**Proje Adı:** Dosya Şifreleme ve Şifre Çözme Aracı

**Sorun ve Önemi**

1. Problem Tanımı

Projem, hassas verilerin güvenliği ve gizliliği açısından kritik öneme sahip olan dosyaların güvenli bir şekilde saklanması ve paylaşılmasını kolaylaştırmayı amaçlamaktadır. Günümüzde, kişisel gizlilik ihlalleri ve hassas veri sızıntıları yaygın bir sorun haline gelmiştir.

2. Sorunun Önemi

Dosyaların şifrelenmemesi, kişisel bilgilerin ve ticari sırların yetkisiz erişime açık olmasına neden olabilir. Bu, hem bireyler hem de işletmeler için ciddi güvenlik sorunlarına yol açabilir.

**Proje Açıklaması:**

Projem, kullanıcıların hassas dosyalarını güvenli bir şekilde şifrelemelerine ve daha sonra ihtiyaç duyduklarında bu dosyaları şifre çözerek orijinal hallerine geri getirmelerine olanak tanıyan bir dosya güvenliği uygulamasıdır. Temel olarak, AES algoritması ve CBC modu kullanılarak şifreleme ve şifre çözme işlemleri gerçekleştirilmektedir.

Uygulama terminal üzerinde çalışır ve kullanıcılara üç seçenek sunar. Bu seçenekler şifreleme, şifre çözme ve çıkış seçenekleridir. Programdan çıkış yapılmadığı takdirde program yapılan her işlem sonunda tekrar bu üç seçeneği sunarak kullanıcılardan seçim yapmasını ister.

Kullanılan Teknolojiler:

Python: Python programlama dilinde geliştirilmiştir.

cryptography Kütüphanesi: AES şifreleme, HMAC doğrulama ve diğer kriptografik işlemler için cryptography kütüphanesi kullanılmıştır.

**Kullanım Senaryosu:**

Kullanıcılar tarafından gerek sadece kendi bilgisayarları içerisinde şifrelemek istedikleri, gerek de başka bir bilgisayara gönderdikten sonra yine aynı program yardımı ile şifrenin çözülerek kullanılabileceği dosya şifreleme ve şifre çözme programı kullanımına dayanmaktadır. Bu bağlamda programa sahip olan herhangi bir bilgisayarda şifre çözme işlemi o bilgisayarda şifreleme gerçekleşmemiş dahi olsa ya da ortak bir veri tabanı bağlantısına gerek duyulmaksızın gerçeklenebilmelidir.

**Güvenlik Mekanizmaları:**

Güçlü Şifre Kontrolü:

Kullanıcıdan alınan şifreler, güçlü şifre kriterlerine uygun olmalıdır (büyük harf, küçük harf, rakam, özel karakter içermeli).

Anahtar Yönetimi:

Şifreleme ve şifre çözme işlemleri için kullanılan anahtarlar, güvenilir bir türetme mekanizması olan PBKDF2HMAC ile üretilir.

Hash algoritması: SHA256

İterasyon sayısı: 100,000

CBC Modunda Şifreleme ve Şifre Çözme:

encrypt\_file\_cbc fonksiyonu, CBC modunda çalışan bir simetrik şifreleme algoritması (AES) kullanarak dosyayı şifreler.

decrypt\_file\_cbc fonksiyonu, şifrelenmiş dosyayı CBC modunda çözer.

Initialization Vector (IV):

CBC modunda çalışan AES algoritması için her şifreleme işlemi için rastgele bir IV oluşturulur. Bu, aynı şifreleme anahtarıyla farklı şifrelemelerin oluşturulmasını sağlar.

HMAC Doğrulama:

Şifrelenmiş dosyaların bütünlüğünü sağlamak için HMAC kullanılır. Dosya içeriği ve IV, HMAC ile birleştirilerek doğrulama yapılır.

Padding:

AES blok boyutu olan 16 byte'a tamamlanması için padding eklenir.

Salting:

Her şifreleme işlemi için rastgele bir salt değeri oluşturulur. Salt, şifreleme güvenliğini artırmak için kullanılır.

Dosya Güvenli Silme:

Şifreleme işlemi tamamlandıktan ve şifre çözme işlemi gerçekleştirildikten sonra, orijinal dosyalar güvenli bir şekilde silinir.

**Kodlama ve Açıklamaları:**

Kullanılan kütüphanelerin eklenmesi aşağıda görülen kısımda gerçekleştirilir.

metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

generate\_key fonksiyonu salt ve şifreyi kullanarak anahtar üretimini gerçekleştirir. Aşağıda görüldüğü gibi SHA256 algoritması kullanılır ve iterasyon sayısı 100 bindir.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

get\_password fonksiyonu kullanıcılardan şifre almak amacı ile kodlanmıştır ancak burada ek olarak alınan şifrenin güçlü bir şifre olması gerekliliği de kontrol edilmektedir. Aşağıdaki kod bloğunda da görüldüğü üzere, bir büyük harf, bir küçük harf, bir sayı ve bir özel karakter bulunma zorunluluğu vardır.

