

Lecture Notes: Kriptografija

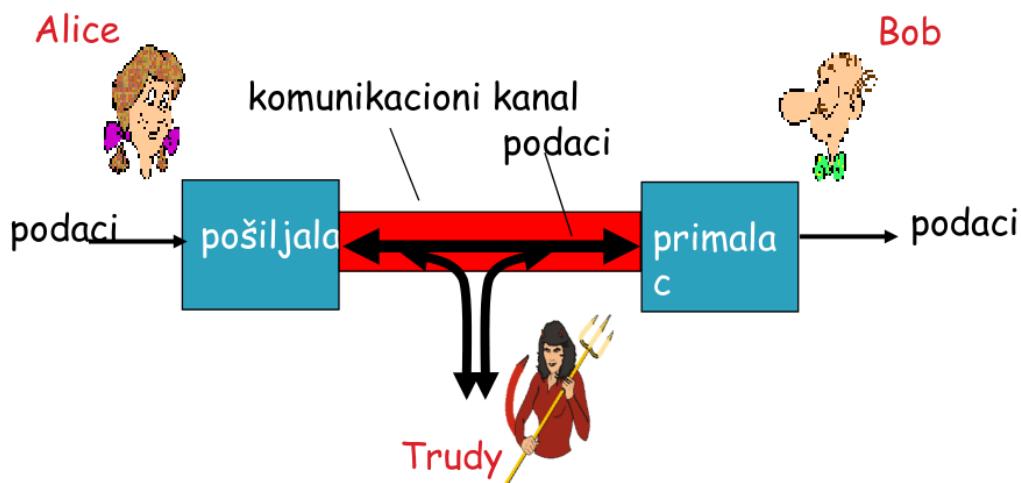
February 12, 2026

Contents

1.	01 - Uvod	1
2.	02 - Iсторијски ѕифрарски алгоритми - I	8
3.	02 - Iсторијски ѕифрарски алгоритми - II	13
4.	03 - Steganografija	18
5.	04 - Moderni simetrični ѕифрарски системи - I	20
6.	04 - Moderni simetrični ѕифрарски системи - II	32
7.	05 - Asimetričни ѕифрарски алгоритми	47
8.	05 - Integritet порука	54
9.	06 - Razmjena ključeva	68
10.	07 - Digitalni потпис	73
11.	08 - Digitalni сertifikati i PKI	75
12.	09 - CA	79
13.	10 - TSA	85

Lecture 1.

01 - Uvod



Alice i Bob žele da komuniciraju “na siguran način”

Trudy (intruder) može presresti, obrisati, mijenjati i generisati nove poruke - tako remeti komunikaciju Alice i Boba

Primjeri:

- DNS server i klijenti
- SMTP server i klijenti
- POP3 server i klijenti
- IMAP server i klijenti
- Web server i klijenti
- Web aplikacija i klijenti
- router-i koji razmjenjuju routing tabele

Napadi:

- Pasivni
- Aktivni

Napad na tajnost

presretanje (interception), prisluskivanje - Trudy presreće komunikaciju između Alice i Boba - *pasivni napad*

napad na dostupnost

ometanje (interruption) - Trudy ometa komunikaciju između Alice i Boba - DoS - aktivni napad

napad na integritet

modifikacija (modification) - Trudy modifikuje podatke koje razmjenjuju Alice i Bob - aktivni napad

napad na autentikaciju i autorizaciju

fabrikacija (fabrication) - Trudy generiše saobraćaj i šalje ga ka Alice i/ili Bobu - aktivni napad

Primjeri napada

Presretanje:

- analiza saobraćaja
- prisluškivanje - otkrivanje sadržaja poruka

Ometanje, modifikacija, fabrikacija:

- lažno predstavljanje
- replay
- modifikacija
- DoS napadi
- DDoS napadi
- MITM napadi

Primjeri zaštite

Ometanje:

- firewall-i, IDS, IPS

Presretanje:

- enkripcija

Modifikacija:

- enkripcija, backup-i, checksum-i, digitalni potpis, authentication kodovi

Fabrikacija:

- autentikacija i autorizacija, firewall-i, digitalni potpis

Fundamentalni sigurnosni zahtjevi

Tajnost (privacy, confidentiality)

- obezbjeđivanje sigurnog komunikacionog kanala između učesnika u elektronskoj transakciji,
- sprečavanje otkrivanja informacija i/ili podataka neautorizovanim stranama
- osnovna komponenta neophodna za očuvanje privatnosti korisnika

- obezbjeđuje se: korištenjem enkripcije

Integritet (data integrity)

- verifikacija da nije došlo do narušavanja integriteta podataka za vrijeme njihovog prenosa kroz komunikacioni kanal
- zaštitu od neautorizovanog mijenjanja podataka u toku njihovog prenosa
- *obezbjeđuje se:* korištenjem digitalnog potpisa

Autentikacija (authentication)

- obezbjeđivanje da su učesnici u transakciji oni za koje se predstavljaju
- sprječava lažno predstavljanje učesnika u komunikaciji
- *obezbjeđuje se:* korištenjem digitalnih sertifikata ili tokena

Neporecivost (non-repudiation)

- obezbjeđivanje da učesnici u transakciji ne mogu poreći svoje učešće
- *obezbjeđuje se:* korištenjem digitalnog potpisa

CIA trijada:

- Confidentiality
- Integrity
- Availability



Drugi sigurnosni zahtjevi

Dostupnost (availability)

- resurs je dostupan u trenutku kada je zahtjevan od strane autorizovanog korisnika

Kontrola pristupa (access control)

- sprječava neautorizovan pristup resursima

Pouzdanost (reliability)

- otpornost na otkaze

Audit

- audit informacije moraju biti sačuvane kako bi akcije koje su ugrozile sigurnost naknadno mogle biti istražene

Principi sigurnosti

Neki od važnijih principa sigurnosti:

- 100% sigurno okruženje ne postoji - ne postoji absolutna sigurnost
- očekivati neočekivano
- što bolje poznajete napadača, to se bolje od njega možete odbraniti
- to vas napadač bolje poznaje, to može bolje izvršiti napad
- pogrešno je prepostavljati da svi napadi dolaze izvana
- pogrešno je prepostavljati da vaši korisnici tačno znaju šta rade - edukacija korisnika je veoma važna
- sigurnost sistema određuje najslabiji dio sistema
- sigurnost je proces, a ne krajnji proizvod
- fizička sigurnost je, takođe, važna
- least privilege
- privilege separation

Kriptologija

Kriptografija

- tehnika koja proučava principe i metode šifrovanja podataka
- termin nastao kao kovanica dvije grčke riječi:
 - kryptos+graphia (tajno + pisanje)

Kriptoanaliza

- tehnika koja proučava principe i metode dešifrovanja kriptografski zaštićenih poruka bez poznavanja kriptografskog ključa

Kriptologija

- kriptografija + kriptoanaliza

Terminologija:

- otvoreni tekst (Plaintext)
- kriptovani tekst (Ciphertext)
- enkripcija, kriptovanje, šifrovanje (Encryption)
- dekripcija, dekriptovanje, dešifrovanje (Decryption)



Kriptografija

je:

- osnova za mnoge sigurnosne mehanizme
- svuda oko nas
- sigurna komunikacija, npr. HTTPS, 802.11i WPA2, ...
- enkripcija datoteka na disku
- enkripcija sadržaja u bazi podataka

nije:

- rješenje za sve sigurnosne probleme
- pouzdana ako nije pravilno implementirana
- pouzdana ako nije pravilno upotrebljena
- nešto što bi se trebalo samostalno razvijati

M - poruka

C - kriptovani tekst

E - funkcija za kriptovanje

D - funkcija za dekriptovanje

Kriptovanje: $E(M) = C$

Dekriptovanje: $D(C) = M$

Važi relacija: $D(E(M)) = M$

Kriptografski algoritmi - matematičke funkcije koje se koriste za kriptovanje i dekriptovanje.

Uopšteno, postoji dvije povezane funkcije, jedna za kriptovanje i druga za dekriptovanje.

Ograničeni (restricted) algoritam - sigurnost algoritma bazirana na čuvanju tajnosti načina rada algoritma.

- neadekvatni današnjim standardima

Algoritmi s ključem (ključevima)

Prostor ključa (keyspace) - opseg mogućih vrijednosti ključa.

Kriptovanje: $E_k(M) = C$

Dekriptovanje: $D_k(C) = M$

Važi relacija: $D_k(E_k(M)) = M$



Napadi na šifrarske sisteme

- Cilj kriptografije je da otvoreni tekst, ključ ili oboje, sačuva od napadača
- Napadač ima pun pristup komunikacionom medijumu između posiljaoca i primaoca

- Kriptoanaliza je nauka o dobijanju otvorenog teksta poruke, na osnovu kriptovanog teksta, bez poznavanja ključa
- Uspješnom kriptoanalizom dolazi se do otvorenog teksta ili do ključa
- Pokušaj kriptoanalize naziva se napadom na šifarski sistem
- Fundamentalna pretpostavka u kriptoanalizi je da tajnost mora u potpunosti da zavisi od ključa

Pasivni:

- Ciphertext-only attack
- Known-plaintext attack

Aktivni:

- Chosen-plaintext attack
- Chosen-ciphertext attack

Posebno "aktivno":

- Rubber-hose attack

Pasivni napadi na šifrarske sisteme**Ciphertext-only attack**

- Napad pri kojem je poznat samo kriptovani tekst
- Kriptoanalitičar ima kriptovani tekst nekoliko poruka, gdje su sve kriptovane korištenjem istog algoritma za kriptovanje
- Cilj kriptoanalitičara je da dođe do otvorenog teksta što većeg broja poruka, ili da dođe do ključa (ili ključeva) koji je (su) korišten(i) za kriptovanje poruka, da bi dekriptovao druge poruke kriptovane istim ključem (ključevima). tajnost mora u potpunosti da zavisi od ključa

Poznato: $C_1 = E_k(P_1), C_2 = E_k(P_2), \dots, C_i = E_k(P_i)$

Utvrđeno: P_1, P_2, \dots, P_i ili k ili algoritam po kom se može doći do $P_i + 1$, na osnovu $C_i + 1 = E_k(P_i + 1)$

Known-plaintext attack

- Napad pri kojem je poznat kriptovani i otvoreni tekst
- Kriptoanalitičar ne samo da ima kriptovani tekst nekoliko poruka, već ima i otvorene tekstove tih poruka
- Njegov cilj je da dođe do ključa (ili ključeva) korištenog za kriptovanje poruka ili da dođe do algoritma pomoću kojeg će moći da dekriptuje bilo koju novu poruku koja je kriptovana istim ključem (ili ključevima)

Poznato: $P_1, C_1 = E_k(P_1), P_2, C_2 = E_k(P_2), \dots, P_i, C_i = E_k(P_i)$

Utvrđeno: k ili algoritam po kom se može doći do $P_i + 1$, na osnovu $C_i + 1 = E_k(P_i + 1)$

Aktivni napadi na šifrarske algoritme

Chosen-plaintext attack

- Napad pri kojem je poznat kriptovani i otvoreni tekst, pri čemu kriptoanalitičar bira otvoreni tekst
- Kriptoanalitičar ne samo da ima kriptovani tekst nekoliko poruka i otvorene tekstove tih poruka, već on bira otvoreni tekst koji će biti kriptovan
- Ova vrsta napada je moćnija od napada pri kojem je poznat kriptovani i otvoreni tekst, iz razloga što kriptoanalitičar može birati specifične blokove otvorenog teksta koji će biti kriptovani, tj. bira one blokove koji bi mogli dati više informacija o ključu
- Cilj je da dođe do ključa (ili ključeva) korištenog za kriptovanje poruka ili da dođe do algoritma pomoću kojeg će moći da dekriptuje bilo koju novu poruku koja je kriptovana istim ključem

Poznato: $P_1, C_1 = E_k(P_1), P_2, C_2 = E_k(P_2), \dots, P_i, C_i = E_k(P_i)$ gdje kriptoanalitičar bira P_1, P_2, \dots, P_i

Utvrđeno: k ili algoritam po kom se može doći do $P_i + 1$, na osnovu $C_i + 1 = E_k(P_i + 1)$

Chosen-ciphertext attack

- Napad pri kom je moguće birati kriptovani tekst
- Kriptoanalitičar može da bira različite kriptovane tekstove koji će biti dekriptovani i ima pristup dekriptovanom otvorenom tekstu

Poznato: Poznato: $C_1, P_1 = D_k(C_1), C_2, P_2 = D_k(C_2), \dots, C_i, P_i = D_k(C_i)$

Utvrđeno: k

Ovaj napad je prvenstveno pogodan za asimetrične algoritme. Ponekad je efikasan protiv simetričnih algoritama.

Posebno "aktivno" napad na šifrarske sisteme

Rubber-hose attack

- Oslanja se na fizičke ili psihološke pritiske
- Napad prijetnjom i ucjenom
- Kriptoanalitičar prijeti, ucjenjuje, ili muči nekog dok ne dođe do ključa
- Ovo je često najefikasniji način da se razbije algoritam

Lecture 2.

02 - Istorijski šifrarski algoritmi - I

- transpozicioni šifarski sistemi
 - pri šifrovanju se vrši permutacija simbola otvorenog teksta
- supstitucioni šifarski sistemi
 - pri šifrovanju se svaki simbol (ili grupa simbola) mijenja drugim simbolom (ili grupom simbola)
- kombinovani

Transpozicioni šifarski sistemi

- Vrše permutaciju simbola otvorenog teksta prema nekoj šemi transformacije
- Primjenjena šema transformacije često se zasniva na nekoj od geometrijskih figura
- Da bi se izvršilo šifrovanje potrebno je da se poznaje kriptografski algoritam i kriptografski ključ
- Geometrijska figura i način upisa simbola otvorenog teksta predstavljaju kriptografski algoritam, dok određeni parametri koji pobliže određuju datu figuru predstavljaju kriptografski ključ

Primjer - simple columnar transposition cipher

Plaintext: OVO JE NAŠA PRVA JEDNOSTAVNA TEST PORUKA

Key: ABCDEFGHIJ → 0123456789

OVOJENAŠAP

RVAJEDNOST

AVNATESTPO

RUKA

Ciphertext: ORAR VVVU OANK JJAA EET NDE ANS ŠOT ASP PTO

Otvoreni tekst je zapisan horizontalno na papiru fiksne širine (matrica).

Kriptovani tekst se dobija čitanjem po vertikali, pri čemu se kolone ne moraju čitati s lijeva na desno, jedna za drugom.

Dekriptovanje se vrši zapisivanjem kriptovanog teksta vertikalno na papiru iste širine i zatim čitanjem po horizontali dolazi se do otvorenog teksta.

Plaintext: OVO JE NAŠA PRVA JEDNOSTAVNA TEST PORUKA

Key: ABCDEFGHIJ -> 0123456789

OVOJENAŠAP

RVAJEDNOST

AVNATESTPO

RUKA

Ciphertext: ORAR VVVU OANK JJAA EET NDE ANS ŠOT ASP PTO

Često se koristila i dvostruka transpozicija, tj. dvostruka primjena *simple columnar transposition cipher*-a - moguće korišćenje istog ključa dvaput ili se koriste dva ključa

Dvostruka transpozicija korištena je u II svjetskom ratu - holandski i francuski pokret otpora, kao i britanski SOE (Special Operations Executive)

Primjer - rail fence cipher

- otvoreni tekst se zapisuje naniže, pri čemu se svako slovo zapisuje u susjedni "kolosjek", do najnižeg kolosjeka, nakon čega se slova zapisuju naviše, do najvišeg kolosjeka, itd.
- postoji i varijanta kod koje se uvijek piše od najvišeg ka najnižem kolosjeku

Plaintext: JEDNOSTAVNA TEST PORUKA

J	O	V	E	O	A				
E	N	S	A	N	T	S	P	R	K
D	T	A	T	U					

Ciphertext: JOVEOA ENSANTSPRK DTATU

J	O	V	E	O	A				
E	N	S	A	N	T	S	P	R	K
D	T	A	T	U					

- kriptovani tekst se dobija čitanjem po horizontali, pri čemu se "space" koristi kao oznaka kraja reda
- Dekriptovanje se vrši zapisivanjem kriptovanog teksta u "rail fence", i zatim čitanjem po "rail fence"

Plaintext: JEDNOSTAVNA TEST PORUKA

J	O	V	E	O	A				
E	N	S	A	N	T	S	P	R	K
D	T	A	T	U					

Ciphertext: JOVEOA ENSANTSPRK DTATU

J	O	V	E	O	A				
E	N	S	A	N	T	S	P	R	K
D	T	A	T	U					

Primjer - route cipher

- otvoreni tekst je zapisan u gridu datih dimenzija
 - kriptovani tekst se dobija čitanjem karaktera sadržanih u gridu prema datom šablonu (ključu)

Plaintext: JEDNOSTAVNA TEST PORUKA



Ciphertext: UKARTTVSDEJNTNEPOSAAO

Dekriptovanje se vrši zapisivanjem kriptovanog teksta u "route", i zatim čitanjem karaktera po horizontali (ili vertikali, u zavisnosti od toga kako je tekst i zapisivan pri enkripciji)

Primjer - Myszkowski cipher

- otvoreni tekst je zapisan u gridu datih dimenzija
 - kriptovani tekst se dobija čitanjem karaktera sadržanih u gridu prema datom šablonu (ključu)

Plaintext: JEDNOSTAVNA TEST PORUKA

JEDNOST

AVNATES

Dekriptovanje se vrši zapisivanjem kriptovanog teksta vertikalno na papiru iste širine prema datom šablonu (ključu) i zatim čitanjem po horizontali

Supstitucioni šifarski sistemi

- Šifra proste zamjene - monoalfabetska (simple substitution cipher - monoalphabetic)
 - Svaki karakter otvorenog teksta zamijenjen je odgovarajućim karakterom kriptovanog teksta
 - Homofonska supstitucionna šifra (homophonic substitution cipher)

- Slična šifri proste zamjene, s tim što se svaki karakter otvorenog teksta može mapirati u jedan ili više karaktera kriptovanog teksta. Na primjer, "A" se može mapirati u 5, 13, 25, ili 56, "B" se može mapirati u 7, 19, 31, ili 42, itd.
- Poligramska šifra supstitucije (polygram substitution cipher)
 - Kriptuju se blokovi karaktera. Na primjer, "ABA" se može mapirati u "RTQ", ili "ARE" u "SLL", itd.
- Šifra višestruke zamjene - polialfabetska šifra (polyalphabetic substitution cipher)
 - Sačinjene su od više šifara proste zamjene. U okviru ove šifre primjenjuje se sukcesivno (obično periodično) više različitih prostih zamjena (ovakva šifra se često zove i Vižnerova). To je, ustvari, niz šifara.

Jednostavni suspostitucioni postupci zamjenjuju svaki simbol otvorenog teksta odgovarajućim simbolom šifrata. Šifarski postupak na bazi pomjerenog alfabetu (shifted alphabets) se zasniva na pomjerenju simbola otvorenog teksta za k poziciju udesno, po modulu veličine alfabetu:

$$E(x) = (x + k) \mod n \quad (2.1)$$

Dešifrovanje:

$$D(y) = (y - k) \mod n \quad (2.2)$$

n - veličina alfabetu A

x - označava simbol otvorenog teksta kodovan njegovom numeričkom reprezentacijom

y - označava simbol šifrovanog teksta

k - je ključ

Cezarova šifra - pomjeraj od 3

Plain:	ABCDEFGHIJKLMNPQRSTUVWXYZ
Cipher:	DEFGHIJKLMNOPQRSTUVWXYZABC

Plaintext:	hello
Ciphertext:	khoor

2006. godine šef Mafije Bernardo Provenzano je uhapšen (dijelom) zbog poruka koje su kriptovane varijantom Cezarove šifre o A->4, B->5, ...

2011. godine Rajib Karim je osuđen u UK za terorizam - koristio je Cezarovu šifru za komunikaciju sa islamskim aktivistima Bangladeša - implementacija u Excel-u

ROT-13 - pomjeraj od 13

Plain:	ABCDEFGHIJKLMNPQRSTUVWXYZ
Cipher:	NOPQRSTUVWXYZABCDEFGHIJKLM

Plaintext:	hello
Ciphertext:	uryyb

Netscape Navigator koristio ROT-13 za lozinke e-mail naloga (1999)

Netscape Navigator koristio ROT-13 za lozinke e-mail naloga (1999)

▶ Atbash cipher

- ABCDEFGHIJKLMNOPQRSTUVWXYZ
- ZYXWVUTSRQPONMLKJIHGFEDECBA

A	B	C
D	E	F
G	H	I

▶ Keyword cipher

- ABCDEFGHIJKLMNOPQRSTUVWXYZ
- KEYWORDABCDEFHIJKLMNOPQRSTUVWXYZ



▶ Pigpen / Masonic Cipher

> >

X M A R K S T H E S P O T

1	2	3	4	5
1	A	B	C	D E
2	F	G	H	I K
3	L	M	N	O P
4	Q	R	S	T U
5	V	W	X	Y Z

J	K	L
M	N	O
P	Q	R

▶ Polybius Square

- Plaintext: This is a secret message

Ciphertext: 44232443 2443 11 431513421544 32154343112215



Kriptoanaliza

- Kriptoanalizu monoalfabetskih supstitionih šifarskih postupaka je relativno jednostavno izvršiti
 - analizom frekvencije pojavljivanja simbola (slova) u otvorenom tekstu
 - analizom digrama
 - analizom trigrama

Kriptoanaliza jednostavnih supstitionih algoritama

- Brute force napad
 - za engleski alfabet broj permutacija ključeva iznosi 26!
 - $26! \approx 4.03 \times 10^{26} \approx 2^{88.4}$
 - ako se jedno dešifrovanje izvrši za jednu mikrosekundu, za kriptoanalizu šifrovane poruke bilo bi potrebno više od 10 triliona godina!
- Kriptoanaliza analizom frekvencije pojavljivanja simbola u otvorenom tekstu - prilično jednostavna i efikasna

Lecture 3.

