Pravděpodobnostní (randomizované) algoritmy

pravděpodobnostní algoritmus dělá (na rozdíl od deterministického algoritmu) náhodné kroky, např. k některým krokům používá hodnoty získané z generátoru náhodných čísel

tím pádem dvě různá spuštění téhož pravděpodobnostního algoritmu na stejných datech mají (s velkou pravděpodobností) různý průběh

pravděpodobnostních algoritmů je mnoho typů, zde zmíníme jen dva a to algoritmy typu Las Vegas a typu Monte Carlo

Algoritmy typu Las Vegas

výsledek je vždy správný, náhodnost ovlivňuje pouze dobu běhu algoritmu, tj. po jaké cestě se algoritmus ke správnému výsledku dobere

Příklad: randomizovaný QuickSort – od deterministické verze se liší náhodnými výběry pivota při každém dělení posloupnosti, což poskytuje následující výhody

• dává dobrý průměrný čas (tj. $O(n \log n)$) i v případě, že data na vstupu nejsou náhodné permutace – žádný vstup není apriori špatný (pro každý deterministický výběr pivota existují apriori špatné vstupy)

• může být spuštěn paralelně v několika kopiích, výsledek je získán z kopie, kde výpočet skončí nejdříve (pro deterministickou verzi nemá takový postup žádný smysl)

Algoritmy typu Monte Carlo

náhodnost ovlivňuje jak dobu běhu, tak správnost výsledku: algoritmus může udělat chybu, ale pouze jednostranně (u odpovědí ANO/NE) a s omezenou pravděpodobností

Příklad: Rabin-Millerův algoritmus na testování prvočíselnosti

Úloha: pro zadané přirozené číslo n (rychle) rozhodnout zda je n prvočíslo

Trocha teorie (Malá) Fermatova věta (bez důkazu):

Nechť p je prvočíslo. Potom $\forall\ k \in \{1,2, \dots ,p-1\}$ platí $k^{p-1} \equiv 1 \pmod p$

Myšlenka: pokud n není prvočíslo, tak zkusíme (náhodně) najít „svědka“ k, porušujícího $k^{n-1} \equiv 1 \pmod n$, který „dosvědčuje“, že n je opravdu číslo složené (není to prvočíslo)

Problém: pro některá složená čísla je svědků příliš málo, takže je ¨příliš malá pravděpodobnost, že nějakého svědka (náhodně) vybereme.

Definice: Nechť T je množina dvojic přirozených čísel, kde $(k,n) \in T$ pokud $0 < k < n$ a je splněna alespoň jedna z následujících dvou podmínek:

1. neplatí $k^{n-1} \equiv 1 \pmod n$,

2. existuje i takové, že $m = (n-1) / 2^i$ je přirozené číslo a platí $1 < NSD(k^{m-1}-1, n) < n$

Věta 1: Číslo n je složené tehdy a jen tehdy, když existuje k takové, že $(k,n) \in T$.

Věta 2: Nechť n je složené číslo. Pak existuje alespoň $(n-1)/2$ takových čísel k, pro které platí $(k,n) \in T$.

```
Rabin-Miller(n);
for i:=1 to počet do
   k_i := náhodné přirozené číslo z intervalu [1,n-1];
   if (k_i,n) ∈ T then Report (n je složené);
                          Abort;
Report (n je prvočíslo)
```

Pokud Rabin-Miller(n) rozhodne, že n je složené, tak je to zaručeně správný výsledek (byl nalezen „svědek“), pokud Rabin-Miller(n) rozhodne, že n je prvočíslo, tak se může jednat o chybu, ale pouze v případě, že všechna vybraná k_i byli „ne-svědci“ pro složené číslo n, což ale může (díky Větě 2) nastat nejvýše s pravděpodobností

$$P(\text{chyba}) \leq (1/2)^{\text{počet}}$$

pokud jsou výběry jednotlivých k_i vzájemně nezávislé

Vlastnosti algoritmu:

- zvyšováním počtu iterací (počtu testovaných k_i) lze dostat libovolně malou (předem zvolenou) pravděpodobnost chyby
- jednotlivé iterace (testy pro různá k_i) lze provádět paralelně

Časová složitost:

každá iterace trvá jen polynomiálně vzhledem k délce zápisu čísla n (tj. k délce vstupu), k tomu je ovšem potřeba ukázat, že test zda $(k_i,n) \in T$ je možno provést v čase polynomiálním v $\log n$, což není triviální (je nutné mít další znalosti z teorie čísel)

