektedir. Kablosuz bağlantı üzerinden veri paketi alan BB'nin ise, gelen paketin şifreli olduğunu kabul ederek şifrelenmiş paketin şifresini çözmesi beklenmektedir. Bu kapsamda, RealTerm aracında Şekil 3.33'deki gibi port ve baudrate ayarlamaları yapılarak bir UART haberleşme kanalı üzerinden "testclient" veri paketi gönderilmiştir. RealTerm uygulamasının status kısmında TX ve RX kutucukları takip edilerek ve USB-TTL dönüştürücü üzerinde bulunan led ışıklar takip edilerek veri gönderildiği doğrulanmıştır.



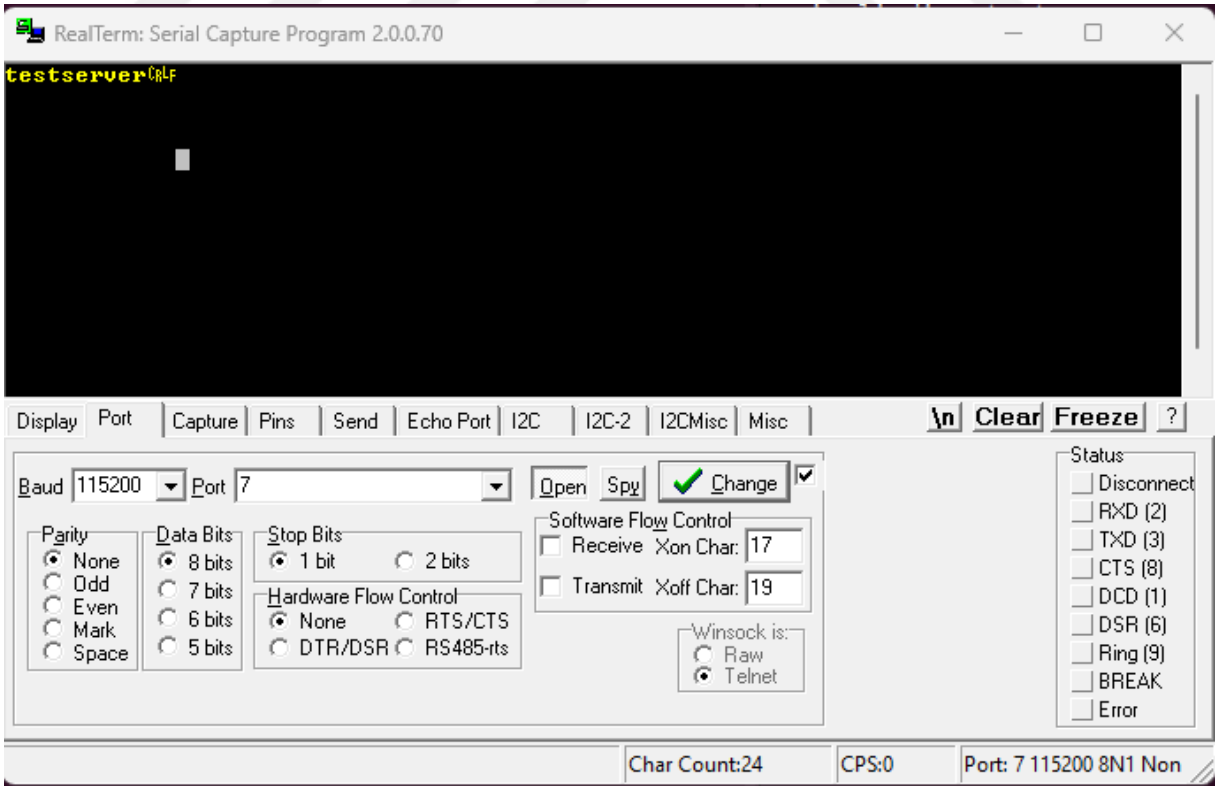
Şekil 3.33: İstemciye Gönderilen UART Test Paketi

Gönderilen veri paketinin hangi veriyi kapsadığı ve BB1'e doğru bir şekilde iletildiğini anlayabilmek için BB1'in kontrol edildiği SSH bağlantısının olduğu konsol üzerine alınan verinin ne olduğu Şekil 3.34'deki gibi yazdırılmıştır. Bu çıktıya göre "testclient" test veri paketi istemci uygulamanın çalıştığı BB tarafından doğru bir şekilde alınmıştır. Ayrıca, ilgili verinin AES 256-bit şifreleme algoritması kullanarak şifrenmiş halinin ne olduğu, şifrelenmeden önceki ve şifrelendikten sonraki veri boyutu aynı konsol üzerine yazdırılmıştır. Sunucu uygulamasının çalıştığı BB2'ye yapılan SSH bağlantısının bulunduğu konsol üzerine ise BB2 ye iletilen şifrelenmiş veri, veri şifresinin çözülmüş hali, şifrenin çözülmeden önceki ve çözüldükten sonraki halinin veri boyutu Şekil 3.34'deki gibi yazılmıştır. Bu konsoldaki çıktılar karşılaştırıldığında, "tesclient" test verisi BB1 tarafından RS232 haberleşme ile okunduktan sonra şifrelenerek BB2'ye gönderildiği ve BB2 içerisinde şifrenin doğru bir şekilde çözülerek verinin ilk haline ulaşıldığı ve paket kaybının olmadığı görülmüştür. Bu yöntemle güvenli haberleşmenin tek yönü doğrulanmıştır.



