KATEDRA
INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI

KMI/KRY - Kryptografie

Poznámky z výuky *(2025 – 2026)*
Verze z *22. January 2026*

Vojtěch Netrh
vojtanetrh@gmail.com

Obsah

1 Poznámky k zakončení předmětu

Zkouška je ústní (takže asi klasické losování otázek). Otázky jsou brány jako názvy kapitol. Jedna otázka z první prezentace, další z druhé. Možnost nějakých důkazů (RSA nebo EC třeba). Zápočet se řeší s Mgr. Foltasovou a bude za odevzdání programovacích úloh (implementace algoritmů). Přednáška není od 7.00, ale od 8.00, ale končí stejně tzn. v 9.30. Materiály budou komunikovány e-mailem.

1. Týden (24. 9. 2025) – slajdy 1-32
2. Týden (1. 10. 2025) – slajdy 33-72
3. Týden (8. 10. 2025) – slajdy 73-89
4. Týden (15. 10. 2025) – ???
5. Týden (22. 10. 2025) – ???
6. Týden (29. 10. 2025) – ???
7. Týden (5. 11. 2025) – ???
8. Týden (12. 11. 2025) – slajdy 160-182
9. Týden (19. 11. 2025) – slajdy 183-210
10. Týden (26. 11. 2025) – další prezentace (slides-2) slajdy 1-28

2 Kryptografie

Jedná se o vědu o utajování zpráv. Moderní metody zajišťují následující:

- **důvěrnost dat**: utajení obsahu komunikace
- **autentičnost**: příjemce má možnost zjistit původ zprávy
- **neodmítnutelnost**: odesílatel nemůže popřít odeslání zprávy
- **integritu zprávy**: příjemce má možnost zjistit zda během přenosu došlo ke změně zprávy

Další pojmy, které jsou nutné k pochopení šifer a komunikace:

- **otevřený text (plain text)**: zpráva určená k odeslání
- **šifrování**: úprava textu, která ukryje jeho obsah (není srozumitelný)
- **zašifrovaný text (ciphertext)**: výsledek aplikace šifrování
- **dešifrování**: převod šifrovaného textu zpět na otevřený
- **šifrovací funkce (encryption function)**: matematická funkce provádějící šifrování
- **dešifrovací funkce (decryption function)**: matematická funkce provádějící dešifrování
- **šifra**: označení pro šifrovací i dešifrovací funkci
- **kanál (channel)**: komunikační spoj

3 Šifry

Máme 2 základní typy šifer rozvedené dále – omezené šifry a šifry založené na klíči. Šifrování založené na klíče se dále dělí na: symetrické, asymetrické a hybridní (toto rozdělení závisí na vztahu klíčů).

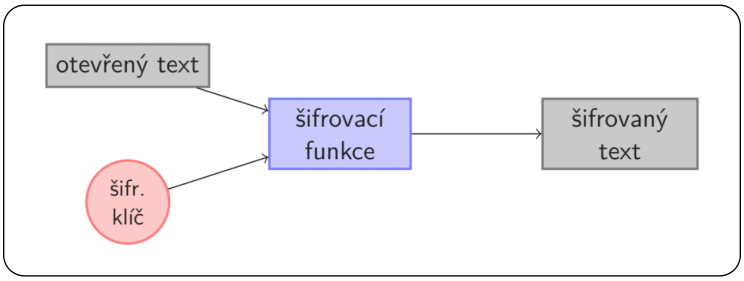
3.1 Omezené šifry

Míra bezpečnosti se zakládá na tom jakým způsobem se šifra (její (de)šifrovací funkce) pracuje. Prakticky toto není moc výhodné a použitelné (může dojít k odhalení principu, při odchodu uživatel ze skupiny nutno vyměnit funkci, nemožnost standardizace, …). první šifra, která tento typ překonala byla *Enigma*.

3.2 Šifry založené na klíči

Používá se tzv. **Kerchoffův princip**. Bezpečnost závisí pouze na utajení klíče nikoliv utajení (de)šifrovací funkce. Funkce pak může být zveřejněna, což přináší další výhody (standardizace). Šifer založených na klíči máme 3 typy – symetrické, asymetrické a hybridní.

Kromě utajení zpráv řeší i jiné problémy např. autentizaci.



Obrázek 1: (De)šifrovací proces

Několik základních pojmů:

- $\mathcal{M}$ … konečná množina všech zpráv
- $\mathcal{C}$ … konečná množina všech kryptogramů
- $\mathcal{K}$ … konečná množina všech klíčů
- $e : \mathcal{M} \times \mathcal{K} \to \mathcal{C}$ … šifrovací funkce
- $d : \mathcal{C} \times \mathcal{K} \to \mathcal{M}$ … dešifrovací funkce

Šifra podle Clauda Shannona **Definice 1**

Šifra založená na klíči je pětice $\langle \mathcal{M}, \mathcal{C}, \mathcal{K}, e, d \rangle$ taková, že pro libovolný šifrovací klíč $k_e \in \mathcal{K}$ a jemu odpovídající dešifrovací klíč $k_d \in \mathcal{K}$ platí

$$d(e, (x, k_e), k_d) = x$$

pro všechna $x \in \mathcal{M}$. Krátce budeme mluvit o šifře.

Tato definice neříká nic o bezpečnosti. Klidně by e a d mohla být identita a definice by byla splněna.

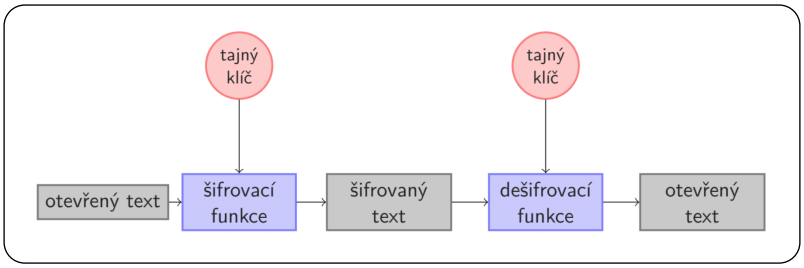
Funkce také musí být injektivní. Pokud by nebyla, způsobilo by to problém, že bychom nebyli schopni zprávu jednoznačně rozšifrovat (ze 2 různých zpráv bychom totiž dostali stejný kryptogram).

Šifrovací a dešifrovací funkce obvykle zprávu zakódují do posloupnosti čísel, s kterou poté manipulují pomocí klasikcýh matematických operací (sčítání, násobení, …). Pro definování těchto operacích na konečných množinách se používá **modulární aritmetika**.

