FIRAT ÜNİVERSİTESİ

TEKNOLOJİ FAKÜLTESİ

Yazılım Mühendisliği Bölümü

YMH453-Yazılım Kalite Güvencesi ve Testi Dersi Final Projesi

ASİMETRİK VE SİMETRİK ŞİFRELEME ALGORİTMALARIN KARŞILAŞTRILMASI

Geliştiren

170541050-Batuhan ŞEN

Proje Yürütücüleri

Prof.Dr.Resul DAŞ

1.GİRİŞ

1.1. Projenin Amacı
1.2. Projenin Kapsamı

2.Temel Bilgiler

2.1.Simetrik Şifreleme Algoritmaları
- 2.1.1 AES Şifreleme Algoritması
- 2.1.2 DES Şifreleme Algoritması
- 2.1.3 TRİPLE DES Şifreleme Algoritması
- 2.1.4 SHA Şifreleme Algoritması
- 2.1.5 MD5 Şifreleme Algoritması

2.2 Asimetrik Şifreleme Algoritmaları
- 2.2.1 Diffie-Hellman Şifreleme Algoritması
- 2.2.2 RSA Şifreleme Algoritması
- 2.2.3 DSA Şifreleme Algoritması

3.UYGULAMA

3.1.Simetrik Şifreleme Algoritmalarının Uygulaması
- 3.1.1 AES Şifreleme Algoritmasının Uygulaması
  - 3.1.1.1 C# Programlama Dili ile Kodlanması
  - 3.1.1.2 Pyhton Programlama Dili ile Kodlanması
- 3.1.2 DES Şifreleme Algoritmasının Uygulaması
  - 3.1.2.1 C# Programlama Dili ile Kodlanması
  - 3.1.2.2 Pyhton Programalama Dili İle Kodlanması
- 3.1.3 SHA Şifreleme Algoritmasının Uygulanaması
  - 3.1.3.1 C# Programlam Dili İle Kodlanması
- 3.1.4 TRİPLE DES Şifreleme Algoritmasının Uygulanması
  - 3.1.4.1 C# Programlama Dili İle Kodlanması
  - 3.1.4.2 Pyhton Programlama Dili İle Kodlanması
- 3.1.5 MD5 Şifreleme Algoritmasının Uygulanması
  - 3.1.5.1 C# Programlama Dili İle Kodlanması
  - 3.1.5.2 Pyhton Programlama Dili İle Kodlanması

3.2 Asimetrik Şifreleme Algoritmalarının Uygulanması
- 3.2.1 RSA Şifreleme Algoritması
  - 3.2.1.1 C# Programlama Dili İle Kodlanması

4. Uygulama Sonuçları

4.1 Simetrik Şifreleme Algoritmalarının Uygulama Sonuçları
- 4.1.1 AES Şifreleme Algoritmasının Uygulama Sonuçları
  - 4.1.1.1 C# Programlama Dili ile Kodlanmasının Sonuçları
  - 4.1.1.2 Pyhton Programlama Dili ile Kodlanmasının Sonuçları
- 4.1.2 DES Şifreleme Algoritmasının Uygulama Sonuçları
  - 4.1.2.1 C# Programlama Dili ile Kodlanmasının Sonuçları
  - 4.1.2.2 Pyhton Programlama Dili ile Kodlanmasının Sonuçları
- 4.1.3 SHA Şifreleme Algoritmasının Uygulanama Sonuçları
  - 4.1.3.1 C# Programlama Dili ile Kodlanmasının Sonuçları
- 4.1.4 TRİPLE DES Şifreleme Algoritmasının Uygulama Sonuçları
  - 4.1.4.1 C# Programlama Dili ile Kodlanmasının Sonuçları
  - 4.1.4.2 Pyhton Programlama Dili ile Kodlanmasının Sonuçları
- 4.1.5 MD5 Şifreleme Algoritmasının Uygulanma Sonuçları
  - 4.1.5.1 C# Programlama Dili ile Kodlanmasının Sonuçları
  - 4.1.5.2 Pyhton Programlama Dili ile Kodlanmasının Sonuçları

4.2 Asimetrik Şifreleme Algoritmalarının Uygulama Sonuçları
- 4.2.1 RSA Şifreleme Algoritmasının Uygulama Sonuçları
  - 4.2.1.1 C# Programlama Dili ile Kodlanmasının Sonuçları
  - 4.2.2.2 Pyhton Programlama Dili ile Kodlanmasının Sonuçları

5. Sonuç

1.GİRİŞ

1.1 Projenin Amacı

Günümüz teknoloji dünyasında bilgi çok önemli bir hazinedir. Teknoloji çağında bilgi saklanması ise daha kıymetlidir. Bu amaçla kriptoloji bilimi ortaya çıkmıştır. Kriptolojinin genel amacı ise çeşitli iletilerin,yazıların belli bir sisteme göre şifrelenmesi,bu mesajların güvenlikli bir ortamda alıcıya iletilmesi ve iletilmiş mesajın deşifre edilmesidir. Kriptoloji biliminin getirisi olarak ta çeşitli şifreleme algoritmaları ortaya çıkmıştır. Bu projenin amacı ise asimetrik ve simetrik şifreleme algoritmalarının çalışma zamanı, bellekte kapladığı alan gibi yazılım metriklerinin kıyaslanması ve en verimli algoritmanın tespit edilmesini sağlamayı amaçlamaktadır.

1.2 Projenin Kapsamı

Projenin kapsamı simetrik ve asimetrik şifreleme algoritmaları olarak belirlenmiştir. Şifreleme algoritmalarının kodlandığı diller ise C# ve Pyhton olarak belirlenmiştir. Bunların belirlenmesindeki sebeplerde şuan dünyada kullanılan en popüler diller bunlardır. Ve birçok web sayfalarının back-end olarak bu dillerin kullanılmasıdır. Kısaca veri akışının en çok yapıldığı gerçek dünya programlarının birçok' u bu dillerde yapıldığı için bu dillerin bu proje için yeterli olacağı düşünülmüştür.

1.2.1 Kullanılan Yazılım Geliştirme Araçları

1.2.1.1 Visual Studio 2019

Microsoft Visual Studio,Microsoft tarafından geliştirilen bir tümleşik geliştirme ortamıdır(IDE).Microsoft Windows,Windows Mobile,Windows CE,.Net Framework ve Microsoft Silverlight tarafından desteklenen tüm platformlar için yönetilen kod ile birlikte yerel kod ve Windows Forms uygulamaları,web siteleri,web uygulamaları ve web servisleri ile birlikte konsol ve grafiksel kullanıcı arayüzü uygulamaları geliştirmek için kullanılır.(1)

ŞEKİL 1.1 VISUAL STUDIO LOGO

1.2.1.2 PYCHARM

PyCharm, en popüler Pyhton IDE' lerden biridir. Çapraz platform uygulaması olarak mevcut olan PyCharm,Linuc,macOS ve Windows platformlarıyla uyumludur. Günümüzde en çok kullanılan Pyhon IDE' lerinden biridir. Hem Pyhton2(2.7) hem de Pyhton 3(3.5 ve üstü) sürümleri için destek sağlar.Pycharm ilk 2010 yılının Şubat ayında halka yayınlandı.

ŞEKİL 1.2 PYCHARM LOGO

2.TEMEL BİLGİLER

2.1 Simetrik Şifreleme Algoritmaları

Simetrik Sifreleme, bilgileri şifreleme ve deşifre etmek için yanlızca bir gizli anahtar içeren en basit şifreleme türüdür. Simetrik şifreleme, kriptografi teknikleri ve şifreleme algoritmaları içincde en eski ve en iyi bilinen tekniktir. Bir sayı, bir kelime veya rastgele harfler dizisi olabilen gizli bir anahtar kullanır. Gönderen ve alıcı, tüm mesajları şifreleme ve şifresini çözmek için kullanılan gizli anahtarı bilmelidir.

Simetrik şifreleme düzenleri,iki yada daha fazla kullanıcının ortak kullandığı tek bir anahtara dayanır. Düz metini(plaintext) şifrelemek ve şifresini çözmek için aynı anahtar kullanılır. Şifreleme işlemi, bir düz metinin(girdi) şifreli metin(ciphertext) yaratılmasından meydana gelir.

Simetrik şifreleme sistemin güvenliği,sistemin karşılık gelen anahtarın kaba kuvvet(brute force) uygulanarak tahmin edilmesinin ne derece zor hale getirdiğine dayanır.
Simetrik şifreleme algoritmaların avantajları:

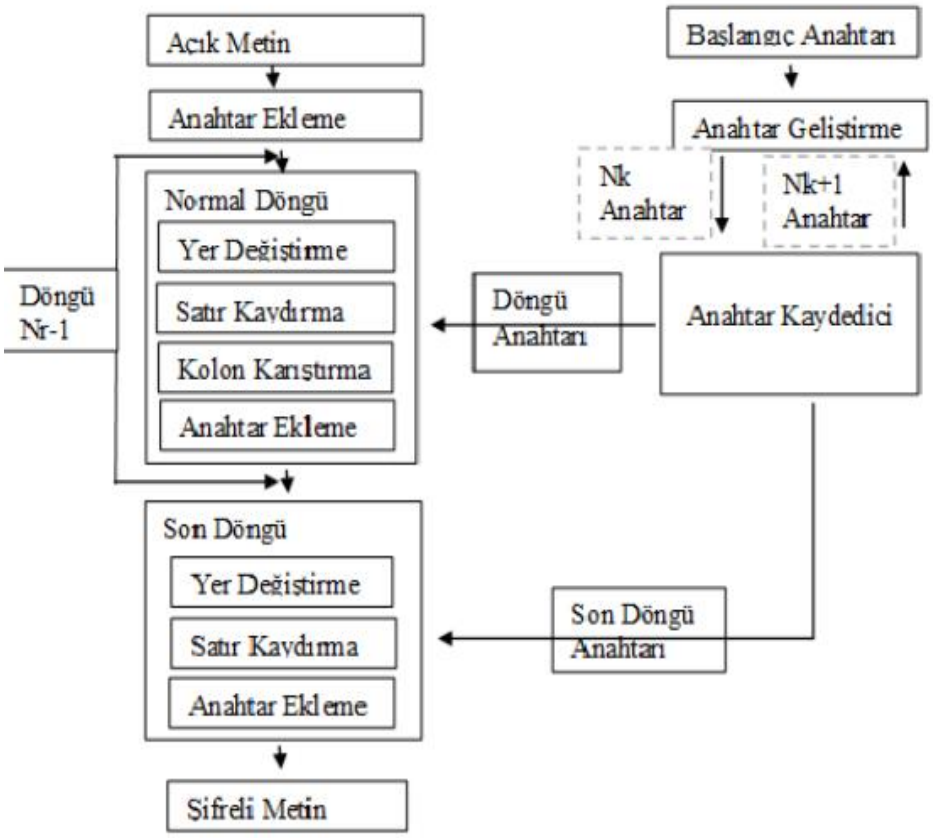
1.Algoritmaları hızlıdır.
2.Donanımla beraber kullanılabilir.
3.Anahtarları kısadır.
4.Kolay ve anlaşılır kullanıma sahiptirler.

Simetrik şifreleme algoritmaların dezavajları:

1.Anahtarın taraflara güvenli ulaştırılması zordur.
2.Anahtarın saklanması zordur.
3.Anahtara herkes ulaşabilir.

2.1.1 AES ŞİFRELEME ALGORİTMASI

AES (Advanced Encryption Standard) Gelişmiş Şifreleme Standardı, elektronik verilerin şifrelemesinde kullanılan, modern bir şifreleme standardıdır. Modern kriptoloji simetrik ve asimetrik şifreleme yöntemleri olmak üzere ikiye ayrılır. Asimetrik şifrelemede açık mesaj (plain text), herkesçe bilinen bir anahtar (key) ile şifrelenir ve karşı tarafın şifreyi çözmek için gizli anahtarı kullanır. Simetrik yöntemde ise şifrelemede de (encrytion) şifre çözmede de (decryption) aynı ve tek bir anahtar kullanılır. AES, simetrik bir blok şifreleme standardıdır ve DES'in yerini almıştır.



SEKİL 2.1 AES ALGORİTMASININ GENEL YAPISI

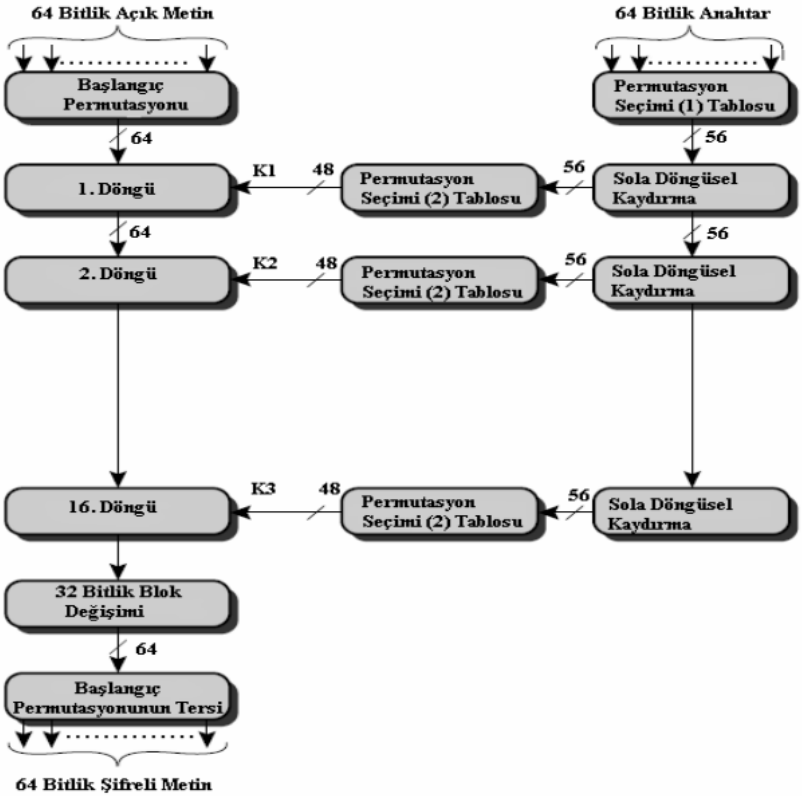
Metin uzunluğu: 128, 192, 256 bit olabilir.