02 - Istorijski šifrarski algoritmi - II

Homofonski šifrarski sistemi

- Kompleksniji način šifrovanja
- sprječavanje mogućnosti kriptoanalize šifrovanih poruka na osnovu analize frekvencije simbola u otvorenom tekstu
- svaki karakter otvorenog teksta može se mapirati u jedan ili više karaktera šifrovanog teksta - simbol otvorenog teksta x se mapira u jedan od, njemu dodijeljenih, skupova simbola šifrovanog teksta $f(x)$
- Plaintext: $M = m_1, m_2, \dots$
- Ciphertext: $C = c_1, c_2, \dots$

Gdje je svaki simbol c_i slučajno izabran iz skupa homofona $f(m_i)$

Homofonski šifrarski sistemi

- ▶ 26 karaktera engleskog alfabetu se mapira u 00 – 99
- ▶ Broj cijelih brojeva koji su dodeljeni jednom slovu alfabetu je proporcionalan relativnoj frekvenciji datog karaktera
- ▶ Jedan cijeli broj može da se dodijeli samo jednom karakteru alfabetu

A	10, 48, 22, 36, 24, 13, 42, 06
B	37
C	57, 77, 92
D	99, 56, 96, 30
E	89, 85, 47, 28, 91, 63, 55, 09, 07, 11, 51, 08
F	02, 01
G	52, 16
H	05, 66, 54, 32, 29, 33
I	81, 67, 00, 74, 58, 72, 71
J	69

K	25
L	76, 17, 95, 46
M	50, 15
N	04, 98, 34, 70, 53, 97
O	90, 64, 40, 18, 62, 73, 49
P	38, 93
Q	31
R	78, 20, 59, 39, 21, 26
S	88, 65, 79, 94, 83, 27
T	41, 35, 68, 19, 87, 82, 60, 23, 14, 43
U	12, 61, 84
V	44
W	86, 45
X	80
Y	03
Z	75

Homofonski šifrarski sistemi

▶ Primjer

- ▶ Plaintext: hello
- ▶ Ciphertext: 05 89 76 17 90
- ▶ Ciphertext 2: 33 08 95 46 49
- ▶ ...
- ▶ Ciphertext n: 32 63 46 17 40

Polialfabetski šifrarski sistemi

- Sačinjeni od više šifara proste zamjene
- U okviru ove šifre primjenjuje se sukcesivno (obično periodično) više različitih prostih zamjena - to je, zapravo, niz šifara
- Eliminišu preslikavanje frekvencije pojavljivanja karaktera otvorenog teksta u njihove supsticione zamjene u šifrovanom tekstu korištenjem višestrukih supstitucija

Vigenereov algoritam

- ključ K je niz simbola $K = k_1, k_2, \dots, k_d$, gdje k_i određuje broj pomjeraja u određenom alfabetu, tako da je, $f_i(x) = (x + k_i) \bmod n$
- Primjer:
 - ključ K = KEY (u numeričkom obliku K = 10 4 24),

Plaintext: hello

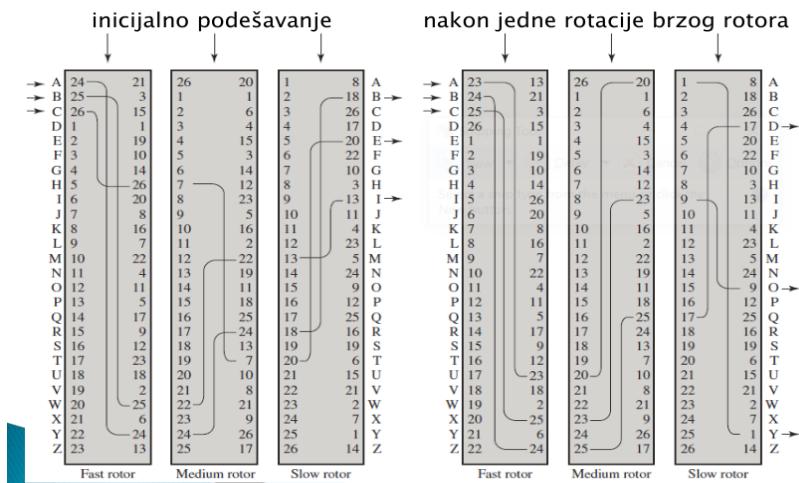
Ciphertext: uryyb

M	h e l l o
K	k e y k e
$C = E_k(M)$	r i j v s

► Vigenereova tabela

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	B	
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	C	
E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	D	
F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	E	
G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	F	
H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	G	
I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	H	
J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	I	
K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	J	
L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	K	
M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	
N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	M	
O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	N	
P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	O	
Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	P	
R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	Q	
S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	R	
T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	
U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	
V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	U	
W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	U	V	
X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	U	V	W	
Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	U	V	W	X	
Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	U	V	W	X	Y	

Rotor mašine



Poligramske šifrarske sisteme

- Šifruju se blokovi karaktera
- Na ovaj način se eliminiše preslikavanje frekvencije pojavljivanja simbola u otvorenom tekstu u odgovarajuće simbole u šifrovanom tekstu čime je otežana mogućnost kriptoanalize
- Primjer:
 - "ABA" se može mapirati u "RTQ", ili "ARE" u "SLL", itd.
 - "AA" se može mapirati u "NO", "AB", "IR", "JU" ili "AQ", itd.

Playfair šifrarski algoritam

- Ključ Playfair algoritma je bila matrica dimenzija 5×5 sa 25 slova (slovo J se ne koristi)

K	E	Y	A	B
C	D	F	G	H
I	L	M	N	O
P	Q	R	S	T
U	V	W	X	Z

- Karakteri ključa se ne ponavljaju, npr. riječ „larkspur“ će se u matrici pojavititi kao „larkspu“
- Karakteri se šifruju u digramima
- Par karaktera otvorenog teksta m_1m_2 se šifruje u skladu sa sljedećim pravilima:
 - Ako su m_1 i m_2 u istom redu, tada su c_1 i c_2 dva karaktera desno od m_1 i m_2 , respektivno (usvojeno je da je prva kolona susjedna desna kolona posljednjoj koloni)
 - Ako su m_1 i m_2 u istoj koloni, tada su c_1 i c_2 dva karaktera ispod m_1 i m_2 , respektivno (usvojeno je da je prvi red susjedni donji posljednjem redu)

- Ako su m_1 i m_2 u različitim kolonama i redovima, tada su c_1 i c_2 ostale dvije ivice (tjemena) pravougaonika koji sadrži ivice m_1 i m_2 , gde je c_1 u istom redu kao i m_1 dok je c_2 u istom redu kao i m_2
- Ako je $m_1 = m_2$, tada se vrši umetanje neutralnog (null) karaktera (na primjer, X) između m_1 i m_2 .
- Ukoliko poruka otvorenog teksta ima neparan broj karaktera tada se na kraj poruke dodaje neutralni (null) karakter.

Playfair šifarski algoritam

▶ Primjer

- ▶ Plaintext: hello
- ▶ Ciphertext: dbnvmi
- ▶ M h e l l o
- ▶ digram: h e l x l o
- ▶ $C = E_k(M)$ d b n v m i

K	E	Y	A	B
C	D	F	G	H
I	L	M	N	O
P	Q	R	S	T
U	V	W	X	Z



One-Time Pads

- *One-time pads* su supstitucioni šifarski sistemi čiji je ključ slučajna sekvenca simbola čija je dužina veća ili jednaka dužini poruke otvorenog teksta
- Izumili su ih major Joseph Mauborgne i Gilbert Vernam
- Ako se kao ključ koristi potpuno slučajan niz slova čija je dužina jednakoj dužini poruke dobija se Vernamova šifra
- Vernamov uređaj je koristio Bodov kod (*Baudot code*) sa 32 karaktera – svaki karakter je predstavljan na papirnoj traci kao kombinacija od ukupno pet bušenih i nebušenih prostora – 1 i 0
- Ključ je bio jednokratna slučajna sekvenca karaktera, koji se takođe sastojao od bušenih i nebušenih prostora (0 i 1)
- Navedeni šifarski postupak je poznat kao Vernamova šifra gde se vrši generisanje niza bita šifrovanog teksta

$$C = E_k(M) = c_1 c_2 \dots \text{ gdje je } c_i = (m_i + k_i) \pmod{2}, i = 1, 2, \dots$$

- Vernamova šifra se implementira korištenjem operacije XOR nad svakim bitom poruke otvorenog teksta i ključa:

$$c_i = m_i \oplus k_i \tag{3.1}$$

- Dešifrovanje:

$$m_i = c_i \oplus k_i \quad (3.2)$$

$$(x \oplus x = 0 \text{ i } x \oplus 0 = x, \text{ za } x = 1 \text{ ili } 0) \quad (3.3)$$

slijedi

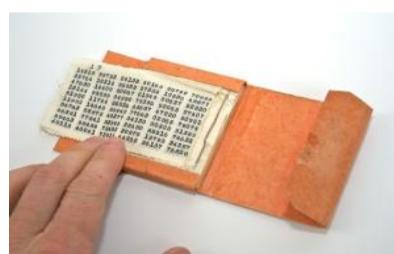
$$(c_i \oplus k_i = m_i \oplus k_i \oplus k_i = m_i \oplus 0 = m_i) \quad (3.4)$$

- $D(E(M)) = M$
- $D(m \oplus k) = m \oplus k \oplus k = m$
- Ključ se ne smije ponavljati
 - known plaintext attack i chosen plaintext attack su u tom slučaju efikasni, jer je $k = c \oplus m$
- ova šifra se teorijski ne može "razbiti" – teorijski perfektna sigurnost (perfect secrecy)
- veoma brza enkripcija
- veoma brza dekripcija
- u praksi nije upotrebljiva
 - $\text{length}(k) \geq \text{length}(m)$
 - dužina ključa \geq dužina otvorenog teksta
- Uslovi za perfektnu sigurnost OTP-a
 - dužina ključa veća ili jednaka dužini poruke
 - OTP bi se trebao sastojati od slučajno generisanih brojeva (karaktera)
 - tačno dvije kopije OTP-a moraju postojati
 - OTP se smije koristiti samo jednom
 - obe kopije OTP-a moraju su uništiti nakon upotrebe

One-Time Pads

- ▶ Primjer
- ▶ Ako je karakter A otvorenog teksta (11000 u Bodovom kodu) šifrovan primjenom karaktera D ključa (10010 u Bodovom kodu), rezultujući karakter šifrovanog teksta je:

$$\begin{aligned} M &= 11000 \\ K &= 10010 \\ C &= 01010 \end{aligned}$$



Lecture 4.

03 - Steganografija

- tehnika skrivanja tajne poruke u nekom tekstu, slici, audio/video zapisu, itd.
- multimedijalne datoteke u pravilu sadrže neupotrebljene ili nevažne dijelove koje različite steganografske tehnike koriste tako da ih popune tajnim informacijama



Kriptografija – skriva se značenje poruke.
 Steganografija - skriva se postojanje poruke.

Podjela prema vrsti medija:

- Steganografija u slikama (Image Steganography)
 - informacije se skrivaju unutar digitalnih slika, najčešće mijenjanjem LSB bita piksela
- Steganografija u audio datotekama (Audio Steganography)
 - informacije se kriju unutar audio datotekama
- Steganografija u video datotekama (Video Steganography)
 - informacije se kriju u video datotekama
- Steganografija u tekstu (Text Steganography)
 - podaci se skrivaju u tekstualnim dokumentima
- Steganografija u mrežnom saobraćaju (Network Steganography)
 - koristi mrežni protokol za prenos skrivenih informacija

tekst

Metode bazirane na formatiranju:

- Line-Shift Coding
 - pomjeraj svake parne linije naviše ili naniže koji označava 1 ili 0, dok se neparne linije ne pomjeraju – originalan dokument nije potreban u procesu dekodovanja
- Word-Shift Coding

- svaka parna riječ se pomjera lijevo ili desno i tako označava 1 ili 0, dok se neparne riječi ne pomjeraju
- Feature Coding
 - promjene individualnih karaktera – npr. dužina horizontalne linije slova „t“, dimenzije tačke u slovima „i“ i „j“
- Open method
 - ubacivanje bjelina između riječi, na kraju linije i sl.
- Luminance Modulation Coding
 - baziran na boji slova

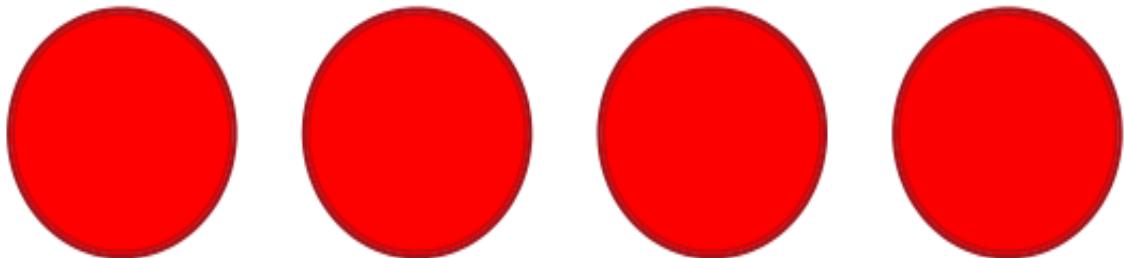
Lingvističke metode-bazirane na sintaksnim i semantičkim osobinama teksta

Sintaksne-ubacivanje ili uklanjanje zareza,zamjena znakova interpunkcije

Semantičke-sinonimi – jedna riječ označava 1, a druga 0

File concatenation steganografija.

Slika.



Primjer

Steganografija – primjer

- ▶ **LSB supstitucija (Least Significant Bit)**
- ▶ **nosilac (8x8 bita):**
 - 10010101 00001101 11001001 10010110
 - 00001111 11001011 10011111 00010000
- ▶ **tajna poruka (8 bita):**
 - slovo z –> ASCII – 122: 01111010
- ▶ **nosilac + tajna poruka (3 lsb bita promijenjena u nosiocu):**
 - 10010100 00001101 11001001 10010111
 - 00001111 11001010 10011111 00010000

- može se kombinovati s kriptografijom:
 - primjer: poruka se prvo kriptuje, pa se onda sakrije u nosiocu
- može se koristiti za kreiranje vodenih žigova (digital watermarking)

Lecture 5.

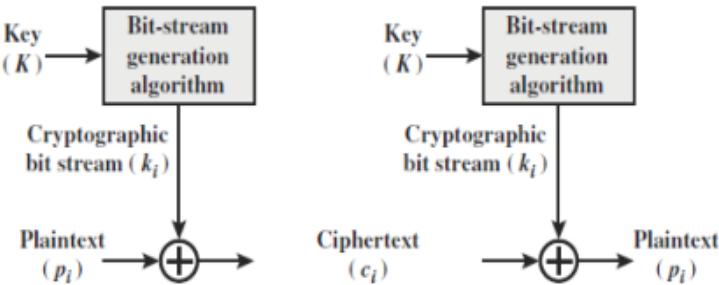
04 - Moderni simetrični šifrarski sistemi - I

Simetrični algoritmi

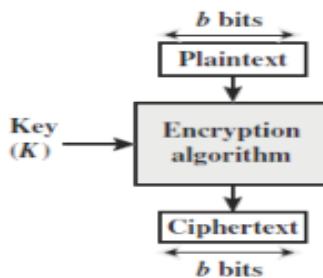
- Ključ se naziva tajnim ili dijeljenim ključem
- Simetrični šifrarski sistemi nazivaju se i šifarskim sistemima sa tajnim ključem
- Sigurnost kriptografskog sistema bazira se na tajnosti ključa, a ne na tajnosti algoritma
- Dužina ključa je najbitnija komponenta – što je ključ duži to je šifrovana poruka sigurnija
- Potencijalni problem – distribucija dijeljenog ključa

- **Podjela:**
 - stream cipher
 - block cipher

- **Stream cipher:**
 - procesiranje poruka bit po bit ili bajt po bajt
 - primjer: OTP – problem distribucije velikog ključa
 - ideja: koristiti ključ male bitske dužine i koristiti PRN generator



- **Block cipher:**
 - poruka se dijeli na blokove koji se procesiraju
 - * tipično 64 ili 128-bitni blokovi
 - većina modernih simetričnih algoritama su blok šifrarski algoritmi



Ključevi

Broj mogućih kombinacija ključa:

◦ 56 bita	72 057 594 037 927 936
◦ 128 bita	3.40×10^{38}
◦ 256 bita	1.16×10^{77}
◦ 512 bita	1.34×10^{150}
◦ 1024 bita	1.80×10^{308}
◦ 2048 bita	3.23×10^{616}

- Ciljevi moderne kriptografije:
 - jedan ključ se može koristiti više puta i
 - da se pomoću kratkog ključa može šifrovati duga poruka, a da pri tome sistem bude dovoljno siguran
- Kada se sistem može smatrati dovoljno sigurnim?

Primitivne operacije za šifrovanje

- **Konfuzija** – cilj je da se veza između ključa i kriptovanog teksta učini što je moguće kompleksnijom – konfuzija čini izlaz zavisnim od ključa, idealno jedan bit ključa utiče na sve bite izlaza – konfuzija otežava pronalazak ključa iz kriptovanog teksta
 - shift cipher – osnovna operacija je supstitucija
 - s-box-ovi u modernim algoritmima (supstitucija)
- **Difuzija** – preraspoređivanje bita ulazne poruke na takav način da se svaka redundantnost u otvorenom tekstu raširi preko kriptovanog teksta – difuzija čini izlaz zavisnim od prethodnih ulaza (otvoren ili plain text), idealno svaki bit ulaza utiče na sve bite izlaza - difuzija otežava pronalazak otvorenog teksta iz kriptovanog teksta

- transpozicija – mijenjanje pozicije slova u otvorenom tekstu o p-box-ovi u modernim algoritmima (permutacija)

- Kombinacija konfuzije i difuzije je dobra praksa za dobijanje sigurne šeme
 - dobri primjeri su DES i AES
- simetrični algoritmi **moraju biti kompleksni** – kako ne bi mogli biti jednostavno analizirani
- simetrični algoritmi **moraju biti jednostavniji** – kako bi mogli biti lako implementirani
- zato moderni algoritmi koriste “mješavinu” difuzije i konfuzije, i to kroz nekoliko rundi, kako bi postigli potrebnu kompleksnost uz dodatnu (ali malu) cijenu nešto kompleksnije implementacije

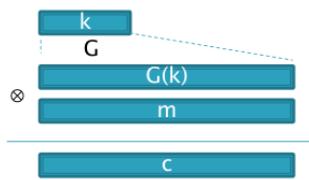
- efekat lavine (avalanche effect)
 - funkcija ima dobar efekat lavine, ako promjena jednog bita ulaza rezultira promjenom polovine bita izlaza, minimalno
- efekat kompletnosti (completeness effect)
 - svaki izlazni bit je kompleksna funkcija svih ulaznih bita
- dobro dizajniran algoritam ispunjava:
 - efekat lavine
 - efekat kompletnosti
 - nepredvidljivost - slučajnost
- loše dizajniran algoritam – ima suviše mnogo predvidljivosti
- mnogi algoritmi su “razbijeni”
- čak i eksperti griješe
- preporuka: ne razvijati svoj algoritam, već koristiti dokazano sigurne algoritme

Stream cipher

- Ideja: koristiti ključ male bitske dužine i koristiti PRN generator
- PRNG

$$G : \underbrace{\{0, 1\}^s}_{\text{seedspace}} \rightarrow \{0, 1\}^n, n \gg s \quad (5.1)$$

- slučajan ulaz + deterministički algoritam -> izlaz bi trebao izgledati "slučajan"
- od k bita slučajnog ključa -> primjenom G dobija se $G(k)$
- $c = E(k, m) = m \oplus G(k)$
- $D(k, c) = c \oplus G(k) = m$



- Da li stream cipher može imati perfektnu sigurnost?
 - da ako je PRNG apsolutno "siguran"
 - ne postoje algoritmi sa perfektnom sigurnošću
 - da, svi algoritmi je imaju
 - ne, jer je ključ kraći od poruke
- PRNG mora biti nepredvidiv – uslov za sigurnost (perfektna sigurnost nije moguća – moguća je samo za ključeve koji su $\geq m$)
- ako je poznato prvih i bita $G(k)$, i ako je moguće odrediti bite $i + 1, \dots, n$, onda je PRNG predvidiv
- u tom slučaju bi na osnovu kriptovanog teksta i dijela otvorenog teksta mogli doći do prvih i bita $G(k)$ – nakon toga možemo izračunati ostatak bita $G(k)$, te dekriptovati kompletan kriptovani tekst
 - primjer slanje mail poruke u SMTP protokolu počinje sa MAIL FROM: - ako imamo šifrat kompletne poruke, možemo odrediti prvih i bita $G(k)$, pa ako je PRNG predvidiv, možemo doći do otvorenog teksta poruke

Napadi

- **two time pad napad**
 - $c_1 = m_1 \oplus PRNG(k)$
 - $c_2 = m_2 \oplus PRNG(k)$
 - korišten je isti ključ dva puta...
 - prisluškivanje – napadač dolazi u posjed c_1 i c_2
 - $c_1 \oplus c_2 = m_1 \oplus m_2$
 - dovoljno redundantnosti u engleskom jeziku i u ASCII omogućava da se iz $m_1 \oplus m_2$ dobije $m_1 m_2$
- **no integrity napad (bit-flipping napad)**
 - enkripcija sa $\oplus k = (m \oplus k)$
 - napadač vrši $\oplus p \geq (m \oplus k) \oplus p$
 - dekripcija sa $\oplus k \geq m \oplus p$
 - modifikacije nad kriptovanim tekstrom se ne mogu detektovati, a imaju predvidljiv uticaj na otvoreni tekst
 - * napadač sada jednostavno može doći do m
 - * isto tako, napadač može izmjenom kriptovanog teksta obmanuti primaoca poruke, a da primalac poruke to ne detektuje
 - original: From Bob
 - napadač želi da primalac vidi From Eve
 - Bob -> 42 6f 62 Eve -> 45 76 65 Bob \oplus Eve -> 07 19 07

- 07 19 07 treba da zamijeni 42 6f 62 (Bob) i primalac će misliti da je poruka stigla od Eve, a ne od Boba

- generalizacija OTP-a
- inicijalizacija relativno kratkim ključem
- na osnovu ovog ključa generiše se keystream
- keystream se koristi kao OTP
 - XOR za enkripciju/dekripciju
- jednostavni i brzi
- preporuke:
 - ne koristiti ključ stream cipher-a više od jednom
 - mrežni saobraćaj: novi ključ za svaku sesiju (npr. TLS)

RC4

- stream cipher, 1987. godine
- razvio ga je Ronald Rivest, RSA
- jednostavan
- na isti način vrši se enkripcija i dekripcija, jer se stream podataka samo XOR-uje sa generisanim ključem (sekvencom)
- koristi se u različitim rješenjima i standardima, poput IEEE 802.11 sa WEP (Wired Equivalent Privacy) protokolom, HTTPS-u, RDP, Kerberos, Bit Torrent
 - koristi/o se u varijantama sa 40-bitnim i 128-bitnim ključevima
- postoje javno dostupne procedure i alati za “razbijanje” ovog algoritma
- preporuke:
 - ne koristiti ključ stream cipher-a više od jednom
 - mrežni saobraćaj: novi ključ za svaku sesiju (npr. TLS)

- ključ promjenljive dužine od 1-256 bajta (8-2048 bita) koristi se za inicijalizaciju vektora stanja S koji se sastoji od 256 elemenata S[0], S[1],...S[255], ovih max 2048 bita se koriste interno od PRNG
- u svakom trenutku, S sadrži permutovane 8-bitne vrijednosti brojeva od 0 do 255
- za enkripciju i dekripciju koristi se bajt k koji je generisan iz S tako što se bira jedan od 255 ulaza iz S
- nakon svakog izabranog k vrši se nova permutacija ulaza u S
- sastoji se iz 2 faze:
 - KSA – key-scheduling algorithm
 - PRGA – pseudo-random generation algorithm
- postoje javno dostupne procedure i alati za “razbijanje” ovog algoritma

- preporuke:
 - ne koristiti ključ stream cipher-a više od jednom
 - mrežni saobraćaj: novi ključ za svaku sesiju (npr. TLS)

RC4

- ▶ KSA – inicijalizacija i permutacija elemenata vektora stanja

```
for i from 0 to 255
    S[i] := i
endfor
j := 0
for i from 0 to 255
    j := (j + S[i] + key[i mod keylength]) mod 256
    swap (S[i], S[j])
endfor
```

- ▶ keylength – broj bajta ključa ($1 \leq \text{keylength} \leq 256$)