Şekil 3.34: İstemci Uygulamasının Okuduğu UART Test Verisi

Kurulması amaçlanan güvenli haberleşmenin çift yönlü olduğunu doğrulamak için ise yine bilgisayar üzerinden SSH bağlantısı ile BB2 ye uzaktan bağlantı yapılmıştır. Port ve baudrate ayarlamaları yapıldıktan sonra RealTerm uygulaması ile "testserver" veri paketi UART haberleşme kanalı üzerinden Şekil 3.35'deki gibi gönderilmiştir.



Şekil 3.35: Sunucuya Gönderilen UART Test Paketi

UART veri paketinin doğru ve eksiksiz bir şekilde gönderildiğini test etmek amacı ile BB2'nin aldığı veri paketi SSH bağlantısının olduğu konsol üzerine Şekil 3.36'daki gibi yazdırılarak kontrol edilmiştir. UART üzerinden alınan ve BB2 içerisinde şifrelenen verinin şifreli hali de konsol üzerine yazdırılmıştır. Ayrıca, BB2 den BB1'e gönderilen şifreli paketlerin BB1 içerinde şifresinin çözüldükten sonraki durumu ve boyutu da BB1 konsoluna yazdırılmıştır.

BB1 ve BB2 konsol çıktıları incelendiğinde güvenli haberleşmenin sunucudan istemciye olan yönünün de doğru bir şekilde çalıştığı doğrulanarak güvenli haberleşmenin çift yönlü olarak çalışabildiği görülmüştür.



Şekil 3.36: Sunucu Uygulamasının Okuduğu UART Test Paketi

4. BULGULAR

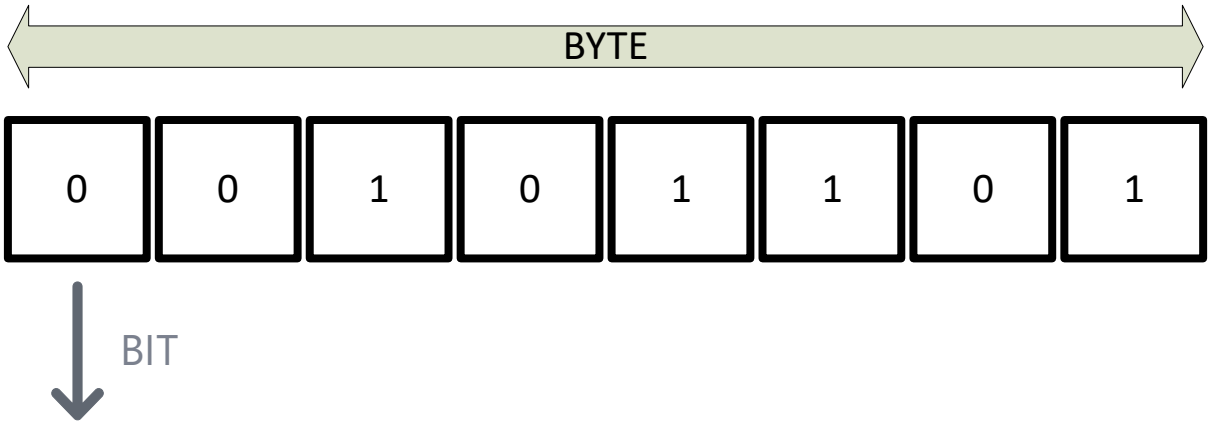
4.1. ŞİFRELİ HABERLEŞMENİN SİSTEM ÜZERİNDE GERÇEKLEŞTİRİLMESİ

Güvensiz haberleşmenin kripto modüller kullanılarak güvenli hale getirilmesi çalışması kapsamında içerisinde Linux tabanlı Debian işletim sistemi kurulu olarak iki tane BeagleBone Blue hazır geliştirme kartı üzerinde geliştirilen Güvenli Haberleşme Uygulaması sorunsuz bir şekilde çalıştırılmıştır. Uygulama TCP ve UART haberleşme protokollerini çift yönlü kullanabilmesinin yanında OpenSSL kütüphanesinden faydalanmak suretiyle şifreleme, şifre çözme ve kripto anahtar üretebilme kabiliyetlerine sahip olarak tasarlanmış ve geliştirmesi tamamlanmıştır. Kriptolu haberleşme, oluşturulan test düzeneğinde test edilmiş ve iki yönlü olarak da doğru bir şekilde çalıştığı gözlemlenmiştir. Sunucu olarak tasarlanmış modüle UART protokolü üzerinden gönderilmiş olan veri sırası ile sunucu tarafından alınmış, şifrelenmiş, istemci olarak çalışan modüle gönderilmiş, istemci tarafından alınmış ve şifre çözme algoritmaları ile asıl metne ulaşılmıştır. Yine aynı şekilden anlaşılabileceği gibi istemci olarak çalışan kripto modüle UART protokolü ile gönderilen veri sırası ile istemci olarak çalışan kripto modül tarafından alınmış, şifreleme algoritmaları kullanılarak şifrelenmiş, sunucu olarak çalışan modüle gönderilmiş, sunucu tarafından paketler okunmuş ve şifre çözme algoritmaları ile şifresi çözülerek asıl metne ulaşılmıştır.

4.2. KRİPTOLU HABERLEŞME SIRASINDA VERİ BOYUTUNDAKİ DEĞİŞİM

AES-256 bit şifrelenmiş verinin boyutu, 128 bit veya 16 bayt olan şifreleme algoritmasının blok boyutu tarafından belirlenir. 1 bayt büyüklüğünde bir veri Şekil 4.1‘deki gibi 8 bit büyüklüğüne karşılık gelmektedir.