3.2.1 Symetrické šifry (private-key cryptography)

Klíč pro šifrování a dešifrování je stejný (nebo je mezi nimi jednoduchý vztah). Primárně to jsou všechny historické šifry. Alice a Bob mají stejný klíč, kterým umí oba šifrovat i dešifrovat. Např. klasické (posouvací, Vigenerova), RC2, DES, AES, Blowfish, …

V tomto případě mluvíme o **tajném klíči** (neplést se soukromým u asymetrické šifry).



Obrázek 2: Symetrické šifrování

Postup

1. Alice a Bob se domluví na klíči
2. Alice zprávu pomocí klíče zašifruje
3. Šifrovaná zpráva může být přes nezabezpečený kanál poslána Bobovi
4. Bob zprávu dešifruje pomocí stejného klíče

\+ **Výhody**: vysoká rychlost

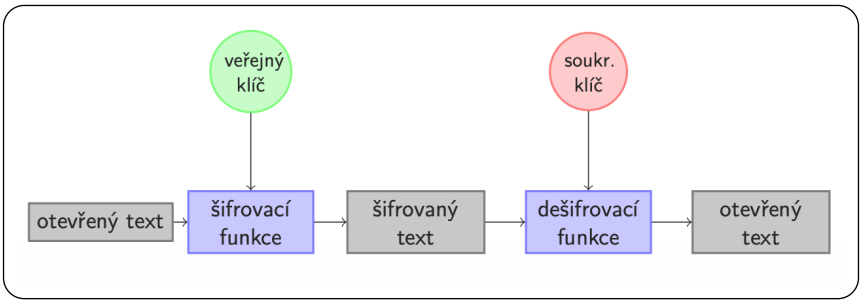
− **Nevýhody**: nebezpečí odhalení klíče 3. stranou (náročnější distribuce), velký počet klíčů (složitý key management) rostoucí kvadraticky $\binom{n}{2}$.

Autentizace za pomocí symetrického šifrování

Částečně brání útoku MITM (man in the middle). Odesílá se jak zašifrovaná zpráva, tak otevřená a jejich porovnáním můžeme ověřit že nebyla po cestě změněna. Pořád ale může útočník menit pořadí více zpráv, různě prohazovat v párech atd.

3.2.2 Asymetrické šifry (public-key cryptography)

Vymyšleno v roce 1976 3 lidmi: Martin Hellman, Ralph Merkle a Whitfield Diffie. Klíče pro šifrování (veřejný klíč) a dešifrování (soukromý klíč) nejsou stejné. Soukromý klíč nesmí být z veřejného rozumné době odvoditelný (opačně to jde). Z logiky věci veřejný může být distribuován a soukromý musí být držen v tajnosti. Např. RSA, šifrování založené na eliptických křivkách, šifrování založené na zavazadlovém problému.



Obrázek 3: Asymetrické šifrování

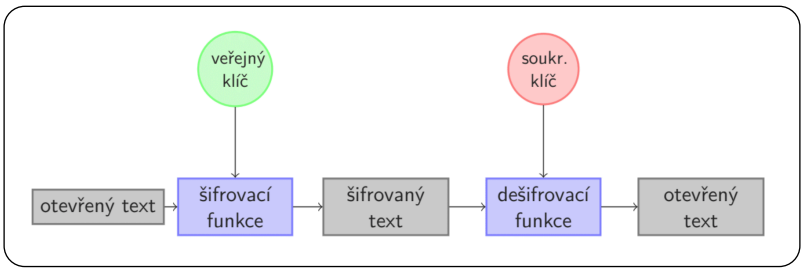
\+ **Výhody**: bezpečnější, není potřeba správa klíčů.

− **Nevýhody**: výrazně pomalejší (o několik řádů).

Postup

1. příjemce vytvoří soukromý a veřejný klíč
2. soukromý si uschová, veřejný zveřejní
3. odesílatel zprávu zašifruje pomocí veřejného klíče
4. odeslání šifrované zprávy nezabezpečeným kanálem

5. příjemce dešifruje pomocí svého soukromého klíče



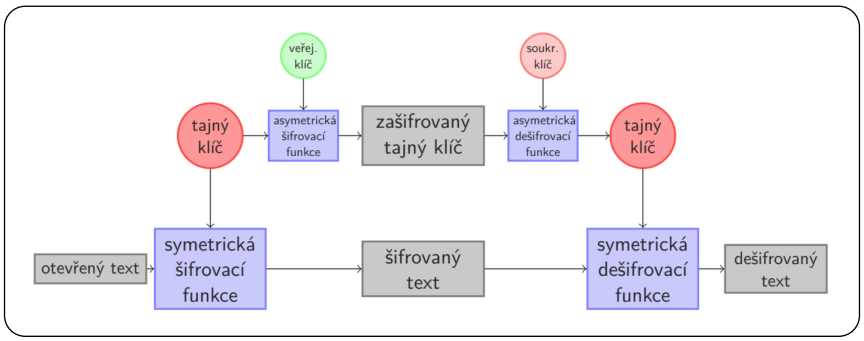
Obrázek 4: Asymetrické šifrování

Vytváření veřejného a soukromého klíče

[TBA]

3.2.3 Hybridní šifry

Kombinuje symetrické a asymetrické šifrování a bere si to lepší z obou. Tím že tajný klíč je krátký, tak nízká rychlost asymetrické šifry není znát. Používá se třeba v protokolech TSL a SSL.



Obrázek 5: Hybridní šifrování

Prvním praktickým použitím hybridního šifrování bylo **Pretty good privacy (PGP)** používající šifry IDEA a RSA.

3.2.4 Protokoly SSL a TLS

Jedná se o kryptografické protokoly zajišťující bezpečnou komunikaci na internetu (dříve používáno v HTTPS).

Při zahájení komunikace zde probíhá handshake, ve kterém se používají digitální certifikáty (jméno serveru, veřejný klíč, certifikační autorita). Klient může ověřit platnost tohoto certifikátu. Klient podporuje různé šifrovací algoritmy a hashovací funkce, z kterých poté server vybírá.