RC4

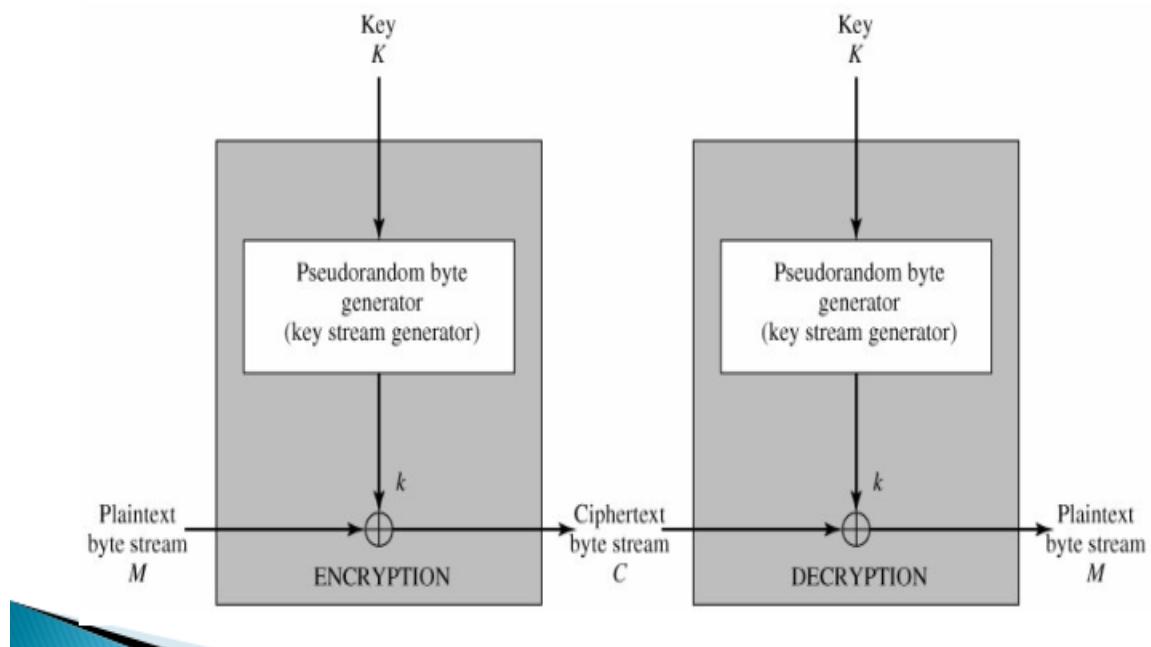
- ▶ PRGA – generiše keystream koji se koristi za enkripciju/dekripciju

```

i := 0
j := 0
while GeneratingOutput:
    i := (i + 1) mod 256
    j := (j + S[i]) mod 256
    swap (S[i], S[j])
    k := S[(S[i] + S[j]) mod 256]
    output k
endwhile

```

- ▶ enkripcija



RC4 - dekripcija

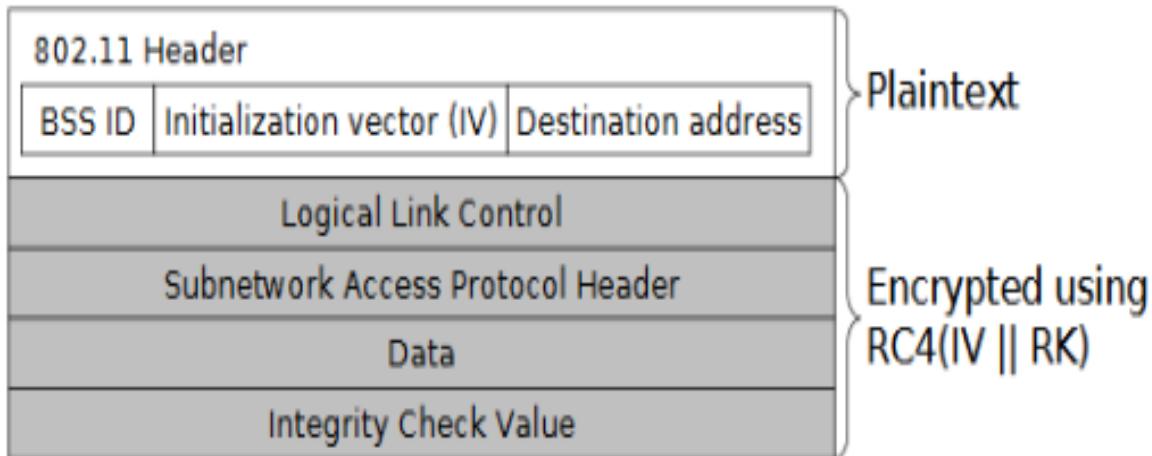
- koristi se isti ključ
- generiše se keystream pomoću KSA i PRGA
- keystream se XOR-uje sa kriptovanim tekstom i dobija se otvoreni tekst:
 - $(M \text{ XOR } k) \text{ XOR } k = M$

RC4 - slabosti

- vjerovatnoća da je drugi bajt jednak 0 je $2/256$, a trebalo bi da je $1/256$, jer imamo 256 mogućih bajta -> vjerovatnoća da drugi bajt poruke uopšte nije kriptovan je dvostruko veća nego što bi trebalo
 - slično je i sa prvim i trećim bajtom
 - preporuka – ne koristiti RC4, a ako je to neophodno, ignorisati prvih 256 bajta ključa
- vjerovatnoća da se u kriptovanom tekstu pojave dva bajta $[0, 0]$, jedan do drugog je veća nego što bi trebala biti – umjesto $1/(256)^2$ iznosi $1/(256)^2 + 1/(256)^3$
- Fluhrer, Mantin & Shamir – prvih nekoliko bajta keystream-a nisu slučajni i daju informaciju o ključu
 - 1 000 000 poruka – moguće doći do ključa
- Klein-ov napad – dodatne korelacije između keystream-a i ključa
 - 104-bitni ključ se otkriva nakon 40 000 poruka sa vjerovatnoćom od 50 %, a nakon 85 000 poruka sa vjerovatnoćom od 95%.
- bit-flipping napad

RC4 - WEP

- WEP ključevi se mijenjaju ručno – ovo se rijetko dešava – zato se koristi 24-bitni IV koji se spaja sa 40-bitnim ili 104-bitnim ključem
 - 24-bitna = $16\ 777\ 216$ različitih vrijednosti ključa – Birthday paradox – vjerovatnoća od 50 % da će se ponoviti nakon manje od 5000 paketa
 - Known Plain-text attack
 - * napadač generiše PT, AP generiše $C = PT \text{ XOR } \text{RC4}(IV \parallel \text{Key})$ – IV je poznat napadaču jer je poslat kao plaintext
 - * kad napadač uhvati drugi paket C_2 sa istim IV, onda može da uradi $PT_2 = C_2 \text{ XOR } \text{RC4}(IV \parallel \text{Key})$
 - * IV može biti i sekvencijalan (na nekim uređajima) – počinje od 0 nakon restarta i raste za 1



Moderni stream cipher: eStream

- eStream projekat okončan 2008. godine
- 7 različitih stream cipher-a
- PRNG:

$$G : \{0, 1\}^s * R \rightarrow \{0, 1\}^n \quad (5.2)$$

↓ ↓
 seed nonce

- Nonce – vrijednost koja se neće ponoviti dok god je ključ fiksan
- $E(k, m; r) = m \oplus PRG(k; r)$
- par (k, r) se ne koristi više od jednog puta
- zbog toga ključ može biti reupotrebljavan

Moderni stream cipher: eStream

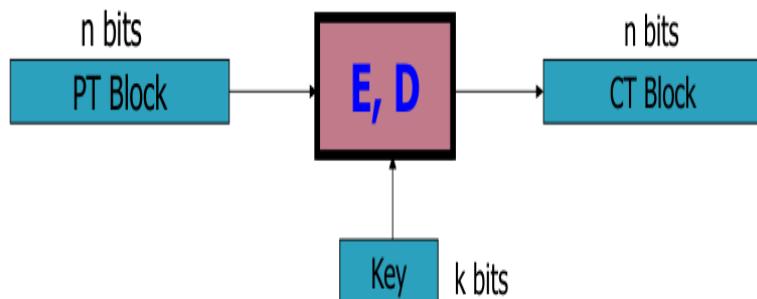
- Salsa20, HC-128, SOSEMANUK, Rabbit, Grain, MICKEY, Trivium

‣ AMD Opteron, 2.2 GHz (Linux)

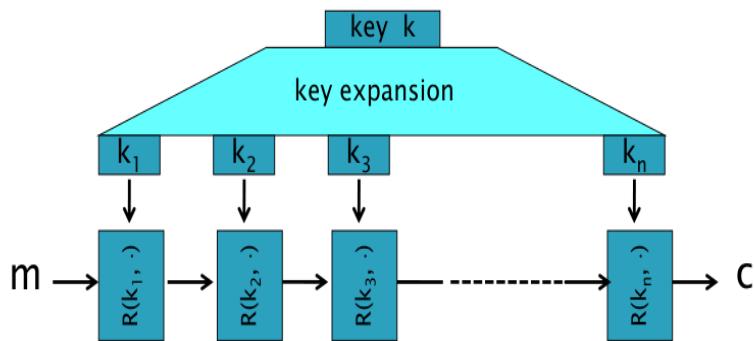
Algoritam	Brzina(MB/sec)
RC4	126
Salsa20/12	643
Sosemanuk	727

Block cipher

- blok iste bitske dužine na ulazu, kao i na izlazu – nakon enkripcije, odnosno nakon dekripcije



- **Primjeri:**
 - 3DES: n= 64 bits, k = 168 bits
 - AES: n=128 bits, k = 128, 192, 256 bits
- iz ključa k nastaju ključevi runde $k_1 - k_n$



- enkripcija se odvija tako što se ista funkcija (round function $R(k,m)$) primjenjuje kroz veći broj runde
 - m je različito kroz runde – izlaz iz jedne runde je ulaz za narednu ($m_1 - m_n$)
- za DES -> $n=16$, za 3DES -> $n=48$
- za AES-128 -> $n=10$, za AES-192 -> $n=12$, za AES-256 -> $n=14$

Block cipher

AMD Opteron, 2.2 GHz (Linux)

<u>Algoritam</u>	<u>Blok/ključ</u>	<u>Brzina (MB/sec)</u>
3DES	64/168	13
AES-128	128/128	109

► stream cipher-i znatno brži od block cipher-a

Data Encryption Standard

- Data Encryption Standard
- FIPS-46-3 (Federal Information Processing Standards Publication 46-3)
- dužina ključa 56 bita
 - smatra se nedovoljno sigurnom
- 3DES – Triple DES – TDEA
 - Svaki blok poruke prolazi kroz tri uzastopne DES iteracije

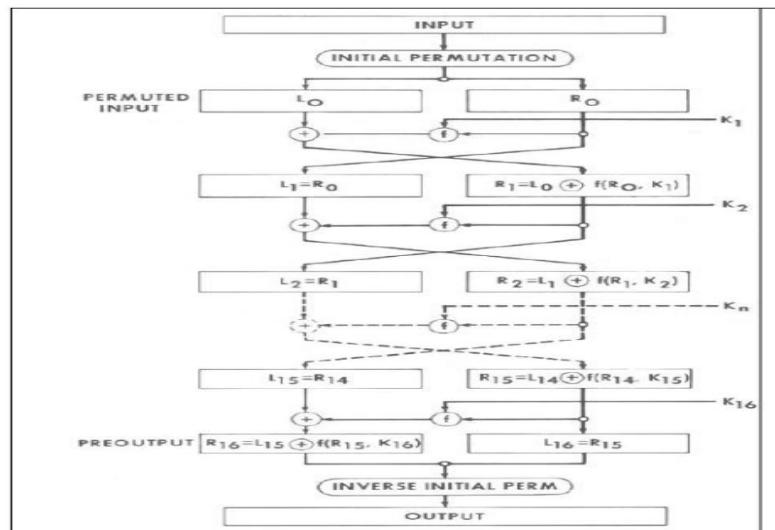
- NIST-ov (U.S. National Institute of Standards and Technology) standard
- početkom 70-ih Horst Feistel je dizajnirao Lucifer algoritam
 - 128-bitni ključ, 128-bitni blok
- 1973. g. NBS (National Bureau of Standards), danas NIST (od 1901 -1988 NBS, kasnije NIST), traži prijedlog block algoritma
- IBM predlaže varijantu Lucifer algoritma
- uticaj NSA
 - smanjena dužina ključa, sa originalnih 128 bita u Lucifer algoritmu, na 56 bita
 - izvršene modifikacije nekih od DES S-boksova - kreiran trap-door koji je omogućavao dešifrovanje šifrovanog teksta bez poznavanja ključa (tvrde Hellman i Diffie)
 - ovo je i dalje nejasno...
- 1976. g. usvojen je kao federalni standard
- dizajn DES-a čuvan kao tajna duže od 20 godina
- 1997. godine – “razbijen”
- osnovna ideja – Feistel network

Lecture 6.

04 - Moderni simetrični šifrarski sistemi - II

Data Encryption Standard

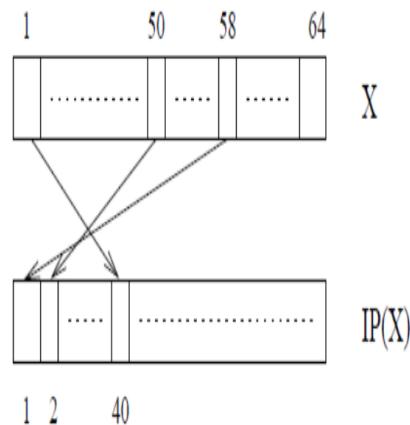
- blok-šifarski algoritam
- dužina bloka: 64 bita
- faze šifrovanja:
 - inicijalna permutacija – IP
 - kompleksna ključ-zavisna izračunavanja – u 16 iteracija
 - inverzna permutacija – IP-1
- ključ-zavisno izračunavanje sastoji se od:
 - funkcije f – šifrovanje
 - funkcije KS – key schedule
- notacija
 - LR predstavlja bite koji su nastali konkatenacijom L i R bita



DES - šifrovanje

- inicijalna permutacija 64 bita ulaznog bloka
- 1. bit izlaza iz IP je 58. bit ulaza, 2. bit izlaza je 50. bit ulaza,..., posljednji bit na izlazu je 7. bit ulaza

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



- izlaz iz IP je ulaz u kompleksno ključ-zavisno izračunavanje, a izlaz iz kompleksnog ključ-zavisnog izračunavanja (naziva se preoutput) je ulaz u inverznu permutaciju

- ulaz u kompleksno ključ-zavisno izračunavanje je izlaz iz IP
- rezultat kompleksnog ključ-zavisnog izračunavanja je preoutput blok
- sastoji se iz 16 iteracija izračunavanja
- funkcija f – šifrovanje
 - operiše nad dva bloka – jedan od 32 i jedan od 48 bita – rezultat je blok od 32 bita
- funkcija KS – key schedule
- ulaz u kompleksno ključ-zavisno izračunavanje
 - LR blok (64 bita) sastoji se iz L (32 bita) i R (32 bita)
- neka je K blok od 48 bita izdvojenih na određen način iz 64-bitnog ključa
- onda je izlaz iz iteracije L'R':

$$L' = RR' = L \oplus f(R, K) \quad (6.1)$$

- ako je L'R' izlaz iz 16. iteracije, onda je R'L' preoutput blok

$$R_{16} = L_{15} \oplus f(R_{15}, K_{16})R' = L \oplus f(R, K) \quad (6.2)$$

- U svakoj iteraciji različit blok K bita ključa kreiran je iz 64-bitnog ključa KEY
- Neka je KS funkcija koja kao argumente prima integer n u opsegu od 1 do 16 i 64-bitni blok KEY, i koja vraća 48-bitni blok Kn koji predstavlja permutovanu selekciju bita iz KEY

$$K_n = KS(n, KEY) \quad (6.3)$$

- KS se naziva key schedule i kreira 16 K_n ključeva potrebnih za 16 iteracija
- K_n je ključ koji se koristi u n-toj iteraciji
- Otuda je:

$$L_n = R_{n-1} \quad (6.4)$$

$$R_n = L_{n-1} \oplus f(R_{n-1}, K_n) \quad (6.5)$$

- Permutacija IP^{-1} koja se primjenjuje na preoutput blok je inverzna IP permutaciji koja se primjenjuje na ulaz

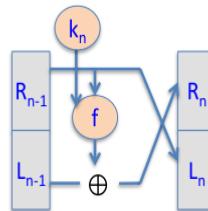
DES - dešifrovanje

DES – dešifrovanje

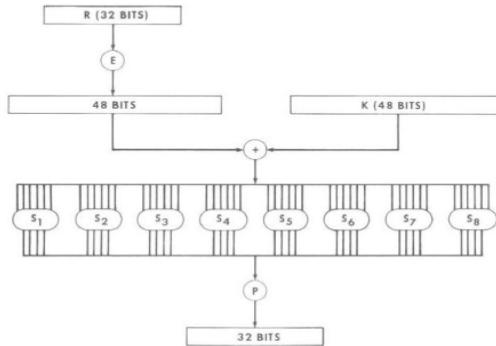
- ▶ Iz

$$\begin{aligned} L_n &= R_{n-1} \\ R_n &= L_{n-1} \otimes f(R_{n-1}, K_n) \end{aligned}$$
- ▶ slijedi da je

$$\begin{aligned} R_{n-1} &= L_n \\ L_{n-1} &= R_n \otimes f(L_n, K_n) \end{aligned}$$
- ▶ za dešifrovanje je potrebno da se isti algoritam primijeni na šifrovani blok, pri čemu se u svakoj iteraciji koristi isti blok K bita ključa koji je korišten i pri šifrovanju bloka
- ▶ $R_{16}L_{16}$ je permutovani ulazni blok za dešifrovanje, a L_0R_0 je preoutput blok
- ▶ K_{16} je ključ u prvoj iteraciji, K_{15} u drugoj, a K_1 u 16-toj iteraciji



DES – funkcija f



DES – dešifrovanje

- ▶ 1. E – ekspanzija (i permutacija)
 - 32 bita (desna strana) se proširuje (i permutuje) na 48 bita

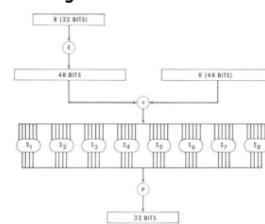
$E(R_{n-1})$

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

DES – dešifrovanje

- ▶ 2. korištenje ključa iteracije K_n
 - 48 bita je XOR-ovano sa 48 bitnim ključem iteracije

$K_n \otimes E(R_{n-1})$



- ▶ 3. dijeljenje
 - rezultirajućih 48 bita dijeli se u grupe po 6 bita – 8 x 6 bita – svaka vrsta predstavlja bite koji ulaze u odgovarajući S-box

$B1\ B2\ B3\ B4\ B5\ B6\ B7\ B8 = K_n \otimes E(R_{n-1})$

DES – dešifrovanje

- ▶ 4. S-box
 - svakih 6 bita ulaze u S-box koji kreira 4-bitni rezultat
 - S – Substitution
 - prvi i posljednji bit ulaza određuju vrstu, dok biti 2–5 određuju kolonu, tj. koji se biti iz izabrane vrste uzimaju kao 4-bitni rezultat
- ▶ Ako su S_1 do S_8 funkcije definisane tabelama (sljedeći slajd), B blok od 6 bita, onda $S_1(B)$ čine
 - prvi i posljednji bit od B daju broj u opsegu 0–3. Neka je označen sa i
 - srednja četiri bita od B daju broj u opsegu od 0–15. Neka je označen sa j
 - broj u tabeli na poziciji i,j je broj u opsegu 0–15, a to je 4-bitni blok $S_1(B1)S_2(B2)S_3(B3)S_4(B4)S_5(B5)S_6(B6)S_7(B7)S_8(B8)$

Primjer

ulaz u S_1 je **011000**

$i = 00 = 0$

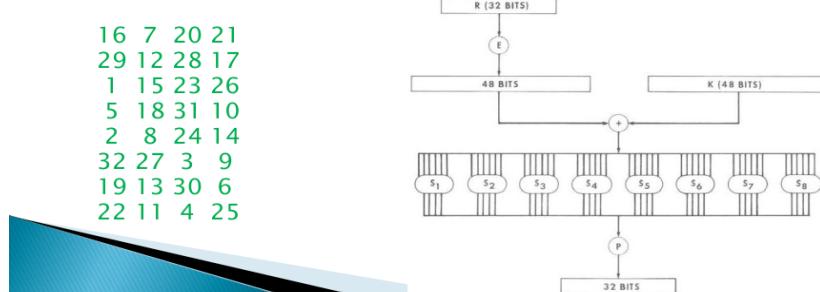
$j = 1100 = 12$

$S_1(B) = 5 = 0101$

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

DES – dešifrovanje

- ▶ 5. P-box
 - 32 bita (izlaz iz 8 S-boxova po 4 bita) se kombinuju i permutuju
 - P = Permutation
 - 8×4 bita iz S-boxova se permutuju na sljedeći način:
 $f = P(S_1(B_1)S_2(B_2)S_3(B_3)S_4(B_4)S_5(B_5)S_6(B_6)S_7(B_7)S_8(B_8))$



DES - ključevi iteracija

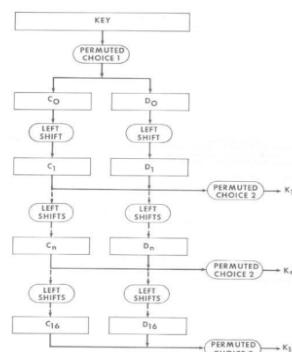
DES – ključevi iteracija

- ▶ $K_n = KS(n, KEY)$, gdje je $n=1, 2, \dots, 16$
- ▶ jedan bit svakog bajta iz KEY može da se koristi za detekciju greške (*odd parity* – biti 8, 16, ..., 64)
- ▶ permutacija PC 1 – permutacija 56 bita ključa

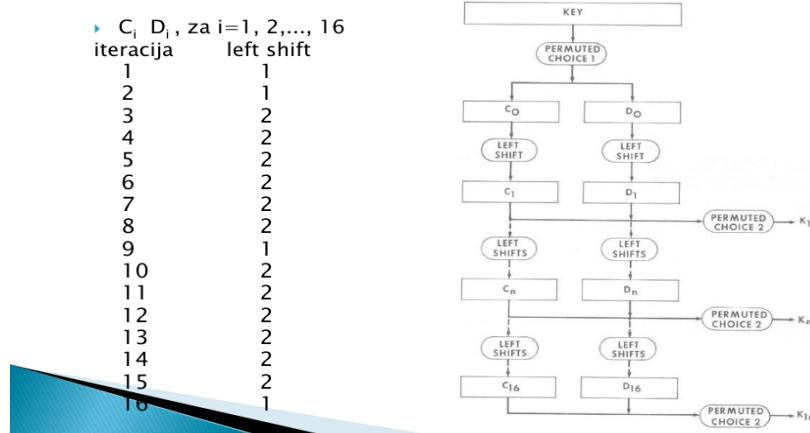
PC 1

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36

63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

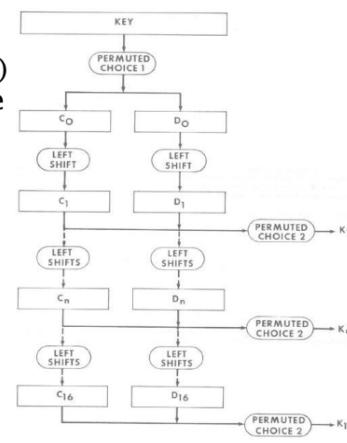
 C_0 – 28 bita D_0 – 28 bita

DES – ključevi iteracija



DES – ključevi iteracija

- ▶ Permutated choice 2
 - ▶ PC 2
 - ▶ ulaz C_i (28 bita) i D_i (28 bita)
 - ▶ izlaz – 48 bitni ključ iteracije
- | | | | | | |
|----|----|----|----|----|----|
| 14 | 17 | 11 | 24 | 1 | 5 |
| 3 | 28 | 15 | 6 | 21 | 10 |
| 23 | 19 | 12 | 4 | 26 | 8 |
| 16 | 7 | 27 | 20 | 13 | 2 |
| 41 | 52 | 31 | 37 | 47 | 55 |
| 30 | 40 | 51 | 45 | 33 | 48 |
| 44 | 49 | 39 | 56 | 34 | 53 |
| 46 | 42 | 50 | 36 | 29 | 32 |



