ÝSTANBUL ÜNÝVERSÝTESÝ MÜHENDÝSLÝK FAKÜLTESÝ YIL : 2001-2002 (23 – 31)
ELEKTRÝK-ELEKTRONÝK DERGÝSÝ CÝLT : 1
SAYI : 1

DES ALGORÝTMASINI KULLANAN GÜVENÝLÝR BÝR E-POSTA ÝLETÝM UYGULAMASI: TUÐRA

Adem KARAHOCA[1] M. Nusret SARISAKAL[2]

[1,2] Ýstanbul Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliði Bölümü
34850, Avcýlar, Ýstanbul
[1]e-posta: kadem@istanbul.edu.tr [2]e-posta: nsarisakal@istanbul.edu.tr

ABSTRACT

In this study focuses on, how we can obtain benefit from data encryption algorithms on security. Original DES algorithm is used for data encryption. SHA-1 algorithm gives support to us for the message correction. We developed an interface that is named as Tuðra for the e-mail encryption at the Windows platform.

ÖZET

Bu çalýþmada, güvenlik için veri þifreleme algoritmasýndan nasýl yararlanabileceðimiz konusu vurgulanmýþtýr. Özgün DES algoritmasý veri þifreleme için kullanýlýr. SHA-1 algoritmasý mesaj doðrulamasýnda bize destek saðlar. Windows ortamýnda e-mail þifrelemede Tuðra olarak isimlendirilen bir arayüz geliþtirilmiþtir.

Anahtar sözcükler: *Kriptografi, Güvenlik, Güvenilir Aðlar*

1. GÝRÝÞ

Günümüzde, Internet alt yapýsýna baðlý olarak, güvenlik ön plana çýkmýþtýr. Verilerin güvenilir bir biçimde aktarýmý ve eldesi için, kriptografi bilimi aracýlýðý ile çeþitli þifreleme, anahtarlama ve çözümleme algoritmalarý sunulmaktadýr. Bu çalýþmada, SSL(secure socekt layer-güvenilir soket katmaný) protokolünün de kullandýðý, DES þifreleme algoritmasý kullanýlmaktadýr. Ayrýca, mesaj doðrulama algoritmasý olarak SHA-1, verilerin güvenilir bir biçimde ºifrelenmesinden sonra, ºifre çözümleme iþlemleri esnasýnda veri kayýbýný engellemek için kullanýlmaktadýr.

Kriptografi literatüründe kullanýlan terimleri sýrasý ile açýklayalým: Gönderici, bir mesajý gönderen, alýcý ise mesajý alan kiþi olarak tanýmlanýr. Bir mesajý, doðrusal olmayan fonksiyonlar kullanarak deðiþtirmeye, ºifreleme denir. ª ifrelenmiº bir metni çözümlemeye, ters ºifreleme, yani matematiksel fonksiyonun tersini kullanarak, ºifrenin çözümlenmesi denir.

Gönderilecek metni M, ºifrelenmiº metni ª ile, ºifreleme fonksiyonunu da F ile gösterelim. Þifrelenmiþ metnin uzunluðu, gönderilecek metnin uzunluðundan fazla olabilir. Bu durumda, þifrelenmiþ metni sýkýþtýrma algoritmalarý yardýmý ile boy olarak küçültmeye çalýþýrýz. Bu tanýmlamalara göre þifreleme olayýnýn matematiksel modeli;

$$F(M) = ª \quad (1)$$

biçiminde verilir. ª ifrenin çözümlenmesi için kullanýlacak ters þifreleme fonksiyonu ise aþaðýdaki gibidir:

$$ª(M) = F \quad (2)$$

Genel olarak her ºifreleme ve ºifre çözümleme algoritmasýnýn birbirlerinin ters fonksiyonlarý olmalarý gerektiði ortadadýr. Fakat, bu fonksiyonlarýn lineer olmamasý da iþleri ayrýca zorlaþtýrmaktadýr.

Kriptografi uzun süre, askeri ve diplomatik haberleþmede kullanýlmaktaydý. Fakat, veri þifreleme standartý (DES) 1974 yýlýnda bir IBM çalýþma takýmý tarafýndan geliþtirilip ve 1977'de ulusal bir standart olarak kabul edilince bu bilim dalý aðlar üzerinde iletilen verilerin þifrelenmesine de taþýndý. Günümüzde Internet üzerinde güvenlikte yaºanan kaosu engellemek için kullanýlan SSL protokolünün temelinde de, bu algoritmanýn geliþtirilmiþ bir sürümü bulunmaktadýr. Dolayýsý ile günümüzde kullanýlan bu algoritmalarýn gerçekten güvenilir olup olmadýklarýný

test etmek için bu uygulamayý geliþtirdik. Þimdi sýrasý ile DES, SHA-1 algoritmalarýný inceleyip, uygulamamýzýn inceliklerini aktaralým.

2. DES ALGORÝTMASININ YAPISI

DES, ºifreleme iºlemleri için FIPS(Federal Information Processing Standarts Publications) tarafýndan geçerliliði onaylanmýþ algoritmalardan biridir.

DES, FIPS 140-1 tarafýndan gerekli gösterilen iki adet FIPS onaylý kriptografik algoritmayý tanýmlar. ANSI (American National Standards Institute – Amerikan Ulusal Standartlar Enstitüsü) X9.52 standardý ile birlikte kullanýldýðýnda, ikili kod durumundaki bilgilerin ºifrelenmesi ve çözümlenmesi için kullanýlacak matematiksel algoritmalar hakkýnda tam bir bilgi verir. Verinin ºifrelenmesi, veriyi "ºifre" denilen anlaþýlmasý imkansýz bir forma sokar. Verinin çözümlenmesi, veriyi "düz metin" denilen ilk formuna geri döndürür. Bu standartta tanýmlanan algoritmalar, "anahtar" adý verilen ikili bir sayý üzerine kurulmuþ hem þifreleme hem de çözümleme algoritmalarýný tanýmlar.

