KARADENİZ TEKNİK ÜNİVERSİTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ
MİKROİŞLEMCİLER LABORATUVARI

Uygulamalı Kriptografi

1. GİRİŞ

Kriptografi, bilgiyi yetkisiz kişilerden korumak amacıyla **matematiksel yöntemler** ve **algoritmalar** kullanarak şifreleme ve şifre çözme bilimi olarak tanımlanır. Temel amacı, verinin gizliliğini, bütünlüğünü, kimlik doğrulamasını ve inkâr edilemezliğini sağlamaktır.

Tarihsel olarak kriptografi, Sezar Şifrelemesi gibi basit yöntemlerle başlamış, günümüzde ise gelişmiş matematiksel yapılar üzerine kurulu modern algoritmalara evrilmiştir. Bu algoritmalar iki ana grupta incelenir:

- **Simetrik Şifreleme:** Şifreleme ve çözme için aynı anahtarın kullanıldığı yöntemlerdir (ör. AES, DES). Hızlıdır fakat anahtar paylaşımı güvenlik riski doğurabilir.
- **Asimetrik Şifreleme:** Açık anahtar ve gizli anahtar çiftine dayanır (ör. RSA, ECC). Güvenli anahtar dağıtımı sağlar, fakat simetriğe göre daha yavaştır.

Bunların yanı sıra, **hash fonksiyonları** (MD5, SHA-256) verinin özetini çıkararak bütünlüğünü doğrulamada kullanılır. Kriptografi, günümüzde sadece mesaj gizliliğinde değil, aynı zamanda dijital imzalar, kimlik doğrulama, blok zinciri teknolojileri ve güvenli internet iletişimi (SSL/TLS) gibi birçok alanda kritik rol oynamaktadır.

1.1. Geleneksel Kriptografi Yöntemleri

Modern bilgisayar tabanlı algoritmalar ortaya çıkmadan önce, insanlar gizli mesajlarını saklamak için daha basit ve elle uygulanabilir yöntemler geliştirmiştir. Bu yöntemler günümüzde **geleneksel kriptografi** ya da **klasik şifreleme yöntemleri** olarak anılır.

- **Sezar Şifrelemesi:** En bilinen klasik yöntemdir. Alfabetik harfler belirli bir kaydırma miktarı kadar ötelenerek şifrelenir. Örneğin, 3 kaydırmada “A” → “D” olur. Basitliği nedeniyle kolayca kırılabilir.
- **Vigenère Şifrelemesi:** Sezar şifresinin daha gelişmiş bir versiyonudur. Şifreleme için bir anahtar kelime kullanılır ve bu kelimenin harfleri sayıya çevrilerek farklı kaydırmalar yapılır. Bir dönem “kırılamaz şifre” olarak bilinse de frekans analizi yöntemleriyle çözülmüştür.
- **Transpozisyon (Yer Değiştirme) Şifreleri:** Harflerin yerleri belirli bir kurala göre değiştirilir. Harflerin kendisi değil, diziliş sırası değiştirilerek gizlilik sağlanır.

- **Substitüsyon (Yerine Koyma) Şifreleri:** Alfabenin her harfi başka bir harfle ya da sembolle değiştirilir. Örneğin, "A" → "X", "B" → "Q" gibi.

Bu yöntemler tarihsel olarak önemlidir çünkü modern kriptografinin temelini oluşturmuşlardır. Ancak günümüzde bilgisayarların yüksek işlem gücü sayesinde kolayca kırılabildikleri için sadece eğitim ve tarihsel inceleme amaçlı kullanılmaktadırlar.

1.1.1. dcode.fr (https://www.dcode.fr/)

dcode.fr, pek çok **klasik kriptografi aracını (Caesar, Vigenère, Atbash, ROT13 vb.)** ve yardımcı çözücüler ile anahtar-araçlarını çevrimiçi sunan basit bir web aracıdır. Başlamak için siteye gidip ilgilendiğiniz **şifreleme aracını seçin**; açılan formda **Plaintext/Ciphertext** alanına metninizi yapıştırın, gerekli **parametreleri** (ör. **shift/anahtar**) girin ve **"Encrypt"** veya **"Decrypt"** düğmesine tıklayın. Anahtar bilinmiyorsa çoğu araçta **"Brute-force"** veya **"All shifts"** gibi otomatik çözüm seçenekleri ve **frekans analizi grafikleri** bulunur — bunları kullanarak hangi sonucun anlamlı olduğuna karar verebilirsiniz.