DES - slabi ključevi

DES – slabi ključevi

- ▶ ključevi čija primjena rezultira identičnim ključevima iteracija
 - 1. 0101010101010101
16 ključeva iteracija: 0000000000000000
 - 2. FEFEFEFEFEFEF
16 ključeva iteracija : 3f3f3f3f3f3f3f
 - 3. EOEEOEOEOF1F1F1
16 ključeva iteracija : 3f3f3f00000000
 - 4. 1F1F1FOE0EOEOE
16 ključeva iteracija: 00000003f3f3f
 - 5. 0000000000000000
16 ključeva iteracija: 0000000000000000
 - 6. FFFFFFFFFFFFFF
16 ključeva iteracija : 3f3f3f3f3f3f
 - 7. E1E1E1E1FOFOFO
16 ključeva iteracija : 3f3f3f00000000
 - 8. FFFFFFFFFFFFFF
16 ključeva iteracija : 00000003f3f3f

DES - polu-slabi ključevi

DES – polu-slabi ključevi

- ▶ parovi ključeva za koje važi da je
 $E_{k1}(E_{k2}(M))=M$
 - 1. 0x011F011F010E010E i 0x1F011F010E010E01
 - 2. 0x01E001E001F101F1 i 0xE001E001F101F101
 - 3. 0x01FE01FE01FE01FE i 0xFE01FE01FE01FE01
 - 4. 0x1FE01FE00EF10EF1 i 0xE01FE01FF10EF10E
 - 5. 0x1FFE1FFE0EF0EFE i 0xFE1FFE1FFEOEF0E
 - 6. 0xE0FEE0F0F1F0F1F i 0xFEE0FEE0F0F1F0F1

DES challenge

DES challenge

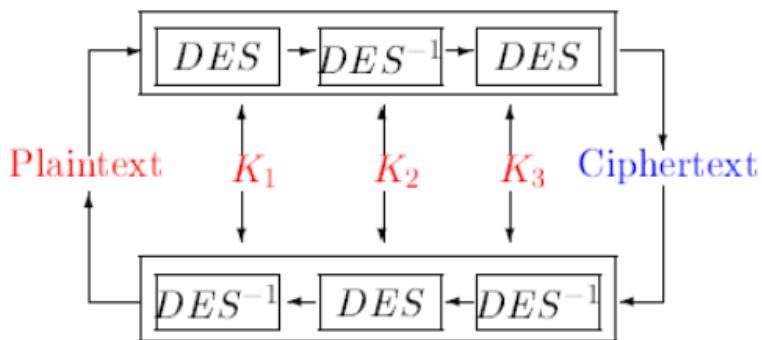
➤ exhaustive search
 $m = "The unknown messages is:| XXXX ... "$
 $CT = c_1 \quad c_2 \quad c_3 \quad c_4$

Cilj: pronaći $k \in \{0,1\}^{56}$ takvo da je
 $DES(k, m_i) = c_i \text{ for } i=1,2,3$

1997: Internet pretraga – distribuirano – **3 mjeseca**
1998: EFF mašina ("Deep Crack") – **3 dana** (250 000 \$)
1999: kombinovano – **22 sata**
2006: COPACOBANA (120 FPGAs) – **7 dana** (10 000 \$) – University of Bochum and Kiel
2016: open-source password cracking software hashcat – brute force – general purpose GPUs – NVIDIA GTX1080Ti GPU (\$1 000) – **15 dana** – 8 x 1080Ti GPU – **manje od 2 dana**

3DES - Triple DES - TDEA

- Neka $E_K(M)$ predstavlja DES šifrovanje poruke M korišćenjem ključa K, a $D_K(C)$ dešifrovanje šifrata C korišćenjem ključa K $C = E_{K3}(D_{K2}(E_{K1}(M)))$
- dekripcija: $M = D_{K1}(E_{K2}(D_{K3}(C)))$

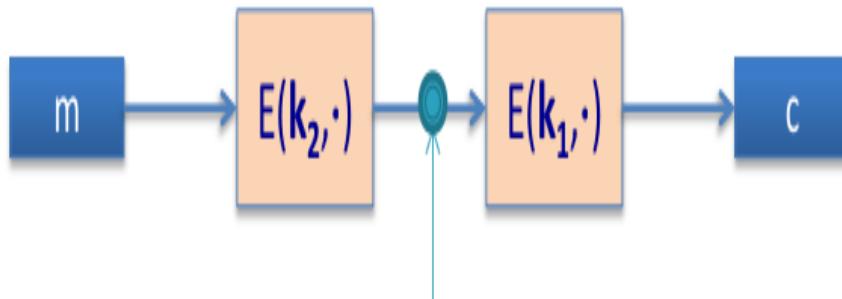
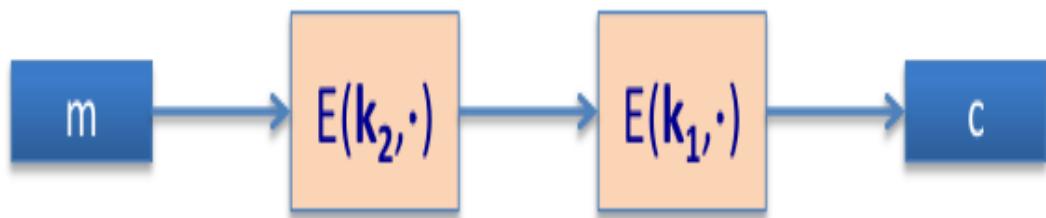


- 3DES – TDEA - kombinacija šifrovanja DES-om i dešifrovanja DES-om
- $3 \times 56\text{-bita} \Rightarrow 168 \text{ bita} \Rightarrow 3 \text{ puta sporiji od DES-a}$
- moguć broj ključeva 2^{168} , postoji MITM napad koji ovaj broj smanjuje na 2^{112}
- ključevi - opcije:
 - K1,K2,K3 – nezavisni ključevi
 - K1,K2, - nezavisni ključevi i K3=K1

$K1=K2=K3 \Rightarrow$ u ovom slučaju TECB, TCBC, TCFB i TOFB modovi su kompatibilni sa DES modovima ECB, CBC, CFB, OFB

Zašto ne Double DES?

- Neka je Double DES:
 - $2E((k_1, k_2), m) = E(k_1, E(k_2, m))$
 - dužina ključa – 112 bita
 - radio bi duplo sporije od DES-a
 - exhaustive search 2^{128} – mnogo – mogao bi se smatrati praktično sigurnim



- Napadač ima: $M = (m_1, \dots, m_{10}), C = (c_1, \dots, c_{10})$
- Cilj: pronaći ključeve k_1 i k_2 takve da je
 - $E(k_1, E(k_2, M)) = C$
- drugačije napisano (primjenom dekripcije):
 - $E(k_2, M) = D(k_1, C)$
- ovo je meet-in-the-middle
- tabela svih $E(k_2, \cdot)$
- sortiranje tabele

$k^0 = 00\dots00$	$E(k^0, M)$
$k^1 = 00\dots01$	$E(k^1, M)$
$k^2 = 00\dots10$	$E(k^2, M)$
\vdots	\vdots
$k^N = 11\dots11$	$E(k^N, M)$

2^{56}
ulaza

- nakon toga, u obrnutom pravcu – dekripcija $D(k_1, \cdot)$ – za svaku potencijalnu dekripciju se provjerava da li postoji ulaz u tabeli
 - za svako $k \in \{0, 1\}^{56}$ provjerava se da li je $D(k, C)$ u drugoj koloni.
 - ako jeste, onda je $E(k_i, M) = D(k, C) \Rightarrow (k_i, k) = (k_2, k_1)$

kreiranje i sortiranje
tabele

► Vrijeme = $2^{56} \log(2^{56}) + 2^{56} \log(2^{56}) < 2^{63}$
 $<< 2^{112}$

pretraga tabele

IDEA

- James Massey, Xuejia Lai
- ETH
- 1991
- patentiran je u mnogim državama
- dozvoljena je nekomercijalna upotreba
- radi sa 64-bitnim blokovima (otvorenog i kriptovanog teksta) i koristi 128-bitni ključevi
- ne koristi s-box-ove
- proces enkripcije je identičan procesu dekripcije

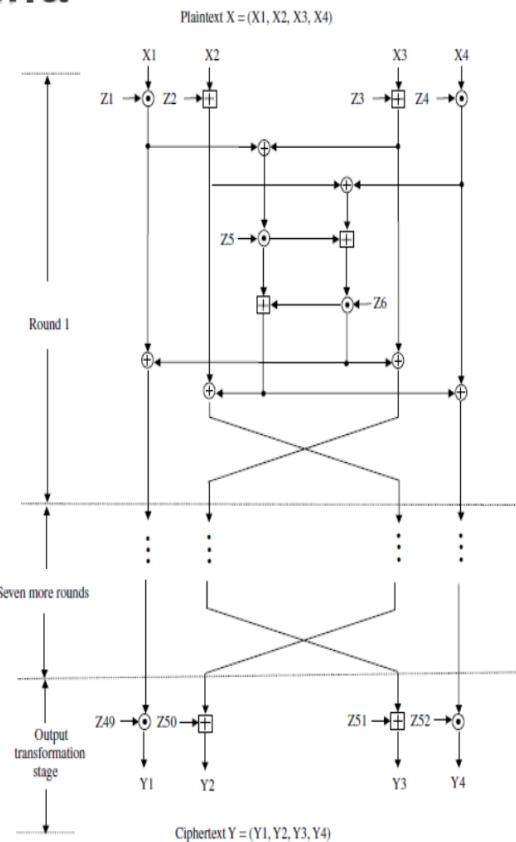
IDEA - generisanje ključeva

- 52 16-bitna podključa se generišu iz originalnog 128-bitnog ključa
 - 6 podključeva za svaku od 8 rundi + 4 podključa za devetu rundu (izlazna transformacija)
- prvo se 128-bitni ključ dijeli u 8 16-bitnih podključeva Z_1, Z_2, \dots, Z_8
- od prvih 8 podključeva 6 se koristi u prvoj rundi, dok se preostala dva koriste u drugoj rundi

- nakon toga, ključ se kružno šiftuje za 25 bita uljevo i ponovo dijeli u 8 podključeva
- ovo se ponavlja dok se ne generiše 52 podključa

IDEA - enkripcija

- 3 operacije:
 - bit-po-bit XOR 16-bitnih podblokova \oplus
 - sabiranje 16-bitnih integer-a po modulu 2^{16} \boxplus
 - množenje 16-bitnih integer-a po modulu $2^{16} + 1$ \odot



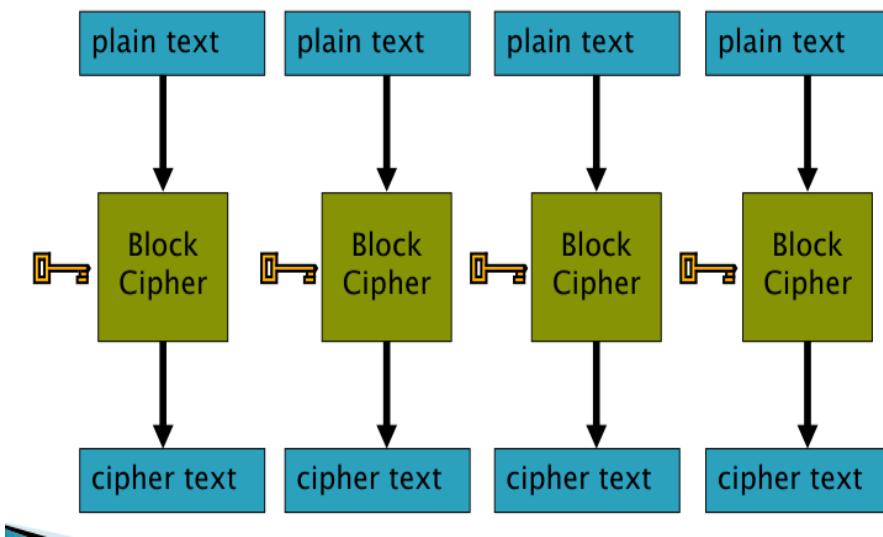
AES

- Advanced Encryption Standard
- FIPS-197 (Federal Information Processing Standards Publication 197)
- nasljednik DES-a
- blok-šifarski sistem
- 128-bitni blokovi
- dužina ključa: 128/192/256
- snaga \approx 3DES
- efikasnost veće od 3DES
- NIST objavio poziv 1997.g.
- 30 kandidata, 20 razmatrano ozbiljnije, 5 finalista (1999.g.)
- izabran Rijndael – 2000. godine
- dizajneri su se odrekli intelektualnih prava

- NIST standard od novembra 2001. godine
- US Federal Administration koristi AES kao government standard od 26.05.2002. godine za zaštitu osjetljivih informacija
- 2003. godine NSA odobrila da se AES-128 koristi za zaštitu SECRET informacija, a AES sa ključevima dužim od 128 bita za zaštitu TOP SECRET informacija
- promjenljiv broj rundi
 - 10 – u slučaju kada je dužina ključa 128 bita
 - 12 – u slučaju kada je dužina ključa 192 bita
 - 14 – u slučaju kada je dužina ključa 256 bita

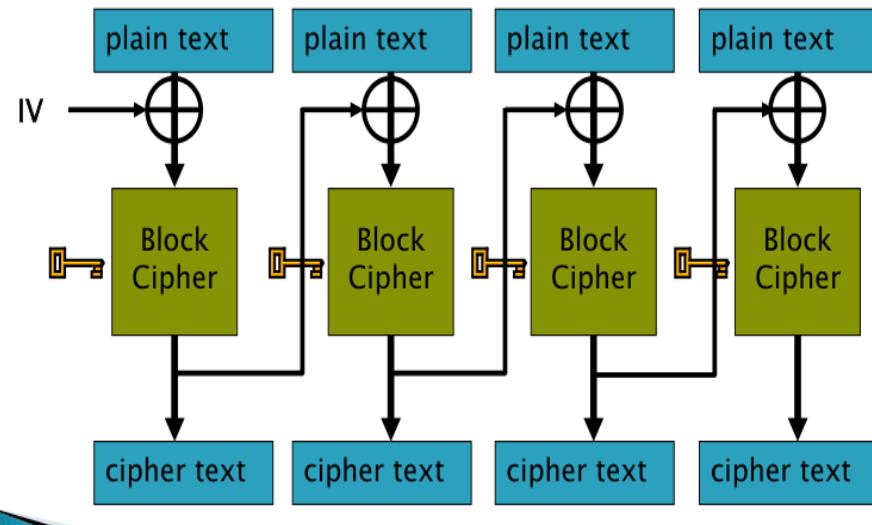
Modovi

ECB - Electronic codebook



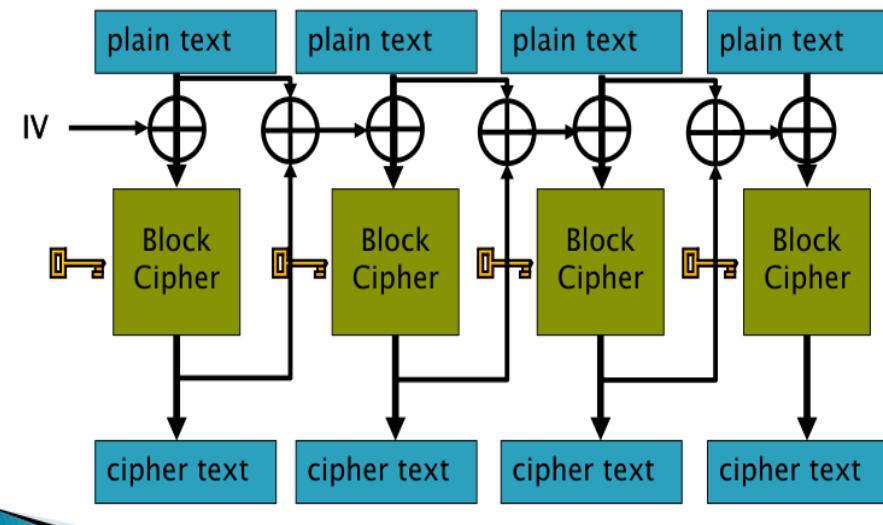
- **Nedostatak:**
 - identični plaintext blokovi kriptovani su u identične ciphertext blokove
- **Prednost:**
 - paralelna enkripcija
 - paralelna dekripcija
 - brzina
- često korišten u komercijalne svrhe

CBC - Cipher-block chaining



- **Nedostatak:**
 - enkripcija je sekvencijalna, ne postoji mogućnost paralelizacije
- **Prednost:**
 - paralelna dekripcija
 - često korišten

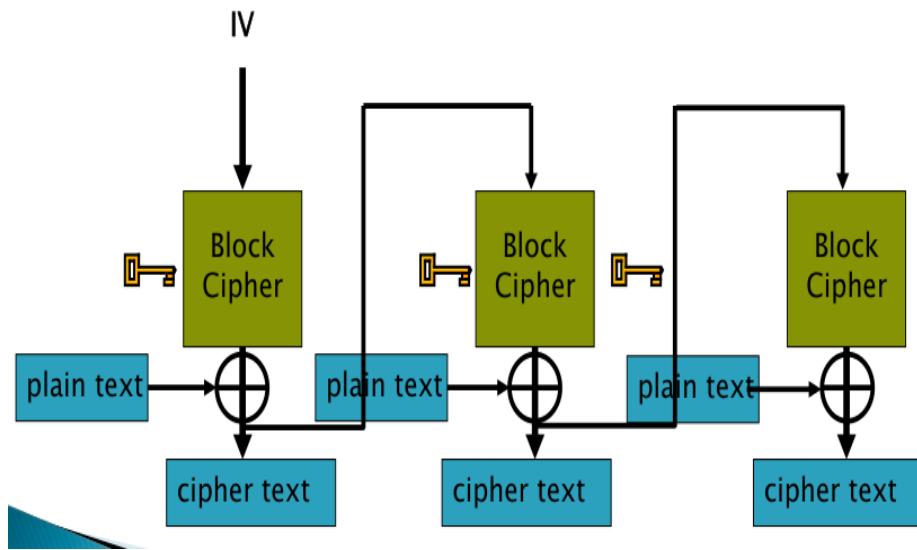
PCBC - Propagation Cipher-block chaining



- **Nedostatak:**
 - enkripcija je sekvencijalna, ne postoji mogućnost paralelizacije enkripcije
 - dekripcija je sekvencijalna, ne postoji mogućnost paralelizacije dekripcije
- **Prednost:**

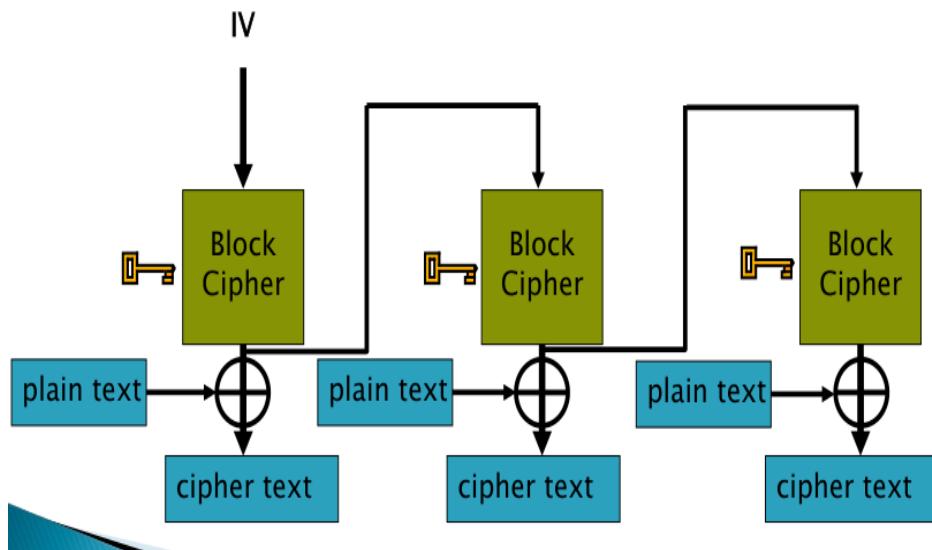
- često korišten

CFB – Cipher feedback/Cipher-block chaining



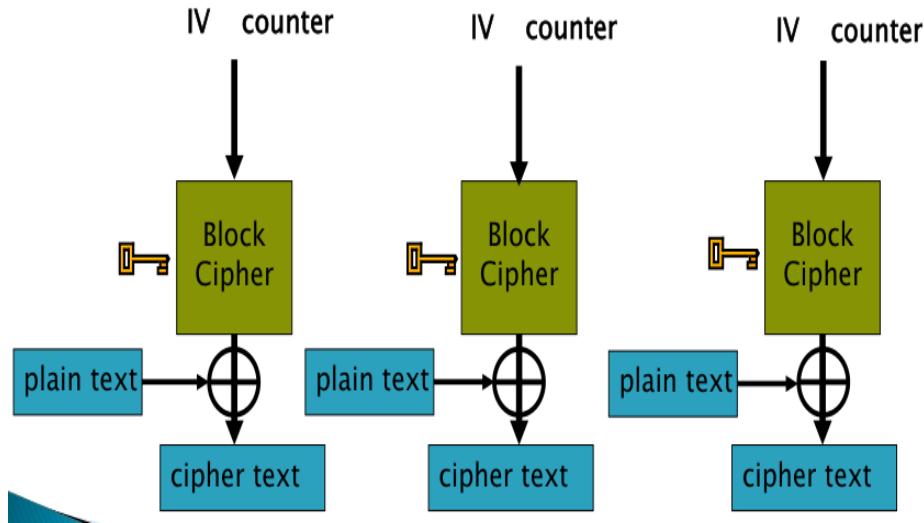
- **Nedostatak:**
 - ne postoji mogućnost paralelizacije procesa enkripcije
- **Prednost:**
 - paralelnja dekripcija

OFB – Output feedback



- **Nedostatak:**
 - ne postoji mogućnost paralelizacije procesa enkripcije
 - ne postoji mogućnost paralelizacije procesa dekripcije

CTR – Counter mode



- **Nedostatak:**

- postoji mogućnost paralelizacije procesa enkripcije
- postoji mogućnost paralelizacije procesa dekripcije dekripcije

Simetrični algoritmi

- bitska dužina ključa – NIST preporuke:
 - za period 2007-2010. godine minimalna bitska dužina ključa simetričnog algoritma iznosi 80
 - za period 2011-2030. godine, minimalna bitska dužina ključa simetričnog algoritma treba da iznosi 112
- Prema radu „Minimal Key Lengths for Symmetric Ciphers“ (autori Blaze, Diffie, Rivest, Schneier, Shimomura, Thompson, Weiner) iz 1996. godine:
 - nesigurnim se smatra korištenje ključeva simetričnih algoritama čija je bitska dužina manja ili jednaka 64

Lecture 7.

