# Diskrétní matematika – 5. týden
# Aplikace teorie čísel – Počítání s velkými čísly, kryptografie

Jan Slovák (výběr z podkladů M. Bulanta)

Masarykova univerzita
Fakulta informatiky

jaro 2017

# Obsah přednášky

# Doporučené zdroje

- Jan Slovák, Martin Panák, Michal Bulant
  **Matematika drsně a svižně**, *e*-text na
  `www.math.muni.cz/Matematika_drsne_svizne`.

# Doporučené zdroje

- Jan Slovák, Martin Panák, Michal Bulant
  **Matematika drsně a svižně**, *e*-text na
  `www.math.muni.cz/Matematika_drsne_svizne`.
- Michal Bulant, výukový text k přednášce **Elementární teorie čísel**, `http://is.muni.cz/el/1431/podzim2012/M6520/um/main-print.pdf`
- Jiří Herman, Radan Kučera, Jaromír Šimša, **Metody řešení matematických úloh**. MU Brno, 2001.
- William Stein, **Elementary Number Theory: Primes, Congruences, and Secrets**, Springer, 2008. Dostupné na `http://wstein.org/ent/ent.pdf`
- Radan Kučera, výukový text k přednášce **Algoritmy teorie čísel**,
  `http://www.math.muni.cz/~kucera/texty/ATC10.pdf`

# Plán přednášky

# Základní úlohy výpočetní teorie čísel

V mnoha praktických úlohách využívajících výsledky teorie čísel je zapotřebí umět rychle provést jeden či více z následujících výpočtů:

1. běžné aritmetické operace (součet, součin, dělení se zbytkem) na celých číslech,

# Základní úlohy výpočetní teorie čísel

V mnoha praktických úlohách využívajících výsledky teorie čísel je zapotřebí umět rychle provést jeden či více z následujících výpočtů:

1. běžné aritmetické operace (součet, součin, dělení se zbytkem) na celých číslech,
2. zbytek mocniny celého čísla $a$ na přirozené číslo $n$ po dělení daným $m$.

# Základní úlohy výpočetní teorie čísel

V mnoha praktických úlohách využívajících výsledky teorie čísel je zapotřebí umět rychle provést jeden či více z následujících výpočtů:

1. běžné aritmetické operace (součet, součin, dělení se zbytkem) na celých číslech,
2. zbytek mocniny celého čísla $a$ na přirozené číslo $n$ po dělení daným $m$.
3. inverzi celého čísla $a$ modulo $m \in \mathbb{N}$,

# Základní úlohy výpočetní teorie čísel

V mnoha praktických úlohách využívajících výsledky teorie čísel je zapotřebí umět rychle provést jeden či více z následujících výpočtů:

1. běžné aritmetické operace (součet, součin, dělení se zbytkem) na celých číslech,
2. zbytek mocniny celého čísla $a$ na přirozené číslo $n$ po dělení daným $m$.
3. inverzi celého čísla $a$ modulo $m \in \mathbb{N}$,
4. největší společný dělitel dvou celých čísel (a případně koeficienty do Bezoutovy rovnosti),

# Základní úlohy výpočetní teorie čísel

V mnoha praktických úlohách využívajících výsledky teorie čísel je zapotřebí umět rychle provést jeden či více z následujících výpočtů:

1. běžné aritmetické operace (součet, součin, dělení se zbytkem) na celých číslech,
2. zbytek mocniny celého čísla $a$ na přirozené číslo $n$ po dělení daným $m$.
3. inverzi celého čísla $a$ modulo $m \in \mathbb{N}$,
4. největší společný dělitel dvou celých čísel (a případně koeficienty do Bezoutovy rovnosti),
5. rozhodnout o daném čísle, je-li prvočíslo nebo složené,

# Základní úlohy výpočetní teorie čísel

V mnoha praktických úlohách využívajících výsledky teorie čísel je zapotřebí umět rychle provést jeden či více z následujících výpočtů:

1. běžné aritmetické operace (součet, součin, dělení se zbytkem) na celých číslech,
2. zbytek mocniny celého čísla $a$ na přirozené číslo $n$ po dělení daným $m$.
3. inverzi celého čísla $a$ modulo $m \in \mathbb{N}$,
4. největší společný dělitel dvou celých čísel (a případně koeficienty do Bezoutovy rovnosti),
5. rozhodnout o daném čísle, je-li prvočíslo nebo složené,
6. v případě složenosti rozložit dané číslo na součin prvočísel.