AES sabit boyutlu veri blokları üzerinde çalışır ve tek bir bayt şifrelenirken, yine de blok boyutuna kadar doldurulması gerekir. Bu gerekliliği karşılamak için yapılan işlemin adı padding olarak belirtilir. AES algoritması kullanılarak yapılan şifreleme işlemlerinde, girdi boyutu ne olursa olsun, çıktı her zaman blok boyutunun bir katı olacaktır. Dolayısıyla, 1 bayt uzunluğundaki bir verinin AES-256 kullanılarak şifrelenmesi, 16 baytlık şifrelenmiş bir çıktı ile sonuçlanacaktır.



Şekil 4.1: Bir Bayt Verinin Boyutu

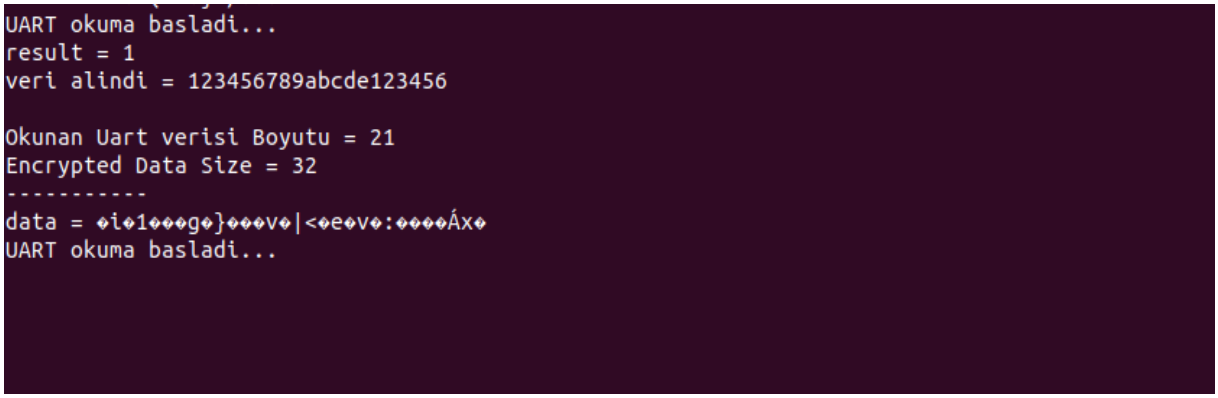
GHU içerisinde olan kripto işlemler incelendiğinde karşımıza aynı sonuçlar çıkmaktadır. Örneğin, Şekil 4.2'deki gibi 2 bayt büyüklüğündeki bir verinin UART kanalı ile kripto modüllerin birisine gönderilmesi ile gelen veri şifrelenmiş ve şifrelenen verinin boyutu konsol üzerine yazdırılmıştır. Veri boyutundaki değişime bakılmış ve orijinal verinin boyutu 2 bayt ve şifrelenmiş verinin boyutu 16 bayt olarak görülmüştür.

```
debian@beaglebone:~/work/PassCrypt/client$ ./build/src/Client
Uart constructor
UART okuma basladi...
Baglandi...
result = 1
veri alindi = A

Okunan Uart verisi Boyutu = 2
Encrypted Data Size = 16
-----------
data = P=◆&(-3◆}~⌊◆◆◆
UART okuma basladi...
```

Şekil 4.2: İki Bayt Verinin AES 256-Bit Şifreleme Sonucu Büyülük Değişimi

Boyutu 16 bayttan daha büyük veri geldiğinde şifrelenmiş verinin boyutunun nasıl değişeceğini görmek için boyutu 21 bayt olan bir test verisi RealTerm uygulaması yardımı ile UART üzerinden BB'ye gönderilmiş ve şifrelenmiş verinin boyutunun 32 bayt olduğu Şekil 4.3'deki gibi gözlemlenmiştir.



Şekil 4.3: 21 Bayt Verinin AES 256-Bit Şifreleme Sonucu Büyüklük Değişimi

Tablo 4.1'de farklı boyutlarda veriler şifrelenmek üzere kripto modüle gönderildiğinde, çıktı olarak alınan şifrelenmiş verilerin boyutları gösterilmiştir. Dolayısı ile AES şifreleme ile şifreleme yapıldığında veri boyutu 16 bayt katları şeklinde olmaktadır. Veri boyutu 16 bayt altında ise, şifreli hali 16 bayt'a tamamlanarak büyütülmektedir.

Tablo 4.1: Farklı Büyüklükteki Verilerin AES 256-Bit Şifreleme Sonucu Büyüklükleri

Okunan UART Veri Boyutu (Bayt)	**Şifrelenmiş Veri Boyutu (Bayt)**
2	16
3	16
4	16
5	16
18	32
36	48

4.3. ŞİFRELİ HABERLEŞMENİN İLETİM HIZINA ETKİSİ

Güvensiz bir haberleşme ile veri iletimi yapılan sistemlerin kripto modüller kullanılarak güvenli hale getirilmesi çalışması kapsamında mevcut güvensiz haberleşmeye ek olarak; UART okuma yazma, sunucu-istemci haberleşmesi, şifreleme ve şifre çözme işlemleri yapıldığı için karşımıza ekstra bir süre maaliyeti çıkmaktadır. Bu süre içerisinde yoğun matematiksel işlemlerle yapılan şifreleme ve şifre çözme işlemlerinin büyük pay sahibi olduğu öngörülerek, bu kriptografik işlemlerin süreleri hakkında bilgi sahibi olmak için Tablo 4.2 ve Tablo 4.3'deki gibi ölçümler yapılmıştır. Bu işlemlerin haberleşme süresine ms(milisaniye) mertebesinden daha az bir etkiye sahip olduğu görülmüştür.