05 - Asimetrični šifrarski algoritmi

Simetrični šifrarski sistemi

- **Prednosti:**

- laka praktična implementacija, pogotovo hardverska, omogućava i veoma velike brzine prenosa
- ključevi su relativno kratki
- sistemi se mogu kombinovati da obezbijede jače šifre

- **Nedostaci:**

- obe strane moraju čuvati ključ u tajnosti
- za velike računarske mreže potreban je veliki broj ključeva
- praksa nalaže promjenu ključa za svaku sesiju

Asimetrični šifrarski sistemi

- Asimetrični algoritmi su dizajnirani tako da je ključ koji se koristi za kriptovanje različit od ključa koji se koristi za dekriptovanje
- Diffie i Hellman, 1976. godine
 - New directions in cryptography , IEEE Transactions on Information Theory
- Svaki korisnik posjeduje par ključeva - privatni i javni ključ:
 - postoji matematička korelacija između javnog i privatnog ključa
 - poruke šifrovane javnim ključem moguće je dešifrovati isključivo korišćenjem privatnog ključa
 - javni ključevi se javnim komunikacionim kanalima distribuiraju ka ostalim korisnicima
- Asimetrični algoritmi se označavaju kao kriptosistemi sa javnim ključem
- Primjer upotrebe. Korisnici A i B žele da razmjene povjerljive podatke:
 - korisnik A posjeduje javni ključ korisnika B (i obrnuto)
 - korisnik A šifruje poruku (otvoreni tekst) koristeći javni ključ korisnika B i šalje je preko standardnog komunikacionog kanala
 - korisnik B može da izvrši dešifrovanje primljene šifrovane poruke koristeći svoj privatni ključ
- Notacija
 - K_{PU_a} označava javni ključ korisnika A;
 - K_{PR_a} označava tajni ključ korisnika A;
 - E_{PU_a} označava vrijednost dobijenu šifrovanjem javnim ključem korisnika A;
 - E_{PR_a} označava vrijednost dobijenu šifrovanjem tajnim ključem korisnika A;
 - D_{PU_a} označava vrijednost dobijenu dešifrovanjem javnim ključem korisnika A;

- D_{PR_a} označava vrijednost dobijenu dešifrovanjem tajnim ključem korisnika A;
- Tako da je:
 - $D_{PR_a}(E_{PU_a}) = X$
 - $D_{PU_a}(E_{PR_a}) = X$
- Sigurnost asimetričnih algoritama zasniva se na postulatu da je računski nemoguće, u realnom vremenu, odrediti PRa na osnovu poznавања PUa (ili pronaći ekvivalentnu transformaciju)
- U sistemima sa javnim ključevima se pravi razlika u postupcima primjenom kojih se realizuje autentičnost i postupak za očuvanje tajnosti

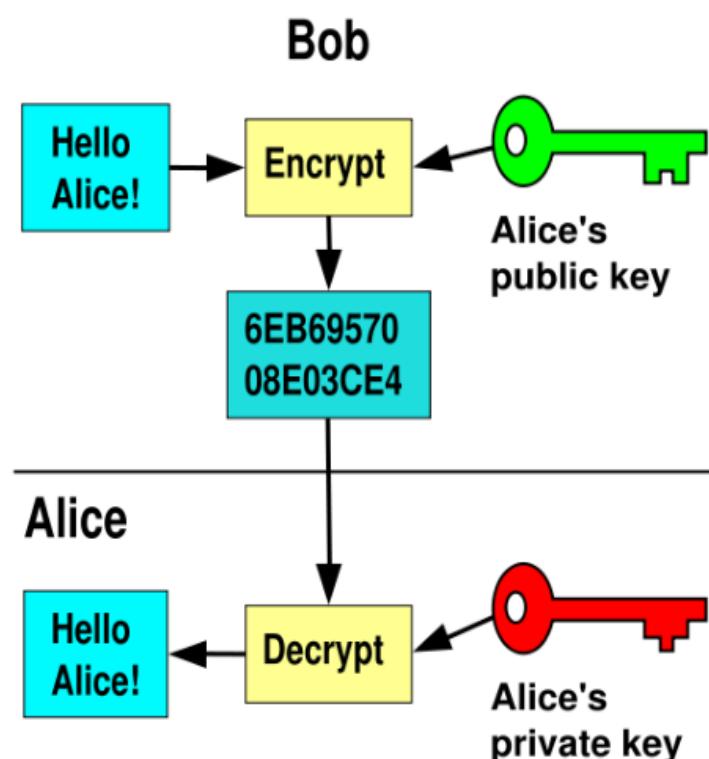
Kriptografija sa javnim ključem

- Da bi se očuvala tajnost korisnik A šalje šifrovani poruku ka korisniku B

$$Y = E_{PU_b}(X) \quad (7.1)$$

- Korisnik B dešifruje kriptografski zaštićenu poruku Y koristeći operaciju dešifrovanja primjenom svog privatnog ključa

$$D_{PR_b}(Y) = D_{PR_b}(E_{PU_b}(X)) = X \quad (7.2)$$



- Da bi se ostvarila autentičnost, korisnik A šalje ka korisniku B poruku

$$Y = E_{PU_a}(X) \quad (7.3)$$

- Korisnik B dešifruje primljenu poruku Y koristeći javni ključ korisnika A

$$D_{PR_a}(Y) = D_{PR_a}(E_{PU_a}(X)) = X \quad (7.4)$$

- Da bi se obezbijedila autentičnost i ostvarilo očuvanje tajnosti korisnik A šalje ka korisniku B poruku

$$Y = E_{PU_a}(X) \quad (7.5)$$

- Korisnik B dešifruje poruku Y koristeći operacije

$$D_{PU_a}(D_{PR_b}(Y)) = D_{PU_a}(D_{PR_b}(E_{PU_b}(E_{PR_a}(X)))) = D_{PU_a}(E_{PR_a}(X)) = X \quad (7.6)$$

- Zahtjevi kod sistema sa primjenom javnog ključa:

- Za legitimnog korisnika B je, u realnom vremenu, izvodljivo da:
 - * Kreira par ključeva (PUB, PRB)
 - * Poznavajući javni ključ korisnika A i poruku M, da generiše šifrovanu poruku $C = E_{PU_a}(M)$
 - * Dešifruje šifrovanu poruku C koristeći svoj privatni ključ i dobije izvornu poruku $M = D_{PR_b}(C) = D_{PR_b}(E_{PU_b}(M))$
- Za kriptoanalitičara (subjekta koji nadgleda komunikaciju) je računski nemoguće, u realnom vremenu, da:
 - * na osnovu poznavanja javnog ključa korisnika B otkrije njegov privatni ključ
 - * otkrije izvornu poruku M na osnovu poznavanja šifrovane poruke C i javnog ključa korisnika B

- Svako ko želi da prima poruku, mora da:

- „objavi“ svoj javni ključ
- posjeduje jedinstven privatni ključ

- Potrebno je obezbijediti autentičnost javnog ključa, tj. potrebno je obezbijediti mehanizam utvrđivanja vlasništva nad javnim ključem

- ako su poznati algoritam i javni ključ pomoću kojeg je moguće kreirati šifrat, kako nije moguće doći do otvorenog teksta ili privatnog ključa?
- jednosmjerna funkcija je funkcija čiji je rezultat lako izračunati, ali je teško uraditi reverzan postupak
 - primjer: jednostavno je pomnožiti dva prosta broja, ali je teško iz njihovog proizvoda odrediti proste brojeve
 - vjeruje se da je množenje prostih brojeva jednosmjerna funkcija

Kriptografija s javnim ključem

- ▶ računanje proizvoda dva prosta broja nije komplikovano
 - $23 \cdot 17 = 391$
- ▶ faktorizacija proizvoda dva velika prosta broja jeste komplikovana
 - 123 018 668 453 011 775 513 049 495 838 496
272 077 285 356 959 533 479 219 732 245 215
172 640 050 726 365 751 874 520 219 978 646
938 995 647 494 277 406 384 592 519 255 732
630 345 373 154 826 850 791 702 612 214 291
346 167 042 921 431 160 222 124 047 927 473
779 408 066 535 141 959 745 986 902 143 413

RSA

- Rivest, Shamir, Adelman algoritam (RSA)
 - jedan od najpoznatijih i najprimjenjivanih asimetričnih algoritama
 - razvijen 1977. godine na MIT-u
 - njegova sigurnost se bazira na činjenici da je izuzetno teško izvršiti faktorisanje velikih brojeva na proste faktore, tj. faktorizacija velikih prirodnih brojeva na proizvod dva prosta broja je izuzetno teška
- RSA algoritam je blok algoritam za šifrovanje u kojem su originalni podaci i šifrovani podaci cijeli brojevi između 0 i n-1, za neko n
- svaki korisnik generiše par ključeva (javni/privatni):
- slučajan izbor dva velika prosta broja **p** i **q**
- nakon toga, izračunava se njihov proizvod $n = p * q$
- (n) - broj prirodnih brojeva, ne većih od n, uzajamno prostih sa n

$$(n) = (p - 1)(q - 1) \quad (7.7)$$

- izbor ključa za enkripciju e
 - gdje je:

$$1 < e < (N), \gcd(e, (N)) = 1 \quad (7.8)$$

- izračunavanje ključa za dekripciju d

$$e * d \equiv 1 \pmod{(n)}, 1 < d < (N) \quad (7.9)$$

- javni ključ: **PU={e,n}**

- tajni ključ: **PR={d,n}**
- i pošiljalac i primalac poruke moraju da znaju vrijednost n
- pošiljalac mora da zna vrijednost e, i samo primalac zna vrijednost d
- Da šifruje poruku M, pošiljalac A, dobavlja javni ključ primaoca B **PU={e,n}** i izračunava:

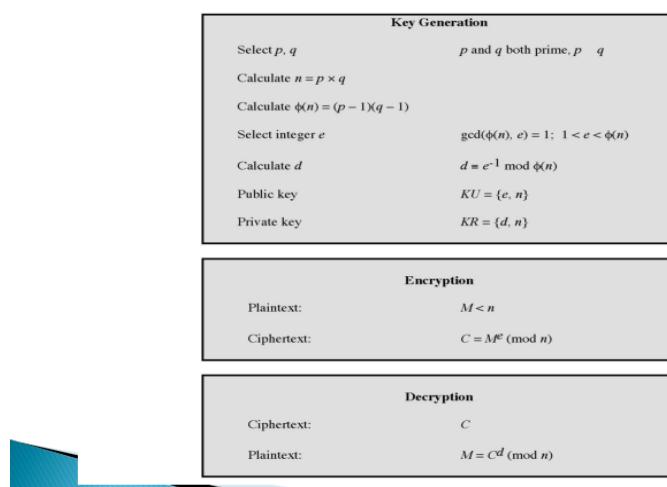
$$C = M^e \pmod{n}, \text{ gdje je } 1 < M < n \quad (7.10)$$

- Po prijemu šifrovanog teksta, primalac B dešifruje originalni tekst izračunavajući (koristeći privatni ključ **PR={d,n}**)

$$M = C^d \pmod{n} \quad (7.11)$$

- bitno: originalna poruka se šifruje u blokovima, takvim da svaki blok ima binarnu vrijednost manju od broja n

RSA



RSA – primjer

- ▶ enkripcija alfabeta
- ▶ plaintext se prvo enkoduje kao riječ u nekom alfabetu, npr. $\{0, 1, \dots, 9\}$
- ▶ onda se dijeli u blokove dužine $i-1$, gdje je $10^{i-1} < n < 10^i$
 - za alphabet od 10 karaktera
- ▶ svaki blok se nakon toga kriptuje, na ranije opisan način

RSA napadi

- Moguća su četiri napada na RSA:
 - pretraga svih ključeva (nezamislivo uvezši u obzir veličinu brojeva)
 - matematički napadi (zasnovani na težini izračunavanja $\phi(n)$, računanjem faktora n tj. p i q)
 - vremenski napadi (zasnovani na vremenu trajanja dešifrovanja)
 - napadi na izabranu šifru (zasnovani na strukturi algoritma) - chosen ciphertext attack

RSA - matematički napadi

- matematički napadi se javljaju u tri oblika:
 - nađu se faktori $n=pq$, pronađe se $\phi(n)$ i zatim d
 - nalaženje $\phi(n)$ direktno, bez određivanja p i q i zatim se nađe d
 - pronalazanje d direktno
- vjeruje se da su svi ekvivalentni faktorizaciji

RSA - vremenski napadi

- razvijeni su sredinom 90-ih godina
- koriste varijacije u trajanju izvršavanja operacija
- pretpostavlja se veličina operanda na osnovu potrošenog vremena
- kod RSA varijacije potrošenog vremena se javljaju prilikom eksponentizacije
- protiv mjere:
 - koristiti konstantno vrijeme eksponentizacije
 - dodavati slučajna zakašnjenja
 - prije eksponentizacije pomnožiti šifru sa slučajnom vrijednošću
- razvijeni su sredinom 90-ih godina
- koriste varijacije u trajanju izvršavanja operacija
- pretpostavlja se veličina operanda na osnovu potrošenog vremena
- kod RSA varijacije potrošenog vremena se javljaju prilikom eksponentizacije

RSA - Napadi na izabranu šifru

- napadač dolazi u posjed poruke

$$C = M^e \mod n \quad (7.12)$$

- napadač generiše novu poruku

$$C_2 = 2^e \mod n \quad (7.13)$$

- napadač kombinuje dvije poruke

$$C_x = C * C_2 = M^e * 2^e \mod n \quad (7.14)$$

- žrtva dekriptuje poruku kojoj napadač ima pristup

$$C_x^d = M^{ed} * 2^{ed} \mod n = M * 2 \mod n \quad (7.15)$$

sad može izračunati M

- varijante algoritma sa *padding scheme-ama* nisu ranjive na ovaj napad

NIST preporuke

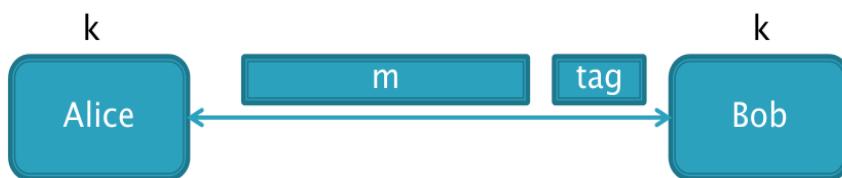
period	algoritam - dužina ključa	
	simetrični	asimetrični
	80	1024
2020 - 2030	112	2048
2019 - 2030+	128	3072
2019 - 2030+	192	7680
2019 - 2030+	256	15360

05 - Integritet poruka

Integritet poruke

- integritet
- tajnost
- primjer:
 - fajlovi na disku ne moraju biti tajni, ali mora biti očuvan njihov integritet (sprječavanje izmjene od strane virusa i sl.)
- MAC

MAC - Message Authentication Code



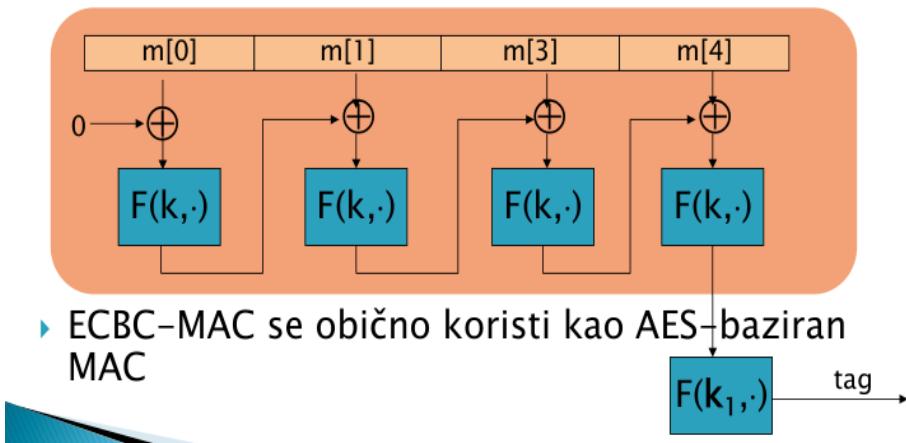
- generisanje taga (S-signing algorithm):
 - $S(k, m) \rightarrow \text{tag}$ - tag je kratak u poređenju s porukom
- verifikacija taga (V-verification algorithm):
 - $V(k, m, \text{tag}) \rightarrow \text{"true"} \text{ ili } \text{"false"}$, 0 ili 1
- MAC je par funkcija (S,V) definisanih nad prostorom ključa, prostorom poruka i prostorom tagova (K,M,T) gdje:
 - $S(k, m)$ daje t iz T
 - $V(k, m, t)$ daje „true“ ili „false“
- mora da važi: $V(k, m, S(k,m)) = \text{"true"}$
- preporuke za bitsku dužinu tag-a – kao i kod simetričnih algoritama
- zahtjeva postojanje tajnog ključa
- primjer – CRC – nema tajnog ključa
 - napadač lako može izmijeniti poruku i izračunati novi CRC – primalac poruke to neće detektovati
- CRC – namijenjen za detekciju slučajne greške, ne i za detekciju malicioznih izmjena

Kad je MAC siguran?

- da bi se mogao smatrati sigurnim mora biti otporan na *chosen plaintext attack*
- Napadač: **chosen message attack (chosen plaintext attack)**
 - za m_1, m_2, \dots, m_q napadač dobija $t_i \Leftarrow S(k, m_i)$
- Cilj napadača:
 - proizvesti novi validan par poruka/tag (m, t) , gdje je: $(m, t) \notin \{(m_1, t_1), \dots, (m_q, t_q)\}$
- MAC se može smatrati sigurnim ako:
 - napadač nije u stanju da kreira novi validan par poruka/tag, čak i za potpuno slučajnu poruku (koja nema konkretno značenje)
 - za dati par (m, t) ne može generisati (m, t') takvo da je $t \neq t'$
- Primjer nesigurnog MAC-a:
 - koristimo PRF koja na izlazu daje $\{0, 1\}^{10}$ – 10 bita
 - napadač sa vjerovatnoćom od $1/2^{10}$, tj. $1/1024$ može pogoditi tag – veoma jednostavno – nesigurno...
- Zaključak – potrebna nam je veća bitska dužina na izlazu

ECBC - encrypted CBC

- poruke proizvoljne dužine dijele se u blokove
- dva ključa k_1, k_2

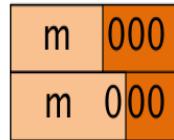


- enkripcija ključem k_1 – obavezna
- bez nje – MAC nije siguran - rawCBC(k, m)
- Napadač bira poruku m dužine jednog bloka
- zahtijeva tag $\rightarrow t = F(m, k)$
- dobijeni tag t može iskoristiti za poruku dužine 2 bloka $(m, t \oplus m)$
- dokaz: $rawCBC(k, (m, t \oplus m)) = F(k, F(k, m) \oplus (t \oplus m)) = F(k, t \oplus (t \oplus m)) = t$
- dvije različite poruke imaju isti t

MAC padding

Šta uraditi kada je posljednji blok poruke kraći od očekivanog?

- dodavanje nekoliko 0 nije opcija
 - kako razlikovati sljedeće dvije poruke?



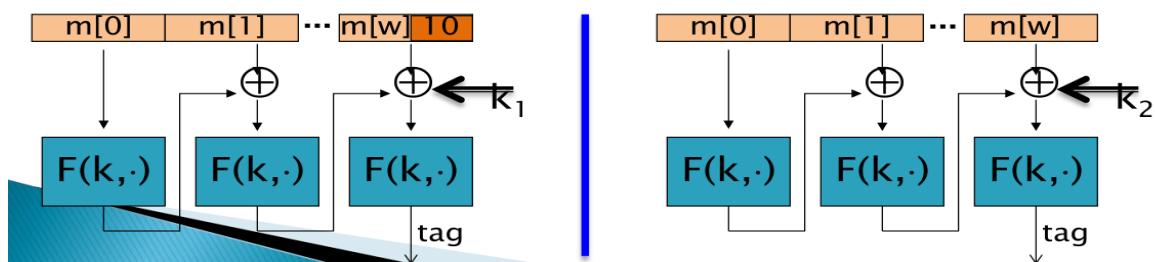
- ISO: na kraj poruke dodati "10...0"
- Dodati dummy blok ako je to potrebno
 - u slučaju kada originalna poruka može biti izdijeljena u blokove (bez viška bita, bez ostatka)
 - * problem ako se ovo ne uradi!



- "1" označava početak pad-a

CMAC

- NIST standard
- Varijanta CBC-MAC-a, gdje je $key = (k, k_1, k_2)$
 - k_1, k_2 se izračunavaju iz k
- Razlike u odnosu na CBC-MAC:
 - nema dodatnog enkripcionskog koraka
 - nema dummy bloka



Hash funkcije

- hash funkcija: $h = H(x)$
- osobine:
 - kompresija: large input domain \rightarrow small fixed output
 - dobra distribuiranost: vjerovatnoća P da je $P(H(x) = i) \sim 1/N$, gdje je N veličina hash tabele
- kriptografska hash funkcija – dodatni zahtjevi:
 - pre-image resistance (“one-way”) – za svako $h = H(x)$, teško je pronaći x
 - weak collision resistance (2nd pre-image resistance): za svako $h = H(x)$, teško je pronaći bilo koje x' , takvo da je $H(x') = h$
 - strong collision resistance: teško je pronaći par x i y (dvije proizvoljne vrijednosti), takav da je $H(x) = H(y)$
- Hash funkcija predstavlja determinističku matematičku proceduru koja uzima podatak proizvoljne dužine i vraća niz fiksne dužine koji je (u idealnom slučaju) jedinstven za bilo koju kombinaciju na ulazu
- Podatak dobijen hash funkcijom se koristi kao “digitalni otisak prsta”, tj. izlazna vrijednost hash funkcije se često naziva kriptografski otisak
- Ovakav nivo determinizma pruža mogućnost upotrebe hash funkcija u gotovo svim sigurnosnim aspektima modernih računarskih sistema
- Razvoj računara i povećanje resursa dostupnih prosječnom korisniku doveli su do potrebe za modernizacijom algoritama koji se koriste za dobijanje hash funkcija, pa je korisnicima danas na raspolaganju veliki broj hash kriptografskih funkcija različite kompleksnosti
- Razvoj hash funkcija počinje krajem 80-tih godina prošlog vijeka
- Ronald Rivest, razvio nekoliko najpoznatijih algoritama, među kojima su MD2, MD4 i MD5
- Neki od ovih algoritama su u današnjim uslovima već prevaziđeni (postoji veliki broj radova koji otkrivaju slabosti nekih starijih algoritama), dok se neki i danas aktivno koriste u velikom broju računarskih sistema (npr. MD5)
- Hash funkcije se u modernim računarskim sistemima najčešće koriste u svrhu obezbjeđivanja integriteta podataka i autentičnosti
- Nizom (pojedinačno) jednostavnih matematičkih operacija od ulaznog niza proizvoljne dužine (koja može biti jednaka i 0) dobija se izlazna vrijednost koja predstavlja jedinstven otisak datog ulaza i time potvrđuje njegov sadržaj
- Svaka hash funkcija mora ispunjavati sljedeće zahtjeve:
 - izračunavanje hash vrijednosti je jednostavna i brza operacija
 - nije moguće iz otiska rekonstruisati poruku na osnovu koje je otisak generisan
 - nije moguće mijenjati originalnu poruku bez promjene hash-a (u praksi promjena jednog bita ulaznog niza rezultuje promjenom većeg dijela otiska, u nekim slučajevima i preko 50%)
 - nije moguće naći dvije poruke koje daju istu hash vrijednost (otpornost na koliziju)

- U većini slučajeva nije moguće doći do dokaza da je neka hash funkcija irreverzibilna, ali je dovoljno dokazati da je za izračunavanje inverzne vrijednosti potrebno dovoljno veliko vrijeme izvršavanja pa da postupak bude neprimjenjiv u praksi
 - Neke od osnovnih karakteristika *hash* funkcija su:
 - dužina otiska,
 - broj potrebnih operacija/koraka za izvršavanje i
 - maksimalna dužina ulaznog stringa – iako ovaj argument najčešće ne predstavlja uslov funkcionalnosti, ipak postoji ograničenje. Kod SHA-384 i novijih algoritama taj broj iznosi $2^{128} - 1$, dok kod starijih algoritama on iznosi $2^{64} - 1$
 - Dužina otiska direktno utiče na sigurnost.
 - MD2, MD4 i MD5 algoritmi imaju otisak dužine 128 bita
 - SHA-1 - 160 bita
 - SHA-2 grupa algoritama - broj iz naziva algoritma označava i dužinu otiska. Tako za SHA-224 imamo dužinu od 224 bita, za SHA-384 384 bita itd.