# Základní aritmetické operace

Základní aritmetické operace se i na velkých číslech obvykle provádějí obdobně jako jsme se to učili na základní a střední škole, kdy umíme sčítat v *lineárním*, násobit a dělit se zbytkem v *kvadratickém* čase.

# Základní aritmetické operace

Základní aritmetické operace se i na velkých číslech obvykle provádějí obdobně jako jsme se to učili na základní a střední škole, kdy umíme sčítat v *lineárním*, násobit a dělit se zbytkem v *kvadratickém* čase. Pro **násobení**, které je základem mnoha dalších operací, existují asymptoticky rychlejší algoritmy (typu *rozděl a panuj*) - např. první takový Karatsubův (1960) časové náročnosti $\Theta(n^{\log_2 3})$

# Základní aritmetické operace

Základní aritmetické operace se i na velkých číslech obvykle provádějí obdobně jako jsme se to učili na základní a střední škole, kdy umíme sčítat v *lineárním*, násobit a dělit se zbytkem v *kvadratickém* čase. Pro **násobení**, které je základem mnoha dalších operací, existují asymptoticky rychlejší algoritmy (typu *rozděl a panuj*) - např. první takový Karatsubův (1960) časové náročnosti $\Theta(n^{\log_2 3})$ nebo algoritmus Schönhage-Strassenův (1971) časové náročnosti $\Theta(n \log n \log\log n)$, který využívá tzv. Fast Fourier Transform. Ten je ale přes svou asymptotickou převahu výhodný až pro násobení čísel majících alespoň desítky tisíc cifer (a používá se tak např. v GIMPS).

# Základní aritmetické operace

Základní aritmetické operace se i na velkých číslech obvykle provádějí obdobně jako jsme se to učili na základní a střední škole, kdy umíme sčítat v *lineárním*, násobit a dělit se zbytkem v *kvadratickém* čase. Pro **násobení**, které je základem mnoha dalších operací, existují asymptoticky rychlejší algoritmy (typu *rozděl a panuj*) - např. první takový Karatsubův (1960) časové náročnosti $\Theta(n^{\log_2 3})$ nebo algoritmus Schönhage-Strassenův (1971) časové náročnosti $\Theta(n \log n \log\log n)$, který využívá tzv. Fast Fourier Transform. Ten je ale přes svou asymptotickou převahu výhodný až pro násobení čísel majících alespoň desítky tisíc cifer (a používá se tak např. v GIMPS).
Pěkný přehled je např. na `http://en.wikipedia.org/wiki/Computational_complexity_of_mathematical_operations`

# GCD a modulární inverze

Jak už jsme ukazovali dříve, výpočet řešení kongruence $a \cdot x \equiv 1 \pmod{m}$ s neznámou $x$ lze snadno (díky Bezoutově větě) převést na výpočet největšího společného dělitele čísel $a$ a $m$ a na hledání koeficientů $k, l$ do Bezoutovy rovnosti $k \cdot a + l \cdot m = 1$ (nalezené $k$ je pak onou hledanou inverzí $a$ modulo $m$).

# GCD a modulární inverze

Jak už jsme ukazovali dříve, výpočet řešení kongruence $a \cdot x \equiv 1 \pmod{m}$ s neznámou $x$ lze snadno (díky Bezoutově větě) převést na výpočet největšího společného dělitele čísel $a$ a $m$ a na hledání koeficientů $k, l$ do Bezoutovy rovnosti $k \cdot a + l \cdot m = 1$ (nalezené $k$ je pak onou hledanou inverzí $a$ modulo $m$).

```
function extended_gcd(a, m)
    if m == 0
        return (1, 0)
    else
        (q, r) := divide (a, m)
        (k, l) := extended_gcd(m, r)
        return (l, k - q * l)
```

Podrobná analýza (viz např. [Knuth] nebo [Wiki]) ukazuje, že tento algoritmus je **kvadratické** časové složitosti.