3.3 Modulární aritmetika

Důležité pojmy:

- prvočísla
- rozklad na prvočísla
- Euklidův algoritmus
- největší společný dělitel (a jeho vlastnosti)
- nesoudělnost
- kongruence modulo n

- množiny zbytkových tříd $\mathbb{Z}_n = \{[a]_n \mid a \in \mathbb{Z}\}$
- inverzní prvky
- číselná tělesa (grupa, monoid, …)
- eulerova funkce
- Bezoutova rovnost

4 základních vlastností největšího společného dělitele:

$$\begin{aligned}
&\gcd(a,b) = \gcd(b,a)\\
&\gcd(a,b) = \gcd(-a,b)\\
&\gcd(a,0) = |a|\\
&\gcd(a,b) = \gcd(a - k \cdot b, b)
\end{aligned}$$

Čínská věta o zbytcích[1] **Theorem 2**

Nechť jsou $n_1, n_2, ..., n_k$ po dvou nesoudělná nenulová přirozená čísla, tzn. $\gcd(n_i, n_j) = 1$ pro $i \neq j$. Dále nechť je $a_1, a_2, ..., a_k \in \mathbb{N}$. Pak systém konfigurací

$$\begin{aligned}
x &\equiv a_1 (\text{mod } n_1)\\
x &\equiv a_2 (\text{mod } n_2)\\
&\vdots\\
x &\equiv a_k (\text{mod } n_k)
\end{aligned}$$

je řešitelný. Jsou-li c a c' řešení tohoto systému, pak platí

$$c \equiv c' (\text{mod } n_1 \cdot n_2 \ ... \ n_k)$$

[Tady by se dala napsat řada dalších věcí, které jsou zmíněny především v prezentacích nebo je známe z jiných předmětů.]

4 Klasické šifry

Jedná se o symetrické šifry. Pro zjednodušení používáme anglickou abecedu s 26 písmeny (A, B, …, Z), která jsou kódovány pořadovým číslem – A → 0, B → 1, …, Z → 25. Počítáme tedy v $\mathbb{Z}_{26}$, ale obecně všechny šifry se dají zobecnit na abecedy s n symboly (tedy $\mathbb{Z}_n$).

Máme dva typy dělení šifer na:
1. monoabecední
2. polyabecední

a nebo
1. blokové šifry
2. proudové šifry

4.1 Monoabecední šifry

Prvky množin $\mathcal{M}$ a $\mathcal{C}$ jsou jednotlivé symboly abecedy. Tedy jediný symbol abecedy je mapován šifrovací funkcí na jediný symbol. Například Caesarova šifra.

[1] Tato věta se používá při rychlém umocnění v RSA.

4.1.1 Posouvací šifra

Písmeno je posunuto o určitý počet pozic, který je dán klíčem (v Caesarově šifře je $k = 3$). Kryptoanalýzu a prolomení provádíme hrubou silou. Tím, že existuje pouze 26 klíčů, tak v průměru klíč odhalíme po 13 pokusech ($\frac{26}{2}$), tedy šifra není bezpečná.

Posouvací šifra **Definice 3**

Nechť $\mathcal{M} = \mathcal{C} = \mathcal{K} = \mathbb{Z}_{26}$. Pro $k \in \mathcal{K}$ definujeme šifrovací funkci

$$e(x, k) = x + k$$

a dešifrovací funkci

$$d(y, k) = y - k.$$

4.1.2 Afinní šifra

Mapování se provádí pomocí funkce $e(x, k) = ax + b$, kde $k = \langle a, b \rangle$. Funkce musí být injektivní. Z definice si všimneme, že jde o speciální případ posouvací šifry ($a = 1, k = b$). Jestliže bereme $\langle a, b \rangle \in \mathbb{Z}_{26} \times \mathbb{Z}_{26}$, pak existuje 312 různých klíčů (velmi málo), protože a musí být nesoudělné s 26 (viz podmínka *gcd*) a b vybíráme libovolně $=> 26 \cdot \varphi(26) = 26 \cdot 12$.

Afinní šifra **Definice 4**

Nechť $\mathcal{M} = \mathcal{C} = \mathbb{Z}_{26}, \mathcal{K} = \{\langle a, b \rangle \in \mathbb{Z}_{26} \times \mathbb{Z}_{26} \mid \gcd(a, 26) = 1\}$. Pro $k = \langle a, b \rangle \in \mathcal{K}$ definujeme šifrovací funkci

$$e(x, k) = ax + b$$

a dešifrovací funkci

$$d(y, k) = a^{-1}(y - b).$$

Injektivnost afinní šifry **Poznámka 5**

Funkce $e(x, k) = ax + b$ je injektivní, jestliže $\gcd(a, 26) = 1$.

4.1.3 Substituční šifra

Písmeno abecedy je mapováno na jiné písmeno stejné abecedy podle dané permutace této abecedy. Připomenout si co je to permutace. Obě předešlé šifry jsou speciálním případem substituční. Je zhruba $4 \cdot 10^{26}$ klíčů, což je sice hodně (nestačí brute-force útok), ale pořád není bezpečná (máme i jiné metody než hrubou sílu viz dále).

Substituční šifra **Definice 6**

Nechť $\mathcal{M} = \mathcal{C} = \mathbb{Z}_{26}, \mathcal{K} = \{\pi \mid \pi \text{ je permutace } 26\}$. Pro $\pi \in \mathcal{K}$ definujeme šifrovací funkci

$$e(x, \pi) = \pi(x)$$

a dešifrovací funkci

$$d(y, \pi) = \pi^{-1}(y).$$

4.2 Polyabecední šifry

Prvky množin $\mathcal{M}$ a $\mathcal{C}$ jsou posloupnosti symbolů abecedy určité délky. Tedy symbol abecedy je mapován na jeden z několika symbolů. Například Vigenèrova šifra. Jsou o něco silnější (bezpečnější).

4.2.1 Vigenèrova šifra

Polyabecední šifra. Existuje 26^m různých klíčů délky m, ale pořád není bezpečná (opět kvůli jiným metodám kryptoanalýzy).

Vigenèrova šifra **Definice 7**

Nechť $\mathcal{M} = \mathcal{C} = \mathcal{K} = \mathbb{Z}_{26}^m,\ m \in \mathbb{N}$. Pro klíč $k = \langle k_1, ..., k_m \rangle \in \mathcal{K}$ definujeme šifrovací funkci

$$e(x, k) = \langle x_1 + k_1, ..., x_m + k_m \rangle$$

a dešifrovací funkci

$$d(y, k) = \langle y_1 - k, ..., y_m - k_m \rangle.$$

Pro všechna $x = \langle x_1, ..., x_m \rangle \in \mathcal{M}$, $y = \langle y_1, ..., y_m \rangle \in \mathcal{C}$.

Proudové šifry

Pomocí klíče $k \in \mathcal{K}$ je generován klíčový proud (*keystream*) $z = z_1 z_2 \cdots z_n$. Takže pak pro každý znak se používá jiný klíč právě z onoho proudu. Generování klíčového proudu může být **synchronní** – generován nezávisle na šifrovaném i původním textu nebo **asynchorní** – využije se šifrovaný nebo původní text.