Dikkat: Türkçe karakterler **(Ç, Ö, Ş, İ, Ü, Ğ)** bazı araçlarda desteklenmeyebilir; önce metni ASCII'ye **(Ç→C, Ö→O vb.)** sadeleştirmek daha güvenilir sonuç verir.

Deneyimizde geleneksel kriptografi yöntemlerini denemek için bu siteden yararlanacağız.

1.2. Modern Kriptografi Yöntemleri

Modern kriptografi, bilgisayar biliminin ve matematiğin gelişmesiyle ortaya çıkmış, günümüzde dijital güvenliğin temelini oluşturan yöntemler bütünüdür. Geleneksel yöntemlerden farklı olarak modern kriptografi, **karmaşık matematiksel algoritmalar** ve **hesaplama zorlukları** üzerine kuruludur.

Başlıca modern kriptografi yöntemleri:

- **Simetrik Şifreleme:** Şifreleme ve çözme işlemlerinde aynı anahtar kullanılır. Hızlı ve etkilidir. En yaygın algoritmalar: **AES (Advanced Encryption Standard)**, **DES**, **3DES**.
- **Asimetrik Şifreleme:** Açık anahtar ve özel anahtar çiftine dayanır. Güvenli anahtar değişimi ve dijital imza sağlar. Örnek algoritmalar: **RSA**, **Diffie-Hellman**, **ECC**.
- **Hash Fonksiyonları:** Veriden sabit uzunlukta özet üreten, tek yönlü fonksiyonlardır. Veri bütünlüğü ve parola saklamada kullanılır. Örnekler: **SHA-256**, **SHA-3**, **BLAKE2**.
- **Dijital İmza ve Sertifikalar:** Verinin bütünlüğünü ve kaynağını doğrulamak için asimetrik kriptografi ile hash fonksiyonlarının birleşimidir. SSL/TLS protokollerinde yaygın olarak kullanılır.
- **Hibrit Sistemler:** Simetrik ve asimetrik yöntemleri birleştirir. Büyük veriler simetrik yöntemle şifrelenir, simetrik anahtar ise asimetrik yöntemle güvenli biçimde paylaşılır.

Modern kriptografi, **internet güvenliği (HTTPS, VPN, e-posta şifreleme)**, **blok zinciri**, **dijital kimlik doğrulama** ve daha birçok kritik alanda kullanılmakta ve siber güvenliğin temel taşını oluşturmaktadır.

1.2.1. Simetrik Şifreleme

Simetrik şifreleme, veriyi gizlemek için kullanılan en temel kriptografi yöntemlerinden biridir. Bu yöntemde **şifreleme (encryption)** ve **şifre çözme (decryption)** işlemleri için **aynı anahtar** kullanılır. Yani, mesajı gönderen de alan da aynı gizli anahtara sahip olmak zorundadır.

Simetrik şifreleme algoritmaları genellikle ikiye ayrılır:

- **Blok Şifreler:** Veriyi sabit boyutlu bloklara ayırarak işler (AES, DES).
- **Akış Şifreleri:** Veriyi bit veya byte düzeyinde işler (RC4 gibi).

Avantajı, **yüksek hız** ve **düşük işlem maliyeti** sunmasıdır. Bu nedenle büyük miktarda verinin hızlı bir şekilde şifrelenmesi gereken uygulamalarda tercih edilir. Ancak en büyük zorluğu, anahtarın güvenli bir şekilde paylaşılmasıdır. Eğer anahtar üçüncü bir kişi tarafından ele geçirilirse, tüm iletişimin gizliliği tehlikeye girer.

Simetrik şifreleme günümüzde dosya şifreleme, disk koruma, VPN tünelleri ve birçok ağ güvenliği protokolünde kullanılmaktadır. Özellikle **AES (Advanced Encryption Standard)**, güvenliği ve performansı nedeniyle uluslararası standart haline gelmiştir.

