

RSA kryptosystemet

- 1 Find to primtal p og q hvor $p \neq q$ (gerne 512 bit)
- 2 Udregn $n = pq$
- 3 Vælg lille ulige heltal e som er relativt primisk med $\phi(n)$
- 4 Udregn d som multiplikativt inverse af e modulo $\phi(n)$
- 5 RSA public key: $P = (e, n)$
- 6 RSA secret key: $S = (d, n)$

I skridt 3: Størrelsen af \mathbb{Z}_n^*

$$\phi(n) = n \prod_{a|n} \left(1 - \frac{1}{a}\right)$$

Da $n = pq$ fås

$$\phi(n) = pq \left(1 - \frac{1}{p}\right) \left(1 - \frac{1}{q}\right) = pq - q - p + 1 = (p-1)(q-1)$$

I skridt 4: Algoritme korollar 31.26 bestemmer d entydigt

Transformationer

- Public key $P = (e, n)$ transformation $P(M) = M^e \bmod n$
 - Secret key $S = (d, n)$ transformation $S(C) = C^d \bmod n$
- hvor $M \in \mathbb{Z}_n$, dvs $M \leq n$

5

RSA kryptosystemet

Indsigt

- Hvis kender faktorisering $n = pq$ da kender jeg $\phi(n) = (p-1)(q-1)$.
- Hvis jeg ikke kender $\phi(n)$ kan jeg ikke udregne multiplikativt inverse af e modulo $\phi(n)$.
- Kinesisk restklassætning tillader at jeg regner i $(\mathbb{Z}_p \times \mathbb{Z}_q)$ frem for \mathbb{Z}_n . Dette udnyttes i korrekthedsbevis for algoritmen.

6

Eksempel

Finder to "store" primtal $p = 5, q = 11$

Udregner $n = pq = 55$

$$\phi(n) = (5-1)(11-1) = 4 \cdot 10 = 40$$

Vælg tilfældigt ulige heltal $e = 7$

Find multiplikativt inverse $d = 23$ af tallet e modulo $\phi(n)$

Public key $(e, n) = (7, 55)$

Secret key $(d, n) = (23, 55)$

Besked $M = 8$

$$\begin{aligned} P(M) &= M^e \bmod n \\ &= 8^7 \bmod 55 \\ &= 2097152 \bmod 55 \\ &= 2 \\ &= C \end{aligned}$$

$$\begin{aligned} S(C) &= C^d \bmod n \\ &= 2^{23} \bmod 55 \\ &= 8388608 \bmod 55 \\ &= 8 \\ &= M \end{aligned}$$

M^e og C^d udregnes ikke direkte
brug MODULAR-EXPONENTIATION

7

Køretid af RSA

Public key og Secret key operationer

- MODULAR-EXPONENTIATION benyttes køretid $O(\beta^3)$ hvis β bit lange tal
- public key (e, n) hvor e er et lille heltal $\log e = O(1)$ og $\log n \leq \beta$
- secret key (d, n) $\log d \leq \beta$ og $\log n \leq \beta$
- public key transformation $M^e \bmod n$ $O(1)$ multiplikationer, $O(\beta^2)$ bit operationer
- secret key transformation $C^d \bmod n$ $O(\beta)$ multiplikationer, $O(\beta^3)$ bit operationer

Vælger e lille så offentlig nøgle transformation hurtig

Et løst overslag

- Med $\beta = 1024$ bit, kræver afkodning 10^9 bit-operationer
- Hurtig CPU kan beregne 10^9 multiplikationer/sek
- Hvis CPU kan multiplicere to 64-bit tal i hardware, svarer det til $64^2 = 4096$ bit-operationer samtidigt
- CPU klarer $10^9 \cdot 4096 = 4 \cdot 10^{12}$ bit-operationer/sek
- Ideelt kan vi kode 4096 tal af længde 1024 bit/sek

8

Korrektthed af RSA kryptosystemet

$$P(M) = M^e \pmod n \quad S(C) = C^d \pmod n$$

Vil vise at S_A og P_A definerer inverse transformation af \mathbb{Z}_n .

$$M = S_A(P_A(M)) \quad M = P_A(S_A(M))$$

Bevis: For enhver besked $M \in \mathbb{Z}_n$

$$P(S(M)) = S(P(M)) = M^{ed} \pmod n$$

Da e og d multiplikativt inverse modulo $\phi(n) = (p-1)(q-1)$

$$\begin{aligned} ed &\equiv 1 \pmod{(p-1)(q-1)} \\ ed &= 1 + k(p-1)(q-1) \end{aligned}$$

for et heltal $k \in \mathbb{Z}$.

