ONDOKUZ MAYIS ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

ŞİFRE YÖNETİCİSİ UYGULAMASI

PROJE TASARIM RAPORU

YELDA SOY - 19060405

Danışman

Dr. Öğr. Üyesi Meryem Soysaldı Şahin

ARALIK, 2024

TEŞEKKÜR

Çalışmam boyunca yardımını esirgemeyen, öneri ve paylaşımlarıyla projemi en iyi şekilde tamamlamamı sağlayan saygıdeğer Dr. Öğretim Üyesi Meryem Soysaldı Şahin'e katkılarından dolayı teşekkür ederim.

ÖZET

Bilinen ilk şifreleme yöntemi M.Ö askeri, ticari vs. gibi nedenlerle bilgilerin güvende tutulması amacıyla ortaya çıkmıştır. Bilgilerin güvende ve gizli tutulması çok eski çağlardan beri insanlığın ihtiyaçlarından birisi olmuştur. Buradan anlaşılacağı üzere insanlığın şifrelemeye ihtiyaç duyması sadece günümüz şartlarından dolayı ortaya çıkmış değildir. Fakat günümüzde bilgilerin güvence altına alınması için üretilen sistemler gerekli korumayı sağlamak için daha karmaşık teknolojiler ve teknikler kullanmaktadır.

Gelişen teknolojiyle beraber internet kullanımı artmış olup her türlü ihtiyacımız sanal gerçekleşir hale gelmiştir. Kişisel bilgilerimiz için güvenlik tehditleri oluşturmuştur. Bu durum kişisel bilgilerin, hesapların ele geçirilmesi isteği sebebiyle siber suçları arttırmıştır. Özellikle banka hesapları gibi online gerçekleştirilen işlemlerde şifrelerin güçlü ve benzersiz olması büyük önem taşımaktadır. Ancak insanlar güçlü şifre oluşturma konusunda veya farklı hesaplar için birbirinden farklı şifreler kullanma konusunda yetersiz kalmıştır. Bu önemli sorunun çözümünde şifre yöneticileri karşımıza çıkmaktadır.

Bu proje kapsamında literatürdeki önemli şifre yöneticilerinden bahsedilmiştir. Aralarında karşılaştırma yapılmıştır ve ortak özelliklerine ve her birinin öne çıkan özelliklerine yer verilmiştir. Bu kapsamda tasarlanacak şifre yöneticisi uygulamasının kullanıcı dostu arayüze sahip olması, kolay kullanılabilirlik ve yüksek güvenlikli ve benzersiz şifreler oluşturması hedeflenmektedir.

İÇİNDEKİLER

ŞEKİLLER LİSTESİ

ÇİZELGELER LİSTESİ

BÖLÜM: İ

GİRİŞ

1

AMAÇ

Giderek daha çok gelişen teknoloji birçok ihtiyacı da beraberinde getirmiştir ve getirmeye de devam etmektedir. Bundan on yıl öncesine kadar teknoloji bu seviyelerde değildi ve bu seviyelere geleceği hayal bile edilemezdi. Günümüzde tüm ihtiyaçlarımız alışveriş, ödemeler, bankalar, seyahat ve bunun gibi sayamayacağımız her türlü ihtiyaç ya da istekler internet kullanarak birçok uygulama üzerinden sağlanmaktadır. Gelişen bu teknoloji de birçok güvenlik sorununu da beraberinde getirmiştir. Siber suçlar da büyük oranda artışlar görülmektedir. Bu nedenle bilgilerin güvende tutulması eskisine göre zorlaşmış durumdadır ve bu da insanları bazı zorunlu ihtiyaçlara itmektedir.

Şifreler günlük hayatımızın bir parçası haline gelmiştir. Banka hesapları, sosyal medya hesapları, mobil uygulamalar, okul bilgi sistemleri vb. bu tür uygulamalarda hesaplar şifreler ile güvende tutulmaktadır. Her biri için güçlü ve benzersiz şifreler oluşturmak hızla eskiyebilir. Bu durumlarda bazı insanlar yıldönümü, doğum günü tarihleri gibi kolayca hatırlayabilecekleri şifreler oluşturarak bu sorunu çözmüşlerdir. Bazı insanlar ise sadece aynı şifreyi hatırlar ve bu yüzden şifreyi çevrimiçi her yerde kullanmayı tercih etmişlerdir. Bu iki durumda da şifreler için amaçlanan koruma gerçek anlamda sağlanamaz ve bu durum kimlik hırsızlığına açık kapı bırakmaktadır. İşte bu noktada şifre yöneticilerinden bahsedilmektedir.

Şifre yöneticileri güçlü şifreler oluşturma ve bu şifreleri depolama görevi görmektedir. Oluşturduğu şifreler çok uzun ve karmaşık yapıdadır, yani büyük küçük harf, semboller, rakamlar, özel karakterlerin hepsini içerebilir ve bu şifreler 15-20 karakter uzunluğunda olabilir. Daha sonra bu oluşturulan her bir şifre belirlediğiniz bir ana şifre arkasında depolanmaktadır. Bu şekilde farklı giriş bilgilerine erişmek istediğinizde hatırlamanız gereken sadece 1 tane şifre olacaktır. Bir insan olarak, şifre yöneticilerini kullanmadan güçlü şifreler oluşturmak, bu şifreleri güvende tutmak ve her birinin akılda kalması mümkün olmayacaktır. Bu nedenle bu gibi sorunlar şifre yöneticilerini her insan için zorunlu bir ihtiyaç haline getirmiştir. Şimdilik kullanımları çok yaygın olmasa da zamanla herkesin vazgeçilmezi olacağı açıkça öngörülebilir durumdadır.

Piyasada halihazırda birçok şifre yöneticisi örneği görülmektedir. Çoğunlukla hepsi aynı şekilde çalışmaktadır ancak ufak özellik farklılıkları bulunmaktadır. Tasarlanacak şifre yöneticisi uygulamasının güvenlik, hızlılık ve kolay kullanılabilirlik açısından elverişli olması ve şifreleri oluşturup depolama, aynı zamanda otomatik doldurma özelliklerine sahip olması amaçlanmıştır ve bu süreçte birçok algoritmaya ihtiyaç duyulmuştur.

Sonuç olarak bu projenin amacı kullanıcılar için şifrelerini yönetebilecekleri kullanımı basit ama işlevi yüksek bir yol sunan güvenli, hızlı bir şifre yönetim sistemi sağlamak olacaktır. Bu sağlanırken güvenlik ön planda tutulacak ve piyasadaki eksiklikler tamamlanmaya çalışılacaktır. Şifreleme algoritmaları, teknikleri ve güvenlik standartlarına dikkat edilecektir. Ana amaç parola çeşitliliğinin sağlanması, parola karmaşıklığının arttırılması ve güvenli bir şekilde depolanması olacaktır.

2

LİTERATÜR ÖZETİ

Şifre yöneticileri kullanıcı şifreleri oluşturmak ve bu şifreleri güvenli bir şekilde depolamak için kullanılan yazılımlardır [6]. Kullanıcı sadece bir ana şifre hatırlamak zorundadır. Şifreler genellikle yerel makine ve sunucularda saklanır ama bazı durumlarda bulut sunucularda depolanmaktadır. Piyasadaki şifre yöneticileri bazısı açık kaynak kodlu olacak şekilde sunulmuştur ve bu durumun avantajları ve dezavantajları bulunmaktadır.

Kullanıcılar bir çok farklı hesap için bir çok farklı şifreye ihtiyaç duymaktadır [27]. Bunları hatırlamak yeterince zorlayıcı bir durum olabileceğinden dolayı insanlar karmaşık şifre kullanmaktan kaçınmıştır. Bu durum insanların, CIA üçlüsünün en zayıf halkası olmasına neden olmuştur. Güvenlik ihlallerinin büyük bir kısmı insanların yaptığı hatalardan kaynaklanmaktadır. Şifre yöneticilerinin tercih edilmesinin nedeni, bu güvenlik sorunlarından kurtulup şifreleri güvenle saklayıp hesapları güven altında tutmaktır.

Şifre yöneticileri yazılım ve donanım tabanlı olmak üzere 2 alt başlık altında sınıflandırılmaktadır [10]. Yazılım tabanlı yöneticilerinde bulut sistemi yaygın ve kullanışlı olmakla birlikte dünyanın her yanından erişebilirlik sağladığı için daha çok tercih edilmektedir. Çevrimiçi şifre yöneticileri açık kaynaklı ve kapalı olmak üzere yine 2 sınıfta toplanmaktadır. (Açık kaynaklı şifre yöneticileri: Podlock, Bitwarden), (kapalı olanlar ise Lastpass, Roboform) gibi örnekler verilmiştir. Açık kaynaklı şifre yöneticileri herkesin kodu inceleyip,

güvenlik açıklarını tespit ederek bunları bildirmesi, uygulamanın gelişmesine katkıda bulunabilmektedir. Bunun yanında dezavantajları da bulunmakta olup açık kaynak olduğu için saldırganlara karşı korunmasız duruma gelebilmektedir. Donanım tabanlı şifre yöneticileri güvenlik açısından yazılım tabanlılara göre üstün çıksa da kullanım açısından (cihaz taşıma zorluğu, çalınma vb.) geride kalmaktadır.

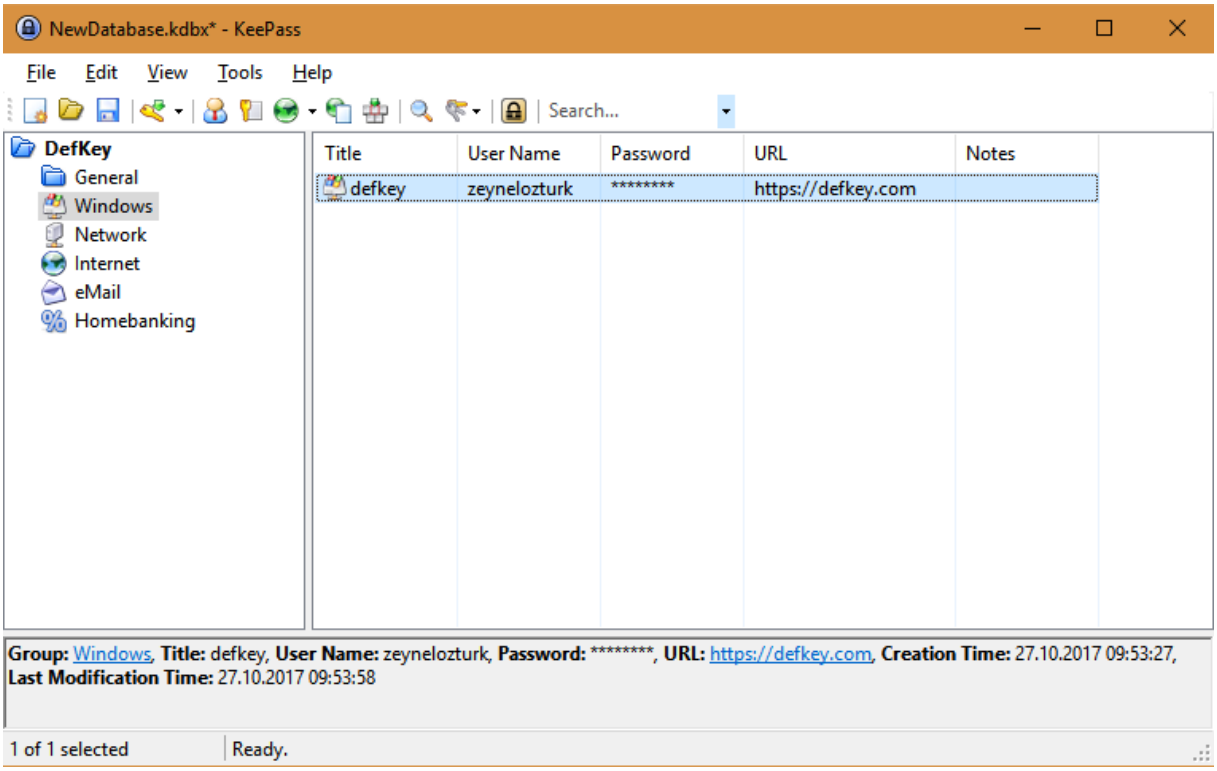
Şifre yöneticilerinin tarihine gidildiği zaman aslında çok da eskiye dayanmadığı görülmektedir [9]. Dominik Reichl tarafından geliştirilen Keepas ilk kez 2003 yılında piyasaya sürülmüştür. Piyasaya ücretli şekilde kullanıma sunulmuş birçok şifre yöneticisi örneğinden bahsedilmektedir. Şifre yöneticileri genel olarak şifreleri saklamak, şifre oluşturmak ve bunları yaparken de gizlilik ve güvenliği esas alan uygulamalardır. Araştırmanın bu kısmında literatürdeki bilgilerden yararlanılmıştır ve şifre yöneticilerinde bulunan eksiklikler ortaya çıkarmaya çalışılmıştır.