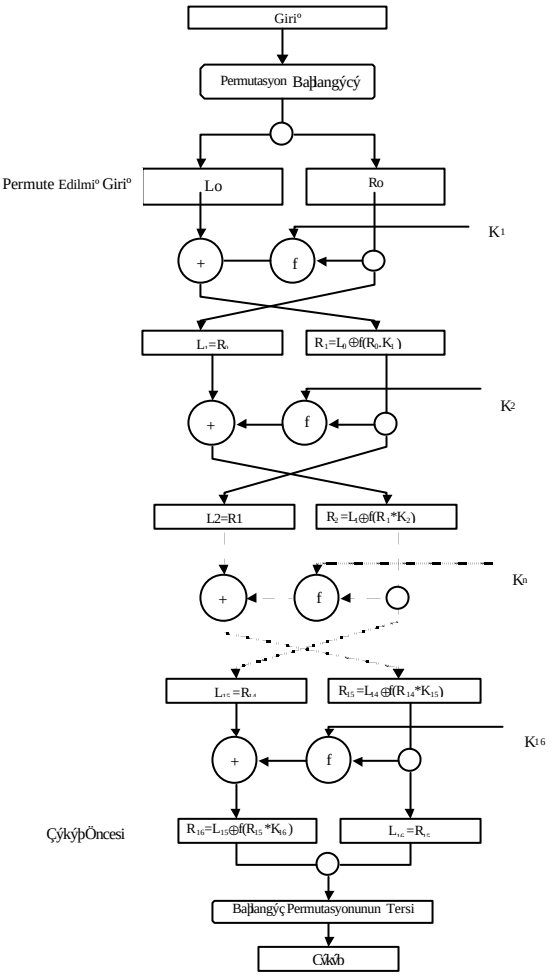
Bir DES anahtarý, 56 biti rasgele oluþturulan ve algoritma tarafýndan doðrudan kullanýlan 64 ikilik sayý ("0"lar ve "1"ler) içerir. Algoritma tarafýndan kullanýlmayan diðer 8 bit, hata denetimi için kullanýlabilir. Hata denetimi için kullanýlan 8 bite, anahtarýn her 8 bitlik kýsmýný tek yapmak için, "1" atanýr. (Bazen anahtarlar þifreli halde oluþturulur. Rasgele 64 bitlik bir sayý üretilir. Bu durumda anahtar çözümlenene kadar, þifrelenmiþ anahtarýn eþlik bitlerine deðer atanamaz.) Bir TDEA anahtarý, anahtar paketi de denilen, üç adet DES anahtarýndan oluþur. Þifrelenmiþ bilgisayar verisinin yetkili kullanýcýlarý veriyi çözümlemek istedikleri takdirde, veriyi þifrelemekte kullanýlan anahtara sahip olmak zorundadýr. Bu standartta açýklanan þifreleme algoritmalarý, genelde, bu standardý kullanan algoritmalardýr. Verinin kriptografik güvenliði, veriyi þifrelemek ve çözümlemek için kullanýlan anahtar için saðlanan güvenliðe baðlýdýr.

Veri þifreleme standardý, aþaðýda açýklanan veri þifreleme algoritmasý (DEA) ve üçlü veri ºifreleme algoritmasý'ndan (TDEA, ANSI X9.52'de belirtildiði gibi) oluþur. Bu algoritmalar, kodlanmýþ veriye kriptografik koruma saðlamak üzere, bir bilgisayar sistemi veya aðýnda kullanýlabilecek þekilde tasarlanmalýdýr. Geliþtirme metodu, uygulama ve çevreye baðýmlýdýr.

3. VERÝ ÞÝFRELEME ALGORÝTMASI-DEA

Bu algoritma, 64 bitlik bir anahtarýn kontrolü altýnda, 64 bitten oluþan veri bloklarýný (bloklar, bitlerin soldan saða numaralandýrýlmasýyla 1..64 oluºturulmuºtur) ºifrelemek ve çözümlemek için tasarlanmýþtýr. Çözümleme, þifrelemede kullanýlan ayný anahtarla, ancak çözümleme iþlemi þifreleme iºleminin tam tersi olacak ºekilde, anahtar bitlerini deðiþtirip adresleyecek bir çizelgeyle yapýlmalýdýr. Þifrelenecek bir blok, bir baþlangýç permütasyonundan (***IP***), anahtar baðýmlý, karmaþýk bir takým hesaplamalardan ve baþlangýç permütasyonunun tersi bir permütasyondan (IP^{-1}) oluþan iþlem adýmlarýndan geçirilir. Anahtar baðýmlý hesaplama, temel olarak, þifre fonksiyonu olarak adlandýrýlan bir ***f*** fonksiyonu ve anahtar listesi olarak adlandýrýlan bir ***KS*** fonksiyonu cinsinden tanýmlanabilir.

Ýlk önce, algoritmanýn þifreleme için kullanýlabileceðinin detaylarýyla birlikte, hesaplamanýn bir tanýmý verilecektir. Ardýndan, çözümleme için kullanýlan algoritma tanýmlanacaktýr. Son olarak, ºifre fonksiyonu ***f***'in tanýmý, seçme fonksiyonlarý (S_i) ve permütasyon fonksiyonlarý (***P***) olarak adlandýrýlan basit fonksiyonlar cinsinden verilecektir.



ªekil 1-Þifreleme Algoritmasý

Veri bloðu için ºu gösterim uygundur: Verilen bir bloðun solunu ***L*** ve saðýný da ***R*** ile simgelendirelim, ***L*** bitlerinden sonra gelen ***R*** bitlerinin oluþturduðu bloðu

LR ile simgeleyelim. Ardýþýllýk baðýmlý olduðundan, örneðin $B_1, B_2, \ldots, B_8$ dizisi B_1 bitlerinin ardýndan gelen B_2 bitlerinin ardýndan gelen, …, $B8$ bitlerinin oluþturduðu bloðu gösterir.

