Sigurnost u računarskim mrežama

Kriptologija

(drugi deo)

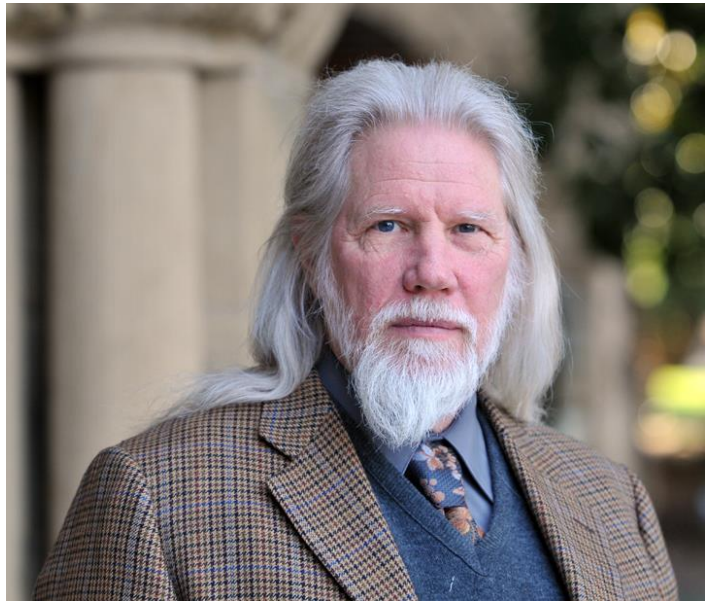
Nemanja Maček

Sadržaj

- Heš funkcije
- Problemi tipični za simetrične algoritme
- Razmena ključeva
- Kriptosistemi sa javnim ključem
- Digitalni sertifikati
- Infrastruktura javnih ključeva
- Osnovi kriptoanalize
- Primer TMTO napada: Rainbow tabele

O tajnama i sigurnosti

- „**The secret to strong security: less reliance on secrets**.“
  Whitfield Diffie



Jednosmerne funkcije

- **Jednosmerna funkcija** je funkcija oblika $y=f(x)$ takva da važi:
  - Za dato x, $f(x)$ se određuje relativno lako i efikasno.
  - Za dato $y=f(x)$, $x=f^{-1}(y)$ se određuje relativno teško.
- To što se $f^{-1}(y)$ se teško određuje ne znači da je nemoguće odrediti x.
  - Ako slomite tanjir (x) dobićete parčiće keramike tj. $f(x)$.
    - Ovo se lako radi, pokušajte.
  - Rekonstrukcija tanjira tj. dobijanje x na osnovu $f(x)$ je malo teži posao ali je izvodljiva.
- Ne može se matematički dokazati da jednosmerne funkcije postoje.
- Funkcija je kandidat za jednosmernu ako se efikasno izračunava, a vrednost inverzne funkcije teško nalazi.
- Primer:
  - Vrednost funkcije $f(x)=x^2$ u konačnom polju relativno se lako određuje.
  - Vrednost $f^{-1}(y)=x^{1/2}$ se teško nalazi.

Jednosmerne funkcije sa zamkom

- **Jednosmerna funkcija sa zamkom**, tj. privatna jednosmerna funkcija je funkcija za koju važi:
  - Za dato x, $f(x)$ se određuje relativno lako i efikasno.
  - Za dato $y=f(x)$, $x=f^{-1}(y)$ se određuje relativno teško.
  - Za dato $y=f(x)$ i tajnu informaciju z (zamka), $x=g(f^{-1}(y),z)$ određuje se relativno lako i efikasno.
- Primer:
  - Rasklapanje časovnika u delove $y=f(x)$ jednostavan je posao.
  - Sklapanje časovnika iz delova, tj. određivanje $x=f^{-1}(y)$ vremenski je zahtevno i komplikovano.
  - Sklapanje časovnika se može relativno brzo obaviti ako na raspolaganju imate uputstvo za sklapanje z.

Jednosmerna heš funkcija

- Jednosmerna heš funkcija $h = H(m)$ je funkcija za koju važi:
  - Na osnovu ulaznog podatka m proizvoljne dužine efikasno se određuje heš vrednost h fiksne dužine n.
  - Na osnovu heš vrednosti h, ulazni podaci m_1, m_2, ... koji generišu dati heš h ne mogu se odrediti ili se određuju veoma teško i neefikasno.
- Heš funkcije se projektuju tako da izmena jednog bita ulaznog podatka m izaziva promenu najmanje polovine bitova heša h.
  - Ovaj efekat se zove **lavinski efekat** i otežava pronalaženje kolizija.
- Heš funkcije su preslikavanja tipa **više-na-jedan** i nisu imune na pojavu kolizija.
  - **Kolizija** je pojava kada dva ili više različita ulaza m_1, m_2, … rezultuju istim izlazom h.
  - Ako se pronađu različite poruke na osnovu kojih heš funkcija generiše identični izlaz, smatra se da je otkrivena kolizija.
  - Heš funkcija je uslovno oslobođena kolizije ukoliko se teško generišu dve poruke na osnovu kojih se proizvodi isti heš $H(m)=H(m')$.

Dužina heša

- Jedan od parametara koji utiču na odabir heš funkcije jeste **dužina proizvedenog heša**.
- Primer rođendankog napada na heš dužine 64-bita:
  - Alisa priprema dve verzije ugovora.
    - Dokument *X* je ispravan.
    - Dokument *Y* dovodi firmu čiji je vlasnik Bob u bankrot ali uvećava sumu na Alisinom računu u banci.
  - Alisa pravi par kozmetičkih izmena u oba dokumenta (dodaje ili briše blanko znakove, poneku tačku ili zarez) generišući ukupno 2^{32} verzije oba dokumenta.
  - Alisa upoređuje heševe dokumenata i nalazi par dokumenta *X*-*Y* sa istim hešom.
  - Alisa i Bob potpisuju dokument koristeći protokol za digitalno potpisivanje u kome Bob potpisuje heš dokumenta.
    - Pri tome, Bob potpisuje ispravan dokument *X*.
  - Nakon par dana Alisa može na sudu da prikaže dokazni materijal u vidu neispravnog ugovora *Y* i dokaže da je Bob potpisao heš vrednost tog ugovora.

Dužina heša

- Pismo u 2^{37} varijacija [5].

Dear Anthony,

{This letter is / I am writing} to introduce {you to / to you} {Mr. / --} Alfred {P. / --}

Barton, the {new / newly appointed} {chief / senior} jewellery buyer for {our / the}

Northern {European / Europe} {area / division}. He {will take / has taken} over {the / --}

responsibility for {all / the whole of} our interests in {watches and jewellery / jewellery and watches}

in the {area / region}. Please {afford / give} him {every / all the} help he {may need / needs}

to {seek out / find} the most {modern / up to date} lines for the {top / high} end of the

market. He is {empowered / authorized} to receive on our behalf {samples / specimens} of the

{latest / newest} {watch and jewellery / jewellery and watch} products, {up / subject} to a {limit / maximum}

of ten thousand dollars. He will {carry / hold} a signed copy of this {letter / document}

as proof of identity. An order with his signature, which is {appended / attached}

{authorizes / allows} you to charge the cost to this company at the {above / head office}

address. We {fully / --} expect that our {level / volume} of orders will increase in

the {following / next} year and {trust / hope} that the new appointment will {be / prove}

{advantageous / an advantage} to both our companies.

Dužina heša

- Ako je heš prekratak a imate poverenja u heš funkciju, dužinu proizvedenog heša možete uvećati pomoću sledećeg algoritma:
  - $h_1 = H(m)$
  - $h_2 = h_1 | H(h_1 | m)$
  - $h_3 = h_2 | H(h_2 | m)$
  - …
- Postupak se nastavlja dok se ne dobije heš željene dužine.
- Primer:
  - Funkcija *H* proizvodi heš dužine 128 bitova.
  - Koristeći ovaj algoritam sa 8 iteracija dobićete heš dužine 1024 bita.