2.1 *Yazılım Tabanlı Şifre Yöneticileri*

Şifreler bilgisayarlarda, mobil cihazlarda ya da bulut sunucularda saklanmaktadır. Şifre yöneticileri Web, mobil uygulama veya masaüstü yazılımlar sayesinde kullanılabilmektedir. Kullanıcı şifrelerini cihazları arasında kolayca senkronize edebilir. Şifreler bulutlarda şifreli olarak saklanmış olsa dahi olası saldırılara açık hale gelebilmektedir. Yazılım tabanlı şifre yöneticileri donanım tabanlılara göre daha esnek ve erişim özgürlüğü sağlamaktadır fakat güvenlik ve çevrimdışı kullanım konusunda geride kalmaktadır. Temelde hatırlanması gereken bir tane master şifredir [1]. Daha sonra bu şifre sayısız şifrelere erişime olanak sağlayacaktır. Araştırmanın bu kısmında literatürdeki yazılım tabanlı şifre yöneticileri incelenmiştir.

2.1.1 *Keepass*

Keepass şifre yöneticisi açık kaynaklı bir uygulamadır [6]. Dil İngilizcedir ama 45 farklı dili de destekleyecek şekilde oluşturulmuştur. Keepass AES, Chacha20 ve Twofish şifreleme algoritmaları kullanılarak tasarlanmıştır. Keepass veritabanı AES ve Twofish algoritmaları kullanılarak şifrelenmiştir. Keepass'in kullandığı algoritmalar, büyük küçük harf, tekrarlanan diziler ve sabit sayılar açısından kuvvetli algoritmalar kullanmıştır [1]. Keepass Windows, Linux ve MacOS sistemlerinde çalışmaktadır [2]. Keepass'te hatırlanması gereken sadece bir master şifredir. Bu master şifre altında yüzlerce şifre saklanmaktadır. Keepass veritabanları her kaydettiğinde, şifreleri veritabanlarının bir sorun teşkil etmemesi için rastgele başlatma vektörü oluşturmuştur. Şekil 1'de Keepass ana menüsü gösterilmiştir.



Şekil 1: Keepass ana menü

Şifrelenen yalnızca şifre değildir. Tüm veritabanı şifrelenmektedir. SHA-256 fonksiyonu kullanılarak şifre karma hale getirilir. Şifrenin güvenliği için ise 256 bit anahtar uzunluğuna sahip Twofish algoritması kullanılmıştır. Veritabanı içeriğini güvence altına almak için kullanılan MAC (Message Authentication Code) (Mesaj Doğrulama Kodu) fonksiyonu ise çift faktörlü kimlik doğrulama ile desteklenmiştir [3].

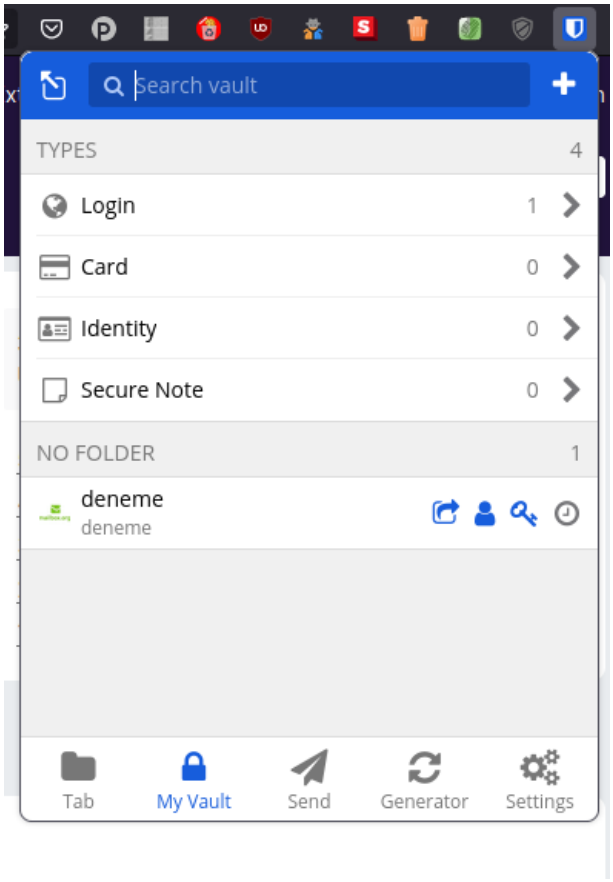
Veri tabanı ana menüde açılır bir menü olarak oluşturulmuştur. Ana şifre girildikten sonra şifrenin tahmin kalitesi kontrol edilmiş ve güvenlik geliştirme konusunda geliştirmeler yapılmıştır. Veritabanını kilitlemek için kullanılan dosya anahtar dosyasıdır. Bu dosyanın hatırlanması gerekmez ancak güvenlik sağlamak zorlaşabilmektedir. Keepass veritabanında Argon2 algoritması kullanılmıştır. Argon2 algoritması bir çeşit hashleme algoritmasıdır. Keepass Argon2'nun birkaç sürümünü desteklemiş ve GPU (Graphics Processing Unit)/ASIC (Aplication-Specific İntegrated Circuit) saldırılarına karşı iyi bir koruma sağlamıştır.

Keepass'in kodları üzerinde detaylı olarak güvenlik testleri yapılmıştır. Bu kod incelemelerinin sonucunda yüksek riskli güvenlik açığı bulunamamıştır fakat güvenli kodlama üzerine 4 risk tespit edilmiştir. Bu riskler:

- Doğru tür dönüşümü sağlamak
- Bir nesne için yeterli bellek sağlamak
- Güvenlik ihlaline neden olacak sistem çağrılarından kaçınmak
- Eski fonksiyonları kullanmamak (ör. rand())

Keepass bulut senkronizasyonu sunmamaktadır [10]. Fakat birçok farklı işletim sistemi ve mobil cihazlar için resmi olmayan Keepass sürümleri bulunmaktadır. Ayrıca Keepass'in resmi kaynak kodları SourceForge adlı siteden indirilebilmektedir.

2.1.2 *Bitwarden*



Şekil 2: Bitwarden ekranı

Bitwarden açık kaynaklı bir şifre yöneticisidir. Tarayıcıda veya Windows, Linux ve MacOS gibi işletim sistemlerinde uygulama olarak çalışmaktadır. Ayrıca bir mobil app olarak IOS ve Android cihazlara indirme özelliği bulunmaktadır. 40 farklı dil desteleyecek şekilde tasarlanmıştır. Şekil 2'de Bitwarden arayüzü verilmiştir. Bitwarden crowdin hizmetlerini kullanmıştır. Crowdin çevrimçi bir çeviri platformudur ve bulut tabanlı olmasından dolayı kullanıcılar farklı yerlerden erişerek yazılımın farklı dillere çevrilmesine yardımda bulunmaktadır. Bitwarden veri iletimi için AES-256 bit şifreleme algoritmasını kullanmıştır. Ana şifreyi hasheleyerek veriyi sunucularına iletmeden önce şifreler [4]. Ayrıca ana şifreden bir anahtar üretmek için SHA-256 algoritmasını kullanmıştır. Bitwarden Google Authenticator

gibi kimlik doğrulayıcı uygulamalar kullanarak veya e-posta gibi çeşitli hizmetler aracılığıyla çift farktörlü kimlik doğrulamayı destekler biçimde tasarlanmıştır. Keepass kullanıcı verilerini yalnızca yerel olarak saklamasına izin verirken Bitwarden bulut tabanlı senkronizasyon seçenekleri de sunmaktadır. Bu Bitwarden'ın Keepass'e göre artı bir özelliği olarak sınıflandırılmıştır.

Lunr adlı bir programlama kütüphanesi kullanmıştır [4]. Bitwarden'da Giriş, Kart ve Kimlik olmak üzere dört kategori bulunmaktadır. Veriler dışarı aktarılabilirdir ancak aktarılan veri şifreli olmadığı için güvenli bir durum değildir. Bitwarden şifrelerin güvenliğini sağlamak için bir çok araç içermektedir. Yerleşik bir şifre oluşturucu sayesinde şifrenin hangi karakter içerip, hangi uzunlukta olacağı daha detaylı bir şekilde belirlenmiştir.

Bitwarden'da aynı anda yalnızca bir tane veritabanı olabilir. Bu veritabanı hesap ile senkronize edilmiştir [5]. Veritabanı AES-256 ve PBKDF2 algoritmaları kullanılarak korunmaktadır. Ana anahtar e-posta adresi ile birlikte tuzlama yapılmış ve hashlenmiştir. Bu işlem veri sunucuya iletilmeden önce yapılmıştır. Daha sonra sunucu bu hashlenmiş veriyi alarak güvenliği daha da arttırmak için tekrar tuzlama ve hashleme işlemleri yaparak veritabanında saklanmaktadır. Bu işlem her oturum açıldığında tekrarlanmaktadır.