Blokové šifry

Všechny předchozí jsou blokové. Neboli všechny prvky z $\mathcal{M}$ jsou šifrovány pomocí jednoho klíče.

4.3 Proudové šifry

Všechny do teď zmíněné šifry byly blokové, což znamená že všechny prvky z otevřené zprávy jsou šifrovány pomocí jednoho klíče. Zatímco u proudových šifer je generován *klíčový proud (keystream)*. Může být generován synchronně (nezávisle na šifrovaném i původním textu) nebo asyncrhonně (s využitím šifrovaného nebo původního textu).

Klíčový proud **Poznámka 8**

Klíčový proud je něco jako potenciálně nekonečný řetězec.

Několik základních pojmů:

- $\mathcal{M}$ … množina zpráv
- $\mathcal{C}$ … množina kryptogramů
- $\mathcal{K}$ … množina klíčů
- $\mathcal{L}$ … abeceda klíčového proudu
- $g : \mathcal{K} \to \mathcal{L}^*$ … generátor klíčového proudu
- $e : \mathcal{M} \times \mathcal{L} \to \mathcal{C}$ … šifrovací funkce
- $d : \mathcal{C} \times \mathcal{L} \to \mathcal{M}$ … dešifrovací funkce

$$g(k) = k \cdot k \cdot k...$$

Proudová šifra **Definice 9**

Proudová šifra je sedmice $\langle \mathcal{M}, \mathcal{C}, \mathcal{K}, \mathcal{L}, g, e, d \rangle$ taková, že pro libovolné $z \in \mathcal{L}$

$$d(e(x, z), z) = x$$

pro všechna $x \in \mathcal{M}$.

4.3.1 Linear feedback shift

Metoda generování klíčového proudu (dříve často ve vojenských). Pracuje nad abecedou

$$\mathbb{Z}_2 : \mathcal{K} = \mathbb{Z}_2^{2m}, m \in \mathbb{N}, \mathcal{L} = \mathbb{Z}_2.$$

Klíč je složen $2m$ hodnot ($k = \langle k_1, ..., k_m, c_0, ..., c_{m-1} \rangle$). Pro daný klíč k rekurzivně vygenerujeme klíčový proud $z = z_1 z_2 \cdots$

$$z_i = k_i \text{ pro } 1 \leq i \leq m$$

$$z_{i+m} = \sum_{j=0}^{m-1} c_j \cdot z_{i+j} \quad \text{pro} \quad i \geq 1$$

Pro komponenty klíč nesmí platit, že buď se sobě nesmí rovnat k a taky nemusí být 0 nebo c (stejně i s nulou).

Hardware implementace pomocí *linear feedback shift registru* (LFSR) a následujících 6 krocích

1. inicializace bitů registru
2. bit b_1 se použije jako další hodnota klíčového proudu
3. počítá se lineární kombinace $B = c_0 b_1 + c_1 b_2 + ... + c_{m-1} b_m$
4. bity $b_2, ..., b_m$ se posunou o 1 pozici doleva
5. poslednímu bitu b_m se přiřadí vypočítaná hodnota B
6. pokračuj na kroku 2

Výpočet linear feedback shift **Příklad 10**

Uvažujme $m = 4$ a $k = \langle 1, 0, 0, 0, 1, 1, 0, 0 \rangle$. Vypočítejte několik prvních hodnot klíčového proudu (to je to $\boldsymbol{z}$).

Kdy se hodnoty začnou opakovat? *Po 15 číslech. Podle Chat GPT je* $2^m - 1$ *maximální délka.*
Jak vypadá LSFR?

4.3.2 Asynchornní proudové šifry

Každý prvek (kromě prvního) klíčového proudu je vypočítán za pomocí prvku předchozího. Výhodou je, že se hodnoty nebudou opakovat.

Asynchornní proudové šifry **Definice 11**

Nechť $\mathcal{M} = \mathcal{C} = \mathcal{K} = \mathcal{L} = \mathbb{Z}_{26}$. Autokey šifry je proudová šifra taková, že

- pro klíč $k \in \mathcal{K}$ a zprávu $x_1 x_2 \cdots$ platí $g(k) = z_1 z_2 \cdots$, kde $z_1 = k$ a $z_i = x_{i-1}$ pro $i \geq 2$,
- pro libovolnou hodnotu $z \in \mathcal{L}$ definujeme šifrovací funkci

$$e(x, z) = x + z$$

a dešifrovací funkci

$$d(y, z) = y - z$$

pro všechna $x \in \mathcal{M}, y \in \mathcal{C}$.

5 Kryptoanalýza klasických šifer

Pokud víme jak šifra funguje je její prolomení výrazně snazší, tzv. *Kerckhoffův princip*. Logicky tím neuvažujeme prolomení omezené šifry. Máme několik typů útoků, v závislosti na tom jaké informace jsou k dispozici. Ve všech případech je **snahou získat klíč**.

1. **Ciphertext only attack** – známe zašifrovanou zprávy
2. **Known plaintext attack** – známe původní i jí odpovídající zašifrovanou zpráv
3. **Chosen plaintext attack** – dočasný přístup k šifrovacímu kryptografickému modulu → můžeme vybrat libovolnou zprávu a tu zašifrovat
4. **Chosen ciphertext attack** – dočasný přístup k dešifrovacímu kryptografickému modulu → můžeme vybrat libovolný kryptogram a ten rozšifrovat

Předpoklad. Často je analýza založena na statistických vlastnostech daného jazyka. Uvažujeme anglický jazyk (s 26 znaky) a neuvažujeme mezery (je to snazší).

Statistické vlastnosti angličtiny **Poznámka 12**

Beker a Pipe pro anglický jazyk zjistili následující vlastnosti:

- písmeno E se vyskytuje s pravděpodobností asi 0,12
- písmena T, A, O, I, N, S, H, R mezi 0,06 a 0,08
- písmena D, L asi 0,04
- C, U, M, W, F, G, Y, P, B mezi 0,015 a 0,028
- V, K, J, X, Q, Z asi 0,01
- nejpoužívanější digramy: TH, HE, IN, ER, AN, RE, ED, ON, ES, TS, EN, AT, TO, NT, HA, ND, OU, EA, NG, AS, OR, TI, IS, ET, IT, AR, TE, SE, HI, OF,…
- nejpoužívanější trigramy: THE, ING, AND, HER, ERE, ENT, THA, NTH, WAS, ETH, FOR, DTH, …

Postup. Najdeme v kryptogramu znak s největší četností, ten je pravděpodobně zašifrovaným písmenem E. Obdobně pokračujeme pro další skupiny písmen, případně i digramy a trigramy.

