Sistemski Procesi

UDŽBENIK, POGLAVLJE 10.

Sistemski procesi – nulti proces

- Za obavljanje pojedinih zadataka operativnog sistema prirodno je koristiti procese. Ovakvi procesi se nazivaju **sistemski procesi** (**daemon**), jer su u službi operativnog sistema.
- Tipičan primer sistemskog procesa je **nulti** ili **beskonačni** (**idle**) proces, na koga se procesor prekljućuje, kada **ne postoji drugi spreman proces**.
- Beskonačni proces izvršava beskonačnu petlju, što znači da je beskonačan proces uvek ili **spreman** ili **aktivan** (on ne prelazi u stanje čekanja).
- Njegov prioritet je **niži od prioriteta svih ostalih procesa**, a on postoji za sve vreme aktivnosti operativnog sistema.

Sistemski procesi – dugoročni raspoređivač

- Drugi primer sistemskog procesa je proces **dugoročni raspoređivač** (**swapper**), koji se brine o zameni slika procesa.
- On se **periodično** aktivira, radi pozivanja operacije za zamenu slika procesa.
- Da bi se proces **uspavao**, odnosno da bi se njegova aktivnost zaustavila do nastupanja zadanog trenutka, on poziva odgovarajuću sistemsku operaciju.
- Ona pripada sloju za **rukovanje kontrolerima**, jer proticanje vremena registruje **drajver sata**.
- I proces dugoročni raspoređivač postoji za **sve vreme** aktivnosti operativnog sistema (jasno, ako ima potrebe za dugoročnim raspoređivanjem).

Sistemski procesi – procesi identifikator i komunikator

- U sistemske procese spada i proces **identifikator** (**login process**), koji podržava predstavljanje korisnika.
- Proces **identifikator** koristi **terminal**, da bi posredstvom njega stupio u interakciju sa korisnikom u toku predstavljanja, radi preuzimanja **imena** i **lozinke** korisnika.
- Po preuzimanju **imena** i **lozinke**, proces identifikator proverava njihovu ispravnost i, **ako je prepoznao** korisnika, tada stvara proces **komunikator**, koji nastavlja interakciju sa korisnikom.

Sistemski procesi – procesi identifikator i komunikator

- Pri tome, proces **identifikator** **prepušta** svoj terminal stvorenom procesu **komunikatoru** i **zaustavlja** svoju aktivnost.
- Ona se nastavlja tek nakon **završetka aktivnosti procesa komunikatora**.
- Tada proces identifikator opet **preuzima** opsluživanje terminala, da bi podržao novo predstavljanje korisnika.
- Prema tome, i proces **identifikator** postoji za **sve vreme** aktivnosti operativnog sistema.

Sistemski procesi – procesi identifikator i komunikator

- Za proveru ispravnosti imena i lozinke korisnika, neophodno je raspolagati **spiskovima imena** i **lozinki** registrovanih korisnika.
- Ovi spiskovi se čuvaju u posebnoj datoteci lozinki (**password file**).
- Svaki slog ove datoteke sadrži:
  - **ime** i **lozinku** korisnika
  - **numeričku oznaku** korisnika
  - **putanju radnog imenika** korisnika
  - putanju **izvršne datoteke**, sa inicijalnom **slikom korisničkog procesa komunikatora**
- Nekada se lozinke korisnika čuvaju u **posebnoj datoteci (shadow file).**

