Osnovi kriptografije

Algoritmi

Bezbednost

- U bitnim programskim sistemima podaci i servisi se moraju **očuvati od bezbednosnih pretnji**.
- Bezbednost u računarskim sistemima je usko povezana sa pojmom **oslonjivosti** (*dependability*).
  - Jer očekujemo da sistem ispuni "obećanje" – da isporuči servis
  - **Tako da se možemo osloniti na njega**

  Između ostalog, oslonjivost uključuje:
  - **Tajnost** (*confidentiality*) – informacija se daje samo ovlašćenim (autorizovanim) licima/servisima,
  - **Integritet** (*integrity*) – modifikacija podataka zahteva autorizovan pristup.

Tipovi bezbednosnih pretnji

- Tipovi pretnji na nivou mreže:
  - **Presretanje** (*interception*) – neautorizovana stranka dobije pristup podacima ili servisima. Ovde spada i ilegalno kopiranje podataka.
  - **Prekidanje** (*interruption*) – servis ili podatak postane nedostupan, neupotrebljiv ili uništen. Npr. *denial of service* (DoS)
  - **Modifikacija** (*modification*) – neautorizovana izmena podataka ili izmena servisa, koji više nije u skladu sa specifikacijom
  - **Fabrikacija** (*fabrication*) – generisanje dodatnih podataka ili aktivnosti koji ne bi postojali u normalnim situacijama. Npr. ilegalno ponavljanje ranijeg zahteva za prenos novca ili dodavanja zapisa u datoteku sa lozinkama
- Malver (malware - “**Mal**icious Soft**ware**”) je pretnja

Mehanizmi sprovođenja bezbednosti

- Za bezbedan sistem potrebno je definisati bezbednosne zahteve.
  - **Bezbednosna politika** (*security policy*) sistema tačno opisuje šta je korisnicima, servisima i računarima u sistemu dozvoljeno, a šta nije.

- **Šifrovanje** (*encryption*) – transformiše podatke u format koji nije razumljiv za napadače.
- **Autentifikacija** (*authentication*) – verifikacija identiteta korisnika, klijenata, servisa, itd.
- **Autorizacija** (*authorization*) – provera da li korisnik ili servis ima pravo na izvršavanje određene akcije.
- **Beleženje istorijata aktivnosti** (*auditing*) – upisivanje u dnevnike događaja (uglavnom tekstualne) koji je entitet čemu pristupio i na koji način.
  - Ovaj mehanizam ne obezbeđuje direktnu zaštitu od napada, ali omogućava naknadnu analizu bezbednosnih problema.

Šta je šifrovanje?

- Šifrovanje je transformacija informacije tako da je njeno pravo značenje skriveno.
  - Potrebno je „posebno znanje“ da se preuzme informacija.
- Šifrovanje koristi tajne ključeve kao „posebno znanje“.
  - Savremeni algoritmi sprečavaju pokušaje otkrivanja ključa grubom silom!

Primer:

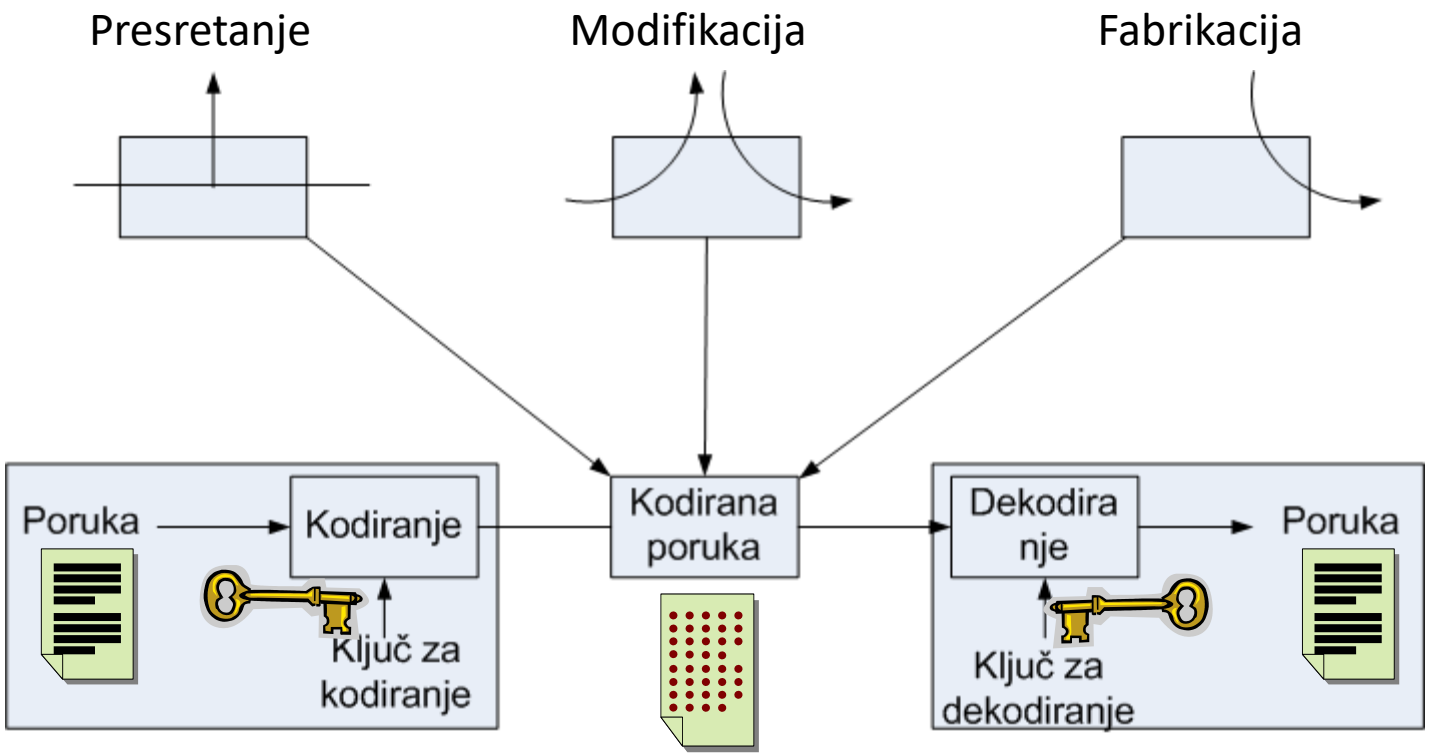
- Ako strana A želi da pošalje poruku *m* strani B, onda da bi zaštitila poruku od bezbednosnih pretnji
  1. A će izvršiti šifrovanje (enkripciju) poruke *m* (dobija se *m*’)
  2. A će poslati šifrovanu poruku (*m*’)
  3. B će izvršiti dešifrovanje (dekripciju) poruke (*m*’ i dobiće *m*)

Osnovni pojmovi kriptografije

- Definicije
  - **poruka** (P) – podatak ili deo podatka koji strana A šalje strani B.
  - **šifrovana poruka** (C) – poruka koja je promenjena sa ciljem da bude nerazumljiva za potencijalne napadače.
  - **ključ** (K) – tajni podatak koji omogućava stranama da poruke šifriraju.
  - **šifrovanje** (**enkripcija**) (E) – proces u kojem predajna strana modifikuje poruku sa ciljem da ona bude nerazumljiva za potencijalne napadače. Rezultat ovog procesa je *šifrovana poruka.*
  - **dešifrovanje** (**dekripcija**) (D) – proces u kojem se od šifrovane poruke pravi izvorna poruka.

Šifrovanje / dešifrovanje

- Šifrovana poruka treba da spreči akcije uljeza (pretnje bezbednosti): presretanje, modifikaciju i fabrikaciju.
- Dešifrovanje bez posedovanja ključa (treba da) je JAKO otežano!



$$C = E_k(P)$$

$$P = D_k(C)$$

Primitivan algoritam šifrovanja

- Jednostavna zamena slova je način kodiranja teksta gde se neko slovo zamenjuje drugim, a dekodiranje ponovi zamenu.

A	B	C	D	E	F	G	H	I	J	K	L	M
0	1	2	3	4	5	6	7	8	9	10	11	12
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

ASCII kodovi

Binary	Oct	Dec	Hex	Glyph
010 0000	040	32	20	space
010 0001	041	33	21	!
010 0010	042	34	22	"
010 0011	043	35	23	#
010 0100	044	36	24	$
010 0101	045	37	25	%
010 0110	046	38	26	&
010 0111	047	39	27	'
010 1000	050	40	28	(
010 1001	051	41	29	)
010 1010	052	42	2A	*
010 1011	053	43	2B	+
010 1100	054	44	2C	,
010 1101	055	45	2D	-
010 1110	056	46	2E	.
010 1111	057	47	2F	/
011 0000	060	48	30	0
011 0001	061	49	31	1
011 0010	062	50	32	2
011 0011	063	51	33	3
011 0100	064	52	34	4
011 0101	065	53	35	5
011 0110	066	54	36	6
011 0111	067	55	37	7
011 1000	070	56	38	8
011 1001	071	57	39	9
011 1010	072	58	3A	:
011 1011	073	59	3B	;
011 1100	074	60	3C	<
011 1101	075	61	3D	=
011 1110	076	62	3E	>
011 1111	077	63	3F	?