3.1. [a] ifreleme

Þifrelemenin akýþ diyagramý Þekil 1'de görüldüðü gibidir. Þifrelenecek girdi bloðunun 64 biti, ilk önce, baþlangýç permütasyonu (***IP***) olarak adlandýrýlan aþaðýdaki permütasyona tabi tutulur.

IP

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Permütasyona tabi tutulmuº girdi ilk bit olarak, orjinal girdinin 58. bitine, ikinci bit olarak 50. bitine, … ve sonuncu bit olarak girdinin 7. bitine sahiptir. Daha sonra, permüte edilmiþ girdi bloðu, aþaðýda tanýmlanan anahtar baðýmlý karmaþýk bir iþlemde girdi olarak kullanýlýr. Bu hesaplamanýn çýktýsý, ilk çýktý olarak adlandýrýlýr, baþlangýç permütasyonunun tam tersi olan, aþaðýdaki permütasyona tabi tutulur.

$\boldsymbol{IP^{-1}}$

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

Ýlk çýktýnýn 40. bitini ilk bit olarak, 8. bitini ikinci bit olarak, …,25. bitini de sonuncu bit olarak alýr. Ýlk çýktýyý üretmek üzere permüte edilmiþ girdiyi giriþ olarak alan hesaplama, bloklarýn son bir kez deðiþ tokuþundan önce olmak üzere aþaðýda tanýmlanan, biri 32 diðeri 48 bitten oluþan iki blok üzerinde iþlem yapan ve 32 bitlik bir blok üreten f ºifreleme fonksiyonu cinsinden ifade edeceðimiz, 16 iterasyonlu bir hesaplamadan oluºur.

Bir iterasyona giriþ olan 64 bitlik girdi bloðu, 32 bitlik bir ***R*** bloðu tarafýndan takip edilen 32 bitlik bir ***L*** bloðundan oluþsun. Baþta ifade ettiðimiz gösterimi kullanarak girdi bloðuna ***LR*** diyelim.

K, 64 bitlik anahtardan seçilmiº 48 bitten oluºan bir blok olsun. Bu durumda, giriºi ***LR*** alarak yapýlan bir iterasyonun çýkýþý $\boldsymbol{L'R'}$ þöyle tanýmlanabilir :

$$\boldsymbol{L' = R}$$
$$\boldsymbol{R' = L \oplus f(R, K)} \quad (3)$$

$\oplus$, mod 2'de bit düzeyinde toplamayý gösterir. Daha önce de belirtildiði üzere permüte edilmiþ girdi bloðu, hesaplamanýn ilk iterasyonuna giriþ olarak kullanýlýr. Eðer $\boldsymbol{L'R'}$, 16. iterasyonun çýkýþý ise $\boldsymbol{R'L'}$, ilk çýktý bloðudur. Her iterasyonda, ***ANAHTAR*** ile tanýmlanmýþ 64 bitlik anahtarýn bitleri arasýndan, farklý bir ***K*** anahtar bitleri bloðu seçilir.

KS, giriº olarak 1'den 16'ya deðiþebilen bir n tam sayýsý ve 64 bitlik ***ANAHTAR*** bloðunu alan ve çýkýþ olarak ***ANAHTAR***'dan seçilmiþ bir takým bitlerin permütasyonu olan 48 bitlik bir blok, $\boldsymbol{K_n}$'i, üreten bir fonksiyon olsun. Bu,

$$\boldsymbol{K_n = KS(n, ANAHTAR)} \quad (4)$$

Kn, ***ANAHTAR***'ýn 48 ayrý bit pozisyonundaki bitlerce belirlenir. ***KS***, anahtar listesi olarak isimlendirilir; çünkü, (3)'ün n'nci iterasyonunda kullanýlan ***K*** bloðu, (4)'de belirlenen ***Kn*** bloðudur.

Permüte edilmiº blok ***LR*** olsun. Son olarak, $\boldsymbol{L_0}$ ve $\boldsymbol{R_0}$, ***L*** ve ***R*** olsun ve n, 1'den 16'ya, ***L*** ve ***R***'nin $\boldsymbol{L_{n-1}}$ ve $\boldsymbol{R_{n-1}}$ olduðu ve ***K***, $\boldsymbol{K_n}$ olduðu durumda, $\boldsymbol{L_n}$ ve $\boldsymbol{R_n}$ (3)'deki $\boldsymbol{L'}$ ve $\boldsymbol{R'}$ olsun.

$$\boldsymbol{L_n = R_{n-1}}$$
$$\boldsymbol{R_n = L_{n-1} \oplus f(R_{n-1}, K_n)} \quad (5)$$

Bu durumda ilk çýktý bloðu, $\boldsymbol{R_{16}L_{16}}$'dýr. Algoritmanýn anahtar listesi ***KS***, algoritma için gerekli olan 16 adet $\boldsymbol{K_n}$'i üretir.