2.1.3 *Padlock*

Padlock Martin Kleinschrodt tarafından minimalist, açık kaynaklı bir şifre yöneticisi olarak geliştirilmiştir [9]. Tamamı Javascript, Html ve CSS kullanılarak yazılmıştır. Padlock Windows, IOS, MacOS, Android gibi işletim sistemlerinde kullanılmaktadır. Linux'da kullanılabilir değildir ancak gelecekte kullanılacağına dair yazılar bulunmaktadır. Padlock'da ilginç olan bir ozellik de uygulamada kullanıcı etkinliği tespit edilmezse bir dakika içerisinde

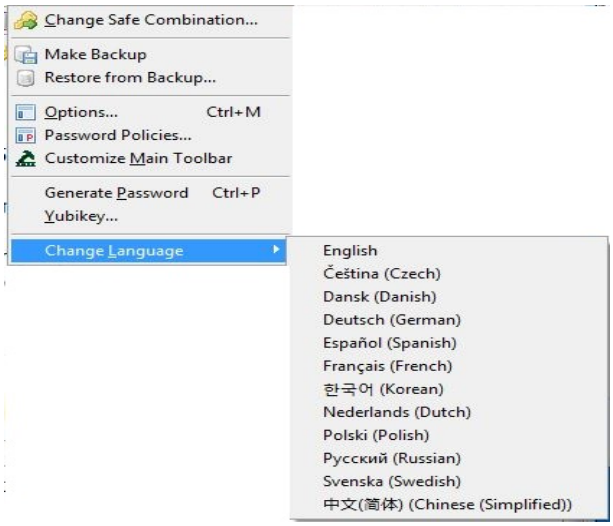
kasa dışı kalmasıdır. Bu özellik değiştirmeye açıktır. Yani kullanıcı isterse bu özelliği devre dışı bırakmaktadır.

Padlock'un çok kapsamlı güvenlik testleri bulunmaktadır. Penetrasyon test ekipleri ve penetrasyon testleri için kullanılan depolar ile birlikte bazı güvenlik açıkları raporlanmıştır. Bu rapor edilen güvenlik açıkları Tap-jacking ve Man-in-the-middle (Ortadaki adam saldırıları) şeklinde sıralanmaktadır. API istekleri sırasında kimlik doğrulama belirteçlerinin ifşa olmasından dolayı ortadaki adam saldırıları meydana gelebilmektedir. Bir diğer rapolanan saldırı ise mobil cihazlarda meydana gelen Dos (Denial of service) hizmet engelleme saldırılarıdır. Saldırılar sonucu kullanıcı uygulamayı sıfırlamak zorunda kalabilir ve bu durumda kullanıcının tüm bilgilerini kaybetmesine neden olmaktadır. Özellikle tüm bilgiler bulutta saklanıyorsa bu bilgileri geri kurtarmanın yolu yoktur. Padlock'da diğer şifre yöneticilerinde olduğu gibi otomatik doldurma özelliği bulunmamaktadır. Bir şifre yöneticisinde bu gibi durumların yaşanması Padlock'un güvenlik açısından geride kaldığını göstermektedir. Özetle Padlock kullanımı basit iyi bir arayüze sahip bir şifre yöneticisi olsa da güvenlik konusunda yetersiz kalmıştır [7].

2.1.4 *Password Safe*

Password Safe, açık kaynaklı bir şifre yöneticisidir [14]. Windows, Linux ve Android'de kullanılabilmektedir. Şu anda IOS işletim sisteminde desteklenmiyor fakat, Password Safe'in bir ikizi denilebilecek bir uygulaması MacOS store ve Appstore'da bulunmaktadır. Windows kurulumunda ki seçilen green sürümünde şuanda 14 farklı dil desteklenmektedir fakat gerekli

dil olması durumunda 'Geliştiriciyle iletişime geç' şeklinde bir seçenek bulunmaktadır. Şekil 3'te Password Safe uygulamasında destelenen bazı diller ve arayüzü gösterilmiştir.



Şekil 3: Password Safe görünüm

Password Safe 256 bit anahtar uzunluğuna sahip, Twofish algoritmasını kullanmıştır. Password veriyi şifrelemede kullanılacak şifreleme anahatarını ve verilerin değiştirilip değiştirilmediğini sağlamak için kullanılan mac anahtarını dosya başlığında saklamıştır [6]. Bu sayede veri tabanı güvenliği sağlanmıştır. Ayrıca Password Safe'te 2FA (Two-factor authentication) (Çift faktörlü kimlik doğrulama) desteklenmektedir.

Uygulamaya başladıktan sonra bir database yüklemek veya database yaratmak gerekmektedir. Veritabanı yaratılırken erişim için yeni bir master şifre oluşturulması gerekmektedir. Uygulama customusible'dır yani uygulama arayüz açısından özelleştirilebilirdir. Password Safe arayüz açısından biraz karmaşıktır ancak eklenen HTML yadrım klavuzu sayesinde uygulamanın rahat kullanımı açısından engel olacak bu durumun önüne rahatça geçilmiştir.

Password Safe şifre güvenliğini sağlamak için bir kaç çözümün üstünde durmuştur. Şifreler için bir giriş politikası, şifre oluşturmak için kurallar, veritabanı yedeği oluşturulması gibi örnekler sıralanmaktadır. Burada özellikle uygulama veritabanı yedeklenmesi konusunda 3 ila 5 yedeklenme önerisinde bulunmuştur çünkü olası veritabanı bozulması veya başka sorunlarla

karşılaşıldığında, veritabanı bu yedeklerden geri yüklenebilmiştir. Veritabanı dışa aktarılmayı destekler şekilde tasarlanmıştır ancak bu verinin korunmasız bir kopyasını oluşturacaktır ki bu kullanıcının dikkatli olmasını gerektirecek bir unsurdur.

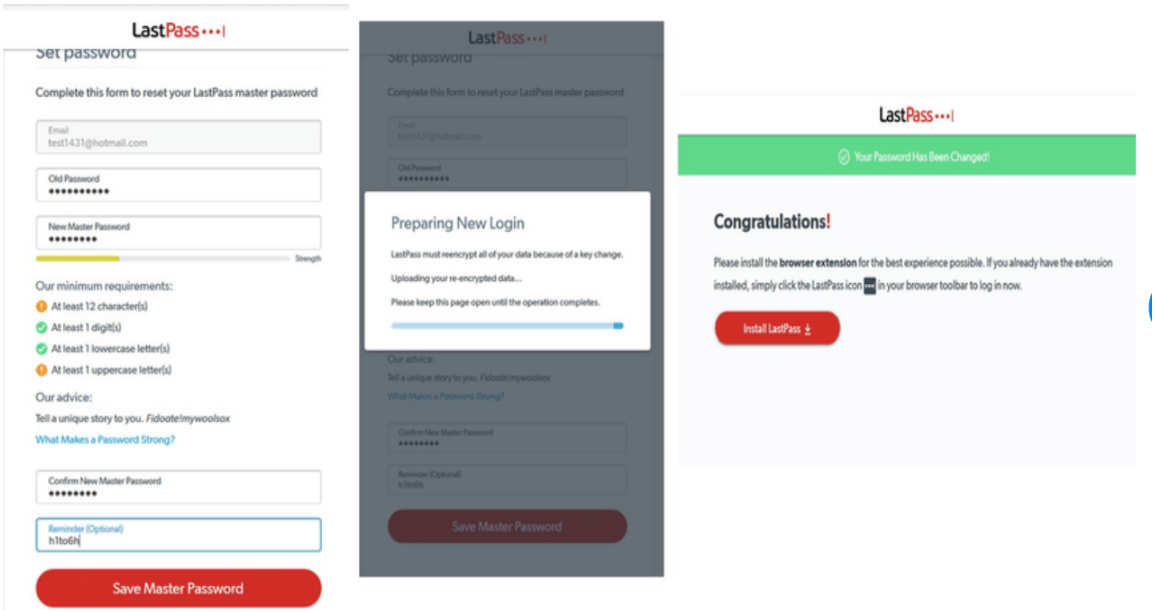
2.1.5 *Literatürdeki Diğer Yazılım Tabanlı Şifre Yöneticileri*

1Password ücretli olarak kullanıma sunulmuş yazılım tabanlı bir şifre yöneticisidir. Açık kaynaklı değildir. Şifreleme işlemlerinde AES-128 şifrelemesini CBC (Cipher Block Chaining) (Şifre Bloğu Zincirleme) modunda kullanmaktadır. Veritabanında birden fazla dosya depolamaktadır. Dosyalar JSON formatında saklanmaktadır. 1Password düşük ve yüksek olmak üzere 2 farklı güvenlik seviyesini kullanıcının her kaydı için seçmesine izin verir. Şifrelerin kullanım amacına göre örneğin kullanıcı adı ve şifre gibi çok önemli bilgiler 'yüksek güvenlik seviyesi' olarak ifade edilmektedir. 1Password veritabanı türü için yapılan güvenlik testlerinde IND-CDBA (İndistinguishability under Chosen Database Attack) ve MAL-CDBA (Manipulation under Chosen Database Attack) gibi oyunlarda saldırılara açık güvenlik açıkları saptanmıştır [6].

Roboform 2000 yılında ortaya çıkmış yazılım tabanlı şifre yöneticisidir. Açık kaynakları mevcut olmamakla beraber yıllık ödemeli premium abonelik seçenekleri bulunmaktadır. Kayıtlı parolaların platformlar arası erişimine izin verir. 2FA özelliği sağlamaktadır. Güçlü parola oluşturma, kullanıcı girişi, şifre senkronizasyonu gibi özellikler sunmakla birlikte çevrimdışı kullanımı desteklemektedir [12]. Arayüz kullanımı basit değildir fakat kullandıkça kullanımı kolaylaşmaktadır. VPN ek hizmetini sunmamaktadır. Roboform şifreleme için AES-256 şifreleme kullanmaktadır. Ayrıca yük şifrelemesi için bazı algoritmalar (AES, Blowfish, DES, RC6, Triple-DES) arasında seçime olanak sağlamaktadır. Roboform şifre biçimi

güvenlik kontrollerinde Advr ve Advrw saldırılarına karşı yeterli güvenlik sağlayamadığından tehditlere karşı savunmasız kalmıştır [6].

LastPass bulut tabanlı bir şifre yöneticisidir [26]. Günümüzde en çok kullanılan şifre yöneticileri arasında ilk sırada yer almaktadır. Kullanımı ücretsizdir. LastPass web tarayıcısı eklentisi ve aynı zamanda kendi web tarayıcısı sayesinde kullanıcı için kullanım çeşitliliği sağlamaktadır. Otomatik doldurma (padding) özelliğine sahiptir. Kullanıcı deneyimini etkileyen bazı özellikler bulunmamaktadır. Örneğin geri alma fonksiyonu (undo) özelliğine sahip değildir. Bu yüzden kullanıcı giriş yaptıktan sonra önemli bir değişiklik yapıp kaydettiği zaman undo fonksiyonunu kullanamaz dolayısıyla yapılan değişiklikler geri alınamayabilir. Master password (ana parola) oluştururken gereksinimleri sağlamakta yetersizdir. Şekil 4'te LastPass arayüzü verilmiştir. Arayüz kullanımı kolay kullanıcı dostu hizmet sağlar ve rastgele üretilen parolalar sayesinde güvenliği artırmaktadır fakat değiştirilen eski parolaların silinmeden kaydedilmesi güvenlik sorunlarına kapı aralamaktadır. Çizelge 1'de bu bölümde ele alınan şifre yöneticilerinin karşılaştırılması yapılmıştır.