Binary	Oct	Dec	Hex	Glyph
100 0000	100	64	40	@
100 0001	101	65	41	A
100 0010	102	66	42	B
100 0011	103	67	43	C
100 0100	104	68	44	D
100 0101	105	69	45	E
100 0110	106	70	46	F
100 0111	107	71	47	G
100 1000	110	72	48	H
100 1001	111	73	49	I
100 1010	112	74	4A	J
100 1011	113	75	4B	K
100 1100	114	76	4C	L
100 1101	115	77	4D	M
100 1110	116	78	4E	N
100 1111	117	79	4F	O
101 0000	120	80	50	P
101 0001	121	81	51	Q
101 0010	122	82	52	R
101 0011	123	83	53	S
101 0100	124	84	54	T
101 0101	125	85	55	U
101 0110	126	86	56	V
101 0111	127	87	57	W
101 1000	130	88	58	X
101 1001	131	89	59	Y
101 1010	132	90	5A	Z
101 1011	133	91	5B	[
101 1100	134	92	5C	\
101 1101	135	93	5D	]
101 1110	136	94	5E	^
101 1111	137	95	5F	_

Binary	Oct	Dec	Hex	Glyph
110 0000	140	96	60	`
110 0001	141	97	61	a
110 0010	142	98	62	b
110 0011	143	99	63	c
110 0100	144	100	64	d
110 0101	145	101	65	e
110 0110	146	102	66	f
110 0111	147	103	67	g
110 1000	150	104	68	h
110 1001	151	105	69	i
110 1010	152	106	6A	j
110 1011	153	107	6B	k
110 1100	154	108	6C	l
110 1101	155	109	6D	m
110 1110	156	110	6E	n
110 1111	157	111	6F	o
111 0000	160	112	70	p
111 0001	161	113	71	q
111 0010	162	114	72	r
111 0011	163	115	73	s
111 0100	164	116	74	t
111 0101	165	117	75	u
111 0110	166	118	76	v
111 0111	167	119	77	w
111 1000	170	120	78	x
111 1001	171	121	79	y
111 1010	172	122	7A	z
111 1011	173	123	7B	{
111 1100	174	124	7C	|
111 1101	175	125	7D	}
111 1110	176	126	7E	~

Primer

```
function out = ShiftCipher(in, shift)
for k = 1 : length(in)
    s = in(k);
    if s >= 'a' && s <= 'z'
        s = mod(s - 'a' + shift, 26) + 'a';
    elseif s >= 'A' && s <= 'Z'
        s = mod(s - 'A' + shift, 26) + 'A';
    else
    end
    out(k) = char(s);
end
```

A	B	C	D	E	F	G	H	I	J	K	L	M
0	1	2	3	4	5	6	7	8	9	10	11	12
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

```
>> p = 'Danas je lep dan. Za 3 dana se prognozira sneg!'
p =
Danas je lep dan. Za 3 dana se prognozira sneg!

>> c = ShiftCipher(p,4)
c =
Herew ni pit her. De 3 here wi tvskrsdmve wrik!

>> p2 = ShiftCipher(c,-4)
p2 =
Danas je lep dan. Za 3 dana se prognozira sneg!
```

Neke druge tehnike ...

- Tekstualne poruke nisu jedine u računarskoj komunikaciji, i kodiranje zamenom slova nije dovoljno sigurno.
  - Ako se koristi ShiftCipher dovoljno je probati 26 slučajeva!

- Kriptografija koristi binarnu predstavu poruka.
- Često se u algoritmima kodiranja koristi ekskluzivno ILI, tj. binarna operacija (XOR)

Tablica istinitosti za XOR:

$0 \oplus 0 = 0$
$0 \oplus 1 = 1$
$1 \oplus 0 = 1$
$1 \oplus 1 = 0$

Osobine (x, y su biti):