3.2. Çözümleme

Ýlk çýktý bloðuna uygulanan permütasyon $\boldsymbol{IP^{-1}}$, giriºe uygulanan baþlangýç permütasyonu ***IP***'nin tersidir.

$$\boldsymbol{R = L'}$$
$$\boldsymbol{L = R' \oplus f(L', K)} \quad (6)$$

Bu nedenle, ***çözümleme*** için, ºifrelenmiº bir mesaj bloðuna ***tamamýyla ayný algoritmayý uygulamak*** gereklidir. Çözümleme sýrasýnda yapýlan hesaplamalarýn bir iterasyonunda, bloðun þifrelenmesi sýrasýnda kullanýlan ***ayný K anahtar bitleri bloðunun kullanýldýðýna*** dikkat edilmelidir. Önceki bölümde kullandýðýmýz gösterimi kullanarak þu eþitliklerle açýklanabilir :

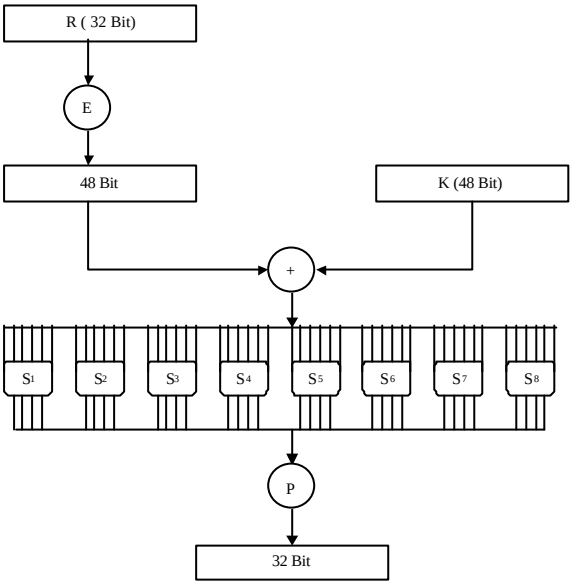
$$\boldsymbol{R_{n-1} = L_n}$$
$$\boldsymbol{L_{n-1} = R_n \oplus f(L_n, K_n)} \quad (7)$$

Çözümleme hesabý için $\boldsymbol{R_{16}L_{16}}$, permüte edilmiº girdi bloðu ve $\boldsymbol{L_0R_0}$ ilk çýktý bloðudur. Bu da, $\boldsymbol{R_{16}L_{16}}$'yý permüte edilmiº giriº olarak alan çözümleme

hesaplamasý için, K_{16} ilk iterasyonda, K_{15} ikincide, …, K_1 16ncý iterasyonda kullanýlacaktýr.

3.3. ª ifreleme Fonksiyonu *f*

f (R, K)'nýn hesaplanmasýna dair bir taslak ***ª ekil 2***'de verilmiºtir.



ª ekil 2-*f*(R, K)'*nýn Hesaplanmasý*

E, 32 bitlik bir bloðu giriþ olarak alýp, çýkýþ olarak 48 bitlik bir blok üreten bir fonksiyonu göstersin. ***E***'nin 48 bitlik çýktýsý, giriþindeki bitlerin aþaðýdaki tabloya göre (8 satýr 6 sütun) seçilmesiyle elde edilir :

E BIT-SEÇÝM TABLOSU

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

E(R)'nin ilk üç biti, sýrasýyla, giriþ olarak aldýðý ***R*** bloðunun 32, 1 ve 2. pozisyonlardaki bitleri ve son iki biti de ***R***'nin 32 ve 1. pozisyonlardaki bitleridir.

Her biri diðerlerinden farklý olan seçim fonksiyonlarý $S_1, S_2, \ldots, S_8$, giriþ olarak 6 bitlik bir blok alýr ve çýkýþ olarak 4 bitlik bir bloðu, örneðin S_1 için aþaðýdaki gibi, bir tablo kullanarak üretir :

S1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

Eðer S_1, bu tabloda tanýmlanan fonksiyonsa ve ***B,*** 6 bitlik bir bloksa, $S_1(B)$ ºu ºekilde belirlenir : ***B***'nin ilk ve son bitleri, 2 tabanýnda, 0'dan 2'ye deðiþen bir sayýyý temsil eder. Bu sayý *i* olsun. ***B***'nin ortadaki 4 biti, 2 tabanýnda, 0'dan 15'e deðiþen bir sayýyý temsil eder. Bu sayý da *j* olsun. Tablonun *i*. satýr, *j*. sütunundaki deðerini ele alalým. Bu, 0'dan 15'e deðiþen ve 4 bitlik bir blok tarafýndan benzersiz olarak temsil edilen bir sayýdýr. Bu blok, ***B*** giriºi için, S_1'in $S_1(B)$ çýkýþýdýr. Örneðin, 011011 giriþi için satýr 01, yani 1, ve kolon 1101, yani 13 olarak belirlenir. Satýr 1, sütun 13'deki deðer 5'tir, o halde, çýkýþ 0101'dir.

Permütasyon fonksiyonu ***P***, giriþ bloðunun bitlerini permüte ederek 32 bitlik bir giriþten 32 bitlik bir çýkýþ üretir. Böyle bir fonksiyon, aþaðýdaki tablo ile tanýmlanýr :

P

16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

P fonksiyonu için bu tabloyla tanýmlanan ***P(L)*** çýkýþý, ***L*** giriºinden ***L***'nin 16. bitini ***P(L)***'nin ilk biti, 7. bitini ***P(L)***'nin 2. biti gibi, böyle devam ederek, ***L***'nin 25. bitini ***P(L)***'nin 32.nci biti alarak saðlar.

ª imdi, $S_1, S_2, \ldots, S_8$ 8 farklý seçim fonksiyonu, ***P*** permütasyon fonksiyonu ve ***E*** yukarýda tanýmlanan fonksiyon olsun.

f(R, K)'yý tanýmlamak için ilk önce her biri 6 bitlik birer blok olan $B_1, B_2, \ldots, B_8$'i tanýmlayalým :

$$B_1, B_2, \ldots, B_8 = K \text{ Å } E(R) \qquad (8)$$

O halde ***f(R, K)*** þöyle tanýmlanýr :

$$P(S_1(B_1)S_2(B_2)\ldots S_8(B_8)) \qquad (9)$$

Bu nedenle K Å $E(R)$, ilk önce (8)'da gösterildiði gibi 8 bloða bölünür. Ardýndan her bir B_i, S_i'ye ait bir giriº olarak alýnýr ve her biri 4 bitlik 8 blok, $S_1(B_1), S_2(B_2), \ldots, S_8(B_8)$, 32 bitlik tek bir blok oluºturacak ºekilde birleºtirilir, bu da giriº ***P***'yi oluþturur. O halde çýkýþ (9), ***R*** ve ***K*** giriºleri için ***f*** fonksiyonunun çýkýþý olur.