5.1 Kryptoanalýza Vigenèrovy šifry

Nejdříve musíme zjistit délku m klíče. K tomu máme 2 testy – *Kasiského test* a *Friedmanův test*.

5.1.1 Kasiského test

Pochází někdy z období roku 1840-1860. Říká, že stejné segmenty původní zprávy budou zašifrovány do identického kryptogramu, pokud je jejich vzdálenost δ celočíselným násobkem délky klíčového slova m, tzn. pokud platí $m \mid \delta$

Postup. Hledáme identické segmenty o délce alespoň 3 v kryptogramu a ukládáme jejich vzájemnou vzdálenost δ_i. S velkou pravděpodobností se totiž jedná o zašifrované identické segmenty původní zprávy. Pokud tomu tak je, pak platí $m \mid \delta_i$ pro všechna i z čehož plyne, že m dělí největšího společného dělitele čísel δ_i (označíme $\gcd(\delta_i)$).

Otázkou je jak získáme m? Viz příklad níže.

Kasiského test **Příklad 13**

- *Otevřený text*: the primary weakness of the Vigenere cipher is evidently the repeating nature of the key
- *Klíč*: abcd
- *Kryptogram* (mezery jsou na svých místech): **TIG** SRJODRZ YHALPHST QI **TIG** YIHGQESG FIQJHR JU HVJFHNUNB **TIG** UEQGDTJPJ NBVXRF QI **TIG** NEZ[2]
- Vzájemné vzdálenosti jsou $\delta_1 = 20, \delta_2 = 28, \delta_3 = 20$ a $m = \gcd(20, 28) = 4$

5.1.2 Friedmanův test (Kappa test)

Založen na tzv. *indexu koincidence* (označovaný jako $\mathcal{K}$).

Friedmanův test **Definice 14**

Uvažujeme řetězec x nad nějakou abecedou. Index koincidence $I(x)$ řetězce x je pravděpodobnost, že dva náhodně vybrané prvky tohoto řetězce jsou identické.

Předpokládejme, že se v řetezci x vyskytuje znak A $n_1$krát, znak B $n_2$krát, ..., znak Z n_{26} krát. Kolika způsoby můžeme vybrat identickou dvojici $i$tého znaku? Je to $\frac{n_i \cdot (n_i - 1)}{2}$ krát. Platí následující rovnost, kde n je délka řetězce x.

$$I(x) = \frac{\sum_{i=1}^{26} n_i \cdot (n_i - 1)}{n \cdot (n - 1)}$$

Můžeme provést výpočet indexu koincidence libovolného anglicky psaného textu. Známe přibližný výskyt všech 26 znaků z abecedy.

$$\begin{aligned} A, p_1 &= 0,082 \\ B, p_2 &= 0,015 \\ C, p_3 &= 0,028 \\ &\vdots \\ Z, p_{26} &= 00,1 \end{aligned}$$

Pak platí tento vzorec

[2] Zvýrazněn je stejný kryptogram.

$$I(\text{eng}) \approx p_1^2 + p_2^2 + ... + p_{26}^2 = \sum_{i=1}^{26} p_i^2 = 0.065$$

který říká že v anglickém textu jsou 2 náhodně zvolené znaky s pravděpodobností 6,5 % stejné. U němčiny je to cca 7,6 % a u náhodně generovaného textu (z anglické abecedy) je to 3,8 % (asi polovina angličtiny).

5.1.3 Index koincidence

Theorem 15

Index koincidence je tím menší, čím je text nepravidelnější (rozložení četností znaků rovnoměrnější).

Vliv šifrování a na index koincidence je **rozdílný u polyabecední a monoabecední šifry**. Tím že monoabecední šifry jen permutují znaky, tak index koincidence zůstává stejný jak pro plaintext tak i kryptogram. U polyabecední šifry záleží na klíči (jeho délce). Čím je klíč delší, tím se koincidence snižuje. Index koincidence tedy umí určit zda-li se jedná o polyabecední nebo monoabecední šifru.

5.2 Kryptoanalýza LSFR proudové šifry

Předpokládáme *known plaintext attack* – známe původní zprávu $x_1...x_n$, zašifrovanou zprávu $y_1...y_n$ i počet bitů m. Prvních n znaků klíčového proudo spočítáme jako

$$z_i = (y_i - x_i) \bmod 2 = (x_i + y_i) \bmod 2$$

a pro $1 \leq i \leq m$ platí

$$z_{i+m} = \sum_{j=0}^{m-1} c_j \cdot z_{i+j} \bmod 2$$

pro výpočet celého proudového klíče potřebujeme koeficienty $c_0, ..., c_{m-1}$. Z toho vyplývá, že počítáme soustavou m rovnic o m neznámých

$$\begin{aligned} z_{m+1} &= c_0 z_1 + c_1 z_2 + ... + c_{m-1} z_m \\ z_{m+2} &= c_0 z_2 + c_1 z_3 + ... + c_{m-1} z_{m+1} \\ &\vdots \\ z_{2m} &= c_0 z_m + c_1 z_{m+1} + ... + c_{m-1} z_{2m-1} \end{aligned}$$

5.3 Pravděpodobnost

Jako opakování je zmíněno v prezentacích, případně předměty KMI/DISK1 a KMI/DISK2. Tady vypíšu jen nějaké základní pojmy co by měl člověk chápat pro další kapitoly.

- elementární jev
- pravděpodobnostní prostor
- jev
- množina jevů
- vzájemně neslučitelné jevy
- σ-algebra
- pravděpodobnostní míra
- distribuce pravděpodobnosti

- podmíněná pravděpodobnost
- apriorní a aposteriorní pravděpodobnost
- náhodná proměnná

Podmíněná pravděpodobnost **Definice 16**

$$P(A|B) = \frac{P(A \cup B)}{P(B)}$$

5.3.1 Bayesova věta

Pozor na zaměnění $P(A|B)$ a $P(B|A)$, protože obecně jsou tyto pravděpodobnosti různé. Vztah mezi nimi udává právě Bayesova věta, kterou budeme dále využívat u perfektní bezpečnosti.