Anahtar uzunluğu: 128, 192, 256 bit olabilir.

Döngü sayısı: Anahtar uzunluğu ve metin uzunluğuna göre değişiklik göstermektedir.

2.1.2 DES ŞİFRELEME ALGORİTMASI

Blok şifreleme algoritmasıdır. Şifrelemeyi metin uzunlukları belli olan bloklar halinde gerçekleştirir. DES algoritması 64 bitlik anahtar uzunluğuna sahip olmasına rağmen 56 bit uzunluğunda simetrik kriptolama tekniği kullanan bir sistemdir. Her kullanımında o kullanıma özel yeni bir anahtar yaratması DES'in güçlü yanı olup, günümüz teknolojisi için algoritmasının yavaş ve 56-bit'lik anahtar uzunluğunun yetersiz kalması DES'in zayıf yönleridir. 2000'li yılların başında kırılmasıyla günümüz teknoliojsi için yetersiz kaldığı görülmüştür ve itibarını kaybetmiştir. DES"in algoritmasından kaynaklanan bu sorunlar "Triple DES" ya da "DES-3" olarak bilinen yeni bir algoritma ile düzeltilmiştir. SSH gibi günümüzde kullanılan çoğu uygulama 3DES"i kullanmaktadır. 3DES algoritması DES şifrelemesinin 3 kere art arda yapılması şeklinde çalışır. Bu yüzden DES'e göre 3 kat daha yavaştır. Bununla birlikte 3DES şifreleme yapmak için uzunluğu 24 bayt olan bir anahtar kullanılır. Her bayt için 1 eşlik biti vardır. Dolayısıyla anahtarın uzunluğu 168 bittir. AES'in geliştirilmesiyle etkinliğini kaybetmiştir çünkü daha gelişmiş bir algoritmaya sahip olan AES şifreleme yöntemine göre 6 kat daha yavaş çalışır.(2)



ŞEKİL 2.2 DES ŞİFRELEME ALGORİTMASININ GENEL BLOK DİYAGRAMI

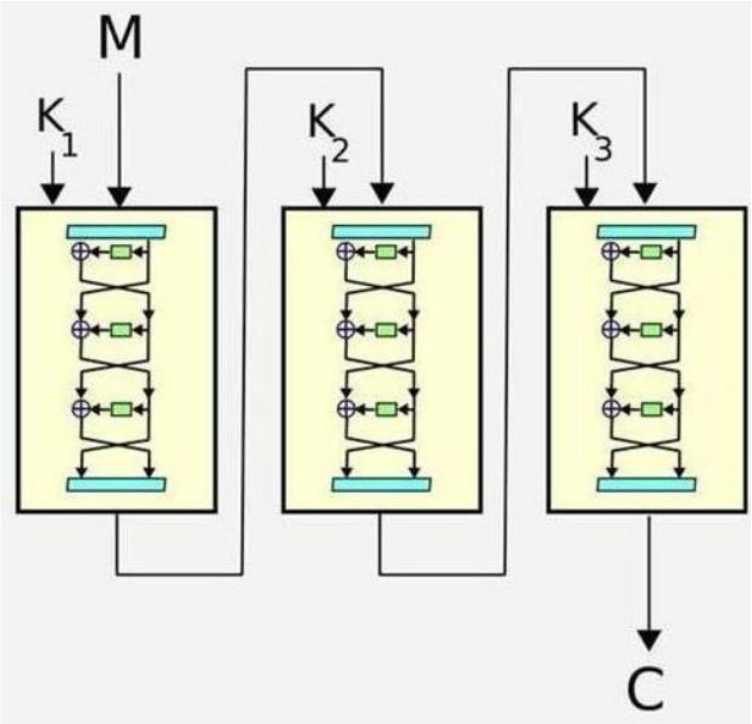
2.1.3 TRİPLE DES Şifreleme Algoritması

3DES (Üçlü DES), 1978 yılında IBM tarafından geliştirilmiş olan bir şifreleme algoritmasıdır. Brute Force saldırılara karşı koymakta zorlanan DES

(Data Encryption Standard-Veri Şifreleme Standardı) algoritmasının üzerine geliştirilmiştir. Bu metot kriptolama anahtarındaki bitleri 3 katına çıkaran, DES`in 3 kez kullanımına dayalı bir şifreleme tekniğidir. 3DES kullanımı DES kullanımına göre 2 kat daha fazla güvenlik sağladığına inanılmaktadır. Bu 112 bitlik koda sahip olunması demektir. Ayrıca kodlama süresince de doğru orantılı olarak arttırmaktadır.

Özellikleri:

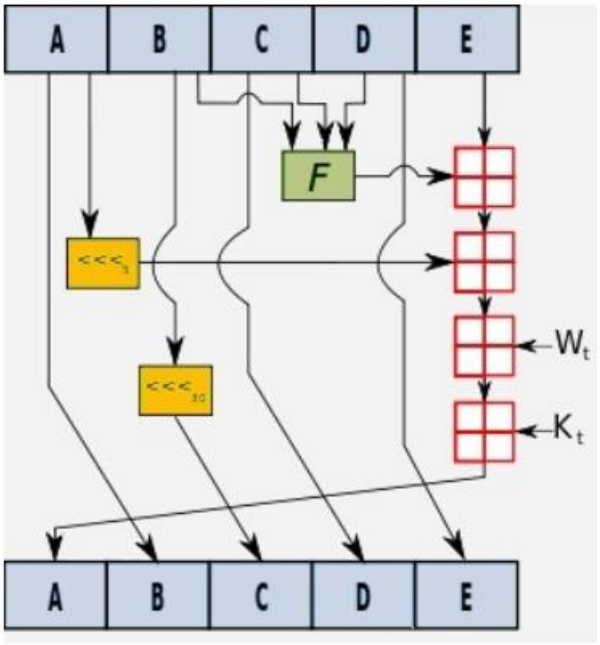
1.Çift yönlü çalışır. Şifrelenmiş veri geri çözülebilir.
2.DES şifrelemesinin 3 kere art arda yapılması şeklinde çalışır.
3.DES şifreleme yöntemine göre 3 kat daha yavaş çalışır.
4.Şifreleme yapmak için uzunluğu 24 bayt olan bir anahtar kullanılır. Her bayt için 1 eşlik biti vardır. Dolayısıyla anahtarın uzunluğu 168 bittir.
5.Veri, 3DES anahtarının ilk 8 baytı ile şifrelenir. Sonra veri anahtarın ortadaki 8 baytı ile çözülür. Son olarak anahtarın son 8 baytı ile şifrelenerek 8 bayt bir blok elde edilir.



ŞEKİL2.3 3DES ALGORİTMASININ AKIŞ ŞEMASI

2.1.4 SHA1 Şifreleme Algoritması

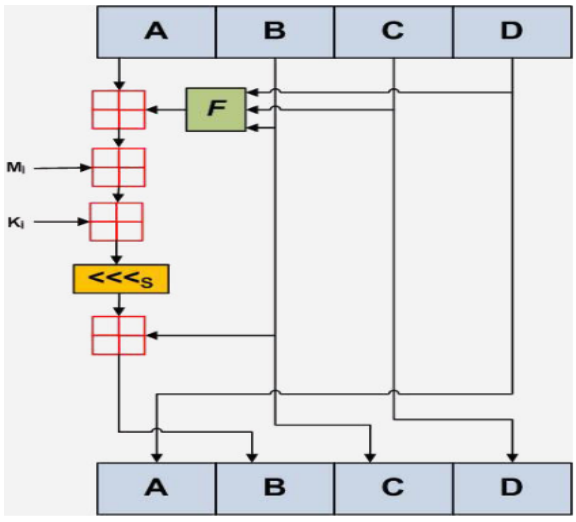
Secure Hashing Algorithm olarak adlandırılan, şifreleme algoritmaları içerisinde en yaygın olarak kullanılan algoritma olduğu kabul gören SHA1, United States National Security Agency tarafından tasarlanmıştır. “Hash” fonksiyonlarına dayalı veritabanı yönetimine (database management) imkan sağlar. SHA1 algoritması ile sadece şifreleme işlemi yapılır, şifre çözümleme işlemi yapılamaz. Diğer SHA algoritmaları içerisinde en yaygın olarak kullanılan SHA1 algoritmasıdır. Günümüzde güvenliği arttırmak amacıyla SHA1 ve MD5 algoritmaları birbiri ardına kullanılarak veriler şifrelenir.



ŞEKİL2.4 SHA1 ALGORİTMASININ AKIŞ ŞEMASI

2.1.5 MD5 Şifreleme Algoritması

Message Digest 5 (MD5) algoritması, verilen dosyanın veya mesajın (şifre vb.) kendine has “parmak izi” nin oluşturulmasını "hash" fonksiyonlarına dayalı olarak sağlayan bir algoritmadır. Bir veritabanı yönetimi (database management) tekniğidir. 1991 yılında MIT (Massachusetts Institute of Technology)’de görev yapan Profesör Ron Rivest tarafından geliştirilmiştir. Profesör Rivest MD5’i MD4’ün bir üst sürümü olarak tasarlamıştır.MD5 algoritması tek yönlü çalışır. Şifreleme yapılır, ancak şifre çözüm işlemi yapılamaz. MD5 algoritması bir alt sürümü olan MD4’e göre yavaş çalışır, ancak şifrelendirme sistemi çok daha karışık ve çözülmesi güçtür.

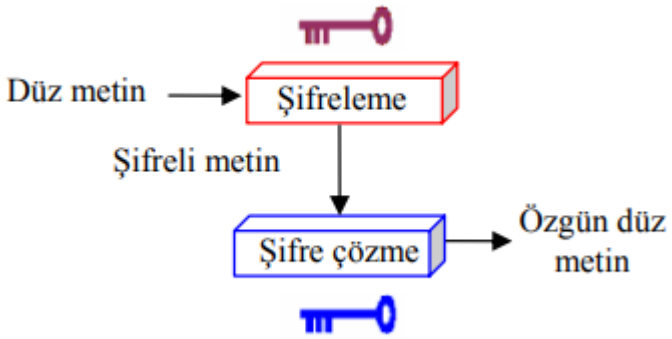


ŞEKİL2.5 MD5 ALGORİTMASININ AKIŞ ŞEMASI

2.2 Asimetrik Şifreleme Algoritmaları

Asimetrik şifreleme işlemlerinde takip edilen yöntemlerin tümü birbirine benzer algoritmalardır. Sayısal değerlerden oluşan bu algoritmalarda iki farklı değer vardır. Bu değerlerin biri herkese açık olarak dahi dağıtılabilecek olan şifreli metindir. Bu metin istenmeyen kişilerin eline geçse dahi şifreli olduğu ve şifrenin yalnızca bir private key tarafından sağlanabileceği için içeriği tam olarak görüntülenemez.

Bu şekilde asimetrik olarak şifrelenen metinlerde genellikle kullanıcılar anlamsız ve metinden tamamen bağımsız olarak meydana getirilmiş karakterler kümesi görürler. Yani, asimetrik şifrelemeyle şifrelenen metinler okunmaya kalkıldığında kesinlikle mantıksız ve rastgele gibi bir araya getirilmiş olan harf ve rakam kümelerinden ibarettir. Asimetrik şifreleme algoritmaları olarak da tanımlayabileceğimiz bu alanda en çok kullanılan yöntemler sosyal imzalar yani RSA ve DSA'lardır.



ŞEKİL2.6 ASİMETRİK ŞİFRELEME ALGORİTMALARIN GENEL YAPISI

2.2.1 Diffie-Hellman Şifreleme Algoritması

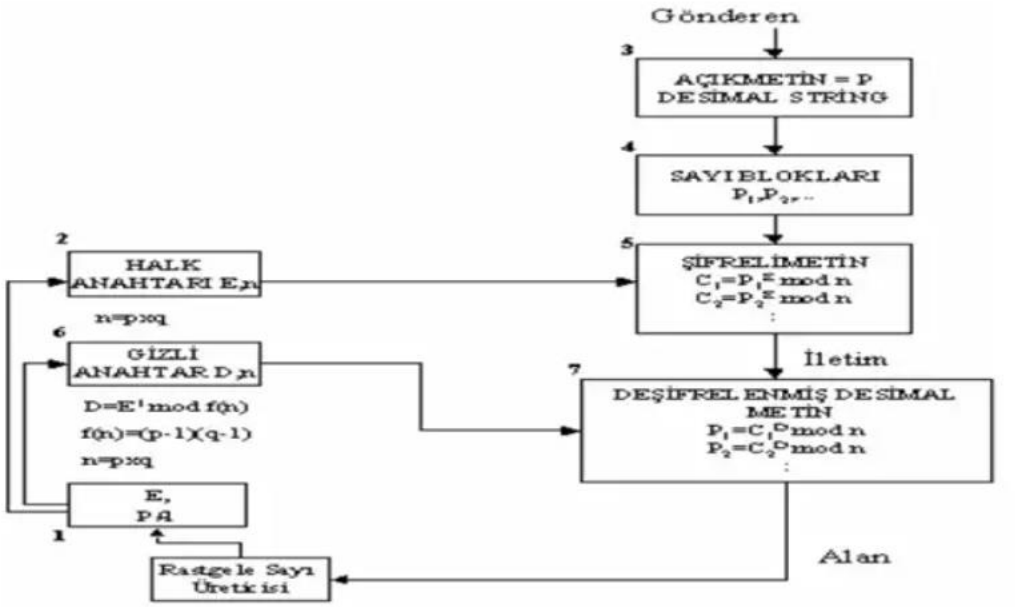
Diffie-Hellman Anahtar Değişimi kriptografik alanlarda kullanılan önemli ve özel yöntemlerden birisidir. Aynı zamanda kriptografi alanında uygulanan ilk pratik anahtar değiştirme yöntemlerinden birisidir. Diffie-Hellman anahtar değişimi metodu karşılıklı iki tarafın ortaklaşa güvensiz medya üzerinden ortak gizli anahtar elde etmelerine olanak sağlar. Bu tasarım ilk defa 1976 yılında Whitfield Diffie ve Martin Hellman tarafından "New Directions in Cryptography" isimli makalelerinde yayımlanmıştır.