4. ÜÇLÜ VERÝ ÞÝFRELEME ALGORÝTMASI -TDEA

$E_k(I)$ ve $D_k(I)$, I için, DES anahtarý K'yý kullanarak DES ºifrelemesini ve çözümlemesini temsil etsin. Her TDEA ºifreleme/çözümleme iºlemi (ANSI X9.52'de belirtildiði gibi), DES þifreleme/çözümleme iºlemlerinin bir bileºimidir. ª ifreleme/çözümleme için aþaðýdaki iþlemler kullanýlýr :

1. TDEA ºifreleme iºlemi: 64 bitlik bir I bloðunun, 64 bitlik bir O bloðuna, aþaðýda tanýmlanan þekliyle çevrimi :
   $O = E_{K3}(D_{K2}(E_{K1}(I)))$

2. TDEA çözümleme iºlemi: 64 bitlik bir O bloðunun, 64 bitlik bir I bloðuna, aþaðýda tanýmlanan þekliyle çevrimi :
   $O = D_{K1}(E_{K2}(D_{K3}(I)))$

Standart, (K_1, K_2, K_3) paketi için aþaðýdaki anahtarlama özelliklerini tanýmlar :

1. Anahtarlama özelliði 1: K_1, K_2 ve K_3 baðýmsýz anahtarlardýr.

2. Anahtarlama özelliði 2: K_1, K_2 baðýmsýz anahtarlar ve $K_1 = K_3$.

3. Anahtarlama özelliði 3 : $K_1 = K_2 = K_3$.

TDEA türü bir iºlem, TDEA iºlemi için uyumlu anahtarlama özelliði ile birlikte aþaðýdaki özellikleri saðlamasý koþuluyla, tek katlý benzeri DES iþlemi ile geriye doðru uyumludur :

1. Tek bir DES türü iºlem kullanarak ºifrelenmiº bir düz metne, karþýlýk gelen bir TDEA türü iþlemle doðru olarak çözülebilmesi.

2. TDEA türü bir iºlem kullanarak ºifrelenmiº bir düz metne, karþýlýk gelen tek bir DES türü iþlemle doðru çözülebilmesi.

Anahtarlama özelliði 3. seçeneði ($K_1 = K_2 = K_3$) kullanýrken, TECB, TCBC, TCFB ve TOFB yollarý, sýrasýyla tekil DES tipi iþlemler ECB, CBC, CFB ve OFB ile geriye doðru uyumludur.

Üçlü DES Blok Diyagramý (ECB Yolu Ýle)

TDEA Þifreleme Ýþlemi :
Giriº → DES EK1→ DES DK2 → DES EK3 → Çýkýþ

TDEA Çözümleme Ýþlemi :
Giriº → DES DK3→ DES EK2 → DES DK1 → Çýkýþ

5. SHA-1 MESAJ DOÐRULAMA ALGORÝTMASI

Güvenli hash algoritmasý (SHA-1 - Secure Hash Algorithm), sayýsal imza standardýnda (DSS - Digital Signature Standart) açýkça belirtildiði gibi, federal uygulamalarda güvenli bir Hash algoritmasý ihtiyaç duyulduðunda, sayýsal imza algoritmasý (DSA – Digital Signature Algorithm) ile birlikte kullanýlýr. 2^{64} bitten daha kýsa uzunlukta bir mesaj için SHA-1, mesajýn 160 bit uzunluðunda, "mesaj özeti" isminde bir özet tanýmýný içerir. Mesaj özeti, mesaj için bir imza üretimi sýrasýnda kullanýlýr. SHA-1 ayný zamanda imzanýn doðrulanmasý iþlemi sýrasýnda, alýnan mesaj sürümü için bir mesaj özeti hesaplamak için de kullanýlýr. Ýletim sýrasýnda mesajda oluþacak herhangi bir deðiþiklik, yüksek ihtimalle, farklý bir mesaj özeti ile sonuçlanacak ve imzanýn doðrulanmasý baþarýsýz olacaktýr.

SHA-1 takip eden özelliklere sahip olmasý için tasarlanmýþtýr. Verilen bir mesaj özetine karþýlýk gelen bir mesajýn bulunmasý veya ayný mesaj özetini üretecek iki farklý mesajýn bulunabilmesi cebirsel olarak olasý deðildir.

5.1. Bit Dizileri ve Tam Sayýlar

Aþaðýdaki terimler dizgesi, kullanýlacak olan bit dizileri ve tam sayýlarla ilgilidir :

a. *Onaltýlýk bir rakam*, {0, 1, … , 9, A, … , F} kümesinin bir elemanýdýr. Bir onaltýlýk rakam, 4 bitlik bir diziyi temsil eder. **Örnek:** 7 = 0111, A = 1010.

b. Bir *kelime*, 8 onaltýlýk rakam dizisi þeklinde temsil edebilecek bir 32 bitlik diziye eºittir. Bir kelimeyi 8 onaltýlýk rakama çevirmek için, her 4 bitlik dizi, yukarýda (a)'da belirtildiði gibi, onaltýlýk eþdeðerine çevrilir. **Örnek :**

   1010 0001 0000 0011 1111 1110 0010 0011 = A103FE23

c. *0 ile $2^{32} - 1$ arasýndaki (sýnýrlar dahil) bir tam sayý*, bir kelime olarak gösterilebilir. Tam sayýnýn en düþük anlamlý dört biti, kelime gösterimindeki en saðdaki onaltýlýk rakam ile gösterilir. **Örnek :** tam sayý $291 = 2^8 + 2^5 + 2^1 + 2^0 = 256 + 32 + 2 + 1$, onaltýlýk kelime 00000123 ile gösterilir.