Kryptografie s veřejným klíčem (asymetrickou šifrou)

- každý účastník X má svůj veřejný klíč PX a soukromý klíč SX
- SX je znám pouze X, veřejný klíč PX může X sdělit všem s kterými komunikuje, nebo může být dokonce zveřejněn ve veřejně dostupném seznamu klíčů (třeba na webu)
- oba klíče specifikují funkce, které lze aplikovat na jakoukoli zprávu: tedy pokud D je množina všech konečných posloupností bitů (množina všech možných zpráv), tak obě funkce musí být prosté funkce zobrazující D na D (tj. jsou to permutace množiny D)
- funkci specifikovanou soukromým klíčem SX značíme SX() a funkci specifikovanou veřejným klíčem PX značíme PX(), přičemž předpokládáme, že každá z těchto funkcí je efektivně vyčíslitelná pokud známe příslušný klíč
- funkce SX() a PX() musí tvořit vzájemně inverzní pár funkcí – pro každou zprávu (konečnou posloupnost bitů) M tedy musí platit SX(PX(M)) = M a PX(SX(M)) = M.
- bezpečnost šifry stojí a padá s tím, že nikdo kromě účastníka X není schopen v „rozumném" čase spočítat SX(M) pro jakoukoli zprávu M, což znamená, že
    1. účastník X musí držet klíč SX v absolutním bezpečí před vyzrazením
    2. funkce SX() nesmí být efektivně vyčíslitelná na základě znalosti PX (a schopnosti efektivně vyčíslit funkci PX()), což je hlavní obtíž při návrhu systému šifrování s veřejným klíčem

Předpokládejme, že máme 2 účastnice: A (Alici) a B (Barboru) s klíči SA, PA, SB a PB

Posílání zašifrované zprávy a její rozšifrování

Barbora chce poslat Alici zašifrovanou zprávu M:

• Barbora si opatří Alicin veřejný klíč PA (přímo od Alice či z veřejného seznamu klíčů)

• Barbora spočítá zašifrovaný text C = PA(M) a pošle ho Alici

• Alice na C aplikuje svůj soukromý klíč SA, tedy spočítá SA(C) = SA(PA(M)) = M

• Pokud C zachytí někdo jiný než Alice, nemá šanci získat M, protože neumí efektivně spočítat SA(C).

Posílání autentizované a podepsané (nešifrované) zprávy

Alice chce odpovědět Barboře tak, aby Barbora měla jistotu, že odpověď Q přichází od Alice a že text odpovědi nebyl pozměněn:

• Alice spočítá svůj digitální podpis q pro zprávu Q pomocí svého soukromého klíče, tj. spočítá q = SA(Q) a pošle Barboře dvojici (Q,q) – tj. zpráva Q odchází nešifrovaně

• Barbora spočítá PA(q) = PA(SA(Q)) = Q a porovná to s došlou zprávou Q

• Pokud se obě zprávy zcela shodují, má Barbora jistotu, že zpráva přichází od Alice a nebyla cestou pozměněna

• Pokud se zprávy liší, tak buď je podpis q falešný (nebyl vytvořen funkcí SA()) nebo je podpis pravý ale nezašifrovaná zpráva Q byla cestou pozměněna

Posílání autentizované a podepsané zašifrované zprávy

Alice chce poslat Barboře zprávu M tak, aby Barbora měla jistotu, že M přichází od Alice a že text M nebyl pozměněn. Navíc Alice chce, aby si M mohla přečíst pouze Barbora a nikdo jiný.

- Alice spočítá svůj digitální podpis pro M, tedy spočítá m = SA(M)
- Alice zašifruje dvojici (M,m) pomocí Barbořina veřejného klíče, tedy spočítá zašifrovaný text C = PB(M,m) a pošle C Barboře
- Barbora rozšifruje C pomocí svého soukromého klíče, tedy spočítá SB(C) = (M,m)
- Barbora ověří platnost Alicina podpisu a autenticitu M pomocí Alicina veřejného klíče, tj. spočítá PA(m) a porovná to s M – při neshodě Barbora ví, že buď bylo C cestou změněno (úmyslně či přenosovou chybou) nebo C nepřichází od Alice.