# Modulární umocňování

Modulární umocňování je, jak jsme již viděli dříve, velmi využívaná operace mj. při ověřování, zda je dané číslo prvočíslo nebo číslo složené. Jedním z efektivních algoritmů je tzv. **modulární umocňování zprava doleva**:

# Modulární umocňování

Modulární umocňování je, jak jsme již viděli dříve, velmi využívaná operace mj. při ověřování, zda je dané číslo prvočíslo nebo číslo složené. Jedním z efektivních algoritmů je tzv. **modulární umocňování zprava doleva**:

```
function modular_pow(base, exponent, modulus)
    result := 1
    while exponent > 0
      if (exponent mod 2 == 1):
        result := (result * base) mod modulus
      exponent := exponent >> 1
      base = (base * base) mod modulus
    return result
```

Algoritmus modulárního umocňování je založen na myšlence, že např. při počítání $2^{64} \pmod{1000}$

- není třeba nejprve počítat $2^{64}$ a poté jej vydělit se zbytkem číslem 1000, ale lépe je postupně násobit „dvojky“ a kdykoliv je výsledek větší než 1000, provést redukci modulo 1000,

Algoritmus modulárního umocňování je založen na myšlence, že např. při počítání $2^{64} \pmod{1000}$

- není třeba nejprve počítat $2^{64}$ a poté jej vydělit se zbytkem číslem 1000, ale lépe je postupně násobit „dvojky“ a kdykoliv je výsledek větší než 1000, provést redukci modulo 1000,
- ale zejména, že není třeba provádět takové množství násobení (v tomto případě 63 naivních násobení je možné nahradit pouze šesti umocněními na druhou, neboť

$$2^{64} = (((((2^2)^2)^2)^2)^2)^2.$$

## Příklad (Ukázka průběhu algoritmu)

Vypočtěme $2^{560} \pmod{561}$.

## Příklad (Ukázka průběhu algoritmu)

Vypočtěme $2^{560} \pmod{561}$. Protože $560 = (1000110000)_2$, dostaneme uvedeným algoritmem

| exponent | base | result | exp's last digit |
|---|---|---|---|
| 560 | 2 | 1 | 0 |
| 280 | 4 | 1 | 0 |
| 140 | 16 | 1 | 0 |
| 70 | 256 | 1 | 0 |
| 35 | 460 | 1 | 1 |
| 17 | 103 | 460 | 1 |
| 8 | 511 | 256 | 0 |
| 4 | 256 | 256 | 0 |
| 2 | 460 | 256 | 0 |
| 1 | 103 | 256 | 1 |
| 0 | 511 | 1 | 0 |

## Příklad (Ukázka průběhu algoritmu)

Vypočtěme $2^{560} \pmod{561}$. Protože $560 = (1000110000)_2$, dostaneme uvedeným algoritmem

| exponent | base | result | exp's last digit |
|---|---|---|---|
| 560 | 2 | 1 | 0 |
| 280 | 4 | 1 | 0 |
| 140 | 16 | 1 | 0 |
| 70 | 256 | 1 | 0 |
| 35 | 460 | 1 | 1 |
| 17 | 103 | 460 | 1 |
| 8 | 511 | 256 | 0 |
| 4 | 256 | 256 | 0 |
| 2 | 460 | 256 | 0 |
| 1 | 103 | 256 | 1 |
| 0 | 511 | 1 | 0 |

A tedy $2^{560} \equiv 1 \pmod{561}$.

# Efektivita modulárního umocňování

V průběhu algoritmu se pro každou binární číslici exponentu provede umocnění základu na druhou modulo $n$ (což je operace proveditelná v nejhůře kvadratickém čase), a pro každou „jedničku“ v binárním zápisu navíc provede jedno násobení. Celkově jsme tedy schopni provést modulární umocňování nejhůře v **kubickém** čase.