Primena heš funkcija: autentifikacija

- Prilikom prvog prijavljivanja na sistem:
  - Korisnik unosi lozinku.
  - OS računa heš unete lozinke i u tabelu smešta uređeni par (korisničko ime, heš lozinke).
- Provera:
  - Korisnik navodi korisničko ime i lozinku.
  - Sistem računa heš unete lozinke i proverava ga sa hešom u tabeli koji odgovara navedenom korisničkom imenu.
- Problemi: napad rečnikom, Rainbow tabelama ...
- Zaštita od napada prethodno generisanim rainbow tabelama: ***salt*** (so, NaCl).
- Opšti model upotrebe *salt* vrednosti:
  - *Salt* se generiše slučajno za svaku lozinku.
  - Računa se heš konkatenacija lozinke i soli: *shash* = H (*password* | *salt*).
  - U bazi se čuva izračunati heš i *salt* koja je slučajno određena: (shash, salt).
- Opasnost: pristup bazi sa lozinkama i salt vrednostima – i dalje je moguć napad rečnikom!

Problemi tipični za simetrične algoritme

- **Razmena ključeva**.
  - Pre šifrovanja pošiljalac i primalac moraju razmeniti ključ preko nekog sigurnog komunikacionog kanala.
  - Šta se dešava ako napadač dođe do ključa?
- **Broj ključeva**.
  - Ukoliko n entiteta međusobno komunicira potrebno je $\frac{1}{2} \cdot n \cdot (n-1)$ ključeva
  - 100 entiteta: 4.950 ključeva
  - 1.000 entiteta: 499.500 ključeva
- **Često menjanje ključeva**.
  - Šifrovanje većeg broja poruka istim ključem znatno smanjuje sigurnost komunikacije
  - Poželjno je da pošiljalac i primalac često menjaju ključ.
- **Nema adekvatne autentifikacije**.
  - Nemoguće je digitalno potpisati poruku, a o sertifikatima nema ni govora!

Problemi tipični za simetrične algoritme

- Rešenje nekih problema tipičnih za simetrične šifarske algoritme leži u naučnom radu.
  - Whitfield Diffie, Martin E. Hellman (1976): New Directions in Cryptography. IEEE Transactions on Information Theory, Vol IT22, No. 6.
- Ovaj naučni rad smatra se začetkom kriptografije sa javnim ključem!
- Dve značajne ideje predstavljene u ovom radu su:
  - **Kriptografija sa javnim ključevima**.
  - **Razmena ključeva** u simetričnim kriptosistemima.
  - ”*We now suggest a new public key distribution system which has several advantages. First, it requires only one key to be exchanged. Second, the cryptanalytic effort appears to grow exponentially in the effort of the legitimate users*.”
  - R. Merkle: “*His approach is different from that of the public key cryptosystems suggested above, and will be termed a public key distribution system. The goal is for two users, A and B, to securely exchange a key over an insecure channel. This key is then used by both users in a normal cryptosystem for both enciphering and deciphering*.”

Diffie-Hellmanov protokol za razmenu ključeva

- Diffie i Hellman ponudili su rešenje problema razmene ključeva zasnovano na **diskretnom logaritamskom problemu** (težina računanja diskretnih logaritama u konačnom polju).
- Diskretan logaritamski problem svodi se na sledeće:
- Dat je prost broj p i broj g takav (ovi brojevi nisu tajna) da je g generator ciklične multiplikativne grupe $G=\{0, 1, ..., p-1\}$.
  - Za datu vrednost y potrebno je naći x, tako da važi $y = g^x \bmod p$.
- Za male vrednosti modula, diskretni logaritam se može odrediti metodom grube sile, tj. prostim isprobavanjem različitih vrednosti.
  - Primer: za dato $p=11$, $g=2$ i $y=9$ možemo probati različite vrednosti x sve dok ne dobijemo onu za koju važi $2^x \bmod 11 = 9$.
- Za velike vrednosti modula (broj p ima 100 decimalnih cifara i više) smatra se da nije moguće rešiti diskretan logaritamski problem pomoću današnje tehnologije.

Diffie-Hellmanov protokol za razmenu ključeva

- Dve osobe (Alisa i Bob) se dogovaraju o ključu za šifrovanje preko nekog nesigurnog komunikacionog kanala.
- Algoritam:
  - Biraju se veliki prost broj p i broj g takav (ovi brojevi nisu tajna) da je g generator ciklične multiplikativne grupe $G=\{0, 1, ..., p-1\}$.
  - Alisa bira veliki slučajni broj x sa intervala $[1, p-1]$
  - Alisa šalje Bobu $g^x \pmod p$
  - Bob bira veliki slučajni broj y sa intervala $[1, p-1]$
  - Bob šalje Alisi $g^y \pmod p$
  - **Tajni ključ** koji Alisa i Bob nezavisno računaju: ključ $k = g^{xy} \pmod p = k' = g^{yx} \pmod p$
- Vrednosti k i k' su jednake $g^{xy} \bmod p$ i ne može ih izračunati neko ko prisluškuje kanal.
- Napadač može doći do p, g, X, i Y, ali da bi dobio vrednost k mora izračunati diskretni logaritam.

Diffie-Hellmanov protokol za razmenu ključeva

- Primer sa malim brojevima:
  - g=7, p=11

Alisa	Bob
Slučajno bira x=3 i čuva x u tajnosti	Slučajno bira y=6 i čuva y u tajnosti
Računa $A = 7^3 \pmod{11} = 2$	Računa $B = 7^6 \pmod{11} = 4$
Šalje Bobu A=2	Šalje Alisi B=4
Računa $k = 4^3 \pmod{11} = 9$	Računa $k = 2^6 \pmod{11} = 9$

- Izbor vrednosti g i p značajno utiče na sigurnost protokola!
  - Najvažnije je da n bude veliki broj jer je sigurnost protokola zasnovana na problemu određivanja diskretnog logaritma.
  - Generator g ne mora da bude ni prost ni veliki broj.
  - Može se izabrati i jednocifreni broj ali taj broj mora generisati grupu G.

Diffie-Hellmanov protokol za razmenu ključeva

- Diffie-Hellmanov protokol (bez autentifikacije poruka koje Alisa šalje Bobu i Bob šalje Alisi) nije otporan na napade tipa **čovek u sredini**!
- Eva izvodi napad na sledeći način:
  - Alisa na slučajan način bira vrednost x, računa $g^x \pmod p$ i tu vrednost šalje Bobu.
    - Eva presreće komunikaciju, bira slučajno v, računa $g^v \pmod p$ i tu vrednost šalje Bobu.
    - Bob je dobio vrednost $g^v \pmod p$ umesto $g^x \pmod p$
      - Bob ne zna da je to poslala Eva, a ne Alisa.
  - Bob na slučajan način bira vrednost y, računa $g^y \pmod p$ i tu vrednost šalje Alisi.
    - Eva presreće komunikaciju, bira slučajno w, računa $g^w \pmod p$ i tu vrednost šalje Alisi.
    - Alisa je dobila vrednost $g^w \pmod p$ umesto $g^y \pmod p$.
      - Alisa ne zna da je to poslala Eva, a ne Bob.
  - Alisa i Bob određuju „zajedničku tajnu“.
    - Alisa određuje ključ $k = g^{wx} \pmod p$, koji je poznat i Evi.
    - Bob određuje ključ, $k' = g^{vy} \pmod p$ koji je takođe poznat Evi.

Ideja javnog ključa

- **Ideja javnog ključa** sastoji se u konstrukciji kriptosistema takvih da je na osnovu javno poznate funkcije šifrovanja nemoguće u razumnom vremenu odrediti tajnu funkciju dešifrovanja!
- Kriptosistem s javnim ključem sastoji se od dve familije funkcija koje se ne kriju (za šifrovanje i dešifrovanje), a konkretne funkcije se izvode na osnovu privatnog i javnog ključa.
  - Funkcija šifrovanja sa javnim ključem na osnovu javnog ključa i ulaznih podataka proizvodi šifrat.
  - Funkcija dešifrovanja na osnovu privatnog ključa i šifrata proizvodi originalnu poruku.
  - Javni ključ je poznat onim osobama sa kojima korisnik želi da komunicira.
  - Tajni ključ poznat samo korisniku koji je ovlašćen da dešifruje poruke.
- Privatni i javni ključ su matematički povezani.
  - Privatni ključ se ne može odrediti na osnovu javnog ključa u konačnom vremenu i s konačnim resursima a na osnovu današnjeg stepena razvoja tehnologije i algoritama.
- Osnovno svojstvo kriptografije s javnim ključem je **poverljivost**: poruku koju Alisa šalje Bobu ne može pročitati niko drugi, jer nema Bobov privatni ključ.