metin, yazı tipi, çizgi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

encrypt\_file\_cbc fonksiyonu dosyanın içeriğini okur ve bunu plaintext olarak alır. Daha sonra veriyi bloklar halinde işleyebilmek adına gereklilik durumuna bağlı olarak padding eklemesi gerçekleştiriliyor. Salt değeri ve iv değerleri rastgele olarak oluşturulur. Bu oluşturulan tuz değeri ve kullanıcıdan alınan şifre yardımı ile anahtar oluşturma işlemi gerçekleştirilir. Daha sonra CBC modunda çalışan AES algoritması yardımı ile dosya şifrelemesi gerçekleştirilir. HMAC kullanılarak dosya bütünlüğünün kontrol edilebilmesi için HMAC oluşturuluyor. Son olarak da aşağıdaki kod bloğunda da görüldüğü gibi şifrelenmiş olan dosya, salt, IV ve HMAC değerleri ile yeni bir dosya oluşturuluyor. Ve dosyanın orijinal hali güvenli olarak siliniyor. Silme işlemini kullanım senaryom kapsamında ek bir mekanizma olarak eklemeyi doğru buldum.

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

decrypt\_file\_cbc fonksiyonu şifrelenmiş dosyanın şifresinin çözülmesi için kullanılır. Öncelikle şifrelenmiş dosyada bulunan data okunur. Dosyanın geçerli bir şifrelenmiş dosya olup olmadığının kontrolü gerçekleştirilir. Dosya şifrelemesi esnasında plaintext ile birlikte eklenen salt, IV ve HMAC değerleri ayrılır. CBC modunda çalışan AES şifre çözme algoritması kullanılarak dosya şifresi çözülür. Daha sonra eklenmiş ise padding kaldırma işlemi gerçekleştirilir. Daha sonra aşağıdaki kod bloğunda da görüldüğü gibi orijinal dosya adı şifrelenmiş dosya adı ve path verisi kullanılarak belirlenir. Belirlenen ad ile orijinal dosya oluşturulur ve şifreli versiyon güvenli şekilde silinir.

metin, ekran görüntüsü, yazı tipi, doküman, belge içeren bir resim

Açıklama otomatik olarak oluşturuldu

Programın main kısmı olarak da nitelendirebileceğimiz kısımda kullanıcıdan gerçekleştirmek istediği işlemi seçmesini bekliyorum. Şifreleme işlemi gerçekleştirmek isteyen kullanıcıdan terminal üzerinden “e” inputunu girmesini beklerken, şifre çözme işlemi gerçekleştirecek olan kullanıcıdan terminal üzerinden “d” inputunu girmesini talep ediyorum. Bu bağlamda programı sonlandırmak isteyen kullanıcılar ise “q” inputu girerek programı sonlandırabiliyor. Eğer geçersiz bir input girişi olursa kullanıcılardan geçerli bir input talep ediliyor. Bu işlem “while true” olarak program sonlandırılana kadar gerçekleşmeye devam ediyor. Kullanıcılardan yapmış oldukları seçim doğrultusunda path ve şifre verileri de yine bu kısımda alınıyor.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Programla bazında kodun açıklaması yukarıda görüldüğü gibi kod üzerinden alınan kesitler de kullanılarak gerçekleştirilmiştir.

**Uygulama Akışı:**

Şifreleme (Encrypt):

* Kullanıcı, güçlü bir şifre belirler.
* Dosyanın şifrelenmesi için IV ve salt oluşturulur.
* Anahtar, kullanıcının şifresinden türetilir.
* Dosya AES algoritması ve CBC modu kullanılarak şifrelenir.
* HMAC ile dosyanın bütünlüğü sağlanır.
* Orijinal dosya silinir.

Şifre Çözme (Decrypt):

* Kullanıcı, şifresini girer.
* Şifrelenmiş dosyanın bütünlüğü HMAC ile kontrol edilir.
* Anahtar, kullanıcının şifresinden ve salttan türetilir.
* Dosya AES algoritması ve CBC modu kullanılarak şifre çözülür.
* Padding kaldırılır ve orijinal dosya oluşturulur.
* Şifrelenmiş dosya silinir.

Çıkış (Quit):

* Program sonlandırılır.

Ekran görüntüleri ile daha detaylı olarak incelemek gerekirse.

İlk olarak kullanıcılardan yapmak istedikleri işlemi seçmeleri istenir.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Daha sonra örneğin “e” encrypt işlemi yapmak isteyen kullanıcı için bir şifre belirlemesi istenir.

metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şifre yeterince güçlü değilse format belirtilerek şifre yeniden talep edilir.

ekran görüntüsü, metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Geçerli bir şifre girişi sağlandıktan sonra kullanıcıdan şifrelemek istedikleri dosyanın konumu talep edilir.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Doğru path girişi gerçekleştirildikten sonra dosyan şifrelenir ve kullanıcıdan yeni işlem seçmesi talep edilir.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şifrelenen dosya .enc olarak saklanır ve orijinal versiyon silinir. metin, ekran görüntüsü, meneviş mavisi, logo içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şifrelenmiş dosya üzerinde şifre çözme işlemi gerçekleştirmek istersek bu sefer de decrypt işlemi için “d” girdisi girilir.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Yukarıda görüldüğü gibi hatalı şifre girişi olduğunda kullanıcıdan yeniden işlem seçmesi istenir ve hatalı şifre HMAC olduğu bildirilir.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Doğru şifre girişi gerçekleştirildiğinde ise şifre çözme işlemi gerçekleştirilir ve dosyanın .enc şifreli versiyonu güvenli olarak silinir.