4.4. VERİ BOYUTUNUN ŞİFRELEME SÜRESİNE OLAN ETKİSİ

GHU içerisinde güvenli haberleşmeyi sağlama amacıyla yapılan şifreleme işlemlerinde değişken veri boyutları ile karşılaşılabileceği öngörülmüş ve bu durumun şifreleme süresine etkisinin ölçümü amacı ile kod içerisinde şifreleme işleminin başlangıcı ve sonunda zaman tutularak işlem için harcanan zaman farklı veri boyutları için Tablo 4.2'deki gibi ölçülmüştür. AES 256-Bit algoritmasında veriler 128 Bit uzunluğunda bloklar halinde şifrelenmektedir. Veri büyüklüğünün artması ve bu blokların çoğalması, şifreleme süresini artıracaktır.

Tablo 4.2: Farklı Boyutlardaki Verilerin Şifrelenme Süreleri

Veri Boyutu (Bayt)	Şifreleme Süresi (s)
1	0.00015713
16	0.000188293
32	0.000196709
1024	0.000195122
2048	0.000260227
4096	0.000281593

Aynı yöntem ile farklı veri boyutları için şifre çözme işleminin de süreleri ölçülerek Tablo 4.3‘de gösterilmiştir.

Bu veriler ışığında denenmiş olan boyut artışlarının haberleşme hızına olan etkisi ihmal edilebilecek kadar az olduğu görülmüştür.

Tablo 4.3: Farklı Boyutlardaki Verilerin Şifre Çözülme Süreleri

Veri Boyutu (Bayt)	Şifreleme Süresi (s)
1	0.00015522
16	0.000178149
32	0.000198353
1024	0.000239451
2048	0.000273184
4096	0.000293659

4.5. ANAHTAR BOYUTUNUN ŞİFRELEME SÜRESİNE ETKİSİ

AES CBC şifreleme algoritmasının desteklediği 128-Bit, 192-Bit ve 256-Bit boyutlarındaki kripto anahtarlar ile testler yapılmış ve sonuçlar tekrar eden denemeler için küçük farklılıklar gösterse de örnek olması için denemelerden bir tanesi Tablo 4.4’de gösterilmiştir. AES şifreleme içerisinde bulunana round sayısı doğrudan anahtar boyutu ile ilişkilidir ve 128-Bit için 10, 192-Bit için 12 ve 256-Bit için 14’tür. Dolayısı ile anahtar uzunluğunun artması işlem süresini etkileyeceği öngörülmüştür. Sonuçlara bakıldığında ise anahtar uzunluğunun şifreleme süresine olan etkisi görülmektedir.

Tablo 4.4: Faklı Anahtar Boyutları İle Şifreleme Süreleri

Anahtar Boyutu (Bit)	Şifreleme Süresi (s)
128	0.00017854
192	0.00020246
256	0.00025713

4.6. AES 256-BIT ŞİFRELEME ALGORİTMASININ GÜVENİRLİĞİ

AES 256-bit CBC şifreleme algoritmaları ile şifrelenmiş bir bayt büyüklüğündeki verinin şifresini gerekli kripto anahtar bulunmadan çözme denemeleri ile hassas veriye yetkisiz erişim girişimleri yapılma durumunda, şifrelenmiş veri AES-256'nın güçlü kriptografik özellikleri sayesinde oldukça güendedir. AES-256 256 bitlik bir kripto anahtar kullanır ve bu anahtar uzunluğu 2^{256} olası anahtar kombinasyonunu beraberinde getirir. Bu kadar fazla anahtar kombinasyonu ise çok geniş bir alana karşılık gelir.

AES-256 algoritması ile şifrelenmiş verinin şifresini anahtar olmadan çözme işlemi, kaba kuvvet saldırısı olarak da tanımlanan bir siber saldırı ile denenirse, tüm anahtar kombinasyonlarını denemek günümüz teknolojisi ile saniyede 10^{18} işlem yapabilen bir süper bilgisayar kullanıldığında bile 2^{256} deneme yapabilmek milyarlarca yıl süreye ihtiyaç vardır.

$$1 \text{ saat} = 60 \text{ dakika} = 60 \times 60 = 3600 \text{ saniye} \quad (4.1)$$

$$1 \text{ gün} = 24 \text{ saat} = 24 \times 3600 = 86400 \text{ saniye} \quad (4.2)$$

$$1 \text{ yıl} = 365 \text{ gün} \times 86400 \text{ saniye/gün} = 31.536.000 \text{ saniye} \quad (4.3)$$

$$1 \text{ yıldaki yapılabilecek deneme sayısı} = 10^{18} \times 31.536.000 \approx 10^{25} \quad (4.4)$$

Günümüz teknolojisi göz önünde bulundurulduğunda GHU güvenilir bir haberleşmeyi sağlamaktadır.

4.7. SİSTEMİN MALİYET TABLOSU

Güvensiz haberleşmeyi güvenli hale getirmek için kripto modülleri geliştirme çalışması veri iletim güvenliğini sağlarken performans ve maliyet dengesi gözetilerek yapılmıştır. Amaç endüstri de hali hazırda kurulu olan sistemlerin mümkün olan en az değişiklik ve maliyetle güvenli hale getirilmesidir. Dolayısı ile bu çalışmada kullanılan yazılım ve donanım maliyetleri önem arz etmektedir. GHU çalışmasında yazılım tarafında kullanılan araçlar özellikle açık kaynak ve ücretsiz olanlardan seçilmiştir. Donanım tarafının maliyeti ise Tablo 4.5'de belirtilmiştir.