Bayesova věta **Definice 17**

$$P(A|B) = \frac{P(A) \cdot P(B|A)}{P(B)}$$

Náhodná proměnná **Definice 18**

Náhodná proměnná $\boldsymbol{X}$ je dvojice $\langle \mathcal{X}, p_{\mathcal{X}} \rangle$, kde $\mathcal{X}$ je konečný prostor elementárních jevů a $p_{\mathcal{X}}$ je distribuce pravděpodobnosti na $\mathcal{X}$. Pravděpodobnost, že náhodná proměnná $\boldsymbol{X}$ nabývá hodnoty $x \in \mathcal{X}$ se značí $p(\boldsymbol{X} = x)$.

[Zdě opět možnost dostudovat řadu dalších věcí, které jsou bud' v prezentacích nebo jiných předmětech.]

6 Perfektní bezpečnost

Máme 2 typy bezpečnosti kryptosystému – **teoretická bezpečnost** (dnes také perfektní nebo informačně-teoretická) a **praktická bezpečnost** (dnes výpočetní bezpečnost).

6.1 Výpočetní bezpečnost (computational security)

Kryptosystém považujeme za takto bezpečný, pokud nejlepší existující algoritmus vyžaduje pro jeho prolomení n kroků (kde n je dostatečně vysoké číslo). Ale už z tohoto je jasné, že je těžké to prokázat, proto se uvažuje vždy k nějakému konkrétnímu typu útoku.

6.2 Dokazatelná bezpečnost (provable security)

Tzv. bezpečnost vzhledem k redukci – jestliže je možné kryptosystém prolomit, pak je možné vyřešit nějaký známý obtížný problém. Např. RSA je dokazatelně bezpečný jestliže dané číslo (modul) nelze faktorizovat v polynomickém čase.

6.3 Perfektní bezpečnost (unconditional security)

Označíme tak kryptosystém, jestliže útočník není schopen získat z kryptogramu žádnou informaci o plaintextu ani s využitím neomezených zdrojů. K přesné formulaci podmínky využijeme počtu pravděpodobnosti. Překvapivě takový systém opravdu existuje.

Z pozorování plyne, že jestliže $p(\boldsymbol{M} = m \mid \boldsymbol{C} = c) = p(\boldsymbol{M} = m)$, pak pravděpodobnost, že byla šifrována zpráva m za podmínky, že byl získán kryptogram c, je stejná jako pravděpodobnost výběru zprávy m bez znalosti kryptogramu c.

Kdy je kryptosystém perfektně bezpečný? **Definice 19**

Jestliže platí

$$p(\boldsymbol{M} = m \mid \boldsymbol{C} = c) = p(\boldsymbol{M} = m)$$

pro všechna $m \in \mathcal{M}$ a $c \in \mathcal{C}$.

$\boldsymbol{M}$ a $\boldsymbol{C}$ musí být nezávislé náhodné proměnné.

Vlastnosti perfektní bezpečnosti jsou:

1. $p(\boldsymbol{C} = c \mid \boldsymbol{M} = m) = p(\boldsymbol{C} = c)$
2. každá zpráva může být zašifrována na libovolný kryptogram
3. $|\mathcal{K}| \geq |\mathcal{C}| \geq |\mathcal{M}|$

Poznámka 20

Pro dosažení perfektní bezpečnosti překvapivě stačí vhodně použít posouvací šifru.

Perfektně bezpečná posouvací šifra **Definice 21**

Předpokládejme posouvací šifru nad $\mathbb{Z}_{26}$. Každá zpráva $m \in M$ je šifrována pomocí náhodně vybraného klíče $k \in \mathcal{K}$ tak, že $p(\boldsymbol{K} = k) = 1$. Taková posouvací šifra je perfektně bezpečná.

[Na slajdu 128 důkaz.] □

Shannonův teorém **Theorem 22**

Nechť $|\mathcal{K}| = |\mathcal{C}| = |\mathcal{M}|$. Kryptosystém je perfektně bezpečný pokud jsou splněny následující podmínky:

1. každá zpráva $m \in \mathcal{M}$ je šifrována pomocí náhodně vybraného klíče $k \in \mathcal{K}$ tak, že $p(\boldsymbol{K} = k) = \frac{1}{|\mathcal{K}|}$
2. pro každé $m \in \mathcal{M}$ a $c \in \mathcal{C}$ existuje právě jeden klíč k_0 takový, že $e(m, k_0) = c$.

6.3.1 One-time Pad (Vernamova šifra)

Autor je Gilbert Standford Vernam (také autor proudové šifry). Původně pro šifrování telegrafních zpráv. Využívá aritmetiku modulo 2. Klíč je stejně dlouhý jako plaintext a je použit pouze jednou (z toho název **one-time**). Nevýhodou je množství klíčů, které se musí používat a jejich následná distribuce.

One-time Pad **Definice 23**

Nechť $|\mathcal{M}| = |\mathcal{C}| = |\mathcal{K}| = \mathbb{Z}_2^m, m \in \mathbb{N}$. Pro klíč $\boldsymbol{k} = \langle k_1, ..., k_m \rangle \in \mathcal{K}$ definujeme šifrovací funkci

$$e(\boldsymbol{x}, \boldsymbol{k}) = \langle x_1 + k_1, ..., x_m + k_m \rangle,$$

a dešifrovací funkci

$$d(\boldsymbol{y}, \boldsymbol{k}) = \langle y_1 + k_1, ..., y_m + k_m \rangle,$$

pro všechna $\boldsymbol{x} = \langle x_1, ..., x_m \rangle \in \mathcal{M}, \boldsymbol{y} = \langle y_1, ..., y_m \rangle \in \mathcal{C}$.

7 Falešné klíče

Předpoklad: Snažíme se najít klíč. Předpokládáme, že: máme neomezené zdroje, používáme angličtinu a uvažujeme útok ciphertext-only attack.

Jak poznáme, že odhalený klíč je správný? Pokud dešifrování za pomocí nalezeného klíče dává smysluplný text. Na druhou stranu více klíčů může jediný kryptogram rozšifrovat na smysluplný text. Toto se nedá nijak odhalit a pravděpodobnost na správný klíče je dána jejich nalezeným počtem.

Falešný klíče **Definice 24**

Falešný klíče (spurious key) je nesprávný klíč, který dešifruje kryptogram na smysluplný text.

Snažíme se určit jak dlouhý kryptogram potřebujeme, aby byl počet falešných klíčů nulový. Nejkratší takovou délku označíme jako **vzdálenost jednoznačnosti (unicity distance)**. *Klíčová ekvivokace* (průměrná míra nejistoty, že kryptogram byl zašifrován určitým klíčem) a *redundance jazyka* (nadbytečnost v jazyce) jsou dva pojmy, které k tomu potřebujeme a souvisí s přirozeným jazykem. Oba pojmy vychází z *entropie*[3]

Theorem 25

Dvě náhodné proměnné $\boldsymbol{X}$ a $\boldsymbol{Y}$ jsou nezávislé jestliže platí $E(X|Y) = E(X)$.

