

**KARADENİZ TEKNİK ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**



VERİ GÜVENLİĞİ

MÜHENDİSLİK TASARIMI

**Onur BUDAK
Batuhan EKİCİ
Hakan AKSOY**

2018-2019 GÜZ DÖNEMİ

**KARADENİZ TEKNİK ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

VERİ GÜVENLİĞİ

MÜHENDİSLİK TASARIMI

**Onur BUDAK
Batuhan EKİCİ
Hakan AKSOY**

2018-2019 GÜZ DÖNEMİ



IEEE Etik Kuralları IEEE Code of Ethics



Mesleğime karşı şahsi sorumluluğumu kabul ederek, hizmet ettiğim toplumlara ve üyelerine en yüksek etik ve mesleki davranışta bulunmaya söz verdiğimi ve aşağıdaki etik kurallarını kabul ettiğimi ifade ederim:

1. Kamu güvenliği, sağlığı ve refahı ile uyumlu kararlar vermenin sorumluluğunu kabul etmek ve kamu veya çevreyi tehdit edebilecek faktörleri derhal açıklamak;
2. Mümkün olabilecek çıkar çatışması, ister gerçekten var olması isterse sadece algı olması, durumlarından kaçınmak. Çıkar çatışması olması durumunda, etkilenen taraflara durumu bildirmek;
3. Mevcut verilere dayalı tahminlerde ve fikir beyan etmelerde gerçekçi ve dürüst olmak;
4. Her türlü rüşveti reddetmek;
5. Mütenasip uygulamalarını ve muhtemel sonuçlarını gözeterek teknoloji anlayışını geliştirmek;
6. Teknik yeterliliklerimizi sürdürmek ve geliştirmek, yeterli eğitim veya tecrübe olması veya işin zorluk sınırları ifade edilmesi durumunda ancak başkaları için teknolojik sorumlulukları üstlenmek;
7. Teknik bir çalışma hakkında yansız bir eleştiri için uğraşmak, eleştiriye kabul etmek ve eleştiriye yapmak; hatları kabul etmek ve düzeltmek; diğer katkı sunanların emeklerini ifade etmek;
8. Bütün kişilere adilane davranmak; ırk, din, cinsiyet, yaş, milliyet, cinsi tercih, cinsiyet kimliği, veya cinsiyet ifadesi üzerinden ayırimcılık yapma durumuna girişmemek;
9. Yanlış veya kötü amaçlı eylemler sonucu kimsenin yaralanması, mülklerinin zarar görmesi, itibarlarının veya istihdamlarının zedelenmesi durumlarının oluşmasından kaçınmak;
10. Meslektaşlara ve yardımcı personele mesleki gelişimlerinde yardımcı olmak ve onları desteklemek.

IEEE Yönetim Kurulu tarafından Ağustos 1990'da onaylanmıştır.

ÖNSÖZ

Günümüzdeki internet ortamında sıklıkla ihtiyaç duyulan SSL (Secure Secret Layer) içinde bulunan “Simetrik Şifreleme” alanında yaptığımız algoritma tasarımı tamamlayıp, bu yapının koda implementasyonunu gerçekleştirmeye çalıştık.

Öncelikle tez konusunu seçerken isteklerimizi göz önünde bulundurup bize yardımcı olan tez danışmanımız Doç. Dr. Güzin ULUTAŞ hocamıza teşekkürlerimizi sunarız.

Proje tasarımını gerçekleştirirken yardımlarını esirgemeyen TÜBİTAK TÜSSİDE çalışanları Ahmet Çakmak ve Öznur Kalkar’a, bu zorlu tez sürecinde bizden desteğini bir an için esirgemeyen değerli sınıf ve ev arkadaşımız Mehmet Alperen Çubuk’a, tüm eğitim hayatımız boyunca bizden maddi ve manevi desteklerini esirgemeyen her zaman yanımızda olan sevgili ailelerimize teşekkürlerimizi bir borç biliriz.

Onur BUDAK
Batuhan EKİCİ
Hakan AKSOY
Trabzon 2018

İÇİNDEKİLER

	Sayfa No
IEEE ETİK KURALLARI.....	II
ÖNSÖZ.....	III
İÇİNDEKİLER.....	IV
ÖZET.....	V
1. GENEL BİLGİLER.....	1
1.1. Giriş.....	1
1.2. SSL' e Giriş.....	2
1.2.1. SSL Nedir?	2
1.2.2. SSL Sertifikaları	3
1.2.3. SSL Bağlantısının Oluşması	4
1.3. Şifreleme Algoritmaları	5
1.3.1. Simetrik Şifreleme Algoritmaları	6
1.3.2. Şifrelemede Kullanılan Anahtar Boyutları	7
1.3.3. Sayısal İmza	7
1.3.4. Şifreleme Modları	8
1.4. Kullanılan Programlama Dilleri Ve Platformlar	9
1.5. İstemci – Sunucu (Client – Server) Mimarisi	10
2. PROJE TASARIMI.....	12
2.1. Gereksinim Analizi.....	12
2.2. Mimari Tasarım.....	13
2.3. Şifreleme Algoritmalarının UML Sıralı Diyagramı (Sequence Diagram)	16
2.4. Yapılan Çalışmalar.....	17
2.4.1. Simetrik Şifreleme Tasarımları	17
2.4.1.1. Simetrik Şifreleme Algoritması – 1	17
2.4.1.2 Simetrik Şifreleme Algoritması – 2	19
2.4.1.3 Simetrik Şifreleme Algoritması – 3	22
2.4.2. Kaba Kuvvet Saldırısı (Brute Force Attack)	24
2.4.3. SSL Protokolü	25
2.4.3.1. Handshake Protokol (El Sıkışma Protokolü).....	25
2.4.3.1.1. Handshake Protokol Aşamaları	26
2.4.3.2. Alert Protokolü (Uyarı Protokolü)	28
2.4.3.2.1. Kapanış Uyarıları (Closure Alerts)	28
2.4.3.2.2. Hata Uyarıları (Error Alerts)	28
2.4.3.3. Record Protokolü (Kayıt Protokolü)	29
2.4.4. Wireshark Testi	30
2.4.4.1. Wireshark'ın Özellikleri	30
3. KAYNAKLAR.....	31
STANDARTLAR ve KISITLAR FORMU.....	32

ÖZET

Proje kapsamında “Simetrik Şifreleme” algoritmalarının tasarımını gerçeklemeye çalışırken, birçok parametreye dikkat ederek algoritmada iyileştirmeler yaptık. Tasarımların görsel halini “flowgorithm” uygulamasını kullanarak gerçekleştirdik.

Daha sonra şifreleme algoritmalarının implementasyonu için Eclipse ve Visual Studio IDE’lerini kullandık. Burada gerçeklemeye çalışılan algoritmaların kodlama kısımlarını tamamladıktan sonra üretilen anahtarın uzunluğu ile bağlantılı olarak BruteForce işlemini gerçekledik.

Algoritmaların uygulamalara implementasyonundan sonra, bir istemci-sunucu uygulamasından şifreleme algoritmalarını test etmeye çalıştık. Bu süreçte ilk olarak açık metni şifreleyip gönderimi sağladık. Gönderilen şifreli metni “Wireshark” üzerinden test ettik. Daha sonra şifreli metne yaptığımız deşifreleme işlemiyle açık metni elde ettik.

Projenin ilerleyen kısmında SSL protokolünün özelliklerini barındıran bir yapının implementasyonu için protokol içinde bulunan alt katmanları detaylı bir şekilde inceledik. Bu süreçte hocamızın tavsiyesiyle “SSL & TLS Essentials - Securing the Web” kitabı üzerinden Handshake-Alert-Record protokolleriyle ilgili araştırmalarımızı tamamladık.

1. GENEL BİLGİLER

1.1. Giriş

Günümüzde veri güvenliği konusundaki yapılan çalışmaların gün geçtikçe daha çok arttığı görülmektedir. Daha sonra tasarım konusunu belirlemek için, bu konuda yapılan çalışmaları araştırmaya başlamıştık.

Bu süreçte bölümümüzde verilen “Şifreleme” dersine giderek konular hakkında bilgi edinmeye başladık. Ders konuları ilk önce Simetrik Şifreleme konusunda olması, bizi bu konuda tasarım konusu belirlemeye etti. Şifreleme konusunda kötü niyetli insanlara karşı savunulması gereken anahtarın iyi belirlenmesi gerekir. Kapalı metnin ve anahtarın frekans analizine karşı dayanıklılığı önemlidir.

Bu konuda yapılabilecek şifreleme yöntemleri; blok şifreleme yapısı (Block cipher – n 'ye n) ve Kelime şifreleme yapısı (Stream cipher – 1'e 1) başlıkları altında düşünülebilir. Blok şifrelemede anahtarın aynı olması gerektiği unutulmamalıdır. Bu yapıda giriş ile çıkış arasındaki ilişki kopmaktadır. Şifrelemede kullanılan döngüler bu özelliğe katkı sağlar.

Simetrik algoritma yapısı olarak DES (Data Encryption Standard) ve AES (Advanced Encryption Standard) örnek verilebilir. DES algoritması NIST (National Institute of Standards and Technology) tarafından yayınlanmış ve FIPS (Federal Information Processing Standards) standartıdır.

Şifreleme ve deşifreleme arasındaki fark, anahtarların veriliş sırasındır. AES yapısının kullanımı DES'ten farklı olarak tablo tutma işlemi yapmamaktadır. Bunun yerine GF (Galois Field) yapısını kullanmaktadır. Bu şifreleme lineerlik SubWord ile bozulabilirken, anahtarın döngü durumu buna karşılık vermektedir. Bu sebepten dolayı zayıf anahtar görülememektedir. Yani bir bitlik değişimin etkisi oldukça fazladır.

1.2. SSL' e Giriş

Netscape tarafından 1994 yılında SSL piyasaya sürüldü ve bir sonraki yıl IETF tarafından standart olarak kabul edildi. Aslında standardın asıl ismi TLS olmasına rağmen genellikle SSL kullanımı tercih edilmektedir. İlk zamanlar sadece HTTP trafiğini şifreleme amaçlı geliştirilmiş olsa da günümüzde TCP tabanlı tüm servisleri şifreleme amaçlı kullanılabilir.

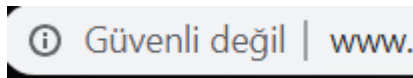
1.2.1. SSL Nedir?

Açılımı "**Secure Sockets Layer**" olan SSL Türkçe anlamı ise "**Güvenli Giriş Katmanı**" demektir. SSL, sunucu ile client arasındaki iletişimde verilerin şifrelenmesi işlemidir. En yaygın ve bilinen kullanımı ise web sitesindeki veri alışverişi sırasında Server ile internet tarayıcısı("browser") arasındaki iletişimi şifrelemesidir. Güvenli veri iletişimi için birçok web sitesi SSL teknolojisini kullanmaktadır. SSL bağlantısı üzerinden paylaşılan bilgiler sadece uygun anahtar değeri ile okunabilir. Dışarıdan yapılan müdahaleler ile bu bilgilere ulaşılsa bile, bu bilgiler şifreli olduklarından tamamen okunamaz halledirler.

SSL Teknolojisini kullanmayan sistemlerde 2 temel güvenlik problemi ortaya çıkabilir bunlar:

- 1) Doğru sunucuya bağlanıp bağlanmadığımızı bilememek.
- 2) İnternet üzerinden paylaşacağımız verilerin 3. kişiler tarafından ele geçirilip geçirilmediğini bilememek.

SSL in varlığını iki şekilde anlayabiliriz. Birincisi **http://**'nin **https://** olmasından anlıyoruz. İkincisi ise, url kısmında çıkan kilit simgesinden anlarız. SSL bağlantısı olduğunda tarayıcının url kısmında bir kilit simgesi çıkar. SSL bağlantısı olmadığı zaman ise bu kilit simgesi görünmez.



(SSL kullanmayan sistem)



(SSL kullanan sistem)

HTTPS, HTTP'nin iletim (Transport) katmanının güvenli hale getirilerek gizliliğin sağlanması ve dijital sertifikalarla istemcinin, sunucunun kimliğini doğrulaması ile çalışır.