# Plán přednášky

# Kryptografie s veřejným klíčem (PKC)

Dva hlavní úkoly pro PKC jsou zajistit

- šifrování, kdy zprávu **zašifrovanou** veřejným klíčem příjemce není schopen rozšifrovat nikdo kromě něj (resp. držitele jeho soukromého klíče)
- podepisování, kdy integrita zprávy **podepsané** soukromým klíčem odesílatele může být ověřena kýmkoliv s přístupem k veřejnému klíči odesílatele

# Kryptografie s veřejným klíčem (PKC)

Dva hlavní úkoly pro PKC jsou zajistit

- šifrování, kdy zprávu **zašifrovanou** veřejným klíčem příjemce není schopen rozšifrovat nikdo kromě něj (resp. držitele jeho soukromého klíče)
- podepisování, kdy integrita zprávy **podepsané** soukromým klíčem odesílatele může být ověřena kýmkoliv s přístupem k veřejnému klíči odesílatele

Nejčastěji používané systémy PKC:

- RSA (šifrování) a odvozený systém pro podepisování zpráv

# Kryptografie s veřejným klíčem (PKC)

Dva hlavní úkoly pro PKC jsou zajistit

- šifrování, kdy zprávu **zašifrovanou** veřejným klíčem příjemce není schopen rozšifrovat nikdo kromě něj (resp. držitele jeho soukromého klíče)
- podepisování, kdy integrita zprávy **podepsané** soukromým klíčem odesílatele může být ověřena kýmkoliv s přístupem k veřejnému klíči odesílatele

Nejčastěji používané systémy PKC:

- RSA (šifrování) a odvozený systém pro podepisování zpráv
- Digital signature algorithm (DSA) a varianta založená na eliptických křivkách (ECDSA)

# Kryptografie s veřejným klíčem (PKC)

Dva hlavní úkoly pro PKC jsou zajistit

- šifrování, kdy zprávu **zašifrovanou** veřejným klíčem příjemce není schopen rozšifrovat nikdo kromě něj (resp. držitele jeho soukromého klíče)
- podepisování, kdy integrita zprávy **podepsané** soukromým klíčem odesílatele může být ověřena kýmkoliv s přístupem k veřejnému klíči odesílatele

Nejčastěji používané systémy PKC:

- RSA (šifrování) a odvozený systém pro podepisování zpráv
- Digital signature algorithm (DSA) a varianta založená na eliptických křivkách (ECDSA)
- Rabinův kryptosystém (a podepisování)

# Kryptografie s veřejným klíčem (PKC)

Dva hlavní úkoly pro PKC jsou zajistit

- šifrování, kdy zprávu **zašifrovanou** veřejným klíčem příjemce není schopen rozšifrovat nikdo kromě něj (resp. držitele jeho soukromého klíče)
- podepisování, kdy integrita zprávy **podepsané** soukromým klíčem odesílatele může být ověřena kýmkoliv s přístupem k veřejnému klíči odesílatele

Nejčastěji používané systémy PKC:

- RSA (šifrování) a odvozený systém pro podepisování zpráv
- Digital signature algorithm (DSA) a varianta založená na eliptických křivkách (ECDSA)
- Rabinův kryptosystém (a podepisování)
- ElGamal kryptosystém (a podepisování)

# Kryptografie s veřejným klíčem (PKC)

Dva hlavní úkoly pro PKC jsou zajistit

- šifrování, kdy zprávu **zašifrovanou** veřejným klíčem příjemce není schopen rozšifrovat nikdo kromě něj (resp. držitele jeho soukromého klíče)
- podepisování, kdy integrita zprávy **podepsané** soukromým klíčem odesílatele může být ověřena kýmkoliv s přístupem k veřejnému klíči odesílatele

Nejčastěji používané systémy PKC:

- RSA (šifrování) a odvozený systém pro podepisování zpráv
- Digital signature algorithm (DSA) a varianta založená na eliptických křivkách (ECDSA)
- Rabinův kryptosystém (a podepisování)
- ElGamal kryptosystém (a podepisování)
- Kryptografie eliptických křivek (ECC)

# Kryptografie s veřejným klíčem (PKC)

Dva hlavní úkoly pro PKC jsou zajistit

- šifrování, kdy zprávu **zašifrovanou** veřejným klíčem příjemce není schopen rozšifrovat nikdo kromě něj (resp. držitele jeho soukromého klíče)
- podepisování, kdy integrita zprávy **podepsané** soukromým klíčem odesílatele může být ověřena kýmkoliv s přístupem k veřejnému klíči odesílatele