Ideja javnog ključa

- Neka je: m – otvoreni tekst, c – šifrat, k_{priv} - privatni ključ, k_{pub} – privatni ključ.
- Funkcija šifrovanja: $c = E(m, k_{pub})$.
- Funkcija dešifrovanja: $m = D(c, k_{priv})$.
- Alisa i Bob međusobno razmenjuju javne ključeve i komuniciraju na sledeći način:
  - Alisa šalje Bobu poruku šifrovanu Bobovim javnim ključem:
    - $c_1 = E(m_1, k_{pub,bob})$.
  - Bob dešifruje poruku svojim privatnim ključem:
    - $m_1 = E(c_1, k_{priv,bob})$.
  - Bob šalje Alisi poruku šifrovanu njenim javnim ključem:
    - $c_2 = E(m_2, k_{pub,alice})$.
  - Alisa dešifruje poruku svojim privatnim ključem:
    - $m_2 = D(c_2, k_{priv,alice})$.
- Ukoliko više korisnika želi da komunicira koristeći kriptografiju sa javnim ključem, onda se javni ključevi tih korisnika mogu smestiti na takozvani server ključeva.

Ideja javnog ključa

- **Digitalni potpis** (engl. *digital signature*) je elektronska verzija potpisa na osnovu kog se može
  - Identifikovati pošiljalac
  - Dokazati verodostojnost poruke.
- Digitalno potpisivanje se uslovno može posmatrati kao šifrovanje podataka privatnim ključem.
- Digitalni potpis pruža obezbeđuje sigurnosnu uslugu **neporecivosti**.
  - Ako je Alisa poslala Bobu potpisanu poruku, ona kasnije može reći da to nije učinila.
  - Bob uvek može na osnovu potpisa dokazati da je poruku koju je primio poslala Alisa, zato što je ona vlasnik privatnog ključa kojim je poruka potpisana.
- Digitalni potpisi su usko povezani sa heš funkcijama.
  - Heš funkcija služi da se dužina potpisa svede na razumnu dužinu a da potpis pri tome ne izgubi svoj integritet.
  - Ukoliko potpišete poruku dužine 20 MB, dobićete potpis dužine 20 MB, što znači da ćete nekom poslati duplo veću količinu podataka (šaljete i poruku i potpis).
  - Ukoliko potpišete heš poruke, poslaćete poruku uvećanu za dužina proizvedenog heša.

Ideja javnog ključa

- Neka je:
  - *H* operacija izračunavanja heša,
  - *S* operacija potpisivanja (šifrovanje privatnim ključem),
  - *V* operacija provere potpisa (dešifrovanje javnim ključem).
- Komunikacija između pošiljaoca i primaoca odvija se prema sledećem protokolu:
  - Alisa računa heš poruke i digitalno ga potpisuje:
    - $s = S\,(H(m), k_{priv,alice})$.
  - Alisa šalje Bobu poruku i potpis: (m, s).
  - Bob proverava potpis, tj. dešifruje ga Alisinim javnim ključem i dobija jednu heš vrednost:
    - $h_1 = V\,(s, k_{pub,alice})$.
  - Bob izračunava heš primljene poruke i dobija drugu heš vrednost:
    - $h_2 = \mathrm{H}(m)$.
  - Ukoliko su dobijene vrednosti za heš jednake, potpis je validan.

Algoritam RSA

- RSA je verovatno najpopularniji asimetrični kriptosistem.
- Tvorci algoritma su Ronald Rivest, Adi Shamir i Leonard Adleman (*RSA Data Security*).
- Postojanje slabih tačaka pravilno implementiranog kriptosistema, sa ključevima generisanim na osnovu određenih preporuka, do sada nije ni dokazano ni opovrgnuto!
- Sigurnost RSA zasniva se na složenosti **faktorizacije velikih brojeva**.
- Javni i tajni ključ određeni su parom velikih prostih brojeva (200 dekadnih cifara i više).
  - Smatra se da je težina određivanja otvorenog teksta na osnovu šifrata, bez adekvatnog privatnog ključa, jednaka težini faktorizacije proizvoda dva velika prosta broja.
- Sam algoritam i njegova sigurnost zasnovani su na činjenicama da je:
  - Lako odrediti da li je veliki broj prost i pomnožiti dva velika prosta broja,
  - Teško faktorisati veliki broj koji je proizvod dva velika prosta broja (odnosno dobiti njegove početne proste faktore).

Algoritam RSA

- Par ključeva generiše se na sledeći način:
  - Generišu se dva velika prosta broja p i q.
  - Računa se Ojlerova funkcija $\varphi(n) = (p-1) \cdot (q-1)$.
  - Bira se celobrojna vrednost e sa intervala $1 < e < \varphi(n)$ koja je uzajamno prosta sa $\varphi(n)$.
  - Izračunava se d tako da važi $e \cdot d = 1 \pmod{\varphi(n)}$, odnosno $d = e^{-1} \pmod{\varphi(n)}$.
  - Drugim rečima, d je multiplikativni inverzni element od e po modulu $\varphi(n)$.
- Privatni ključ je uređeni par (d, n).
- Javni ključ je uređeni par (e, n).
- Ako je m otvoreni tekst, tj. poruka, a c šifrat, operacije šifrovanja i dešifrovanja su sa:
  - $c = m^e \pmod{n}$.
  - $m = c^d \pmod{n}$.
- Ukoliko je poruka m mora veća od n, pošiljalac deli svoju poruku na blokove čija je vrednost manja od n i šifruje ih jedan za drugim.

Algoritam RSA

- Primeri sa malim brojevima.
- Javni ključ (n, e)=(55, 27). Poruka m=2. Šta je šifrat?
  - $c = m^e \pmod{n} = 2^{27} \bmod 55 = 18$.
- Javni ključ (n, e)=(55, 27), šifrat c=10. Koji je otvoreni tekst?
  - Potrebno je odrediti privatni ključ kako bi mogli da dešifrujemo c.
  - Ovo je za male vrednosti n moguće.
  - Ako su p i q odabrani tako da su generisani ključevi od 2048 bita, faktorizacija u razumnom vremenu je nemoguća.
  - Postupak je sledeći:
    - Parametar n se faktoriše na proste činioce: $n = 55 = 11 \cdot 5$.
    - Faktori su p=11, q=5.
    - Određuje se: $\varphi(n) = (11–1) \cdot (5–1) = 40$.
    - Iz relacije koja povezuje ključeve $27 \cdot d = 1 \pmod{40}$ određuje se privatni ključ d=3.
    - Otvoreni tekst se dobija jednačinom $m = 10^3 \pmod{55} = 10$.

Algoritam RSA

- Primeri sa malim brojevima.
- Javni ključ (n, e)=(403, 103). Treba potpisati poruku m=3.
  - Potrebno je odrediti privatni ključ kako bi mogli da potpišemo poruku m.
  - Ovo je za male vrednosti n moguće.
  - Ako su p i q odabrani tako da su generisani ključevi od 2048 bita, faktorizacija u razumnom vremenu je nemoguća.
  - Postupak je sledeći:
    - Parametar n se faktoriše na proste činioce: $n = 403 = 13 \cdot 31$.
    - Ffaktori su p=13, q=31.
    - Određuje se: $\varphi(n) = (13–1) \cdot (31–1) = 360$.
    - Iz relacije koja povezuje ključeve $103 \cdot d = 1 \pmod{360}$ određuje se privatni ključ d=7.
    - Digitalni potpis poruke m=3 određuje se pomoću izraza:
      $s = m^d \pmod{n} = 3^7 \pmod{403} = 172$.