Diffie-Hellman gizli iletişimlerde kullanılabilecek ortak gizli anahtar üretir. Bu anahtar da ortak ağlarda (güvenli olmayan kanaldan) güvenli veri alışverişini sağlar.

2.2.2 RSA Şifreleme Algoritması

Diffie–Hellman algoritması oluşturularak simetrik şifreleme algoritmaları için büyük problemi olan gizli anahtarı koruma ve dağıtım büyük ölçüde aşılmıştır. Diffie-hellman algoritması sadece ortak gizli anahtarı belirlemekte kullanılmaktadır. 1977 yılında R.Rivest, A.Shamir ve L.Adleman isminde üç bilim adamının oluşturduğu yeni asimetrik şifreleme algoritması RSA, anahtar dağıtımının yanında şifreleme ve deşifreleme işlemlerini de gerçekleştirmektedir. RSA şifreleme sistemin oluşturulmasıyla birlikte asimetrik şifreleme algoritmalarının günümüzde daha yaygın olarak kullanılması sağlanmıştır. RSA kripto sistemi,

hem gizlilik hem de dijital imza sağlamak amacıyla kullanılabilir. Bu sistemin güvenliği tamsayılarda çarpanlara ayırma probleminin kolaylıkla olmaması temeline dayanır. RSA kripto sisteminde kişilere şifreli mesaj gönderilebilmesi için o kişilerin açık anahtarlarına ihtiyacı vardır. Mesajı alan kişinin de mesajı okuyabilmesi için gizli bir anahtarın olması gerekir.[3]



ŞEKİL2.7 RSA ALGORİTMASININ AKIŞ ŞEMASI

2.2.2 DSA Şifreleme Algoritması

DSA (Digital Signature Algorithm) yani Sayısal İmza Algoritması 1991 yılında NIST (National Institute of Standards and Technology) tarafından sayısal imza algoritması olarak tasarlanmıştır. RSA gibi açık anahtarlı bir şifreleme metodu kullanılmakta olan DSA aynı zamanda özetleme (hash) fonksiyon algoritmalarından da yararlanır. İşleyiş olarak basitçe verimizin içine eklediğimiz hash (özet)' in değişmesiyle verinin değiştirildiğini anladığımız bir algoritmadır. ElGamal imza algoritmasının da varyansı olarak bilinir. Teknolojinin hızla büyüdüğü ve birçok verinin dijitale geçiş yaptığı günümüzde de veri güvenliği açısından DSA' nın önemi büyüktür. Sayısal imzanın ıslak imzadan en büyük farkı, imzanın veriye göre değişmesidir.(Hash Fonksiyonları)

3.UYGULAMA

3.1.1 AES Şifreleme Algoritmasının Uygulaması

3.1.1.1 C# Programlama Dili ile Kodlanması

```
tic byte[] Encrypt(string plainText, byte[] Key, byte[] IV)

 byte[] encrypted;
 //  AesManaged yazma.
using (AesManaged aes = new AesManaged())
 {
    //Şifreleyici oluşturma
    ICryptoTransform encryptor = aes.CreateEncryptor(Key, IV);
    // MemoryStream  oluşturma(Kısa bir süre için bellekte tutulacak akım (stream) oluşturur.)
    using (MemoryStream ms = new MemoryStream())
    {
       //CryptoStream sınıfını kullanarak kripto akışı oluşturun. Bu sınıf şifrelemenin anahtarıdır
       // and encrypts and decrypts data from any given stream. In this case, we will pass a memory stream
       // to encrypt
       using (CryptoStream cs = new CryptoStream(ms, encryptor, CryptoStreamMode.Write))
       {
          // StreamWriter oluşturun ve bir akışa veri yazın
          using (StreamWriter sw = new StreamWriter(cs))
              sw.Write(plainText);
          encrypted = ms.ToArray();
       }
    }
 }
 // Return encrypted data
 return encrypted;
```

ŞEKİL 3.1 AES ENCRYPT METODU

```
static string Decrypt(byte[] cipherText, byte[] Key, byte[] IV)
{
    string plaintext = null;
    // Create AesManaged
    using (AesManaged aes = new AesManaged())
    {
        // Create a decryptor
        ICryptoTransform decryptor = aes.CreateDecryptor(Key, IV);
        // Create the streams used for decryption.
        using (MemoryStream ms = new MemoryStream(cipherText))
        {
            // Create crypto stream
            using (CryptoStream cs = new CryptoStream(ms, decryptor, CryptoStreamMode.Read))
            {
                // Read crypto stream
                using (StreamReader reader = new StreamReader(cs))
                    plaintext = reader.ReadToEnd();
            }
        }
    }
    return plaintext;
}
```

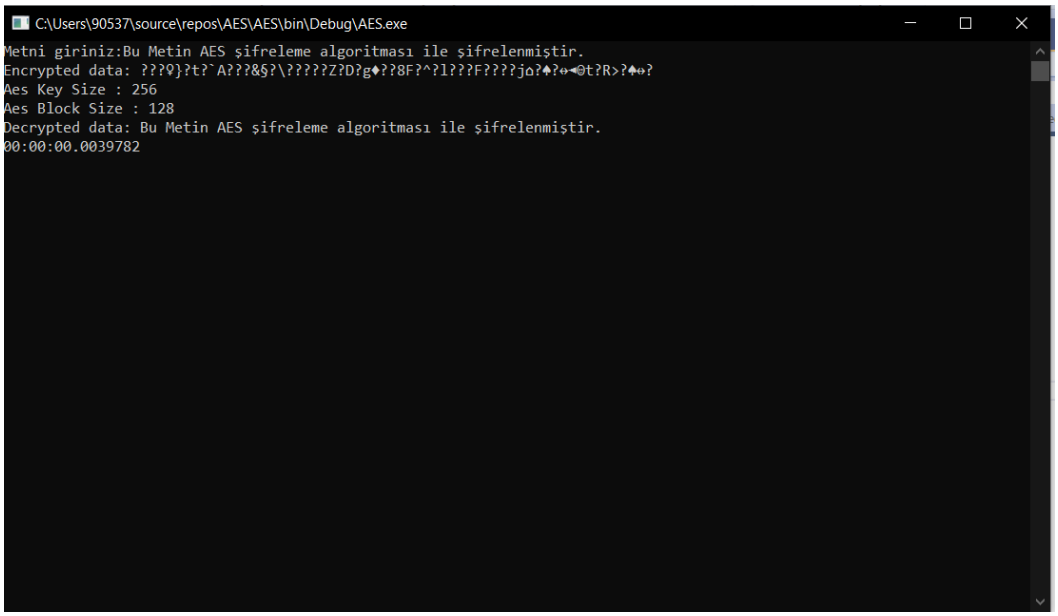
ŞEKİL 3.2 AES DECRYPT METODU

```
public void EncryptAesManaged(string raw)
{
    try
    {
        //Yeni bir anahtar ve başlatma vektörü (IV) oluşturan Aes oluşturma
        //Şifreleme ve şifre çözmede aynı anahtar kullanılmalıdır
        using (AesManaged aes = new AesManaged())
        {
            Stopwatch stop = new Stopwatch();
            // Encrypt string
            stop.Start();
            byte[] encrypted = Encrypt(raw, aes.Key, aes.IV);
            // Şifreli Metin Yazdırma
            Console.WriteLine("Encrypted data" + (System.Text.Encoding.UTF8.GetString(encrypted)));
            // Şifresi Çözülmüş metin.

            string decrypted = Decrypt(encrypted, aes.Key, aes.IV);
            // Print decrypted string. It should be same as raw data
            Console.WriteLine("Decrypted data" + decrypted);
            stop.Stop();
            Console.WriteLine(stop.Elapsed);
        }
    }
    catch (Exception exp)
    {
        Console.WriteLine(exp.Message);
    }
    Console.ReadKey();
}
static byte[] Encrypt(string plainText, byte[] Key, byte[] IV)
```

ŞEKİL 3.3 AES ENCRYPTAESMANAGED METODU

```
public static void Main()
{

    Program pg = new Program();

    Console.WriteLine("Enter text that needs to be encrypted..");
    string data = Console.ReadLine();
    pg.EncryptAesManaged(data);

    Console.ReadLine();
    Console.ReadKey();

}
```

ŞEKİL 3.3 AES ANA METODU



ŞEKİL 3.4 PROGRAM ÇIKTISI

3.1.1.2 Pyhton Programlama Dili ile Kodlanması

Pyhton Progralama dili ile Kodlanması için pycryptodome ve secrets kütüphanelerinin import edilmesi lazımdır. Secrets kütüphanesinin fonksiyonu olan token_bytes() byte türünde kullanılır.1 byte'in ise 8 bit olduğu bilindiği için AES'te bit türünden şifreleme yaptığı için 16,24,32 byte kullanılabilir. Bu sayıların bit karşılığı ise sırası ile 128,192,256 Bit'e denk gelir. Memory_profile,StopWatch kütüphanesi ise 4.bölümde yazılacak olan Uygulama Sonuçlarını yazabilmemiz için bize gerekli verileri yazdırır.Memory_profile hafızada her metodun ne kadar yer kullandığını yazdırır.StopWatch ise programın çalışma zamanını yazdırır. Crypto.Cipher kütüphanesinin çok önemli bir fonksiyonu daha vardır.MODE fonksiyonun kullanıldığı yerdir.MODE farklı fonksiyonlarda alabilir.En çok kullanılanları ise şunlarıdır:

1.MODE_ECB: Elektronik Kod Defteri (ECB) modu her bloğu ayrı ayrı şifreler. Aynı ve aynı mesajda olan veya aynı anahtarla şifrelenmiş farklı bir mesajda bulunan düz metin blokları, aynı şifre metin blokları, aynı şifre metin bloklarına dönüştürülecektir. Önemli: Bu mod önerilmez çünkü birden fazla güvenlik istismarının kapısını açar. Şifrelenecek düz metin önemli bir tekrar içeriyorsa, şifre metninin her seferinde bir blok kırılması mümkündür. Şifreleme anahtarını belirlemek için blok analizi kullanmak da mümkündür. Ayrıca, aktif bir

düşman, blokların algılanmadan başka blokların yerine geçebilir ve değiştirilebilir, bu da blokların kaydedilmesine ve diğer noktalarda akışa algılanmasına izin verir.

2.MODE_CBC:Şifreleme Bloğu Zinciri (CBC) modu geri bildirim getirir. Her düz metin bloğu şifrelenmeden önce, bitsel olarak özel bir OR işlemi ile önceki bloğun şifre metni ile birleştirilir. Bu, düz metin birçok özdeş blok içeriyor olsa bile, her birinin farklı bir şifre metin bloğuna şifrelenmesini sağlar. Başlatma vektörü, blok şifrelenmeden önce bitwise özel OR işlemi ile ilk düz metin bloğu ile birleştirilir. Şifre metin bloğunun tek bir biti karıştırılırsa, karşılık gelen düz metin bloğu da karıştırılır. Ek olarak, sonraki blokta, orijinal karıştırılmış bit ile aynı konumda bulunan bir bit, karıştırılacaktır.

3. MODE_CFB: Şifreleme Geri Bildirimi (CFB) modu, düz metnin küçük artışlarını bir seferde tüm bloğu işlemek yerine şifre metnine işler. Bu mod, bir blok uzunluğunda ve bölümlere ayrılmış bir kaydırma kaydı kullanır. Örneğin, blok boyutu 8 bayt ise, bir seferde bir bayt işlenirse, kaydırma kaydı sekiz bölüme ayrılır. Şifre metnindeki bir bit karıştırılırsa, bir düz metin biti karıştırılır ve kaydırma kaydı bozulur. Bu, sonraki birkaç düz metin artışının, kötü bit kaydırma kaydından kaydırılana kadar karıştırılmasıyla sonuçlanır. Varsayılan geri bildirim boyutu algoritmaya göre değişebilir, ancak genellikle 8 bit veya blok boyutunun bit sayısıdır. Geri bildirim bitlerinin sayısını kullanarak.