/etc/passwd

```
[veljko@Episteme ~]$ cat /etc/passwd
root:x:0:0::/root:/bin/bash
nobody:x:65534:65534:Nobody:/:/sbin/nologin
dbus:x:81:81:System Message Bus:/:/sbin/nologin
bin:x:1:1::/:/sbin/nologin
daemon:x:2:2::/:/sbin/nologin
mail:x:8:12::/var/spool/mail:/sbin/nologin
ftp:x:14:11::/srv/ftp:/sbin/nologin
http:x:33:33::/srv/http:/sbin/nologin
systemd-journal-remote:x:982:982:systemd Journal Remote:/:/
systemd-coredump:x:981:981:systemd Core Dumper:/:/sbin/nolo
uuidd:x:68:68::/:/sbin/nologin
dnsmasq:x:980:980:dnsmasq daemon:/:/sbin/nologin
rpc:x:32:32:Rpcbind Daemon:/var/lib/rpcbind:/sbin/nologin
avahi:x:979:979:Avahi mDNS/DNS-SD daemon:/:/sbin/nologin
colord:x:978:978:Color management daemon:/var/lib/colord:/s
cups:x:209:209:cups helper user:/:/sbin/nologin
deluge:x:977:977:Deluge BitTorrent daemon:/srv/deluge:/sbin
git:x:976:976:git daemon user:/:/usr/bin/git-shell
lightdm:x:620:620:Light Display Manager:/var/lib/lightdm:/s
nm-openconnect:x:975:975:NetworkManager OpenConnect:/:/sbin
nm-openvpn:x:974:974:NetworkManager OpenVPN:/:/sbin/nologin
ntp:x:87:87:Network Time Protocol:/var/lib/ntp:/bin/false
polkitd:x:102:102:PolicyKit daemon:/:/sbin/nologin
usbmux:x:140:140:usbmux user:/:/sbin/nologin
veljko:x:1000:1000:Veljko Petrovic:/home/veljko:/bin/bash
systemd-network:x:973:973:systemd Network Management:/:/sbi
systemd-resolve:x:972:972:systemd Resolver:/:/sbin/nologin
systemd-timesync:x:971:971:systemd Time Synchronization:/:/
geoclue:x:970:970:Geoinformation service:/var/lib/geoclue:/
```

/etc/passwd

```
veljko:x:1000:1000:Veljko Petrovic:/home/veljko:/bin/bash
```

Sistemski procesi – procesi identifikator i komunikator

- Za stvaranje procesa komunikatora **proces identifikator** koristi **putanju izvršne datoteke** koja sadrži inicijalnu sliku korisničkog procesa komunikatora.
- Tom prilikom on upotrebi **numeričku oznaku** prepoznatog korisnika kao **numeričku oznaku** vlasnika stvaranog **procesa komunikatora**, a putanju radnog imenika prepoznatog korisnika upotrebi kao putanju **radnog imenika** stvaranog procesa komunikatora.

Sistemski procesi – procesi identifikator i komunikator

- **Slogovi datoteke lozinki** se razlikuju obavezno po **imenima** i po **numeričkim oznakama** korisnika.
- Oni se mogu razlikovati po **putanjama korisničkih radnih imenika**, pa i po putanjama izvršnih datoteka.
- Za procese komunikatore, koji su prilagođeni posebnim potrebama pojedinih korisnika, se **ne smatra da su sistemski procesi**, jer oni ne predstavljaju sastavni deo operativnog sistema.
- Zbog ovakvih procesa komunikatora, moguće je zauzeti stanovište da **sloj za spregu sa korisnikom** i **nije deo operativnog sistema**, nego da pripada **višem** (**korisničkom**) **sloju** kao npr. **tekst editor** ili **kompajler**.

Sistemski procesi – procesi identifikator i komunikator

- Iz funkcije procesa komunikatora sledi da on ostvaruje **interaktivni nivo** komunikacije korisnika i operativnog sistema.
- Proces komunikator prihvata **sve komande**, koje dolaze **sa terminala** i tretira ih kao komande prepoznatog korisnika.
- Za proces komunikator prava **pristupa datotekama** se određuju na osnovu **numeričke oznake** njegovog vlasnika.

Sistemski procesi – procesi identifikator i komunikator

- **Procesi**, koje **stvara** proces komunikator (radi izvršavanja pojedinih komandi korisnika) imaju **ista prava** kao i **proces komunikator**, jer se podrazumeva da stvoreni procesi nasleđuju **numeričku oznaku** vlasnika procesa stvaraoca.
- Dok je u interakciji sa procesom komunikatorom, njegov vlasnik nema mogućnosti za **narušavanje zaštite datoteka**.
- Ako, **umesto vlasnika**, u interakciju sa procesom komunikatorom stupi **neki drugi korisnik**, zaštita datoteka se **narušava**, jer ovaj drugi korisnik dobija priliku da uživa **tuđa prava** pristupa datotekama.