Algoritam RSA

- Primer sa malo većim brojevima.
- Neka je p = 47 i q = 71.
- Generisaćemo par RSA ključeva i šifrovati poruku m = 688232687966668 3.
  - Najpre se određuje: n = p · q = 3337 i $\varphi(n)$ = (p-1) · (q-1) = 46 · 70 = 3220.
  - Zatim se bira *e* iz intervala [1, 3219].
  - Može se uzeti *e* = 79, što je uzajamno prosto sa 3220.
  - Za određivanje multiplikativnog inverznog elementa broja 79 po modulu 3220, primenjujemo Euklidov algoritam na brojeve 3220 i 79:
    - 3220 = 40 · 79 + 60
    - 79 = 1 · 60 + 19
    - 60 = 3 · 19 + 3
    - 19 = 6 · 3 + 1
    - 3 = 3 · 1

Algoritam RSA

- Primer sa malo većim brojevima.
  - Primenom jednakosti po obrnutom redu, dobija se izraz za jedinicu kao linearnu kombinaciju brojeva $e = 79$ i $r = 3220$.
    - $1 = 19 – 6 \cdot 3 = 19 – 6 \cdot (60 – 3 \cdot 19) = -6 \cdot 60 + 19 \cdot 19 = -6 \cdot 60 + 19 \cdot (79 – 60) =$ $= 19 \cdot 79 – 25 \cdot 60 = 19 \cdot 79 - 25 \cdot (3220 – 40 \cdot 79) = 1019 \cdot 79 – 25 \cdot 3220$
  - Dobijena jednačina po modulu 3220 postaje $1019 \cdot 79 = 1 \bmod 3220$
  - To znači da je 1019 inverzna vrednost za 79 tj. d = 1019.
  - Privatni ključ je (79, 3337), a javni (1019, 3337).
  - Poruku m = 688232687966668 3 razbićemo na blokove tako da svaki bude manji od n=3337.
  - To su blokovi: 688, 232, 687, 966, 668, 3.
  - Šifrovanjem prvog bloka dobija se: $68879 \bmod 337 = 1570$.
  - Nastavljajući na isti način dobijamo šifrat: c = 1550 2756 2714 2276 2423 0158.

Digitalni sertifikati

- Kriptografija s javnim ključevima rešava problem sigurnog kanala za razmenu ključeva i broja ključeva potrebnih za sigurnu komunikaciju većeg broja osoba.
- Problem koji nije rešen je integritet javnih ključeva!
  - Da li napadač na server ključeva može da podmetne svoj javni ključ i lažno se predstavi?
- Ovaj problem se rešava pomoću sertifikata i infrastrukture javnih ključeva.
- **Digitalni sertifikat** (engl. *certificate*) čine:
  - javni ključ,
  - informacije o identitetu (ime, identifikator korisnika – UID, ...),
  - informacije koje se tiču ovlašćenja korisnika, npr. dozvole za pristup resursima (opciono),
  - jedan ili više digitalnih potpisa.
- Digitalni potpis je **overa sertifikata**.
- Sertifikate potpisuju **strane kojima se veruje**.
- Potpisom se overava veza između identiteta korisnika i javnog ključa.
- Sertifikat je: javni ključ sa opisom identiteta korisnika i potpisom koji je izdala strana kojoj se veruje kojim je overena veza između identiteta i ključa.

Digitalni sertifikati

- Digitalni sertifikati obezbeđuju podršku za:
- **Proveru identiteta**.
  - Primer 1: provera identiteta klijenta od strane servera i obrnuto (SSL).
  - Primer 2: digitalni potpis e-pošte u kombinaciji sa sertifikatom koji identifikuje pošiljaoca obezbeđuje snažne dokaze da je osoba identifikovana sertifikatom zaista poslala tu poruku.
- **Proveru integriteta**.
  - Digitalni sertifikat obezbeđuje integritet poruka, to jest onemogućava da korisnik primi izmenjenu ili oštećenu poruku.
- **Autorizaciju pristupa**.
  - Digitalni sertifikat može da zameni korisnička imena i lozinke.
- **Neporicanje**.
  - Digitalni sertifikati potvrđuju korisnički identitet!
  - Primer 1: Korisnik ne može kasnije da odbaci digitalno označene transakcije, poput kupovina preko Web lokacije.
  - Primer 2: Potpisnik ne može kasnije da porekne slanje digitalno potpisane e-pošte.

Infrastruktura javnih ključeva

- **Infrastruktura javnih ključeva** (engl. *public key infrastructure*, *PKI*) je strukturirani sistem koji obezbeđuje funkcije:
  - Izdavanja sertifikata
  - Poništavanja sertifikata
  - Skladištenja sertifikata i listi poništenih sertifikata
  - Uspostavljanje relacija poverenja.
- PKI objedinjuje kriptografske tehnike asimetričnog šifrovanja i digitalnog potpisivanja, softver i mrežne servise u kompletnu, široko rasprostranjenu sigurnosnu arhitekturu.
- Tri osnovne komponente infrastrukture javnih ključeva su:
  - sertifikacioni centar (engl. *certificate authority*, *CA*),
  - registracioni centar (engl. *registration authority*, *RA*),
  - skladište sertifikata.

Infrastruktura javnih ključeva

- **Sertifikacioni centar**.
  - *CA* je centralna komponenta PKI koja:
    - Generiše i izdaje sertifikate
    - Potpisuje izdate sertifikate svojim privatnim ključem *CA*
    - Poništava sertifikate
  - Korišćenjem javnog ključa *CA* svako može proveriti potpis *CA* na sertifikatu i samim tim integritet sertifikata.
  - Analogiaja: *CA* je odgovoran za generisanje sertifikata i njihov integritet slično kao što je MUP odgovoran za lične karte i vozačke dozvole!
  - *CA* se štiti metodama samouništenja u slučaju napada (*tamper-proof* metode)
    - U slučaju napada koji ugrožava integritet *PKI, CA* uništava sve ključeve.
  - Može se realizovati zatvoreno pomoću gotovih *PKI* rešenja ili pomoću javnih *CA* servisa.
  - Osnovni zadatak ustanove koja pruža uslugu izdavanja digitalnih sertifikata jeste da bude **poverljiva treća strana** kojoj veruju učesnici u komunikaciji.

- **Registracioni centar**.
  - *RA* je komponenta *PKI* koja osigurava proces registracije korisnika, prihvata i obrađuje zahteve za izdavanje sertifikata, i iste prosleđuje *CA* radi izdavanja sertifikata.
  - *RA* se odnosi na ljude, procese i alate za registrovanje i administriranje korisnika *PKI*.
  - *RA*/*CA* podseća na službu za izdavanje pasoša:
    - Određena grupa ljudi (*RA*) proverava identitet čoveka koji želi da mu se izda pasoš.
    - *CA* pravi pasoš i prosleđuje ga korisniku.
  - Identifikacija korisnika prilikom registracije je ključni je korak u izdavanju sertifikata i najvažnija karika u realizaciji neporecivosti!
- **Skladište sertifikata**.
  - U skladištu sertifikata se prave javni ključevi i sertifikati korisnika kao i liste poništenih sertifikata (engl. *certificate revocation list*, *CRL*).
  - Najčešće se realizuje pomoću servera sa LDAP kompatibilnim direktorijumom.

Infrastruktura javnih ključeva

- Sertifikat koji *CA* izdaje sadrži:
  - ime entiteta (ime osobe, naziv organizacije ili naziv servera),
  - javni ključ,
  - datum isticanja sertifikata,
  - *CA* organizacije koja je izdala sertifikat,
  - serijski broj,
  - druge informacije.
- Sertifikat je digitalno potpisan privatnim ključem *CA*.
  - Na taj način se uspostavlja veza između entiteta i para ključeva.
- Sertifikovani korisnici veruju izdavaču po pitanju verodostojnosti izdatih sertifikata.
- Primeri sistema koji zahtevaju upotrebu sigurnosnih mehanizama zasnovanih na *PKI*:
  - elektronska pošta,
  - razmena podataka u elektronskoj trgovini.

Infrastruktura javnih ključeva

- Osnovne funkcije *PKI* su izdavanje, osvežavanje, potvrda i opoziv sertifikata.
- **Izdavanje sertifikata**.
  - Sertifikati se najčešće izdaju na određeno vreme.
- **Osvežavanje sertifikata**.
  - Pošto sertifikati prestaju da važe posle određenog vremena, potrebno ih je „osvežavati“ ako ništa značajno nije izmenjeno u okruženju.
- **Potvrda sertifikata**.
  - Podaci u sertifikatu su podložni izmenama.
  - Korisnik želi da bude siguran u tačnost podataka što zahteva potvrdu sertifikata.
  - Postoje dva načina za potvrđivanje sertifikata, a *PKI* može da koristi oba:
    - *On-line* potvrda: korisnik zahteva potvrdu sertifikata direktno od *CA* svaki put kad mu je potrebna.
    - *Off-line* potvrda: *CA* izdaje vreme važenja sertifikata, tj. par datuma koji definišu period unutar kojeg se informacija sadržana u sertifikatu može smatrati validnom.