7.1 Entropie

Zařízení generující posloupnost symbolů, kde výstup v jednom okamžiku neovlivňuje výstup v dalším okamžiku.

Míra překvapení **Definice 26**

Jak moc jsme překvapeni že se objeví i-tý symbol. Zapsáno matematicky:

$$-\log_n(P_i)$$

kde P_i je pravděpodobnost objevení se i-tého symbolu.

[3]tzv. míra překvapení, že se objeví i-tý symbol $-\log_n(P_i)$

Základ logaritmu určuje jednotku. Obvykle $n = 2$ a nazývá se *shannon* nebo bit. Používá se protože je klesající a má asymptotu $x = 0$.

Příklad 27

Jestliže produkujeme tři symboly A, B, C se stejnou četností, je pravděpodobnost výskytu každého z nich $\frac{1}{3}$. Tedy míra překvapení se spočítá jako $-\log_2(\frac{1}{3}) \doteq 1,585$.

Pokud bychom měli jen 1 symbol, tak by pravděpodobnost byla 1 a platilo by, že $-\log_2(1) \doteq 0$. Čili vůbec nás nepřekvapí, že se objeví právě onen jediný symbol.

Entropie diskrétní náhodné proměnné **Definice 28**

Je definována jako *průměrná míra překvapení* neboli: *[TBA]*

Platí následující vztahy:

- $E(X \mid Y) = E(X, Y) - E(Y)$
- X a Y jsou závislé proměnné právě tehdy když $E(X \mid Y) = E(Y)$

7.1.1 Entropie kryptosystému

Uvažujeme náhodné proměnné:

- $M = (\mathcal{M}, p_{\mathcal{M}})$
- $K = (\mathcal{K}, p_{\mathcal{K}})$
- $C = (\mathcal{C}, p_{\mathcal{C}})$

a zajímá nás entropie těchto tří proměnných $E(M)$, $E(C)$ a $E(K)$.

Na základě těchto věcí můžeme entropii aplikovat na konkrétní kryptosystém a spočítat přesné číselné hodnoty.

Příklad entropie kryptosystému **Příklad 29**

- kryptosystém z příkladu o perfektní bezpečnosti
- $\mathcal{M} = \{a, b\}$; $p(\mathbf{M} = a) = \frac{1}{4}$, $p(\mathbf{M} = b) = \frac{3}{4}$
- $\mathcal{K} = \{k_1, k_2, k_3\}$; $p(\mathbf{K} = k_1) = \frac{1}{2}$, $p(\mathbf{K} = k_2) = p(\mathbf{K} = k_3) = \frac{1}{4}$
- $\mathcal{C} = \{1, 2, 3, 4\}$ $p(\mathbf{C} = 1) = \frac{1}{8}$, $p(\mathbf{C} = 2) = \frac{7}{16}$, $p(\mathbf{C} = 3) = \frac{1}{4}$, $p(\mathbf{C} = 4) = \frac{3}{16}$
- z definice entropie vypočítáme:

$$E(\mathbf{M}) = -\tfrac{1}{4}\log_2\tfrac{1}{4} - \tfrac{3}{4}\log_2\tfrac{3}{4} \doteq 0{,}81$$
$$E(\mathbf{K}) \doteq 1{,}5$$
$$E(\mathbf{C}) \doteq 1{,}85$$

7.1.2 Klíčová ekvivokace

Podmíněná entropie $E(K|C)$ se nazývá *klíčová ekvivokace*. Dá se přeložit jako nejistota nebo neurčitost (udává totiž průměrnou nejistotu klíče když známe nějaký kryptogram).

Platí pro ni tento vzorec: $E(K|C) = E(M|C + (E(M))$

7.1.3 Entropie přirozeného jazyka

Pokud máme věty v přirozeném jazyce, tak se jedná v podstatě o věty generované zdrojem s nenulovou pamětí (slova/písmena mají mezi sebou souvislost podle předchozích). Z tohoto se dá

vyčíst že zdroje produkují znaky s nějakou pravděpodobností. Entropii E_L přirozeného jazyka L můžeme shora omezit:

$$E_L \approx E(M) = -\sum_{m \in M} P(M = m) \cdot \log P(M = m)$$

Existuje i korelace mezi jednotlivými písmeny textu

$$E_l \approx \frac{E(M^2)}{2}$$

Pro angličtinu platí $E(M) \doteq 4,19$ a $\frac{E(M^2)}{2} \doteq 3.9$.

Pokud bychom brali n-gramy textu mohli bychom číslo zpřesňovat

$$E_L = \lim_{n \to \infty} \frac{E(M^n)}{n}$$

Přirozený jazyk má i poměrně vysokou redundanci (angličtina cca 75 %). To neznamená, že bychom 75 % znaků mohli vypustit, ale můžeme zkonstruovat Huffmanův kód s kompresním poměrem 4:1.

7.1.4 Počet falešných klíčů

Falešný klíč je klíč, který transformuje zašifrovaný text na smysluplnou zprávu, ale nejedná se o zprávu, kterou odesílatel opravdu odeslal.

Theorem 30

Předpokládejme kryptosystém, pro který platí $|C| = |M|$ a každý klíč je vybírán se stejnou pravděpodobností. Pak pro kryptogram délky n (kde n je dostatečně velké) platí:

$$s_n \geq \frac{|K|}{|M|^{n \cdot R_L}} - 1.$$

kde s_n je počet falešných klíčů a R_L je jejich redundance.

7.1.5 Vzdálenost jednoznačnosti kryptosystému

Zvyšováním délky kryptogramu získává útočník více informací o otevřené zprávě. Při určité délce n_0 jich má dostatek k jednoznačnému určení pravého klíče. n_0 nazveme **vzdálenost jednoznačnosti**. Platí tento vzorec

$$n_0 \approx \frac{\log_2 |K|}{R_L \cdot \log_2 |M|}$$

Pro anglický jazyk a substituční šifru by byl výsledek 25. Jinak řečeno útočníkovi stačí zašifrovaný text o délce 25 znaků, aby byl schopen jednoznačně určit klíč k celé zprávě.

Čím větší n_0 je, tím lepší. Zvětšení se dá provést zmenšením R_L, což můžeme udělat pomocí komprese otevřeného textu (vynechání u po q nebo vynechání některých samohlásek).