$x \oplus 0 = x$
$x \oplus 1 = \sim x$
$(x \oplus y) \oplus y = x$

„Binarna podloga“ - *One-Time Pad* algoritam

- Posmatramo niz bita koji čine poruku p.
- „Binarna podloga“ k je ključ – slučajno izabran niz bita iste dužine kao poruka.
- Poruku „postavimo na podlogu“ i dobijamo kodiranu poruku: $c = p \oplus k$
  - Operacija XOR se primenjuje na svaki par bita k_i i p_i
  - Kodirana poruka je u obliku gde se teško može rekonstruisati neka dodatna informacija o originalnoj poruci.
- Prijemna strana takođe poseduje isti ključ i poruku postavlja na podlogu: $p_2 = c \oplus k = (p \oplus k) \oplus k = p$

- Osobine algoritma: čini kodiranu poruku bezbednom, koristi ključ dužine poruke, i ponovna upotreba ključa je rizična.
  - Primernom XOR operacije na kodirane poruke c_1 i c_2 dobija se $(c_1 \oplus k) \oplus (c_2 \oplus k) = c_1 \oplus c_2$, tako da se uticaj ključa gubi i otkrivaju se mesta gde poruke c_1 i c_2 imaju iste bite.

Primer

- Poruka: ONETIMEPAD
- Ključ: TBFRGFARFM
- Šifrovana poruka: IPKLPSFHGQ

- Dešifrovanje:
  - Ključ TBFRGFARFM dešifruje poruku u ONETIMEPAD
  - Ključ POYYAEAAZX dešifruje poruku u SALMONEGGS
  - Ključ BXFGBMTMXM dešifruje poruku u GREENFLUID

Rad sa blokovima i ulančavanje

- Prema prethodnom algoritmu za dugačku poruku je potreban dugačak ključ (što je nezgrapno!)
- Stoga se kod simetrične kriptografije koriste:
  1. Kraći ključevi, i
  2. Poruke se „seckaju“ na blokove bita
     (ključ se primenjuje na svaki od blokova)
- Dodatno, „ulančavanje blokova“ je tehnika koja sprečava da isti blokovi daju isto šifrovan rezultat.
  - Obrada narednog bloka koristi rezultat obrade prethodnog bloka

- Praktični algoritmi su dosta složeniji u odnosu na prethodni algoritam ...

Šifrovanja pomoću ključeva

- Šifrovanje (kodovanje) $C = E_k(P)$
  - Generiše kodiranu poruku na osnovu originalne poruke i tajnog ključa
- Dešifrovanje (dekodovanje) $P = D_k(C)$
  - Generiše originalnu poruku na osnovu kodirane poruke i tajnog ključa

- Važne osobine:
  - Nemoguće je naći ključ K ako su poznate poruka P i kodirana poruka C
  - Nemoguće je naći drugi ključ K' za koji bi bilo $E_K(P)=E_{K'}(P)$

- Podele algoritama šifrovanja
  - Simetrična kriptografija
  - Asimetrična kriptografija

Veliki brojevi

- Upotreba stvarno velikih brojeva
  - 1 od 2^{61} šansi da neko osvoji loto i da ga udari grom u istom danu
  - 2^{92} atoma je u prosečnom ljudskom telu
  - 2^{128} mogućih vrednosti u 128-bitnom ključu
  - 2^{170} atoma na planeti
  - 2^{190} atoma na Suncu
  - 2^{233} atoma u galaksiji
  - 2^{256} mogućih vrednosti u 256-bitnom ključu

Da li možemo „silom“ dešifrovati poruku?

- Termodinamička ograničenja ...
  - Iz fizike: energija potrebna da se postavi vrednost bita (0 ili 1) je ne manja od kT, gde je:
    - k Bolcmanova konstanta ($1.38*10^{-23}$ J/K)
    - T je apsolutna temperatura sistema
  - Ako je $T = 3.2K$ (ambijentalna temperatura u svemiru)
    - kT = $4.4*10^{-23}$ J
  - U godini dana Sunce emituje $1.21*10^{34}$ J
    - Dovoljno da provrtimo 187-bitni brojač
  - Akumulacija energije Sunca u 32 godine je dovoljna da provrtimo 192-bitni brojač
  - Supernova proizvodi 10^{44} J u okruženju
    - Dovoljno da provrtimo 219-bitni brojač
    - 256-bitni brojač zahteva 2^{37} = 137.438.953.472 puta više energije!!!