Şekil 4: LastPass uygulaması

Özellik	1Password	Roboform	LastPass
Çıkış Yılı	2006	2000	2008
Tür	Yazılım Tabanlı	Yazılım Tabanlı	Yazılım Tabanlı
Açık Kaynak	Değil	Değil	Değil
Güvenlik Seviyesi	Düşük ve Yüksek	Yüksek	Yüksek
Şifreleme	AES-128	AES-256	AES-256
Çevrimdışı Kullanım	Var	Var	Yok
Şifrenin Güveniliği	IND-CDBA ve MAL-CDBA saldırılarına karşı yetersiz savunma	Advr ve Advrw saldırılarına karşı yetersiz savunma	Güvenlik açıkları tespit edilmedi
Arayüz	Kullanıcı dostu	Kullanımı ilk başta zor	Kolay, kullanıcı dostu
Bulut Depolama	Var	Yok	Var
Master Parola Güvenliği	Güçlü, dikkatli seçilmeli	Güçlü, dikkatli seçilmeli	Zayıf, gerekli güvenliği sağlamakta yetersiz kalabilir

Çizelge 1: Yazılım tabanlı bazı şifre yöneticilerinin karşılaştırılması

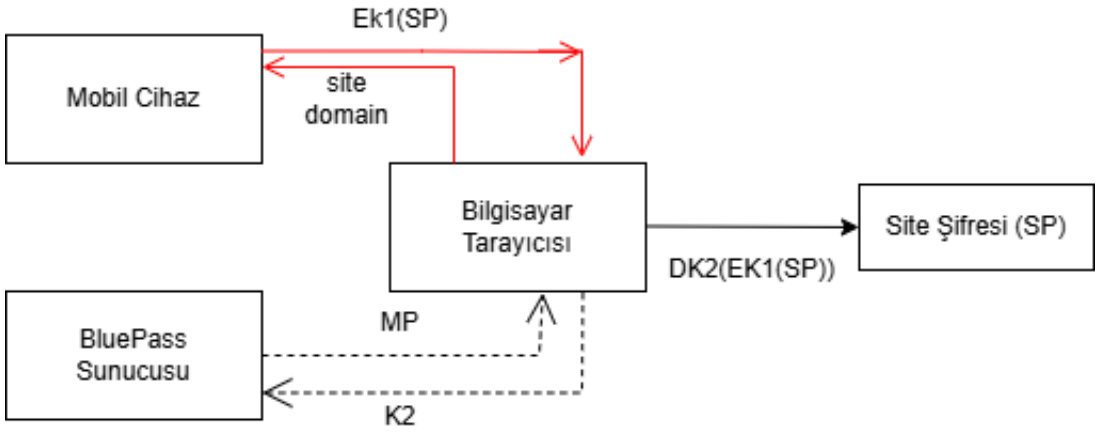
2.2 *Donanım Tabanlı Şifre Yöneticileri*

Yazılım tabanlı şifre yöneticilerinde olduğu gibi donanım tabanlı şifre yöneticilerinde de amaç, kullanıcılar için güçlü şifreler oluşturmak ve bu şifreleri güvenli bir şekilde depolamaktır [18]. Yazılım tabanlı şifre yöneticilerinden farkı şifrelerin bilgisayarın sabit diskinde saklanmamasıdır. Bundan dolayı şifrelere ek bir güvenlik özelliği sağlamaktadır. Ancak her ne kadar güvenlik konusunda ön plana çıksa da, taşıma zorunluluğu, kaybolma riskleri gibi özelliklerinden dolayı günümüzde daha az tercih edilmesine neden olmaktadır. Araştırmanın bu kısmında literatürdeki donanım tabanlı şifre yöneticilerinden bahsedilmiştir.

2.2.1 *Bluepass*

Bluepass açık kaynak kodlu, donanım tabanlı, eller serbest bir şifre yöneticisidir. Bluepass çift faktörlü bir kimlik doğrulama kullanmıştır. Bluepass 2 ilkenin üzerinde durmuştur [11]. Bunlar Bluepass sunucusu, (K1,K2) RSA (Rivest-Shamir-Adleman) anahtar çifteridir. Bluepass sunucusunun görevi kullanıcı anahtarlarını kaydetmek ve bilgisayarlara anahtar dağıtmaktır. K1,K2 anahtar çifti veri şifreleme ve şifre çözmede kullanılan anahtar çiftleridir. K1 yalnızca mobil cihazlarda kullanılırken, K2 anahtarları Bluepass sunucusunda yeniden dağıtmak üzere saklanmaktadır. Kullanıcının güvenilir bulduğu bir bilgisayarda (örneğin kullanıcının kişisel bilgisayarı) uzun süre saklanabilmektedir. Kullanıcı Bluepass mobil uygulamasını cihazına yükler bu uygulama sayesinde şifrelenmiş kullanıcı şifreleri saklanır ancak kullanıcı bluetooh aracılığıyla ancak bu verileri kişisel bilgisayarına iletebilmektedir. Şekil 5'te Bluepass kimlik doğrulama süreci şematik olarak gösterilmiştir. Kırmızı çizgiler Bluetooth, kesikli çizgiler ise interneti temsil etmektedir.

Tüm şifre yöneticilerinde olduğu gibi Bluepass'de de hatırlanması gereken bir master şifredir. Master şifre kullanılarak, kullanıcı kendi şifre çözme anahtarı olan K2'yi almaktadır. Bluepass çevrimiçi olarak hizmet verebilmek için şifreler kullanmıştır. Bu şifreler SP (site şifresi) olarak adlandırılmıştır. Bu şifreler K1 anahtarı ile şifrelenmekte olup, Bluepass mobil uygulamasında saklanmaktadır.



Şekil 5: Bluepass kimlik doğrulama

Bluepass çalışma prensibi olarak temelde 3 aşamada incelenmiştir. Kayıt, yapılandırma ve kimlik doğrulama olarak sıralanmıştır. Yapılandırma aşamasında K2 anahtarının güvenli bilgisayarda kalıcı ya da geçici olarak saklanmaktadır. Uygulamanın çalışma prensibinde eğer girilen şifre doğruysa sunuculardan K2'ye yanıt gitmektedir ve bluetooth bağlantısı kullanılarak istek oluştulur daha sonra mobil cihaz şifreli site şifresini yanıt olarak göndermektedir. Burada önemli nokta K1 ve K2 RSA anahtar çiftleridir bu yüzden K2, K1 ile şifrelenmiş içeriği çözmek için kullanılmıştır [10]. Bluepass'de bluetooh kullanıldığı için olası saldırılarda, saldırganın cihaza yakın olması gerekmektedir. Bu durum Bluepass'in kullanım güvenirliğini göstermiştir. Ancak bluetooh bağlantısı olmadığında Bluepass kullanılamadığı için kullanılabilirlik açısından yeteriz kalmıştır.

Bluepass Mobil Uygulaması: BluePass mobil uygulaması android platformunda çalışan ve gelen bluetooth parçalarını dinlemek için bir Bluepass servisi başlatmaktadır [11]. Bu servis kullanıcının ilgili web sayfasını şeffaf bir şekilde kimlik doğrulama yapmasını sağlamaktadır. Bluepass uygulamasında arka planda çalışan Bluepass servisinin durumunu 'çalışıyor' veya 'askıya alında' olarak gösteren bir arayüz tasarlanmıştır. Kullanıcı farklı butonlar kullanarak bu durumları değiştirebilmektedir.

Mobil cihaz ve bilgisayar arasında iletişim kurmak için RFCOMM (Radio frequency communication) (Radio frekans iletişimi) Bluetooh protokolü kullanmıştır. Bunun nedeni bu ağın bir çok işletim sistemi tarafından desteklenmesidir. Android RFCOMM bağlantılarının güvenli mod ve güvensiz mod olacak şekilde 2 modunu desteklemektedir. Güvenli mod RFCOMM bağlantısı kurulmadan önce başarılı bir eşleştirme gerektirirken Güvensiz mod bu gereksinim olmaksızın bağlantıya izin vermektedir. Güvenli mod ek bir şifreleme katmanı ekler ancak Bluepass iletişimi temelde bluetooh güvenliğine dayanmadığı için (K1,K2) anahtar çiftleri ile güvenlik altında alınmıştır. Sonuç olarak güveli modda eşleştirme bir kez yapılır ve eşleşen cihazlar silinmedikçe tekrarına gerek yoktur.

2.2.2 *Pico*

Pico şifre yöneticisi token tabanlı olarak tasarlanmıştır [10]. Pico bir fiziksel bir cihaz olarak taşınabilir bir şifre yöneticisidir ve QR kodlarını tarayabilen web sayfalarına ve uygulamalara kimlik doğrulaması yapabilen 2D görsellere sahiptir. Stajano'nun çalışmalarında kimlik doğrulama yönteminin 'belleksiz, ölçeklenebilir ve güvenli' verilen bu 3 ana kriteri sağlaması gerektiğini belirtmiştir. Şifrenin belleksiz olması kullanıcının bilgileri aklında tutması gerekmediği, ölçeklenebilir olması uygulamanın kullanıcı sayısı ne kadar artarsa artsın

performansının aynı şekilde kalması gerektiğini ve güvenli olmasının ise şifrelerin güvenli bir şekilde saklanması gerektiği şeklinde açıklamıştır. Pico şifre yöneticisi bu 3 kriteri sağlamaktadır. Ayrıca Pico'nun tasarımında güvenlik önlemleri ve özellikleri de bulunmaktadır.

Pico tasarımında 2D kodları tarayabilen bir kameraya sahiptir. Bilgisayar ile pico arasındaki iletişim rasdyolar aracılığıyla gerçekleştirilmektedir. Bilgilerin tutulduğu çok sayıda hafıza yuvasına sahiptir. Giriş yapılmak istenilen uygulama kendi imzalı 2D görsellerini sunmaktadır. Pico bu uygulamının doğruluğunun kanıtlanması için özel anahtarlar ile mesaj imzalamasını istemektedir. Uygulamanın mesajı başarılı bir şekilde çözüp Pico 'ya yanıt verdiği durumda oturum başlatılıp kullanıcı giriş yapmaktadır. Stajano ayrıca Pico için cihazı şarj etmeyi ve depolama alanına yedeklemeyi mümkün kılacak bir istasyon önermiştir.

Pico ilk satın alındığında veriler henüz yüklenmemiştir. Kullanıcı ya sıfırdan yükler ya da yedekleme yapmaktadır. Pico yazılım ve donanımsal olarak açık kaynaklı olarak sunulmuştur. Fakat cihazın şu anda satışı bulunmamaktadır. Özetle Pico güvenli ve kullanıcı dostu olmasından dolayı kimlik doğrulama çözümü olarak önerilmektedir.

