

Avatud võtmeaga krüpteerimine.  
Asümmeetrilised krüptosüsteemid. RSA

Erika Matsak, PhD

# Sissejuhatus

- Asümmeetriliste algoritmide loomine on suurim krüptograafiline saavutus.
- Sümmeetriliste algoritmide juures oli nõrgaks kohas ühine salavõti, mida oli vaja ka “päris salaja” edastada isikule, kellele saadetak sõnum oli mõeldud.
- Teine oluline küsimus – kuidas teha kindlaks, et see isik, kellega oleme kirjavahetuses on ka päriselt see isik. Vajadus digitaalselt allkirjastada.

# Asümmeetrilised algoritmid

- Asümmeetrilised algoritmid = avaliku võtmega algoritmid
- Avalik võti krüpteerimise jaoks (KU)
- Privaatvõti dekrüpteerimise jaoks (KR)
- Ei ole võimalik arvutada privaatvõtit teades avaliku võtit ja algoritmi ülesehitust.
- Lähtume sellest, et kõik grupi liikmed teavad avalikku võtit, aga privaatvõtmed luuakse lokaalselt ning neid ei jagata omavahel.
- Igal hetkel saab grupi liige muuta oma privaatvõtit ning avalikustada paarilise avaliku võtme teistele grupi liikmetele.

# Asümmeetrilised algoritmid. Nõudmised

- Whitfield Diffie ja Martin Hellman on esitanud nõudmisi, millele peaks vastama avaliku võtmega algoritm:
  - On arvutuslikult lihtne moodustada paari: {avalik võti, privaatvõti}
  - On arvutuslikult lihtne, omades avalikku võtit KU ning avateksti M moodustada vastavat krüptogrammi:  $C = E_{KU}[M]$
  - On arvutuslikult lihtne dešifreerida tekst, omades privaatvõtit:  $M = D_{KR}[C] = D_{KR}[E_{KU}[M]]$
  - On arvutuslikult mittevõimalik tuletada privaatvõtit KR, teades avalikku võtit KU
  - On arvutuslikult mittevõimalik taastada avateksti M, teades avalikku võtit KU ning krüptogrammi
  - Lisatingimus: šifreerimis- ja dešifreerimisfunktsioone saab rakendada suvalises järjekorras:  $M = E_{KU}[D_{KR}[M]]$



Whitfield Diffie ja Martin Hellman: esimese avaliku võtmega krüptograafia alase publikatsiooni autorid

# Asümmeetrilised algoritmid.

## Diffie –Hellman'i algoritm.

- Algoritmi eesmärk – võimaldada kahele isikul vahetada oma võtmeid turvaliselt, nii et neid võtmeid oleks võimalik kasutada sümmeetrilistes algoritmides.
- Algoritmi aluseks on raskused, mis tekivad diskreetsete logaritmide arvutusel:
  - Etteantud  $g$  ja  $a$  jaoks on vaja arvutada  $x$  funktsioonis  $g^x = a$ . Tavaliselt tuginetakse seejuures sobivale korpusele.

Näide: Lahendame võrrandi  $3^x \equiv 13 \pmod{17}$

Kirjutame tabelisse arvu 3 kõik astmed kasutades mod 17:

$3^1 \equiv 3$	$3^2 \equiv 9$	$3^3 \equiv 10$	$3^4 \equiv 13$	$3^5 \equiv 5$	$3^6 \equiv 15$	$3^7 \equiv 11$	$3^8 \equiv 16$
$3^9 \equiv 14$	$3^{10} \equiv 8$	$3^{11} \equiv 7$	$3^{12} \equiv 4$	$3^{13} \equiv 12$	$3^{14} \equiv 2$	$3^{15} \equiv 6$	$3^{16} \equiv 1$

Vastuseks saame  $x=4$ , kuna  $3^4 \equiv 13$ .

# Asümmeetrilised algoritmid.

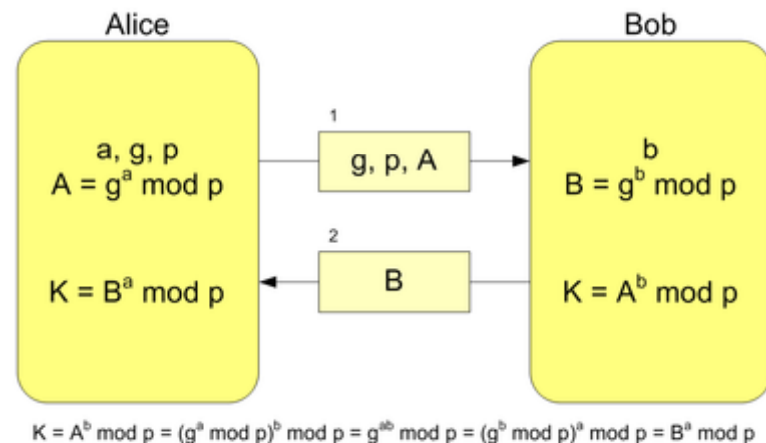
## Diffie –Hellman'i algoritm.