4.MODE_EAX:Bazı şifreleme kitaplıklarında bir şifre modunu (belirtmek için kullanılan bir sabittir, bir blok şifrenin nasıl kullanılması gerektiğini belirleyen bir dizi kural). "Şifre Çalma ile Elektronik Kod Defteri" anlamına gelir ve belirli saldırı türlerine karşı daha iyi güvenlik ve direnç sağlayan Elektronik Kod Defteri (ECB) modunun bir çeşididir.
ECB modunda, aynı düz metin bloğu her zaman aynı şifreli metin bloğuna şifrelenir ve bu da onu belirli saldırı türlerine karşı savunmasız hale getirebilir. MODE_EAX, düz metin ve (tuşuna göre hesaplanan kısa bir veri parçası ve her bir şifreleme metni bloğuna) anahtarını ekleyerek bir koruma katmanı ekler, iletinin bütünlüğünü doğrulamak için kullanılabilir.

5.MODE_OFB: Çıktı Geri Bildirimi (OFB) modu, düz metnin küçük artışlarını bir seferde tüm bloğu işlemek yerine şifre metnine işler. Bu mod benzer CFB; iki mod arasındaki tek fark, kaydırma kaydının doldurulma şeklidir. Şifre metnindeki bir bit karıştırılırsa, karşılık gelen düz metin biti karıştırılır. Ancak, şifre metninde fazladan veya eksik bitler varsa, düz metin o noktadan itibaren karıştırılır.

```python
from Crypto.Cipher import AES
from secrets import token_bytes
from stopwatch import Stopwatch
import time
from memory_profiler import profile
import cProfile
key=token_bytes(16)#24,32
stopwatch = Stopwatch()


#@profile(precision=5)
def encrypt(msg):
    stopwatch.start()
    cipher=AES.new(key,AES.MODE_EAX)
    nonce=cipher.nonce
    ciphertext,tag=cipher.encrypt_and_digest(msg.encode('ascii'))

    return nonce,ciphertext,tag
```

ŞEKİL 3.5 PYHTON AES ENCRYPT KODLAMASI

```python
def decrpt(nonce,ciphertext,tag):
    cipher=AES.new(key,AES.MODE_EAX,nonce=nonce)
    plaintext=cipher.decrypt(ciphertext)
    try:
        cipher.verify(tag)
        return plaintext.decode("ascii")
    except:
        return False
nonce,ciphertext,tag=encrypt(input("Mesajınızı giriniz"))
plaintext=decrpt(nonce,ciphertext,tag)
print("Şifreli hali",str(ciphertext))
```

ŞEKİL 3.5 PYHTON AES DECRPT KODLAMASI

```
Mesajınızı giriniz:Bu metin aes ile sifrenmistir
Şifreli hali b';\xfa\xaa_R9\x1c\x80\xd7F\xea\xaf\x95C\xfawb\xc4\xa2\xca\xae\x00\xbc.\xec$\xda\x03 '
```

ŞEKİL 3.6 PYHTON AES PROGRAM ÇIKTISI

3.1.2 DES Şifreleme Algoritmasının Uygulaması

3.1.2.1 C# Programlama Dili ile Kodlanması

```
public string Encypt(string message, string password)
{
    byte[] messagebytes = ASCIIEncoding.ASCII.GetBytes(message);//messagebytes byte  dizisi oluşturma
    byte[] passwordbytes = ASCIIEncoding.ASCII.GetBytes(password);//passwordbytes  byte  dizisi oluşturma
    DESCryptoServiceProvider provider = new DESCryptoServiceProvider();//DESCryptoServiceProvider sınıfını değişkene atama
    ICryptoTransform transform = provider.CreateEncryptor(passwordbytes, passwordbytes);///Simetrik Veri Şifreleme Standardı ( oluştururDES) belirtilen anahtar ( ile şifreleme n
    CryptoStreamMode mode = CryptoStreamMode.Write;
    MemoryStream stream = new MemoryStream();
    CryptoStream crypto = new CryptoStream(stream, transform, mode);
    crypto.Write(messagebytes, 0, messagebytes.Length);
    crypto.FlushFinalBlock();//Temel alınan veri kaynağını veya depoyu arabelleğin geçerli durumuyla güncelleştirir.
    byte[] encyptedMessagBytes = new byte[stream.Length];
    stream.Position = 0;
    stream.Read(encyptedMessagBytes, 0, encyptedMessagBytes.Length);
    string encyptMessage = Convert.ToBase64String(encyptedMessagBytes);
    return encyptMessage;
```

ŞEKİL 3.7 C# DES ENCYPT METODU

```
public string Descrypto(string encryptedMessage, string password)
{
    byte[] encryptedMessageBytes = Convert.FromBase64String(encryptedMessage);
    byte[] passwordBytes = ASCIIEncoding.ASCII.GetBytes(password);
    DESCryptoServiceProvider provider = new DESCryptoServiceProvider();
    ICryptoTransform transform = provider.CreateDecryptor(passwordBytes, passwordBytes);
    CryptoStreamMode mode = CryptoStreamMode.Write;
    MemoryStream memory = new MemoryStream();
    CryptoStream cryptoStream = new CryptoStream(memory, transform, mode);
    cryptoStream.Write(encryptedMessageBytes, 0, encryptedMessageBytes.Length);
    cryptoStream.FlushFinalBlock();

    byte[] decryptoMessageBytes = new byte[memory.Position];
    memory.Position = 0;
    memory.Read(decryptoMessageBytes, 0, decryptoMessageBytes.Length);
    string message = ASCIIEncoding.ASCII.GetString(decryptoMessageBytes);
    return message;


}
```

ŞEKİL 3.8 C# DES DESCRYPTO METODU

```
static void Main(string[] args)
{
    Stopwatch stopwatch = new Stopwatch();
    Program pg = new Program();
    string password = "password";
    stopwatch.Start();
    Console.WriteLine("Şifrelenecek metni giriniz");
    string metin = Console.ReadLine();

    string crypto = pg.Encypt(metin, password);
    stopwatch.Stop();
    Console.WriteLine(crypto);
    string decrypto = pg.Descrypto(crypto, password);
    Console.WriteLine($"Çözülmüş metin:{decrypto}");
    Console.ReadLine();

}
public string Encypt(string message, string password)
```

ŞEKİL 3.9 C# DES ANA METODU

ŞEKİL 3.10 C# DES PROGRAM ÇIKTISI

3.1.2.2 Pyhton Programlama Dili ile Kodlanması

```
from Crypto.Cipher import DES
from secrets import token_bytes
from stopwatch import Stopwatch
from memory_profiler import profile
import cProfile

key=token_bytes(8) #64 bit ile şifreleme yapılacak
stopwatch=Stopwatch()
#@profile(precision=6)
def encrypt(msg):#Encrypto metodunun tanımlanması
    cipher=DES.new(key,DES.MODE_EAX)#DES.MODE_EAX metodunu sectik
    nonce=cipher.nonce
    ciphertext,tag=cipher.encrypt_and_digest(msg.encode('ascii'))
    return nonce,ciphertext,tag
```

ŞEKİL 3.11 PYHTON DES ENCRYPT METODU

```
def decrypt(nonce,ciphertext,tag):
    cipher=DES.new(key,DES.MODE_EAX,nonce=nonce)
    plaintext=cipher.decrypt(ciphertext)
    try:
        cipher.verify(tag)
        return plaintext.decode('ascii')
    except:
        return False
stopwatch.start()
nonce,ciphertext,tag=encrypt(input('Enter a message'))
plaintext=decrypt(nonce,ciphertext,tag)
print(f'CipherText:{ciphertext}')
print(f'PlainText:{plaintext}')
```

ŞEKİL 3.12 PYHTON DES DECRYPT METODU

```
Enter a messageBu metin DES ile sifrelenmistir
CipherText:b'm\xf0Dj_\xc0\x82d\xd1\x83l\xd0\xe4\xd8\xc7\xb1\x0b\xa6`\nYK\x00o[\x85\x9c\x98n\xee\xd0'
PlainText:Bu metin DES ile sifrelenmistir
```

ŞEKİL 3.13 PYHTON DES PROGRAM ÇIKTISI

3.1.3 SHA Şifreleme Algoritmasının Uygulanması

3.1.3.1 C# Programlama Dili ile Kodlanması

```
static string ComputeSha256Hash(string rawData)
{
    // Create a SHA384
    using (SHA384 sha256Hash = SHA384.Create())//SHA384 Kütüphanesinden bir metot oluşturduk.
    {
        // ComputeHash - bayt dizisini döndürür
        byte[] bytes = sha256Hash.ComputeHash(Encoding.UTF8.GetBytes(rawData));

        // Bayt dizisini bir dizgeye dönüştür
        StringBuilder builder = new StringBuilder();
        for (int i = 0; i < bytes.Length; i++)
        {
            builder.Append(bytes[i].ToString("x2"));
        }
        return builder.ToString();
    }
```

SEKİL 3.14 C# SHA HASH FONKSİYONUN OLUSTURULMASI

```
static void Main(string[] args)
{
    Stopwatch stop = new Stopwatch();
    stop.Start();
    Console.WriteLine("Metni giriniz");
    string plainData = Console.ReadLine();
    Console.WriteLine("Raw data: {0}", plainData);
    string hashedData = ComputeSha256Hash(plainData);
    Console.WriteLine("Hash {0}", hashedData);
    stop.Stop();
    Console.WriteLine(stop.Elapsed);
    Console.ReadLine();
}
```

SEKİL 3.15 C# SHA ANA METODU

```
Metni giriniz
Bu metin SHA384 ile sifrenmistir
Hash dccb2ba97560fb4051a12d4934eeafce8c631947fe983bc96f1cc558ad686bb242438a4fd7e7ecc7e2543594cf29aa7f
Raw data: Bu metin SHA384 ile sifrenmistir
```

SEKİL 3.16 SHA PROGRAMININ ÇIKTISI

3.1.4 TRİPLE DES Şifreleme Algoritmasının Uygulanması

3.1.4.1 C# Programlama Dili ile Kodlanması

```
public class ClsTripleDES

    private const string mysecurityKey = "MyTestSampleKey";

    public static string Encrypt(string TextToEncrypt)
    {
        byte[] MyEncryptedArray = UTF8Encoding.UTF8
           .GetBytes(TextToEncrypt);

        MD5CryptoServiceProvider MyMD5CryptoService = new
           MD5CryptoServiceProvider();

        byte[] MysecurityKeyArray = MyMD5CryptoService.ComputeHash
           (UTF8Encoding.UTF8.GetBytes(mysecurityKey));

        MyMD5CryptoService.Clear();

        var MyTripleDESCryptoService = new
           TripleDESCryptoServiceProvider();

        MyTripleDESCryptoService.Key = MysecurityKeyArray;

        MyTripleDESCryptoService.Mode = CipherMode.ECB;

        MyTripleDESCryptoService.Padding = PaddingMode.PKCS7;

        var MyCrytpoTransform = MyTripleDESCryptoService
           .CreateEncryptor();

        byte[] MyresultArray = MyCrytpoTransform
           .TransformFinalBlock(MyEncryptedArray, 0,
           MyEncryptedArray.Length);

        MyTripleDESCryptoService.Clear();

        return Convert.ToBase64String(MyresultArray, 0
```

SEKİL 3.17 3DES ENCRYPT METODU

```
public static string Decrypt(string TextToDecrypt)
{
    byte[] MyDecryptArray = Convert.FromBase64String
       (TextToDecrypt);

    MD5CryptoServiceProvider MyMD5CryptoService = new
       MD5CryptoServiceProvider();

    byte[] MysecurityKeyArray = MyMD5CryptoService.ComputeHash
       (UTF8Encoding.UTF8.GetBytes(mysecurityKey));

    MyMD5CryptoService.Clear();

    var MyTripleDESCryptoService = new
       TripleDESCryptoServiceProvider();

    MyTripleDESCryptoService.Key = MysecurityKeyArray;

    MyTripleDESCryptoService.Mode = CipherMode.ECB;

    MyTripleDESCryptoService.Padding = PaddingMode.PKCS7;

    var MyCrytpoTransform = MyTripleDESCryptoService
       .CreateDecryptor();

    byte[] MyresultArray = MyCrytpoTransform
       .TransformFinalBlock(MyDecryptArray, 0,
       MyDecryptArray.Length);

    MyTripleDESCryptoService.Clear();

    return UTF8Encoding.UTF8.GetString(MyresultArray);
```

SEKİL 3.18 3DES DECRYPTO METODU

```
static void Main(string[] args)
{
    Stopwatch stop = new Stopwatch();
    stop.Start();
    Console.WriteLine("Metini Giriniz");
    var text = Console.ReadLine();
    var encryptedText = ClsTripleDES.Encrypt(text);
    var decryptedText = ClsTripleDES.Decrypt(encryptedText);
    Console.WriteLine("Şifrelenmeden önceki metin: = " + text);
    Console.WriteLine("Şifrelenmiş metin = " +
       encryptedText);
    Console.WriteLine("Şifresi Çözülmüş metin = " +
       decryptedText);
    stop.Stop();
    Console.WriteLine("Çalışma zamanı" + stop.Elapsed);
    Console.ReadLine();

}
```

SEKİL 3.19 3DES ANA METODU

```
C:\Users\90537\source\repos\TripleDes\TripleDes\bin\Debug\TripleDes.exe
Metini Giriniz
Bu metin TripleDes ile şifrelenmiştir
Şifrelenmiş metin = 8efCoICwNqNTSJo17M8y2XkcAe6UCv4ovnzDzIQcizrkaQBn29H+gQ==
Şifresi Çözülmüş metin = Bu metin TripleDes ile şifrelenmiştir
```