Ekran görüntüsünde de görüldüğü gibi dosyanın orijinal hali yeniden oluşturulmuş olur.  Dosya içeriği okunabilir ve bozunma olmaksızın kullanılabilir bir durumdadır.

“q” girişi ile de program sonlandırılır.

Terminal üzerinde ekran görüntüleri kullanarak görselleştirdiğim Programın çalışma düzeni yukarıda görüldüğü gibidir. Program üzerinde farklı senaryolarda şifrelemeler yapılarak daha sonra talep edilen dosyalar üzerinde istenilen zamanda şifre çözme işlemi gerçekleştirilebilmektedir.

HMAC Doğrulama Hatası:

HMAC doğrulama hatası durumunda kullanıcıya hatayı bildirir.

Şifrelenmiş Dosya Formatı Hatası:

Şifrelenmiş dosyanın formatı uygun değilse hata mesajı gösterir.

Zayıf Şifre Uyarısı:

Kullanıcıdan alınan şifre zayıf ise uyarı mesajı gösterir.

**Uzun Vadede Yapılabilecek Eklemeler:**

Kullanıcı Arayüzünün Geliştirilmesi: Şu anda terminal üzerinde çalışmakta olan bu program daha kullanışlı bir arayüz yazılarak geliştirilebilir. Örneğin şu an path bilgisi el ile girilirken arayüz yardımı ile dosya seçimi yardımı ile path bilgisi otomatik olarak alınabilir.

Anahtar Yönetimi ve Depolama: Şu anda, kullanıcı tarafından belirlenen şifre, anahtarı türetmek için kullanılıyor. Ancak, güvenli bir anahtar yönetimi ve depolama sistemi eklemek daha iyi bir uygulama sağlayabilir.

Dosya Şifreleme ve Şifre Çözme İlerlemesi: Kullanıcıya şifreleme veya şifre çözme işleminin tamamlandığını gösteren bir ilerleme çubuğu veya mesaj ekleyebilirsiniz. Bu, büyük dosyaların şifrelenmesi veya çözülmesi sırasında kullanıcıya beklenen süreyi bildirmenize yardımcı olabilir.

Şifreleme Algoritmalarının Seçimi: Kullanıcıya şifreleme algoritmalarını seçme imkanı verebiliriz. Örneğin, AES dışında farklı bir simetrik şifreleme algoritması kullanma seçeneği sunabiliriz.

Daha Fazla Güvenlik Kontrolü: Kullanıcının güçlü bir şifre girmesini zorlamak ve şifre güvenliği için daha fazla kontroller eklemek, projenin güvenliğini artırabilir.

Günlükleme (Logging): Kullanıcı etkileşimlerini ve şifreleme/şifre çözme işlemlerini günlüklemek, uygulamanın hata ayıklama sürecini kolaylaştırabilir.

Kapsamlı Hata Kontrolü: Şu anda, bir hata oluştuğunda kullanıcıya sadece genel bir hata mesajı veriliyor. Daha detaylı hata mesajları ve kullanıcı dostu hata iletişimleri ekleyerek kullanıcı deneyimini iyileştirebiliriz.

**Sonuç:**

Proje, dosya şifreleme ve şifre çözme işlemlerini kullanıcı dostu bir arayüzle gerçekleştiren güvenli bir Python uygulamasıdır. CBC modunda çalışan AES şifreleme algoritması ve HMAC doğrulama mekanizması, kullanıcıların dosyalarını güvenle şifrelemelerine ve çözmelerine olanak bilinmektedir.

**Tartışma:**

Projeye farklı güvenlik mekanizmaları da eklenebilir ancak senaryom çerçevesinde gerekli gördüğüm mekanizmaları eklemeye özen gösterdim. Başlangıçta eklemiş olduğum ancak sonrasında kaldırdığım mekanizmalardan bir tanesi Şifrelerin ve salt değerlerinin güvenli olarak saklandığı bir mekanizmaydı ancak web tabanlı ya da database içeren bir sistem olmamasından dolayı bu mekanizmadan vazgeçerek bu değerleri dosya içerisinde saklamayı tercih ettim. Bu dosyanın kırılması açısından bir güvenlik açığı oluşturabilir ancak senaryo gereksinimleri gereği bu yaklaşımın daha doğru olduğunu düşündüm.

Proposal kapsamında belirtmiş olduğum bütün problemleri çözdüm. Programlama esnasında bazı hatalar almış olsam da bu hataları yapmış olduğum araştırmalar ve farklı yaklaşımlar yardımı ile çözmeyi başardım.