Simetrična kriptografija

- Drugo ime: kriptografski sistemi sa **deljenim ključem**

- Koristi isti ključ za šifrovanje i dešifrovanje
- Da bi komunikacija bila bezbedna, ključevi moraju biti tajni (kao i kod svih kriptografskih sistema)
- Oznaka za deljeni ključ: $K_{A,B}$

$$C = E_{K_{A,B}}(P) \quad \leftarrow \text{Na predajnoj strani}$$

$$P = D_{K_{A,B}}(C) \quad \leftarrow \text{Na prijemnoj strani}$$

AES

- AES (*Advanced Encryption Standard*) je javno dostupan i besplatan algoritam šifrovanja simetričnim ključem.
  - koristi ključ od 128, 192 ili 256 bita, a blokove dužine 128 bita.
- Nastao 2001. nakon 4-godišnjeg takmičenja i saradnje: američke vlade, privatne industrije i naučnika za izbor najboljeg algoritma koji je:
  - Bezbedan – otporan na kriptoanalizu, ispravnosti matematike, slučajnosti izlaza, itd.
  - Jeftin – brz i ima male memorijske zahteve
  - Dobrih karakteristika – jednostavan, fleksibilan, pogodan za hardversku i softversku implementaciju.
- Zamenio 3DES algoritam

Asimetrična kriptografija

- Odvojeni ključevi za šifrovanje i dešifrovanje
  - Svaki od učesnika u komuniciji ima dva ključa: javni ključ i tajni ključ.
  - Tajni ključ K_D (zove se i privatan ključ) se nikome ne daje.
  - Javni ključ K_E može posedovati svako.
- Ovakvi sistemi zovu i “sistemi sa javnim ključevima”

- Princip rada:
  - Proces šifriranja koristi javni ključ K_E da napravi šifrovanu poruku
  - Takvu poruku može da „pročita“ samo proces dešifrovanja koji koristi tajni ključ.

$$P = D_{K_D}(E_{K_E}(P))$$

Računanje po modulu

- Primer

  (23 + 7) mod 11 = 30 mod 11 = (2*11 + 8) mod 11 = 8

- Koristi se u nekim algoritmima šifrovanja
- Osobine:
  - $(a + b) \bmod n = ((a \bmod n) + (b \bmod n)) \bmod n$
  - $ab \bmod n = ((a \bmod n)(b \bmod n)) \bmod n$
  - $a^b \bmod n = (a \bmod n)^b \bmod n$
  - $xn \bmod n = 0$

Prosti brojevi

- Prosti brojevi su prirodni brojevi deljivi samo sa 1 i samim sobom.
- Primer: 1, 2, 3, 5, 7, 11, 13, ...
  - Npr. 6 nije prost broj jer je $6 = 2 \cdot 3$

- Generisanje velikog prostog broja treba da proveri da li je slučajno izabran broj prost.
  - Postoje algoritmi koji ovo testiraju

RSA

- RSA algoritam šifrovanja asimetričnim ključem
- Algoritam se oslanja na broj koji je proizvod 2 velika prosta broja
  - Dovoljno je velik da niko ne može otkriti činioce u razumno dugom vremenskom periodu.
  - Npr. broj sa 2048 bita je dovoljno velik.

- Naziv je dobio po tvorcima: Ronald Rivest, Adi Shamir i Leonard Adelman.

RSA algoritam

1. Izabrati 2 velika (najmanje 1024 bita svaki) različita prosta broja p i q
2. Izračunati $n = pq$
3. Izračunati $r = (p-1)(q-1)$
4. Izabrati mali broj e tako da su e i r uzajamno prosti (nemaju zajedničke delioce, sem 1)
5. Izračunati d kao inverzan element za množenje, tj. da je $ed \bmod r = 1$
6. RSA javni ključ je tada par $K_E = (e, n)$
7. RSA privatni ključ je $K_D = (d, n)$
8. Funkcije šifrovanja i dešifrovanja su:

$$E_{K_E}(m) = c = m^e \bmod n$$

$$D_{K_D}(c) = c^d \bmod n$$

Primer (sa malim brojevima)

1. Izabrati: $p = 17$ i $q = 29$
2. Izračunati $n = pq = 493$
3. Izračunati $r = (p-1)(q-1) = 448$
4. Izabrati mali broj $e = 5$ jer su 5 i 448 uzajamno prosti.
5. Izračunati $d = 269$
   provera: $ed \bmod r = (5 \cdot 269 \bmod 448) = 1345 \bmod 448 = (3 \cdot 448 + 1) \bmod 448 = 1$
6. RSA javni ključ je $K_E = (5, 493)$
7. RSA privatni ključ je $K_D = (269, 493)$
8. Primer šifrovanja i dešifrovanja:
   $E_{K_E}(327) = 327^5 \bmod 493 = 3.738.856.210.407 \bmod 493 = 259$
   $D_{K_D}(259) = 259^{269} \bmod 493 = \cdots = 327$

Detalji RSA algoritma

- Kako računati sa velikim brojevima (sa velikim brojem cifara)?
- Kako pronaći veliki prost broj u kratkom vremenu?
- Kako izračunati e tako da bude uzajamno prost sa r?
- Kako sračunati d da zadovolji formulu $ed \bmod r = 1$?
- Ako je d velik broj kako računati x^d u kratkom vremenu?
- Kako znamo da su $E_{K_E}(x)$ i $D_{K_D}(x)$ inverzne funkcije?