Hvis $M \not\equiv 0 \pmod p$ gælder

$$\begin{aligned} M^{ed} &\equiv M^{1+k(p-1)(q-1)} \pmod p \\ &\equiv M(M^{p-1})^{k(q-1)} \pmod p \\ &\equiv M(1)^{k(q-1)} \pmod p \quad [\text{theorem 31.31, } M \in \mathbb{Z}_p^*] \\ &\equiv M \pmod p \end{aligned}$$

Hvis $M \equiv 0 \pmod p$ gælder

$$M^{ed} \equiv M \pmod p$$

Så ovenstående gælder for alle M .

9

Korrektthed af RSA kryptosystemet

Tilsvarende vises

$$M^{ed} \equiv M \pmod q$$

for alle M .

Kinesisk restklassesætning, korollar 31.29 giver

$$\left. \begin{aligned} M^{ed} &\equiv M \pmod p \\ M^{ed} &\equiv M \pmod q \end{aligned} \right\} \Leftrightarrow M^{ed} \equiv M \pmod n$$

for alle M .

10

RSA kryptosystemets sikkerhed

- Offentlige nøgle er $P = (e, n)$.
- Hvis fjenden kan faktorisere $n = pq$ så kan han bruge RSA algoritmens skridt 2-4 til at finde d i den hemmelige nøgle $S = (d, n)$.

Hvis faktorisering er let \Rightarrow nemt at bryde RSA

Hvis faktorisering er svært \Rightarrow måske svært at bryde RSA

Kryptering af lang besked

Offentlig-nøgle system sikrer

- Kontrol af afsender-modtager
- Kryptering af besked

Et offentlig-nøgle system er beregningsmæssigt dyrt

- Offentlig-nøgle systemet bruges til at kontrollere afsender-modtager
- Offentlig-nøgle systemet bruges til at udveksle simple nøgler
- Simple kryptosystem bruges til lange besked

11

Effektiv digital signatur (s.866)

Hash-funktion $h : M \rightarrow \mathbb{N}$

$$M \neq M' \Rightarrow h(M) \neq h(M')$$

(eller i hvert fald svært at finde M')

$h(M)$ er et kort (fx. 160-bit) "fingeraftryk" af M

Alice

udregner $h(M)$
sender $(M, S_A(h(M)))$

Bob

udregner $t_1 = h(M)$
udregner $t_2 = P_A(S_A(h(M)))$
hvis $t_1 = t_2$ er besked ikke blevet ændret
afsender er helt sikkert Alice

Offentlig myndighed T

Antag at alle kan stole på T

Alle kender public key P_T

Alice får certifikat $S_T(M')$ på at $M' =$ "Alices offentlige nøgle er P_A "

Alice sender certifikat $S_T(M)$ med M

Alle kan afkode $S_T(M)$, og ved at Alice signerede M

12

At finde primtal

Der findes heldigvis mange primtal
Lad $\pi(n)$ være antallet af primtal mindre end n .

$$\lim_{n \rightarrow \infty} \frac{\pi(n)}{n/\ln n} = 1$$

Dvs. $\pi(n)$ er tæt på $n/\ln n$.

$$n = 10^9 \Rightarrow \pi(n) \approx 48.254.942$$

Naiv algoritme

For at teste om n er et primtal, divideres det med alle heltal

$$2, \dots, \sqrt{n}$$

Eksponentiel køretid!

Input: $\Theta(\log n)$ bits

Køretid: $O(\sqrt{n})$

Lad $m = \log \sqrt{n}$, dvs. $2m = \log(\sqrt{n})^2 = \log n$

Input: $\Theta(2m)$ bits

Køretid: $O(2^m)$

13

Egenskaber ved primtal

Lad

$$\mathbb{Z}_n^+ = \mathbb{Z}_n \setminus \{0\} = \{1, 2, \dots, n-1\}$$

Husk at

$$\mathbb{Z}_n^* = \{a \in \mathbb{Z}_n : \gcd(a, n) = 1\}$$

Hvis n er primtal så er

$$\mathbb{Z}_n^* = \mathbb{Z}_n^+$$

Theorem 31.34

Hvis n er primtal og $e \in \mathbb{N}$ ($n > 2$, $e \geq 1$) så har ligningen

$$x^2 \equiv 1 \pmod{n^e}$$

kun to rødder: $x = 1$ og $x = -1$ (trivielle rødder)