Hybridní šifrování

Pokud je zpráva M, kterou chce Barbora poslat Alici, velmi dlouhá a výpočet C = PA(M) a následně M = SA(C) by trval příliš dlouho, je možné použít šifrování s veřejným klíčem v kombinaci s nějakou symetrickou šifrou K, která šifruje zprávy rychle:

- Barbora spočítá C = K(M), což je opět dlouhá posloupnost bitů, k tomu spočítá PA(K), což je krátká posloupnost bitů (ve srovnání s M a PA(M)) a pošle (C,PA(K)) Alici
- Alice rozšifruje PA(K) pomocí svého SA, takže dostane K, pomocí kterého rozšifruje C a tak získá M

Hybridní autentizace a podepisování

Pro dlouhou zprávu M je také časově náročné počítat digitální podpis m = SA(M). Zde si vypomůžeme (veřejně známou) hashovací funkcí h, která má následující dvě vlastnosti:

1. I pro dlouhé M lze h(M) spočítat velmi rychle, typicky je h(M) krátký (např. 128 bitový) otisk (fingerprint) zprávy M.

2. Je výpočetně velmi obtížné (v rozumném čase nemožné) najít k M jinou zprávu Q takovou, aby platilo h(M) = h(Q)

Pokud chce Alice podepsat dlouhou zprávu posílanou Barboře, může postupovat takto:

• Alice spočítá otisk h(M) zprávy M, udělá z něj digitální podpis m = SA(h(M)) a pošle Barboře dvojici (M,m)

• Barbora obdrží M a také spočítá otisk h(M) který poté porovná s rozšifrovaným Aliciným digitálním podpisem PA(m) = PA(SA(h(M))). Pokud byla M cestou změněna, tak dojde k neshodě, protože díky vlastnosti 2 je těžké pozměnit M tak, aby se její otisk nezměnil.

Certifikační autority

Pokud si Alice pořizuje Barbořin veřejný klíč z veřejně dostupného seznamu (nebo jí ho Barbora posílá po síti), jak může mít jistotu, že nejde o podvrh? Pokud by byl klíč podvržen a následné zprávy modifikovány nebo podvrhovány stejným člověkem, který podvrhl svůj klíč jako Barbořin, tak jejich nepravost nelze zjistit (protože daný člověk bude mít k podvrženému veřejnému klíči i odpovídající soukromý klíč). Řešení:

• Existuje certifikační autorita Z, jejíž veřejný klíč PZ má každý účastník (tedy i Alice) nainstalován u sebe (například přišel na CD s šifrovacím softwarem).

• Barbora pak má od autority Z vydán certifikát ve tvaru C = „Barbořin klíč je PB“ podepsaný autoritou Z, tedy dvojici (C, SZ(C)) – toto může mít Barbora také již od koupě šifrovacího softwaru, nebo certifikát získá jinou bezpečnou cestou

• Tuto dvojici (C, SZ(C)) připojí Barbora ke každé podepisované zprávě, takže Alice (i kdokoli jiný) zjistí pomocí veřejného klíče PZ, že C bylo opravdu vydáno autoritou Z, a že tedy PB opravdu je Barbořin veřejný klíč

RSA (Rivest, Shamir, Adelman) šifra

pro vysvětlení RSA potřebujeme řadu pojmů a tvrzení z teorie čísel

Věta: Nechť a,b jsou přirozená čísla. Pak NSD(a,b) je nejmenší kladný prvek množiny

$$L = \{ax + by \mid x,y \in \mathbf{Z}\}$$

Důsledek: Nechť a,b jsou přirozená čísla. Pokud d je přirozené číslo, které dělí a i b, tak d dělí také NSD(a,b).

Věta: Nechť a,b jsou přirozená čísla, kde b>0. Pak NSD(a,b) = NSD(b, a mod b).

EUCLID(a,b)

if b=0 then Return(a)

else Return(EUCLID(b, a mod b))

Lemma: Nechť $a > b \geq 0$ a EUCLID(a,b) udělá $k \geq 1$ rekurzivních kroků. Pak $a \geq F(k+2)$ a $b \geq F(k+1)$, kde $F(i)$ je i-té Fibonacciho číslo.

Důsledek (Lamého věta): Nechť $a > b \geq 0$ a $F(k) \leq b < F(k+1)$. Pak EUCLID(a,b) udělá nejvýše $k - 1$ rekurzivních kroků.

Věta (bez Dk): $F(k) = \Theta(\varphi^k)$, kde $\varphi = (1+\sqrt{5})/2$ (což je tzv. „zlatý řez").