Tablo 4.5: Maliyet Tablosu

Bileşen Adı	Adet	Birim Fiyat (USD)	Maliyet (USD)
BeagleBone Blue	2	35	70
USB-TTL Dönüştürücü	2	15	30
BB Blue UART konektörü	2	1	2

5. TARTIŞMA

Endüstrideki cihazlar arasında güvenli iletişim için kriptografik modüllerin kullanımı, günümüzün dijital ortamında giderek daha da önemli hale gelmektedir. Bağlı cihazların ve IoT(Internet of Things)'nin yaygınlaşması ile birlikte, güvenli iletişim ve veri aktarımının sağlanması endüstriyel sistemler için kritik bir konu haline gelmiştir. Güvenli haberleşme uygulaması çalışmasının bize sağladığı küçük simülasyon ortamı sayesinde, endüstride güvenli iletişim için kriptografik modüller kullanılmasının faydaları ve zorlukları hakkında fikir sahibi olunabilecektir.

İlk olarak, kriptografik modüller, veri ve iletişim güvenliğini sağlamak için sağlam ve kanıtlanmış bir yöntem sunmaktadır. Kriptografik modüller, AES, RSA ve Transport Layer Security(TLS) gibi güvenlik özellikleri kapsamlı bir şekilde incelenmiş ve test edilmiş köklü kriptografik algoritmalara ve protokollere dayanmaktadır. Endüstriyel sistemler bu modülleri kullanarak iletişimlerinin yetkisiz erişim, müdahale ve kurcalamaya karşı korunmasını sağlamaktadır.

Kriptografik modülleri kullanmanın bir diğer önemli avantajı da iletişimin güvenliğini sağlamak için standartlaştırılmış bir çerçeve sunmalarıdır. Standartlaştırılmış kriptografik algoritmaların ve protokollerin kullanılması, farklı üreticilerden veya satıcılardan olsalar bile farklı cihazların ve sistemlerin birbirleriyle güvenli bir şekilde iletişim kurabilmesini sağlar. Bu standardizasyon, yeni cihazların endüstriyel sistemlere entegre edilmesi sürecini basitleştirerek kuruluşların operasyonlarını genişletmelerini ve modernize etmelerini kolaylaştırır.

Bununla birlikte, endüstriyel ortamlarda kriptografik modülleri kullanmanın bazı zorlukları da vardır. En büyük zorluklardan biri kriptografik anahtarları ve sertifikaları yönetme ihtiyacıdır. Kriptografik anahtarlar verileri şifrelemek ve şifrelerini çözmek için, sertifikalar ise cihazların ve kullanıcıların kimliklerini doğrulamak için kullanılır. Bu anahtarları ve sertifikaları yönetmek karmaşık olabilir ve güvenliklerini sağlamak için dikkatli bir planlama ve uygulama gerektirir. Kuruluşlar, anahtarlarını ve sertifikalarını korumak ve yanlış ellere geçmelerini önlemek için sağlam anahtar yönetimi politikalarına ve prosedürlerine sahip olmalıdır. Bir diğer zorluk ise kriptografik işlemlerin endüstriyel sistemler üzerindeki performans etkisidir. Kriptografik işlemler hesaplama açısından yoğun olabilir ve doğru şekilde uygulanmazsa endüstriyel sistemlerin performansını yavaşlatabilir. Kuruluşlar sistemlerinde kriptografik

modülleri uygularken güvenlik ihtiyacı ile performans ihtiyacını dikkatli bir şekilde dengelemelidir.

GHU çalışması kapsamında AES 256-Bit CBC şifreleme algoritmaları kullanılmıştır. Bu algoritmanın seçiminde, büyük boyuttaki verilerin şifreleneceği ve bu verilerin hassasiyetinin yüksek olduğu kabulü ile yola çıkılmıştır. AES 256-Bit CBC algoritması blok şifreleme yapmaktadır ve 16 bayt katları olacak şekilde şifrelenmiş veriler üretmektedir. Şifrelenecek veri 16 bayt altında ise şifreleme esnasında veri boyutu artırılmaktadır. Dolayısı ile hassas olmayan ve düşük boyuttaki veri şifrelemelerinde veri boyutu yükseleceği için verimli bir yöntem olmayacaktır. Şifrelenmek istenen verinin hassasiyetine göre kullanılacak olan kripto algoritma seçimi yapılabilir ve kullanılacak kripto anahtarın uzunluğunun belirlenebilir. Ayrıca, ihtiyaç duyulan performansa göre işlemci seçimi yapılabilir.

Ferreira ve Silva yaptıkları çalışmada gömülü sistemler arasındaki haberleşmenin şifreli olmasını sağlamak için AES kripto algoritmalarını kullanmışlardır. Bu çalışma için Relic adı verilen, gömülü sistemler için oluşturulmuş bir kripto kütüphane kullanmışlardır. Çalışma kapsamında genel amaçlı bir bilgisayar ve bir gömülü sistemin haberleşmesi yanında iki gömülü sistemin haberleşmesi de AES algoritmaları ile şifreli hale getirilmiştir. Çalışmada gömülü sistem olarak Raspberry Pi 3 kullanılmıştır. Ayrıca bu çalışmada farklı mesaj boyutları ve algoritmanın desteklediği farklı anahtar boyutları ile işlemler yapılmış ve bulgular incelendiğinde AES 256-Bit ile şifreleme süresinin AES 128-Bit ile şifrelemekten %22 ve AES 192-Bit ile şifrelemek ise AES 128-Bit ile şifrelemekten %12 daha uzun olduğu görülmektedir[36].