Fermats [lille] theorem

$$n \text{ er primtal} \Rightarrow a^{n-1} \equiv 1 \pmod{n}$$

hvilket er ækvivalent med

$$n \text{ er sammensat} \Leftarrow a^{n-1} \not\equiv 1 \pmod{n}$$

Desværre gælder ikke

$$n \text{ er primtal} \Leftarrow a^{n-1} \equiv 1 \pmod{n}$$

14

Pseudoprimtal

n er base- a pseudoprimtal hvis n er sammensat og

$$a^{n-1} \equiv 1 \pmod{n}$$

Eksempel

$341 = 31 \cdot 11$ er base-2 pseudoprimtal da

$$2^{341-1} \equiv 1 \pmod{341}$$

Pseudoprimtalstest

Givet n som vi skal afgøre om er et primtal

Vælg $a \in \mathbb{Z}_n^+$

Udregn om $a^{n-1} \equiv 1 \pmod{n}$

Hvis $a^{n-1} \not\equiv 1 \pmod{n}$ så helt sikkert sammensat

Hvis $a^{n-1} \equiv 1 \pmod{n}$ så primtal eller pseudoprimtal

Rimeligt god primtalstest

Brug $a = 2$ i ovenstående algoritme

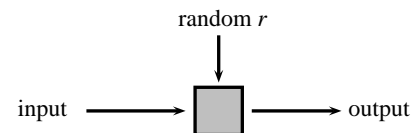
$$2^{n-1} \equiv 1 \pmod{n} \Rightarrow \begin{cases} \text{primtal} \\ \text{pseudoprimtal} \end{cases}$$

Fejler kun for 22 værdier af $n \leq 10.000$

15

Randomiserede algoritmer

Algoritme som benytter tilfældigt tal r i sine beregninger



Varianter

- 1 Algoritmen giver altid korrekt svar, men køretiden afhænger af r .
- 2 Algoritmen returnerer et korrekt svar med en vis sandsynlighed, køretid er altid samme.

Eksempler på randomiserede algoritmer:

1 QUICKSORT

worst-case tid: $O(n^2)$

randomiseret: $O(n \log n)$ med stor sandsynlighed.

2 MIN-CUT

finder min-cut i graf (V, E) med sandsynlighed $\frac{1}{n^2}$

køretid $O(n^2 \log n)$, $n = |V|$.

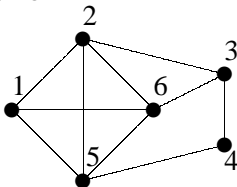
16

Randomiserede algoritmer (eksempel)

Algoritme til at finde minimalt snit i graf $G = (V, E)$

```
MINCUT( $V, E$ )
1 while  $|V| > 2$  do
2   pick randomly an edge  $(u, v)$  in  $E$ 
3   contract the edge, while preserving multi-edges
4   remove all loops
5 output remaining edges
```

Sammentrækning (contracting) af en kant (u, v) betyder at der tilføjes en ny knude w , alle kanter (u, x) eller (v, x) erstattes med (w, x) og herefter slettes u og v fra grafen G .



[1] R. Motwani and P. Raghavan. Randomized Algorithms. Cambridge University Press, New York (NY), 1995.

17

Randomiserede algoritmer

- Antag at vi skal besvare et ja-nej spørgsmål S .
- Vi har en randomiseret algoritme $R(S, r)$ som
 - Baseret på et tilfældigt tal r svarer “ja”-“nej”
 - Hvis den svarer “nej” er den helt sikker på sit svar
 - Hvis den svarer “ja” er den $p < 1$ sikker på sit svar

Hvis vi kalder den randomiserede algoritme k gange med forskellige r og hver gang får “ja” er sandsynligheden for at $S = \text{“nej”}$:

$$(1 - p)^k$$

Eksempel:

Hvis $p = 0.5$, og svarer “ja” $k = 100$ gange, er sandsynligheden for at $S = \text{“nej”}$

$$(1 - 0.5)^{100} = 7.89 \cdot 10^{-31}$$

Hvis $p = 0.1$, og svarer “ja” $k = 100$ gange, er sandsynligheden for at $S = \text{“nej”}$

$$(1 - 0.1)^{100} = 2.66 \cdot 10^{-5}$$

18

Randomiseret algoritme

Hvad er bedst:

1. For 50% af input svarer algoritmen rigtigt
2. For alle input svarer algoritmen 50% rigtigt

Formelt: sandsynligheden p for at et ja-svar er rigtigt

1. $\forall r, \exists I' \subseteq I, |I'| \geq \frac{1}{2}|I|, \forall i \in I' : p(i, r) = 100\%$
2. $\forall i \in I, \forall r : p(i, r) \geq 50\%$

19

Pseudoprimtalstest

Vælg tilfældigt $a \in \mathbb{Z}_n^*$

$$a^{n-1} \equiv 1 \pmod{n} \Rightarrow \begin{cases} \text{primtal} \\ \text{pseudoprimtal} \end{cases}$$

Carmichael tal

$$n = 561, 1105, 1729, 2465, 2821, 6601, 8911, \dots$$

For alle a

$$a^{n-1} \equiv 1 \pmod{n}$$

Pseudoprimtal for alle baser a

Pseudoprimtalstest kan ikke udvides til randomiseret algoritme der afhænger af a

Carmichael tal er sjældne:

- 255 stk under 100.000.000.
- 105212 stk under 10^{15} .
- 585355 stk under 10^{17} .