Napadi na osobine hash funkcije

- brute force napadi
 - brute force pre-image napad:
 - * $2^n / 2 = 2^n - 1$ pokušaja
 - brute force 2^{nd} pre-image napad:
 - * $2^n / 2 = 2^n - 1$ pokušaja
 - brute force collision napad:
 - * paradoks rođendana - koliko osoba mora biti u sobi da bi vjerovatnoća da su bilo koje od njih rođene istog dana bila veća od 50%? - 23
 - * kolizija će se desiti u približno: $1.2 * \sqrt{2^m} = 1.2 * 2^{m/2}$
- Primjer:
 - računar koji hešira milion poruka u sekundi
 - 64-bitna heš funkcija
- brute force 2nd pre-image napad:
 - potrebno je oko 600.000 godina za uspješno izvršavanje brute force 2nd pre-image napada
- kolizija
 - potrebno je oko 85 minuta ($1.2 * 2^{32}$ pokušaja).
- zaključak: kompleksnost rođendanskog napada (birthday attack) jednaka je korijenu kompleksnosti regularnog brute force napada
- Iz tog razloga potrebno je udvostručiti broj bita da bi se dobila zahtijevana sigurnost
 - ako je današnjim sigurnosnim preporukama za simetrične algoritme definisana minimalna bitska dužina ključa od 112 bita, onda ekvivalentna sigurnost kod heš funkcija može da se postigne heš funkcijom koja na izlazu daje 224-bitnu vrijednost

- sve heš funkcije su izložene *brute force* napadima, bez obzira na njihovu strukturu
- zato je bitno da bitska dužina otiska bude dovoljno duga
- postoje i druge vrste napada:
 - kriptoanalitički napadi
 - * specifični su za pojedine heš funkcije
 - *side channel* napadi
 - * eksploatišu propuste u implementaciji algoritma

- heš funkcija se smatra „razbijenom“, ako postoji napad kojim se može kreirati pre-image, 2^{nd} pre-image ili kolizija sa manjom kompleksnošću nego što je kompleksnost *brute force* napada
- npr. ako je heš funkcija 160-bitna, a ako kolizija može da se nađe u 2^{77} pokušaja, onda se smatra „razbijenom“
 - ovo važi i u slučaju kada je taj broj pokušaja dovoljno veliki da se napad praktično ne može izvesti u praksi

MD4

- MD4 – algoritam nastao 1990. godine
- tvorac algoritma je Ronald Rivest
- zastario i ustupio mjesto novijim, sigurnijim algoritmima
- njegov značaj u istoriji razvoja kriptoalgoritama se i danas ne može zanemariti
- prvi uspješni napadi su objavljeni sredinom devedesetih godina prošlog vijeka, a danas je vrijeme potrebno za generisanje kolizije svedeno na svega nekoliko milisekundi

- na ulazu se nalazi string dužine b bita
- potrebno je kreirati izlazni string dužine 128 bita, odnosno 4 32-bitne riječi
- prvi korak je popunjavanje prostora (padding) dok se ne dobije ulazni string čija je dužina za 64 bita kraća od n^*512 bita
- padding se dodaje uvijek, čak i kada ulazni string već ima odgovarajuću dužinu
 - na kraju stringa se dodaje bit čija je vrijednost 1, za kojim slijedi proizvoljan broj bita vrijednosti 0, dok se ne postigne odgovarajuća dužina. Na kraj ulaznog stringa se ovim postupkom mora dodati najmanje 1, a najviše 512 bita.

- 64-bitna reprezentacija broja b (dužine ulaza) se dodaje na kraj rezultata dobijenog prethodnim korakom
- u slučaju da je $b > 2^{64}$, dodaju se samo donja 64 bita. Sada je dužina kompletne poruke jednaka n^*512 bita, odnosno n^*16 (32-bitnih) riječi
- $M[0 \dots N-1]$ - skup 32-bitnih riječi od kojih je poruka sastavljena, gdje je N umnožak broja 16
- matrica stanja koja se koristi pri računanju otiska – 4 32-bitne riječi
- riječi su označene sa A, B, C, D i prikazuju se heksadecimalnim ciframa

- definisane su tri pomoćne funkcije koje na ulazu uzimaju 3, a na izlazu daju 1 32-bitnu riječ
- Prvi postupci za kreiranje kolizije na MD4 algoritmu su objavljeni 2004. godine
- Do danas je taj postupak značajno usavršen, i na Internetu postoje besplatne softverske implementacije generatora kolizije za MD4 algoritam
- Preporučuje se da se, što je moguće prije, MD4 izbaci iz upotrebe i da se u svim aplikacijama koje ga koriste zamjeni nekim od modernijih algoritama

MD5

- MD5 (Message Digest 5) je jedan od najčešće korištenih hash algoritama u računarstvu
- Razvio ga je 1991. godine Ronald Rivest kao odgovor na probleme i nedostatke koji su otkriveni kod prethodnog, MD4 algoritma
- MD5 se koristi u velikom broju aplikacija. U dokumentu kojim se opisuje MD5 (RFC 1321) navodi se da je ovaj algoritam nešto sporiji od svog prethodnika, što se nadoknađuje povećanom sigurnošću
- MD5 proizvodi izlaz dužine 128 bita (4 32-bitne riječi), isto kao MD4 algoritam
- na ulazu se nalazi string dužine b bita
- potrebno je kreirati izlazni string dužine 128 bita, odnosno 4 32-bitne riječi
- prvi korak je popunjavanje prostora (padding) dok se ne dobije ulazni string čija je dužina za 64 bita kraća od n^*512 bita
- padding se dodaje uvijek, čak i kada ulazni string već ima odgovarajuću dužinu
 - na kraju stringa se dodaje bit čija je vrijednost 1, za kojim slijedi proizvoljan broj bita vrijednosti 0, dok se ne postigne odgovarajuća dužina. Na kraj ulaznog stringa se ovim postupkom mora dodati najmanje 1, a najviše 512 bita.
- 64-bitna reprezentacija broja b (dužine ulaza) se dodaje na kraj rezultata dobijenog prethodnim korakom
- sada je dužina kompletne poruke jednaka n^*512 bita, odnosno k^*16 (32-bitnih) riječi
- matrica stanja koja se koristi pri računanju otiska – 4 32-bitne riječi
- definisane su četiri pomoćne funkcije koje na ulazu uzimaju 3, a na izlazu daju 1 32-bitnu riječ
- Sve definisane funkcije imaju osobinu da, ako su biti ulaznih riječi nekorelisani, tada će i izlazne vrijednosti biti nekorelisane
- U ovom koraku se definiše i tabela od 64 elementa označena sa $T[1\dots64]$, čije se vrijednosti dobijaju primjenom sinusne funkcije. Element na i-toj poziciji se izračunava prema sljedećoj formuli:

$$T[i] = [4294967296 * \sin(i)] \quad (8.1)$$

- gdje je uglastim zagradama označena cjelobrojna vrijednost dobijenog rezultata

- Prvi algoritmi za kreiranje kolizija nad MD5 algoritmom su objavljeni 2005. godine
- ako je za koliziju potrebno nešto više vremena nego što je to bio slučaj sa MD4, preporučuje se da se MD5 izbaci iz upotrebe iz svih sistema u kojima se koristi, te da se unaprijedi nekim od modernijih algoritama iste namjene
- 2005. godine kreirani su parovi PostScript dokumenata i X.509 sertifikata sa istim hash-om

MD4 - MD5 razlike

- uvedena je još jedna pomoćna funkcija ($I(X,Y,Z)$)
- za razliku od MD4 algoritma, gdje se u svakom koraku dodavala ista konstanta, MD5 algoritam za svaki korak daje drugu konstantu koja zavisi od trenutne iteracije, čime se dodatno povećava sigurnost
- funkcija G se razlikuje od funkcije koju koristi MD4
- vrijednosti za koje se vrši rotiranje vrijednosti u svakom koraku (označena sa s) su različite za svaku operaciju unutar jedne iteracije
- MD4 ima 3 runde, a MD5 ima 4 runde

SHA-1

- SHA-1 je najčešće korišten algoritam iz SHA familije algoritama
- korišten u:
 - OpenPGP, S/MIME, IPSec, SSH itd.
- SHA-0 (povučen iz upotrebe brzo nakon objavljivanja od strane Američke agencije za bezbjednost (NSA), zbog sigurnosnih nedostataka)
- SHA-2 grupa algoritama (svi algoritmi iz ove grupe koriste isti postupak za generisanje otiska, uz promjenljivu dužinu izlaznog niza)
- SHA-3 – KECCAK algoritam odabran na otvorenom konkursu 02.10.2012. godine
 - Nije namijenjen za zamjenu SHA-2, jer ne postoji poznat napad na SHA-2
- SHA-2 pretvara ulaznu poruku maksimalne dužine $2^{64} - 1$ ($2^{128} - 1$ bita za SHA-384 i SHA-512) bita u izlazni string dužine 160 bita
- po svom dizajnu, SHA-1 je sličan Message Digest algoritmima

SHA-1

Algoritam	Poruka	Blok	Riječ	Dužina
SHA-1	$<2^{64}$	512	32	160
SHA-224	$<2^{64}$	512	32	224
SHA-256	$<2^{64}$	512	32	256
SHA-384	$<2^{128}$	1024	64	384
SHA-512	$<2^{128}$	1024	64	512

- Prvi korak u implementaciji SHA-1 je padding (dopuna) koja se vrši na identičan način kao i kod MD algoritama
- Ulazni string se dopunjava jednim bitom vrijednosti 1, n bita vrijednosti 0 i 64-bitnom reprezentacijom dužine niza prije dopune, tako da ukupna vrijednost ovako dobijenog stringa bude $n \cdot 512$ bita
- SHA-1 dalje obrađuje niz u blokovima od po 512 bita

- U sljedećem koraku definišemo 80 logičkih funkcija ($f(0), f(1), \dots, f(79)$) koje će se koristiti u izračunavanju međuvrijednosti
- Funkcije koriste 3 32-bitne riječi, označene sa B, C i D i, u zavisnosti od indeksa t ($0 \leq t \leq 79$) definišu se na sljedeći način:

- ▶ Definišimo se i niz konstanti ($k(0 \dots 79)$), u zavisnosti od istog indeksa:

$$\begin{array}{ll} K(t) = 5A827999 & (0 \leq t \leq 19) \\ K(t) = 6ED9EBA1 & (20 \leq t \leq 39) \\ K(t) = 8F1BBCDC & (40 \leq t \leq 59) \\ K(t) = CA62C1D6 & (60 \leq t \leq 79) \end{array}$$

- ▶ Ovako pripremljeni podaci se koriste za generisanje SHA-1 otiska
- ▶ Postoji nekoliko potpuno ravnopravnih metoda u implementaciji algoritma koji daju identične rezultate

SHA-1 -implementacija 1

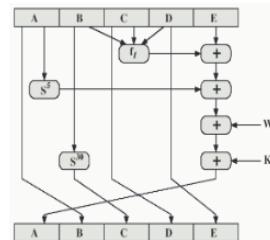
- ## **SHA-1 -implementacija 1**
- ▶ Prva implementacija koristi dvije matrice stanja dužine 5 32-bitnih riječi, kao i sekvencu od 80 32-bitnih riječi označenih sa $W(0), W(1), \dots, W(79)$
 - ▶ Riječi iz matrice stanja su označene sa A, B, C, D, E i H_0, H_1, H_2, H_3, H_4 , respektivno. Osim toga, koristi se i pomoćna varijabla dužine 1 riječi (32 bita)
 - ▶ Procesiranje bilo kojeg od 512-bitnih blokova ulaznog stringa se odvija u 80 koraka
 - ▶ Prije procesiranja, matrice stanja se inicijalizuju na sljedeće vrijednosti:

$$\begin{aligned} H_0 &= 67452301 \\ H_1 &= EFCDAB89 \\ H_2 &= 98BADCCE \\ H_3 &= 10325476 \\ H_4 &= C3D2E1F0 \end{aligned}$$

SHA-1 -implementacija 1

- ▶ Nakon toga se procesiraju blokovi $M(1), M(2), \dots M(n)$. Da bi se blok sa indeksom i obradio, mora proći kroz sljedeći postupak:
 - $M(i)$ se podijeli u 16 32-bitnih riječi, označenih sa $W(0), W(1), \dots W(15)$, gdje se $W(0)$ nalazi na krajnjoj lijevoj strani ulaznog stringa
 - za svako t u intervalu od 16 do 69 izvršimo sljedeću operaciju:

$$W(t) = S<<<1(W(t-3) \text{ XOR } W(t-8) \text{ XOR } W(t-14) \text{ XOR } W(t-16)),$$
 gdje operator $<<<$ predstavlja rotiranje u lijevu stranu.
 - postavimo $A=H_0, B=H_1, C=H_2, D=H_3, E=H_4$
- ▶ za svako t u intervalu od 0 do 79 izvršimo sljedeće operacije:
 - $\text{TEMP} = S<<<5(A) + f(t;B,C,D) + E + W(t) + K(t);$
 - $E = D;$
 - $D = C;$
 - $C = S<<<30(B);$
 - $B = A;$
 - $A = \text{TEMP};$



SHA-1 -implementacija 1

- ▶ postavimo
 - $H_0 = H_0 + A$
 - $H_1 = H_1 + B$
 - $H_2 = H_2 + C$
 - $H_3 = H_3 + D$
 - $H_4 = H_4 + E$
- ▶ otisak predstavlja sljedeći skup 32-bitnih riječi: $H_0\ H_1\ H_2\ H_3\ H_4$

SHA-2

- SHA-2 je zajedničko ime za nekoliko hash algoritama koji se međusobno razlikuju samo u dužini otiska koji proizvode
- postupak dobijanja otiska je identičan
- specifikacija objavljena 2001. godine i uključuje sljedeće algoritme: SHA-224, SHA-256, SHA-384 i SHA-512

- broj u nazivu označava dužinu otiska u bitima
- svi SHA-2 algoritmi su slični SHA-1 algoritmima
- još nisu objavljeni značajni napadi koji bi ozbiljno kompromitovali sigurnost SHA-2 algoritama

SHA-256 i SHA-224

- SHA-256
 - maksimalna dužina poruke je $2^{64} - 1$ bita
 - u postupku se koristi 8 32-bitnih varijabli (označenih sa A, B, C, D, E, F, G, H), dok sekvenca W ima 64 člana
 - koriste se 2 pomoćne riječi TEMP1 i TEMP2
 - izlazna vrijednost ima dužinu 256 bita (8 32-bitnih riječi)
- SHA-224
 - po implementaciji je identičan prethodnom, ali se od rezultata uzima samo prvih 224 bita, počevši sa lijeve strane

SHA-512 i SHA-384

- SHA-512
 - maksimalna dužina poruke je $2^{128} - 1$ bita
 - dužina bloka koji se obrađuje je 1024 bita,
 - koriste se 2 pomoćne riječi TEMP1 i TEMP2
 - u postupku se koristi 8 64-bitnih varijabli (označenih sa A, B, C, D, E, F, G, H), dok sekvenca W ima 80 članova, kao i SHA-1
 - koriste se 2 pomoćne riječi TEMP1 i TEMP2
 - izlazna vrijednost ima dužinu od 512 bita (8 64-bitnih riječi)
- SHA-384
 - postupak generisanje je identičan postupku za SHA-512, ali se od izlaznog stringa uzima samo prvih 384 bita sa lijeve strane, kako bi otisak bio odgovarajući za ovaj algoritam

Primjena hash funkcija

- zaštita korisničkih podataka (lozinke)
- digitalni potpis
- integritet podataka – provjera integriteta poruka i dokumenata prenesenih preko mreže
- anonimizacija podataka
- za kreiranje HMAC (*Hash Message Authentication Code*)

Primjer - čuvanje lozinke

- većina savremenih aplikacija koristi *username/password* kombinaciju za autentikaciju korisnika
- korisnici često koriste istu *username/password* kombinaciju za različite aplikacije
- često se lozinke čuvaju nekriptovane, tj. u plaintext obliku
 - – potencijalno ugrožene od DB administratora, SQL Injection napada, itd.
- backup je, takođe, potencijalno ugrožen
- da bi se riješio ovaj problem, lozinke se moraju čuvati u kriptovanom obliku – dva načina:
 - primjena hash funkcija
 - primjena kriptografskih algoritama (DES, AES, ...) – ovo generalno nije dobar način za čuvanje kredencijala – lozinke su tajne i nema razloga da se pod bilo kojim uslovima vrši njihova dekripcija
- načini za dolazak u posjed hash-irane lozinke su:
 - *brute force* napad (računanje hash vrijednosti svih mogućih lozinki)
 - *dictionary* napad (računanje hash vrijednosti često korištenih lozinki)
- primjena hash funkcija nad čuvanim lozinkama – identične lozinke imaju identičan hash
- 2 nedostatka primjene hash funkcija nad čuvanim lozinkama:
 - paradoks rođendana
 - *rainbow* tabela – unaprijed izračunate hash vrijednosti mogu se iskoristiti za brzo otkrivanje lozinki
- rješenje: **salt**
 - *salt* je slučajan broj fiksne dužine
 - *salt* mora biti različit za svaki čuvani podatak – lozinku
 - mora se čuvati u *plaintext*-u, uz *hash*-iranu lozinku
 - u ovom slučaju, napadač mora vršiti *brute force* napad na svaku pojedinačnu lozinku
 - na ovaj način postiže se otpornost na pomenuta 2 nedostatka
 - RSA PKCS#5 standard (Password-Based Cryptography Standard) preporučuje 64-bitnu dužinu salt-a:
 - u ovom slučaju postoji 2^{64} mogućih salt vrijednosti za svaku lozinku
 - PKCS #5: Password-Based Cryptography Specification Version 2.1
- hardening:
 - usporavanje izračunavanja – iteracija *hash* operacije n puta

Primjer – čuvanje lozinki

- ▶ *hash*-iranje lozinke n puta usporava izračunavanje *hash* vrijednosti i za legitimnog korisnika i za napadača
 - legitimni korisnici ovo ne primjećuju – vrijeme *hash*-iranja predstavlja mali procenat vremena u interakciji legitimnog korisnika sa sistemom
 - napadač troši skoro 100% vremena na *hash*-iranje, tako da se na ovaj način napadač usporava, i to faktorom n
- ▶ RSA PKCS5 standard preporučuje minimalno 1000 operacija
- ▶ čuvana lozinka je:
$$\text{hash}(\text{hash}(\text{hash}(\text{hash}(\text{.....hash(password||salt)))))))))))$$
- ▶ za autentikaciju korisnika gornja operacija takođe mora biti izvršena, nakon čega slijedi poređenje *hash*-eva

Primjer – HMAC

- ▶ $\text{HMAC } (\text{K}, \text{m}) = \text{H } ((\text{K} \oplus \text{opad}) \parallel \text{H } ((\text{K} \oplus \text{ipad}) \parallel \text{m}))$ gdje je
 - H – hash funkcija
 - K – tajni ključ *padded to the right* nulama do veličine ulaznog bloka hash funkcije ili hash originalnog ključa – ako je duži od veličine bloka
 - m – poruka
 - \parallel – konkatenacija
 - \oplus XOR
 - opad – *outer padding* (0x5c5c5c...5c5c, konstanta dužine jednog bloka – heksadecimalna
 - ipad – *inner padding* (0x363636...3636, konstanta dužine jednog bloka – heksadecimalna)
- ▶ Primjeri:
 - HMAC_MD5, HMAC_SHA1, HMAC_SHA256,...

Lecture 9.

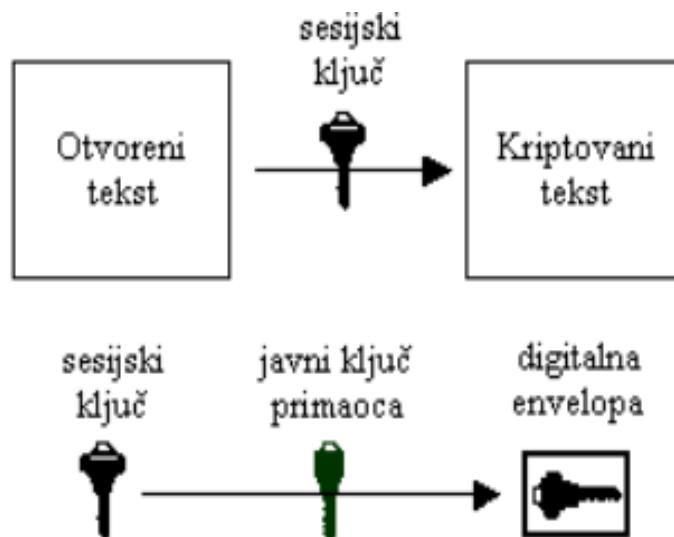
06 - Razmjena ključeva

Digitalna envelopa

- ozbiljan nedostatak asimetričnih šifarskih sistema jeste brzina, koja je za više redova veličine manja nego kod simetričnih šifarskih sistema
- ovaj nedostatak čini asimetrične šifarske sisteme neadekvatnim za kriptovanje velikih dokumenata
- simetrični algoritmi su brži i adekvatni su za kriptovanje velikih dokumenata
- rješenje ovog problema jeste kombinacija simetričnih i asimetričnih šifarskih sistema

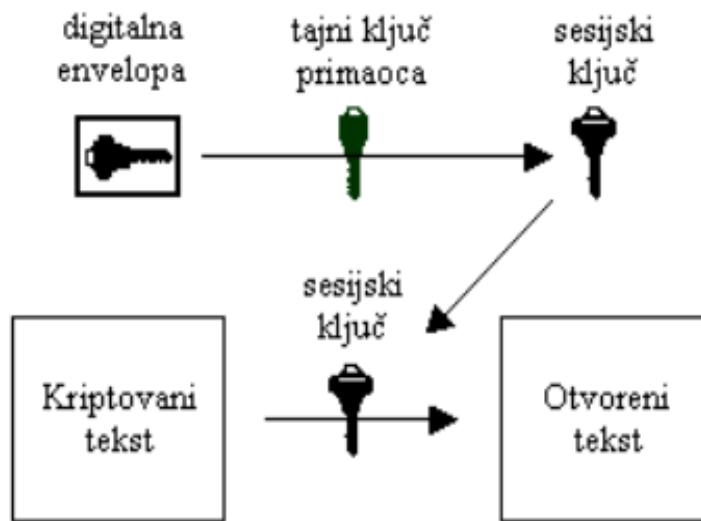
- poruka je u toku prenosa zaštićena, jer je kriptovana simetričnim sesijskim ključem koji je poznat samo pošiljaocu, jer ga on kreirao
- sesijski ključ je zaštićen u toku prenosa, jer je kriptovan javnim ključem primaoca, tako da ga samo primalac može dekriptovati
- digitalna envelopa se može koristiti za uspostavljanje sigurne dvosmjerne komunikacije, tj. za razmjenu tajnog (sesijskog) ključa simetričnog algoritma

- osnovni koraci prilikom slanja poruke su sljedeći:
- pošiljalac generiše slučajni tajni ključ (sesijski ključ)
- pošiljalac kriptuje poruku simetričnim algoritmom korišćenjem generisanog sesijskog ključa



- pošiljalac kriptuje sesijski ključ javnim ključem primaoca i na ovaj način kreira **digitalnu envelopu**

- šalje kriptovanu poruku i digitalnu envelopu
- kada primalac primi digitalnu envelopu, dekriptuje je svojim tajnim ključem i na taj način dolazi do sesijskog ključa



- sesijskim ključem primalac dekriptuje primljenu poruku
- Mogućnost MITM napada?