Aritmetika velikih brojeva

- Brojevi koji se ovde pominju ne mogu stati u 64-bitne registre.
- Srećom, postoje biblioteke koje vrše takve proračune
  - npr. Python programski jezik podržava aritmetiku velikih brojeva
- Dodatno, aritmetika u RSA koristi račun po modulu tako da se međurezultati mogu ograničiti.
  - npr. $x^d \bmod n$ može da ima međurezultate u intervalu $[0, n-1]$.

Generisanje velikih prostih brojeva

- Generišemo slučajan neparan veliki broj i proverimo da li je prost.
  - npr. Miller-Rabin test se koristi za proveru da li je broj prost
- Veliki prosti brojevi nisu retki
  - Teorema prostih brojeva: šansa da veliki broj m bude prost broj iznosi 1 u $\ln m$.
  - Npr. za 1024 bitni broj treba probati oko 710 brojeva ($\ln 2^{1024} = 710$).

Uzajamno prosti brojevi

- Dva prirodna broja su uzajamno prosti ako nemaju zajednički delilac veći od 1.
- Euklidov algoritam nalazi najveći zajednički delilac:
  - U osnovi, algoritam za brojeve e i r nalazi najveći zajednički delilac g tako što računa $g = ei + rj$, gde su i i j celi brojevi i jedan od njih može biti negativan.
    Primer: zajednički delilac za 30 i 18 je 6.
    $6 = 30i + 18j, i = -1, j = 2.$
  - Složenost algoritma je $O(\log_2 e)$
- Algoritam se primenjuje za nalaženje e tako što proba redom sa prostim brojevima koji su manji od r
  - Ima oko $r/\ln r$ prostih brojeva manjih od r, ali r ima najviše $\log_2 r$ faktora tako da su solidne šanse da se e brzo pronađe.
- Takođe, rezultat Euklidovog algoritma se koristi za računanje d.

Brzo računanje celobrojnog stepena nekog broja

- Brzi algoritam računa izraz $x^d \bmod n$ u $\Theta(\log_2 d)$ vremenu upotrebom tehnike ponavljanja stepenovanja
  - za d parno $x^d = \left(x^{\frac{d}{2}}\right)^2$, a
  - za d neparno $x^d = \left(x^{\frac{d-1}{2}}\right)^2 \cdot x$

Hibridni sistemi šifrovanja

- Primena RSA algoritma na duge poruke zahteva dosta vremena i tada je nepraktična.
- Hibridno rešenje predlaže da strane koje su u komunikaciji inicijalno razmene ključ koji će se koristi za simetrično šifrovanje i dešifrovanje poruka (npr. AES algoritam). Za inicijalno slanje takvog ključa se može upotrebiti algoritam za asimetrično šifrovanje i dešifrovanje poruka (npr. RSA algoritam). Znači:
  - prva poruka sadrži simetričan ključ šifrovan RSA algoritmom, a
  - ostale poruke se šifruju AES algoritmom.