Sistemski procesi – procesi identifikator i komunikator

- Zato, nakon predstavljanja, korisnik sme **prepustiti** svoj terminal drugom korisniku, tek **nakon što je završio** aktivnost svog procesa komunikatora.
- Radi toga, među komandama procesa komunikatora, obavezno postoji **posebna komanda**, namenjena baš procesu komunikatoru, a koja izaziva njegovo **uništenje**.
- Znači, svaki korisnik **započinje rad prijavom**, u toku koje se **predstavi** i **pokrene aktivnost** svog **procesa komunikatora**, a **završi rad odjavom**, u toku koje okonča aktivnost svog procesa komunikatora.

Sistemski procesi – procesi identifikator i komunikator

- Za zaštitu datoteka ključno je **onemogućiti neovlaštene pristupe datoteci lozinki**.
- Prirodno je da njen **vlasnik bude administrator** i da jedino sebi dodeli pravo **čitanja** i **pisanja** ove datoteke.
- Pošto su procesi identifikatori **sistemski procesi**, koji nastaju pri pokretanju operativnog sistema, nema prepreke da njihov **vlasnik bude administrator**.
- Iako na taj način **procesi identifikatori** dobijaju i pravo **čitanja** i pravo **pisanja datoteke lozinki**, to pravo korisnici **ne mogu da zloupotrebe**, jer, posredstvom procesa identifikatora, **jedino mogu proveriti da li su registrovani u datoteci lozinki**.
- Pri tome je bitno da svoja prava proces identifikator ne prenosi na proces komunikator. Zato proces komunikator **ne nasleđuje numeričku oznaku** vlasnika od svog stvaraoca procesa identifikatora.

Sistemski procesi – procesi identifikator i komunikator

- **Administrator** bez problema pristupa **datoteci lozinki**, radi **izmene** njenog sadržaja, **jer je on vlasnik svih procesa**, koje je stvorio i pokrenuo da bi izvršili njegove komande.
- Pošto **korisnici nemaju načina** da pristupe **datoteci lozinki**, javlja se problem **kako omogućiti korisniku da sam izmeni svoju lozinku**.
- Taj problem se može rešiti po uzoru na **proces identifikator**, koga **koriste** svi korisnici, **iako nisu njegovi vlasnici**.
- Prema tome, ako je **administrator vlasnik izvršne datoteke sa inicijalnom slikom procesa za izmenu lozinki**, dovoljno je naznačiti da **on treba da bude vlasnik i procesa nastalog na osnovu ove izvršne datoteke (SUID - Switch User IDentification program)**.

Sistemski procesi – procesi identifikator i komunikator

- Zahvaljujući tome, **vlasnik ovakvog procesa je administrator**, bez obzira ko je stvorio proces.
- U tom slučaju, nema smetnje da korisnik, koji izazove **stvaranje procesa za izmenu lozinke**, **pristupi datoteci lozinki**.
- Pri tome, proces za **izmenu lozinki dozvoljava korisniku samo da izmeni sopstvenu lozinku**, tako što od korisnika primi njegovo **ime**, **važeću** i **novu lozinku**, pa, ako su ime i važeća lozinka **ispravni**, on **važeću lozinku zameni novom**.

Sistemski procesi – procesi identifikator i komunikator

- Datoteka lozinki je **dodatno zaštićena**, ako su lozinke u nju upisane u izmenjenom, odnosno u **kriptovanom** (**encrypted**) **obliku**, jer tada administrator može da posmatra sadržaj datoteke lozinki **na ekranu**, ili da **ga štampa**, **bez straha** da to može biti **zloupotrebljeno**.
- Da bi se sprečilo **pogađanje tuđih lozinki**, proces identifikator treba da reaguje **na više uzastopnih neuspešnih pokušaja predstavljanja**, **obaveštavajući** o tome **administratora**, ili **odbijajući** neko vreme **da prihvati nove pokušaje** predstavljanja.
- Takođe, **korisnici moraju biti oprezni** da sami ne odaju svoju lozinku **lažnom procesu identifikatoru**. To se može desiti, ako se njihov **prethodnik ne odjavi**, nego ostavi svoj proces da opslužuje terminal, **oponašajući proces identifikator**.
- Ovakvi procesi se nazivaju **trojanski konji** (**trojan horse**).