1.2.2. SSL Sertifikaları

SSL, web sunucusunu tanımak için, dijital olarak imzalanan sertifikalar kullanır. Sertifika, aslında, o organizasyon hakkında bazı bilgiler içeren bir veri dosyasıdır. Aynı zamanda da, kuruluşun açık-kapalı anahtar çiftinin “açık” anahtarı (“public key”) da sertifika içinde yer alır. Sertifikalar, “güvenilir” sertifika kuruluşları tarafından dağıtılır. Sunuculara verilen SSL sertifikaları, istemcilerin bağlandıkları sunucuların kimlik doğrulamasının yapılarak sahteciliğin önlenmesini ve istemci-sunucu iletişimde verilerin şifrelenmesini sağlar.

Sertifika sahibi olmak için web sunucusunun bir “Sertifika İmzalama İsteği (CSR)” oluşturması gereklidir. Bu CSR oluşturma süreci sunucuda açık ve kapalı anahtarların oluşturulmasını sağlayacaktır. Güvenilir bir sertifika sağlayıcısına gönderilen CSR dosyası, sunucuya ait açık anahtarı barındırmaktadır. Sertifika otoritesi tarafından bu CSR dosyasına ait sertifikayı oluşturur ve bu sertifika web sunucusuna yüklenir.

İhtiyaca göre 3 farklı SSL sertifikası vardır bunlar:

SSL sertifikası: Tek bir sunucu adı için verilen bir sertifikadır. Sertifikanın içinde adı yazılı olan sunucuya yüklenerek SSL güvenliği sağlar.

SAN SSL sertifikası: Birden fazla sunucu veya alan adını (domain) içerebilen bir SSL sertifikasıdır. Ortalama olarak 10-15 farklı sunucu adını içerebilir.

Wildcard SSL sertifikası: Tek bir alan adı için alınabilen ve o alan adının tüm alt alan adlarını kapsayan bir sertifikadır.

Wildcard ile SAN SSL sertifikası arasındaki fark aynı alan adı için kullanılan birçok sunucu sertifikası için Wildcard SSL sertifikaları; aynı kurumun farklı alan adlarına ait sunucular için ise SAN SSL sertifikaları kullanılabilir.

SSL sertifikalarında kriptografik olarak bir şifreleme algoritması ve bir özetleme (“hashing”) algoritması kullanılmaktadır. Yaygın olarak kullanılmakta olan RSA, DSA, ECC gibi asimetrik şifreleme algoritmaları ve MD5, SHA-1, SHA-256 gibi özetleme algoritmaları SSL sertifikalarının da temelini oluşturur.

Sürüm	V3
Seri No	11213a7cd8db3a9c77183eb7f...
İmza algoritması	sha256RSA
İmza karma algoritması	sha256
Sertifikayı Veren	AlphaSSL CA - SHA256 - G2, G...
Geçerlilik başlangıcı	23 Şubat 2015 Pazartesi 17:2...
Geçerlilik sonu	23 Şubat 2020 Pazar 17:28:48

E-devlet'in RSA asimetrik şifreleme algoritması ve SHA-256 özetleme algoritmasını kullanmaktadır.

1.2.3. SSL Bağlantısının Oluşması

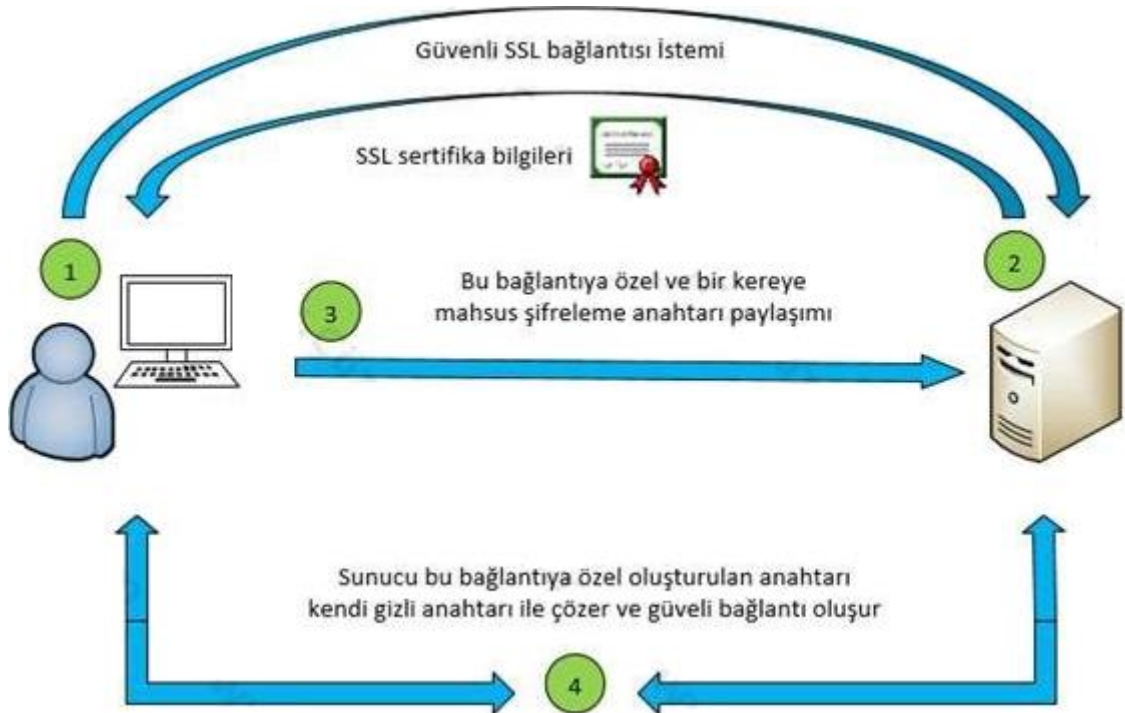
Adım 1: İstemci, bilgisayar ağı üzerinde bağlanmak istediği sunucuya güvenli olarak bağlanma talebi (https) gönderir. Bu işlem, varsayılan ayarların etkin olduğu durumda, SSL portu olarak tabir edilen 443 portu üzerinden yapılır. Sunucu yapılandırmasına bağlı olarak farklı bir port numarası da kullanılabilir.

Adım 2: Ardından, sunucu SSL sertifikasını istemciye gönderir. Bunun üzerine istemci bu sertifikayı üreten kurumun sağladığı çevrimiçi doğrulama servislerini kullanarak bu sertifikanın geçerliliğini kontrol eder. Ayrıca, bağlantı sırasında kullanılmakta olan web tarayıcısı da bu sertifikayı üreten kurumun güvenilirliğini tarayıcıda önceden tanımlı olan kurumlarla kıyaslayarak kontrol eder. Bu aşamada sunucu, açık anahtarını istemciyle SSL sertifikası aracılığıyla paylaşmış olur.

Adım 3: Eğer tüm kontroller doğrulanırsa, istemci sunucuya tek kullanımlık simetrik oturum anahtarını ("session key" veya "encryption key"), sunucunun açık anahtarıyla şifreleyerek gönderir.

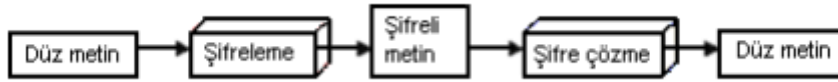
Adım 4: Sunucu, gizli anahtarını kullanarak şifreyi çözer ve tek kullanımlık simetrik oturum anahtarını alarak, istemciyle arasında güvenli bir SSL bağlantısı kurar. Böylece, bağlantı süresince bu iki cihaz arasındaki tüm iletişim güvenli bir şekilde devam eder.

SSL bağlantısının oluşmasının görsel anlatımı



1.3. Şifreleme Algoritmaları

Bir göndericinin alıcıya açık ağ üzerinden bir ileti göndermek istediği zaman, açık ağda gönderilen iletiler üçüncü şahıslar tarafından dinlenme ve değiştirilme tehdidi altındadır. Burada söz konusu ileti düz metin (plaintext)'dir. Bir iletinin içeriğini saklamak üzere yapılan gizleme işlemi de şifrelemedir (encryption). Bu işlem düz metni anahtar kullanarak şifreli metne dönüştürmektedir. Şifrelenmiş bir ileti şifreli metindir (ciphertext). Şifreli metni düz metne geri çevirme işlemi şifre çözümdür (decrypt). Şekil 1'de bu süreç gösterilmiştir.



Şekil 1 Şifreleme ve şifreyi çözme işlemleri

Simetrik Şifreleme: Şifreleme ve şifre çözmenin daha önceden paylaşılmış bir anahtar ile yapıldığı şifreleme türüdür. 128 veya 256 bitlik anahtarlar kullanılabilir. Web sunucusu ve tarayıcı arasında SSL Handshake sırasında tasarlanan oturum (session key), simetriktir.

Asimetrik Şifreleme: Şifreleme ve şifre çözmenin ayrı anahtarlar ile yapıldığı şifreleme türüdür. Burada herkes public (açık) anahtar ile şifreli mesaj gönderebilmekte, yalnız private (kapalı) anahtar sahibi tarafından bu mesajlar çözümlenebilmektedir.

Asimetrik ve simetrik şifrelemenin nasıl kullanıldığı aşağıda belirtilmiştir.

- ❖ Sunucu tarayıcıya asimetrik public anahtarını gönderir.
- ❖ Tarayıcı simetrik oturum anahtarını oluşturur. Sunucu asimetrik anahtarı ile oturum anahtarını şifreler ve web sunucusuna gönderir.
- ❖ Sunucu kendi private anahtarı ile şifreyi çözer ve simetrik oturum anahtarını elde eder.
- ❖ Sunucu ve tarayıcı bu simetrik anahtarı kullanarak güvenli bir oturum başlatabilir.

1.3.1 Simetrik şifreleme algoritmaları

Simetrik şifreleme algoritmaları şifreleme ve şifre çözme işlemleri için tek bir gizli anahtar kullanmaktadır. Bu durum veri şifreleme için matematiksel açıdan daha az problem çıkaran bir yaklaşımdır ve çok kullanılan bir yöntemdir. Bu tip algoritmalarda şifreleme işlemi gerçekleştirildikten sonra şifreli metni alıcıya gönderirken şifreli metinle birlikte gizli anahtarı da alıcıya güvenli bir şekilde göndermek gerekmektedir. Simetrik şifreleme algoritmaları çok hızlı bir şekilde şifreleme ve şifre çözme işlemlerini gerçekleştirebilmektedir. Tablo 1’de çeşitli simetrik şifreleme algoritmaları hakkında bilgiler verilmiştir.

Tablo 1: Bazı simetrik şifreleme algoritmaları bilgileri

Algoritmanın Adı	Geliştiren	Tarihi	Tipi (Blok Uzunluğu)	Anahtar Uzunluğu	Döngü Sayısı	Çözülme Durumu	Kullanım Koşulları
DES (Data Encryption Standard)	IBM (ABD)	1977	Feistel Blok (64 bit)	56 bit (parity ile 64 bit)	16	Sağlam; 8 döngülü çeşidi çözülebiliyor; 16 zayıf anahtar	Serbest
IDEA (International Data Encryption Algorithm)	Lai-Massey, ETH Zurich (İsviçre)	1992	Blok (64 bit)	128 bit	8	Sağlam; 2 ⁵¹ zayıf anahtar	Ticari faaliyetler hariç serbest
RC2 (Rivest's Cipher veya Ron's Code2)	Rivest, RSA Data Security (ABD)	1992	Katar	2048 bite kadar	Bilinmiyor	Zayıflık bulunmadı	Algoritma RSA tarafından saklı tutuluyor
RC5 (Rivest's Cipher veya Ron's Code5)	Rivest, RSA Data Security (ABD)	1995	Blok (32, 64 veya 128 bit)	2048 bite kadar	255'e kadar	64 bit blok ve 12 döngü ile diferansiyel ve doğrusal şifre çözüme dayanlı	Serbest
GOST 28147-89	I.A.Zabotin, G.P.Glazkov, V.B.Isaeva (Sovyetler Birliği)	1989	Feistel Blok (64 bit)	256 bit; 512 bit tanımlanabilir sübsitüsyon; 610 bit etken gizli bilgi	32	SSCB tarafından bütün gizlilik derecelerindeki bilgiler için uygun görülmüştür	Serbest
ASEKAL-21	Aselsan (Türkiye)	-	Doğrusal olmayan katar	57 bit ?	-	Ulusal olarak onaylanmış algoritma	Aselsan 2101, 2010 veri ve ses şifreleme birimlerinde kullanılıyor

1.3.2. Şifrelemede kullanılan anahtar boyutları

Anahtarın deneme-yanılma yöntemiyle bulunmasını engellemek için, kullanılan anahtarların uzunluğunun mümkün olduğunca büyük olması gerekmektedir. Tablo 2’de farklı anahtar boyları için, saniyede bir milyon, bir milyar ve bir trilyon şifre deneyebilen bilgisayarlar için anahtar çözme süreleri verilmiştir.