Nejčastěji používané systémy PKC:

- RSA (šifrování) a odvozený systém pro podepisování zpráv
- Digital signature algorithm (DSA) a varianta založená na eliptických křivkách (ECDSA)
- Rabinův kryptosystém (a podepisování)
- ElGamal kryptosystém (a podepisování)
- Kryptografie eliptických křivek (ECC)
- Diffie-Hellmanův protokol na výměnu klíčů (DH)

# Princip digitálního podpisu

## Podepisování

1. Vygeneruje se otisk (hash) $H_M$ zprávy pevně stanovené délky (např. 160 nebo 256 bitů).
2. Podpis zprávy $S_A(H_M)$ je vytvořen (pomocí dešifrování) z tohoto hashe s nutností znalosti soukromého klíče podepisujícího.
3. Zpráva $M$ (případně zašifrovaná veřejným klíčem příjemce) je spolu s podpisem odeslána.

# Princip digitálního podpisu

## Podepisování

1. Vygeneruje se otisk (hash) $H_M$ zprávy pevně stanovené délky (např. 160 nebo 256 bitů).
2. Podpis zprávy $S_A(H_M)$ je vytvořen (pomocí dešifrování) z tohoto hashe s nutností znalosti soukromého klíče podepisujícího.
3. Zpráva $M$ (případně zašifrovaná veřejným klíčem příjemce) je spolu s podpisem odeslána.

## Ověření podpisu

1. K přijaté zprávě $M$ se (po jejím případném dešifrování) vygeneruje otisk $H'_M$
2. S pomocí veřejného klíče (deklarovaného) odesílatele zprávy se rekonstruuje původní otisk zprávy $V_A(S_A(H_M)) = H_M$.
3. Oba otisky se porovnají $H_M = H'_M$?.

# RSA

*Ron Rivest, Adi Shamir, Leonard Adleman* (1977; C. Cocks,GCHQ – 1973)

- každý účastník $A$ potřebuje dvojici klíčů – veřejný $V_A$ a soukromý $S_A$

# RSA

*Ron Rivest, Adi Shamir, Leonard Adleman* (1977; C. Cocks,GCHQ – 1973)

- každý účastník $A$ potřebuje dvojici klíčů – veřejný $V_A$ a soukromý $S_A$
- generování klíčů: zvolí dvě velká prvočísla $p, q$, vypočte $n = pq$, $\varphi(n) = (p-1)(q-1)$ [$n$ je veřejné, ale $\varphi(n)$ nelze snadno spočítat ]

# RSA

*Ron Rivest, Adi Shamir, Leonard Adleman* (1977; C. Cocks,GCHQ – 1973)

- každý účastník $A$ potřebuje dvojici klíčů – veřejný $V_A$ a soukromý $S_A$
- generování klíčů: zvolí dvě velká prvočísla $p, q$, vypočte $n = pq$, $\varphi(n) = (p-1)(q-1)$ [$n$ je veřejné, ale $\varphi(n)$ nelze snadno spočítat ]
- zvolí **veřejný klíč** $e$ a ověří, že $(e, \varphi(n)) = 1$

# RSA

*Ron Rivest, Adi Shamir, Leonard Adleman* (1977; C. Cocks,GCHQ – 1973)

- každý účastník $A$ potřebuje dvojici klíčů – veřejný $V_A$ a soukromý $S_A$
- generování klíčů: zvolí dvě velká prvočísla $p, q$, vypočte $n = pq$, $\varphi(n) = (p-1)(q-1)$ [$n$ je veřejné, ale $\varphi(n)$ nelze snadno spočítat ]
- zvolí **veřejný klíč** $e$ a ověří, že $(e, \varphi(n)) = 1$
- např. pomocí Euklidova algoritmu spočítá **tajný klíč** $d$ tak, aby $e \cdot d \equiv 1 \pmod{\varphi(n)}$

# RSA

*Ron Rivest, Adi Shamir, Leonard Adleman* (1977; C. Cocks,GCHQ – 1973)