   Eðer z, $0 \quad z < 2^{64}$, aralýðýnda bir tam sayýysa,
   $z = 2^{32} x + y$,
   $0 \quad x < 2^{32}$ ve $0 \quad y < 2^{32}$.
   'x' ve 'y', kelime X ve Y olarak gösterilebildiðinden 'z', kelime çifti (X, Y) olarak temsil edilebilir.

d. *blok* = 512 bitlik dizi. Bir blok (örneðin B), 16 kelimelik bir seri ile gösterilebilir.

5.2 Kelimeler Üzerinde Ýþlemler

Aþaðýdaki mantýksal iþlemler, kelimelere uygulanabilir :

a. Bit düzeyinde mantýksal kelime iþlemleri

$X \wedge Y$ = X ve Y'nin bit düzeyinde mantýksal "ve"si
$X \vee Y$ = X ve Y'nin bit düzeyinde mantýksal "veya"sý
X XOR Y= X ve Y'nin bit düzeyinde mantýksal "XOR"u
~X = X ve Y'nin bit düzeyinde mantýksal tersi

Örnek :
01101100101110011101001001111011
XOR
01100101110000010110100110110111

=00001001011110001011101111001100

b. *Ýþlem X + Y* þu þekilde tanýmlanýr : X ve Y kelimeleri, x ve y tam sayýlarýný $0 \le x < 2^{32}$ ve $0 \le y < 2^{32}$ aralýðýnda temsil eder. Pozitif tam sayýlar n ve m için n mod m, n'in m'e bölünmesi sonucu kalan olsun. Hesaplarsak,

$$Z = (x + y) \bmod 2^{32}$$

O halde, $0 \le z < 2^{32}$ aralýðýnda z'yi bir kelimeye dönüºtürürsek,

$$Z = X + Y \text{ olur.}$$

c. *Dairesel sola kaydýrma iþlemi* $S^n(X)$, X bir kelime ve n, $0 \le n < 32$ aralýðýnda bir tam sayý, þu þekilde tanýmlanýr :

$$S^n(X) = (X << n) \vee (X >> 32-n).$$

Yukarýda X << n þu þekilde elde edilir : X'in en solundaki n biti atýlýr ve sonucu saðdan itibaren sýfýrlarla doldurulur (sonuç hala 32 bittir). X >> n, X'in en saðýndaki n bitin atýlýp sonucun soldan itibaren sýfýrlarla doldurulmasý ile saðlanýr. Böylece $S^n(X)$, X'in sola doðru n pozisyon dairesel kaymasýna denktir.

5.3. Mesaj Doldurma

SHA-1, giriþ olarak saðlanan bir mesaj veya veri dosyasý için bir mesaj özeti hesaplamak için kullanýlýr. Mesaj veya veri dosyasý bir bit dizisi olarak ele alýnmalýdýr. Mesajýn uzunluðu, mesajýn içindeki bitlerin sayýsýdýr (boþ mesaj, 0 uzunluðuna sahiptir). Eðer mesajýn içindeki bitlerin sayýsý 8'in katýysa, sadelik için mesajý onaltýlýk olarak ifade edebiliriz. Mesaj doldurmanýn amacý, doldurulmuþ bir mesajýn toplam uzunluðunu, 512'nin katý yapmaktýr. SHA-1 mesaj özetini hesaplarken, 512 bitlik bloklarý sýrasal olarak iþler. Aþaðýda, bu doldurmanýn nasýl gerçekleþtirildiði açýklanmýþtýr. Bir özet olarak, 512 × n uzunlukta doldurulmuþ bir mesajýn üretilmesi için, mesajýn sonuna sýrasýyla bir "1", m adet "0" ve 64 bitlik bir tam sayý eklenir. 64 bitlik tam sayý *l*, orjinal mesajýn uzunluðudur. Bundan sonra mesaj, SHA-1 tarafýndan 512 bitlik n blok olarak iþlenecektir.

Bir mesajýn $l < 2^{64}$ uzunluðunda olduðunu varsayalým. SHA-1'e giriþ olmadan önce, mesaj saða aþaðýdaki þekilde kaydýrýlýr :

a. "1" eklenir. Orjinal mesaj "01010000" ise, "010100001" olur.

b. "0"lar eklenir. "0"larýn sayýsý, mesajýn orjinal uzunluðuna baðlýdýr. Son 512 bitlik bloðun son 64 biti, orjinal mesajýn uzunluðu *l* için ayrýlýr.

Örnek : Aþaðýdaki bit dizisi orjinal mesajýmýz olsun :
01100001 01100010 01100011 01100100 01100101.

Adým (a)'dan sonra mesaj þu hale gelir :

01100001 01100010 01100011 01100100 01100101 1.

l = 40 iken, yukarýdaki bitlerin sayýsý 41 olduðundan, 407 adet "0" eklenir ve toplamda 448 bit olur. Bu, onaltýlýk olarak, aþaðýdaki diziyi verir :

61626364	65800000
00000000	00000000
00000000	00000000
00000000	00000000
00000000	00000000
00000000	00000000
00000000	00000000.

c. Orjinal mesajdaki bitlerin sayýsý *l*'nin 2 kelimelik gösterimi ile saðlanýr. Eðer $l < 2^{32}$ ise, ilk kelime tamamýyla sýfýrdan oluþur. Bu iki kelime doldurulmuº mesaja eklenir.

Örnek: Orjinal mesajýn (b)'deki gibi olduðunu düºünelim. O zaman *l* = 40 olur (*l*, hiçbir doldurma iþlemi yapýlmadan önce hesaplanýr). 40'ýn iki kelime gösterimi, onaltýlýk düzende, 00000000 00000028 dir. Artýk doldurulmuþ mesajýmýzýn son hali, onaltýlýk düzende, aþaðýdaki þekildedir :

61626364	65800000
00000000	00000000
00000000	00000000
00000000	00000000
00000000	00000000
00000000	00000000
00000000	00000000
00000000	00000028.