Sistemski procesi – procesi identifikator i komunikator

- Zbog istovremenog postojanja **više procesa** i **nepredvidivosti preključivanja**, postoji nezanemarljiva mogućnost da više procesa **istovremeno pokuša da pristupi datoteci lozinki**.
- Ako su to samo **procesi identifikatori**, to i nije problematično, jer oni samo **preuzimaju** njen sadržaj.
- Ali, ako ovoj datoteci **istovremeno pokušaju pristupiti procesi** koji **menjaju** njen sadržaj, ili procesi koji **preuzimaju i/ili menjaju** njen sadržaj, rezultat pristupa je **nepredvidiv**, znači zavisan od redosleda pristupa i različit od rezultata potpuno **sekvencijalnog** pristupa.

Sistemski procesi – procesi identifikator i komunikator

- **Nepredvidivost** rezultata pristupa je posledica činjenice da **prekljućivanja** mogu da učine vidljivim samo **delimično izmenjen** sadržaj datoteke, što je **nemoguće** u slučaju **potpuno sekvencijalnog** pristupa.
- Vidljivost **delimično izmenjenog** sadržaja datoteke uzrokuje da **ukupna izmena** može da bude posledica **delimičnih izmena**, napravljenih u toku aktivnosti raznih procesa, **što je neprihvatljivo**.
- Iz istog razloga moguće je preuzimanje dela **novog** (**izmenjenog**) i dela **starog** (**neizmenjenog**) sadržaja datoteke, što je, takođe, **neprihvatljivo**.
- Zato je potrebno **sinhronizovati procese** koji pristupaju sadržaju iste datoteke.

Kriptografija

UMETNOST PRIMENJENE PARANOJE

Istorija kriptografije

Od starogrčkog: κρυπτός (skrivena) i λόγος (reč)

Sa tim je povezan i termin 'šifra' odn. engleski 'cypher' ili 'cipher'

Cipher/šifra dolaze od arapskog 'al sifr' što znači nula.

Zašto? Zato što su srednjevekovni Evropljani imali sujeveran strah od nule kao *koncepta.*

Istorija kriptografije

Najranije forme su stvari kao što je Cezarova šifra čiji je moderan ekvivalent rot13.

Šta je problem? Ako znam algoritam, znam da provalim šta je šifra. Jedini parametar cezarove šifre je za koliko mesta pomeramo azbuku, budući da je to 26 mogućih vrednosti odn. ni 5 bita bezbednosti za brute-force, to i nije nekakva zaštita.

Istorija kriptografije

Možemo to i bolje, uzmite Polibijev Kvadrat.

To je već 25! mogućih varijacija, odn. 15511210043330985984000000 mogućih ključeva. To je skoro 84 bita bezbednosti. Mnogo bolje.

Uprkos tome, mogli bi ste da potpuno razbijete bilo koji Polibijev Kvadrat bez puno napora.

Zašto? Informacije cure kroz *distribuciju slova,* u engleskom, na primer, najčešćih prvih 12 slova su, redom, ETAOIN SHRDLU. Ovo je osobina koju Polibijev kvadrat *uopšte* ne krije.

Šenonovi kriterijumi i šta čini dobar metod šifrovanja

1. Konfuzija
2. Difuzija

Istorijski, ovo nam i nije najbolje išlo

1. Prvi pokušaj da se to ispravi jesu polialfabetske šifre: prvo slovo menjamo po prvoj azbuci zamene, drugo po drugoj itd. Nije baš radilo zbog napada preko dekompozicije.
2. Drugi pokušaj? Kontinualno kližuća azbuka preko rotora.
    1. Ovo može da bude prilično bezbedno, čak i danas. RC4 je rotorska šifra. Čak postoje moderno-bezbedni algoritmi koji se mogu raditi *rukom* kao što je Šajnerov 'pasijans' algoritam.
    2. No, lako je napraviti grešku — primer Enigme.

Da li postoji savršena šifra?

Da! Od 1882, štaviše!

U pitanju je tkzv 'jednostruka zamena' odn. 'one-time pad'

Problem?

1. Trebaju nam *stvarno* slučajni brojevi
2. Treba nam razmena ključa velikog koliko i poruka sa *savršenom bezbednošću.* To je… problematično, jer ako već imao sigurni kanal komunikacije, što šifrovati?