- Tuuakse sisse primitiivse juure mõiste:  $A$  on algarvu  $Q$  primitiivne juur, kui on võimalik moodustada sellest arvust lähtudes jadad  
 $A \bmod Q, A^2 \bmod Q, \dots, A^{Q-1} \bmod Q,$   
kus  $Q$  on väärtusteks kõik võimalikud täisarvud vahemikus  $1 \dots Q-1$
- Sellisel juhul leidub iga  $Y < Q$  ning primitiivse juure  $A$  ning algarvu  $Q$  jaoks ainult üks selline  $x$ , et  
 $Y = A^x \bmod Q,$  kus  $0 \leq x \leq (Q - 1)$
- Arvu  $x$  nimetatakse diskreetseks logaritmiks

# Asümmeetrilised algoritmid.

## Diffie –Hellman'i algoritm.

- Olgu mõlemale isikule teada  $g$  ja  $p$ . Need arvud ei ole salastatud ja võivad olla ka teistele teada.
- Selleks, et saada salastatud võtmed, tuleb mõlemal isikul (nt Alice ja Bob) genereerida suured arvud: näiteks  $a$  ja  $b$ .
- Alice arvutab  $A = g^a \bmod p$  ning saadab Bobile.
- Bob arvutab  $B = g^b \bmod p$  ning saadab Alicele.
- Alice teab arvu  $a$  ning arvu  $B$  ja saab arvutada  $K = B^a \bmod p = g^{ab} \bmod p$
- Samuti Bob teab  $b$  ning  $A$  ja saab arvutada  $K = A^b \bmod p = g^{ab} \bmod p$



Arvu  $K$  kasutatakse krüpteerimisvõtmeks teistes algoritmides, kuna vastase jaoks ei ole mõistliku ajaga lahendatav ülesanne leida  $g^{ab} \bmod p$ , kui on saadud teada  $g^a \bmod p$  ja  $g^b \bmod p$  (juhul kui  $a$  ja  $b$  on tõesti suured)

# Asümmeetrilised algoritmid.

## Ühesuunaline funktsioon

- Ühesuunaliseks funktsiooniks on selline funktsioon, mille iga argumendil on ainult üks pöördväärtus, kusjuures arvutada funktsiooni ennast on lihtne, pöördfunktsiooni aga raske:
  - $Y = f(X)$  – lihtne
  - $X = f^{-1}(Y)$  – raske
- Lihtne tähendab seda, et on võimalik arvutada nn polünomiaalse ajaga, lähtudes sisendi suurusest. Ehk kui sisendbittide arv on  $n$ , siis ajaline keerukus on  $n^a$ , kus  $a$  on fikseeritud konstant.
- Mõiste “raske” on veelgi keerulisem. Lepime kokku, et nimetame probleemi keeruliseks, kui seda ei ole võimalik lahendada polünomiaalse ajaga. Reaalselt tähendab “raske” seda, et kaasaegse tehnikaga pole väärtust võimalik välja arvutada mõistliku aja vältel.

# Keerukus. Näiteid keerukustest

$O(1)$	triviaalne keerukus	paiksaldvestus
$O(\log_2 n)$	logaritmiline keerukus	kahendotsing
$O(n)$	lineaarne keerukus	vektorite sisestamine, väljastamine, liitmine, lahutamine ja skalaarkorrutamine
$O(n \cdot \log_2 n)$		poolitussortimine, kiirsortimine
$O(n \cdot \sqrt{n})$		Shelli sortimine
$O(n^2)$	ruutkeerukus	maatriksite sisestamine, väljastamine, liitmine ja lahutamine, mullsortimine, valiksortimine
$O(n^3)$	kuupkeerukus	maatriksite korrutamine
$O(2^n)$	eksponentsiaalne keerukus	kõigi $n$ -kohaliste kahendsüsteemi arvude tekitamine
$O(n!)$	faktoriaalne keerukus	$n$ elemendi kõigi võimalike järjestuste leidmine