SEKİL 3.20 3DES PROGRAM ÇIKTISI

3.1.4.2 Pyhton Programlama Dili ile Kodlanması

```
from secrets import token_bytes
from stopwatch import Stopwatch
from memory_profiler import profile
stopwatch=Stopwatch()


key=DES3.adjust_key_parity(token_bytes(24))

#@profile(precision=6)
def encrypt(msg):
        cipher=DES3.new(key,DES3.MODE_EAX)
        nonce=cipher.nonce
        ciphertext=cipher.encrypt(msg.encode('ascii'))
        return nonce,ciphertext
```

SEKİL 3.21 3DES ENCRYPT METODU

```python
#@profile(precision=6)
def decrypt(nonce,ciphertext):
        cipher=DES3.new(key,DES3.MODE_EAX,nonce=nonce)
        plaintext=cipher.decrypt(ciphertext)
        return plaintext.decode('ascii')
stopwatch.start()
nonce,ciphertext=encrypt((input('Mesajı giriniz')))
plaintext=decrypt(nonce,ciphertext)
print(f'Şifrelemiş Hali:{ciphertext}')
print(f'Plain text:{plaintext}')
stopwatch.stop()
#print(stopwatch.duration)
```

SEKİL 3.22 3DES DECRYPT METODU

```
C:\Users\98537\PycharmProjects\TripleDes\venv\Scripts\python.exe C:/Users/98537/PycharmProjects/TripleDes/main.py
Mesajı giriniz:Bu metin TripleDes ile sifrelenmistir
Şifrelemiş Hali:b'\xa2q1\xfb\xf9\x12\x15T\xb5\x1f\xc0\xbe\x84\xe1\xa6\xe2\x169\xf5\x96\x8e\xa0\xa9{\xfc\x9a\xfa\x80\xaa\xb2\xb3z\xd9\x94\xeebr'
Plain text:Bu metin TripleDes ile sifrelenmistir
```

SEKİL 3.23 3DES PROGRAMIN ÇIKTISI

3.1.5 MD5 Şifreleme Algoritmasının Uygulanması

3.1.5.1 C# Programlama Dili ile Kodlanması

```
public static string MD5Sifrele(string sifrelenecekMetin)
{

    // MD5CryptoServiceProvider sınıfının bir örneğini oluşturduk.
    MD5CryptoServiceProvider md5 = new MD5CryptoServiceProvider();
    //Parametre olarak gelen veriyi byte dizisine dönüştürdük.
    byte[] dizi = Encoding.UTF8.GetBytes(sifrelenecekMetin);
    //dizinin hash'ini hesaplattık.
    dizi = md5.ComputeHash(dizi);
    //Hashlenmiş verileri depolamak için StringBuilder nesnesi oluşturduk.
    StringBuilder sb = new StringBuilder();
    //Her byte'i dizi içerisinden alarak string türüne dönüştürdük.

    foreach (byte ba in dizi)
    {
        sb.Append(ba.ToString("x2").ToLower());
    }

    //hexadecimal(onaltılık) stringi geri döndürdük.
    return sb.ToString();
}
```

SEKİL 3.25 MD5 ENCRYPT

```
static void Main(string[] args)
{
    Stopwatch stopwatch = new Stopwatch();
    stopwatch.Start();
    Console.WriteLine("Şifrelenecek metni giriniz");
    string text = Console.ReadLine();
    string sifre = MD5Sifrele("Şifrelenmiş Metin"+text);
    Console.WriteLine(sifre);
    stopwatch.Stop();
    Console.WriteLine("Çalışma Zamanı" + stopwatch.Elapsed);
    Console.ReadKey();
}
```

SEKİL 3.26 MD5 ANA METODU

```
Şifrelenecek metni giriniz
Bu metin Md5 ile sifrelenmistir
e4034f40be402cb76999af3b69fc693f
```

SEKİL 3.26 MD5 PROGRAM ÇIKTISI

3.1.5.2 Pyhton Programlama Dili ile Kodlanması

```
import cProfile
from memory_profiler import profile
from stopwatch import Stopwatch
from hashlib import md5

a=input("Metin giriniz:")
class MD5:

   def __init__(self, data=a):
      self.data = data

   #@profile(precision=6)
   def encrypt(self):
      self.data = md5(self.data.encode()).hexdigest()
      return "Crypted: " + self.data
```

SEKİL 3.27 MD5 ENCRYPT METODU

```
   #@profile(precision=6)
   def decrypt(self, data):
      if md5(data.encode()).hexdigest() == self.data:
         return "Decrypted: " + data
         del self.data
      else:
         return "Error"


crypt = MD5()
print(crypt.encrypt())  # Encrypt
print(crypt.decrypt(data=a))  # Decrypt data argument
```

SEKİL 3.28 MD5 DECRYPT METODU

3.2 Asimetrik Şifreleme Algoritmalarının Uygulanması

3.2.1 RSA Şifreleme Algoritması

3.2.1.1 C# Programlama Dili ile Kodlanması

```
public class RsaEncryption
{
    public static RSACryptoServiceProvider csp = new RSACryptoServiceProvider(512);
    public RSAParameters _privateKey;
    public RSAParameters _publicKey;
    public RsaEncryption()
    {
        _privateKey = csp.ExportParameters(true);
        _publicKey = csp.ExportParameters(false);
    }
```

SEKİL 3.29 RSA RSAENCRYPTION METODU

```
public string GetPublicKey()
{
    var sw = new StringWriter();
    var xs = new XmlSerializer(typeof(RSAParameters));
   xs.Serialize(sw, _publicKey);
    return sw.ToString();
}
```

SEKİL 3.30 RSA GETPUBLİCKEY METODU

```
public string Encrypto(string plainText)
{
    csp = new RSACryptoServiceProvider();
    csp.ImportParameters(_publicKey);
    var data = Encoding.Unicode.GetBytes(plainText);
    var cypher = csp.Encrypt(data, false);
    return Convert.ToBase64String(cypher);


}
```

Sekil 3.31 Rsa Encrypto Metodu

```
public string Decrypto(string chperText)
{
    var dataBytes = Convert.FromBase64String(chperText);
    csp.ImportParameters(_privateKey);
    var plainText = csp.Decrypt(dataBytes, false);
    return Encoding.Unicode.GetString(plainText);
}
```

Sekil 3.32 Rsa Encrypto Metodu

```
static void Main(string[] args)
{
    RsaEncryption rsa = new RsaEncryption ();
    string cypher = string.Empty;
    Console.WriteLine($"Public Key: {rsa.GetPublicKey()} \n");
    Stopwatch stopwatch = new Stopwatch();
    Console.WriteLine("Metini giriniz:");
    var text = Console.ReadLine();
    stopwatch.Start();
    cypher = rsa.Encrypto(text);
    stopwatch.Stop();
    Console.WriteLine($"Şifrelenmiş Metin:{cypher}");
    var plaintext = rsa.Decrypto(cypher);
    stopwatch.Stop();
    Console.WriteLine("Çözülmüş Hali:"+plaintext+ "\n Çalışma Zamanı"+ stopwatch.Elapsed);


    Console.ReadLine();


}
```

Sekil 3.33 Rsa Main Metodu

```
C:\Users\90537\source\repos\RSA\RSA\bin\Debug\RSA.exe
Public Key: <?xml version="1.0" encoding="utf-16"?>
<RSAParameters xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <Exponent>AQAB</Exponent>
  <Modulus>r9azIAkusqYncAY8GKluDNrjncbBBycNCwzTc1hrhS7sScZexUHrCEbxQgyrakgDKxJ4tiU1Ri3CnygSRowz3Q==</Modulus>
</RSAParameters>

Metini giriniz:
Rsa sifreleme algoritması
Şifrelenmiş Metin:SD1t4e1j8YDCqsCu5SO4WHPLrZuxuKgsx4Uc9LH8y59qtHy/3OtXyQaM+imyHJI2YxlcMRQwybJuWyj25NxUWw==
Çözülmüş Hali:Rsa sifreleme algoritması
 Çalışma Zamanı00:00:00.0003142
```

Sekil 3.33 Rsa Program Çiktisi Metodu

4. UYGULAMA SONUÇLARI

Bu bölümde şifreleme algoritmaları karşılaştırılırken zaman, bellek kullanımı, yazılım metrikleri kullanılacaktır.

4.1.1 AES Şifreleme Algoritmasının Uygulama Sonuçları

4.1.1.1 C# Programlama Dili ile Kodlanmasının Sonuçları

1.Bakım Dizini: Kodu korumanın göreli kolaylığını temsil eden 0 ile 100 arasında bir dizin değeri hesaplar. Yüksek bir değer daha iyi bakım anlamına gelir. Renk kodlu derecelendirmeler, kodunuzdaki sorun noktalarını hızla tanımlamak için kullanılabilir. Yeşil derecelendirme 20 ile 100 arasındadır ve kodun iyi bakım olanağına sahip olduğunu gösterir. Sarı derecelendirme 10 ile 19 arasındadır ve kodun orta düzeyde korunabilir olduğunu gösterir. Kırmızı derecelendirme, 0 ile 9 arasında bir derecelendirmedir ve düşük bakım olanağını gösterir.

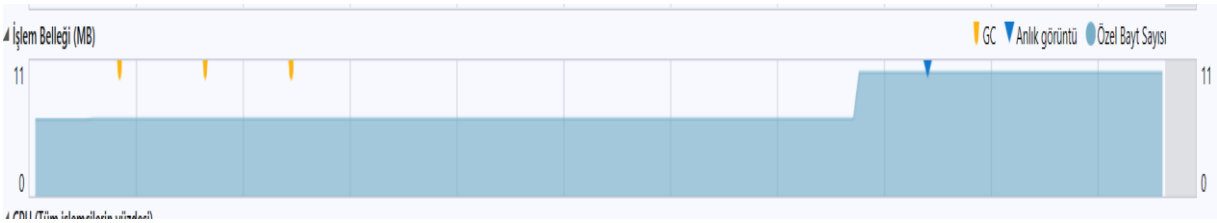
2.Döngüsel Karmaşıklık(CC):Kodun yapısal karmaşıklığını ölçer. Programın akışındaki farklı kod yollarının sayısı hesaplanarak oluşturulur. Karmaşık denetim akışına sahip bir program, iyi kod kapsamı elde etmek için daha fazla test gerektirir ve daha az bakım yapılabilir.

3.Kaynak Kod Satırları(SLOC):Boş satırlar dahil olmak üzere kaynak dosyanızda bulunan kaynak kod satırlarının tam sayısını gösterir.

4.Yürütebilir Kod Satırları(PLOC):Yürütülebilir kod satırlarının veya işlemlerinin yaklaşık sayısını gösterir. Bu, yürütülebilir koddaki işlem sayısıdır.

Hiyerarşi	Bakım Dizini	Döngüsel Karmaşıklık	Devralma Derinliği	Sınıf Bağlantısı	Kaynak kodu satırları	Yürütülebilir kod satırları
AES (Debug)	64	4	1	14	101	34
AES	64	4	1	14	101	34
Program	64	4	1	14	98	34
Main() : void	71	1		2	13	6
EncryptAesManaged(string) : void	58	1		7	33	14
Encrypt(string, byte[], byte[]) : byte[]	66	1		7	26	7
Decrypt(byte[], byte[], byte[]) : string	66	1		7	22	7

ŞEKİL 4.1 AES PROGRAMININ YAZILIM METRİKLERİ



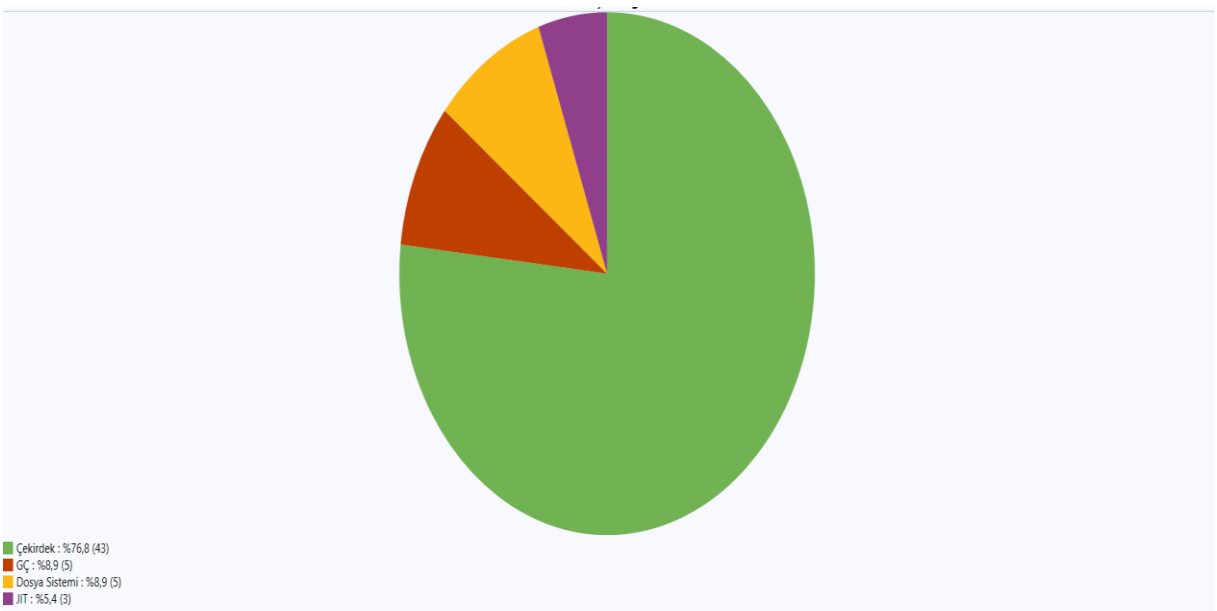
ŞEKİL 4.2 AES PROGRAMININ KULLANDIĞI BELLEK MİKTARI(MB)