Infrastruktura javnih ključeva

- **Opoziv sertifikata** (engl. *revocation*).
  - Poništenje je proces objavljivanja u javnosti da je informacija u sertifikatu postala netačna.
  - Do opoziva može doći u slučaju:
    - Da je privatni ključ entiteta kompromitovan
    - Usled izmena podataka (npr. izmenjen broj telefona korisnika), što je češći razlog.
  - Ako se sertifikat potvrđuje *on-line* mogućnost *CA* može jednostavno dati do znanja da sertifikat više ne važi.
  - Ako se sertifikat potvrđuje *off-line* metod opoziva postaje kritičan, naročito u slučaju kompromitovanja privatnog ključa!
    - U nedostatku *on-line* pristupa najčešće se koristi **lista poništenih sertifikata** (CRL) sa poništenim sertifikatima koje je potpisao i izdao *CA*.
    - Prilikom potvrde sertifikata veoma je važno da korisnik pogleda poslednju verziju liste kako bi proverio da li je sertifikat koji namerava da koristi aktivan ili opozvan.

Infrastruktura javnih ključeva

- Probemi tipični za off-line metod opoziva.
- **Čekanje na novu *CRL***.
  - Period između trenutka u kom *CA* dobije informaciju o prestanku važenja određenog sertifikata i trenutka izdavanja nove liste.
  - Sve dok se opozvani sertifikat ne pojavi na listi, svaki korisnik koji proverava listu neće znati da je došlo do opoziva i prihvatiće sertifikat kao validan.
  - Problem se može rešiti uvođenjem inkrementalnih (delta) listi.
- **Veličina *CRL***.
  - Ako *CA* sertifikuje desetine ili stotine hiljada entiteta predvidivo je da će *CRL* biti velika.
  - Jedan od načina za rešenje problema veličine *CRL* je izdavanje različitih lista.
  - Na primer, *CA* može izdati:
    - Jednu listu za standardne opozive (na primer, izmena podataka o entitetu)
    - Drugu listu za slučajeve narušavanja sigurnosti.
  - Pažnja se u tom slučaju po potrebi može obratiti samo na sertifikate koji su opozvani zbog narušavanja sigurnosti.

Infrastruktura javnih ključeva

- Primer upotrebe *PKI* u hibridnom kriptosistemu.
  - U ovom primeru Alisa i Bob koriste serifikate koje je potpisao isti *CA*.
  - Primer ilustruje hibridni kriptosistem pogodan za efikasno šifrovanje i slanje većih poruka u sprezi sa *PKI*.
  - Postupci generisanja ključeva i sertifikacije:
    - Alisa i Bob generišu po jedan par ključeva.
    - Alisa i Bob prosleđuju svoje javne ključeve nazive i opisne informacije *RA*.
    - *RA* proverava njihove akreditive i prosleđuje zahtev za izdavanjem sertifikata ka *CA*.
    - *CA* generiše sertifikate i potpisuje ih svojim privatnim ključem.
    - Alisa i Bob razmenjuju javne ključeve i proveravaju ih na osnovu sertifikata koje preuzimaju sa *PKI* servera.

Infrastruktura javnih ključeva

- Primer upotrebe *PKI* u hibridnom kriptosistemu.
- Alisa šalje Bobu šifrovanu i potpisanu poruku m na sledeći način:
  - Alisa računa heš $h_1 = H(m)$ poruke m i potpisuje heš svojim privatnim ključem:
    - $s = S\,(H(m), k_{priv,alice})$
  - Alisa generiše tajni simetrični ključ k.
  - Alisa šifruje poruku m i potpisani heš s tajnim simetričnim ključem koji se koristi za komunikaciju:
    - $c = E\,(m|s, k) = E\,(m|S\,(H(m), k_{priv,alice}), k)$
  - Alisa formira **digitalni omot**, tj. šifruje tajni simetrični ključ Bobovim javnim ključem:
    - $w = E\,(k, k_{pub,bob})$
  - Alisa šalje Bobu šifrovanu poruku (sa potpisom) i digitalni omot $c|w$.

Infrastruktura javnih ključeva

- Primer upotrebe *PKI* u hibridnom kriptosistemu.
- Bob prima šifrovanu poruku i digitalni omot i radi sledeće:
  - Dešifruje omot svojim privatnim ključem i na taj način dobija tajni simetrični ključ:
    - $k = D\,(w, k_{priv,bob})$
  - Dešifruje poruku i potpisani heš koristeći tajni simetrični ključ koji je izračunao u prethodnom koraku:
    - $m \mid s = D(c, k)$
  - Računa heš poruke:
    - $h_2 = H(m)$
  - Proverava Alisin potpis pomoću njenog javnog ključa, čime dobija heš koji je Alisa potpisala:
    - $h_1 = V\,(s, k_{pub,alice})$
  - Upoređuje heš koji je on izračunao sa hešom koji je dobio proverom potpisa.
  - Ukoliko se heš vrednosti poklapaju, tj. $h_1 = h_2$, Bob je siguran u integritet poruke.

Osnovi kriptoanalize

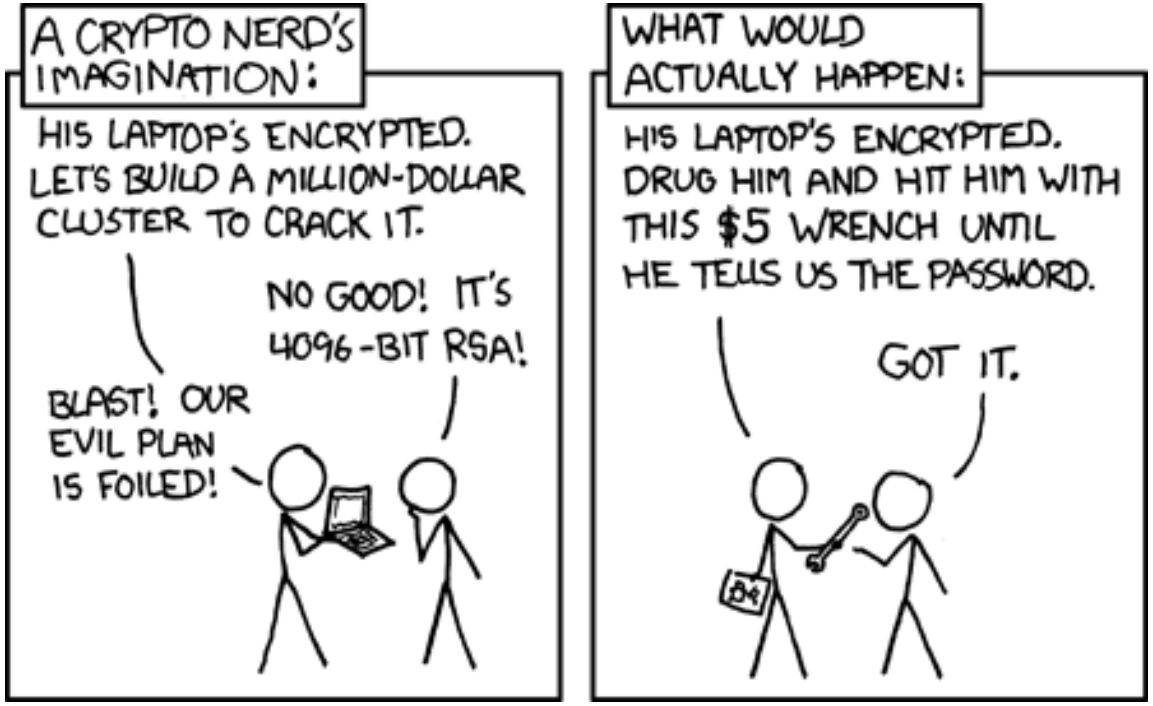
- Osnovna svojstva kriptoanalize: fascinantna oblast na granici umetnosti.
- Razlog leži u nekim fundamentalnim paradoksima računarstva i matematike.
- Edgar Alan Po: „Zagonetke kreirane ljudskim genijem, ljudski genije može i razrešiti.“
- U toku 5000 godina duge istorije kriptografije, najbolji umovi su bili angažovani na dizajniranju i analiziranju šifarskih sistema.
- Mnogo je lakše dizajnirati šifarski sistem, nego ga razbiti.
- Mnogo je lakše naći slabosti jednog šifarskog sistema, nego napraviti jak šifarski sistem.