- každý účastník $A$ potřebuje dvojici klíčů – veřejný $V_A$ a soukromý $S_A$
- generování klíčů: zvolí dvě velká prvočísla $p, q$, vypočte $n = pq$, $\varphi(n) = (p-1)(q-1)$ [$n$ je veřejné, ale $\varphi(n)$ nelze snadno spočítat ]
- zvolí **veřejný klíč** $e$ a ověří, že $(e, \varphi(n)) = 1$
- např. pomocí Euklidova algoritmu spočítá **tajný klíč** $d$ tak, aby $e \cdot d \equiv 1 \pmod{\varphi(n)}$
- zašifrování numerického *kódu* zprávy $M$: $C = C_e(M) \equiv M^e \pmod{n}$

# RSA

*Ron Rivest, Adi Shamir, Leonard Adleman* (1977; C. Cocks,GCHQ – 1973)

- každý účastník $A$ potřebuje dvojici klíčů – veřejný $V_A$ a soukromý $S_A$
- generování klíčů: zvolí dvě velká prvočísla $p, q$, vypočte $n = pq$, $\varphi(n) = (p-1)(q-1)$ [$n$ je veřejné, ale $\varphi(n)$ nelze snadno spočítat ]
- zvolí **veřejný klíč** $e$ a ověří, že $(e, \varphi(n)) = 1$
- např. pomocí Euklidova algoritmu spočítá **tajný klíč** $d$ tak, aby $e \cdot d \equiv 1 \pmod{\varphi(n)}$
- zašifrování numerického *kódu* zprávy $M$: $C = C_e(M) \equiv M^e \pmod{n}$
- dešifrování šifry $C$: $OT = D_d(C) \equiv C^d \pmod{n}$

# Rabinův kryptosystém

Prvním veřejným kryptosystémem, k jehož prolomení je prokazatelně potřeba faktorizovat modul, je **Rabinův kryptosystém**, který si uvedeme ve zjednodušené verzi:

- každý účastník $A$ potřebuje dvojici klíčů – veřejný $V_A$ a soukromý $S_A$

# Rabinův kryptosystém

Prvním veřejným kryptosystémem, k jehož prolomení je prokazatelně potřeba faktorizovat modul, je **Rabinův kryptosystém**, který si uvedeme ve zjednodušené verzi:

- každý účastník $A$ potřebuje dvojici klíčů – veřejný $V_A$ a soukromý $S_A$
- generování klíčů: $A$ zvolí dvě podobně velká prvočísla $p, q \equiv 3 \pmod 4$, vypočte $n = pq$.

# Rabinův kryptosystém

Prvním veřejným kryptosystémem, k jehož prolomení je prokazatelně potřeba faktorizovat modul, je **Rabinův kryptosystém**, který si uvedeme ve zjednodušené verzi:

- každý účastník $A$ potřebuje dvojici klíčů – veřejný $V_A$ a soukromý $S_A$
- generování klíčů: $A$ zvolí dvě podobně velká prvočísla $p, q \equiv 3 \pmod 4$, vypočte $n = pq$.
- $V_A = n$, $S_A = (p, q)$

# Rabinův kryptosystém

Prvním veřejným kryptosystémem, k jehož prolomení je prokazatelně potřeba faktorizovat modul, je **Rabinův kryptosystém**, který si uvedeme ve zjednodušené verzi:

- každý účastník $A$ potřebuje dvojici klíčů – veřejný $V_A$ a soukromý $S_A$
- generování klíčů: $A$ zvolí dvě podobně velká prvočísla $p, q \equiv 3 \pmod 4$, vypočte $n = pq$.
- $V_A = n$, $S_A = (p, q)$
- zašifrování numerického *kódu* zprávy $M$: $C = C_e(M) \equiv M^2 \pmod n$

# Rabinův kryptosystém

Prvním veřejným kryptosystémem, k jehož prolomení je prokazatelně potřeba faktorizovat modul, je **Rabinův kryptosystém**, který si uvedeme ve zjednodušené verzi:

- každý účastník $A$ potřebuje dvojici klíčů – veřejný $V_A$ a soukromý $S_A$
- generování klíčů: $A$ zvolí dvě podobně velká prvočísla $p, q \equiv 3 \pmod 4$, vypočte $n = pq$.
- $V_A = n$, $S_A = (p, q)$
- zašifrování numerického *kódu* zprávy $M$: $C = C_e(M) \equiv M^2 \pmod n$
- dešifrování šifry $C$: vypočtou se (čtyři) odmocniny z $C$ modulo $n$ a snadno se otestuje, která z nich byla původní zprávou.