2.2.3 *Literatürdeki Diğer Donanım Tabanlı Şifre Yöneticileri*

SPHINX'in kullandığı yaklaşımdaki amaç çok basit şifreleri (pwd) bile kullanmasına olanak tanımaktır [10]. Bu şifre (pwd) web sitesine girilirken @ karakteri ile başlamalıdır. Bu şifre girildikten sonra tarayıcı otomatik bir şekilde alan adını (domain) ikinci cihaza yani mobil cihaza göndermektedir. Bu şekilde cihazda SPHINX uygulaması çalışmaktadır. Bu uygulama aldığı bilgiden yararlanarak rastgele şekilde güvenliği yüksek bir şifre oluşturur. Her site için farklı rwd (rastgele oluşturulmuş şifre) hesaplanmaktadır. Bu şifre (rwd) basit şifrenin (pwd)'nin yerine geçer. Bu şekilde şifre güvenirliliği arttırılmaktadır. SPHINX mobil

uygulaması Ortada Adam Saldırıları'na (Man In The Middle) karşı FK-PTR adlı protokolü kullanmaktadır. OPRF şifreleme yöntemi sayesinde mobil cihazın şifreyi öğrenmesinin önüne geçer ve pwd veya rwd depolanmaz. Kısaca SPHINX mobil cihazlar aracılığıyla basit şifreleri güçlü şifrelere dönüştürerek güvenliği arttırmayı amaçlayan sistemdir fakat kullanılabilirlik ve geri uyumluluk açısından sınıfta kalır.

PUF (Physically Unclonable Function) (Fiziksel Klonlanamaz Fonksiyonlar) benzersiz özellikleri temsil etmektedir [10]. PUF'ların 2 önemli özelliği bulunmaktadır. Birinci özellik Uniformluk (0 ve 1'lerin dengeli üretilmesi) ve diğer özelliği ise benzersizlik (farklı IC'lerin benzersiz yanıtlar üretmesi) olarak açıklanmaktadır. PUF parolaların güvenliklerini arttırmakta kullanılmaktadır. Kullanıcı parolası ve alan adı bir karma fonksiyona gönderildiğinde bu fonksiyon kullanılamaz yerine PUF'tan oluşturulan farklı bir çıkış kullanılmaktadır. Bu çıkış her donanımda farklı olduğundan parolanın saldırganlar tarafından ele geçirilmesi ancak saldırgan fiziksel cihazın kendisine erişim sağladığında mümkün olmaktadır. PUF sisteminin kullanılabilmesi için var olan şifre veritabanlarında değişiklikler yapılmalıdır. Bunun nedeni PUF çıkışlarının kullanılacak olmasıdır. Bu durum şifre güvenliğini arttırmaktadır. Ancak şifrelerin kurtarılmasındaki zorluk, geri kullanım konusundaki zayıflık ve ek donanım ihtiyaçları PUF tabanlı sistemlerin dezavantajlarındandır.

Bölüm 2.1 ve Bölüm 2.2'de anlatılan yazılım ve donanım tabanlı şifre yöneticilerinin karşılaştırılması Çizelge 2'de gösterilmiştir.

Özellikler	KeePass	Bitwarden	Padlock	Password Safe	BluePass	Pico
Tür	Yazılım Tabanlı	Yazılım Tabanlı	Yazılım Tabanlı	Yazılım Tabanlı	Donanım Tabanlı	Donanım Tabanlı
Açık Kaynak	Evet	Evet	Evet	Evet	Evet	Evet
Veri Saklama	Yerel	Bulut ve Yerel	Yerel	Yerel	Yerel (Bluetooth ile PC'ye iletim)	Yerel (Cihazda saklama)
Şifreleme Algoritmaları	AES, Twofish, Chacha20	AES-256, SHA-256, PBKDF2	-	Twofish	RSA anahtar çiftleri	-
Bulut Senkronizayonu	Hayır	Evet	Hayır	Evet	Evet	Hayır
Şifre Güvenliği	Yüksek	Yüksek	Orta	Yüksek	Yüksek	Yüksek
Kullanıcı Arayüzü	Karmaşık, Özelleştirilebilir	Kullanıcı dostu, Modern	Basit, Minimalist	Karmaşık, Özelleştirilebilir	Basit	Basit
2FA (Çift faktörlü doğrulama)	Yubikey	Evet, Google Authenticator destekli	Hayır	Yubikey	Evet, Bluetooth ile	Evet, 2D QR kodları
Ücretsiz	Evet	Evet	Evet	Evet	Evet	Evet

Çizelge 2: Şifre Yöneticilerinin Karşılaştırılması

BÖLÜM: İİ

MATERYAL

3

ALGORİTMALARA BAKIŞ

Bir şifre yöneticisi uygulaması tasarlamak için birçok algoritmadan yardım alınması gerekmektedir. Buradaki amaç oluşturulacak uygulamanın güvenilir, hızlı ve karmaşık yapıda olmasıdır. Şifre yöneticilerinin genel kullanım amacı şifre oluşturmak, bu şifreleri depolamak ve gerektiği zaman rahatça kullanmaktır. Bundan dolayı uygulamada kullanılacak algoritmaların şifreleme, şifre çözme, hashleme, şifreleri güvenli şekilde saklama ve bunları yaparken güvenilir ve hızlı bir şekilde yapması ana hedef olacaktır. Bu sayılan birçok kriter için birçok ayrı ayrı algoritmaya ihtiyaç duyulacaktır. Bölüm 3.1 şifreleme algoritmalarını inceleyecek, nasıl çalıştıklarını görecek ve aralarında karşılaştırma yapılacaktır.

3.1 *Şifreleme Algoritmaları*

Şifreleme algoritmaları, birçok matematiksel işlemlerle şifre üzerinde karmaşıklık yaratarak bu şifrelerin güvenli bir biçimde saklanmasında kullanılmaktadır. Kullanıcının kullandığı birçok hesap için güçlü ve karmaşık şifreler oluşturarak bu şifrelerin güvenli bir şekilde depolanmasında yardımcı olmaktadır.

Simetrik Şifreleme	Asimetrik Şifreleme
Tek anahtar kullanılır (gizli anahtar).	İki anahtar kullanılır (bir özel, bir açık anahtar).
Anahtar gönderiminde güvenlik riskleri bulunmaktadır.	Özel anahtar yalnızca sahibi tarafından bilinmelidir.
Daha hızlıdır.	Daha yavaştır.
Veri şifreleme, dosya şifreleme, VPN vs. kullanılır.	Dijital imza, e-posta şifreleme SSL, TSL bağlantılarında kullanılır.
Anahtar kaybolursa şifre çözülür.	Şifreleme ve şifre çözme için farklı anahtarlar kullanıldığı için daha güvenlidir.

Çizelge 3: Simetrik ve asimetrik şifreleme

Özetle şifreleme algoritmaları şifreler üzerinde güvenlik oluşturmayı amaçlamaktadır. Asimetrik ve Simetrik şifreleme olmak üzere ikiye ayrılmıştır. Tablo 3'de Asimetrik ve Simetrik şifreleme özellikleri kısaca gösterilmiştir.

3.1.1 *RSA Şifreleme Algoritması*

RSA asimetrik şifreleme algoritması Rivest, Shamir ve Adleman tarafından 1977 yılında bulunmuştur [23]. Burada RSA'nın asimetrik şifre grubuna dahil olmasını şu şekilde açıklanmaktadır. Asimetrik şifreleme aslında açık anahtarlı şifreleme demektir. Asimetrik şifrelemede açık ve kapalı olmak üzere iki tane anahtar bulunur. Şifreleme ve şifre çözme için kullanılan anahtarlar birbirinden farklıdır. Şifre anahtarının herkese açık olması gerektiğinden bu algoritmaya açık anahtarlı şifreleme denilmektedir. Bundan dolayı kullanıcının açık anahtarı ile şifrelenen bir metin sadece bu kullanıcıya ait olan özel anahtar ile metnin şifresi

çözülebilmektedir.

RSA şifreleme algoritması nerelerde kullanılır?

- Şifreleme ve dijital imza
- SSL, PTC protokollerinde
- Web sitelerinde güvenlik sertifikasi için

RSA nasıl çalışır? RSA algoritmasının temelinde asal sayılar yatmaktadır. RSA'nın güvenirliği seçilen tam sayıların büyüklüğüyle orantılıdır ama bu işlemlerin yavaş olması da RSA'nın dezavantajı olarak değerlendirilmektedir. Ayrıca büyük tam sayılarla işlem yapmak zor olacağı için büyük tam sayılar seçilmemelidir.

RSA'nin temel adımları aşağıda verilmiştir.

1) İki büyük asal sayı seçimi:

Seçilecek sayılar p ve q; ne kadar büyük olursa algoritmanın karmaşıklığı o kadar artar. Bu da şifrenin kırılmasını zorlaştıracaktır.

2) Modül Hesaplama(n):

p ve q kullanılarak (iki asal sayı) n değeri hesaplanır. Bu n değeri hem şifrelemede hem de şifre çözmede kullanılır. Eşitlik 1'de verilmiştir.

$$n = p \times q \tag{1}$$

3) Euler'in Totient Fonksiyonu $\phi(n)$:

$\phi(n)$, n'den küçük olan ve n ile aralarında asal sayı olan sayıların sayısını verir. Eşitlik 2'de görebilirsiniz.

$$\phi(n) = (p-1) \times (q-1) \tag{2}$$

4) Şifreleme anahtarı (e) seçimi:

Hesaplanan totient fonksiyonu değeri $(\phi(n))$ ile aralarında asal olan öyle bir sayı alınır ki $1 < e < (\phi(n))$ olmalıdır. Bu seçilen e sayısı public anahtar olarak ilan edilmektedir.

5) Şifre çözme anahtarı (d) hesaplama:

private key değeri d ile temsil edilecek olursa Şekil 3'teki formül kullanılarak hesaplanır.

$$d \cdot e \equiv 1 \mod \phi(n) \tag{3}$$

Bu d değeri gizli(private) şifre olarak saklanmaktadır.

Şifreleme ve şifre çözme işlemleri:

- Şifreleme:[$C \equiv M^e$ modn], M mesajı, C şifrelenmiş veriyi temsil eder.
- Şifre çözme: [$M \equiv C^d$ modn] d özel anahtar kullanılarak şifre çözülmektedir.

3.1.2 *Chacha20 Algoritması*

Daniel J.Bernstein tarafından 2008 yılında akış şifreleyici olarak geliştirilmiştir [13]. Akış şifreleyici, şifreleri sürekli akışlar halinde şifrelemesi olarak açıklanır. Sürekli bir rastgele bit akışı üretmektedir. Verileri aynı anahtarı kullanarak şifreler ve deşifrelemektedir. Simetrik bir şifreleme algoritmasıdır ve 256 bit anahtar kullanır. Mobil cihaz ve gömülü sistemlerde daha hızlıdır. Chacha20 algoritması hız ve güvenlik arasında denge kurmayı amaçlamıştır. Yüksek paralelleştirilebilirliği sayesinde çok çekirdekli CPU'lar ve diğer yüksek performanslı hesaplama platformlarına kolayca entegre edilebilmektedir. Chacha20 algoritması diğer algoritmalarla karşılaştırıldığında kullanımı oldukça kolaydır. Salsa20 şifre ailesinin bir çeşidi olan şifreler grubuna dahildir.