Güvenli Haberleşme Uygulamasında ise Tablo 5.1'de olduğu gibi kripto kütüphane olarak yaygın bir kullanıma sahip olan OpenSSL kütüphanesi ve gömülü sistem olarak ise Beaglebone Blue geliştirme kartı kullanılmıştır. Ana mikrodenetleyici olarak ARM Cortex A8 kullanılmış olsa da kullanılan platform, işletim sistemi ve kripto kütüphane farklıdır. Bulgulara bakıldığında, Ferreira ve Silva' nın yaptığı çalışmada olan şifreleme sürelerine benzer sürelerin elde edildiği görülmektedir. Ayrıca, GHU çalışmasında AES algoritmalarının desteklediği anahtar boyutları olan 256-Bit, 192-Bit ve 128-Bit uzunluğundaki anahtarlar ile şifreleme işlemleri yapıldığında, 256-Bit şifreleme süresinin 128-Bit şifreleme süresinden yaklaşık %28 uzun olduğu ve 192-Bit şifrele süresinin 128-Bit şifreleme süresinden yaklaşık olarak %9 uzun

olduğu görülmüştür. Bu sonuçlar değerlendirildiğinde anahtar uzunluğunun şifreleme süresini artırdığı görülecektir.

Tablo 5.1: Performans Karşılaştırması

İşlemci	Platfotm	Kütüphane	Hafıza	Algoritma	Veri Boyutu	Şifreleme Süresi
AM335x 1GHz ARM Cortex-A8	BeagleBone Blue	OpenSSL	eMMC Flash	AES 256-bit CBC	2MB	0.000295sn
OMAP3530 720MHz ARM cortex-A8	Raspberry Pi 3	Relic	Micro SD	AES 256-bit CBC	2MB	0.000183sn

Bunun yanında gömülü sistem olarak Raspberry Pi yerine BeagleBone Blue kullanmanın önemli bir avantajı, dahili olarak eMMC bellek içermesi ve işletim sistemini bu bellek üzerinde çalıştırabilmesidir. Raspberry Pi ise SD kart üzerinden işletim sistemi çalıştırmaktadır. Simetrik şifrelemede anahtar güvenliğinin önemini düşünecek olursak, sistemden kolayca çıkarılabilecek bir modül olan SD kart kullanmanın güvenlik zafiyeti doğurabileceği görülmektedir.

Son olarak, haberleşmede kablosuz kripto modüllerin kullanılması ile mevcutta kullanılan kablolu iletişim kaldırılarak kablo maliyet kalemi ortadan kalkacak ve bu durum atık kabloların çevreye olan yükünün de azalmasına katkı sağlayacaktır. Bunun yanında kablosuz güvenli iletişim ile lokal ağ içerisinde veri iletim mesafesi sorunu yaşanmayacak ve çalışma alanındaki fiziksel kablo yoğunluğu azalacağı için iş güvenliği artacaktır.

6. SONUÇ VE ÖNERİLER

Sonuç olarak, endüstrideki cihazlar arasında güvenli iletişim için kriptografik modüllerin kullanılması, operasyonlarını modernize etmek isteyen tüm kuruluşlar için çok önemli bir husustur. Kriptografik modülleri uygulamanın bazı zorlukları olsa da, güvenlik ve standardizasyon açısından sağladığı faydalar bunu değerli bir yatırım haline getirmektedir. Mevcut veri iletimine eklenebilecek kripto modüller ile oldukça düşük maliyetlerle bilgi güvenliği artırılabilmektedir. Güvenli iletişim ve veri aktarımına öncelik veren kuruluşlar, günümüzün dijital ortamında başarılı olmak için daha iyi bir konuma sahip olacaklardır.

GHU yüksek güvenlikli bir haberleşme sağlamaktadır ve güncel teknoloji ile bu güvenliği aşmak oldukça zordur. Fakat ileride erişilebilecek teknolojiler ile üretilecek süper bilgisayarların birim zamanda yapabildiği işlem sayısının artmasıyla, şifreleme için kullanılan anahtarın güvenliği zamanla azalacağı öngörülmüştür. Dolayısı ile belli zaman aralıkları ile sunucu ve istemci de bulunması gereken ilgili anahtar, yeni üretilen bir anahtar ile değiştirilirse sistemin güvenlik seviyesi artırılabilir. Bunu gerçekleştirmek için mevcut implementasyona ek olarak, sunucu veya istemci uygulamasında belli periyodlarda anahtar üretimi yapılması ve bu üretilen anahtarın güvenli bir şekilde karşı taraf ile paylaşılması gerçekleştirilebilir. Kripto anahtarı karşı taraf ile güvenli bir şekilde paylaşılması ise; güvenli bir anahtar değişim protokolü olan Diffie-Helman anahtar değişim algoritmalarının kod implementasyonu yapılarak uygulamaya eklenmesi ile sağlanabilecektir[37].

KAYNAKLAR

[1] . Yıldız, Ö., 2017, *Gömülü Bir Sistemde Taşınabilir Kripto Cihazın Geliştirilmesi*, Thesis (MSc), Karabük University

[2] . Reynard, R., 1997, *Secret Code Breaker : A Cryptanalyst's Handbook*, Smith & Daniel Marketing, Jacksonville FL, ISBN: 1-889668-06-0.

[3] . Al-Kadit, I.A., 1992, Origins of cryptology: The Arab contributions, *Cryptologia*, 16(2), 97-126.