Kaydýrýlmýþ mesaj, her n > 0 için, 16n kelime içerir. Kaydýrýlmýþ mesaj, onaltýþar kelime ve mesajýn ilk

karakterlerini (veya bitlerini) içeren her M_i için, M_1, M_2, …, M_n'den oluþan n bloklu bir dizi olarak ele alýnýr.

5.4. Kullanýlan Fonksiyonlar

SHA-1'de bir dizi mantýksal fonksiyon, f_0, f_1, …, f_{79}, kullanýlýr. Her f_t, 0 t 79, 32 bitlik kelimeler üzerinde iþlem yapar ve çýkýþ olarak 32 bitlik bir kelime üretir. f, B, C ve D kelimeleri için þu þekilde tanýmlanýr:

$f_t(B,C,D) = (B \wedge C) \vee (\sim B \wedge D)$
(00 t 19)

$f_t(B,C,D) = B$ XOR C XOR D
(20 t 39)

$f_t(B,C,D) = (B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$
(40 t 59)

$f_t(B,C,D) = B$ XOR C XOR D
(60 t 79)

5.5. Kullanýlan Sabitler

SHA-1'de bir dizi sabit, K_0, K_1, …, K_{79}, kullanýlýr. Onaltýlýk olarak aþaðýdaki gibi listelenebilirler:

K_t = 5A827999
(00 t 19)
K_t = 6ED9EBA1
(20 t 39)
K_t = 8F1BBCDC
(40 t 59)
K_t = CA62C1D6
(60 t 79)

5.6. Mesaj Özetini Hesaplamak

Mesaj özeti, doldurulmuþ mesajýn son hali kullanýlarak hesaplanýr. Hesaplama, her biri 5 tane 32 bitlik kelimeden ve 80 adet 32 bitlik kelimenin oluþturduðu bir diziden oluþan iki tampon kullanýr. Ýlk 5 kelimelik tampondaki kelimeler, A, B, C, D, E olarak adlandýrýlýr. Ýkinci 5 kelimelik tampondaki kelimeler, H_0, H_1, H_2, H_3, H_4 olarak adlandýrýlýr. 80 kelimelik dizideki kelimeler, W_0, W_1, …, W_{79} olarak adlandýrýlýr. TEMP isimli tek kelimelik bir tampon da, ayrýca kullanýlýr.

Mesaj özetini oluþturmak için, 16 kelimelik bloklar, M_1, M_2, …, M_n, sýrasýyla iþlenir. Her M_i'nin iþlenmesi, 80 adýmdan oluþur.

Herhangi bir blok iþlenmeden önce H_j'ler, onaltýlýk düzende, aþaðýdaki þekilde atanýr :

H_0 = 67452301
H_1 = EFCDAB89
H_2 = 98BADCFE
H_3 = 10325476
H_4 = C3D2E1F0

[a] imdi M_1, M_2, …, M_n, aþaðýdaki þekilde, iþlenebilir :

a. M_i'yi, W_0 en soldaki kelime olacak þekilde 16 kelimeye bölün : W_0, …, W_{15}.

b. t = 16'dan 79'a kadar, $W_t = S^1(W_{t-3}$ XOR W_{t-8} XOR W_{t-14} XOR $W_{t-16})$ olsun.

c. $A = H_0$, $B = H_1$, $C = H_2$, $D = H_3$, $E = H_4$ olsun.

d. t = 0'dan 79'a kadar, aþaðýdaki iþlemler gerçekleþtirilir:

$TEMP = S^5(A) + f_t(B,C,D) + E + W_t + K_t$;

$E = D$; $D = C$; $C = S^{30}(B)$; $B = A$; A= TEMP;

e. $H_0 = H_0 + A$, $H_1 = H_1 + B$, $H_2 = H_2 + C$, $H_3 = H_3 + D$, $H_4 = H_4 + E$ olsun.

M_n iþlendikten sonra, 160 bitlik mesaj özeti, aþaðýdaki 5 kelime ile gösterilebilir :

H_0, H_1, H_2, H_3, H_4

6. GÜVENÝLÝR E-POSTA ÝLETÝM UYGULAMASI-TUÐRA

Bu yazýlýmda, daðýtýk sistemler için tasarlanmýþ þifreleme algoritmalarýndan DES kullanýlarak mesaj bütünlüðü ve göndericinin doðrulanmasý iþlemleri gerçekleþtirilmektedir.

Tuðra yazýlýmý mesaj imzalama iþleminde öncelikle SHA-1algoritmasýyla mesaj özeti oluþturulmaktadýr. Daha sonraki adýmda, private.txt dosyasý DES algoritmasýný kullanarak, public.txt dosyasýný oluþturmak için kullanýlmaktadýr. Bir sonraki adýmda public.txt dosyasý, oluþturulan mesaj özeti ile birleþtirilerek DES algoritmasý bir daha þifreleme iþlemini gerçekleþtirmektedir. Mesaj, gönderilen bilgisayarda public.txt dosyasý, göndericinin doðruluðunu kanýtlamak ve kontrol etmek için kullanýlýr. Private.txt dosyasý ise mesajý alacak kiþinin bilgisayarýnda bulunmalýdýr.