1.2.2. Asimetrik Şifreleme

Asimetrik şifreleme, modern kriptografinin en önemli yöntemlerinden biridir ve **açık anahtar (public key)** ile **özel anahtar (private key)** olmak üzere iki farklı anahtar kullanır. Açık anahtar herkese dağıtılabilirken, özel anahtar gizli tutulur.

Bu yöntemde, veriyi şifrelemek isteyen kişi alıcının **açık anahtarını** kullanır. Şifrelenen veri yalnızca alıcının **özel anahtarıyla** çözülebilir. Böylece güvenli iletişim sağlanır, çünkü özel anahtarı yalnızca hedef kişi bilir. Bunun tersi de mümkündür: Özel anahtarla imzalanan bir mesaj, ilgili açık anahtarla doğrulanabilir. Bu mekanizma **dijital imzalar** sayesinde kimlik doğrulama ve inkâr edilemezlik sağlar.

Asimetrik şifreleme, simetrik yöntemlere göre daha güvenlidir ancak **daha yavaştır**. Bu nedenle genellikle anahtar değişimi, dijital imzalama veya küçük boyutlu verilerin şifrelenmesi için kullanılır. Büyük verilerin şifrelenmesinde ise performans açısından simetrik algoritmalarla birlikte hibrit şekilde uygulanır.

En bilinen asimetrik algoritmalar arasında **RSA**, **Diffie-Hellman** ve **Eliptik Eğri Kriptografisi (ECC)** yer alır. Günümüzde HTTPS protokolünden e-posta güvenliğine kadar pek çok alanda kritik bir rol oynar.

1.2.3. Hash Fonksiyonları

Hash fonksiyonları, herhangi bir boyuttaki veriyi alıp sabit uzunlukta **özet** (hash değeri) üreten matematiksel algoritmalardır. Hash fonksiyonlarının temel özellikleri şunlardır:

- **Tek yönlülük:** Hash değerinden orijinal veriye geri dönmek pratikte imkânsızdır.

- **Çığ etkisi (Avalanche Effect):** Girdide yapılacak en küçük bir değişiklik, hash çıktısında büyük farklılıklar oluşturur.
- **Sabit uzunluklu çıktı:** Girdi ne kadar büyük olursa olsun, çıktı belirli uzunluktadır (örn. SHA-256 her zaman 256 bit).
- **Çakışma direnci:** Farklı iki girdinin aynı hash'i üretmesi olasılığı çok düşük olmalıdır.

Hash fonksiyonları, verilerin bütünlüğünü kontrol etmek, parolaları güvenli şekilde saklamak, dijital imza ve blok zinciri teknolojilerinde kritik rol oynar.

Örnek algoritmalar arasında **MD5**, **SHA-1** ve **SHA-2 (SHA-256, SHA-512)** bulunur. Günümüzde MD5 ve SHA-1, çakışma zafiyetleri nedeniyle zayıf kabul edilirken, SHA-256 ve daha yeni algoritmalar (SHA-3, BLAKE2) güvenlik uygulamalarında tercih edilmektedir.

Hash fonksiyonları şifreleme değildir; çünkü geri dönüşü yoktur. Bunun yerine temel işlevleri, verilerin **bütünlüğünü sağlamak** ve kimlik doğrulama süreçlerinde güvenilir bir özet üretmektir.

1.2.3.1. Zayıf Hashing Algoritmaları ve Kırılma Yöntemleri

Hash fonksiyonları veri bütünlüğünü sağlamak için tasarlansa da, bazı eski algoritmalar günümüzde **yetersiz güvenlik** sunmaktadır. En bilinen zayıf hashing algoritmaları **MD5** ve **SHA-1**'dir.

- **MD5,** 1991'de geliştirilmiş olup 128 bit uzunluğunda özet üretir. Günümüzde hızlı çalışmasına rağmen ciddi güvenlik açıklarına sahiptir. Çakışma (collision) üretmek kolaydır; yani farklı iki veri aynı hash değerini üretebilir. Bu durum sahte sertifikaların üretilmesi gibi kritik saldırılara yol açabilir.
- **SHA-1,** 1995'te kullanılmaya başlanmış, 160 bitlik çıktı üretir. 2017'de Google ve CWI tarafından gerçekleştirilen **SHAttered** saldırısı ile pratik çakışmalar üretilmiştir. Bu da SHA-1'i kriptografik uygulamalar için güvensiz hale getirmiştir.