Osnovi kriptoanalize

- Kriptoanaliza (William Friedman, 1920) je napad na šifrat s ciljem otkrivanje otvorenog teksta ili, još češće, ključa!
- Pretpostavke:
  - Shannon: kriptoanalitičar poznaje sistem.
  - Kerckhoffs: kriptoanalitičar poznaje sistem, ali ne poznaje ključ
    - Ova pretpostavka ne mora biti tačna ali se složenost procedure b.itno ne menja čak i ako kriptoanalitičar treba da proveri nekoliko mogućih kriptosistema.
  - Kriptosistem je pravilno implementiran i pravilno se upotrebljava.
    - Napad na kriptosistem se značajno olakšava u slučaju nepravilne implementacije ili upotrebe.
  - **Čovek je najslabiji faktor**! (ilustrovano slikom na sledećem slajdu)
- Sumarna pretpostavka: tajnost šifrata u potpunosti leži u ključu.

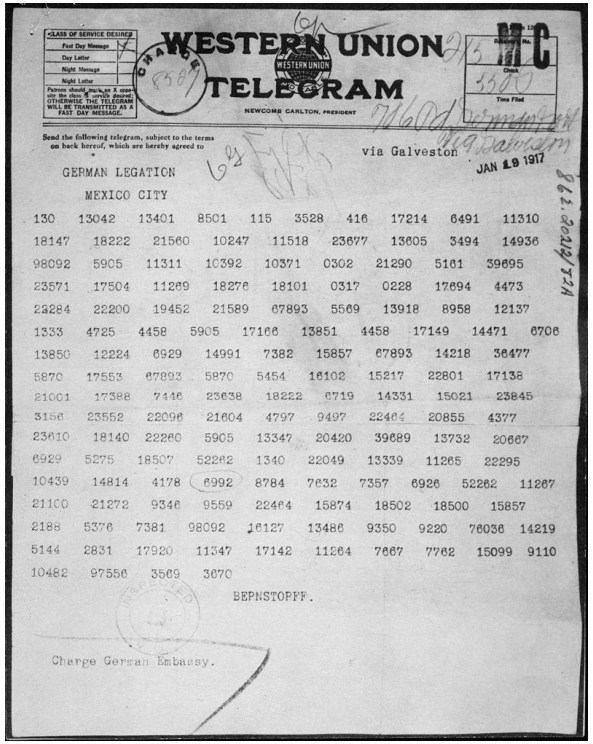
Osnovi kriptoanalize

- Realniji scenario.



Zimmermanov telegram

- *„In early 1917, … Germany also offered to help Mexico regain territories lost in the Mexican–American War in the Zimmermann Telegram.“*
- *„On December 7, 1917, the U.S. declared war on Austria-Hungary.“*



WESTERN UNION TELEGRAM

NEWCOMB CARLTON, PRESIDENT

Send the following telegram, subject to the terms on back hereof, which are hereby agreed to

via Galveston

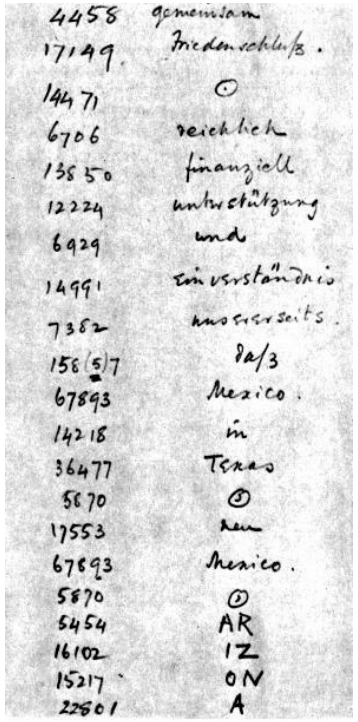
JAN 19 1917

GERMAN LEGATION
MEXICO CITY

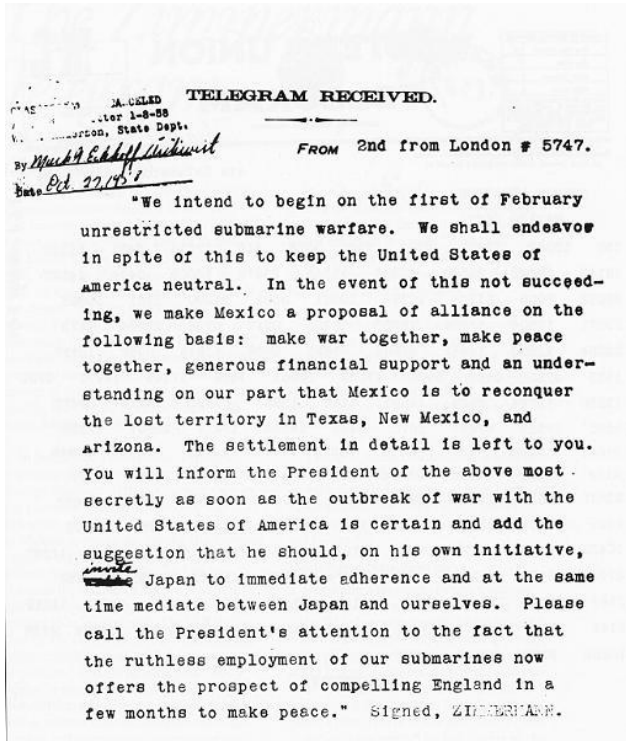
130 13042 13401 8501 115 3528 416 17214 6491 11310
18147 18222 21560 10247 11518 23677 13605 3494 14936
98092 5905 11311 10392 10371 0302 21290 5161 39695
23571 17504 11269 18276 18101 0317 0228 17694 4473
22284 22200 19452 21589 67893 5569 13918 8958 12137
1333 4725 4458 5905 17166 13851 4458 17149 14471 6706
13850 12224 6929 14991 7382 15857 67893 14218 36477
5870 17553 67893 5870 5454 16102 15217 22801 17138
21001 17388 7446 23638 18222 6719 14331 15021 23845
3156 23552 22096 21604 4797 9497 22464 20855 4377
23610 18140 22260 5905 13347 20420 39689 13732 20667
6929 5275 18507 52262 1340 22049 13339 11265 22295
10439 14814 4178 6992 8784 7632 7357 6926 52262 11267
21100 21272 9346 9559 22464 15874 18502 18500 15857
2188 5376 7381 98092 16127 13486 9350 9220 76036 14219
5144 2831 17920 11347 17142 11264 7667 7762 15099 9110
10482 97556 3569 3670

BERNSTORFF.

Charge German Embassy.



4458 gemeinsam
17149 Friedenschluß.
14471 ⊙
6706 reichlich
13850 finanziell
12224 unterstützung
6929 und
14991 Einverständnis
7382 unserseits.
15857 Da/3
67893 Mexico.
14218 in
36477 Texas
5870 ⊙
17553 neu
67893 Mexico.
5870 ⊙
5454 AR
16102 IZ
15217 ON
22801 A



TELEGRAM RECEIVED.

FROM 2nd from London # 5747.

"We intend to begin on the first of February unrestricted submarine warfare. We shall endeavor in spite of this to keep the United States of America neutral. In the event of this not succeeding, we make Mexico a proposal of alliance on the following basis: make war together, make peace together, generous financial support and an understanding on our part that Mexico is to reconquer the lost territory in Texas, New Mexico, and Arizona. The settlement in detail is left to you. You will inform the President of the above most secretly as soon as the outbreak of war with the United States of America is certain and add the suggestion that he should, on his own initiative, invite Japan to immediate adherence and at the same time mediate between Japan and ourselves. Please call the President's attention to the fact that the ruthless employment of our submarines now offers the prospect of compelling England in a few months to make peace." Signed, ZIMMERMANN.

Tipovi napada

- **Samo šifrat** (engl. *ciphertext-only attack*).
  - Kriptoanalitičar ima samo šifrate nekoliko poruka šifrovanih pomoću istog algoritma.
  - Najmoćniji kriptoanalitički napad, budući da zahteva samo pasivnog napadača u cilju dobijanja šifrata.
  - Znanje o otvorenom tekstu je minimalno i sastoji se od opštih znanja o statistici jednog jezika ili podjezika.
  - Primer uspešnog dešifrovanja ovom metodom su sve šifre proste zamene
- **Poznat otvoreni tekst** (engl. *known-plaintext attack*).
  - Kriptoanalitičar ima šifrat neke poruke i njemu odgovarajući otvoreni tekst.
  - Cilj je da se na osnovu ove informacije ili dodje do tajnog ključa ili izvršiti dešifrovanje ostatka ili dela šifrata.
  - Ovaj scenario je vrlo realan, budući da je teško sprečiti kriptoanalitičara da pretpostavi neke delove otvorenog teksta (metod verovatne reči, kontekst komuniciranja).