# Keerukus. Polünoom- ja eksponentfunktsioonid

- Polünoomfunktsiooniks nimetatakse funktsioon kujul  $x^a$ , kus  $a$  on konstant ja  $x$  funktsiooni argument.
- Eksponentfunktsiooniks nimetatakse funktsiooni kujul  $a^x$ , kus  $a$  on konstant ja  $x$  funktsiooni argument.
- Ahto Truu järgi: *kui algoritmi keerukusfunktsioon on polünoom, siis arvuti jõudluse kasvamisel kordades kasvab **kordades** ka resultatiivselt töödeldavate andmete maht. Kui aga algoritmi keerukus avaldub eksponentfunktsioonina, siis arvuti jõudluse kasvades kordades, kasvab resultatiivselt töödeldavate andmete maht ainult mingi **konstandi võrra***

a) korrutamine - polünomiaalne keerukus -  $10433 * 16453 = ?$

b) tegurdamine - eksponentsiaalne keerukus -  $? * ? = 171654149$

# Miks me piirdume polünoomiga?

- Kui sooritame liitmise ja korrutamise operatsioone polünoomidega, siis vastuseks on samuti polünoom:

$$O(n^k) + O(n^l) = O(n^{\max\{k,l\}})$$

$$O(n^k) \times O(n^l) = O(n^{k+l})$$

Kõik digitaalarvutid on üksteisega keerukuse osas polünoomiaalselt seotud

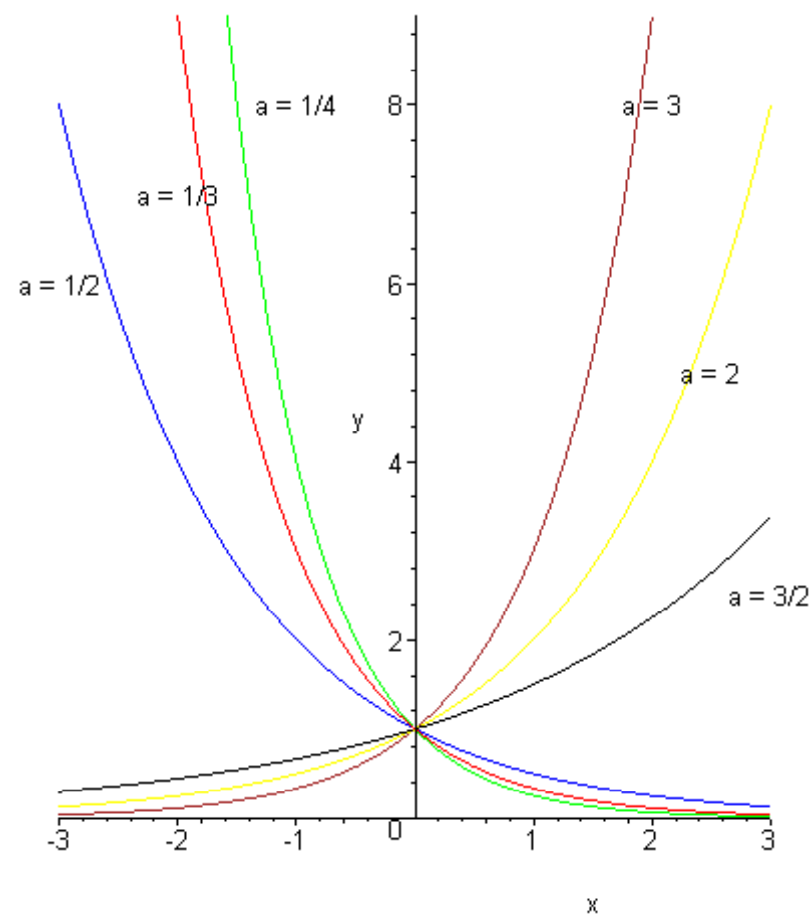
Mitte-polünoomide (faktoriaal, eksponent) korral kasvavad lähteandmetest tulemusteni viivate vajalike arvutuste mahud võrratult kiiremini kui polünoomide korral

[Valdo Prausti järgi]