İşlev Adı	Toplam CPU [birim, %]	İç CPU [birim, %]
[Yerel]	56 (%100,00)	37 (%66,07)
AES.Program.Main()	18 (%32,14)	9 (%16,07)
[Harici Kod]	8 (%14,29)	8 (%14,29)
AES.Program.EncryptAesManaged(string)	5 (%8,93)	2 (%3,57)

Etkin Yol

İşlev Adı	Toplam CPU [birim, %]	İç CPU [birim, %]
[Yerel]	56 (%100,00)	37 (%66,07)
AES.Program.Main()	18 (%32,14)	9 (%16,07)

ŞEKİL 4.3 AES PROGRAMININ KULLANDIĞI CPU MİKTARI



ŞEKİL 4.4 AES PROGRAMININ KULLANDIĞI CPU'NUN YÜZDESEL OLARAK HANGİ İŞLEMLERDE KULLANILDIĞI

imlik	Zaman	Nesneler (Fark)	Yığın Boyutu (Fark)
1	21,00s	380 (yok)	50,44 KB (yok)

ŞEKİL 4.5 AES PROGRAMININ ÇALIŞMA ZAMANI

4.1.1.2 Pyhton Programlama Dili ile Kodlanmasının Sonuçları

```
Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
    14  22.585938 MiB  22.585938 MiB           1   @profile(precision=6)
    15                                         def encrypt(msg):
    16  22.589844 MiB   0.003906 MiB           1       stopwatch.start()
    17  22.621094 MiB   0.031250 MiB           1       cipher=AES.new(key,AES.MODE_EAX)
    18  22.621094 MiB   0.000000 MiB           1       nonce=cipher.nonce
    19  22.632812 MiB   0.011719 MiB           1       ciphertext,tag=cipher.encrypt_and_digest(msg.encode('ascii'))
    20
    21  22.632812 MiB   0.000000 MiB           1       return nonce,ciphertext,tag
    22                                             stopwatch.stop()
```

ŞEKİL 4.6 PYHTON AES ENCRYPT METODUNUN BELLEK KULLANIMI

```
Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
    23  22.636719 MiB  22.636719 MiB           1   @profile(precision=6)
    24                                         def decrpt(nonce,ciphertext,tag):
    25  22.636719 MiB   0.000000 MiB           1       cipher=AES.new(key,AES.MODE_EAX,nonce=nonce)
    26  22.636719 MiB   0.000000 MiB           1       plaintext=cipher.decrypt(ciphertext)
    27  22.636719 MiB   0.000000 MiB           1       try:
    28  22.640625 MiB   0.003906 MiB           1           cipher.verify(tag)
    29  22.640625 MiB   0.000000 MiB           1           return plaintext.decode("ascii")
    30                                             except:
    31                                                 return False
```

ŞEKİL 4.7 PYHTON AES DECRPT METODUNUN BELLEK KULLANIMI

```
Şifreli hali b'{\x8c\x04\x11\xb9\x89\xad\xe2\xd0"U50\xe7m6(\xe0E\xf0\xa5U\x18E\xd1V\x838hY\xc0'
Gecen süre(sn): 17.523309899999997
```

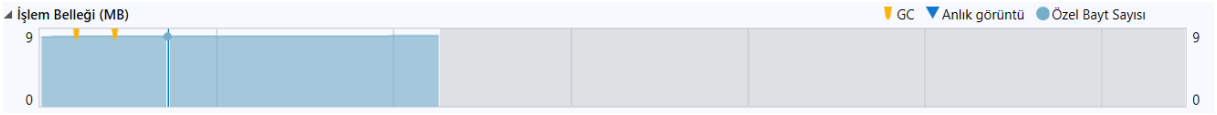
ŞEKİL 4.7 PYHTON AES UYGULAMASININ ÇALIŞMA ZAMANI

4.1.2 DES Şifreleme Algoritmasının Uygulama Sonuçları

4.1.2.1 C# Programlama Dili ile Kodlanmasının Sonuçları

Hiyerarşi	Bakım Dizini	Döngüsel Karmaşıklık	Devralma Derinliği	Sınıf Bağlantısı	Kaynak kodu satırları	Yürütülebilir kod satırları
Des (Debug)	58	3	1	11	65	41
{} Des	58	3	1	11	65	41
Program	58	3	1	11	62	41
Main(string[]) : void	60	1		3	18	13
Encypt(string, string) : string	57	1		8	20	14
Descrypto(string, string) : string	57	1		8	21	14

ŞEKİL 4.7 DES PROGRAMININ YAZILIM METRİKLERİ



ŞEKİL 4.7 DES PROGRAMININ KULLANDIĞI BELLEK ALANI

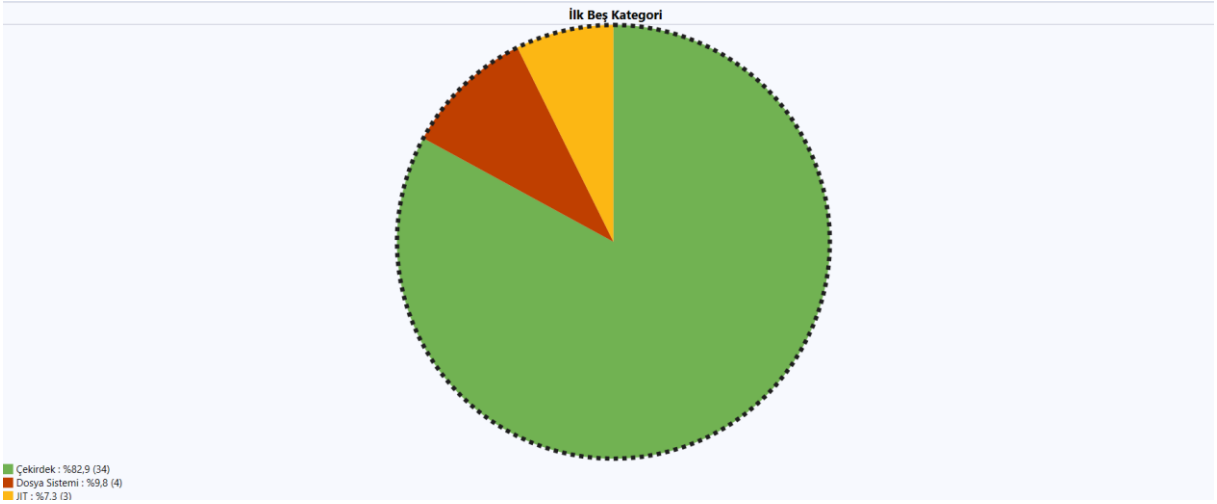
En Çok Kullanılan İşlevler

İşlev Adı	Toplam CPU [birim, %]	İç CPU [birim, %]
[Yerel]	40 (%97,56)	35 (%85,37)
[Harici Kod]	4 (%9,76)	4 (%9,76)
Des.Program.Main(System.String[])	5 (%12,20)	1 (%2,44)
[Ulaşılamaz]	1 (%2,44)	0 (%0,00)
Des.Program.Encypt(string, string)	3 (%7,32)	0 (%0,00)

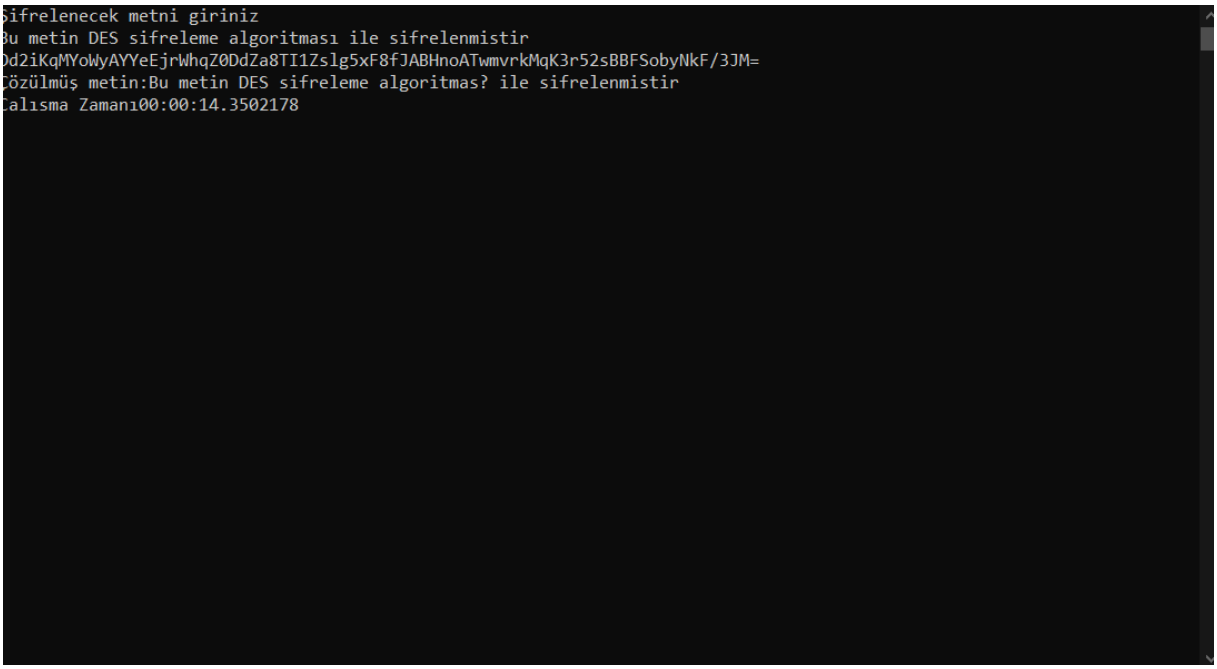
Etkin Yol

İşlev Adı	Toplam CPU [birim, %]	İç CPU [birim, %]
[Yerel]	40 (%97,56)	35 (%85,37)

ŞEKİL 4.8 DES PROGRAMININ KULLANDIĞI CPU MİKTARI



ŞEKİL 4.9 DES PROGRAMININ KULLANDIĞI CPU'NUN YÜZDESEL OLARAK HANGİ İŞLEMLERDE KULLANILDIĞI



ŞEKİL 4.9 DES PROGRAMININ CALISMA SÜRESİ

4.1.2.2 Pyhton Programlama Dili ile Kodlanmasının Sonuçları

Line #	Mem usage	Increment	Occurrences	Line Contents
9	23.332031 MiB	23.332031 MiB	1	@profile(precision=6)
10				def encrypt(msg):#Encrypto metodunun tanımlanması
11	23.410156 MiB	0.078125 MiB	1	cipher=DES.new(key,DES.MODE_EAX)#DES.MODE_EAX metodunu sectik
12	23.410156 MiB	0.000000 MiB	1	nonce=cipher.nonce
13	23.425781 MiB	0.015625 MiB	1	ciphertext,tag=cipher.encrypt_and_digest(msg.encode('ascii'))
14	23.425781 MiB	0.000000 MiB	1	return nonce,ciphertext,tag

ŞEKİL 4.10 PYHTON DES ENCRYPT METODUNUN BELLEK KULLANIMI

Line #	Mem usage	Increment	Occurrences	Line Contents
15	23.441406 MiB	23.441406 MiB	1	@profile(precision=6)
16				def decrypt(nonce,ciphertext,tag):
17	23.468750 MiB	0.027344 MiB	1	cipher=DES.new(key,DES.MODE_EAX,nonce=nonce)
18	23.472656 MiB	0.003906 MiB	1	plaintext=cipher.decrypt(ciphertext)
19	23.472656 MiB	0.000000 MiB	1	try:
20	23.539062 MiB	0.066406 MiB	1	cipher.verify(tag)
21	23.539062 MiB	0.000000 MiB	1	return plaintext.decode('ascii')
22				except:
23				return False

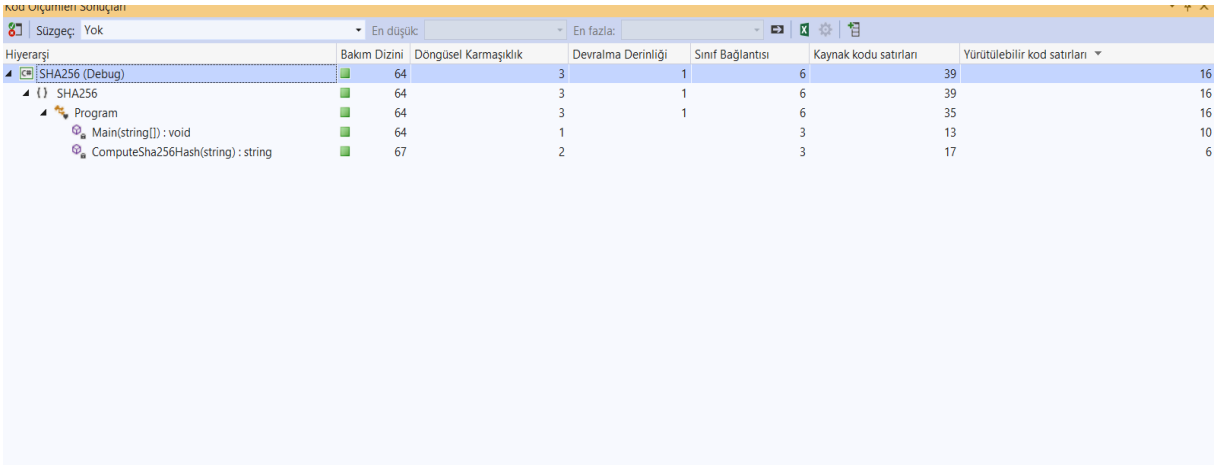
ŞEKİL 4.11 PYHTON DES DECRYPT METODUNUN BELLEK KULLANIMI

```
CipherText:b'\x05=\xe3`}\xc6%\x91\xba\x8f{\x0e\x16\r\x0c54\x96<\xee\xae\x19\\r\x84\x16\xd5\x04\x14\xbbW'
PlainText:Bu metin DES ile sifrelenmistir
12.077471
```