Diffie Helman Key Exchange

- jedan od prvih protokola javnog ključa
- zapravo se radi o *key agreement* protokolu, a ne o *key exchange* protokolu
- metod za sigurnu razmjenu kriptografskih ključeva preko javnog kanala, tj. za sigurnu uspostavu tajnog ključa
- razmjenjuju se javni dijelovi ključa, nakon razmjene se kombinuju sa privatnim i dobija se dijeljeni ključ

Diffie Helman Key Exchange

- ▶ učesnici: A i B
- ▶ obe strane znaju g i n
- ▶ n – veliki prost broj (npr. 2 000 ili 4 000 bita)
 - ne treba biti ni previše velik, jer se gubi na efikasnosti, a ne dobija se mnogo na sigurnosti
- ▶ g – prost broj manji od n ($g < n$)
- ▶ A posjeduje a – privatni ključ – veliki broj, slučajna vrijednost manja od n ($a < n$)
- ▶ B posjeduje b – privatni ključ – veliki broj, slučajna vrijednost manja od n ($b < n$)
- ▶ A
 - $g^a \text{ mod } n$ – ovo A šalje preko mreže ka B
- ▶ B
 - $g^b \text{ mod } n$ – ovo B šalje preko mreže ka A
- ▶ A
 - $(g^b)^a \text{ mod } n = g^{ba} \text{ mod } n = g^{ab} \text{ mod } n$
- ▶ B
 - $(g^a)^b \text{ mod } n = g^{ab} \text{ mod } n$

Diffie Helman Key Exchange

- ▶ učesnici: A i B
- ▶ $n=11$
- ▶ $g=2$
- ▶ A posjeduje $a = 9$
- ▶ B posjeduje $b = 4$
- ▶ A
 - $g^a \text{ mod } n = 2^9 \text{ mod } 11 = 512 \text{ mod } 11 = 6$ – ovo A šalje preko mreže ka B
- ▶ B
 - $g^b \text{ mod } n = 2^4 \text{ mod } 11 = 16 \text{ mod } 11 = 5$ – ovo B šalje preko mreže ka A
- ▶ A
 - $(g^b)^a \text{ mod } n = 5^9 \text{ mod } 11 = 1\ 953\ 125 \text{ mod } 11 = 9$
- ▶ B
 - $(g^a)^b \text{ mod } n = 6^4 \text{ mod } 11 = 1\ 296 \text{ mod } 11 = 9$

Pasivni napadač

- Koji samo prisluškuje komunikacioni kanal.
- Ne može da izračuna isto što mogu A i B
- pasivni napadač može da dođe u posjed:
 - $g^a \text{ mod } n$
 - $g^b \text{ mod } n$
- može da izračuna:
 - $g^a \text{ mod } n * g^b \text{ mod } n = g^{a+b} \text{ mod } n$
- izračunati a ili b za dato g i n nije lako
- izračunati $g^a \text{ mod } n$ ili $g^b \text{ mod } n$ jestе lako

Aktivni napadač napadač

- Prisluškuje komunikacioni kanal, ali i manipuliše porukama.
- Ima svoj privatni ključ m.
- M
 - $g^m \text{ mod } n$ - ovo M šalje preko mreže ka B i pretvara se da je A
- B
 - $g^b \text{ mod } n$ - ovo B šalje preko mreže ka M
 - ovu poruku presreće M
- M
 - izračunava $(g^m)^b \text{ mod } n = g^{bm} \text{ mod } n$
- B
 - izračunava $(g^b)^m \text{ mod } n = g^{bm} \text{ mod } n$
- aktivni napadač M sada ima dva ključa koja je uspostavio sa A i B, te može s njima da komunicira
- pri tome A misli da komunicira s B, a B misli da komunicira s A
- sada svaku poruku koju A kriptuje i pošalje ka B, M može da dekriptuje (nakon što je presretne) i, nakon toga, može da je kriptuje ključem koji dijeli sa B i prosljedi ka B
- isto je i s porukama koje B šalje ka A
- aktivni napadač M može i da mijenja ove poruke
- ovaj napad se zove MITM (Man-in-the-Middle)

Kako riješiti ovaj problem koji postoji u prisustvu aktivnog napadača?

- A i B imaju svoje parove ključeva (privatni i javni) RSA algoritma
- javne ključeve RSA algoritma su razmijenili na siguran način
- A
 - $g^a \text{ mod } n$ - ovu poruku A digitalno potpisuje i šalje preko mreže ka B
- B
 - $g^b \text{ mod } n$ - ovu poruku B digitalno potpisuje i šalje preko mreže ka A

Zašto nam treba DH, ako imamo RSA ključeve?

- *Perfect Forward Secrecy*

- ako smo ključeve razmjenjivali isključivo korištenjem RSA algoritma, onda je, u slučaju kompromitacije privatnog ključa RSA algoritma, pasivni napadač koji je prisluskivao saobraćaj na mreži (i snimao ga) u stanju da naknadno dekriptuje sve ranije razmijenjene poruke
 - jer je u stanju da dekriptuje sve simetrične ključeve razmijenjene u digitalnim envelopama
- ako smo ključeve razmjenjivali pomoću DH i RSA algoritma, onda je, u slučaju kompromitacije privatnog ključa RSA algoritma, pasivni napadač koji je prisluskivao saobraćaj na mreži (i snimao ga) nije u stanju da dekriptuje ranije razmijenjene poruke
 - jer nije u stanju da se naknadno „ubaci“ u komunikaciju između A i B i izvrši MITM, tj. može samo da dođe u posjed $g^a \text{ mod } n$ i $g^b \text{ mod } n$, ali ne može naknadno da izračuna $g^{ab} \text{ mod } n$ (može samo da izračuna $g^{a+b} \text{ mod } n$)

Koristi se u različitim rješenjima:

- Secure Sockets Layer (SSL)/Transport Layer Security (TLS)
- Secure Shell (SSH)
- Internet Protocol Security (IPSec)
- Public Key Infrastructure (PKI)
- Viber, WhatsUp, Signal, ...

Lecture 10.

07 - Digitalni potpis

Digitalni potpis predstavlja elektronsku analogiju stvarnog potpisa rukom.

Slično potpisu rukom, digitalni potpis mora biti otporan na falsifikovanje, primalac mora biti u stanju da ga verifikuje, i pošiljalac, tj. onaj koji potpisuje, ne smije kasnije biti u stanju da ga porekne, tj. digitalnim potpisom se obezbeđuje i neporecivost poruke

Glavna razlika između potpisa rukom i digitalnog potpisa je sljedeća:

- digitalni potpis ne može biti konstanta, već mora biti funkcija čitavog dokumenta na kojem se pojavljuje - ako digitalni potpis ne bi bio funkcija čitavog dokumenta na kom se pojavljuje, onda bi bilo moguće kopirati ga i dodati na bilo koji drugi dokument

Digitalni potpis predstavlja postupak kojim se određeni segment bloka podataka, ili standardizovane poruke, kriptografski obilježava potpisnikovim privatnim ključem.

Cilj razvoja tehnike digitalnog potpisa je realizacija elektronskog ekvivalenta rukopisnog potpisa.

Osnovne karakteristike:

- **autentičnost** - digitalni potpis je garancija primaocu da je određeni pošiljalac poslao poruku
- **nemogućnost falsifikovanja** - digitalni potpis je dokaz da je pošiljalac, i niko drugi, potpisao dokument
- **nije ponovo upotrebljiv** - digitalni potpis je dio dokumenta, tako da maliciozni učesnik ne može potpis upotrijebiti na drugom dokumentu
- **očuvanje integriteta** - digitalno potpisani dokument se ne može promijeniti, a da se ta promjena ne detektuje
- **neporecivost** - pošiljalac koji je potpisao dokument ne može kasnije poreći da je to učinio

Dokumenti se mogu potpisati na različite načine:

- pomoću **simetričnih** šifarskih sistema i arbitrator-a,
- pomoću **asimetričnih** šifarskih sistema
- pomoću **asimetričnih** šifarskih sistema i jednosmjernih **hash** funkcija

Pomoću simetričnih šifarskih sistema i arbitrator-a

Arbitrator je treći učesnik kojem vjeruju strane u komunikaciji.

On komunicira sa učesnicima u komunikaciji pomoću tajnih ključeva, unaprijed razmijenjenih sigurnim kanalom

Sa svakim od učesnika u komunikaciji arbitrator dijeli zaseban tajni ključ, tj. broj tajnih ključeva jednak je broju učesnika u komunikaciji.

Koraci pri kreiranju digitalnog potpisa su sljedeći:

- pošiljalac kriptuje poruku za primaoca tajnim ključem K_A i šalje je arbitratoru
- arbitrator dekriptuje poruku pomoću ključa K_A
- dekriptovanu poruku i tvrdnju da je poruku primio od pošiljoca zajedno kriptuje tajnim ključem K_B
- arbitrator šalje poruku primaocu
- primalac dekriptuje poruku ključem K_B . Sada primalac može pročitati i poruku i tvrdnju arbitratora da je poruku poslao pošiljalac

Pomoću asimetričnih šifarskih sistema

Algoritmi s javnim ključem mogu se koristiti za kreiranje digitalnog potpisa. Kod nekih algoritama, kao što je RSA, i tajni i javni ključ mogu se koristiti za enkripciju.

Ukoliko se dokument enkriptuje koristeći tajni ključ dobiće se siguran digitalni potpis.

drugi algoritmi, kao što je DSA, koriste se isključivo za kreiranje digitalnog potpisa i ne mogu se koristiti za enkripciju.

Koraci pri kreiranju digitalnog potpisa su sljedeći:

- pošiljalac enkriptuje dokument svojim tajnim ključem i na taj način ga i potpisuje
- potpisani dokument šalje primaocu
- primalac dekriptuje dokument pomoću javnog ključa pošiljaoca i na taj način verifikuje potpis

Pomoću simetričnih šifarskih sistema i arbitrator-a

Asimetrični kriptografski algoritmi su, u praksi, često neefikasni pri potpisivanju velikih dokumenata - iz tog razloga se digitalni potpsi kreiraju pomoću jednosmjernih hash funkcija - umjesto potpisivanja čitavog dokumenta potpisuje se samo hash dokumenta

Kod ovog protokola jednosmjerna hash funkcija i algoritam za kreiranje digitalnog potpisa utvrđeni su unaprijed.

Koraci pri kreiranju i verifikaciji digitalnog potpisa su sljedeći:

- pošiljalac pravi digitalni sažetak (hash) H dokumenta D ,
- pošiljalac enkriptuje sažetak svojim privatnim ključem, potpisuje dokument, tj. kreira digitalni potpis DP,
- šalje dokument D i potpisani sažetak DP primaocu
- primalac pravi sažetak H_2 dokumenta koji je primio od pošiljaoca. Zatim, algoritmom za digitalni potpis, dekriptuje potpisani sažetak pomoću javnog ključa pošiljoca. Ako se primljeni i kreirani sažetak slažu, digitalni potpis je validan.

Dodati slike

08 - Digitalni sertifikati i PKI

Digitalni sertifikati

- digitalni sertifikat je struktura podataka digitalno potpisana od strane entiteta kojem se vjeruje – sertifikacionog tijela (Certificate Authority - CA)
- digitalni sertifikat je digitalno potpisana izjava jednog entiteta (sertifikacionog tijela), kojom se tvrdi da javni ključ i druge informacije drugog entiteta imaju tačno određene vrijednosti
- digitalni sertifikat je komponenta kojom se utvrđuje veza između identiteta subjekta i njegovog javnog ključa primjenom asimetričnog algoritma
- tehnika digitalnih sertifikata omogućuje realizaciju mehanizma kojim se pouzdano pridružuje javni ključ asimetričnog algoritma identitetu određenog subjekta
- kreiranje i digitalno potpisivanje sertifikata realizuje sertifikaciono tijelo (sertifikacioni autoritet), kao treća strana od poverenja (TTP) pasivnog tipa, čime se garantuje veza između javnog ključa i identiteta vlasnika sertifikata
- digitalni sertifikat najčešće ima specifičan format - X.509 standard
- polja koja sadrži standardni sertifikat definisana su u specifikaciji X.509 sertifikata

Svi X.509 sertifikati, pored potpisa imaju:

- verziju (version) - označava verziju sertifikata prema X.509 standardu. Verzija označava koje informacije se nalaze u sertifikatu. Do danas su definisane tri verzije.
- serijski broj (serial number) - entitet koji je izdao sertifikat dužan je da dodijeli jedinstven serijski broj (pozitivan) sertifikatu kako bi ga mogao razlikovati od svih do sada izdatih sertifikata. Ova informacija se koristi u različite svrhe, npr. kada se sertifikat povuče iz upotrebe njegov serijski broj se smješta u listu opozvanih sertifikata (certificate revocation list - CRL).
- identifikator algoritma za potpisivanje (signature algorithm identifier) - označava algoritam koji je sertifikaciono tijelo koristilo za potpisivanje sertifikata. Npr. RSA sa SHA-256, RSA sa SHA1, DSA.
- ime izdavača (issuer name) - ime entiteta (sertifikacionog tijela) koje je potpisalo sertifikat, prema X.500 standardu. Implementacije koje poštuju standard, moraju da podrže sljedeće atribute (ovo važi i za subject):
 - country,
 - organization,
 - organizational unit,
 - distinguished name qualifier,
 - state or province name,
 - common name (e.g. "Susan Housley"), and
 - serial number.
- Dodatno, implementacije koje poštuju standard moraju da podržavaju i domainComponent atribut.

- period validnosti (validity period) - svaki sertifikat je validan tačno određeno vrijeme. Period validnosti zavisi od različitih faktora, kao što je snaga privatnog ključa. Ovo polje se sastoji od 2 potpolja: ne prije (not before) – koje označava datum prije kog sertifikat nije validan; i ne poslije (not after) – koje označava datum poslije kog sertifikat nije validan,
- ime subjekta (subject name) - ime entiteta kojem se izdaje sertifikat (tj. ime vlasnika javnog ključa), u X.500 standardu. Primjer:
 - dvoslovni niz koji označava državu,
 - region u okviru države,
 - elektronska adresa (e-mail),
 - mjesto,
 - organizacija,
 - odeljenje u organizaciji i
 - ime vlasnika sertifikata.
- informacije o javnom ključu subjekta (subject public key info) - javni ključ subjekta, zajedno sa identifikatorom algoritma,
- jedinstveni identifikator izdavača (issuer unique identifier),
- jedinstveni identifikator subjekta (subject unique identifier) i
- ekstenzije.

Verzija 3, X.509v3, podržava ekstenzije. Neke ekstenzije koje se obično upotrebljavaju su:

- osnovna ograničenja (Basic Constraints) – da li se radi o sertifikatu CA tijela ili ne, te određuje maksimalnu „dubinu“ putanje koja uključuje i tekući sertifikat
- upotreba ključa (Key Usage) - koja ograničava upotrebu ključa u tačno određene svrhe – ako ova ekstenzija postoji, barem jedna upotreba ključa mora biti definisana (digitalSignature, nonRepudiation, keyEncipherment, dataEncipherment, keyAgreement, keyCertSign, cRLSign, encipherOnly, decipherOnly)
- politike sertifikata (Certificate Policies) – definiše politike pod kojim je sertifikat izdat, te politike upotrebe sertifikata u formi OID-a
- alternativna imena subjekta (Subject Alternative Name) - dozvoljava da i drugi entiteti budu povezani s ovim subjektom, npr. DNS imena, e-mail adresu, IP adresu. Postoji i Issuer Alternative Name

Verzija 3, X.509v3, podržava ekstenzije. Neke ekstenzije koje se obično upotrebljavaju su:

- proširena upotreba ključa (Extende Key Usage) - koja ograničava upotrebu ključa u tačno određene svrhe, zajedno sa ili bez Key Usage – ako ova ekstenzija postoji zajedno sa Key Usage, onda moraju biti usaglašene (TLS WWW server authentication, TLS WWW client authentication, Signing of downloadable executable code, Email protection, Binding the hash of an object to a time, Signing OCSP responses)
- CRL distribucione tačke (CRL Distribution Points) – lokacije sa kojih se može preuzeti CRL lista. Liste se mogu preuzeti različitim protokolima (npr. HTTP, FTP, LDAP)
- Delta CRL distribucione tačke (Freshest CRL ili Delta CRL Distribution Point) - lokacije sa kojih se može preuzeti delta CRL lista
- AKI (Authority Key Identifier) – definiše način za identifikaciju javnog ključa koji odgovara privatnom ključu korištenom za potpisivanje sertifikata. U slučaju samopotpisanih sertifikata može biti izostavljen

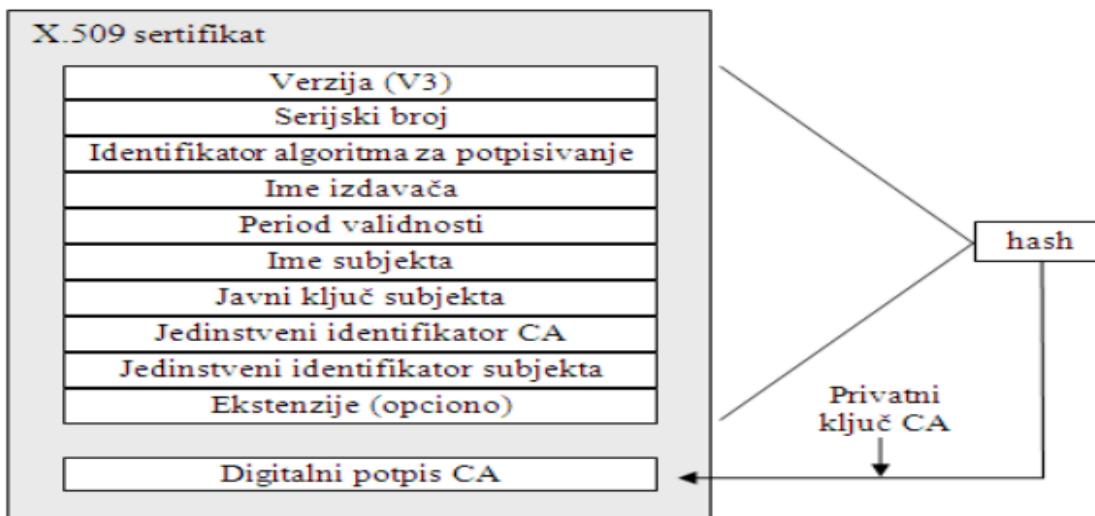
- AKI obično sadrži:
 - Key Identifier – hash javnog ključa CA-a
 - Issuer Name – ime izdavača (opcionalno)
 - Serial Number – serijski broj sertifikata izdavača (opcionalno)

Verzija 3, X.509v3, podržava ekstenzije. Neke ekstenzije koje se obično upotrebljavaju su:

- AIA (Authority Information Access) – pruža informacije o CA koji je izdao sertifikat. Cilj je da klijent može pronaći dodatne informacije o izdavaču i uslugama koje nudi. Ovim putem se navode putanje do sertifikata CA tijela i OCSP servisa (ali ne i CRL putanje). Postoji i Subject Information Access.
- identifikator ključa subjekta (Subject Key Identifier) – izvodi se iz ključa (npr. heš otisak vrijednosti ključa)
- ograničenja imena (Name Constraint) – koristi se samo kod CA sertifikata i označava prostor imena u kojem moraju da se nađu imena subjekata svih sertifikata koji su izdati od strane CA koji posjeduje ovaj atribut

Ekstenzije se mogu označiti kritičnim čime se označava da je provjera ekstenzija obavezna

- ako sertifikat ima ekstenziju KeyUsage označenu kao kritičnu i ako je njena vrijednost "keyCertSign", a ovaj sertifikat je prikazan za vrijeme uspostavljanja SSL komunikacije, trebao bi biti odbačen, jer ekstenzija KeyUsage označava da bi se privatni ključ koji odgovara ovom sertifikatu trebao koristiti samo za potpisivanje sertifikata, a ne i za uspostavljanje SSL konekcije



Uobičajene ekstenzije fajlova X.509 digitalnih sertifikata:

- .pem, .cer, .crt, .key – Privacy Enhanced Mail – Base64 enkodovani DER sertifikat – počinje sa
 - "—BEGIN CERTIFICATE—" i završava sa
 - "—END CERTIFICATE—"“
 - mnogi serveri očekuju javne i privatne ključeve u posebnim datotekama
- .cer, .der, .crt – obično u binarnom DER formatu, ali moguće je i da su Base64 enkodovani, mogu sadržavati i privatne ključeve
- .p7b, .p7c – PKCS#7 format, Base64, struktura koja sadrži samo sertifikat(e) ili CRL, ne i privatni ključ

- .p12 – PKCS#12 – može sadržavati i privatni ključ, zaštićen je lozinkom
- .pfx – PFX, prethodnik PKCS#12

Validacija digitalnog sertifikata:

- verifikacija da je digitalni sertifikat izdat od strane CA kojem se vjeruje – provjera potpisa
- verifikacija perioda validnosti
- verifikacija da digitalni sertifikat nije povučen
- verifikacija da se digitalni sertifikat koristi prema definisanoj politici

Kod web sajtova – dodatno:

- da je naziv domena odgovarajući

09 - CA

Sertifikaciono tijelo

- Sertifikaciono tijelo je tijelo sa najvećim stepenom autoriteta u PKI sistemu, tj. strana kojoj se vjeruje i koja jamči za identitete pojedinaca i organizacija.
- dva osnovna zadatka sertifikacionog tijela jesu:
 - da za određenog korisnika izvrši generisanje digitalnog sertifikata
 - da održava liste povučenih sertifikata
- Sertifikaciono tijelo svojim digitalnim potpisom, na bazi asimetričnog kriptografskog algoritma, garantuje vezu između javnog ključa i identiteta vlasnika sertifikata

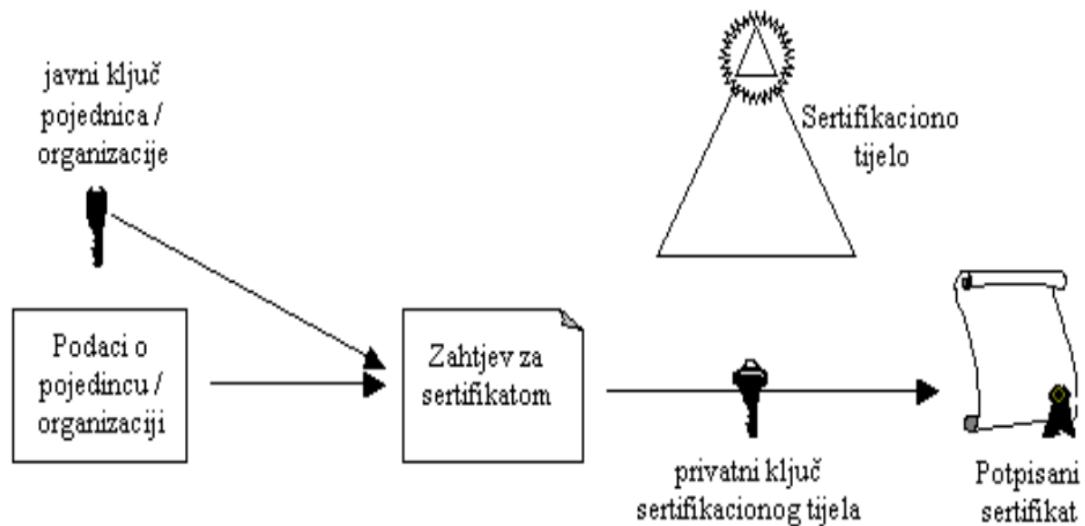
Postupak izdavanja digitalnog sertifikata pojedincu ili organizaciji može se vršiti na dva načina.