	Ülesande suurus $n$					
	10	20	30	100	300	500
$n$	0.00001 sekundit	0.00002 sekundit	0.00003 sekundit	0.0001 sekundit	0.0003 sekundit	0.0005 sekundit
$n^2$	0.0001 sekundit	0.0004 sekundit	0.0009 sekundit	0.01 sekundit	0.09 sekundit	0.25 sekundit
$n^3$	0.001 sekundit	0.008 sekundit	0.027 sekundit	1 sekund	27 sekundit	2.08 minutit
$n^5$	0.1 sekundit	3.2 sekundit	24.3 sekundit	2.78 tundi	28.13 ööpäeva	361.4 ööpäeva
$1.2^n$	0.000006 sekundit	0.000038 sekundit	0.00024 sekundit	1.38 minutit	$1.80 \cdot 10^8$ sajandit	$1.2 \cdot 10^{24}$ sajandit
$2^n$	0.001 sekundit	1.05 sekundit	17.9 minutit	$4 \cdot 10^{14}$ sajandit	$6.5 \cdot 10^{74}$ sajandit	$1 \cdot 10^{135}$ sajandit
$3^n$	0.059 sekundit	58.1 minutit	6.52 aastat	$1.6 \cdot 10^{32}$ sajandit	$4 \cdot 10^{127}$ sajandit	$1 \cdot 10^{2233}$ sajandit
$n!$	3.63 sekundit	77094 aastat	$8.4 \cdot 10^{16}$ sajandit	$3 \cdot 10^{142}$ sajandit	$1 \cdot 10^{599}$ sajandit	$4 \cdot 10^{1118}$ sajandit
$n^n$	2.78 tundi	$3.3 \cdot 10^{10}$ sajandit	$6.5 \cdot 10^{28}$ sajandit	$3 \cdot 10^{184}$ sajandit	$4 \cdot 10^{727}$ sajandit	$1 \cdot 10^{1334}$ sajandit

Valdo Praust

## EkspONENTFUNKTSIOONIDE KEERUKUS PRAKTIKAS



# Asümmeetrilised algoritmid.

## Krüptoanalüüs ja võimalikud “ohupunktid”

- Võtme pikkus on oluline, lisab kindlust rünnete puhul, mille käigus proovitakse kõikvõimalikke sümbolite jadasid võtme kombinatsiooniks ( $n$ -õ otserünnakud). Siin kasvab koos võtme pikkusega ka vastava ühesuunalise funktsiooni keerukus. Võtme pikkus peab olema optimaalne, et kaitsta rünnete eest, aga samas peab võimaldama kiiret šifreerimist.
- Teist tüüpi rünne (võrreldes otserünnakuga) on seotud privaatvõtme murdmisega lähtudes avalikust võtmest. Ei ole võimalik matemaatiliselt tõestada, et selline rünne ei oleks võimalik asümmeetriliste algoritmide juures.
- Spetsiifiline rünne asümmeetriliste algoritmide vastu. Genereeritakse tõenäosuslik avatekst. Oletame, et keegi saadab teise (sümmeetrilise) algoritmi jaoks 56 bitise võtme. Vastane saab krüpteerida kõikvõimalikud 56 bitised sümbolite jadad avaliku võtmega. Ning nende abil dešifreerida kätte saadud tekste.

# Algoritm RSA

- RSA (Rivest-Shamir-Adleman) on avaliku võtmesüsteemiga krüpteerimise algoritm andmeedastuseks.
- Esimene algoritm, mis võimaldab samaaegselt krüpteerimist ja digiallkirjastamist



Ron Rivest,  
1947



Adi Shamir,  
1952



Leonard Adleman,  
1945

# Algoritm RSA. Ajalugu

- Esimene algoritmi kirjeldus on avaldatud aastal 1977, ajakirjas Scientific American.
- National Security Agency (NSA) nõudis algoritmi levitamise lõpetamist. Peale väikse koodi avalikustamist (5 rida Perlis) keelasid USA võimud algoritmi levitamist väljaspool riiki. Protestiks trükkisid inimesed selle koodi T-särkidele.
- 1977 RSA loojad krüpteerisid fraasi "*The Magic Words are Squeamish Ossifrage*" ning kuulutasid välja algoritmi murdmise konkursi. Alles aastal 1995 peaaegu õnnestus murda 500 bitise võtmega algoritmi. Selleks oli vaeva näinud 600 inimest ja 1600 masinat.
- 1983 – algoritmile on välja antud patent 4405829 USA, mille kestvus oli kuni 21.sept 2000 ([link](#))
- Aastal 1997 tuli esile info, et Briti matemaatik Clifford Cocks on kirjeldanud analoogset süsteemi juba aastal 1973. Seejärel aastal 1974 Malcolm John Williamson leiutas algoritmi (salastatud kuni 1997), mis on analoogiline Diffie-Hellman algoritmiga.