O slučajnim brojevima

Generacija slučajnih brojeva je *apsolutno ključna* za efektnu kriptografiju, ne samo za jednostruke zamene.

Da stvar bude zanimljivija, to je funkcionalnost koju *baš* očekujemo od operativnog sistema.

Ipak, to su samo slučajni brojevi… koliko teško to može biti?

Veoma

O slučajnim brojevima

Anyone who considered arithmetical methods of producing random digits is, of course, in a state of sin.

—John von Neumann
General-Purpose Genius

O slučajnim brojevima

Random number generation is too important to be left to chance.

—Robert Coveyou
Oak Ridge National Laboratory

Pseudo-slučajni brojevi

Računari su determinističke mašine. Kao takve *ne mogu* da stvaraju slučajne brojeve.

Sa druge strane mogu da stvaraju *pseudo* slučajne brojeve, odn. niz *nepredvidivih vrednosti* koje su takve da ako znaš proizvoljno mnogo slučajnih brojeva i dalje ne možeš da predvidiš sledeću.

Da bi ovo radilo potreban je odličan algoritam koji se mora na početku nahraniti sa malo istinske slučajnosti: izvorom entropije. **Ovo je ograničen, dragocen resurs.**

Linux

/dev/random je kako Linux pokušava da nam ovde pomogne: to je izvor kvalitetne entropije koji se formira na osnovu šuma sa sistemske magistrale.

Ovo je odličan izvor *ali* se lako isprazni, a ako se to desi pokušaj da se pozove 'read' na tome će blokirati dok ne dobijemo svežu entropiju što zahteva jako puno vremena.

Ovo je napadačka površina za DDOS iscrpljivanjem entropije.

Linux

/dev/urandom nikada ne blokira, ali su brojevi generisani algoritmom.

Srećom, to je barem jako kvalitetan algoritam, barem od verzije 4.8.

Hardverski generatori slučajnih brojeva

Za *jako* bitne primene postoje hardverski generatori bazirani na, npr, atmosferskom šumu ili, najčešće, kvantno-mehanički procesi kao što je raspad nuklearnih izotopa. Ovo je skupo i nezgodno, budući da niko ne želi radio-izotop na svojoj matičnoj ploči.

Intel nudi rešenje: svaki Ivy Bridge i kasniji procesor ima na čipu RDRAND instrukciju koja proizvodi hardverski-generisane (rezistivan šum) slučajne brojeve.

Intel nudi problem

Prvi problem: Bezbednosno ključni mehanizam je sada (bukvalno) crna kutija iz koje izlaze bitovi, bez načina da se uverimo da se prave na pravi način.

Gore, mnogo gore od toga, jeste da je RDRAND implementacija… *podešena* od strane NSA.

NSA je permanentan protivnik kvalitetne kriptografije. Već se zna da su ključne NIST konstante manipulisane.

Šta hoćemo od moderne kriptografije?

1. Poverljivost
2. Integritet
3. Identitet

Kripto-sistem

Kripto-sistem je mehanizam koji postiže jedan od prethodnih ciljeva, najčešće sva tri, i sastoji se od protokola slanja, primanja, verifikacije poruka između nekog broja učesnika u nekom redosledu.

Kripto-sistemi se grade od kripto-primitiva, bazičnih matematičkih konstrukta koji nam omogućavaju da operišemo.

Kripto-primitivi

1. Jednosmerne, heš, funkcije.
2. Simetrična, tajni-ključ, enkripcija.
3. Asimetrična, javni-ključ, enkripcija.

Heš funkcije

1. Kriptografske
    1. HMAC
    2. Lozinka
    3. KDF
2. Opšte
    1. Verifikacija
    2. Adresiranje

Kriptografska heš funkcija

Neka funkcija H je kriptografski heš ako je komputaciono *izuzetno* skupo da:

- Za neku vrednost h naći ulaz i takav da H(i) = h
- Za neki ulaz i naći drugi ulaz j takav da H(i) = H(j)
- Naći bilo koja dva ulaza i i j takva da H(i) = H(j)

Moderne kriptografske heš funkcije

SHA-2 i SHA-3 i koncept kripto-agilnosti.

Šta je problem sa SHA-2, današnjom najčešće korišćenom heš funkcijom?