Hashing algoritmalarının kırılması için aşağıdaki yöntemler uygulanmaktadır:

- **Dictionary Attack (Sözlük Saldırısı):** Önceden hazırlanmış kelime listeleri kullanılarak her kelimenin hash'i hesaplanır ve hedef hash ile karşılaştırılır. Eğer hash eşleşirse parola bulunmuş olur.
- **Brute Force Attack (Kaba Kuvvet):** Tüm olası kombinasyonlar denenerek hash değeri çözülmeye çalışılır. Uzun ve karmaşık parolalar için zaman maliyeti çok yüksektir.
- **Rainbow Tables:** Büyük veri tabanlarında önceden hesaplanmış hash–parola eşleşmelerinin saklandığı tablolardır. Bu tablolar sayesinde hedef hash değeri çok hızlı şekilde çözülebilir. Ancak modern uygulamalarda **salt** kullanımı bu yöntemi etkisiz hale getirir.
- **Collision Attack:** Farklı girdilerin aynı hash değerini üretmesi sağlanarak sahte belgeler ya da sertifikalar oluşturulabilir. Özellikle MD5 ve SHA-1 bu saldırılara karşı zayıftır.

1.2.4. Sertifika ve Dijital İmzalama

Dijital dünyada güvenli iletişim için sadece şifreleme yeterli değildir; aynı zamanda **verinin bütünlüğünü** ve **gönderenin kimliğini** doğrulamak da gerekir. Bu noktada **dijital imzalar** ve **sertifikalar** devreye girer.

- **Dijital İmza:** Bir verinin ya da mesajın, göndericiye ait özel anahtar ile hash'lenerek imzalanmasıdır. Alıcı, göndericinin açık anahtarını kullanarak bu imzayı doğrular. Böylece:
  - Mesajın gerçekten ilgili kişiden geldiği (kimlik doğrulama),
  - Mesajın değiştirilmediği (bütünlük),
  - Gönderenin mesajı gönderdiğinin kanıtı (inkâr edilemezlik) sağlanır.
- **Dijital Sertifika:** Açık anahtarın gerçekten ilgili kişiye veya kuruma ait olduğunu kanıtlayan elektronik belgedir. Sertifikalar genellikle **Sertifika Otoriteleri (CA)** tarafından imzalanır. Bu sayede kullanıcı, internette bağlandığı sitenin gerçekten doğru kişi/kuruma ait olduğundan emin olur.

Günümüzde **SSL/TLS sertifikaları**, web siteleri ile kullanıcı arasındaki güvenli iletişimi garanti altına almak için kullanılır. Ayrıca dijital imza, e-posta güvenliği, yazılım dağıtımı ve e-devlet uygulamalarında kritik rol oynamaktadır.

1.2.5. CyberChef (https://gchq.github.io/CyberChef/)

CyberChef, **"yemek tarifi (recipe)"** benzetmesiyle çok sayıda **dönüşüm işlemini (hashing, encoding/decoding, XOR/bit-manipulation, compress/decompress, regex, base64, hex, AES, SHA, vb.) sürükle-bırak** ile zincirleyebildiğiniz güçlü bir web aracıdır. Başlamak için sol panelden ihtiyacınız olan işlemi **(örn. "From Base64", "MD5", "XOR")** seçip ortadaki **tarife (recipe)** bölgesine sürükleyin; sağ panelde **giriş (input)** metnini yapıştırın ve anında **çıktıyı (output)** göreceksiniz. İşlem sırasını kolayca **değiştirebilir, ara sonuçları inceleyebilir ve recipe'i kaydedip paylaşabilirsiniz** — bu, karmaşık dönüştürme adımlarını tekrarlanabilir hale getirir. CyberChef, farklı **şifreleme ve hash algoritmalarını denemek, küçük değişikliklerin çığ etkisini gözlemlemek veya ikili/hex düzenlemeleri** hızlıca yapmak için idealdir.