20

Bedre pseudoprimalstest

For alle $e \in \mathbb{N}$ (bruger $e = 1$)

n er primtal $\Rightarrow x^2 \equiv 1 \pmod{n^e}$ har løsning $x = \pm 1$

n er sammensat $\Leftarrow x^2 \equiv 1 \pmod{n^e}$ har løsning $x \neq \pm 1$

For alle $a \in \mathbb{N}$

n er primtal $\Rightarrow a^{n-1} \equiv 1 \pmod{n}$

n er sammensat $\Leftarrow a^{n-1} \not\equiv 1 \pmod{n}$

WITNESS-COMPOSITE(a, n)

1 $n - 1 = 2^t u$ hvor $t \geq 1$ og u ulige

2 $x_0 \leftarrow \text{MODULAR-EXPONENTIATION}(a, u, n)$

3 **for** $i \leftarrow 1$ **to** t **do**

4 $x_i \leftarrow x_{i-1}^2 \pmod{n}$

5 **if** $x_i = 1$ and $x_{i-1} \neq 1$ and $x_{i-1} \neq n - 1$ **then**

6 **return** TRUE $\triangleright x_{i-1} \not\equiv \pm 1 \pmod{n}$ og
 $x_i \equiv x_{i-1}^2 \equiv 1 \pmod{n}$

7 **if** $x_t \neq 1$ **then**

8 **return** TRUE $\triangleright x_t = (a^{n-1} \pmod{n}) \neq 1$

9 **return** FALSE

Hvis returnerer TRUE så er n er sammensat

Hvis returnerer FALSE så er n måske primtal

21

Eksempel på primtalstest

Carmichael tal $n = 561 = 11 \cdot 51$

Vælg tilfældigt a f.eks. $a = 7$

$$n - 1 = \langle 1000110000 \rangle = 35 \cdot 2^4 = u \cdot 2^t$$

Ønsker at teste

n er sammensat $\Leftarrow a^{n-1} \not\equiv 1 \pmod{n}$

MODULAR-EXPONENTIATION

• Første 6 bit køres normalt

• Sidste 4 bit alene kvadreringer

n er sammensat $\Leftarrow x^2 \equiv 1 \pmod{n}$ har løsning $x \neq \pm 1$

Håndkørsel af MODULAR-EXPONENTIATION

i	9	8	7	6	5	4	3	2	1	0
bit	1	0	0	0	1	1	0	0	0	0
c	1	2	4	8	17	35	70	140	280	560
d	7	49	157	526	160	241	298	166	67	1

$$(a^{280})^2 = a^{560} = 7^{560} \equiv 1 \pmod{n}$$

Dvs. med $x = a^{280}$ have

$$x^2 \equiv 1 \pmod{n}$$

så er n sammensat

22

Miller-Rabin's algoritme

Input: ulige heltal n større end 2, s antal forsøg

MILLER-RABIN(n, s)

1 **for** $j \leftarrow 1$ **to** s **do**

2 $a \leftarrow \text{RANDOM}(1, n - 1)$

3 **if** WITNESS-COMPOSITE(a, n) **then**

4 **return** COMPOSITE \triangleright Definitely

5 **return** PRIME \triangleright Almost for sure

Theorem 31.38: Hvis n er ulige og sammensat, så er antallet af vidner til at n er sammensat mindst $\frac{n-1}{2}$.

Theorem 31.39: For et ulige heltal $n > 2$ og pos. heltal s er chancen for at MILLER-RABIN(n, s) svarer forkert højst $(\frac{1}{2})^s$.

Hvis man bruger $s = 50$ så er chancen for fejl:

$$\left(\frac{1}{2}\right)^{50} = 8.88 \cdot 10^{-16}$$

23

Faktoriserings af heltal er svært (Afsnit 31.9 kursorisk)

Det er svært at faktorisere heltal

- Vi har ikke bevis for at effektiv algoritme ikke findes
- Hvis nogen finder en effektiv algoritme kan de bryde RSA og indkassere stor præmie
- Bedste algoritmer kørt på supercomputere kan ikke faktorisere vilkårligt 1024-bit tal (p og q blev valgt som 512-bit så n er 1024-bit)

24

Opsummering

- RSA offentlig nøgle krypteringssystem
- Primalstest
- Randomiserede algoritmer

If, using a randomized method, the probability of error is 2^{-1000} , the philosophical question arises: is this a mathematical proof?