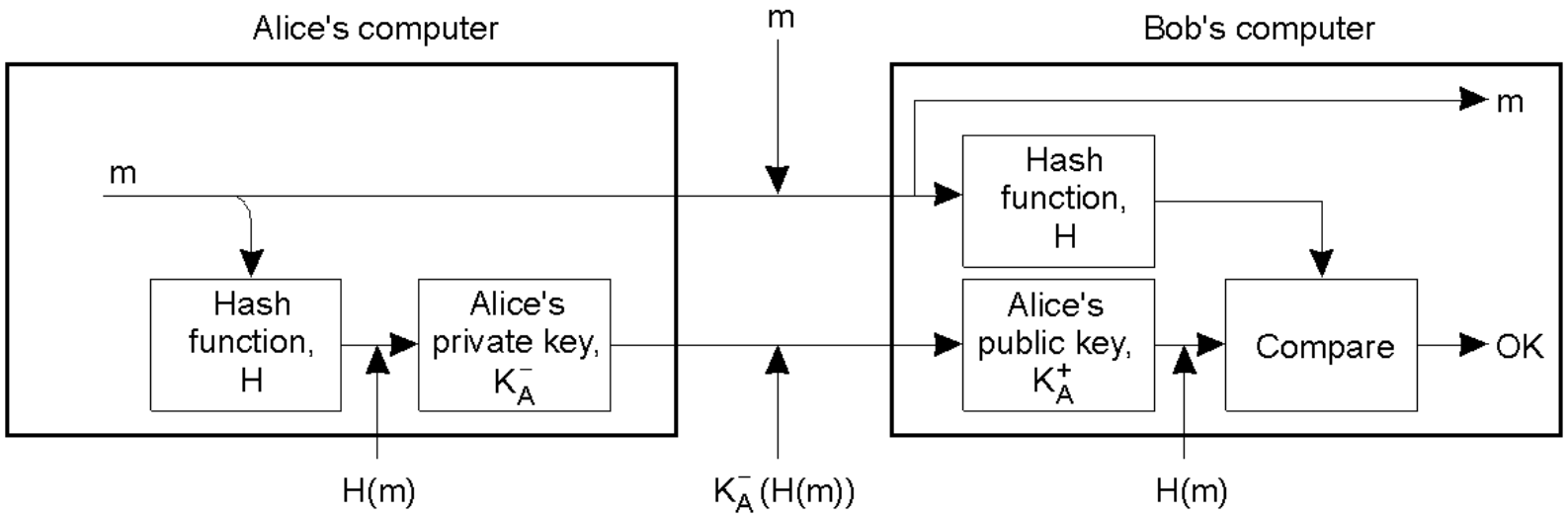
Hash funkcije

- Služe za "žigosanje poruke" – prave žig na osnovu poruke
- Hash funkcija *H* je jednosmerna funkcija
  - na osnovu žiga se ne može rekonstruisati poruka
- Važne osobine
  - Slaba otpornost na koliziju – za poruku *m* se ne može naći poruka *m*' za koju je *H*(m)=*H*(*m*')
  - Jaka otpornost na koliziju – nemoguće naći dve poruke *m* i *m*' za koje će biti *H*(*m*)=*H*(*m*')

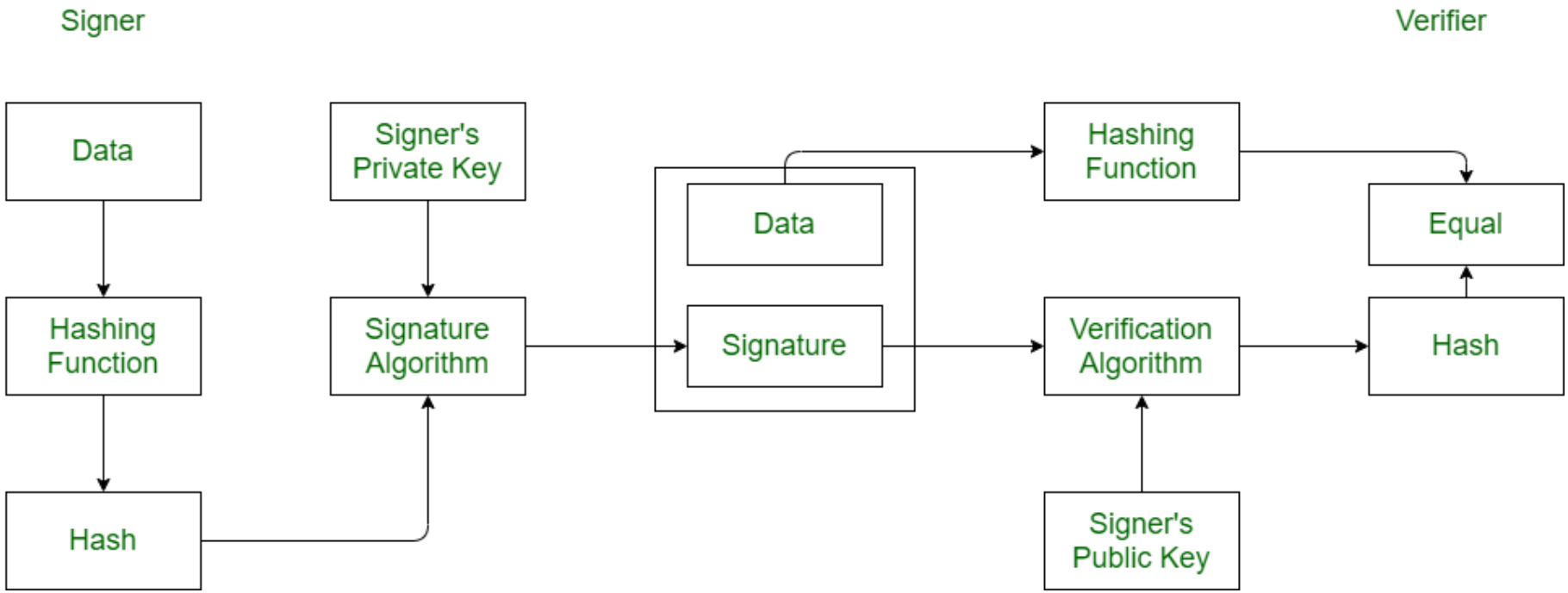
- Primer: MD5 (*Message Digest* 5)
  - MD5 (*Message Digest* 5) – staro/prevaziđeno
    - od poruke proizvoljne dužine pravi jedinstveni 128-bitni *message digest*
  - SHA-256 – skoriji, dobar izbor
  - ...