Tipovi napada

- **Odabran otvoreni tekst** (engl. *chosen-plaintext attack*).
  - Kriptoanalitičar je dobio privremeni pristup alatu za šifrovanje, tako da može dobiti šifrat odabranog otvorenog teksta.
  - U ovom scenariju se predpostavlja da kriptoanalitičar može da šifruje otvoreni tekst po sopstvenom izboru.
  - Realni scenariji:
    - Zarobljavanje šifarskog uredjeja sa napunjenim tajnim ključevima, do kojih se ne može doći fizički.
    - Slanje izabranog otvorenog teksta vlasniku šifarskog sistema, koji zatim šifruje istu poruku trećoj strani.
  - Oba scenarija zahtevaju aktivno učešće napadača i stoga je manje verovatan od prethodnih scenarija.
  - Primer: diferencijalna kriptoanaliza blokovskih šifarskih sistema.

Tipovi napada

- **Odabrani šifrat** (engl. *chosen-ciphertext attack*).
  - Kriptoanalitičar je dobio pristup alatu za dešifrovanje, tako da može dobiti otvoreni tekst odabranog šifrata.
  - Ovo je tipičan napad na kriptosisteme sa javnim ključem.
  - Ovaj scenario je vrlo sličan prethodnom, sa izabranim otvorenim tekstom, s tom razlikom što u njemu kriptoanalitičar ima mogućnost izbora šifrata za zadati šifarski sistem.
  - Realni scenario:
    - Zarobljavanje šifarskog uredjeja sa napunjenim tajnim ključevima, do kojih se ne može doći fizički, a koji može da radi u režimu dešifrovanja.

Tipovi napada

- **Prilagodljivi odabrani otvoreni tekst** (engl. *adaptive chosen-plaintext*).
  - U ovom slučaju napadač može izabrati sledeći otvoreni tekst na osnovu informacija koje je prikupio u prethodno opisanom načinu dešifrovanja.
  - Ovaj napada zahteva još veću aktivnost kriptoanalitičara, koji aktivno bira naredni otvoreni tekst na osnovu dobijenog rezultata šifrovanja prethodno izabranog otvorenog teksta.
  - U teorijskom smislu je od velikog značaja, budući da se ovaj tip napada može povezati sa oblašću mašinskog učenja.
    - Ako je P otvoreni tekst a $C=f(P, K)$ šifrat, kriptoanalitičar treba da nauči nepoznati parametar K, korišćenjem minimalnog broja upita funkcije $f(P, K)$.
    - Obučavajući skup su parovi (P_i, C_i), $i=1, ...n$.
    - Dobar je onaj šifarski sistem koji se ne može efektivno „naučiti“.
- **NAPOMENA**: izraz kriptoanaliza ponekad se odnosi na pokušaj zaobilaženja sigurnosti kriptografskih algoritama ili protokola, a ne samo kriptografske zaštite!

Lars Knudsen-ova klasifikacija rezultata kriptoanalitičkih napada

- Klasifikacija prema količini i kvalitetu otkrivenih tajnih informacija (od najboljeg rezultata ka najlošijem):
  - **Potpuno probijanje** (engl. *total break*).
    - Napadač je otkrio tajni ključ.
  - **Globalna dedukcija** (engl. *global deduction*).
    - Napadač je otkrio funkciju koja je ekvivalent algoritma za (de)šifrovanje, ali ne i ključ.
  - **Lokalna dedukcija** (engl. *instance or local deduction*).
    - Napadač je otkrio dodatne otvorene tekstove (ili šifrate) koji ranije nisu bili poznati.
  - **Informaciona dedukcija** (engl. *information deduction*).
    - Napadač dobija meru informacije sadržane u određenoj poruci o otvorenim tekstovima (ili šifratima) koji ranije nisu bili poznati.
  - **Algoritam koji omogućuje razlikovanje** (engl. *distinguishing algorithm*).
    - Napadač može razlikovati šifrat od slučajne permutacije.

Klasifikacija prema složenosti postupka

- Prema složenosti postupaka, kriptoanaliza se može klasifikovati u kategorije u odnosu na količinu resursa koje zahteva.
- Kategorije su određene količinom:
  - **Vremena**: broj potrebnih operacija.
  - **Memorije**: količina memorije potrebne za skladištenje podataka za vreme napada.
  - **Podataka**: količina potrebnih otvorenih tekstova i šifrata.
- Primer:
  - Ako imate sračunatih 2^{128} heševa, onda vam vreme nije problem, ali memorija jeste.
  - Ako nemate sračunate heševe, onda vreme jeste problem a memorija nije.
- Obično se pravi kompromis**:** *Time-Memory Trade-Off*.
  - Primer: *Rainbow* tabele.

Logika Time-Memory Trade-Off napada

- ***Time-Memory Trade-Off*** (*TMTO*) je je pokušaj postizanja balansa između količine upotrebljene memorije i vremena izvršavanja.
- *TMTO* je generalna tehnika čijom se primenom mogu poboljšati performanse mnogih vrsta algoritama.
- Suština *TMTO* je u tome da se:
  - Deo posla odradi unapred i samo jednom (*some one time work*)
  - Dobijeni rezultati obavljenog posla koriste svaki put kada se algoritam izvršava.
- *TMTO* podrazumeva pripremu podataka koje algoritam koristi pri svom izvršavanju, čime se značajno povećava njegova efikasnost.
- U opštem slučaju, veći unapred urađen posao zahteva veći inicijalni rad i veću memoriju, ali nakon toga sva računanja traju kraće.

Logika Time-Memory Trade-Off napada

- Na primer, neka je:
  - Vrednost x 32-bitni broj (*integer*)
  - cnt(x) funkcija koja određuje broj jedinica u binarnom zapisu broja x.
- Nejjednostavniji način za računanje funkcije cnt(x) je:

```
cnt (x)
   t = 0
   for i = 0 to 31
      t = t + ((x >> i) & 1)
   next i
   return t
end cnt
```

Logika Time-Memory Trade-Off napada

- Vrednost cnt se može izračunati na efikasniji način:
  - Unapred se izračuna cnt za svih 256 mogućih bajtova.
  - Sačuvaju se izračunate vrednosti u tabeli.
  - Vrednost x se deli na 4 bajta i za svaki bajt se iz tabele pročitaju odgovarajuće vrednosti.
  - Sabiraju se pročitane vrednosti i tako se izračunava cnt(x).
- Može se zapaziti da su pripremna izračunavanja (*pre-computation*) urađena samo jednom.
- Primenom *TMTO* metode izračunavanje svakog cnt(x) sada zahteva 4 koraka, a ne 32 kao u ranijem postupku.
- Realizacija:

```
table[i] = cnt(i) za i = 0,1,…,255 // generisanje tabele
cnt(x)
    p = table[ x & 0xff ] + table[ (x >> 8) & 0xff ] + table[ (x >> 16) & 0xff ] + table[ (x >> 24) & 0xff ]
return p
end cnt
```

Primer TMTO napada: Rainbow tabele

- Napadač koji pribavi heš lozinke može da pokuša da pronađe lozinku koja taj heš generiše.
- Najjednostavnija metoda za napad je metoda *brute force* (potpuna pretraga).
  - Za heš dužine 128 bita napadač mora da ispita najmanje 2^{128} vrednosti.
    - Za ovo je potrebno mnogo vremena.
- Realni primeri napada tipa *brute force* (uvode se pretpostavke):
  - Scenario 1. Pretpostavlja se da lozinka sadrži mala i velika slova i da je dužine 8 karaktera.
    - Napadač isprobava $62^8 \sim 2.2 \times 10^{14}$ lozinki.
  - Scenario 2. Pretpostavlja se da lozinka sadrži mala i velika slova i da je dužine 10 karaktera.
    - Napadač isprobava $62^{10} \sim 8.4 \times 10^{17}$ lozinki.
  - Pazite! Dodata su dva karaktera, a broj loziki koje treba ispitati se povećava 4000 puta!
- Prosečno vreme se računa kao polovina vremena potrebnog za potpunu pretragu.
  - Na osnovu ovih podataka se postavlja jasna granica izvodljivosti napada.
- Pitanje: šta se dešava ukoliko uračunamo mala i velika slova, brojeve i neke specijalne karaktere i pretpostavimo da je lozinka dužine 8-12 karaktera?