# Algoritm RSA. Kuidas aru saada?

- Esitame lihtsa näite, mis aitab aru saada algoritmi põhimõttest.
- **Paroolide hoidmine arvutis.** Igal kasutajal on oma parool. Sisse logides sisestab kasutaja salastatud parooli ja siseneb arvutisse. Arvutis peab olema “osa”, mille abil kontrollitakse, kas sisestatud parool on õige. Aga selle “osa” hoidmine avatud kujul kõvakettal on ohtlik (kättesaadav teistele). Probleemi lahendamiseks – ühesuunaline funktsioon. Arvutis hoitav osa on selle funktsiooni rakendamise tulemus. Olgu Alice parooliks “Gladiool”. Arvutatakse  $f(\text{Gladiool})$ , olgu tulemuseks “Kummel” ning seda hoitakse arvutis parooli kontrollimiseks.

# Algoritm RSA. Kuidas aru saada?

- Näide “salalüంగా”. Kui on vaja ikkagi leida võimalust taastada esialgne tekst, siis vajame salajasi lünki.
- Olgu meil šifreerimise vahendiks suur mitmekõiteline telefoni raamat, kus nime järgi on lihtne leida telefoni numbrit, aga vastupidi telefoni järgi on üsna keeruline leida inimese nime. Šifreerimisel valitakse iga tähe jaoks selle algustähega perekonnanimi, tulemuseks antakse meie tähele vasteks telefoni number.

Avatekst	Valitud nimi	Krüptotekst
K	Kass	5643452
A	Adomson	3572651
D	Dior	4673956
E	Esmar	3517289
D	Dorbek	7755628
U	Uusküla	1235267
S	Sibul	8492746

Perekonnanimi  
valitakse  
juhuvalikuga  
samatäheliste  
hulgast

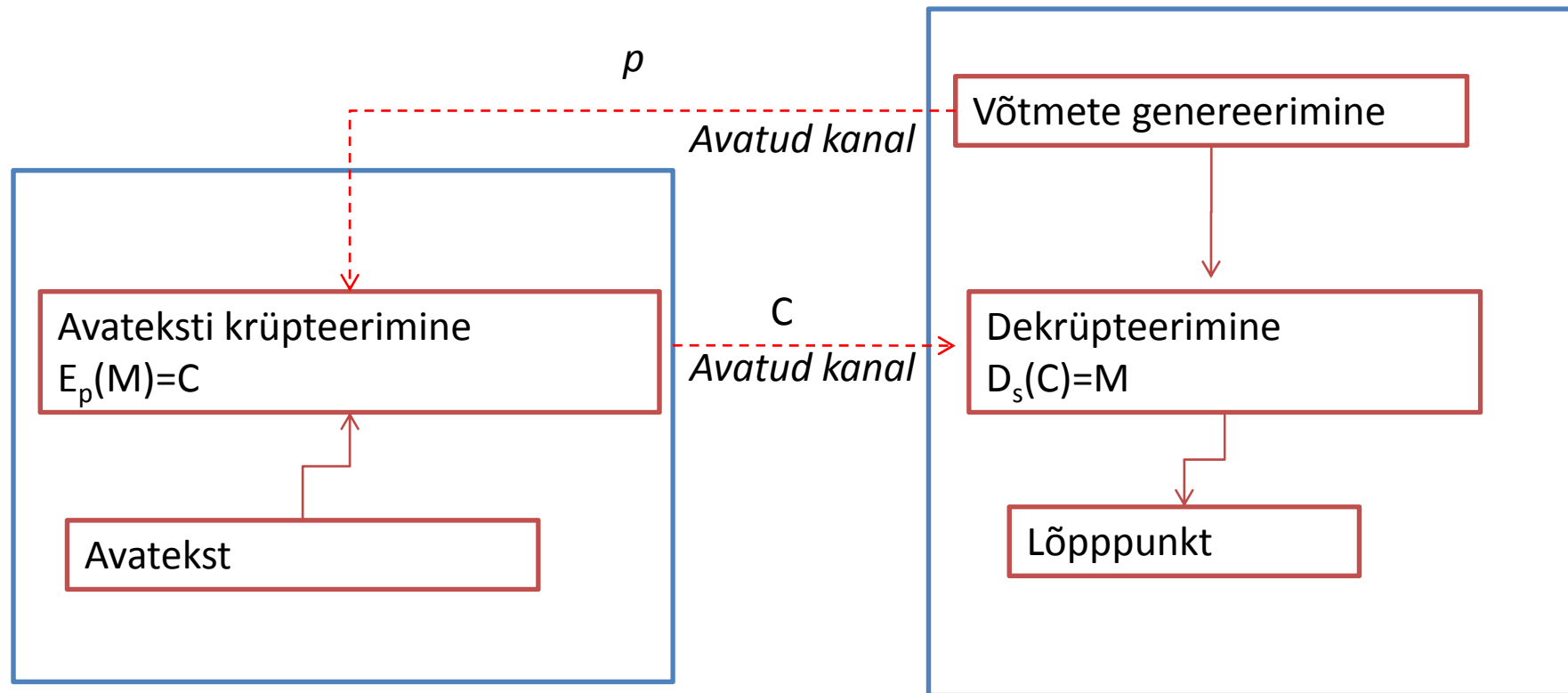
# Algoritm RSA. Kuidas aru saada?