- **Chacha20'nin Bir Kaç Basit Adımı:**

- **İlk adımı** anahtar üretmektir. Kullanıcıdan alınan bir anahtar ve rastgele üretilen 96 bit nonce kullanarak (buradaki nonce açılımı "number used once" bir kez kullanılan ve rastgele uretilen bir sayıdır. Nonce tekrar edilen değerlerin kullanılmasını önler bu sayede şifreleme işlemi, saldırganlara karşı verileri tahmin etmek daha zor olacağı için daha güvenilir hale gelmiş olur.)
  256 bitlik bir anahtar oluşturulur.

- **2. adımda** ise anahtar nonce ve anahtar değerini kullanarak şifreleyicinin durumunu ayarladığı başlatma aşamasıdır.

- **3. adımında** chacha20'nin her veri bloğuna, mevcut şifreleme durumu ile şifrelediği veri şifreleme aşamasıdır. Bu durum her blok işlendiğinde değiştirilir.

- **Son adımında** çıktı (output) aşamasıdır. Şifreli metin düz metni veri şifreleme aşamasının çıktısı ile XOR'layarak üretir. (Burada XOR'un görevi iki veriyi karıştırmaktır. Bu da veriyi şifrelemek ve gizlemek için etkili bir yöntemdir.)

Chacha20 algoritmasında başlangıç durumu Çizelge 4'te verildiği üzere 16 adet 32-bit kelime olarak temsil edilmektedir.
1. Satır algoritmanın adını ve anahtar üretme sürecini belirtmektedir. 32 bitlerden oluşur. Bu satır "expand 32-byte k" sabit dizesidir. 2. ve 3. satırlarda ise toplamda 256 bit anahtar bilgisini içermektedir. 4. satırda ise verilmiş olduğu üzere ilk blokta 32 bit sayaç, diğer 3 blokta ise 96-bit nonce bilgisi bulunmaktadır.

Chacha20 her bir turda 512 bit anahtar akışı üretir [19]. Bu şekilde 512 bitlik düz metin bloğunu şifreler. Sonuç olarak Çizelge 4'teki matris chacha20 algoritmasında şifreleme ve şifre çözme işleminde kullanılır. Her bir işlemde güncellenmesi gerekir.

Çizelge 4: 4x4 matris yapısı

cons	cons	cons	cons
key	key	key	key
key	key	key	key
counter	nonce	nonce	nonce

Chacha20 algoritmasında anahtarın yeterince rastgele üretilmesi çok önemlidir. Diğer şifreleme algoritmalarına bakınca RSA ve EC gibi, değerlerinin belirli bir matematiksel koşulla sağlanmasına gerek yoktur.

Dikkat edilmesi gereken bazı noktalar: Anahtar üretilirken CSPNRG veya HSM modu kullanılabilmektedir. Bunlar rastgele sayı üretme sürecinde yardımcı olacaktır. Her yeni işlemde, yeni bir rastgelelik ile yeni anahtar üretmek büyük önem taşımaktadır.
Chacha20 şifreleme algoritmasında algoritmanın sabit uzunluğundan (256 bit) şifreyi direkt anahtar olarak kullanmak mümkün değildir. Kullanıcının seçeceği şifre üzerinde bazı sınırlamalara yol açar. Bu yüzden bir anahtar üretme algoritması, şifreden chacha20 uyumlu anahtar üretmektedir. Bu durumda PBKDF2 kullanılabilecek bir anahtar üretme fonksiyonudur.

Chacha20 algoritmasının kullanıldığı bazı alanlar:

- Güvenli mesajlaşma
- Vpn'ler
- Dosya şifreleme
- Web güvenliği

3.1.3 *AES Şifreleme Algoritması*

2001 yılında NIST (National Institute of Standards and Technology) tarafından elektronik şifreleme için oluşturulmuştur. AES algoritması simetrik bir blok şifreleme algoritmasıdır [20]. Simetrik şifreleme olmasının nedeni hem şifreleme hem de şifre çözme için aynı anahtarı kullanmasından kaynaklanır. Bu durum algoritmanın hızlı olmasını ve basit çalışmasını sağlar. Fakat anahtar seçimi kritik bir öneme sahiptir.

AES algoritmasında şifreleme yapmak için farklı uzunlukta anahtarlar kullanılabilir [21]. Bunlar 128, 192, 256 bit uzunluğunda olabilir yani 3 durum karşımıza çıkar. Ancak şifreleyeceği veri 128 bit uzunluğundadır ve 4x4 matrislere bölünür. Bu matrisin her bir bloğu 8 bit boyutundadır ve her satır ve sütun 32 bitten oluşur. Bu oluşan matris durum matrisi olarak adlandırılmaktadır. Çizelge 5'te durum matris yapısı gösterilmiştir. Şifrelenecek mesaj ve anahtar için durum matrisleri üzerinde işlemler yapılır ve sürekli güncellenir.

AES şifreleme birkaç aşamada (round) gerçekleşir ve round sayısı anahtar uzunluğuna bağlı olarak değişir.

- 128 bitlik anahtar - 10 tur
- 192 bitlik anahtar - 12 tur
- 256 bitlik anahtar - 14 tur

olacak şekilde gerçekleşir. AES şifreleme algoritmasında sadece orijinal anahtar kullanılmaz. Her turda yeni anahtar oluşturulmalıdır. 4 ana adımdan oluşur [22].

AES ŞİFRELEME ADIMLARI:

- **1) SubBytes:** Döngünün ilk gerçekleştirilen eylemidir. Aynı zamanda da tek doğrusal olmayan işlemidir. Bu adımda her byte'ın S-box tablosuna göre ikame edilmesiyle

Çizelge 5: AES durum anahtarı

a(0,0)	a(0,1)	a(0,2)	a(0,3)
a(1,0)	a(1,1)	a(1,2)	a(1,3)
a(2,0)	a(2,1)	a(2,2)	a(2,3)
a(3,0)	a(3,1)	a(3,2)	a(3,3)

gerçekleşir. S-Box durum matrisinin elemanları onaltılık tabana göre oluştuğu için 16x16 lık bir tablodur. Örneğin binary değer olarak 1010,1101 değelerini ele alalım. Bu değerlerin onluk tabanda karşılığı 10,13 şeklindedir. Bu durumda durum matrisindeki değeri, S-box tablosunun 10. satır ve 13. sütunla karşılık gelen değerle yer değiştirme işlemi subByte adımını açıklar. Bu adım sonunda yeni bir durum matrisi elde edilmiş olur. Bu işlem veriyi karmaşıklaştırarak saldırılara karşı koruma sağlar.

- **2) ShiftRows:** Bu adımda matrisin satırları üzerinde değişiklik yapılır. Matrisin her satırı belirli bir sayıda sola kaydırılır. 1. satır hariç, 2. satır 1 sola, 3. satır 2 sola ve 4. satır 3 sola olacak şekilde kaydırılır. Bu adımda veri bloklarının yerleri değiştiği için veri karmaşıklığı arttırılmış olur.

- **3) MixCloumns:** Her sütun sabit bir polinomiyal dönüşümle karıştırılır. Yani bu işlem, her sütunun sabit bir matrisle çarpılması işleminden oluşur. Bu adım son turda uygulanmaz. Eşitlik 6'da bu işlemin nasıl yapılacağına dair bir örnek verilmiştir.

- **4) AddRoundkey:** Her tur sonunda, geçerli veri bloğu tur anahtarı işlemi gerçekleştirilir. Her turda yeni anahtar üretildiği için tur sayısı kadar anahtar yeni gereklidir. İlk belirlenen anahtar en başta matrisle XOR işlemi yapılır. Geriye kalan anahtarlar ilk anahtar yardımıyla üretilir ve her tur sonunda oluşan mesaj üzerine eklenir.

Buraya kadar yapılan işlemler 9 kez tekrar eder ta ki 10. işlemde Mixcloumn adımı yoktur. Bu adımlardan sonra şifrelenmiş metin ortaya çıkar. AES algoritmasının adımları ve genel şeması şekil 6'da verilmiştir.

Şifre Çözme:

Şifreleme adımları tersten yapıldığı zaman şifre çözme işlemi tamamlanmış olur. Turdaki aşamalar kolayca geri alınabilmektedir. Şifre çözme işlemi için aynı anahtar kullanılmalıdır. Şifrelenmiş veride padding (doldurma) işlemi varsa, çözümleme sırasında 'unpad' fonksiyonu ile bu doldurmanın çıkarılması gereklidir.

AES şifreleme algoritmasında **Galois alanından** bahsedilmektedir. Galois alanı $GF(p^n)$ şeklinde gösterilmektedir. Genel olarak sonlu alan kriptografide kullanılmaktadır. Bu sonlu alanda toplama, çarpma, bölme gibi matematiksel işlemler polinomlar üzerinden yapılmaktadır. Galois alanında her elemanın bir tersi bulunmaktadır. Bu özellikle, Aes'in şifre çözme işlemleri için önemlidir. Galois alanında işlemler 8 bitlik polinomlar kullanılarak gerçekleştirilmektedir.

Galois alanı AES'in birçok aşamasında kullanılmaktadır. Bu aşamalar MixCloumns, S-box ve Ters S-box işlemleridir. Örneğin Mixcloumns aşamasında her sütundaki byte'lar Galois alanındaki matematiksel işlemler kullanılarak çarpılır ve yeni değer elde edilir bu şekilde karmaşıklık artırılmaktadır. Özellikle şifre çözme sürecinde ters alma işlemleri Galois alanında gerçekleştirilmektedir. Özetle AES, karmaşıklık, güvenlik gibi bir çok koşulu arttırmak için yaptığı polinomsal matematiksel işlemleri Galois alanında gerçekleştirmektedir.

Mixcloumns aşamasında çarpılacak matris sabittir ve Galois alanındaki polinomlar kullanılmılştır. Dönüşüm matrisi 4'te verilmiştir.