Tablo 2 Farklı anahtar boyutları için anahtar çözme süreleri

Anahtar	Olası	10 ⁶ şifre/s hızında	10 ⁹ şifre/s hızında	10 ¹² şifre/s hızında
Uzunluğu değeri	sayısı	ortalama	ortalama	ortalama
(n)	(2 ⁿ)	çözme süresi	çözme süresi	çözme süresi
32 bit	~4x10 ⁹	36 dak	2.16 s	2.16 ms
40 bit	~10 ¹²	6 gün	9 dak	1 s
56 bit	~7.2x10 ¹⁶	1142 yıl	1 yıl 2 ay	10 saat
64 bit	1.8x10 ¹⁹	292 000 yıl	292 yıl	3.5 ay
128 bit	1.7x10 ³⁸	5.4x10 ²⁴ yıl	5.4x10 ²¹ yıl	5.4x10 ¹⁸ yıl

1.3.3. Sayısal imza

Sayısal imza elektronik mesaja eklenmiş bilgidir. Çift anahtarlı bir şifreleme algoritmasıyla hazırlanan sayısal imza, hem gönderilen bilginin sayısal içeriğinin değiştirilmediğinin hem de gönderen tarafın kimliğinin ispatlanması için kullanılır ve gönderilecek mesajdan üretilen “mesaj özetinin” sayısal içeriği, gönderen tarafın kendi özel anahtarına bağlı olarak oluşturulur. Sayısal imzanın doğruluğunu kanıtlamak için mesajı alan taraf, kendisine gelen mesajın ve sayısal imzanın sayısal içeriği ile gönderen tarafın açık anahtarını kullanmaktadır.

1.3.4. Şifreleme Modları

Simetrik anahtar şifreleme, modern blok şifreler kullanılarak yapılabilir. DES veya AES kullanılan herhangi bir boyuttaki metni şifrelemek için operasyon modları tasarlanmıştır.

Burada blok seviyesindeki desenler korunur. Blok bağımsızlığı, anahtar bilinmeden bazı şifreli metin blokları değiştirmesi için fırsatları içinde barındırdığı için güvenlik sorunu teşkil edilmektedir. Aktarımdaki tek bir bit hatası, ilgili blokta birkaç hata oluşturabilir. Bu yönden hata yayılımı konusunda, hatanın diğer bloklar üzerinde herhangi bir etkisi yoktur. ECB’de dolgulama (padding) işlemi vardır.

Şifreleme modlarının tablo halinde incelenmesi

<i>Operation Mode</i>	<i>Description</i>	<i>Type of Result</i>	<i>Data Unit Size</i>
ECB	Each n -bit block is encrypted independently with the same cipher key.	Block cipher	n
CBC	Same as ECB, but each block is first exclusive-ored with the previous ciphertext.	Block cipher	n
CFB	Each r -bit block is exclusive-ored with an r -bit key, which is part of previous cipher text	Stream cipher	$r \leq n$
OFB	Same as CFB, but the shift register is updated by the previous r -bit key.	Stream cipher	$r \leq n$
CTR	Same as OFB, but a counter is used instead of a shift register.	Stream cipher	n

Şifreleme modlarında kullanılan blok şifreleme yönteminin girişten gelen veriye bağımlılığı kötü bir özelliktir ve şifrenin çözülme oranını arttırmaktadır. Şifreleme yaparken kullanılan uzayın büyüklüğü, şifrelemenin kuvvetini etkiler.

Uygulanan şifreleme modunun etkisiyle şifrelenen açık metnin geri dönüşü bir o kadar önemli bir noktadır. Bu noktada CTR (Counter Mode Encryption) modunun geriye dönüşü bulunmamaktadır. Burada aslında mükemmeline tekrar etmeyen bir sayı üreticine dayanmaktadır. Yani matematiksel olarak bir fonksiyon sürekli sayı üretmekte ancak bu üretilen sayılar kendini tekrar etmemektedir. Ancak bu şekildeki fonksiyonun da ürettiği sayıların mümkün olduğunca rastgele olması ve önceden kestirilemez olması istenir.

1.4. Kullanılan Programlama Dilleri ve Platformlar

Java James Gosling tarafından geliştirilmeye başlanmıştır. Sun Microsystem Java'yı ilk olarak 1995 yılında piyasaya sunmuştur. Java günümüzde neredeyse her platformda kullanılmaktadır. Nesneye dayalı, açık kodlu ve yüksek seviyeli bir programlama dilidir. Java dili C ve C++ dan birçok sözdizimi ve yapı kullanmasına rağmen daha basit ve anlaşılabilir. Java platformu aynı yazılımın birçok değişik bilgisayar ortamında veya değişik tür makinelerde çalışması fikri ile geliştirilmiştir. Platformdan (Unix, Microsoft, MacOS) bağımsızdır. İnternet üzerinden uygulama geliştirme yapılabilir. Java yazılımlarını geliştirmek için JDK (Java Development Kit) programının kurulumu yapılması gerekmektedir. En çok tercih edilen programlama dillerinden biridir.

Eclipse IBM Kanada'nın bir projesi olarak 2001 yılında Object Technology International tarafından piyasa sunulmuştur. Açık kaynak kod geliştirme ortamıdır. Temel olarak Java ile ilgili bir teknoloji ise de C, C++, Ruby gibi programlama dilleri içinde kullanılır. Basit ara yüzü ve kolay kullanımı ile çok kısa bir zamanda en çok kullanılanlar arasına girmiştir. Java da olduğu gibi Eclipse de platformdan bağımsızdır. Unix, Microsoft, MacOS işletim sistemlerini desteklemektedir.

Yazılım sektörü içerisinde en sık kullanılan iki yazılım dili olan C ve C++ etkileşimi ile türetilmiştir. Ayrıca C#, ortak platformlarda taşınabilir bir (portable language) programlama dili olan Java ile pek çok açıdan benzerlik taşımaktadır. En önemli özelliği ise .Net Framework platformu için hazırlanmış tamamen nesne yönelimli bir yazılım dilidir. Yani nesneler önceden sınıflar halinde yazılıdır. Programcıya sadece o nesneyi sürüklemek ve sonrasında nesneyi amaca uygun çalıştıracak kod satırlarını yazmak kalır.

Microsoft Visual Studio, Microsoft tarafından geliştirilen bir tümleşik geliştirme ortamıdır (IDE). Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, .NET Compact 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.

1.5. İstemci – Sunucu (Client – Server) Mimarisi

Ağıdaki her bilgisayarın rolü farklıdır. Bu roller istemci (client) ve sunucu (server) olabilir. İstemci genel olarak kullanıcı ihtiyaçlarını etkileşime sokmak için sunum hizmetleri, veri tabanı hizmetleri ve bağlantı sağlayan bir yapıdır.

Sunucu ise iş prosedürleriyle ilgili arabirimlerin yanı sıra bağlantı ve veri tabanı hizmetleri sağlayan ayrıca istemciye göre daha yüksek donanım özelliklerine sahip bir istasyondur. Bu mimaride merkezde sunucu bilgisayar bulunur ve kaynakların kullanımını, kullanıcıların yetkilerini ayarlamakla sorumludur. İstemci bilgisayarların tümü sunucu bilgisayara bağlıdır ve sunucunun verdiği yetkiler doğrultusunda işlemler gerçekleştirilir. İstemci tek kullanıcı gibi davranırken sunucu çok kullanıcı yapıya sahiptir. Bunun nedeni istemci tek bir hizmeti almak için sunucuya istek gönderirken, sunucu tarafında ise birden çok istemcinin hizmet talebinin sağlanması amaçlanmaktadır.

Genellikle, birden çok istemci programı, ortak bir sunucu programının hizmetlerini paylaşır. Hem istemci programları hem de sunucu programları genellikle daha büyük bir programın veya uygulamanın bir parçasıdır. Örneğin web tarayıcıları (Web'in gönderilmesi sayfalar veya dosyalar) internetteki başka bir bilgisayarda bulunan web sunucusundan (teknik olarak Köprü Metni Aktarım Protokolü veya http sunucusu olarak adlandırılır) servis talep eden bir istemci programıdır.

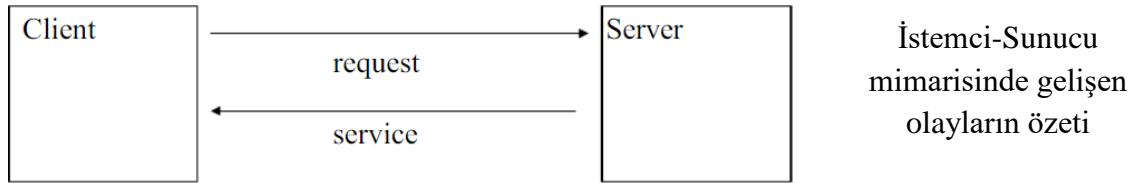
Sunucu yapısı gereği çoklu-thread yapısını kullanmaktadır. Bunun için öncelikle thread'i açıklanması gerekir. Thread, bir program içerisinde farklı kod kısımlarını aynı zamanda çalıştırılmasının bir yoludur. Bunun yanında socket kullanımında önemlidir. Socket, süreçler arası iletişim için kullanılır. Interprocess iletişimi istemci-sunucu modeline dayanır. Bu durumda, istemci-sunucu birbiriyle etkileşime giren uygulamalardır. İstemci ve sunucu arasındaki etkileşim için bir bağlantı gereklidir. Soket programlaması, etkileşime girecek uygulamalar arasındaki bağlantıyı kurmaya yarar. Kısacası ağ iletişiminin gerçekleşmesini sağlayan yapıdır.

Sunucu – istemci yapısı ile internetteki herhangi bir siteye bağlanırken bu yapıdan faydalanılır. Sitenin sunucusu varsayılan olarak belirtilmiş olan porta gelen bağlantı istekleri için sürekli dinler. Gelen bağlantıları diğer bir port ile meydana getirdiği soket ile tanımlar. Bundan dolayı sunucu sürekli varsayılan portu dinlemiş olur.

İstemci bağlantı isteğinde bulunur. Eğer sunucu, istemcinin isteğini kabul ederse ona bir soket nesnesi açar ve farklı bir porttan kendisiyle iletişim kurmasını sağlar. Sunucu tarafında sürekli olarak bir dinleyici (Listener) soketi bulunur. Bu soketler belli bir port üzerinde dinleme yaparlar. Tüm bağlantı isteklerini bu port üzerinden kabul eder ve ya reddeder. Wireshark programı kullanırken yukarıda bahsedilen yapıların da kullanımı gerekmektedir.

Açık metni şifreli olarak gönderen taraf ile şifreli metni alıp şifresini çözen alıcı taraf (İstemci-sunucu) arasında oluşan şifreli iletişimi görebilmek adına sıkça kullanılan wireshark programı kullanılmıştır. Wireshark, bağlanılan internetin tüm protokolleri kullanarak ne tip veriler işlendiğini görebilmemize olanak sağlayan bir programdır. Anlık olarak da, daha önce kaydedilmiş veriler de tekrar tekrar gözden geçirilebilir.

Wireshark, Windows bilgisayara kurulduğu zaman WinPcap adında bir araçla beraber gelir. WinPcap, o an bağlı olunan internet bağlantısını, bilgisayar donanımı olan Ethernet kartı üzerinden izlemesine yardımcı olur. Aslında işi WinPcap halleder ama rahat kullanılması ve grafiksel olması için Wireshark arayüzü kullanılır. Bu program ile şifrelenen metin ağ üzerinde incelenerek doğruluğu test edilmiştir. Burada inceleme yaparken ağ üzerinde başka paketlerde gönderilmiştir. İnceleme yaparken buna dikkat edilmiştir. Ayrıca inceleme esnasında görmek istedinen şifreli metinde de kayıplar oluşabilmektedir.