- Kuna perekonnanime valitakse juhuslikult, siis krüpteerimistulemusi võib olla päris mitu

Krüptotekst 1	Krüptotekst 2	Krüptotekst 3
1235267	5643452	1235267
3572651	3572651	3517289
4673956	4673956	4673956
3517289	3517289	3572651
7755628	7755628	7755628
5643452	1235267	5643452
8492746	8492746	8492746

Selleks, et dekrüpteerida tekst peab olema meil tagurpidi telefoniraamat, mis on sorteeritud telefoninumbrite järgi

# Algoritm RSA. Kuidas aru saada?



# Algoritm RSA. Võtmete moodustamine

- Avaliku võtme krüptosüsteemi aluseks on  $n$ -ö ühesuunaline ülesanne, mis on seotud korrutamisega ja kordarvude esitamisega algarvude astmete korrutistena.
- Vajame kahte võtit: avalikku võtit ja privaatvõtit, mis on üksteise pöördväärtusteks

Avatekstid  $M$  peavad olema pärit hulgast  $W$ , kus  $W$  on kõikide lubatavate avatekstide hulk

Iga avatud võtme  $P$  ja privaatvõtme  $S$  korral

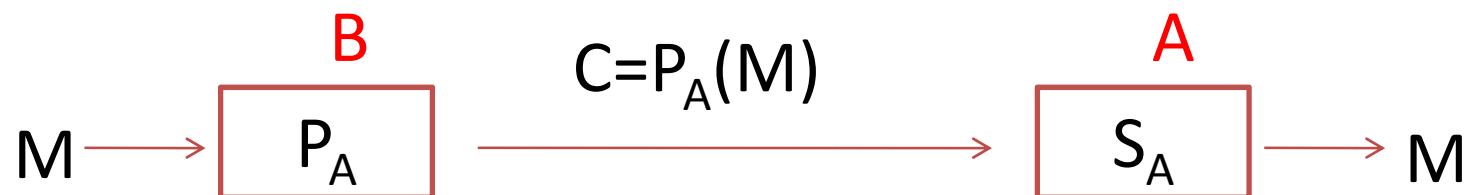
leidub vastav šifreerimise funktsioon  $E_p()$  ja dešifreerimise funktsioon  $D_s()$ , mille korral:

$$M = D_s(E_p(M))$$

$$M = E_p(D_s(M))$$

# Algoritm RSA.

## Krüpteerimine ja dekrüpteerimine



Võtta vastu avatud võti  $(e, n)$  isikult  $A$   
Võtta avatekst  $M$   
Edastada krüpteeritud tekst  
 $P_A(M) = M^e \bmod n$

Võtta vastu krüpteeritud tekst  $C$   
Rakendada privaatvõtit  $(d, n)$   
sõnumi dekrüpteerimiseks  
 $S_A(C) = C^d \bmod n$

Sõnum ise (avatekst) koosneb täisarvudest vahemikus  $0 \dots n-1$

# Algoritm RSA. Võtmete moodustamine

- Valitakse kaks etteantud suurusega algarvu  $p$  ja  $q$  (näiteks mõlemad suurusega 1024 bitti)
- Leitakse nende korrutist  $n=p \times q$ , mida nimetatakse mooduliks
- Arvutatakse Euleri funktsiooni väärtus  $\phi(n)=(p-1)(q-1)$
- Valitakse täisarv  $e : 1 < e < \phi(n)$  nii, et  $e$  ning  $\phi(n)$  **ei oma ühiseid tegureid**. Tavaliselt võetakse arvuks  $e$  mõni sobiv Ferma algarv - näiteks 17, 257, või 65537; ehk üldiselt mõni arvudest  $F_n = 2^{2^n} + 1$ , kus  $n$  on mittenegatiivne täisarv
  - Arvu  $e$  nimetatakse avatud eksponendiks (*public exponent*)
  - Aeg, mis on vajalik šifreerimiseks on võrdeline “üheste” bittide arvuga arvu  $e$  kahendkoodi esituses.
  - Väga väiksed  $e$  arvud, näiteks 3 võivad nõrgendada RSA algoritmi