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \tag{4}$$

Örnek bir işlem; Üzerinde işlem yapılacak matris 5 olsun. (İşlem yapmak için ikilik tabanda yazmak işlemi kolaylaştırmaktadır. Hexadecimal değer verilmiştir.)

$$\begin{bmatrix} 03 & b4 & 04 & 02 \\ 02 & 81 & 01 & 02 \\ 55 & bf & 08 & 02 \\ 30 & a0 & 01 & 02 \end{bmatrix} \tag{5}$$

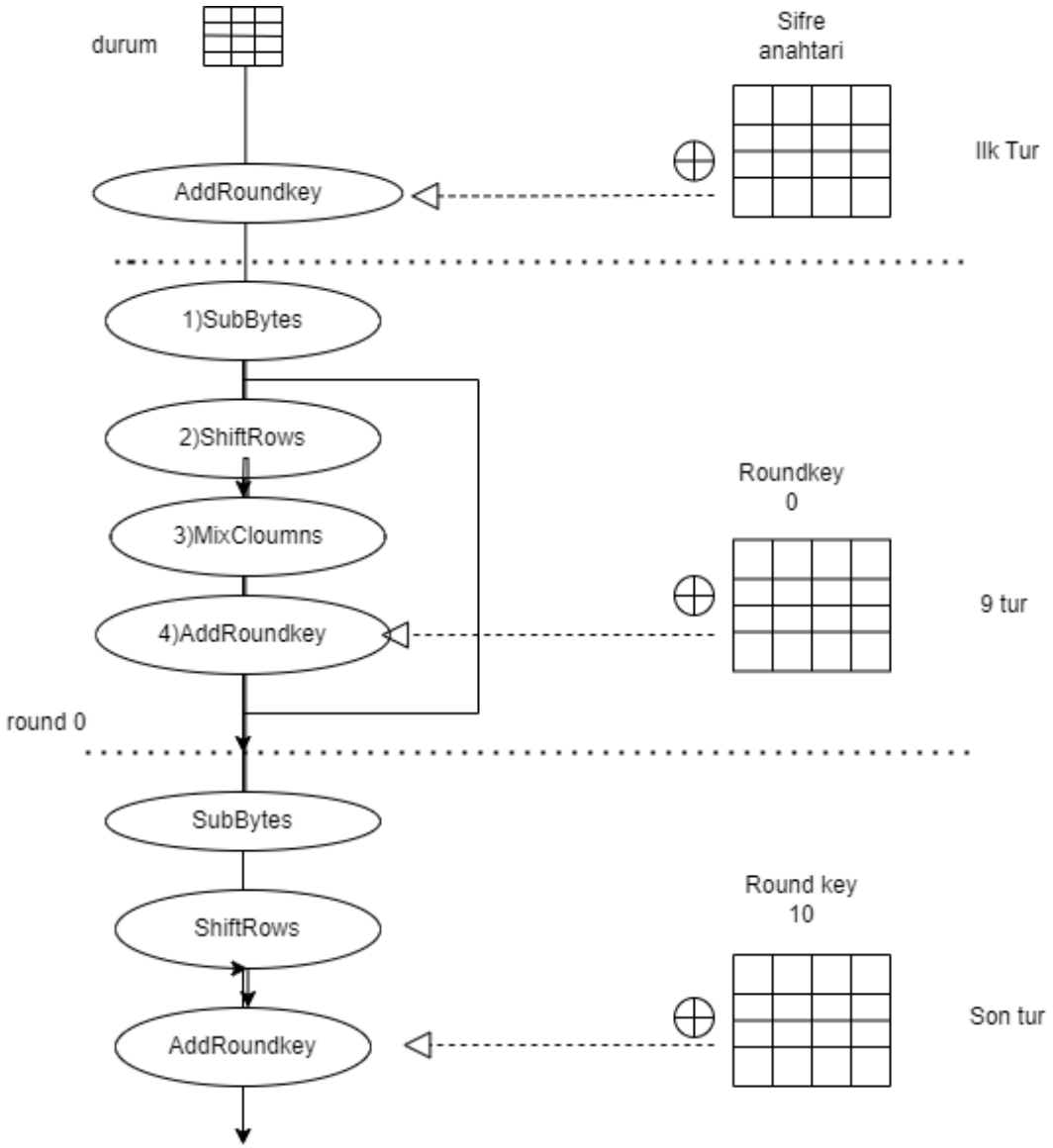
Mixcloumn adımını gerçekleştirmek için Eşitlik 5'de verilerin matrisin her sütunundaki byte'lar, Eşitlik 4 matrisi ile çarpılır. Her sütun için tekrar eden bu işlemlerden sonra yeni durum matrisi oluşmuştur.

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \cdot \begin{bmatrix} 03 \\ 02 \\ 55 \\ 30 \end{bmatrix} \tag{6}$$

$$\begin{aligned} (2 \cdot 03) \oplus (3 \cdot 02) \oplus (1 \cdot 55) \oplus (1 \cdot 30) \\ 2.03 = 06 \\ 3.02 = 06 \\ 1.55 = 30 \end{aligned} \tag{7}$$

$$06 \oplus 06 \oplus 55 \oplus 30 = (06 \oplus 06 = 00) \oplus 55 \oplus 30 = 55 \oplus 30 = 45 \tag{8}$$

Eşitlik 6'da ilk sütun için yapılacak işlemler verilmiştir. İlk sütun ile matrisin birinci sütunu çarpıldığı zaman ilk eleman bulunmaktadır. Eşitlik 7'de Galois alanında çarpım gösterilmiştir. Daha sonra XOR (toplama işlemi) (8) uygulanır. Bu işlem sonucu bulunan 45 yeni matrisin ilk sütunun ilk elemanıdır. Bu işlemler her sütunun diğer satırları için de devam eder ve ilk sütun elde edilmektedir. Aynı işlemler'e devam edildiğinde yeni matris elde edilmektedir.



Şekil 6: AES algoritması genel şeması

3.2 *Hash (Özet) Algoritmaları*

Hash (özetleme) rastgele uzunlukta girilen bir veriyi (dosya, mesaj, şifre, vs.) sabit uzunlukta çıktıya dönüştüren matematiksel bir fonksiyon veya algoritmadır. Bu dönüşümü yaparken algoritmik işlemler kullanmaktadır. Bu fonksiyon tek yönlü çalışmaktadır. Bu durumda kaynak veriye ulaşmak mümkün olmayacaktır. Hash fonksiyonu her zaman sabit uzunlukta bir çıktı üretir. Bu durum verinin güvenliğini arttırır çünkü şifrenin kırılması zorlaşacaktır.

Kriptografiye göre Hash algoritmalarının çıktılarının aşağıda verilen 3 özelliği sağlaması gerekmektedir [24].

- **Ön-İmaj direnci:** Hashing algortimasının tek yönlü çalışması gerektiğini belitmektedir. Verilen değerden orjinal veriye erişim sağlanamayacağından dolayı saldırılara engel olmaktadır.
- **İkinci ön-imaj direnci:** Verilen bir mesajın aynı hash değerini üreten başka bir mesaj oluşturmak yoluyla kırılmasını, bu mesajın saldırganlar tarafından bulunmasını zorlaştırmaktır.
- **Çakışma direnci:** Her mesajın birbirinden farklı 12 hash değerine sahip olması gerektiğini ve aynı hash değerine sahip iki mesajın bulunmasının ancak çok çok uzun süreye ve çok büyük bir hafızaya sahip olan bir bilgisayarla sağlanabileceği belirtilmiştir. Bu durumun gerçekleşmesi yıllar süreceğinden gerçekleşmesi zor olarak ifade edilmiştir.

İdeal bir hash algoritmasının sağlaması gereken 4 özellik aşağıda verilmiştir.

- Hash hesaplamak kolay olmalıdır.
- Mesajın Hash'e göre düzenlenmesi zor olmalıdır.

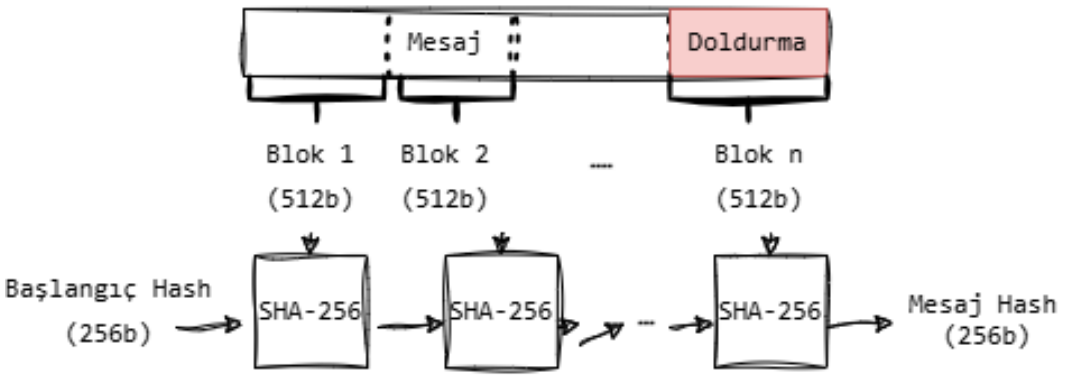
- Mesajın değişmesi zor olmalıdır ve bu şekilde Hash'in değişmemesi sağlanmalıdır.
- Aynı Hash'e sahip iki farklı mesajı oluşturmanın zor olması gerekir.

Sistemin kırılmasını zorlaştırmak için Hash uzunluğu yüksek tutulmalıdır. Hash ne kadar uzun olursa sistemi kırmak o kadar zorlaşacaktır.

3.2.1 *SHA-256 Algoritması*

SHA-256 algoritması SHA2 (Secure Hashing Algorithm 2)'ye göre oluşturulmuştur. Adında 256 olmasının nedeni, algoritmanın girdi boyutuna bağlı kalmaksızın girdiyi 256 bitlik ve 64 karakter olacak şekilde yeniden boyutlandırmasıdır. Çok büyük mesajlar için veri 512 bitlik bloklara bölünmektedir ve son blok eğer ki 512 bitten küçükse doldurma işlemi ile hash işleminden önce mesaj sonuna ek bitler eklenir. Bu şekilde mesaj uzunluğunun 512 bitlik boyutunun tam katları olmasını sağlanmaktadır [25]. Uzun bir mesaj için hash hesaplama Şekil 7'de gösterilmiştir.

SHA-256 algoritması veri blokları için hash değerlerini hesaplarken ara değerleri tek tek hesaplamaktadır. Örneğin veri bloğunun hash sonucu bir sonraki veri bloğunun hash hesaplamasında kullanılmaktadır.



Şekil 7: Uzun mesaj için SHA-256 oluşturma aşamaları

BÖLÜM: İİİ

YÖNTEM

4

YÖNTEM

Bu projede tasarlanacak şifre yöneticisi uygulamasının mobil uygulama olarak geliştirilmesi amaçlanmıştır. Araştırma sonuçlarına göre şifre yöneticilerinin web, masaüstü ya da mobil uygulama olarak veya her üçü içinde çalışan şifre yöneticileri örneklerine rastlanmıştır. Günümüzde yazılım tabanlı şifre yöneticilerinin daha çok tercih edilmesinden ve donanım tabanlı olanların kullanılabilirlik açısından geride kalmasından dolayı yazılım tabanlı bir şifre yöneticisi geliştirilmeye karar verilmiştir.

4.1 *Uygulama Tasarlanırken Kullanılacak Teknolojiler*

Uygulamanın frontend (Bölüm 4.1.1) ve backend (Bölüm 4.1.2) kısmında kullanılması planlanan teknolojiler ve veri şifreleme için kullanılması planlanan algoritmalar nedenleriyle birlikte verilmiştir (Bölüm 4.1.3).

4.1.1 *Frontend*

Uygulama arayüzü flutter dili kullanılarak tasarlanması hedeflenmiştir. Flutter seçilmesinin nedeni Flutter'ın hem IOS hem de Andorid Platformları için sadece tek kod tabanı kullanılarak uygulama geliştirme sağlamasıdır [15]. Flutter widgetları sayesinde arayüz tasarlamayı kolaylaştırır. Flutter seçilmesinin bir diğer nedeni oluşturulacak uygulamanın arayüz açısından şık, kolay anlaşılır, kolay kullanılabilir ve kullanıcı dostu olmasını sağlamaktır. Flutter dilini kullanmanın avantajları:

- Tek bir kod tabanı kullanılarak IOS ve Andorid ortamlarında çalışmasını sağlar bu sayede zamandan tasarruf sağlanır.
- Özelleştirilebilir Widget'lar sayesinde kullanıcı dostu ve şık bir arayüz oluşturmaya olanak sağlamaktadır.
- Çok sayıda açık kaynak kütüphaneye sahiptir bu sayede olası problemlere çözümü kolaylaştırmaktadır.