- ❖ Kullanıcı herhangi bir işlem yapmak için istemci yapısını çalıştırır.
- ❖ İstemci sunucuya bağlanır.
- ❖ İstemci sunucuya istek gönderir.
- ❖ Sunucu bu isteği analiz eder.
- ❖ Sunucu isteğin sonucunu bulur.
- ❖ Sunucu bulduğu sonucu istemciye iletir.
- ❖ İstemci sonuçları kullanıcıya gösterir.
- ❖ Bu işlemler gerektiği kadar tekrarlanır.

2. PROJE TASARIMI

2.1. Gereksinim Analizi

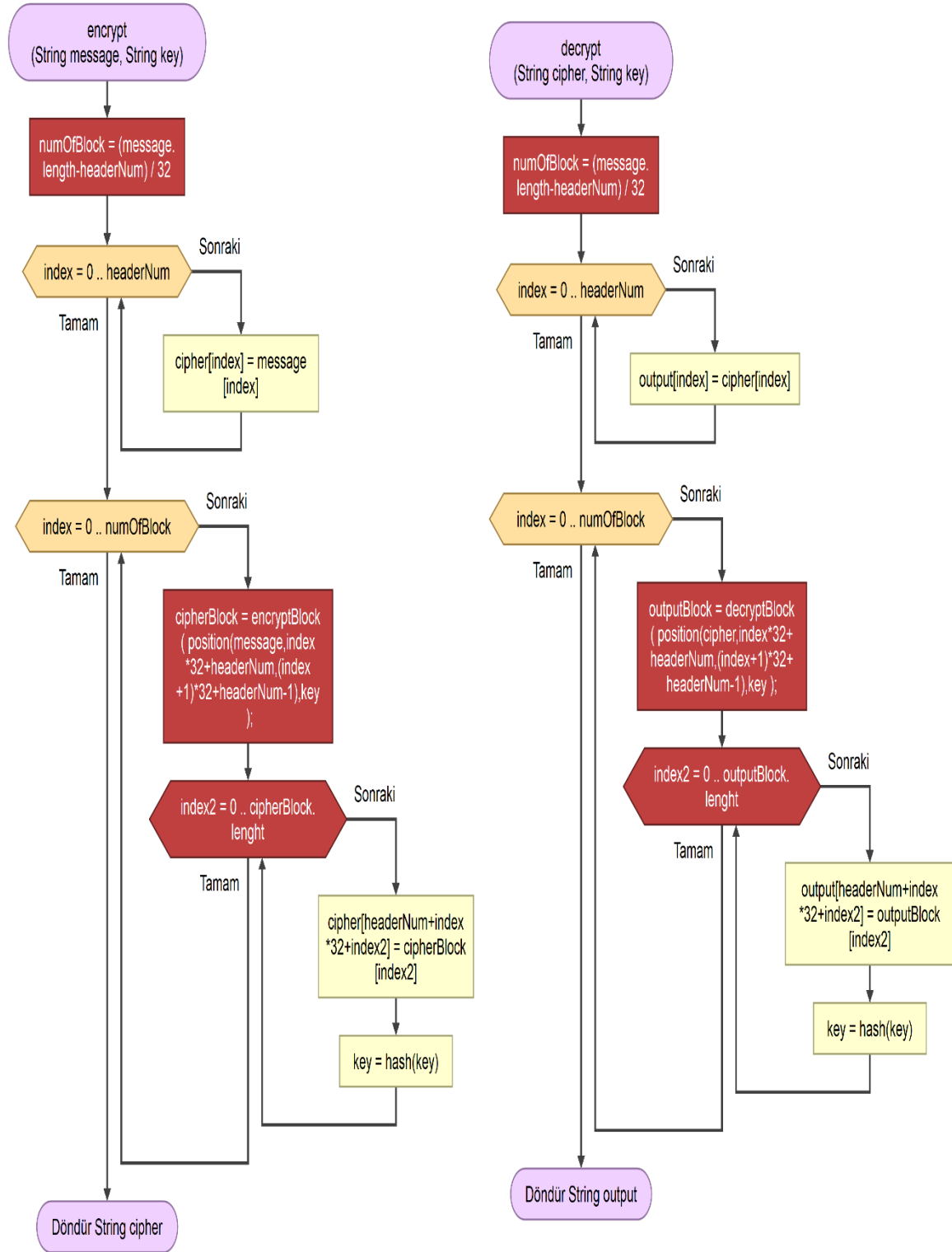
Algoritma tasarımı:

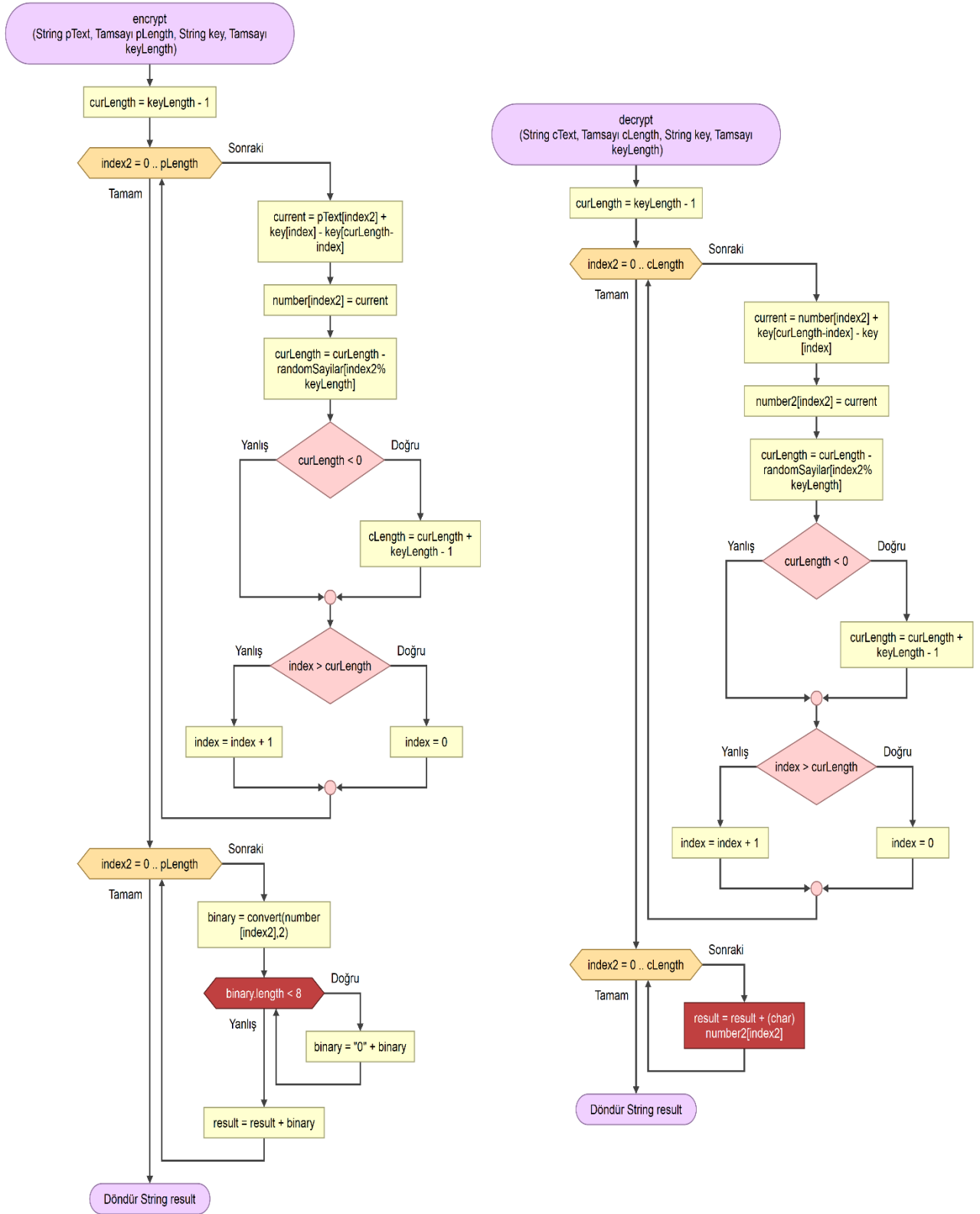
- ❖ SSL (Secure Socket Layer) protokolüne benzer, günümüz bilgisayarlarının kıramayacağı güçlükte şifreleme yapan algoritmaların gerçekleştirilmesi
- ❖ Algoritmanın optimizasyonu
- ❖ Algoritmanın bir programlama dili ile implementasyonu
- ❖ Şifreleme algoritmalarının deşifreleme doğruluğu, kararlılık ve hız testi
- ❖ Algoritmaya şifreleme tekniklerinde kullanılan test yaklaşımlarına karşı dayanıklılık kazanması
- ❖ Şifreleme yöntemlerinde anahtar için hash (özet fonksiyon) kullanımı

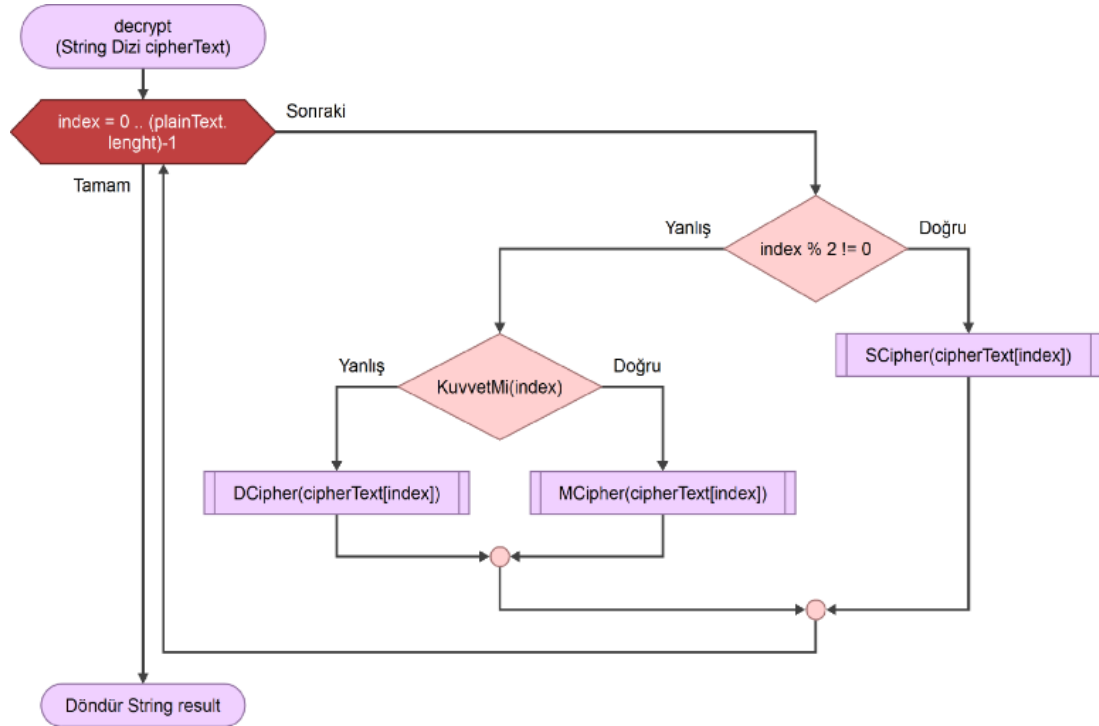
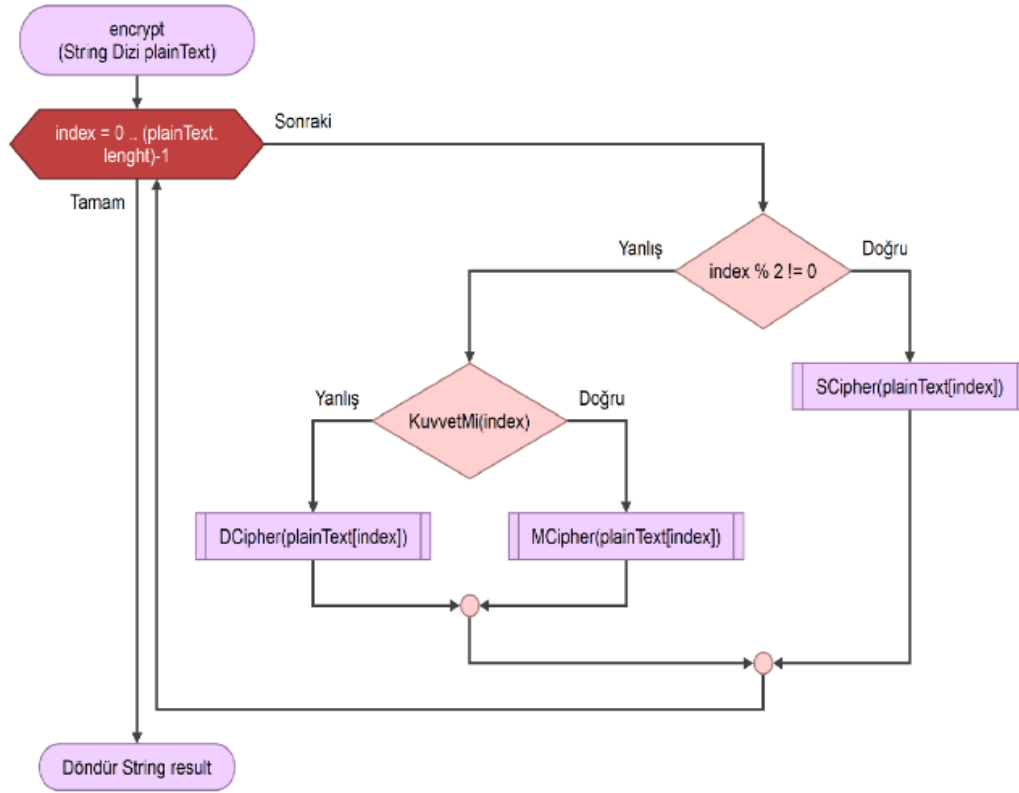
Client&Server:

- ❖ Client tarafında algoritmalarımızı kullanarak şifreleme işlemini gerçekleştirilmesi
- ❖ Şifrelenmiş metnin http (değişebilir) protokolü üzerinden sunucuya gönderilmesi
- ❖ Sunucu üzerinde şifreli gelen metnin, deşifrelenerek plaintext'in elde edilmesi
- ❖ Elde edilen plaintext'in karşılığında sunucudan client'a geri gönderilecek metnin, tekrardan aynı algoritma kullanılarak şifrelenmesi
- ❖ Client tarafında şifrelenmiş metnin çözülerek iletişimin tamamlanması

2.2. Mimari Tasarım



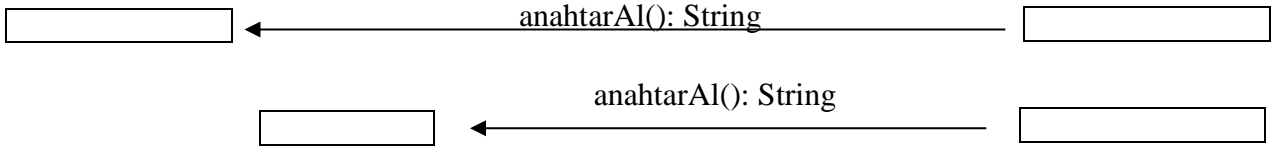




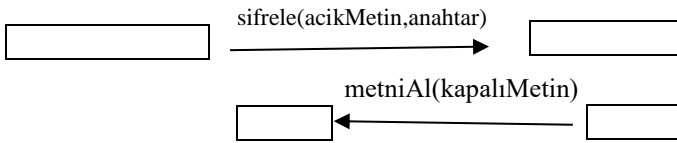
2.3. Şifreleme Algoritmalarının UML Sıralı Diyagramı (Sequence Diagram)

SUNUCU	İSTEMCİ	ŞİFRELEME	DEŞİFRELEME	ANAHTAR
--------	---------	-----------	-------------	---------

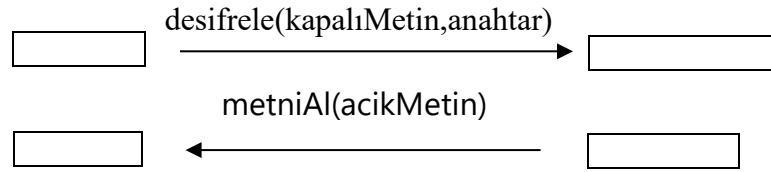
ANAHTAR GÖNDERİM AŞAMASI



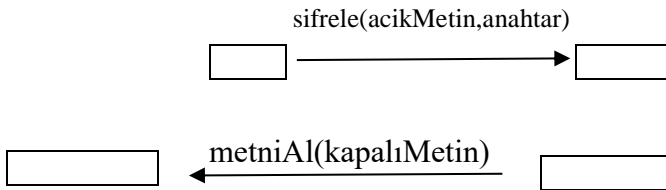
SUNUCUDAN ALINAN AÇIK METNİN ŞİFRELENİP İSTEMCİYE GÖNDERİM AŞAMASI



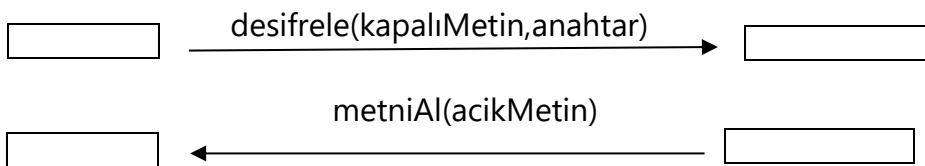
İSTEMCİYE GELEN ŞİFRELİ METNİ DEŞİFRELEME İŞLEMİ



İSTEMCİDEN ALINAN AÇIK METNİN ŞİFRELENİP SUNUCUYA GÖNDERİM AŞAMASI



SUNUCUYA GELEN ŞİFRELİ METNİ DEŞİFRELEME İŞLEMİ



2.4. Yapılan Çalışmalar

2.4.1. Simetrik Şifreleme Tasarımları

2.4.1.1. Simetrik Şifreleme Algoritması – 1

Simetrik şifreleme tasarımı için ilk önce kullanılacak yapıların belirlenmesi gerekmektedir. Bu süreçte şifrelenecek açık metin (plaintext), SHA256 özet fonksiyonu ile şifrelenir. SHA256 ile şifrelemek için açık metin 256 bitlik parçalara bölünür. Açık metin parçaları sırasıyla anahtar ve hash(key) ile XOR'lanır. Her şifrelenmesinden sonra anahtar hashlenmiş anahtar ile güncellenmektedir.

$$\begin{aligned} C1 &= P1 \oplus \text{Key} \oplus H(\text{Key}) \\ \text{Key2} &= H(\text{Key}) \\ C2 &= P2 \oplus \text{Key2} \oplus H(\text{Key2}) \\ \text{Key3} &= H(\text{Key2}) \\ &\dots \\ Cn &= Pn \oplus \text{Key} \oplus H(\text{Key}) \\ \text{Keyn} &= H(\text{Key}(n-1)) \end{aligned}$$

Blok Şifreleme Kısım

$$\begin{aligned} P1 &= C1 \oplus \text{Key} \oplus H(\text{Key}) \\ \text{Key2} &= H(\text{Key}) \\ P2 &= C2 \oplus \text{Key2} \oplus H(\text{Key2}) \\ \text{Key3} &= H(\text{Key2}) \\ &\dots \\ Pn &= Cn \oplus \text{Key} \oplus H(\text{Key}) \\ \text{Key} &= H(\text{Key}(n-1)) \end{aligned}$$

Blok Deşifreleme Kısım

Deşifreleme işleminde de şifrelenmiş metin 256 bitlik parçalara bölünür. Şifrelenmiş metin parçası anahtar ve hashlenmiş key ile XOR'lanır. Bu işlemle açık metin parçası elde edilir. Her metin parçasının şifrelenmesinden sonra anahtar hashlenmiş anahtar ile değiştirilir.

```
private static byte[] XOR(byte[] a, byte[] b)
{
    if(a.length!=b.length)
    {
        System.err.println("ERROR -> Array lengthes are different.");
        return null;
    }
    byte[] c = new byte[a.length];
    for(int i=0; i<c.length; i++)
        c[i]=(byte) (a[i] ^ b[i]);
    return c;
}
```

Açık metni şifrelerken kullanılan anahtarla birlikte yapılan XOR işlemi için bir metot tanımlanmaktadır. Burada alınacak 2 tane dizinin boyutunun eşit olması gerekir.

Şifreleme yapmak için kullanılacak metotta (encrypt), mesajı ve anahtarı byte array tipinde alır. Daha sonra 32 byte'lık parçalara bölünür. Toplam blok sayısını bulmak için, gelen mesaj uzunluğunda boş bir dizi açılır. Bu dizinin ilk 35 değeri direkt cipher'a aktarılır. Bu kısmın şifrlenmemesinin sebebi; dosyanın türünü değiştirmek istenmemesidir.

```
String file_path = "C:\\Users\\Hknaksoyy\\Desktop\\a.jpg";
File input_file = new File(file_path);
byte[] plainText = Files.readAllBytes(input_file.toPath());
String key = "asdfgytasdfgytrfghtuytorfghtuyto";
byte[] cipher = encrypt(plainText, key.getBytes());
Path path = Paths.get("C:\\Users\\Hknaksoyy\\Desktop\\b.jpg");
Files.write(path, cipher);
```

Bu kısımda ilk olarak şifrelenecek dosyanın konumu yazılır. Okunacak dosya bitlere çevrilip diziye aktarılır. Belirlenecek gizli anahtar 32 karakter uzunluğunda bulunmalıdır. Daha sonra şifrenilmiş mesaj cipher adlı diziye koyulup, istenilen konuma koyulabilmektedir.

```
String file_path = "C:\\Users\\Hknaksoyy\\Desktop\\b.jpg";
File cipher_file = new File(file_path);
byte[] cipher = Files.readAllBytes(cipher_file.toPath());
String key = "asdfgytasdfgytrfghtuytorfghtuyto";
byte[] output = decrypt(cipher, key.getBytes());
Path path2 = Paths.get("C:\\Users\\Hknaksoyy\\Desktop\\c.jpg");
Files.write(path2, output);
```

Daha sonra şifrenilmiş mesajın konumundan şifrenilmiş mesajı cipher isimli bir dizinin içine aktarılır. Bu kısımda yapılacak deşifrelemek işleminde kullanılacak anahtarın, şifrenilirken kullanılan anahtarla aynı olması gerekir. Deşifreleme işlemi tamamlandıktan sonra çözülmüş mesajı yazdırmak istenilen konuma gönderilebilmektedir.