# Algoritm RSA. Võtmete moodustamine

- Arvutatakse arv  $d$ , mis on multiplikatiivne pöördväärtus arvule  $e \bmod \phi(n)$ , ehk:  
 $de \equiv 1 \pmod{\phi(n)}$ , või  $de = k\phi(n) + 1$ , kus  $k$  on suvaline täisarv  
Arvu  $d$  nimetatakse salastatud eksponendiks  
Tavaliselt arvutatakse Eukleidese laiendatud algoritmiga  
(Extended Euclidean algorithm)  
[http://en.wikipedia.org/wiki/Euclidean\\_algorithm](http://en.wikipedia.org/wiki/Euclidean_algorithm)
- Paar  $P=(e, n)$  on avalik võti
- Paar  $S=(d, n)$  esitab privaativõtit ning seda hoitakse saladuses

# Algoritm RSA. Näide

Etapp	Operatsiooni kirjeldus	Operatsiooni tulemus
Võtmete genereerimine	Valida kaks algarvu	$p=3557, q=2579$
	Arvutada moodul	$n=p \times q = 3557 \times 2579 = 9173503$
	Arvutada Euleri funktsiooni väärtust	$\phi(n) = (p-1)(q-1) = 9167368$
	Valida avatud eksponent $e$	$e=3$
	Arvutada salastatud eksponent $d$	$d=6111579 (k=2)$
	Avalikustada avavõti	$(e,n) = (3, 9173503)$
	Salvestada privaativõti	$(d,n) = (6111579, 9173503)$
Šifreerimine	Valida tekst šifreerimiseks	$M=111111$
	Arvutada krüptogramm	$P(M) = M^e \bmod n = 111111^3 \bmod 9173503 = 4051753$
Dešifreerimine	Leida avatekst	$S(C) = C^d \bmod n = 4051753^{6111579} \bmod 9173503 = 111111$

# Algoritm RSA.

## Probleemid ja nende ületamine

- Võtmete genereerimist kasutatakse tunduvalt harvem kui šifreerimist/dešifreerimist. Kiiruse mõjutaja on  $a=b^e \bmod n$  arvutus. Vajame algoritmi, mis kiirendaks astmesse tõstmist.

Keerukus kasvab, kui kahendkoodis on palju ühtesid. Seega oleme huvitatud arvudest, mille kahendkoodid sisaldavad rohkem nulle.

Ferma arvud:  $17 = 10001$ ,  $257 = 100000001$ ,  $65537 = 1000000000000000001$

# Algoritm RSA.

## Probleemid ja nende ületamine

### LR Binary Method

*Input:*  $M, e, n$

$h=6, e=55=110111$

*Output:*  $C := M^e \pmod n$

1. **if**  $e_{h-1} = 1$  **then**  $C := M$  **else**  $C := 1$
2. **for**  $i = h - 2$  **downto**  $0$ 
  - 2a.  $C := C \cdot C \pmod n$
  - 2b. **if**  $e_i = 1$  **then**  $C := C \cdot M \pmod n$
3. **return**  $C$

$i$	$e_i$	Step 2a ( $C$ )	Step 2b ( $C$ )
4	1	$(M)^2 = M^2$	$M^2 \cdot M = M^3$
3	0	$(M^3)^2 = M^6$	$M^6$
2	1	$(M^6)^2 = M^{12}$	$M^{12} \cdot M = M^{13}$
1	1	$(M^{13})^2 = M^{26}$	$M^{26} \cdot M = M^{27}$
0	1	$(M^{27})^2 = M^{54}$	$M^{54} \cdot M = M^{55}$

# Algoritm RSA.

## Probleemid ja nende ületamine

- Probleemid  $p$  ja  $q$  valimisega.
- Kuna nende arvude korrutis, ehk moodul  $n$  on arv, mida avalikustatakse, siis  $p$  ja  $q$  peaksid olema võimalikult suured. Samas algoritm, mis otsib suurt algarvu peab olema efektiivne.
- Selleks kasutatakse protseduuri, mis valib suvalise paaritu arvu ning kontrollib, kas see arv on algarv. Kui see arv ei ole algarv, siis valitakse järgmine paaritu juhuarv ning kontrollitakse. Korratakse, kuni algarv on leitud. Suvalise suurusega algarve praeguste lahendustega otsida ei saa.
- On töötatud välja testid, mis kontrollivad, kas arv on algarv teatud tõenäosusega.
- Keerukust tõstab ka see, mitu arvu “praagitakse välja”, enne kui leitakse algarv. Arvuteooriast on teada, et algarvude kogus, mis paiknevad arvu  $a$  ümbruses on keskmiselt üks igale  $\ln(a)$  jaoks. Paaris arve jätame välja. Järelikult tuleb kontrollida  $\approx \ln(a)/2$  arvu enne kui leitakse algarv. Näiteks, kui otsitakse algarvu diapsoonis  $2^{200}$ , siis tuleb kontrolli teostada  $\approx \ln(2^{200})/2=70$  korda