[4] . Habeeb, R., 2018, *Improving the Security of Building Automation Systems Through an seL4-based Communication Framework*, Thesis (MSc), University of South Florida

[5] . Bovard, D.A., 2021, *Industrial Control System Data Resiliency*, Thesis (MSc), Boise State University

[6] . Dzung,D., Naedele, M., Von Hoff, T. P. and Crevatin, M., 2005, Security for Industrial Communication Systems, *Proceedings of the IEEE*, 93 (6), 1152-1177.

[7] . Slunjski,M., Sumina, D.,Groš, S. and Erceg,I., 2022, Off-the-Shelf Solutions as Potential Cyber Threats to Industrial Environments and Simple-To-Implement Protection Methodology, *IEEE Access*, 10, 114735-114748.

[8] . Acosta, A.J., Addabbo, T., and Tena-Sánchez, E., 2017,*E*mbedded electronic circuits for cryptography, hardware security and true random number generation: an overview, *International Journal of Circuit Theory and Applications*, 45 (2), 145-169.

[9] . Farooq-i-Azam, M. and *A*yyaz, M.N., 2012, *Hardware and Embedded Systems Security*, Cyber Security Standards, Practices and Industrial Applications: Systems and Methodologies, In: Zubairi J.A (ed.), Chapter 10, Information Science Reference by IGI Global, Pennsylvania, USA, 180-201.

[10] . Gupta, S. and Sharma, J., 2012, A hybrid encryption algorithm based on RSA and Diffie-Hellman, *2012 IEEE International Conference on Computational Intelligence and Computing Research*, 18-20 December 2012 Coimbatore, India, IEEE, 1-4.

[11] . Lemaire, E., 2019, *Pretty Modular Symmetric Encryption (PMSE), compact algorithm for "embedded cryptography" with quite low computational cost,* https://arxiv.org/pdf/1905.08150, [05.04.2023].

[12] . Pyrgas, L. and Kitsos, P., 2020, Compact Hardware Architectures of Enocoro-128v2 Stream Cipher for Constrained Embedded Devices, *Electronics*, 9(9), 1505.

[13] . Atmaja, I. M. A. D. S., Astawa, I. N. G. A., Wisswani, N. W., Nugroho, I. M. R. A., Sunu, P. W. and Wiratama, I. K.,2020, Document Encryption Through Asymmetric RSA Cryptography, *2020 International Conference on Applied Science and Technology (iCAST)*, 24-24 October 2020 Padang, Indonesia, IEEE, 46-49.

[14] . Singh, S.R., Khan, A.K. and Singh, S.R.,2016, Performance evaluation of RSA and Elliptic Curve Cryptography, 2*016 2nd International Conference on Contemporary Computing and Informatics (IC3I)*, 14-17 December 2016 Greater Noida, India, IEEE.

[15] . Liu, H., 2023, Audio block encryption using 3D chaotic system with adaptive parameter perturbation, *Multimedia Tools and Applications*, https://doi.org/10.1007/s11042-023-14572-1

[16] . Lienkov, S., Zhyrov, G., Pampukha, I. and Chetverikov, I., 2019, Block Encryption Algorithm for Digital Information Using Open Keys for Selfgeneration of Closed Random Private Keys, *2019 IEEE International Conference on Advanced Trends in Information Theory (ATIT)*, 18-20 December 2019 Kyiv, Ukraine, IEEE, 200-203.

[17] . Beletsky, A.,2020, Generalized Algorithms for Stream Encryption of Signals in Communication Channels with the UAV, *2020 IEEE Ukrainian Microwave Week (UkrMW)*, 21-25 September 2020 Kharkiv, IEEE, 397-402.

[18] . Barrett, S.F. and Pack, D.J., 2005, *Microcontrollers*, Handbook of Networked and Embedded Control Systems,In: Hristu-Varsakelis, D. and Levine, W.S (ed.), Birkhäuser, Boston, MA, ISBN:978-0-8176-4404-8, 295–322.

[19] . Rozov, A., Anureev, I., Garanina, N., Liakh, T. and Zyubin,V., 2019, Towards Safe Embedded Systems: IndustrialC Translational Semantics for AVR Microcontrollers, *2019 International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON)*, 21-27 October 2019 Novosibirsk, Russia, IEEE, 0857-0861.

[20] . Walden, J., 2022, OpenSSL 3.0.0: An exploratory case study, *2022 IEEE/ACM 19th International Conference on Mining Software Repositories (MSR)*, 23 - 24 May 2022 Pittsburgh, PA, USA, Association for Computing Machinery, ISBN:9781450393034, 735-737.

[21] . Drake, N., 2023, *What is OpenSSL?*, https://www.techradar.com/vpn/what-is-openssl, [Ziyaret tarihi: 8 May 2023].

[22] . Thiruneelakandan, A. and Thirumurugan, T., 2011, An approach towards improved cyber security by hardware acceleration of OpenSSL cryptographic functions, *2011 International Conference on Electronics, Communication and Computing Technologies*, 12-13 September 2011 Villupuram, India, IEEE, 13-16.

[23] . Matysek, M., Adamek M. and Neumann, P., 2003, The Monitoring and Control Network System of a Enzymatic Processor VIA the TCP/IP Protocol Under OS Linux, *2003 4th International Conference on Control and Automation Proceedings*, 12 June 2003 Montreal, QC, Canada, IEEE, ISBN:0-7803-7777-X, 486-490.