1. Kletva NSA
2. Izuzetna efikasnost implementacije
3. Delimični napadi
4. Napad povećanjem dužine

Napad sa povećanjem dužine

Funkcije porodice SHA-2, tj. svih koje koriste tkzv. Merkle-Damgard konstrukciju kao svoju osnovu imaju slabost gde ako znamo H(M1) i dužinu M1 onda možemo da izračunamo H(M1 || M2) za proizvoljni M2 gde je || operator konkatenacije.

Alternativne Heš Funkcije i Heš Funkcije Posebne Namene

1. Blake2b (Opšta kriptografska)
2. SipHash (Štiti od DDOS potencijala malicioznih kolizija u okviru MurmurHash3)
3. Argon2 (Šifre i KDF)

Simetrični algoritmi

Imamo deljenu tajnu između dve strane koje tajno komuniciraju.

Bezbednost zavisi 100% of toga da je ta tajna *deljena* a za druge *tajna.* Ta tajna se obično zove 'tajni ključ' ili samo 'ključ'

Simetrični algoritmi mogu raditi na blokovima ili na tokovima.

Algoritmi na tokovima rade kao jednostruka zamena, samo što postoji algoritam koji generiše 'blokče' koje koristimo. Jako moćan sistem, ali *jako* osetljiv na periodičnost i početna podešavanja.

ChaCha

```
#define ROTL(a,b) (((a) << (b)) | ((a) >> (32 - (b))))
#define QR(a, b, c, d) (            \
    a += b,  d ^= a,  d = ROTL(d,16),   \
    c += d,  b ^= c,  b = ROTL(b,12),   \
    a += b,  d ^= a,  d = ROTL(d, 8),   \
    c += d,  b ^= c,  b = ROTL(b, 7))
#define ROUNDS 20

void chacha_block(uint32_t out[16], uint32_t const in[16])
{
    int i;
    uint32_t x[16];

    for (i = 0; i < 16; ++i)
        x[i] = in[i];
    // 10 loops × 2 rounds/loop = 20 rounds
    for (i = 0; i < ROUNDS; i += 2) {
        // Odd round
        QR(x[0], x[4], x[ 8], x[12]); // column 0
        QR(x[1], x[5], x[ 9], x[13]); // column 1
        QR(x[2], x[6], x[10], x[14]); // column 2
        QR(x[3], x[7], x[11], x[15]); // column 3
        // Even round
        QR(x[0], x[5], x[10], x[15]); // diagonal 1 (main diagonal)
        QR(x[1], x[6], x[11], x[12]); // diagonal 2
        QR(x[2], x[7], x[ 8], x[13]); // diagonal 3
        QR(x[3], x[4], x[ 9], x[14]); // diagonal 4
    }
    for (i = 0; i < 16; ++i)
```

ChaCha

Koristi je interno Google, OpenSSH polako prelazi na ChaCha, proizvodi CSPRNG za Linux od 4.8 (simetrični algoritmi enkripcije su također dobri generatori PRN-ova)

Jedini ulaz su podaci i nonce.

Nonce?

Number used ONCE.

Veliki broj algoritama za enkripciju *zahteva* neki početni broj koji se *apsolutno ne sme ponovo koristiti.*

Veliki broj naizgled kompetentnih programera ponovo upotrebi Nonce ili IV i dobije se debakl kao što je DRM za PS3.

Blok-šifre

Rade na blokovima fiksne dužine.

Trik kod njih su lukavo složene operacije mešanja i zamene nad ulaznim podacima na način koji tajni ključ parametrizuje.

Ideal je da se postigne efekat lavine—svaki bit izlaza zavisi od svakog bita ulaza.

AES

Izuzetno dobar standard za simetričnu enkripciju, naročito AES-256.

Postoje kvalitetne hardverske implementacije.

Ali, AES takođe ima mračnu stranu: strašno je lako upucati se u nogu kada se koristi. Zašto? Zato što simetrične šifre enkriptuju *jedan jedini blok* i to je to.

Ako hoćemo više (hoćemo) treba nam operativni mod AES-a, a tu problemi nastaju.

Operativni Modovi i IV

IV je kao nonce ali sa sledećim osobinama:

Može biti javan

Ne sme da se ponavlja

Ne sme da bude predvidiv

Zamka ECB

Trivijalan način da se AES primeni na više podataka jeste da se podaci podele na blokove i ista operacija izvrši na svakom bloku.