Primer TMTO napada: Rainbow tabele

- Alternativno, napadač može unapred da napravi tabelu sa svim mogućim lozinkama i njihovim heš vrednostima.
- Ovakva tabela se čuva na memorijskom medijumu, i prilikom napada se samo upoređuju heš vrednosti dok ne dođe do poklapanja.
  - Scenario 1. Lozinka dužine 6 karaktera (velika i mala slova) i MD5 heš (128 bita).
    - Veličina tabele je oko 850 GB (prihvatljivo, ali je i potpuna pretraga prihvatljiva).
  - Scenario 2. Lozinka dužine 8 karaktera.
    - Veličina tabele oko 3000TB! (neprihvatljivo).
- *Rainbow* tabele:
  - Metoda smanjenja veličine baze unapred izračunatih heševa ulančavanjem heš vrednosti.
  - Tipičan kompomis između veličine baze za pretragu i vremena pretrage.
  - Baza se smanjuje tako što se ne čuvaju sve heš vrednosti u lancu, već samo određene na osnovu kojih se može proveriti poklapanje nepoznatog heša i rekonstruisati nedostajuće.
  - Jednom kreirana tabela se može koristiti ponovo ukoliko heš nije posoljen.

Primer TMTO napada: Rainbow tabele

- Jednosmerna heš funkcija računa heš otvorenog teksta.
- **Redukciona funkcija** je jednosmerna funkcija koja generiše neki tekst na osnovu heša!
- NAPOMENE:
  - Heš je jednosmerna funkcija.
  - Redukciona funkcija ne daje početni otvoreni tekst od koga je nastao heš.
  - Ona generiše neki drugi tekst na osnovu heša.
- Redukciona funkcija služi za određivanje tipa karaktera i dužinu lozinki.
  - Pod pretpostavkom da lozinka ima šest karaktera (samo brojevi), može se predložiti jednostavna redukciona funkcija *R* koja uzima prvih šest brojeva iz heša.
  - *H* (493823) = **222**f**00**dc**4**b7f9131c89cff641d1a8c50
  - *R* (222f00dc4b7f9131c89cff641d1a8c50) = 222004

Primer TMTO napada: Rainbow tabele

- **Lanac heševa** je niz uređenih parova (lozinka, heš): (p_0, h_0) (p_1, h_1) (p_2, h_2) ... (p_n, h_n).
- Generisanje lanca izodi se na skedeći način:
  - Bira se lozinka (kandidat) i odgovarajuća redukciona funkcija ja shodno dužini i tipu znakova:
    - $h_0 = H(p_0)$
    - $p_1 = R(h_0)$
    - $h_1 = H(p_1)$
    - $p_2 = R(h_1)$
    - $h_2 = H(p_2)$
  - Funkcije R i H se naizmenično ponavljaju da bi se dobio lanac željene dužine.
- Ceo lanac se može u svakom trenutku rekonstruisati samo na osnovu početne vrednosti p_0 ako je poznata dužina lanca, heš algoritam i redukciona funkcija.

Primer TMTO napada: Rainbow tabele

- **Tabela lanaca**.
- Pretraživanje lanca na osnovu početne vrednosti nije efikasno.
- Zato se za svaki lanac čuva i poslednja vrednost (poslednji izračunati heš).
- Veliki broj ovakvih lanaca se može napraviti za različite početne vrednosti p_0.
- Tabela od N lanaca sadrži početnu vrednost (kandidat) i poslednji izračunati heš svakog lanca:
  - lanac 1. $p_{0,1}\ h_{n,1}$
  - lanac 2. $p_{0,2}\ h_{n,2}$
  - …
  - lanac N. $p_{0,N}\ h_{n,N}$

Primer TMTO napada: Rainbow tabele

- **Pretraga tabele**.
- Heš H (za koji tražimo odgovarajuću lozinku na osnovu koje je generisan) se najpre upoređuje sa krajnjim vrednostima u $h_{n,i}$ svakom lancu.
  - Ukoliko do poklapanja, lanac se rekonstruiše na osnovu početne vrednosti.
  - Lozinka je u ovom slučaj otvoreni tekst koji je generisao poslednji heš u lancu.
- Ukoliko ne dođe do poklapanja, ponavlja slična procedura kao pri generisanju lanca.
- Početna vrednost je heš H koji želimo da razbijemo.
  - $P_1 = R\ (H)$
  - $H_1 = H\ (P_1)$

Primer TMTO napada: Rainbow tabele

- **Pretraga tabele**.
- Ako bi se vrednost H_1 tražila u tabeli skraćenih lanaca, u slučaju poklapanja lanac može da se rekonstruiše.
- Kako se H_1 poklopila sa hešom na kraju lanca, i P_1 se poklapa sa poslednjim otvorenim tekstom iz lanca:
  - Početni lanac: (p_0, h_0) (p_1, h_1) ... (p_{n-1}, h_{n-1}) $(\boldsymbol{p_n}, \boldsymbol{h_n})$
  - Par koji se poklapa: $(\boldsymbol{P_1}, \boldsymbol{H_1})$
- To znači da se i vrednost nepoznatog heša H poklapa poklapa sa h_{n-1} u lancu:
  - Početni lanac: (p_0, h_0) (p_1, h_1) … $(\boldsymbol{p_{n-1}}, \boldsymbol{h_{n-1}})$ (p_n, h_n)
  - Par koji se poklapa: $(\boldsymbol{P}, \boldsymbol{H})$ (P_1, H_1)

Primer TMTO napada: Rainbow tabele

- **Pretraga tabele**.
- Drugim rečima, otvoreni tekst na osnovu kog je dobijen heš *H* u rekonstruisanom lancu se nalazi na poziciji *n*-1.
- Proces pretrage, redukcije i ponovnog hešovanja može da se ponavlja onoliko puta koliko je bilo redukcija u stvaranju lanaca, dok ne dođe do poklapanja vrednosti.
  - Ako je nad hešom urađeno *k* redukcija pre poklapanja, tražena vrednost se nalazi na p_{n-k} poziciji rekonstruisanog lanca.
  - Ukoliko uopšte ne dođe do preklapanja, u celoj tabeli ne postoji otvorenu tekst koji odgovara traženom hešu *H*.

Primer TMTO napada: Rainbow tabele

- **Kolizije i petlje**.
- Dva nedostatka opisane metode su kolizije i petlje lanaca.
- Do **kolizije** dolazi kada se u jednom lancu posle određenog broja redukcija dobije ista vrednost kao i u drugom – sa različitom početnom vrednošću.
  - Kolizije (preklapanja) se mogu otkriti ukoliko se prilikom kreiranja tabela čuvaju celokupni lanci.
- **Petlja** se u lancu formira ukoliko se prilikom kreiranja lanca pojavi sledeće:
  - $h_0 = H(p_0)$
  - $p_1 = R(h_0)$
  - Drugim rečima, heš otvorenog teksta dobijenog redukcionom funkcijom generiše isti heš!
  - Sve vrednosti posle petlje su iste i neupotrebljive!

Primer TMTO napada: Rainbow tabele

- Rešenje problema su ***Rainbow* tabele**:
- Prilikom kreiranja pojedinačnog lanca u svakom koraku se na heš primenjuje različita redukciona funkcija (R_1, R_2, R_3 ...).
  - $h_0 = H(p_0)$
  - $p_1 = R_1(h_0)$
  - $h_1 = H(p_1)$
  - $p_2 = R_2(h_1)$
  - $h_2 = H(p_2)$
  - $p_3 = R_3(h_2)$
  - $h_3 = H(p_3)$
- Kolizije (preklapanja) su kratke, a petlje su nemoguće!

Literatura

1. D. Pleskonjić, N. Maček, B. Đorđević, M. Carić (2007): Sigurnost računarskih sistema i mreža. Mikro knjiga, Beograd.
2. M. Veinović, S. Adamović (2013): Kriptologija 1. Univerzitet Singidunum, Beograd.
3. M. Milosavljević, S. Adamović (2014): Kriptologija 2. Univerzitet Singidunum, Beograd.
4. M. Stamp (2006): Information Security. John Wiley and Sons.
5. W. Stallings (2005): Cryptography and Network Security – Principles and Practice. Prentice Hall.
6. Predavanja prof. dr Milana Milosavljevića iz predmeta Kriptoanaliza 1 i Kriptoanaliza 2.

Hvala na pažnji

Pitanja su dobrodošla.