## Výpočet druhé odmocniny z $C$ modulo $n = pq$, kde $p \equiv q \equiv 3 \pmod 4$

- vypočti $r = C^{(p+1)/4} \pmod p$ a $s = C^{(q+1)/4} \pmod q$

[a] Uvědomte si, že jde vlastně o aplikaci Čínské zbytkové věty!

## Výpočet druhé odmocniny z $C$ modulo $n = pq$, kde $p \equiv q \equiv 3 \pmod 4$

- vypočti $r = C^{(p+1)/4} \pmod p$ a $s = C^{(q+1)/4} \pmod q$
- vypočti $a, b$ tak, že $ap + bq = 1$

[a] Uvědomte si, že jde vlastně o aplikaci Čínské zbytkové věty!

## Výpočet druhé odmocniny z $C$ modulo $n = pq$, kde $p \equiv q \equiv 3 \pmod 4$

- vypočti $r = C^{(p+1)/4} \pmod p$ a $s = C^{(q+1)/4} \pmod q$
- vypočti $a, b$ tak, že $ap + bq = 1$
- polož[a] $x = (aps + bqr) \pmod n$, $y = (aps - bqr) \pmod n$

[a] Uvědomte si, že jde vlastně o aplikaci Čínské zbytkové věty!

## Výpočet druhé odmocniny z $C$ modulo $n = pq$, kde $p \equiv q \equiv 3 \pmod 4$

- vypočti $r = C^{(p+1)/4} \pmod p$ a $s = C^{(q+1)/4} \pmod q$
- vypočti $a, b$ tak, že $ap + bq = 1$
- polož[a] $x = (aps + bqr) \pmod n$, $y = (aps - bqr) \pmod n$
- druhými odmocninami z $C$ modulo $n$ jsou $\pm x$, $\pm y$.

[a] Uvědomte si, že jde vlastně o aplikaci Čínské zbytkové věty!

**Příklad**

V Rabinově kryptosystému Alice zvolila za svůj soukromý klíč $p = 23, q = 31$, veřejným klíčem je pak $n = pq = 713$. Zašifrujte zprávu $m = 327$ pro Alici a ukažte, jak bude Alice tuto zprávu dešifrovat.

**Příklad**

V Rabinově kryptosystému Alice zvolila za svůj soukromý klíč $p = 23, q = 31$, veřejným klíčem je pak $n = pq = 713$. Zašifrujte zprávu $m = 327$ pro Alici a ukažte, jak bude Alice tuto zprávu dešifrovat.

**Řešení**

$c = 692$, kandidáti původní zprávy jsou $\pm 4 \cdot 23 \cdot 14 \pm 3 \cdot 31 \cdot 18 \pmod{713}$.

# Diffie-Hellman key exchange

*Whitfield Diffie, Martin Hellman* (1976; M. Williamson, GCHQ - 1974)
Výměna klíčů pro symetrickou kryptografii bez předchozího kontaktu (tj. náhrada jednorázových klíčů, kurýrů s kufříky, . . . ).

# Diffie-Hellman key exchange

*Whitfield Diffie, Martin Hellman* (1976; M. Williamson, GCHQ - 1974)
Výměna klíčů pro symetrickou kryptografii bez předchozího kontaktu (tj. náhrada jednorázových klíčů, kurýrů s kufříky, . . . ).

- Dohoda stran na **prvočísle** $p$ a primitivním kořenu $g$ modulo $p$ (veřejné)

# Diffie-Hellman key exchange

*Whitfield Diffie, Martin Hellman* (1976; M. Williamson, GCHQ - 1974)
Výměna klíčů pro symetrickou kryptografii bez předchozího kontaktu (tj. náhrada jednorázových klíčů, kurýrů s kufříky, . . . ).