Digitalni potpis *hash*-om

- Koraci su sledeći:
  - A napravi $h=H(m)$ gde je H hash funkcija, m poruka a h *message digest*
  - A kodira h svojim privatnim ključem
  - A pošalje i m i h (h je malo u odnosu na m)
  - B primi m i h
  - B napravi $h_B=H(m)$
  - B dekodira h (javnim ključem od A) i uporedi ga sa h_B (treba da budu jednaki)



DSA algoritam (Digital Signature Algorithm)



DSA algoritam - parametri

p – prost broj duzine L bitova (L je u interval 512,1024)

q – 160-bitni prost faktor od p-1

g = $h^{((p-1)/q) \bmod(p)}$ gde je h bilo koji factor manji od p-1 takav da je $h^{((p-1)/q) \bmod(p)}$ vece od 1

x – broj manji od q

y = $g^x \bmod(p)$

h=H(m) je jednosmerna hes funkcija Secure Hash Algorithm – ovo je odredjena standardom

Parametri p, q i g su javni I mogu biti dostupni svima u mrezi

Privatni kljuc je x, a javni y.

DSA algoritam - koraci

1. A generiše slučajan broj k manji od q
2. A generiše

$r=(g^k \bmod(p)) \bmod(q)$

$s=(k^{-1}*(H(m)+x*r)) \bmod(q)$

Parametri r i s su potpis od A i oni se šalju B

3. B verifikuje potpis tako sto izracunava

$w=s^{-1}\bmod(q)$

$u1=(H(m)*w)\bmod(q)$

$u2=(r*w)\bmod(q)$

$v=((g^{u1}*y^{u2})\bmod(p))\bmod(q)$

Ako je v=r, tada je potpis ispravan

Primeri

DSA algoritam - primer

```
q = 11          # prost broj
p = 23          # p- prost broj: (p-1) mod q = 0
h = 3           # has vrednost poruke

g = 9 (not 4)# računa se : 1 < g < p, g**q mod p = 1:
             #           i g = h**((p-1)/q) mod p
             #           3**((23-1)/11) mod 23 = 9
             #           9**11 mod 23 = 1

x = 7           # bira se: 0 < x < q
y = 4           # računa se: y = g**x mod p = 9**7 mod 23 = 4,782,969 mod23
{23,11,9,4}   # public key: {p,q,g,y}
{23,11,9,7}   # private key: {p,q,g,x}
```

DSA algoritam – Alisa generiše

```
k = 5       # generiše: 0 < k < q
r = 8       # računa: r = (g**k mod p) mod q = (9**5 mod 23) mod 11
i = 9       # računa se: k*i mod q = 1: 5*i mod 11 = 1
s = 3       # računa se: s = i*(h+r*x) mod q = 9*(3+8*7) mod 11
{8,3}       # digitalni potpis: {r,s}
```

DSA algoritam – Bob izračunava

(r,s) =(8,3) - digitalni potpis

```
h = 3       # hash vrednost poruke
w = 4       # računa: s*w mod q = 1 ( 3*4 mod 11 = 1)
u1 = 1      # računa: u1 = h*w mod q = 3*4 mod 11 = 1
u2 = 10     # računa: u2 = r*w mod q = 8*4 mod 11 = 10
```

PROVERA POTPISA

```
v = 8       # računa: v = (((g**u1)*(y**u2)) mod p) mod q
            #         = (((9**1)*(4**10)) mod 23) mod 11 = 8

v == r      # verification passed
```