4.1.2 *Backend*

Backend kısmında Phyton dilinden yaralanılacaktır. Phyton, API geliştirme için yavaş olmayan güvenli yollar sunmaktadır [16]. Şifre verilerinin güvenli olarak çekilmesini sağlamak için Phyton framework'leri (Flask veya Django vb.) kullanarak güvenli API'ler oluşturulacaktır. Ayrıca AES-256 gibi algoritmaları kullanmak için uygun kütüphanelere sahiptir. Kullanıcı yönetimi, şifre sıfırlama, iki faktörlü kimlik doğrulama gibi işlemler Phyton dili ile kolayca uygulanmaktadır. Python dilinde AES-256 şifrelemesi kolayca uygulanabilir.

Veritabanı MySQL kullanırak oluşturulması hedeflenmiştir. MySQL hızlı, güvenilir ve kolay bir kullanım sağlamaktadır [17]. Kullanım ortamı çok geniştir. Veritabanı kaydı ve sorgulamasında güçlüdür ve özellikle verilerin güvenli bir şekilde saklanmasını sağlaması MySQL tercih etmemizin en büyük nedenidir. AES-256 MySQL ile entegre çalışmaktadır.

4.1.3 *Veri Şifreleme*

Veritabanında saklanacak olan şifreler AES-256 şifreleme algoritması kullanılarak şifrelenip saklanması hedeflenmiştir. AES-256 algoritmasının seçilmesinin nedeni 256 bitlik anahtar uzunluğu ile diğer algoritmalarla karşılaştırıldığında günümüzde kullanılan en güvenli şifreleme algoritmalarından biri olmasıdır. Uygulamada kullanıcının birden çok belki 100'lerce şifresi saklayacak şekilde oluşturulması hedeflenmektedir bundan dolayı olası saldırılara karşı şifrenin çok güçlü algoritmalarla saklanması ve şifrelenmesi sağlanmalıdır. AES-256 algoritması bir çok matematiksel adımlara ve karıştırma işlemlerine sahip olmasından dolayı şifre çözme çok uzun sürer ve modern bilgisayalar kullanılarak çözülmesi neredeyse imkansızdır bu sayede olası saldırılara karşı şifreler güvende tutulması sağlanmış olacaktır. Tasarlanacak şifre yöneticisi şifreleri depolamadan önce AES-256 ile şifreleyecektir ve sadece kullanıcı giriş yaptığı zaman çözülecektir.

Şifreleri korumak için ek bir güvenlik olan 2FA yöntemi de uygulamaya dahil edilecektir. 2FA yöntemi kullanıcı adı ve şifrenin kimlik doğrulama için yeterli olmayıp SMS veya farklı bir yolla alınan kod ile ancak kimlik doğrulamayı sağlamaktadır. Bu teknoloji sayesinde hesabın çalınması gibi durumlara karşı önlem alınmış olacaktır ve uygulamanın güvenlik seviyesi arttırılmış olacaktır.

4.2 *Şifre Yönetimi Ve Veri Depolama*

Uygulama oluşturulurken ana hedef kullanıcının güçlü ve benzersiz şifreler oluşturmasını sağlamak (bu şifreler manuel oluşturulabileceği gibi uygulama tarafından rastgele, güvenli de oluşturulması amaçlanmaktadır) ve birden çok şifre depolamaya olanak sağlayan bir veritabanı oluşturmak olacaktır.

Kullanıcı verileri hem bulut hem de yerel olarak saklanabilecektir. Bulut depolama için Firebase servisinden yararlanılacaktır. Yerel depolamda ise şifreler AES algoritması ile şifrelenip veritabanında saklanacaktır. Bu iki yoldan birini seçmek kullanıcı kararına bırakılacaktır. Sonuç olarak bu proje yöntemi özet olarak Çizelge 6'da amaç, hedef ve sonuç şeklinde gösterilmiştir.

Amaç	Hedef	Sonuç
Şifre Yöneticisi Uygulaması Geliştirme	Mobil uygulama olarak şifre yöneticisi tasarımı	Kullanıcıların şifre oluşturup güvenli şekilde depolaması sağlanır
Frontend	Flutter kullanarak uygulamanın arayüzünü tasarlamak	Hem IOS hem de Android ortamda çalışan şık, kullanıcı arayüz
Backend	Phyton kullanarak güvenli API geliştirme	Güvenli kullanıcı yönetimi ve şifre işlemleri yapılabilir
Veri Şifreleme	AES-256 algoritması kullanılarak şifreleme	Kullanıcı şifreleri güvenli bir şekilde şifrelenip saklanır
Güvenlik	2FA (Çift Faktörlü Kimlik Doğrulama) kullanmak	Güüvenlik arttırılır, şifreler daha güvenlihalde saklanır
Veri Depolama	Bulut (Firebase ile) ve yerel depolama seçenekleri	Kullanıcı verileri bulut veya yerel olarak güvenli bir şekilde saklanabilir

Çizelge 6: Uygulama amaç, hedef, sonuç

BÖLÜM: İV

SONUÇ

5

SONUÇ

Sonuç olarak bu projeye literatürdeki şifre yöneticilerinin araştırılmasıyla başlanmıştır. Aralarında karşılaştırma yapılmış ve özellikleriyle birlikte raporlanmıştır. Şifreleme ve güvenlik için kullanılan algoritmalar özellikleriyle ve çalışma aşamalarıyla birlikte raporlanmıştır. Şifre yöneticilerinde eksik bulunan ya da birçok şifre yöneticisinin kendi içinde barındırdığı fakat tek bir şifre yöneticisinde toplanmayan özelliklerin tek bir şifre yöneticisinde bulunmadığı farkedilmiştir. Bu özellikleri tek bir yerde toplamak ve güvenlik problemlerinden arınmış sağlam, kullanıcı dostu arayüze sahip bir şifre yöneticisi tasarlamak için gerekli algoritma seçilmiş ve kullanılacak teknolojiler yöntem kısmında verilmiştir. Sonuç olarak hedeflenen şifre yöneticisi uygulaması kullanıcı hesaplarını güvende tutmak için şifreler adına etkili bir çözüm sunmaktadır. Kullanılacak algoritmalar ve ek güvenlik hizmetleri ile hayatı kolaylaştıracak şifre yöneticisi mobil uygulaması tasarımı yapılmıştır. Bu uygulama mobil cihazlarda güvenli şifre yönetimini sağlayacaktır.

KAYNAKÇA

[1] Dominik Reichl,Password Quality Estimation*Keepass*, https://keepass.info/help/kb/pw_quality_est.html.

[2] Dominik Reichl Keepass Helpcenter *Keepass*, https://keepass.info/help/base/index.html.

[3] Dominik Reichl Keepass Security *Keepass*, https://keepass.info/help/base/security.html.

[4] Bitwarden help center (encryption) *Bitwarden*, https://bitwarden.com/help/what-encryption-is-used/.

[5] Bitwarden help center (security) *Bitwarden*, https://bitwarden.com/help/vault-data/.

[6] PAOLO GASTI, KASPER B. RASMUSSEN , "On the security of password Manager Formats" ". *In: Computer Security, Berlin,* (2012), 9-12.

[7] Nist Password guideliness 2024 *Password managers*, https://www.auditboard.com/blog/nist-password-guidelines/.

[8] PAUL A. GRASSI, JAMES L. FENTON AND ELAINE M. NEWTON, WILLIAM E. BURR "Nist special Publication 800-63B " *Digital Identitiy Guidelines* (01 2023)

[9] Carlos Luevanos, John Elizarraras, Khai Hirschi and Jyh-haw Ye, "Analysis on the Security and Use of Password Managers" *18th International Conference on Parallel and Distributed Computing, Applications and Technologies* (12 2017).

[10] E2PM: Högskolan i Halmstad, Enclosed Portable Password Manager *Master Thesis* (06 2022).

[11] Yue Li, Haining Wang, And Kun Sun, BluePass: A Secure Hand-Free Password Manager *Lecture Notes of the Institute for Computer Sciences* (01 2018).

[12] RoboForm Review: features, ratings, pricing and more *website*, https://www.techradar.com/reviews/roboform.

[13] History of ChaCha20 *Chacha20*, https://www.tutorialspoint.com/cryptography/cryptography_chacha20_encryption_algorithm.htm.

[14] Simple and Secure Password Management *Password Safe*, https://pwsafe.org/index.shtml.

[15] Flutter *Wikipedia Website*, https://en.wikipedia.org/wiki/Flutter_(software).

[16] Phyton *Wikipedia Website*, https://tr.wikipedia.org/wiki/Python).

[17] MySql *Wikipedia Website*, https://en.wikipedia.org/wiki/MySQL.

[18] Fernando, W.P.K.1,* Dissanayake, D.A.N.P.1, Dushmantha, S.G.V.D.1, Liyanage, D.L.C.P.1, and Karunatilake, C1, "Challenges and Opportunities in Password Management: *A review of current solutions. Sri Lanka Journal of Social Sciences and Humanities"* 3(2) (08 2023), 9-20.

[19] Chacha20 Algorithm *website*, https://xilinx.github.io/Vitis_Libraries/security/2019.2/guide_L1/internals/chacha20.html.

[20] Advanced Encryption Standart*AES*, https://www.techtarget.com/searchsecurity/definition/Advanced-Encryption-Standard.

[21] Gelişmiş şifreleme standartı *AES*, https://www.geeksforgeeks.org/advanced-encryption-standard-aes/.

[22] Ahmad-Loay Sousi, Dalia Yehya And Mohamad Joudi, AES Encryption: Study Evaluation (11 2020).

[23] Zhengping Jay Luo,Ruowen Liu And Aarav Mehta, Understanding the RSA algorithm *DOI:10.48550/arXiv.2308.02785 LicenseCC BY-NC-ND 4.0* (08 2023).

[24] Erkan ÜNSAL,Humar KAHRAMANLI ÖRNEK AND Şakir TAŞDEMİR, A Review of Hashing Algorithms in Cryptocurrency *1st International Conference on Frontiers in Academic Research* (02 2023), Turkey.

[25] Thi Hong Tran, Pham Hoai Luan And Yasuhiko Nakashima , A High-Performance Multimem SHA-256 Accelerator for Society 5.0 *DOI:10.1109/ACCESS.2021.3063485 LicenseCC BY-NC-ND 4.0* IEEE Access PP(99):1-1 (March 2021), 45–52.

[26] Fahad Alodhyani, George Theodorakopoulos and Philipp Reinecke, Password Managers—It's All about Trust and Transparency *DOI:10.3390/fi12110189 LicenseCC BY 4.0* (10 2020), 12(11):189.

[27] Elizabeth A. Gallagher, Choosing the Right Password Manager, Serials Review, *DOI: 10.1080/00987913.2019.1611310* (05 2019), VOL. 45, NOS. 1–2, 84–87.