Açık Metin



Şifrenilmiş Metin



Deşifrenilmiş Metin

2.4.1.2. Simetrik Şifreleme Algoritması - 2

Şifreleme algoritmasının en başında kullanıcıdan şifrelenmesini istediği input'u alabilmek için bir input değişkeni tanımlandı. Daha sonra Encrypt ve Decrypt fonksiyonlarında kullanılmak üzere sayılar ve sayılar2 adında 2 tane int liste tanımladı. Üretilen random key içinde key adında bir değişken ve Decrypt fonksiyonunda kullanılmak üzere bir int dizi tanımlanmıştır.

```
string input = "";
List<int> sayilar = new List<int>();
List<int> sayilar2 = new List<int>();
string key = "";
int[] random_sayilar = new int[32];
```

Random key üretmek için kurulan algorithmada current değişkenine 65 ile 122 arasında değerler atanmaktadır. Bu değerlerin ASCII karşılıklarını kullanarak harf üretilebilmektedir. 65'in ASCII karşılığı A iken 122'nin ASCII karşılığı z'dir. Yani bu algoritma ile 26 tane farklı (Türkçe ve özel karakterler dahil değil) büyük harf 26 tanede küçük harf tanımlanabilmektedir.

```
for (int i = 0; i < 32; i++)
{
    current = rand.Next(65,123);
    while (current > 90 && current < 97)
        current = rand.Next(65, 123);
    key = key + (char)current;
}
```

Burada dikkat edilmesi gereken nokta ise ASCII tablosunda 91 ile 96 arasındaki değerler özel karakterler olduğu için, 91 ile 96 arasında bir değer üretildiğinde bu değer aralığındaki değerler gelmeyene kadar random sayı üretimine devam edilir. Üretilen değerler ASCII değerlerine karşılık düşen karakterlere dönüştürülerek key değişkenine atanır ve bu döngü 32 kez dönerek 32 karakterli random bir key üretilir.

```
for (int i = 0; i < key.Length; i++)
{
    random_sayilar[i] = key[i] % key.Length;
}
```

Buradaki algorithmada ise key uzunluğu kadar for döngümüz dönmektedir ve her dönüşünde key de ki harflerin ASCII değerlerinin mod 32'sini almaktadır. Örneğin *A harfi* için $65(A'nın\ ASCII\ karşılığı) \% 32 = 1$ olarak random_sayilar dizimize atanmaktadır. Diğer bir örnek ise *a harfi* için inceleyecek olursak $97(a'nın\ ASCII\ karşılığı) \% 32 = 1$ olarak random_sayilar diziyeye atanmaktadır. Bu diziyi daha sonra Decrypt fonksiyonda şifreli mesajı(cipher text) çözmek için kullanılmaktadır.

Altta kısımda şifrelemede kullanılan `current_length` ve `j` değişkenleri tanımlandı. `Current_length` değişkeninin ilk değeri keyin son indexini gösterecek şekilde, `key.Length-1` (31) değerini alır. `J` değişkeni ise keyin ilk indexini gösterecek şekilde 0 değerine setlenmiştir.

```
int current_length=key.Length-1;
int j=0;
for (int i = 0; i < input.Length; i++)
{
    if (j > current_length)
        j = 0;
    int current = int.Parse((input[i] + key[j] - key[current_length - j]).ToString());
    sayilar.Add(current);

    current_length = (current_length - random_sayilar[i % random_sayilar.Length]);
    if (current_length <= 0)
        current_length = key.Length - 1 + current_length;
    j++;
}
```

Döngü kullanıcıdan alınan `text`'in uzunluğu kadar dönmektedir. Döngünün her dönüşünde kullanıcıdan alınan `text`'in karakterleri sırasıyla şifrelenmektedir. Şifrelemede örnek ile açıklayacak olursak girilen `text`'in 'bilgisayar' olduğu düşünüldüğünde, `current` değişkenine (b harfinin `ascii` karşılığı) + (key'in ilk harfinin `ascii` karşılığı) – (key'in son harfinin `ascii` karşılığı) atanmaktadır. Çıkan sonuç sayılar listesine eklenir ve bu sayılar listesi *Decrypt* fonksiyonumuzda kullanılacaktır. Bilgisayar kelimesinin uzunluğu kadar (10) döngümüz diğer harflerinin şifrelenmesi içinde tekrarlanacaktır.

`Current_length` değişkenini her döngüde anahtarın harfine göre bir sayı atanmaktadır. Örneğin anahtarın o anki harfinin 'O' olduğu düşünülürse, bu harfin `ASCII` karşılığı $79 \% 32 = 15$, `current_length = 31-15 = 16` olarak güncellenir. Her döngüde key değerine göre `current_length` değişkeni güncellenebilir, eğer `current_length` değişkeni 0'dan küçük olursa `current_length=(key.Length-1) + current_lenght` değeri ile güncellenir.

```
for (int i = 0; i < sayilar.Count; i++)
{
    string binary = Convert.ToString(sayilar[i], 2);
    while(binary.Length < 8)
    {
        binary = "0" + binary;
    }
    Output = Output + binary;
}
```

Encrypt fonksiyonunun son kısmında ise şifreleme algoritmasının yardımıyla elde edilen değerlerin binary karşılığını elde edilir. Eğer çevrilen binary değerın uzunluğu 8 değilse, bu uzunluğu 8'e tamamlayana kadar binary değerin başına 0 eklenir ve son olarak bu binary değeri output değişkenine atanmaktadır.

```
int j = 0;
int current_length = key.Length - 1;
for (int i = 0; i < input.Length; i++){
int current=(sayilar[i]+int.Parse((key[current_length-j] - key[j]).ToString()));
sayilar2.Add(current);
```

Encrypt fonksiyonunda da kullanılan j ve current_lenght değişkenleri Decrypt fonksiyonunda da tanımlanıp kullanıldı. Current_length değişkeninin ilk değeri anahtarın son indexini gösterecek şekilde key.Length-1 (31) değerine setlenmiştir. Döngümüz kullanıcıdan alınan text'in uzunluğu kadar dönmektedir. Döngünün her dönüşünde kullanıcıdan alınan text'in karakterleri sırasıyla deşifrelenmektedir.

```
for (int i = 0; i < sayilar2.Count; i++)
    Output = Output + (char)sayilar2[i];
```

Döngünün içerisinde current değişkenine şifreli değerleri tutulan sayılar listesini ve şifrelerken kullanılan keyleri ters şekilde kullanarak, şifreli değeri çözüp sayılar2 listesine eklenmektedir. Çözülmüş ve sayılar2 listesine eklenmiş değerler başta input olarak girilen text deki harflerin ASCII karşılığıdır. En son olarak deşifrelediğimiz değerleri karaktere çevirip ekrana yazdırmaktayız.

Uygulamanın ekran görüntüsü

The screenshot shows a web application with a light gray background. At the top, there is a 'Create Key' button and a text input field containing the key 'QulALAHSVffzBiYudXCXMsBNEUMlyqm'. Below this, there are three main sections. The first section is labeled 'Enter Your Message:' and contains a text input field with the word 'bilgisayar'. To the right of this field is a button labeled 'Encrypt Message'. The second section is labeled 'Cipher Message:' and contains a text input field displaying the binary string '0100011001110101011001010100001001100010011011101100111100000110100011010000001'. To the right of this field is a button labeled 'Decrypt Message'. The third section is labeled 'Original Message:' and contains a text input field with the word 'bilgisayar'.

2.4.1.3. Simetrik Şifreleme Algoritması - 3

Bu yöntemde ilk olarak her türkçe harf karakterine karşılık bir şifreli karakter kullanılmaktadır. Şifrelemedeki key(anahtar) bu karakterlerin tutulduğu tablolar olmaktadır. Bu tablolar her bir açık metnin her karakterinin teklik, çiftlik ve 2'nin kuvveti durumlarına göre oluşturulmuştur. Örneğin "A" karakteri için çift tablosunda "." karakteri belirlenmiştir. Tek tablosu için ise "A" karakterine karşılık "m" karakteri belirlenmiştir.

```
Dictionary<string, string> dictionary = new Dictionary<string, string>();  
Dictionary<string, string> dictionary1 = new Dictionary<string, string>();  
Dictionary<string, string> dictionary2 = new Dictionary<string, string>();  
ArrayList aList = new ArrayList();
```

Yukarıdaki kod kısmında tabloların tanımlamalarını ve tiplerini belirlendi ayrıca girilen karakterlerin tutulduğu dizi listesi oluşturulmuştur.

```
private void dictionaryCift()  
{  
    dictionary.Add("A", ".");  
    dictionary.Add("B", "z");  
    dictionary.Add("C", "@");  
    dictionary.Add("Ç", "y");  
    .      .  
    .      .  
}
```

Tablolara (tek, çift, ikinin kuvveti) değer atama işlemleri yapıldı. Dictionary anahtar kelimesi ile her karaktere karşılık bir şifreli karakter eşleme işlemi yapılmıştır.

```
for (int i = 1; i <= aList.Count; i++)  
{  
    int index = i;  
    if (index % 2 == 0 && ((index & (index - 1)) != 0))  
        cift(aList[i - 1].ToString());  
    else if (index % 2 != 0)  
        tek(aList[i - 1].ToString());  
    else if (index != 0 && ((index & (index - 1)) == 0))  
        two_us(aList[i - 1].ToString());  
    aList[index - 1] = " ";  
}
```

Bu kısımda girilen düz metnin boyutunun sayısı kadar bu işlemler tekrarlanmaktadır. (Encrypt kısmıdır). Girilen düz metnin her bir karakterinin indis'ine karşılık düşen tablo değerlerine göre şifreli metni oluşturan kısımdır. (Decrypt kısmı) Şifrenin çözüldüğü kısımda aynı mantıkla çalışmaktadır. Şifreli metnin her bir karakterine karşılık gelen açık metni oluşturur.

Algoritmada için ilk yapılması gereken açık metnin seçimidir. Açık metinde dikkat edilmesi gereken nokta, harflerin konumlarıdır. Harflerin konumlarına göre teklik, çiftlik, ikinin kuvveti şeklinde oluşturulan tablolara göre şifreleme yapılır.

Bir karakterin konumu hem çift hem ikinin kuvveti ise ikinin kuvvetine öncelik verilir. Örneğin “1.” Konuma denk gelen karakterin tek tablosuna göre şifrelemesi yapılır. “2.” Konuma denk gelen karakterin ikinin üssü tablosuna göre şifrelemesi oluşturulmuştur. “6.” Konuma denk gelen karakterin ise çift tablosuna göre şifrelemesi oluşturulmuştur.

Tasarlanan şifreli metnin aynı tablolar yardımıyla alıcı tarafta her bir şifreli karaktere karşılık düşen açık metin karakteriyle çözümlenmesi yapılmaktadır. Yöntemde şifre oluşturulmaya çalışıldığında bir harfin farklı konumlara göre şifreli karakter karşılığı değişecektir. Bu yöntemin böyle kullanılması, bruteforce ataklarına karşı dayanıklılığını söz konusudur. Ayrıca farklı kombinasyonlara göre belirlenen tabloların sayısı artırılarak da şifrelemenin sağlamlığı artırılabilir.

Bu şifrelemenin oluşturulmasında günümüzde kullanılan simetrik şifreleme tekniklerinden faydalanılmıştır. Bu teknikleri kullanırken hem simetrik şifrelemenin yapısının anlaşılabilmesi hem de olan şifreleme tekniklerinden farklı bir şifreleme tekniğinin olması amacı ile oluşturulmuştur. Bu şifreleme tekniği veri güvenliği sağlaması amacı ile oluşturulacak olan güvenlik protokolü yapısında diğer şifreleme teknikleriyle birlikte uyumlu bir şekilde kullanılabilir.

Açık Metnin Şifrenmesi

<u>GİRDİ METNİ</u>	<u>ÇIKTI METNİ</u>
DENEME	03=3k/

ŞİFRELE

Şifreli Metnin Deşifrenmesi

<u>GİRDİ METNİ</u>	<u>ÇIKTI METNİ</u>
03=3k/	DENEME

ÇÖZ

2.4.2. Kaba Kuvvet Saldırısı (Brute Force Attack)

Bilgisayar bilimlerinde bir metnin içerisinde başka bir metnin aranması için kullanılan en ilkel ve dolayısıyla en düşük performanslı arama algoritmasıdır. Brute Force, genellikle bir parolayı (veya anahtarı) ele geçirmek için yapılan bir çeşit dijital ve kriptografi saldırısıdır. Brute Force tekniğinde elde herhangi bir bilgi bulunmuyor olmasına karşın belli şifreler denenerek doğru şifreye ulaşmaya çalışılır.

Bu saldırı yönteminde genellikle insanların sıklıkla kullandığı 123456789 ve 987654321 gibi basit şifreler başta olmak üzere birçok şifreden oluşan bir liste hazırlanır. Daha sonra meydana getirilen yardımcı bir yazılım yardımıyla veya el yordamıyla bu şifreler parolası bulunmak istenen hesaba giriş yapabilmek için tekrar tekrar denir. Yazılım, doğru şifre bulunduğu anda bir sinyal vererek işlemi durdurur.