- Dohoda stran na **prvočísle** $p$ a primitivním kořenu $g$ modulo $p$ (veřejné)
- Alice vybere náhodné $a$ a pošle $g^a \pmod{p}$

# Diffie-Hellman key exchange

*Whitfield Diffie, Martin Hellman* (1976; M. Williamson, GCHQ - 1974)
Výměna klíčů pro symetrickou kryptografii bez předchozího kontaktu (tj. náhrada jednorázových klíčů, kurýrů s kufříky, . . . ).

- Dohoda stran na **prvočísle** $p$ a primitivním kořenu $g$ modulo $p$ (veřejné)
- Alice vybere náhodné $a$ a pošle $g^a \pmod{p}$
- Bob vybere náhodné $b$ a pošle $g^b \pmod{p}$

# Diffie-Hellman key exchange

*Whitfield Diffie, Martin Hellman* (1976; M. Williamson, GCHQ - 1974)
Výměna klíčů pro symetrickou kryptografii bez předchozího kontaktu (tj. náhrada jednorázových klíčů, kurýrů s kufříky, . . . ).

- Dohoda stran na **prvočísle** $p$ a primitivním kořenu $g$ modulo $p$ (veřejné)
- Alice vybere náhodné $a$ a pošle $g^a \pmod{p}$
- Bob vybere náhodné $b$ a pošle $g^b \pmod{p}$
- Společným klíčem pro komunikaci je $g^{ab} \pmod{p}$.

# Diffie-Hellman key exchange

*Whitfield Diffie, Martin Hellman* (1976; M. Williamson, GCHQ - 1974)
Výměna klíčů pro symetrickou kryptografii bez předchozího kontaktu (tj. náhrada jednorázových klíčů, kurýrů s kufříky, . . . ).

- Dohoda stran na **prvočísle** $p$ a primitivním kořenu $g$ modulo $p$ (veřejné)
- Alice vybere náhodné $a$ a pošle $g^a \pmod{p}$
- Bob vybere náhodné $b$ a pošle $g^b \pmod{p}$
- Společným klíčem pro komunikaci je $g^{ab} \pmod{p}$.

# Diffie-Hellman key exchange

*Whitfield Diffie, Martin Hellman* (1976; M. Williamson, GCHQ - 1974)
Výměna klíčů pro symetrickou kryptografii bez předchozího kontaktu (tj. náhrada jednorázových klíčů, kurýrů s kufříky, . . . ).

- Dohoda stran na **prvočísle** $p$ a primitivním kořenu $g$ modulo $p$ (veřejné)
- Alice vybere náhodné $a$ a pošle $g^a \pmod p$
- Bob vybere náhodné $b$ a pošle $g^b \pmod p$
- Společným klíčem pro komunikaci je $g^{ab} \pmod p$.

**Poznámka**

- Problém diskrétního logaritmu (DLP)
- Nezbytná autentizace (*man in the middle attack*)

# Kryptosystém ElGamal

Z protokolu DH na výměnu klíčů odvozen šifrovací algoritmus ElGamal:

- Alice zvolí prvočíslo $p$ spolu s primitivním kořenem $g$
- Alice zvolí **tajný klíč** $x$, spočítá $h = g^x \pmod{p}$ a zveřejní **veřejný klíč** $(p, g, h)$
- šifrování zprávy $M$: Bob zvolí náhodné $y$ a vypočte $C_1 = g^y \pmod{p}$ a $C_2 = M \cdot h^y \pmod{p}$ a pošle $(C_1, C_2)$
- dešifrování zprávy: $OT = C_2 / C_1^x$

# Kryptosystém ElGamal

Z protokolu DH na výměnu klíčů odvozen šifrovací algoritmus ElGamal:

- Alice zvolí prvočíslo $p$ spolu s primitivním kořenem $g$
- Alice zvolí **tajný klíč** $x$, spočítá $h = g^x \pmod{p}$ a zveřejní **veřejný klíč** $(p, g, h)$
- šifrování zprávy $M$: Bob zvolí náhodné $y$ a vypočte $C_1 = g^y \pmod{p}$ a $C_2 = M \cdot h^y \pmod{p}$ a pošle $(C_1, C_2)$
- dešifrování zprávy: $OT = C_2 / C_1^x$

**Poznámka**

Analogicky jako v případě RSA lze odvodit podepisování.