Ovo se zove Electronic Code Book

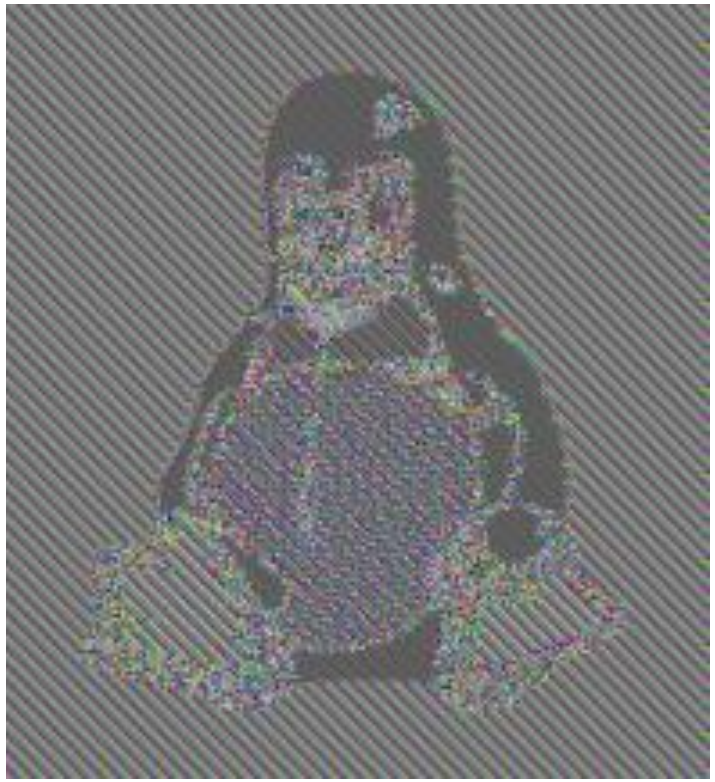
To vam je čak i *ponuđeno* u nekim implementacijama.

Ovo je *kataklizmično* loša ideja.

ECB primenjen na slici

ORIGINAL

ECB ENKRIPTOVANA VERZIJA



Brojački režimi

Bezbedan režim za upotrebu AES-a ovih dana jeste da se, efektivno, pretvori u sistem za enkripciju preko toka, tako što se tok generiše kroz AES nad nekakvim brojačem koji se dodaje (ne aritmetički) na IV.

Današnji state-of-the-art je AES-GCM

Asimetrična kriptografija

- **Simterična** kriptografija **nije podesna** za **kriptovanje poruka**, jer tada ključ kriptovanja **mora znati svaki pošiljalac poruke**, što ga dovodi u poziciju da može da **dekriptuje poruke drugih pošiljalaca**.
- To nije moguće u **asimetričnoj kriptografiji** (**public-key cryptography**), jer je njena osobina da se iz **ključa kriptovanja ne može odrediti ključ dekriptovanja**, pa poznavanje ključa kriptovanja **ne omogućuje dekriptovanje**.
- Ovakav ključ kriptovanja se zove **javni ključ** (**public key**), jer je on **dostupan svima**.

Asimetrična kriptografija

- Njemu odgovarajući **ključ dekriptovanja** je **privatan** (**tajan**), jer je dostupan samo osobama ovlašćenim za dekriptovanje. Zato se on naziva privatni ključ (private key).
- Prema tome, svaki **pošiljalac** poruke raspolaže **javnim ključem**, da bi mogao da **kriptuje poruke**, dok **privatni ključ** poseduje samo **primalac poruka**, da bi jedini mogao da **dekriptuje** poruke.
- **Asimetrična kriptografija** se temelji na korišćenju **jednostavnih** algoritama **kriptovanja** kojima odgovaraju **komplikovani** algoritmi **dekriptovanja**.

Asimetrična kriptografija

- Zato je asimterična kriptografija **mnogo sporija** od simetrične.
- Obično se **asimetrična kriptografija** koristi samo za **razmenu ključeva** potrebnih za **simetričnu kriptografiju**, pomoću koje se, zatim, kriptuju i dekriptuju poruke.