# Algarvu kontrollimise testid. Näited

- Algoritm, mis kontrollib kas arv on algarv. Aastal 2002 on tõestatud, et antud ülesanne on polünoomiaalse keerukusega.

Lihtne täisarvude uuring diapsoonis 2 kuni  $j=\sqrt{n}$ .

Kui  $n$  jagub ilma jäägita, siis see arv ei ole algarv

```
bool is_prime(int n)
{
    if (n <= 1) {
        return false; }
    for (int j = 2; j * j <= n; j++) {
        if (n % j == 0) {return false;}
    }
    return true; }
```

# Algarvu kontrollimise testid. Näited

## Miller–Rabin'i test

**Sisend:**  $m > 2$  – paaritu arv, mida on vaja kontrollida

$r$ - parameeter, mis esitab testi täpsust

**Väljund:** mitte algarv – ehk on olemas tegurid arvule  $m$

algarv – arv  $m$  on suure tõenäosusega algarv

Esitada arv  $m-1$  kujul  $2^s t$ , kus  $t$  on paaritu, saab teha järjestiku jagamisega arvu  $m-1$  arvuga 2.

Tsükkel A: korrata  $r$  korda

Valida juhuarv  $a$  vahemikus  $[2, m - 2]$

$x := a^t \bmod m$

kui  $x=1$  või  $x=m-1$ , siis minna tsükli A järgmisele ringile

Tsükkel B: korrata  $s-1$  korda

$x := x^2 \bmod m$

kui  $x=1$ , siis tagastada “mitte algarv”

kui  $x=m-1$ , siis minna järgmisele ringile tsükli A

väljastada “mitte algarv”

väljastada “algarv”

$a$  on algarvu tunnus  
kui  $m$  ei jagu  $a$ -ga ning  
 $a^t \equiv 1 \pmod{m}$  või  
 $\exists$  täisarvuline  $k$ ,  
 $0 \leq k \leq s$ , selline, et  
 $A^{2^k t} \equiv -1 \pmod{m}$

# RSA krüptoanalüüs

Valdo Praust:

- *70-kohalise arvu algteguriteks lahutamine nõuab kaasajal keskmiselt tööjaamalt ca 40 minutit*
- *100-kohaline – samalt arvutilt ca kaks nädalat*
- *140-kohaline arv lahutati 1996 teguriteks 5 aastaga, ühendades maailma paljude serverite jõupingutused*
- *Praegu on suurim teguriteks lahutatud arv 199-kohaline ehk 663-bitine (tehti A.D. 2005 hajusarvutusvõrgu abil)*
- *300-kohaline arv (1024-bitine RSA) nõuab kogu praeguselt arvutivõimsuselt tööd kauemaks kui on Päikese eluiga (kümned miljardid aastad)*
- *On kaheldud, et 1024-bitine RSA võib 5-10 aasta pärast olla praktikas murtav, kuid sama 2048/4096-bitise RSA kohta enam ei arvata*
- *Kuni 256-bitise RSA murrab kaasaja tavaline personaalarvuti ca tunniga*
- *Piisavalt võimas kvantarvuti murraks ka RSA kiiresti lahti, kui arvatakse, et lähema 10 aasta jooksul need 1024-bitist RSAd ei ohusta*

# RSA algoritmi miinused

- Sobib väikeste avatekstide krüpteerimiseks, suurte tekstide puhul muutub aeglaseks. Lahendus: krüpteerida sümmeetrilise krüptoalgoritmiga ning selle võti edastada asümmeetrilise algoritmiga
- Võrreldes sümmeetriliste algoritmidega (AES, IDEA, Blowfish vms) on sadu tuhandeid kordi aeglasem

# Avaliku võtmeaga krüptograafia rakendusi

- Turvaline andmevahetus:
  - VPN
  - Turvaline e-mail
- Juurdepääsu kontrollimine:
  - Andmebaasi kaitse
  - Võrgu kaitse
  - Serveri kaitse
- Digitaalalkiri