[24] . Lim, M., 2019, Directly and Indirectly Synchronous Communication Mechanisms for Client-Server Systems Using Event-Based Asynchronous Communication Framework, *IEEE Access*, vol. 7, 81969-81982.

[25] . Yang, Z., Walker, B., Harris, J.R., Li, Y. and Cao, G., 2020, Optimal Use Of The TCP/IP Stack in User-space Storage Applications with ADQ feature in NIC, *2020 IEEE 26th International Conference on Parallel and Distributed Systems (ICPADS)*,02-04 December 2020 Hong Kong, IEEE, 363-371.

[26] . Bhattacharya, S.P. and Apte, V., 2006, A Measurement Study of the Linux TCP/IP Stack Performance and Scalability on SMP systems, *2006 1st International Conference on Communication Systems Software & Middleware*, 08-12 January 2006 New Delhi, India, IEEE, ISBN:0-7803-9575-1, 1-10.

[27] . Almeida Pereira Zuquim, L.D., Coelho, C.J.N., Fernandes, A.O., Oliveira,M.P. and Tavares, A.I., 2001, An embedded converter from RS232 to Universal Serial Bus, *Symposium on Integrated Circuits and Systems Design*, 15 September 2001 Pirenopolis, Brazil, IEEE, ISBN:0-7695-1333-6, 91-96.

[28] . Basford, M.J., Smartt, C., Thomas, D.W.P., Greedy, S., 2018, On the Disruption of Wired Serial Communication Links by Time Domain Interference, *2018 IEEE International Symposium on Electromagnetic Compatibility and 2018 IEEE Asia-Pacific Symposium on Electromagnetic Compatibility (EMC/APEMC)*, 14-18 May 2018 Suntec City, Singapore, IEEE, ISBN:978-1-5090-3955-5, 183-186.

[29] . Gupta, A.K., Raman, A., Kumar, N. and Ranjan, R., Design and Implementation of High-Speed Universal Asynchronous Receiver and Transmitter (UART), *2020 7th International Conference on Signal Processing and Integrated Networks (SPIN)*, 27-28 February 2020 Noida, India, IEEE, 295-300.

[30] . Mäkinen, S., Leppänen, M., Kilamo,T., Mattila, A.L., Laukkanen, E., Pagels, M. and Männistö, T., 2016, Improving the Delivery Cycle: A Multiple-Case Study of the Toolchains in Finnish Software Intensive Enterprises, *Information and Software Technology*, Vol. 80, 175-194.

[31] . Povazan, I., Popovic, M., Djukic, M. and Cetic, N., 2013, A Retargetable C Compiler for Embedded Systems, *2013 3rd Eastern European Regional Conference on the Engineering of Computer Based Systems*, 29-30 August 2013 Budapest, Hungary, IEEE, 48-54.

[32] . Stewart, J.G., 2015, *Verified Separate Compilation for C*, Thesis (PhD), Princeton University.

[33] . Douglas, M., 2017, *An Empirical Analysis of GNU Make in Open Source Projects*, Thesis (PhD), Queen's University (Canada).

[34] . Tran, A., 2020, *Encrypted Collaborative Editing Software*, Thesis (MSc), University of North Texas.

[35] . Ganapati, Ch., 2001, *On "Secured" Encryption Schemes*, Thesis (MSc), Indian Statistical Institute.

[36] . Ferreira, A.C. and Silva, N.B. F., **2020,** Comparison of secure communication with AES between embedded system and general purpose computer, *2020 IEEE Symposium on Computers and Communications (ISCC)*, 07-10 July 2020 Rennes, France, IEEE, 1-6.

[37] . Chang, R.Y., Lin, S.J. and Chung, W.H., 2013, Diffie-Hellman key distribution in wireless multi-way relay networks, *2013 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*, 29 October- 01 November 2013 Kaohsiung, Taiwan, IEEE, 1-4.

EKLER

İNTİHAL RAPORU İLK SAYFASI

Mevlut_Gonen

ORİJİNALLİK RAPORU

%2	%2	%0	%1
BENZERLİK ENDEKSİ	İNTERNET KAYNAKLARI	YAYINLAR	ÖĞRENCİ ÖDEVLERİ

BİRİNCİL KAYNAKLAR

1	Submitted to The Scientific & Technological Research Council of Turkey (TUBITAK) Öğrenci Ödevi	<%1
2	acikbilim.yok.gov.tr İnternet Kaynağı	<%1
3	www.binance.vision İnternet Kaynağı	<%1
4	www.coursehero.com İnternet Kaynağı	<%1
5	gist.github.com İnternet Kaynağı	<%1
6	www.freepatentsonline.com İnternet Kaynağı	<%1
7	Submitted to Mugla University Öğrenci Ödevi	<%1
8	documents.mx İnternet Kaynağı	<%1
9	citeseerx.ist.psu.edu İnternet Kaynağı	<%1

ETİK KURUL İZİN YAZISI

Uyarı: Canlı denekler üzerinde yapılan tüm araştırmalar için Etik Kurul Belgesi alınması zorunludur.

☐ **Etik Kurul izni gerekmektedir.**

☒ **Etik Kurul izni gerekmemektedir.**

Öğrenci Adı SOYADI
(İmza)

KURUM İZNİ YAZILARI

Uyarı: Canlı ve cansız deneklerle yapılan tüm çalışmalar için kurum izin belgelerinin eklenmesi zorunludur. Gizlilik ve mahremiyet içeren durumlarda kurum adı kapatılmalıdır.

☐ **Kurum izni gerekmektedir.**

☒ **Kurum izni gerekmemektedir.**

Öğrenci Adı SOYADI
(İmza)