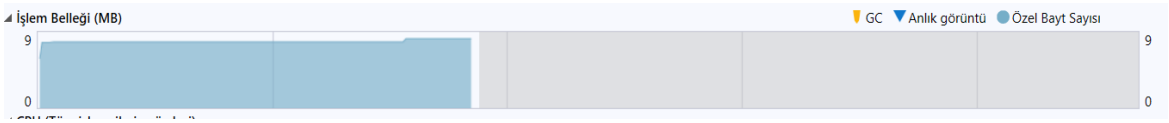
ŞEKİL 4.12 PYHTON DES UYGULAMASININ ÇALISMA ZAMANI

4.1.3 SHA Şifreleme Algoritmasının Uygulama Sonuçları

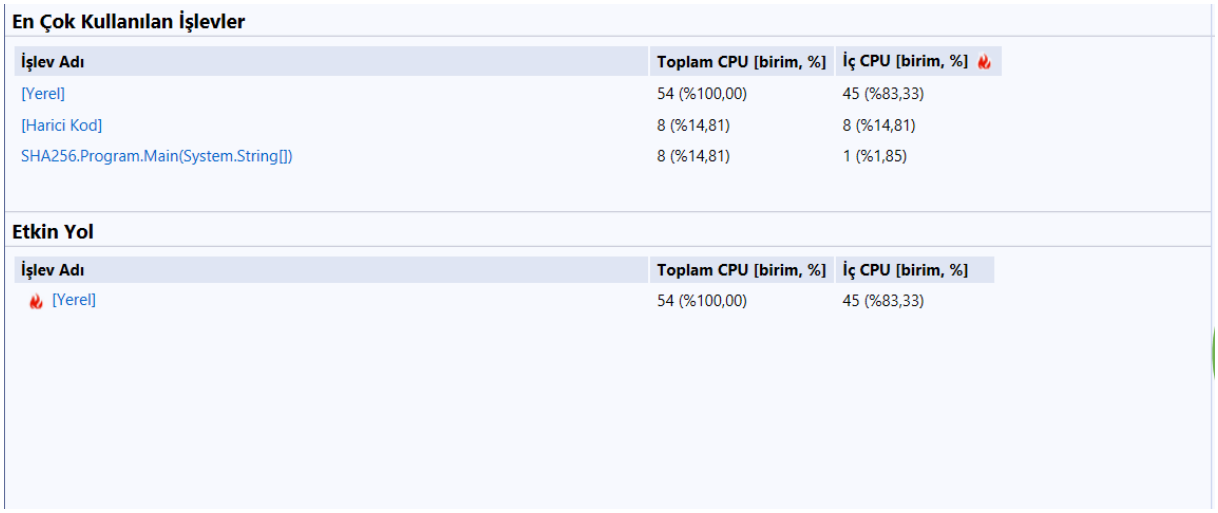
4.1.3.1 C# Programlama Dili ile Kodlanmasının Sonuçları



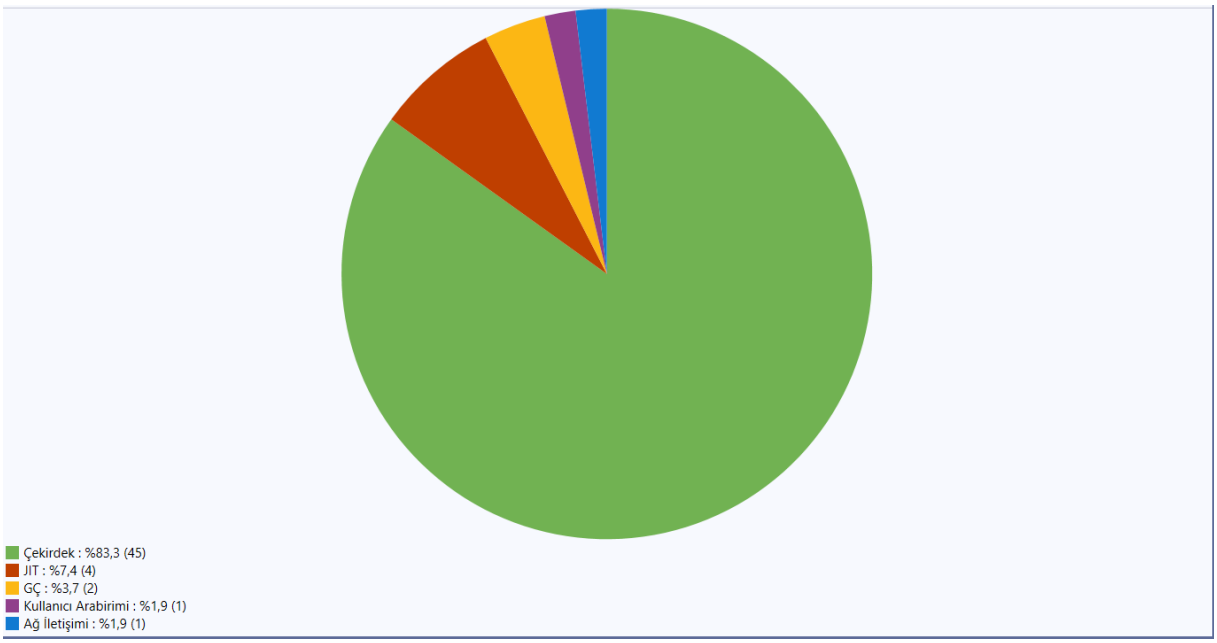
ŞEKİL 4.12 SHA PROGRAMININ YAZILIM METRİKLERİ



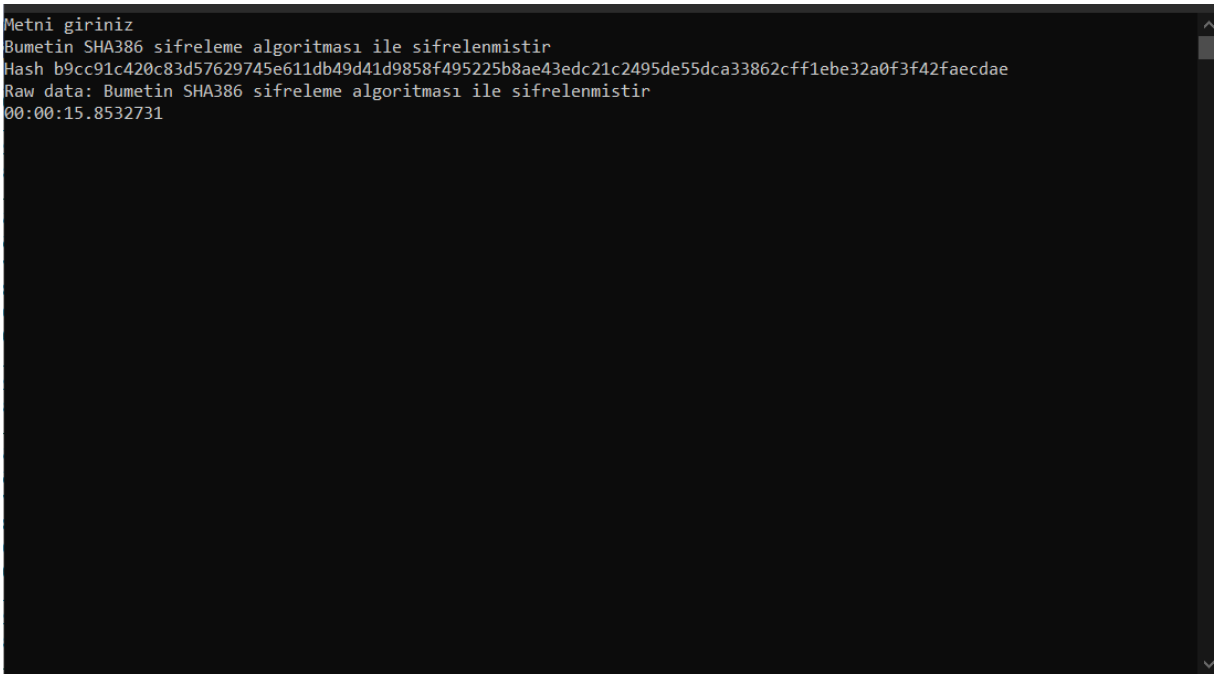
ŞEKİL 4.13 SHA PROGRAMININ BELLEK KULLANIMI



ŞEKİL 4.15 SHA PROGRAMININ CPU KULLANIMI



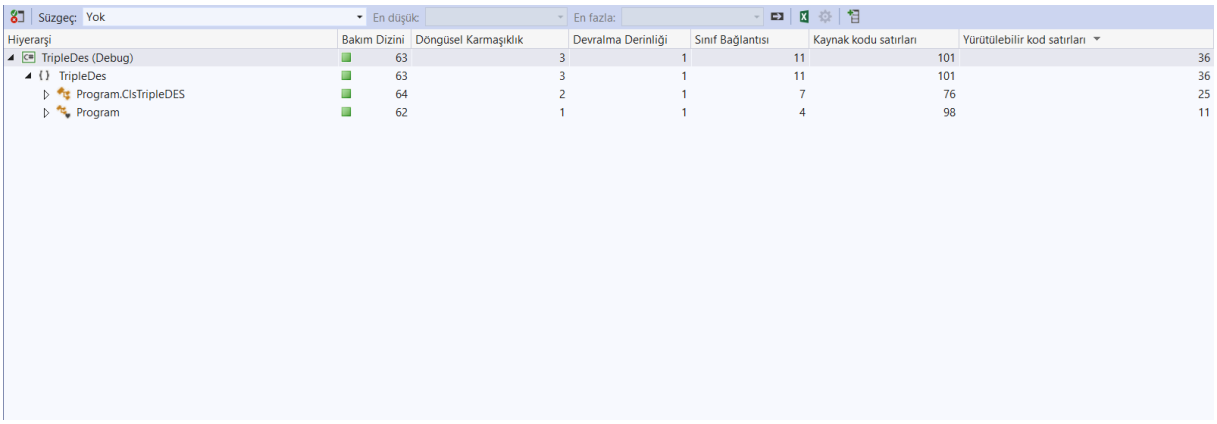
ŞEKİL 4.16 SHA PROGRAMININ KULLANDIĞI CPU'NUN YÜZDESEL OLARAK HANGİ İŞLEMLERDE KULLANILDIĞI



ŞEKİL 4.17 SHA PROGRAMININ ÇALIŞMA ZAMANI

4.1.4 TRİPLE DES Şifreleme Algoritmasının Uygulama Sonuçları

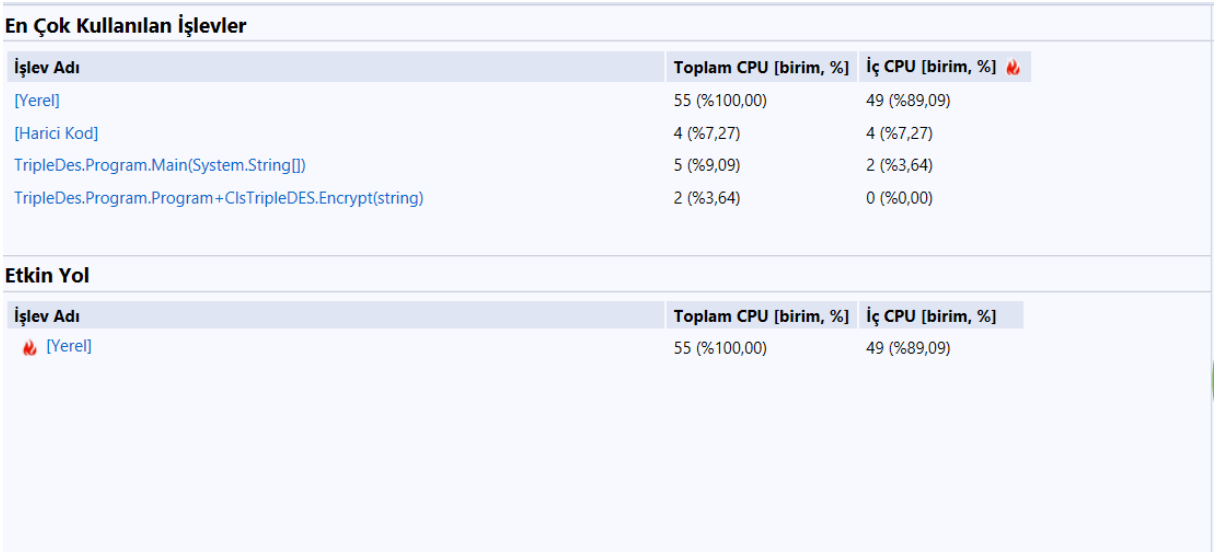
4.1.4.1 C# Programlama Dili ile Kodlanmasının Sonuçları


Süzgeç: Yok En düşük: En fazla:

Hiyerarşi	Bakım Dizini	Döngüsel Karmaşıklık	Devralma Derinliği	Sınıf Bağlantısı	Kaynak kodu satırları	Yürütülebilir kod satırları
TripleDes (Debug)	63	3	1	11	101	36
TripleDes	63	3	1	11	101	36
Program.ClsTripleDES	64	2	1	7	76	25
Program	62	1	1	4	98	11