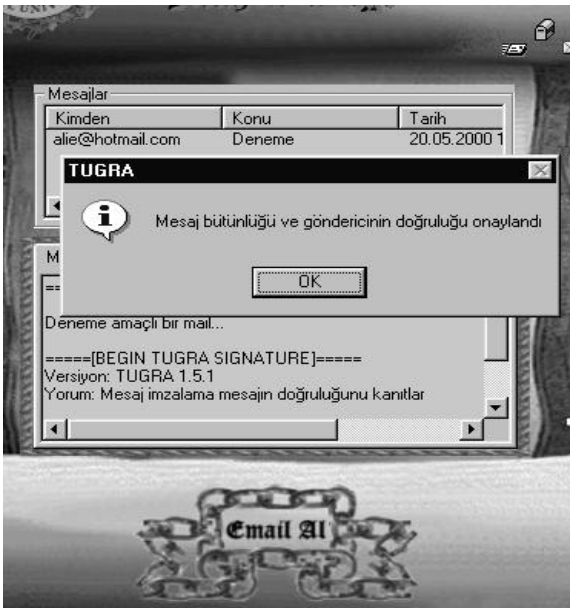
Yazýlým ilk çalýþtýrýldýðýnda e-mail gönderebilmek için SMTP sunucunuzu belirlemeniz gerekmektedir. Örneðin, **smtp.istanbul.edu.tr** deðerini bu forma girmeniz gerekmektedir. Ayrýca, bu forma gönderen kiþinin posta adresini ve postayý alacak kiþinin posta adresini belirlememiz gerekmektedir.

Mesaj oluþturma adýmýnda, kullanýcý, formun sað üst köþesindeki Tuðra simgesini týklayarak o anda, metin kutusu içerisinde bulunan mesajýn bir özetinin hazýrlanmasý saðlanabilir. Mesaj özeti oluþturulurken, mesajýn gövdesinin nereden baþlayýp nerede bittiðini,

imza gövdesinin nerede baþlayýp nerde bittiðini belirlemek için özel baþlýklar kullanýlmaktadýr. Ýmza kontrolünün gerçekleþtirilebilmesi için bu baþlýklarýn deðiþtirilmemesi gerekmektedir. Mesaj özeti oluþturma iþlemi keyfidir. Bu iþlemi gerçekleþtirmeden de elektronik mesajýnýzý gönderebilirsiniz. (ª ekil 3)

ª ekil 3 - Bir mesajýn Tuðra yazýlýmý aracýlýðý ile imzalanmasý

Mesaj alma iþlemi esnasýnda da, imza oluþturmak için Tuðra simgesine týklamak gerekmektedir. Ýmza kontrolü doðrulanýrsa, Þekil 4 ile mesaj bütünlüðü ve göndericinin doðruluðu saðlanamazsa Þekil 5 ile karþýlaþýlýr.



ª ekil 4 - Gerekli anahtarlara sahip bir istemcinin mesajý almasý

ª ekil 5 - Gerekli anahtarlara sahip olmayan bir istemcinin mesajý alamamasý

7. SONUÇ VE ÝLERÝYE YÖNELÝK ÇALIª MALAR

Bu yazýlým ile literatürde mevcut olan DES þifreleme algoritmasý ile SHA-1 mesaj doðrulama algoritmasý kullanýlarak þifreleme ve çözümleme uygulamasý gerçekleþtirilmiþtir. Yazýlýmýn performansý kullanýlan donanýma baðlý olmakla beraber, hýzlý sayýlabilir. Bu algoritmalarýn çözümleyicileri de Internet üzerinde mevcut olduðundan, bu konularda iyi olan birileri bu þifreleme iþlemini çözebilir. Dolayýsý ile, bu gibi problemlerle karþýlaþmamak için, daha yavaþ fakat daha güvenilir olan 128 bitlik algoritmalar tercih edilebilir.

Bu çalýþmamýzýn sonuçlarýný, diðer þifreleme algoritmalarýný test edip karþýlaþtýrmak ve de bu algoritmalarýn hibrid kullanýmlarý üzerinde durmak, ekibimizin bu projedeki temel amacýný oluþturmaktadýr.

REFERENCES

[1] Schneier B., Kelsey J., Whiting D., Wagner D., Hall C., Ferguson N., The Twofish Encryption Algorithm, John Wiley & Sons, 2000

[2] Schneier B., Open Source and Security, Crypto-Gram , 15 sep 1999

[3] Schneier B., E-Mail Viruses, Worms, and Trojan Horses, Crypto-Gram, 15 june 1999

[4] S. Moriai, T. Shimoyama, T. Kaneko, Higher Order Differential Attack of a CAST Chiper, Proceedings of The Fifth International Workshop on Fast Software Encryption, LNCS 1372, Springer, pp. 17-31, Paris, France, March 1998

[5] C. Adams, The Cast-128 Encyrption Algorithm, RFC2144, May 1997

[6] Schneier B., Banisar D., The Electronic Privacy Papers: Documents on the Battle for Privacy in the Age of Surveillance, John Wiley & Sons, 1997

[7] T. Ritter, The Fenced DES Chiper: Stronger Than DES But Made From DES, http://www.io.com/~ritter/fenced.htm, November 10, 1996

[8] S. Lucks, Faster Luby Rackoff Ciphers, Proceedings of The Third International Workshop on Fast Software Encryption, Cambridge UK, Springer, LNCS 1039, pp.189-203, Februrary 1996

[9] Stallings W., Practical Cryptography For Data Internetworks, IEEE Comp. Soc. Press, 1996

[10] Schneier B., Applied Cryptography, Second Edition, John Wiley & Sons, 1996

[11] Stinson D. R., Cryptography Theory and Practice, CRC Press, 1995, Florida

[12] C. Adams, Simple and Effective Key Scheduling for Symmetric Ciphers, SAC'94, pp. 129-133, Kingston, Canada, May 1994,

[13] Coopersmith D., The Data Encryption Standart(DES) and its strength against attacks, IBM JOURNAL RESEARCH AND DEVELOPMENT, Vol.38, No. 3, pp. 243-250, May 1994.

[14] M. Luby, C. Rackoff, How to Construct Pseudorandom Permutations From Pseudorandom Functions, SIAM Journal of Computing, vol. 17, no. 2, pp. 373-386, April 1988

[15] Froomkin A. M., The Metaphor is The Key: Cryptography, The Clýpper Chip, And The Constitution, http://www.law.miami.edu /~froomkin/articles/clipper.htm