Prvi način izdavanja digitalnog sertifikata:

- pojedinac ili organizacija generiše par ključeva - tajni i javni,
- javni ključ, zajedno sa potrebnim identifikacionim informacijama, šalje sertifikacionom tijelu u formi zahtjeva za sertifikatom (certificate request),
- sertifikaciono tijelo verifikuje informacije pošiljaoca, tj. verifikuje da je pojedinac ili organizacija taj za koga se i izdaje,
- ako su svi podaci ispravni sertifikaciono tijelo generiše sertifikat koji sadrži javni ključ i druge identificujuće informacije,
- sertifikaciono tijelo generiše digitalni sažetak sertifikata i potpisuje ga svojim privatnim ključem, i na taj način potpisuje sertifikat. Potpisani sertifikat vraća pojedincu ili organizaciji koja je uputila zahtjev za sertifikatom

Drugi način izdavanja digitalnog sertifikata:

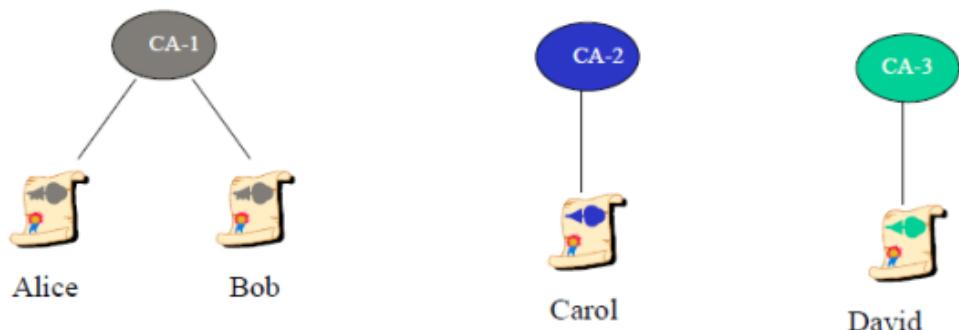
- sertifikaciono tijelo generiše par ključeva za pojedinca (ili organizaciju) koji zahtjeva digitalni sertifikat
 - tajni ključ mu šalje zajedno sa potpisanim digitalnim sertifikatom
- Prednost prvog načina izdavanja digitalnog sertifikata jeste u tome da se tajni ključ asimetričnog algoritma nalazi samo na jednoj lokaciji u sistemu, koja pripada vlasniku datog ključa.
- Prednost drugog načina izdavanja digitalnog sertifikata jeste u tome da sertifikaciono tijelo, kao strana od povjerenja, može kontrolisati i održati jedinstven kvalitet kreiranih ključeva i jedinstvenost procedure bezbjednog čuvanja generisanih ključeva. U savremenim PKI sistemima, ključevi se generišu i isporučuju subjektima, najčešće, kao sadržaj zaštićene memorije smart kartice ili sličnog medija.



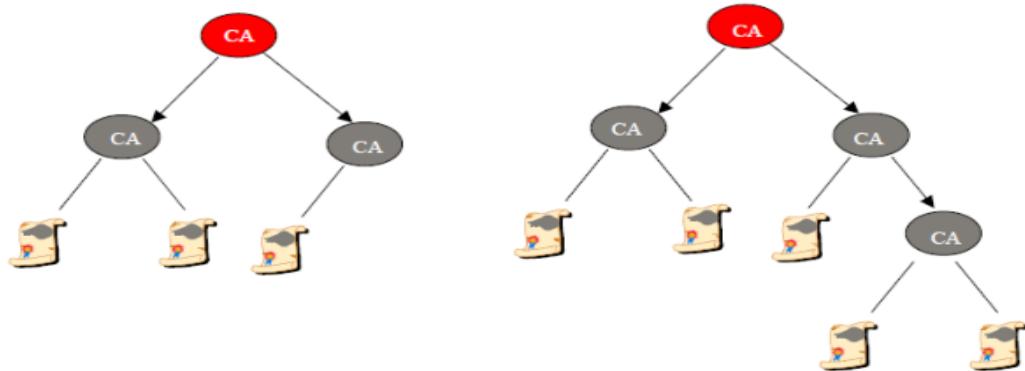
Vrste sertifikacionih tijela

- Hjerarhijski PKI sistem
 - Jedinstveni CA
- Mesh PKI sistem - *cross certification*
- Bridge CA sistem
- Lista povjerenja (Web model)

► Jedinstveni CA

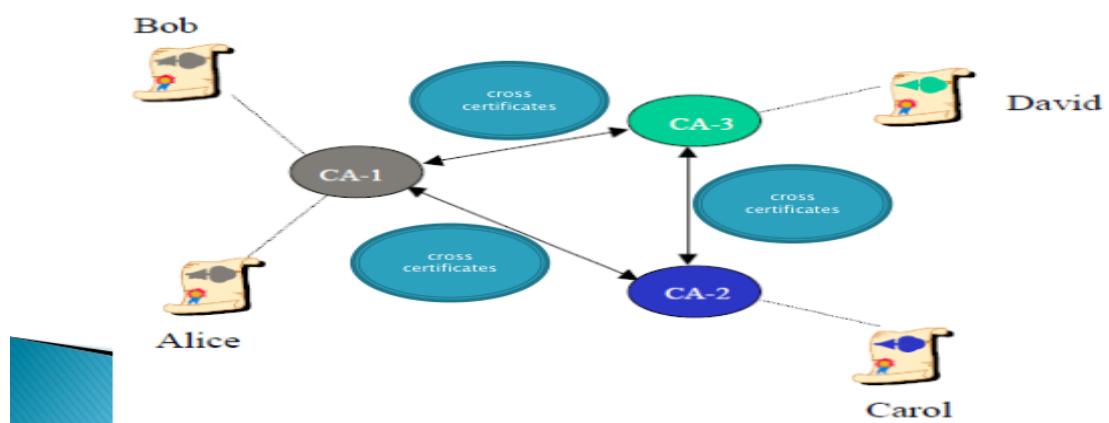


► Hjerarhijski PKI sistem

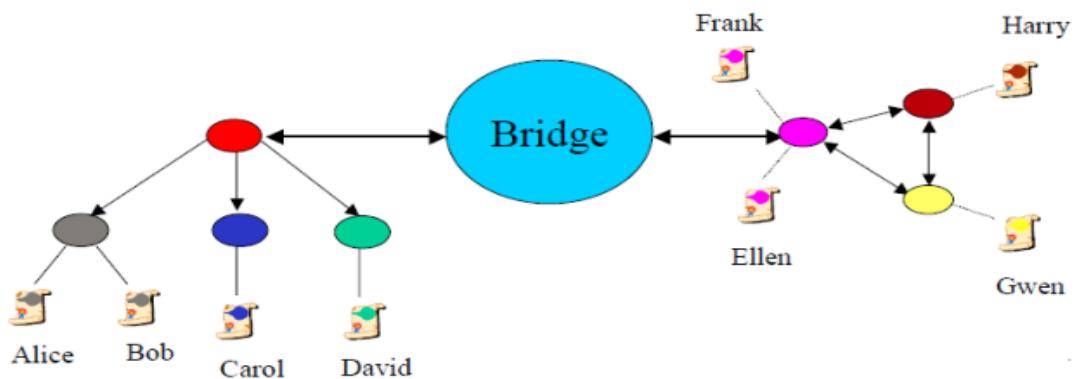


► Mesh PKI sistem – *cross certification*

- Hibridni mesh PKI sistemi

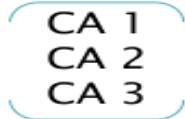


► Bridge CA sistem

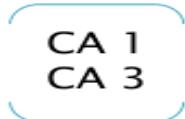


► **Lista povjerenja (*trust list model*)**

► **Korisnik A**



► **Korisnik B**



Namjena i vrste sertifikata

Različite vrste sertifikata koriste se za različite namjene, tako da možemo razlikovati:

- sertifikate sertifikacionih tijela (*certifying authority certificates*) - sertifikati koji sadrže javne ključeve sertifikacionih tijela,
- sertifikate web sajtova (*site certificates*) - sertifikati koje za autentikaciju koriste web serveri,
- lične sertifikate (*personal certificates*) - sertifikati koji autentikuju pojedince,
- sertifikate proizvođača softvera (*software publisher certificates*) - sertifikati koje proizvođači softvera koriste za potpisivanje izvršnih programa

Hijerarhija sertifikata

- Sertifikaciono tijelo može izdati digitalni sertifikat i drugim sertifikacionim tijelima - na ovaj način nastaje hijerarhija sertifikacionih tijela
- Gledajući s vrha, korijensko sertifikaciono tijelo (root CA) izdaje sertifikate sertifikacionim tijelima koja se nalaze na nivou neposredno ispod, dok ova sertifikaciona tijela izdaju digitalne sertifikate sertifikacionim tijelima na sljedećem nivou, i tako sve do dna hijerarhijskog stabla gdje se digitalni sertifikati izdaju krajnjim korisnicima
- Ovaj hijerarhijski model čini osnovu infrastrukture javnih ključeva (PKI)
- Validacija sertifikata - *bottom-up*
- Pojedinac ili organizacija može sam sebi izdati sertifikat i tada je riječ o samopotpisanim digitalnim sertifikatu (*self-signed certificate*)

Provjera validnosti sertifikata

Liste opozvanih sertifikata - CRL (*Certificate Revocation List*)

- omogućavaju provjeru validnosti digitalnih sertifikata učesnika u komunikaciji

- listu opozvanih sertifikata, uporedo sa generisanjem digitalnih sertifikata, kreira sertifikaciono tijelo. Sertifikaciono tijelo digitalno potpisuje listu opozvanih sertifikata, tako da su korisnici PKI sistema sigurni u autentičnost i integritet informacija sadržanih u dатој listi

OCSP (*Online Certificate Status Protocol*)

- Internet protokol koji se koristi za dobijanje informacija o statusu X.509 digitalnog sertifikata
- opisan je u RFC 2560 dokumentu
- status može biti: *good*, *revoked* i *unknown*

CRL

Standard X.509 v2 (RFC 2459) definiše osnovni skup informacija koje treba da sadrži lista opozvanih sertifikata:

- verzija formata liste opozvanih sertifikata predstavlja oznaku strukture date liste koja je specificirana u standardu X.509 pri čemu su validne verzije 1 i 2
- naziv izdavača liste opozvanih sertifikata
- vrijeme objavljivanja liste opozvanih sertifikata
- vrijeme kada će sertifikaciono tijelo objaviti sljedeću, ažuriranu verziju liste opozvanih sertifikata - nova verzija može biti objavljena i prije navedenog vremena, ali se ne smije objaviti poslije navedenog vremena
- popis opozvanih sertifikata sa razlogom njihovog povlačenja - opozvani digitalni sertifikati jednoznačno su određeni njihovim serijskim brojem
- ekstenzije - ovaj identifikator može se koristiti samo u verziji 2 i sadrži dodatna identifikaciona polja kao što su: redni broj liste, distribucione tačke liste, itd.
- identifikator algoritma koji je primjenjen za digitalno potpisivanje date liste opozvanih digitalnih sertifikata (RSA sa MD5, RSA sa SHA1, DSA, itd) i
- digitalni potpis liste opozvanih sertifikata

base i delta CRL liste

OCSP

- u pozadini može biti CRL lista
- OCSP reply mora biti digitalno potpisano kako bi obezbjedio dokaze da je:
 - odgovor došao od OCSP responder-a
 - da nije bio izmijenjen u prenosu
- krajnji entitet mora obezbijediti kopiju digitalnog sertifikata OCSP responder-a, kako bi mogao verifikovati odgovor
- digitalno potpisivanje ima negativan efekat na performanse
- RFC 6960

Sertifikaciona tijela

- globalna i nacionalna
 - globalna: Let's Encrypt CA, Comodo CA, Symantec CA, DigiCert CA, GeoTrust CA
 - Srbija: Pošte Srbije, MUP, Privredna komora Srbije, Halcom, „E-Smart Systems“ d.o.o.
- javna i privatna
- implementacije
 - komercijalne i
 - besplatne

Registraciono tijelo

- prije izdavanja digitalnog sertifikata CA mora, strogo poštujući odgovarajuću regulativu (npr. nacionalnu regulativu, evropsku uredbu o eIDAS - electronic IDentification, Authentication and trust Services, ...), jasno identifikovati budućeg vlasnika sertifikata - provjeriti i potvrditi njegove lične podatke
 - obično face-to-face procedura, uz odgovarajući identifikacioni dokument - ličnu kartu ili pasoš
- u većini slučajeva CA delegira ovu ulogu trećoj strani od povjerenja – registracionom tijelu (RA)
 - to može biti neka kompanija ili institucija

RA verifikuje podatke koji će se naći u digitalnom sertifikatu, a koji potiču iz identifikacionog dokumenta

Prihvata:

- zahtjeve za izdavanje digitalnih sertifikata
- povlačenje digitalnih sertifikata
- suspenziju digitalnih sertifikata
- promjenu podataka
- ...

RA i CA su često jedan entitet

- primjer: tijela koja izdaju EV SSL (Extended Validation SSL) digitalne sertifikate
 - pored EV SSL sertifikata, CA tijela izdaju i OV (Organization Validated) i DV (Domain Validated) SSL sertifikate, kao i Wildcard SSL sertifikate i Multi-Domain SSL sertifikate

Lecture 13.

10 - TSA

Kriptografski hardverski moduli

- Softverska kriptografska rješenja su ograničena sa stanovišta brzine centralnog procesora računara, efikasnosti prenosa podataka, arhitekture računarskog sistema, operativnog sistema, realizacije sistemskih funkcija i drugih faktora.
- Softverska rješenja zaštite posjeduju sigurnosne nedostatke, pošto se osjetljive kriptografske funkcije izvršavaju u okruženju potencijalno nebezbjednih operativnih sistema.
- Hardverski kriptografski moduli, realizovani u vidu računarskih koprocesora, predstavljaju bitnu karakteristiku savremenih rješenja zaštite računarskih mreža.

Osnovne funkcije:

- zaštita sigurnosno kritičnih podataka, kao što su kriptografski ključevi, korisničke šifre i privatni algoritmi zaštite,
- sigurna realizacija kriptografskih algoritama i ostalih funkcija zaštite nezavisno od uticaja ostalih aplikacija na računaru

Smart kartice predstavljaju mikroračunare sa centralnim procesorom (CPU), različitim tipovima memorije (ROM, RAM, NVM), ulazno/izlaznim sistemom i u najvećem broju primena poseduju koprocesorski modul

Smart kartice

Tipovi memorije:

- ROM - za smještanje operativnog sistema smart kartice - kapacitet od 4kB do 96 kB
- RAM - za privremeno smještanje promjenljivih u toku izvršenja kriptografskih procedura ili rutina operativnog sistema
- NVM - memorija čiji se sadržaj pamti i nakon prestanka napajanja - realizacija korištenjem E2PROM memorije (Electrically Erasable Programmable Read- Only Memory)

Tipovi kartica:

- memorijske kartice - Najstariji tip smart kartica koji posjeduje čip na kome se nalazi NVM memorija i potrebna logika za upis i čitanje iz date memorije - u savremenim primjenama najviše se koriste kao telefonske kartice
- beskontaktne smart kartice - primjena u sistemima kod kojih se mora izvršiti kontrola većeg broja ljudi u kratkom vremenskom intervalu
- klasične smart kartice sa kontaktnim interfejsom prema čitaču

TimeStamp Authority - TSA

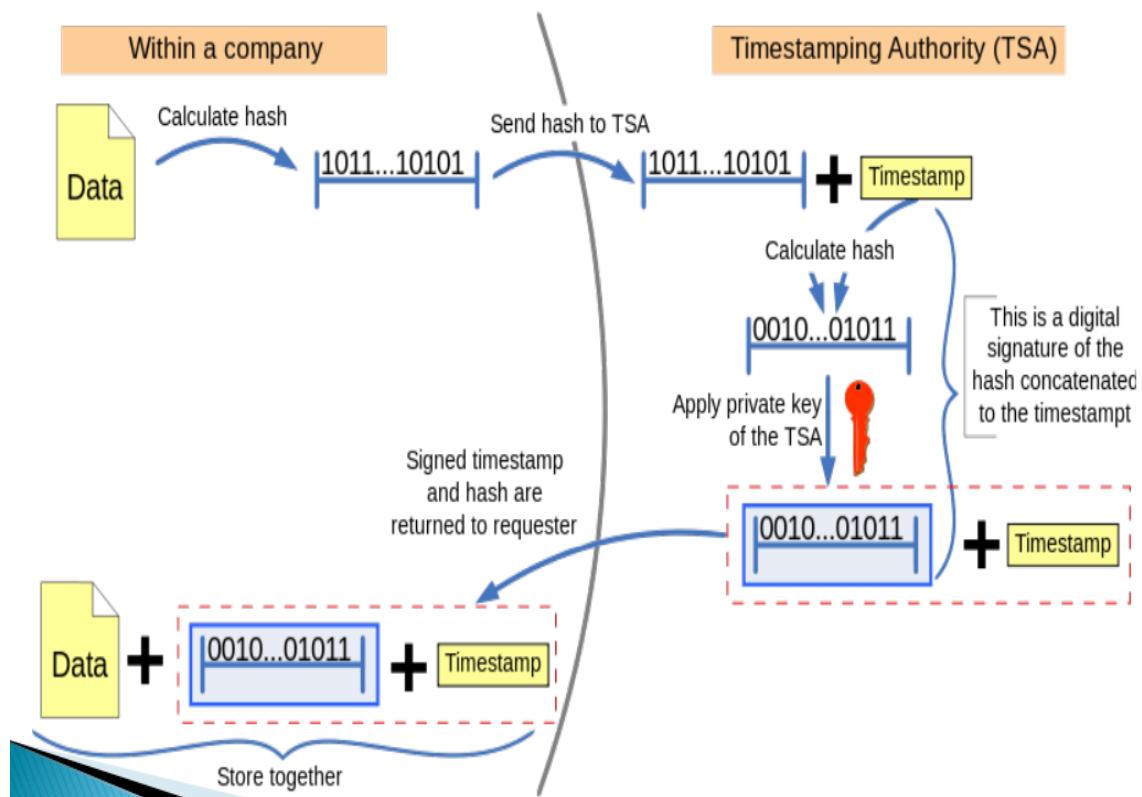
TSA je TTP koji kreira vremenski pečat (time-stamp token) radi mogućnosti naknadnog utvrđivanja da je datum postojao u određenom vremenskom trenutku, tj. da je podatak, dokument, transakcija i sl. postojao u određenom vremenskom trenutku

TSA je obavezan da:

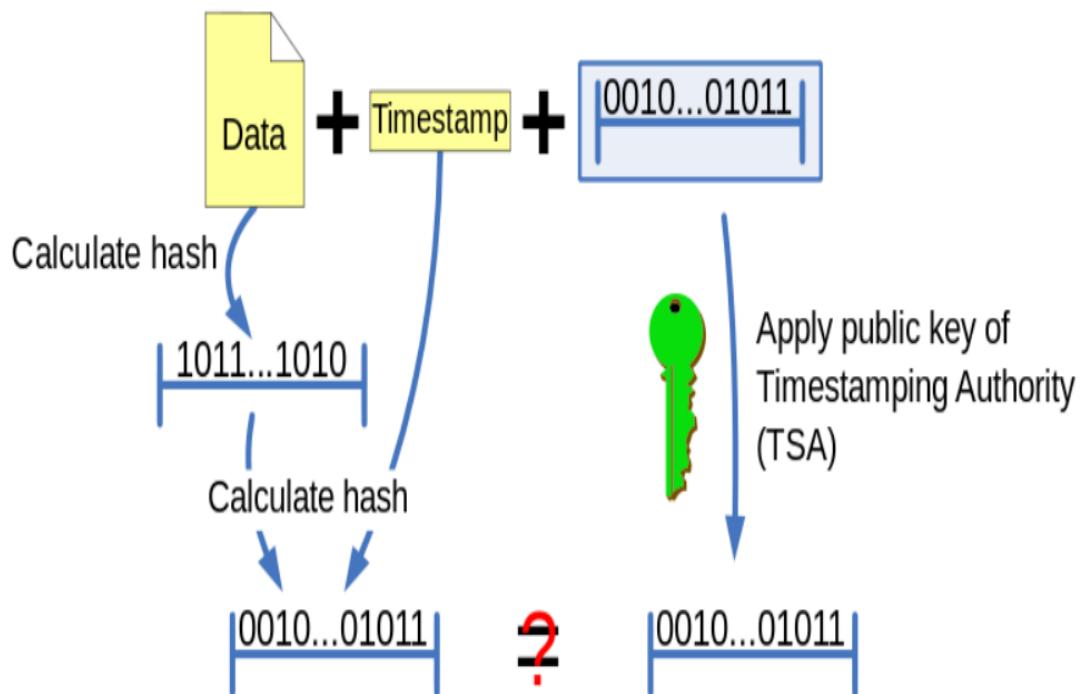
- koristi pouzdan izvor vremena
- da u svaki time-stamp token uključi vrijeme generisano iz pouzdanog izvora
- da uključi jedinstvenu integer vrijednost za svaki novogenerisani time-stamp token
- da proizvede validan time-stamp token nakon prijema validnog zahtjeva
- da sa svakim time-stamp tokenom uključi i identifikator za jedinstvenu identifikaciju sigurnosne politike pod kojom je token kreiran
- da se vremenski pečatira hash reprezentacija datuma
- da ne uključuje identifikaciju entiteta (koji je zahtijevao time-stamp) u time-stamp token
- da svaki time-stamp token potpiše koristeći ključ generisan ekskluzivno za ovu namjenu
- da u time-stamp token uključi dodatne informacije, ako to traži requester. Dodatne informacije su upisuju u extension polja, i to samo one ekstenije koje podržava TSA. Ako to nije moguće, TSA treba da vrati poruku o grešci.

RFC 3161

Trusted timestamping



Checking the trusted timestamp



Praktičan rad

- openSSL
- xCA
- ejbCA
- Microsoft Certificate Services