Důsledek: Nechť $a > b \geq 0$ a $F(k) \leq b < F(k+1)$. Pak EUCLID(a,b) udělá nejvýše $O(\log b)$ rekurzivních kroků.

Pozorování: Pokud a,b jsou dvě nejvýše t-bitová binární čísla, tak EUCLID(a,b) provede $O(t)$ rekurzivních kroků a v každém z nich $O(1)$ aritmetických operací na (nejvýše) t-bitových číslech, tj. $O(t^3)$ bitových operací, pokud předpokládáme, že každá aritmetická operace na t-bitových číslech potřebuje $O(t^2)$ bitových operací (což je snadné ukázat). EUCLID je tedy polynomiální algoritmus vzhledem k velikosti vstupu.

Euklidův algoritmus lze snadno rozšířit tak, aby počítal také koeficienty x,y, pro které $NSD(a,b) = ax + by$.

EXTENDED-EUCLID(a,b)

if $b=0$ then Return(a,1,0)

  else (d',x',y') := EXTENDED-EUCLID(b, a mod b);

    $(d,x,y) := (d',y',x' - \lfloor a/b \rfloor y')$;

    Return(d,x,y)

Věta (bez Dk): Nechť $n > 1$ a $a < n$ jsou dvě nesoudělná přirozená čísla. Pak má rovnice $ax \equiv 1 \pmod{n}$, právě jedno řešení $0 < x < n$ (a pokud jsou a,n soudělná, tak nemá žádné řešení).

Definice: Řešení rovnice $ax \equiv 1 \pmod{n}$ značíme $(a^{-1} \bmod n)$ a nazýváme multiplikativní inverz čísla a modulo n (aby existoval, tak musí být a,n nesoudělná).

Pozorování: $(a^{-1} \bmod n)$ snadno získáme pomocí rozšířeného Euklidova algoritmu.

Věta (speciální důsledek tzv. „čínské věty o zbytcích" – bez Dk): Nechť a,b jsou nesoudělná přirozená čísla. Pak pro každá přirozená čísla x,y platí: $x \equiv y \pmod{ab}$ tehdy a jen tehdy, když $x \equiv y \pmod{a}$ a zároveň $x \equiv y \pmod{b}$.

Věta (malá Fermatova): Nechť p je prvočíslo. Pak $\forall k \in \{1,2, \dots ,p-1\}$ platí $k^{p-1} \equiv 1 \pmod{p}$.

Nyní máme vše co potřebujeme k definici a vysvětlení RSA:

1. Náhodně vyber dvě velká prvočísla p a q (např. každé s 200 binárními ciframi).

2. Spočítej $n = pq$ (v uvedeném případě má n cca 400 binárních cifer).

3. Vyber malé liché číslo e, které je nesoudělné s číslem $(p-1)(q-1)$.

4. Spočítej multiplikativní inverz d čísla e modulo $(p-1)(q-1)$.

5. Zveřejni (e,n) jako veřejný RSA klíč a uschovej (d,n) jako soukromý RSA klíč.

Věta (korektnost RSA): Funkce $P(M) = M^e \bmod n$ a $S(M) = M^d \bmod n$ definují dvojici inverzních funkcí na množině všech zpráv, tj. na množině všech čísel v $Z_n = \{0,1, \dots ,n-1\}$.

Proč je RSA bezpečná?

Na základě (e,n) není (zatím) nikdo schopen spočítat d aniž by znal rozklad $n = pq$ a tím pádem také číslo $(p-1)(q-1)$. A faktorizace velkých čísel je výpočetně těžký problém.

Jak je RSA rychlá?

To, jak rychle lze spočítat $P(M)$ a $S(M)$ závisí na tom, jak rychle umíme počítat zbytek modulo n při umocňování, tj. jak rychle lze spočítat $a^b \bmod n$.

UMOCNI (a,b,n) {kde binární zápis čísla b je $\langle b_k, \ldots, b_0 \rangle$ }

```
c := 1; d := a mod n;
for i := k-1 downto 0 do
   c := 2·c;
   d := (d·d) mod n;
   if b_i = 1 then c := c + 1;
                   d := (d·a) mod n;
Return(d)
```

Časová složitost: Pokud a,b jsou nejvýše t-bitová binární čísla, tak UMOCNI provede $O(t)$ aritmetických operací na (nejvýše) t-bitových číslech, tj. $O(t^3)$ bitových operací, pokud předpokládáme, že každá aritmetická operace na t-bitových číslech potřebuje $O(t^2)$ bitových operací (což je snadné ukázat).