Brute Force garanti işe yarayan bir yöntem olmadığı gibi aynı zamanda yoğun zaman isteyen bir saldırı türüdür. Çünkü eldeki liste ne kadar büyük olursa bu bir avantaj olacak olmasına karşın denenecek şifreler de o kadar olacak demektir. Bu şartlar altında Brute Force ile başarıya ulaşmak saatler, aylar ve hatta yılları bile alabilmektedir.

```
static const char words[] =  
"abcdefghijklmnopqrstuvwxyz"  
"ABCDEFGHIJKLMNOPQRSTUVWXYZ"  
// "0123456789";
```

```
static const int wordSize = sizeof(words) - 1;
```

```
for ( i=0; i<wordSize; i+=1 )  
{  
    str[index] = words[i];  
    if ( index==maxSize-1) printf("%s\n", str);  
    else bruteImplament(str, index + 1, maxSize);  
}
```

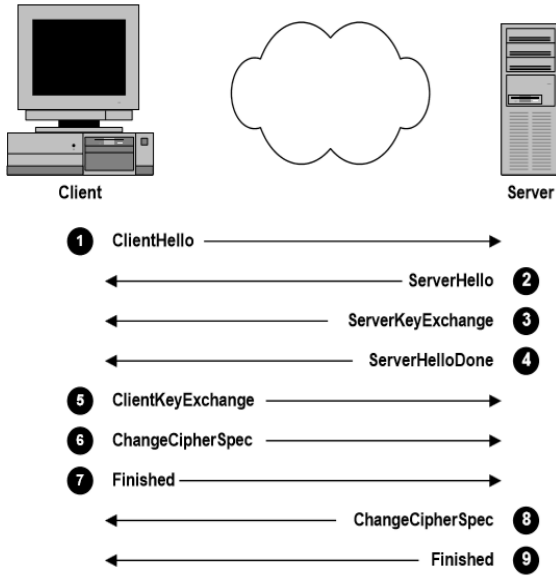
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z	aa ab ac ad ae af ag ah ai aj ak al am an ao ap aq ar as at au av aw ax ay az aA aB aC aD	YZK YZL YZM YZN YZO YZP YZQ YZR YZS YZT YZU YZV YZW YZX YZY YZZ Zaa Zab Zac Zad Zae Zaf Zag Zah Zai Zaj Zak Zal Zam Zan	ZZZA ZZZB ZZZC ZZZD ZZZE ZZZF ZZZG ZZZH ZZZI ZZZJ ZZZK ZZZL ZZZM ZZZN ZZZO ZZZP ZZZQ ZZZR ZZZS ZZZT ZZZU ZZZV ZZZW ZZZX ZZZY ZZZZ
0.08621 seconds	2.544 seconds	21.93 seconds	657.3 seconds

2.4.3. SSL Protokolü

2.4.3.1. Handshake Protokol (El sıkışma Protokolü)

Bir tarayıcı SSL ile korunan bir sunucuya bağlanmak istediğinde, sunucu ve tarayıcı arasında “SSL Handshake” denilen bir süreç başlatılmış olur. Bu süreç kullanıcılardan soyutlanmış anlık bir süreçtir. Bu süreçte üç adet anahtar kullanılmaktadır ve bunlar; public, private ve simetrik oturum anahtarıdır. Asimetrik algoritmanın işlem gücünün maliyetli oluşu, sadece simetrik oturum anahtarının paylaşımında kullanılmasına sebep olmuştur. Bu aşamadan sonra güvenlik simetrik oturum anahtarı ile sağlanmaktadır.

- ❖ Tarayıcı SSL ile güvenliğini sağlanan bir web sunucusuna (https) bağlanır ve sunucudan kendini tanıttırmasını ister.
- ❖ Sunucu; sahip olduğu SSL sertifikasının kopyasını ve açık anahtarını gönderir.
- ❖ Browser gelen sertifika için zamanının geçmediğini ve geçerli bir sertifika olduğu kontrollerini yapar. Tarayıcı sertifikanın bağlandığı siteye ait olduğunu teyit ederse, simetrik oturum anahtarını sunucunun public anahtarı ile şifreleyip gönderir.
- ❖ Sunucu şifreyi çözerek, simetrik oturum anahtarını elde ettiğine dair acknowledge bilgisini simetrik oturum anahtarı ile şifreleyerek tarayıcıya gönderir.
- ❖ Bu aşamadan sonra sunucu ve istemci iletilecek mesajı şifrelenmiş olarak gönderebilir.



SSL handshake protokolü, sertifika transferi ve oturum anahtarlarının tanıtılması için, istemci-hizmet biriminin her birinin hangi şifre takımlarını karşılıklı kimlik tanımlamada kullanacağını belirler.

SSL TCP protokolüne ihtiyaç duyar. Dolayısıyla SSL iletişiminden bahsedebilmek için öncelikle TCP 3-way handshake ile TCP bağlantının kurulması gereklidir. Paket dosyasındaki ilk 3 paket 3-way handshake mekanizmasına aittir ve TCP bağlantı kurulduktan sonraki aşama SSL Handshake'tir.

Şifre takımı; DES, DSA, KEA, MD5, SHA-1, RC2, RC4, RSA, RSA SKIPJACK, Triple DES ve Key Exchange algoritmalarını tanır.

Handshake protokolü her mesaj tür, uzunluk ve içerik bilgilerini içerir. Dört aşamadan oluşmaktadır. Birinci aşama güvenlik yetenekleri belirlenmesi (client hello ve server hello), ikinci aşama sunucu kimlik doğrulama ve anahtar değişimi (Sertifika, RSA, Diffie Hellman), üçüncü aşama istemci kimlik doğrulama ve anahtar değişimi (anahtar değişimi, sertifikayı doğrulama), son aşamada güvenli bağlantı oluşturmayı tamamlama (change cipher spec)'dir.

2.4.3.1.1. Handshake Protokol Aşamaları

Client Hello Aşaması

İstemci, SSL iletişimi kurma isteğini ClientHello mesajı göndererek belirtebilmektedir. Bu mesajın içine desteklediği SSL protokol versiyonu, şifreleme mekanizmaları (Cipher Suites olarak gösterilen alan, key exchange + cipher + hash algoritmaları bütünü), destekleyebildiği sıkıştırma yöntemi, SessionID ve RandomNumber denen ve 32 byte'lık bir değer (bu değer şifreleme anahtarı üretiminde kullanılır, ilk 4 byte'ı tarih ve saat bilgisini taşır) yerleştirir.

Server Hello Aşaması

Server Hello, Certificate ve Server Hello Done mesajları paket dosyasındadır. Sunucu Client Hello mesajını aldığı anda Server Hello mesajıyla yanıt verir. İstemcinin Client Hello mesajından destekleyebildiği protokolleri, mekanizmaları öğrenen sunucu, SSL iletişimde kullanılacak şifreleme mekanizmasını, SSL versiyonu, SessionID numarasını, sıkıştırma metodunu belirler ve yine şifreleme (simetrik şifreleme) anahtarının oluşturulmasında kullanılmak üzere ürettiği randomNumber değerini de bu mesaj içine yerleştirir.

Certificate Aşaması

Sunucu public sertifikasını (Public Key) bu mesajla gönderebilmektedir. Public sertifika CA (Certificate Authority) private sertifikası tarafından imzalandığından ve CA public sertifikasıyla bu imza doğrulanabildiğinden, sunucu sertifikasının güvenilirliği kontrol edilebilmektedir. Kontrol edilemezse sertifikanın güvenilir olmadığına dair kullanıcı uyarılır.

Server Hello Done Aşaması

Bu mesaj anlaşmanın ilk fazının bittiğini belirtir ve istemci handshake'i tamamlamak üzere ClientKeyExchange, ChangeCipherSpec ve Finished mesajlarını gönderebilir.

ClientKeyExchange Aşaması

Bu aşamada istemci daha önce değiş-tokuş edilen randomNumber'lar ile birlikte kullanılarak simetrik key (anahtar) üretiminde kullanılacak 46 byte uzunluğunda "pre-master secret" da denen gelişigüzel bir numara üretir ve bu numarayı, MAC (Message Authentication Code) değeriyle birlikte sunucunun daha önce "Certificate" mesajıyla kendisine gönderdiği public sertifikasını kullanarak şifrelenebilmektedir. Bu mesajın yapısı ve içeriği, uzunluğu da kullanılan key-exchange algoritmasına göre değişebilmektedir.

"Certificate" mesajında gönderilen public sertifikayla şifrelenen bu değeri deşifre edebilecek olan da sadece bu public sertifikanın private sertifikasına sahip olan uçtur (Private Key). Şifreleme işleminde anahtar bu sertifika değil; sertifika aslında şifreleme işleminde kullanılacak anahtarı üretmede kullanılan "ön anahtar"dır. Asıl anahtar istemci ve sunucu arasında karşılıklı veya tek taraflı gönderilmemektedir. Anahtarı, daha önce iletilen bilgiler (randomNumber) ışığında istemci ve sunucu (pre-master secret ı da kullanarak) üretmektedir ve sonuç olarak ayrı ayrı üretilen bu anahtar aslında aynı anahtarı(simetrik) ifade etmektedir.

ChangeCipherSpec Aşaması

İstemci bu mesajla sunucuya bundan sonraki iletişimde üzerinde mutabık olunan protokoller, algoritmalar, anahtarla SSL protokolünü kullanma isteğini bildirir.

Finished Aşaması

ChangeCipherSpec mesajının gönderilmesiyle, istemci ve sunucu kendilerinden gelecek mesajların simetrik anahtarla şifreleneceği bilgisini içeren bir mesaj gönderir. Finished mesajı şifrelenen ilk mesajdır ve bundan sonraki iletişim de şifrelenir. Şifrelenen veri ise daha önceki SSL handshake mesajlarının hashi ve gönderen tarafın istemci veya sunucu olduğunu belirten bir başka değerdir.

TCP 3-way handshake'ler TCP bağlantı, SSL Handshake'le de güvenli iletişim kanalı kurulduktan sonraki iletişim, gönderilen Request ve Response'lar aynıdır,hiç bir farkı yoktur.

2.4.3.2. Alert Protokol (Uyarı Protokolü)

SSL kayıt katmanı tarafından desteklenen içerik türlerinden biri uyarı türüdür. Uyarı mesajları, mesajın ciddiyetini ve uyarının açıklamasını iletir. Ölümcül seviyeli uyarı mesajları, bağlantının derhal sonlandırılmasına neden olur.

Bu durumda, oturuma karşılık gelen diğer bağlantılar devam edebilir, ancak başarısız olan oturumun yeni bağlantılar kurmak için kullanılmasını engelleyen oturum tanımlayıcısının geçersiz kılınması gerekir. Diğer mesajlarda olduğu gibi, geçerli mesajda belirtildiği gibi uyarı mesajları şifrelenir ve sıkıştırılır.

2.4.3.2.1. Kapanış Uyarıları (Closure Alerts)

İstemci ve sunucu, bir kesme saldırısını önlemek için bağlantının sona erdiği bilgisini paylaşmalıdır. Her iki taraf da kapanış mesajı değişimini başlatabilir.

`close_notify`: Bu mesaj alıcıya gönderenin bu bağlantıda daha fazla mesaj göndermeyeceğini bildirir. Herhangi bir bağlantı düzgün kapatılmadan sonlandırılırsa, uyarı uyarıya eşit seviyedeki mesajları bildirmeden oturum önemsizleşir. Her iki taraf da `close_notify` uyarısı göndererek kapanış başlatabilir. Bir kapatma uyarısından sonra alınan veriler göz ardı edilir.

Her bir taraf, bağlantının yazma tarafını kapatmadan önce bir `close_notify` uyarısı göndermelidir. Karşı tarafın kendi tarafından bir `close_notify` uyarısı ile yanıt vermesi ve bekleyen tüm yazıları atmadan hemen bağlantıyı kapatması gerekir. Bağlantının okunan tarafını kapatmadan önce kapanış başlatıcısının yanıt veren `close_notify` uyarısını beklemesine gerek yoktur. Not: Bir bağlantının kapatılmasının taşımayı yok etmeden önce bekleyen verileri güvenilir bir şekilde sağladığı kabul edilir.