Deneyimizde modern kriptografi yöntemlerini gerektiren bütün işlemler için bi siteden yararlanacağız.

2. DENEYE HAZIRLIK

Deneye başlamadan önce yapılması gerekenler:

- Bu föyün giriş kısmını okuyarak kavramlar hakkında bilgi edinin.
- decode.fr **(https://www.dcode.fr/)** ve CyberChef **(https://gchq.github.io/CyberChef/)** sitelerine göz atın.
- Bilgisayarınıza GCC gibi bir C/C++ derleyicisi yükleyin (Laboratuvardaki bilgisayarlarda Code::Blocks ile GCC yüklü olarak gelecektir).

3. DENEYİN YAPILIŞI

3.1. Geleneksel şifreleme:

- dcode.fr sayfasını açın: Önce Caesar cipher aracını kullanarak kısa bir metin girerek içeriğini 3 farklı anahtar (ör. shift = 3, 7, 13) ile şifreleyin.
- Aynı metin üzerinde Vigenère şifrelemesi yapın. En az iki farklı anahtar kelime ile (KEY1, MUSIC gibi) şifreleyip saklayın.
- Çözümler: Her şifrelenmiş metin için dcode.fr üzerinden otomatik çözüm (frequency analysis / brute force) seçeneklerini uygulayın ve hangi durumlarda otomatik çözümün başarılı olduğunu not edin.

3.2. Modern şifreleme:

- CyberChef'i açın: Önce **DES** yöntemini kullanarak kısa bir metin girerek içeriğini isdetiğiniz bir anahtar ile şifreleyin.
- Aynı metin üzerinde **AES** şifrelemesi yapın.
- Şifrelenmiş metinleri ve şifreleri yanınızdaki arkadaşınıza gönderin, arkadaşınızdan aldığınız metinleri şifrenin yardımıyla çözün.

3.3. Hash Fonksiyonları ve Çığ Etkisi:

- CyberChef'i açın, kısa bir metin yazıp içeriğini MD5 ile hashleyin.
- Metinde bir harfi değiştirin, yeni hash değerini alın.
- Hash değerleri arasındaki farkı karşılaştırın: hex dizilimlerini bit/byte düzeyinde inceleyin (kaç bit farklı).
- SHA-1 ve SHA-256 ile tekrarlayın.

3.4. C Kodu ve Hashleme:

- Basit bir C kodu yazın, aşağıdaki örnek gibi bir şey düşünebilirsiniz.

Örnek kod:

```
#include <stdio.h>

int main()
{
	printf("Hello world!");
}
```

- Bu kodu test.c gibi bir isimle kaydedin.
- gcc test.c -o test.exe komutuyla derleyin. Oluşan dosyayı terminalde çalıştırırın.
- Dosyayı CyberChef sayfasında input kısmına sürükleyip bırakararak yükleyin.
- Oluşan hash değerini not edin.
- Aynı kaynak kodu tekrar derleyin, terminalde tekrar çalıştırınca sonucun aynı olduğunu gözlemleyin.
- Dosyayı CyberChef'e yükleyin. Input kutusundaki küçük farkları ve oluşan hash değerindeki farkı gözlemleyin.

3.5. GPG ile Asimetrik Şifreleme ve Dijital İmza:

- CyberChef ile GPG public/private anahtar çifti oluşturulur.
- Bir sekmede key çifti oluşturun, ikincide bir mesajı açık anahtar ile şifreleyin, üçüncüde de özel anahtar ile mesajı çözün.
- Birbirinize açık anahtarı gönderin, şifreli mesajı geri isteyin. Son olarak özel anahtar ile çözün.
- Dijital imza ile bütünlük ve kimlik doğrulama deneyimlenir.

4. RAPOR SORULARI

- Geleneksel şifreleme yöntemleri ile modern şifreleme yöntemleri arasındaki farklar nelerdir?
- Hash fonksiyonlarının temel özelliklerini kısaca açıklayınız.
- MD5 niye zayıf bir hash algoritmasıdır? Collision attack nedir?
- Asimetrik şifrelemenin sertifikalardaki önemi nedir? Kısaca açıklayınız.