ŞEKİL 4.18 TRİPLEDES PROGRAMININ YAZILIM METRİKLERİ

ŞEKİL 4.19 TRİPLEDES PROGRAMININ BELLEK KULLANIMI

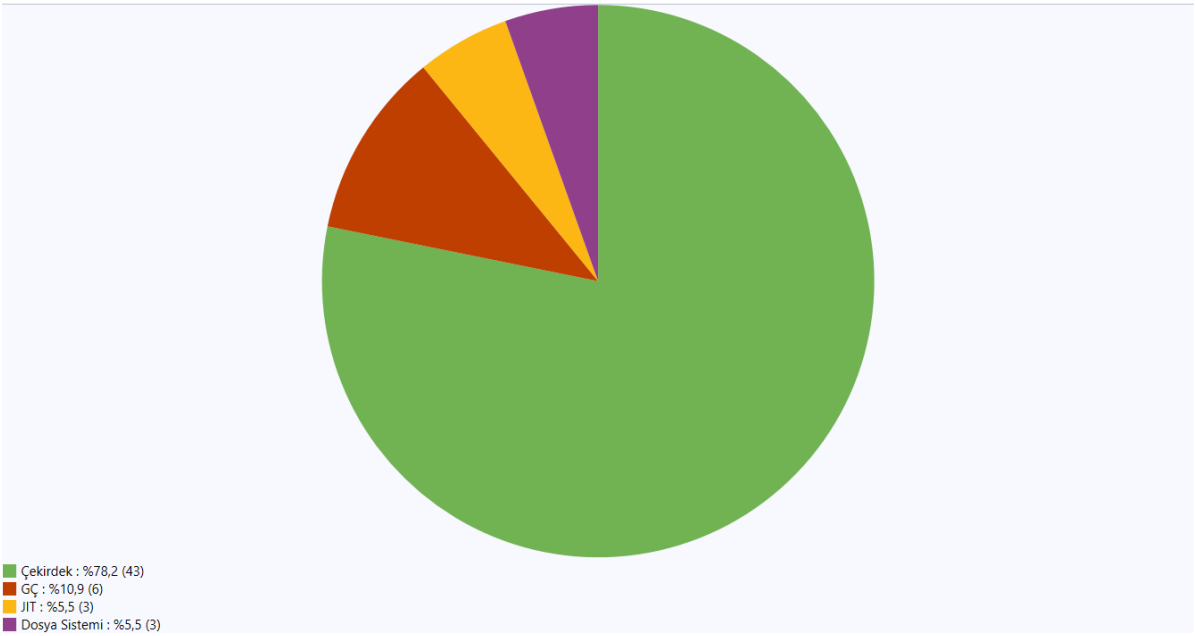

En Çok Kullanılan İşlevler

İşlev Adı	Toplam CPU [birim, %]	İç CPU [birim, %]
[Yerel]	55 (%100,00)	49 (%89,09)
[Harici Kod]	4 (%7,27)	4 (%7,27)
TripleDes.Program.Main(System.String[])	5 (%9,09)	2 (%3,64)
TripleDes.Program.Program+ClsTripleDES.Encrypt(string)	2 (%3,64)	0 (%0,00)

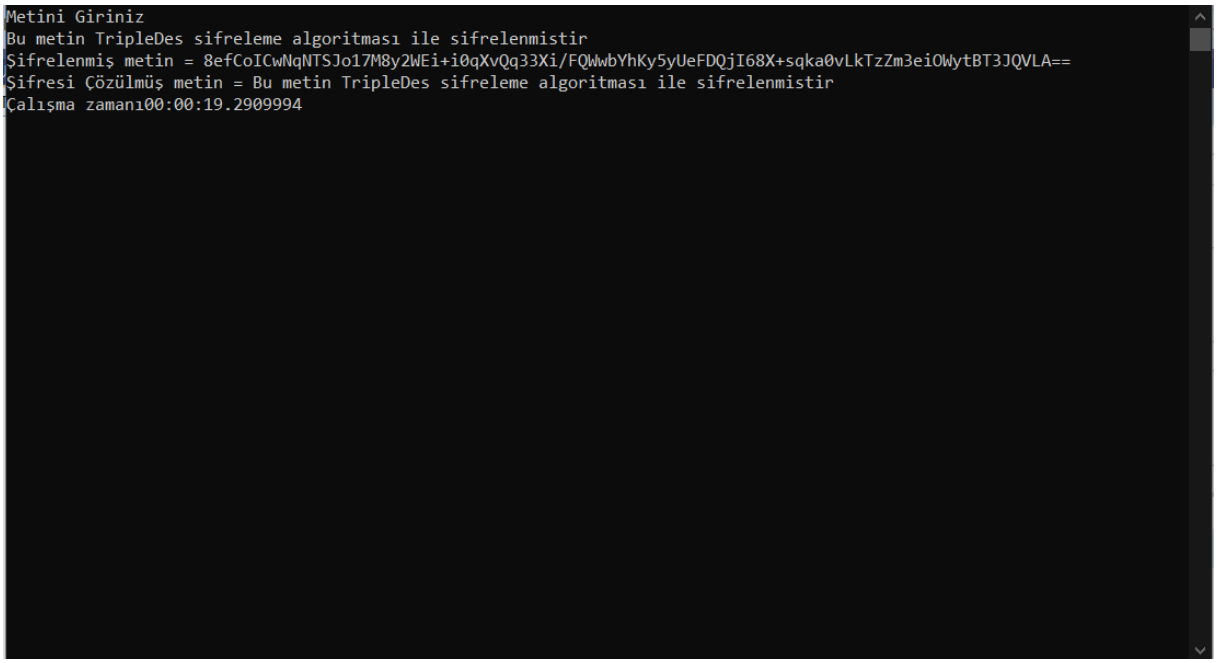
Etkin Yol

İşlev Adı	Toplam CPU [birim, %]	İç CPU [birim, %]
[Yerel]	55 (%100,00)	49 (%89,09)

ŞEKİL 4.20 TRİPLEDES PROGRAMININ CPU KULLANIMI



ŞEKİL 4.21 TRİPLEDES PROGRAMININ KULLANDIĞI CPU'NUN YÜZDESEL OLARAK HANGİ İŞLEMLERDE KULLANILDIĞI



ŞEKİL 4.22 TRİPLEDES PROGRAMININ ÇALISMA ZAMANI

4.1.4.2 Pyhton Programlama Dili ile Kodlanmasının Sonuçları

ŞEKİL 4.23 PYHTON TRİPLEDES ENCRYPT METODUNUN BELLEK KULLANIMI

```
Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
    16  23.421875 MiB  23.421875 MiB           1   @profile(precision=6)
    17                                         def decrypt(nonce,ciphertext):
    18  23.425781 MiB   0.003906 MiB           1           cipher=DES3.new(key,DES3.MODE_EAX,nonce=nonce)
    19  23.425781 MiB   0.000000 MiB           1           plaintext=cipher.decrypt(ciphertext)
    20  23.425781 MiB   0.000000 MiB           1           return plaintext.decode('ascii')
```

ŞEKİL 4.24 PYHTON TRİPLEDES DECRYPT METODUNUN BELLEK KULLANIMI

```
Şifrelemiş Hali:b'K\xb7Ub;\xb1\x8bZq\x84r\xae\xebh\xc5\xdbj\x02\xfd\x1a\x8cB\x94\xe6\x03l9_\x94\x91.\x15v\xcd\xcbr-\xc00\xe2\xbf\xa8\x131\x916\xe1S\xf4\xcf^\t\x01uY\xa2sW\xc3'
Plain text:Bu metin TripleDes sifreleme algoritmasi ile sifrelenmistir
18.8729482
```

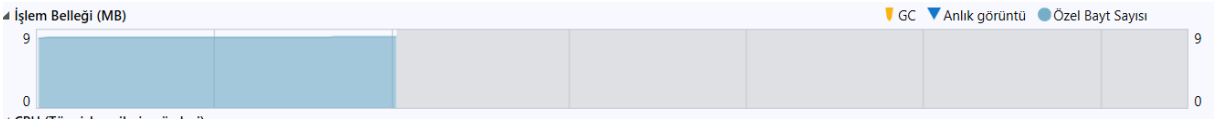
ŞEKİL 4.25 PYHTON TRİPLEDES ÇALIŞMA ZAMANI

4.1.5 MD5 Şifreleme Algoritmasının Uygulanması

4.1.5.1 C# Programlama Dili ile Kodlanmasının Sonuçları

Hiyerarşi	Bakım Di...	Döngüsel Karmaşıklık	Devralma Derinliği	Sınıf Bağlantısı	Kaynak kodu satırları	Yürütülebilir kod satırları
Md5.1 (Debug)	65	3	1	8	40	16
Md5_1	65	3	1	8	40	16
Program	65	3	1	8	37	16
Main(string[]) : void	65	1		4	12	9
MD5Sifrele(string) : string	66	2		4	21	7

ŞEKİL 4.26 MD5 PROGRAMININ YAZILIM METRİKLERİ



ŞEKİL 4.27 MD5 PROGRAMININ BELLEK KULLANIMI

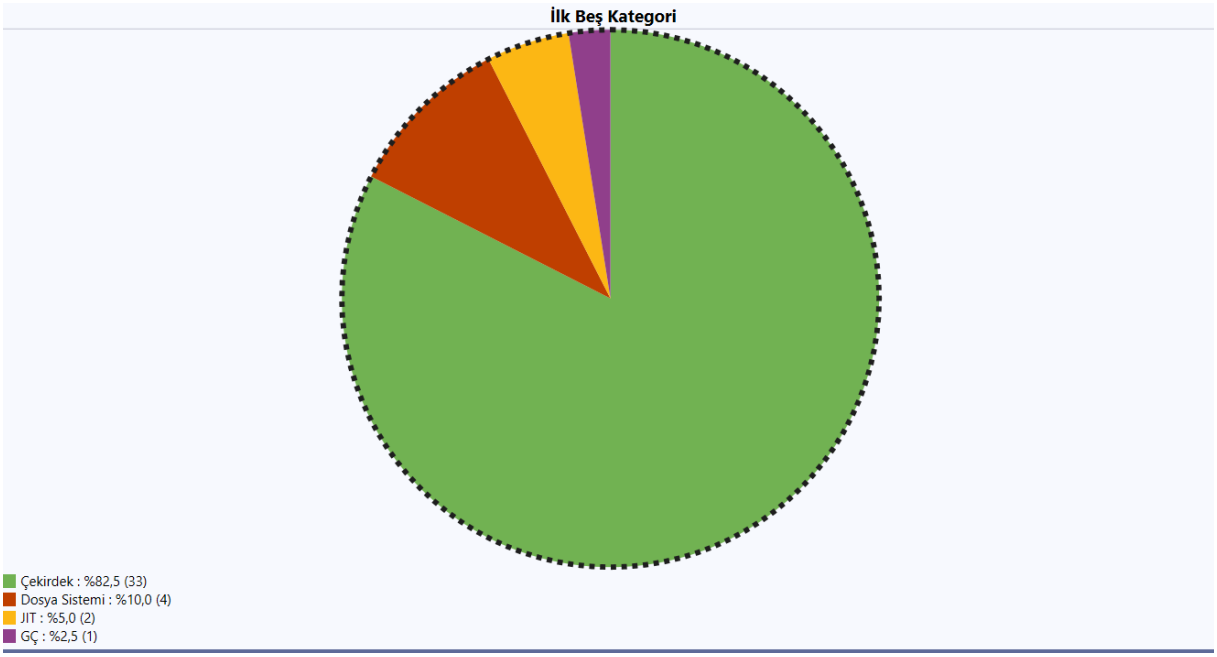
En Çok Kullanılan İşlevler

İşlev Adı	Toplam CPU [birim, %]	İç CPU [birim, %]
[Yerel]	40 (%100,00)	36 (%90,00)
[Harici Kod]	3 (%7,50)	3 (%7,50)
Md5._1.Program.Main(System.String[])	3 (%7,50)	1 (%2,50)

Etkin Yol

İşlev Adı	Toplam CPU [birim, %]	İç CPU [birim, %]
[Yerel]	40 (%100,00)	36 (%90,00)

ŞEKİL 4.28 MD5 PROGRAMININ CPU KULLANIMI



ŞEKİL 4.29 MD5 PROGRAMININ KULLANDIĞI CPU'NUN YÜZDESEL OLARAK HANGİ İŞLEMLERDE KULLANILDIĞI

4.1.5.2 Pyhton Programlama Dili ile Kodlanmasının Sonuçları

```
Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
    14  19.769531 MiB  19.769531 MiB           1      @profile(precision=6)
    15                                            def encrypt(self):
    16  19.792969 MiB   0.023438 MiB           1          self.data = md5(self.data.encode()).hexdigest()
    17  19.792969 MiB   0.000000 MiB           1          return "Crypted: " + self.data
```

ŞEKİL 4.30 PYHTON MD5 ENCRYPT METODUNUN BELLEK KULLANIMI

```
Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
    19  19.796875 MiB  19.796875 MiB           1      @profile(precision=6)
    20                                            def decrypt(self, data):
    21  19.796875 MiB   0.000000 MiB           1          if md5(data.encode()).hexdigest() == self.data:
    22  19.796875 MiB   0.000000 MiB           1              return "Decrypted: " + data
    23                                                    del self.data
    24                                                else:
    25                                                    return "Error"
```

ŞEKİL 4.31 PYHTON MD5 DECRYPT METODUNUN BELLEK KULLANIMI