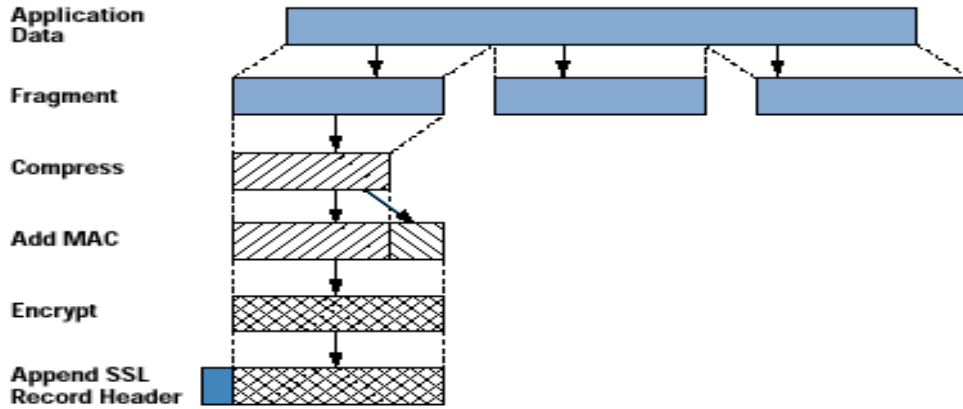
2.4.3.2.2. Hata Uyarıları (Error Alerts)

SSL anlaşması protokolünde hata işleme çok basittir. Bir hata tespit edildiğinde, tespit eden taraf diğer tarafa bir mesaj gönderir. Ölümcül bir uyarı mesajı alındığında veya alındığında, her iki taraf da bağlantıyı derhal kapatır. Sunucular ve istemciler, başarısız bir bağlantıyla ilişkili tüm oturum tanımlayıcılarını, anahtarlarını ve sırlarını unutmak zorundadır.

2.4.3.3 Record Protocol (Kayıt Protokolü)

Kayıt protokolü bağlantılar için gizlilik ve mesaj bütünlüğü sağlamaktadır. Gizliliği, uygulama verisini şifreleyerek, mesaj bütünlüğünü ise mesaj doğrulama kodu (MAC) kullanarak gerçekleştirir.

Kayıt Protokolü iletilmek üzere bir uygulama mesajını alır, bu mesaj boyutu belirli olan bloklara (2^{14} bytes) bölünür. Sonra bloklara isteğe bağlı olarak sıkıştırma (compression) uygulanır. Simetrik anahtar ile mesaj doğrulama kodu (MAC) hesaplanır ve sıkıştırılmış veriye eklenir. MAC'li ve sıkıştırılmış veri bloğu simetrik (AES, DES, Fortezza) olarak şifrelenir ve SSL Record Header bloğa eklenerek SSL Kayıt İşlemi (SSL Record Operation) tamamlanır.



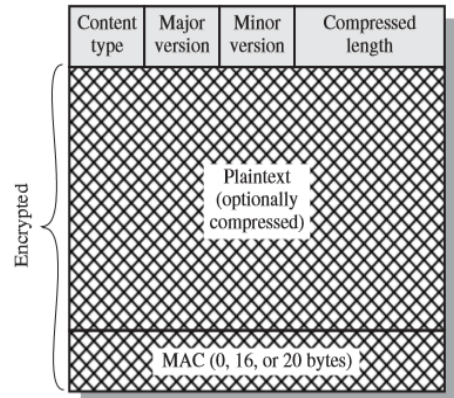
SSL Kayıt Protokolü işleminin son adımı, bir header(başlık) hazırlamaktır ve bu header aşağıdaki alanlardan oluşmaktadır.

- Content Type (İçerik Türü)(8 bit): Ekteki parçayı işlemek için kullanılan üst katman protokolü. Tanımlı olan içerik türleri change cipher spec, alert, handshake ve application data'dır.

- Major Version (Ana versiyon) (8 bit): Kullanılan SSL'in ana sürümünü gösterir. Örneğin SSLv3 için değer 3'tür.

- Minor Version (Küçük versiyon)(8 bit): Kullanılan minnor(küçük) versiyonu gösterir. Örneğin SSLv3 için değer 0'dır.

- Compressed Length (Sıkıştırılmış uzunluk) (16 bit): Düz metnin bayt cinsinden uzunluğu eğer sıkıştırma kullanılıyorsa sıkıştırılmış parçanın uzunluğudur.



SSL Record Protokol Formatı

2.4.4. Wireshark Testi

Wireshark network trafiğinin, bir grafik arayüz üzerinden izlenmesini sağlayan, pek çok zaman hayat kurtarıcı öneme sahip bir programdır. Uygulamanın kurulu olduğu bilgisayar üzerinden anlık network trafiği izlenebileceği gibi, Wireshark daha önce kaydedilmiş dosyaların incelenmesi amacı ile de kullanılabilir.

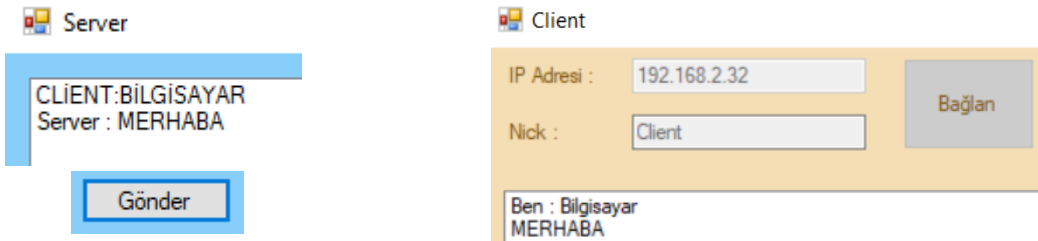
2.4.4.1. Wireshark'ın Özellikleri

- ❖ Önceki ismi Ethereal'dir ve GPL lisansı ile dağıtılır.
- ❖ Unix/Linux ve Windows işletim sistemlerinde kullanılabilir.
- ❖ Yerel ağdaki paketleri tutar ve ayrıntılı olarak protokol bilgileri de dahil olmak üzere bizlere görüntüler. Yakalamış olduğu paketleri kaydedebilir.
- ❖ Oluşturacağınız kriterlere göre paket arama ve filtreleme yapabilirsiniz.
- ❖ Filtreleme sayesinde paketleri renklere ayırır ve kategorileştirir.
- ❖ İstatistikleri bizim yapacağımız ayarlar ile bizlere sunar.
- ❖ Protokoller için şifre çözme desteği vardır.

Aynı ağ üzerinde bağlı olan 2 bilgisayar üzerindeki iletişim

No.	Time	Source	Destination	Protocol	Length	Info
13	2.847061	192.168.2.57	192.168.2.32	TCP	77	49392 → 4108
14	2.887843	192.168.2.32	192.168.2.57	TCP	54	4108 → 49392
108	30.057828	192.168.2.32	192.168.2.57	TCP	64	4108 → 49392
109	30.358996	192.168.2.32	192.168.2.57	TCP	64	[TCP Retrans
110	30.369487	192.168.2.57	192.168.2.32	TCP	66	49392 → 4108
373	61.034058	192.168.2.57	192.168.2.32	TCP	71	49392 → 4108 [PSH, ACK] Seq=1 Ack=1 Win=68 Len=17
376	61.274411	192.168.2.32	192.168.2.57	TCP	54	4108 → 49392 [ACK] Seq=1 Ack=18 Win=513 Len=0

Proje tasarımında gerçeklemeye çalışılan şifreleme algoritmalarının testlerini bir sunucu-istemci uygulaması üzerinden testi yapıldı. Bu test üzerinde ilk önce server tarafında açılan port ile dinleme işlemi başladı. Bu kısımda socket programlamada thread yapısının kullanımı söz konusudur. Daha sonra istemci tarafından belirlenen ip numarası ile bağlantı sağlanmış oldu.



```
48 d2 24 8a 54 2b 3c f8 62 a3 41 6b 08 00 45 00 H-$-T+<- b-Ak--E-
00 39 7b 16 40 00 80 06 f9 fe c0 a8 02 39 c0 a8 -9{-@... ..9..
02 20 c0 f0 10 0c 7f 75 f1 e6 89 88 00 37 50 18 .-...-u .....7P-
00 44 8a 81 00 00 6e 3f 74 2f e2 89 a0 7b 3a ce -D...-n? t/...{:
b8 76 3f 6d 31 0d 0a -v?m1..
```

3. Kaynaklar

- ❖ SSL and TLS Essentials – Securing the Web (Stephen Thomas)
- ❖ Şifreleme (Doktora dersi) – (cs.stfx.ca/~ltyang/csci-467)
- ❖ <http://sutod.selcuk.edu.tr/sutod/article/viewFile/120/601>
- ❖ <https://crypto.stanford.edu/~nagendra/papers/dtls.pdf>
- ❖ <https://www.cise.ufl.edu/~nemo/crypto/slides/ch16.pdf>
- ❖ <https://tools.ietf.org/html/rfc5246#page-29>
- ❖ <http://blog.btrisk.com/2014/04/ssl-nedir-2bolum.html>
- ❖ <https://www.iacr.org/archive/fse2006/40470265.pdf>
(A Study of the MD5 Attack: Insights and Improvements)
- ❖ <https://doc.lagout.org/security/Crypto/Implementing../.pdf>
(Implementing Using Cryptography and PK)
- ❖ <https://get-itlabs.com/category/security/>
- ❖ [https://www.exploit-db.com/docs/\[.\]turkish/40448-wireshark-important-tips.pdf](https://www.exploit-db.com/docs/[.]turkish/40448-wireshark-important-tips.pdf)

STANDARTLAR ve KISITLAR FORMU

Projenin hazırlanmasında uyulan standart ve kısıtlarla ilgili olarak, aşağıdaki soruları cevaplayınız.

1. Projenizin tasarım boyutu nedir? (Yeni bir proje midir? Var olan bir projenin tekrarı mıdır? Bir projenin parçası mıdır? Sizin tasarımınız proje toplamının yüzde olarak ne kadarını oluşturmaktadır?)

Projenin ilk kısmında simetrik şifreleme algoritması tasarımı gerçekleştirilmiştir. İlerleyen bölümde bu algoritmaları kullanarak SSL protokolüne benzer bir yapı tasarlanacaktır.

2. Projenizde bir mühendislik problemini kendiniz formüle edip, çözdünüz mü? Açıklayınız.

Tasarlanılan simetrik şifreleme algoritmalarını formüle edip, örnek bir sunucu ve istemci uygulaması üzerinden testi gerçekleştirilmiştir.

3. Önceki derslerde edindiğiniz hangi bilgi ve becerileri kullandınız?

Projemizde nesne yönelimli programlama yapılarını sıklıkla kullanıldı. Bu yapılarla birlikte; sunucu ve istemci arasında kurulan soket programlama yapısı için “Bilgisayar Ağ Programlama” dersinden yararlanılmıştır.

4. Kullandığınız veya dikkate aldığınız mühendislik standartları nelerdir? (Proje konunuzla ilgili olarak kullandığınız ve kullanılması gereken standartları burada kod ve isimleri ile sıralayınız).

Şifrelemelerin implementasyonunda Java ve C# programlama dilleri kullanıldı. Daha sonra testleri için “Wireshark” programından faydalanılmıştır.

5. Kullandığınız veya dikkate aldığınız gerçekçi kısıtlar nelerdir? Lütfen boşlukları uygun yanıtlarla doldurunuz.

a) Ekonomi

Projenin ekonomik açıdan bir maliyeti bulunmamaktadır.

b) Çevre sorunları:

Projenin çevreye olumsuz bir etkisi bulunmamaktadır.

c) Sürdürülebilirlik:

Tasarlanacak şifreleme algoritma sayılarının artması, ilerleyen zamanda kullanımı olumlu yönde etkileyecektir.

d) Üretilbilirlik:

Projenin ilerleyen kısmında tasarlanılan algoritmaların verimliliği ile kullanıcıların etkinliği artmaktadır.

e) Etik:

Proje etik açıdan bir problem teşkil etmemektedir.

f) Sağlık:

Proje sağlık açısından bir sorun teşkil etmemektedir.

g) Güvenlik:

Projenin amacı “Veri güvenliği” olduğu için, bu konuda yapılan istemci-sunucu testlerinde verinin şifrelendikten sonra gönderilmesi, araya girebilecek kötü niyetli kişilerin açık metne erişimini engellemektedir.

h) Sosyal ve politik sorunlar:

Projenin herhangi bir sosyal ve politik bir sorunu bulunmamaktadır.