7.1.6 Ekvivokace zprávy

Dá se tak označit vzorec $E(M|C)$.

Nezajímá nás celkový absolutní počet falešných klíčů, ale jejich průměrný počet.

8 Současné symetrické šifry

DES a AES jsou dvě soudobé symetrické šifry. Jsou to tzv. *iterační blokové šifry* (opakuje se v ní použití nějaká funkce v rámci mnoha iterací). Pro zvýšení bezpečnosti kryptosystému se používají dvě dále vysvětlené operace – *konfúze* a *difúze*.

Konfúze je operace snažící se skrýt vztah mezi kryptogramem a použitým klíčem (např. v DES a AES pomocí substituce).

Difúze je myšlenka, že znak plaintextu by měl ovlivnit co nejvíce znaků kryptogramu (např. dnes změna 1 bitu plaintextu ovlivní asi $\frac{1}{2}$ bitů kryptogramu). V DES se jedná o S-boxy kde 1 bit vstupuje do více boxů.

Klasické šifry využívají pouze konfúzi. Moderní šifry používají obě operace zřetězeně.

8.1 Data Enycryption Standard (DES)

Je založena na *Feistelově šifře*. V ní je šifrování prováděno iterativně v tzv. *rundách*. V tomto případě nemusí být šifrovací funkce invertibilní kvůli poskládání vzorce šifrování (XOR).

Invertibilita šifrovací funkce **Poznámka 31**

Nutné myslet na to, že šifrovací funkce by měla být invertibilní. Abychom mohli provádět i dešifrování.

8.1.1 Feistelova šifra

Bloková šifra. Blok $m \in M$ je rozdělen na 2 poloviny L_0 a R_0. Šifrování prováděno v iteracích (*rundách*). Čím více rund tím bezpečnější. Výpočet i-té rundy

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \oplus f(k_i, R_{i-1})$$

Po provedení rund se šifrovaný blok získá složením L_r a R_r.

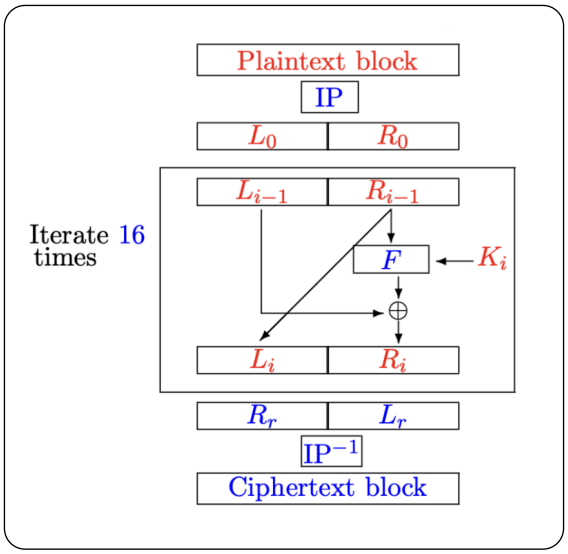
Rundovní funkce nemusí být invertibilní[4]. Šifrování i dešifrování má stejnou podobu. Z těchto faktů plynou 2 výhody – je možné použít libovolnou rundovní funkci a implementace šifrovací a dešifrovací funkce je stejná (jen se rundovní klíče používají v opačném pořadí).

Jde o blokovou, bitově orientovanou šifru (pracuje nad $\mathbb{Z}_2^m$, kde m je délka bloku). Tím, že je založena na Feistelově šifře tak:

- počet rund = 16
- počet rundovních klíčů = 16
- délka bloku = 64 bitů
- délka klíče = 56 (64) bitů (každý 7 bit se doplňuje 1 paritním bitem)
- délka rundovních klíčů $k_0, ..., k_{15}$ = 48 bitů.

[4]Invertibilní funkce je taková funkce ke které existuje funkce inverzní.

Před první a poslední rundou se provádí permutace (ty nemají vliv na bezpečnost, byly přidány pouze pro lepší HW v té době).



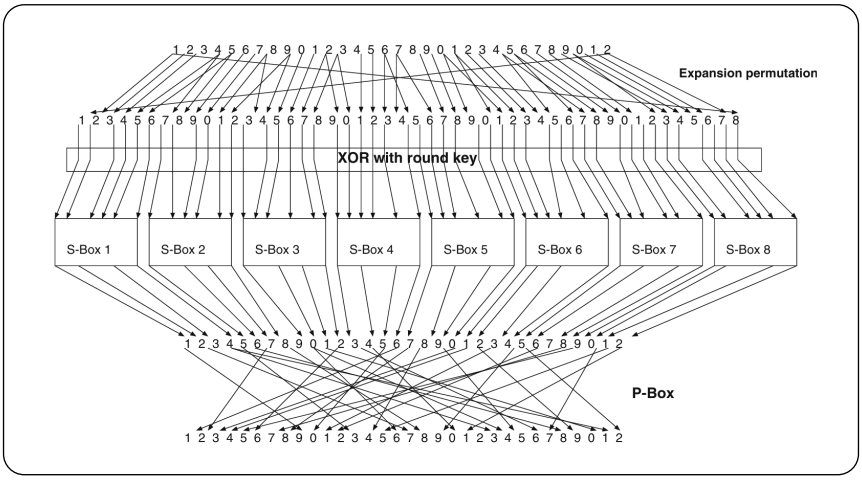
Obrázek 6: Blokové schéma DES.

DES konkrétním způsobem definuje rundovní funkci F, což je její největší přínos (zbytek stejný jako Feistelova šfira).

Postup

1. inicializace: počáteční permutace
2. rozdělení na 2 bloky: 6č bitů rozděleno na poloviny L_{i-1} a R_{i-1}
3. expanzní permutace: 32 bitů se permutuje a expanduje na délku rundovního klíče (48 bitů)
4. XOR s rundovním klíčem: jediné kde se používá rundovní klíč
5. substituce pomocí S-boxů: 6 bitová hodnota vstoupí do S-boxu (ty jsou nelineární) a 4 bity vystoupí (*nibble*); 1. a 6. bit adresují řádky, 2. - 5. adresují sloupce; výsledkem všech je $8 \cdot 4 = 32$ bitů
6. permutace pomocí P-boxů: permutace 8 nibblů
7. spojení: spojí se 32 bitové bloky $L_i = R_{i-1}$ a $R_i = L_{i-1} \oplus f(k_i, R_{i-1})$
8. konečná permutace

Rundovní funkce se aplikuje v krocích 3. až 6 (viz obrázek).



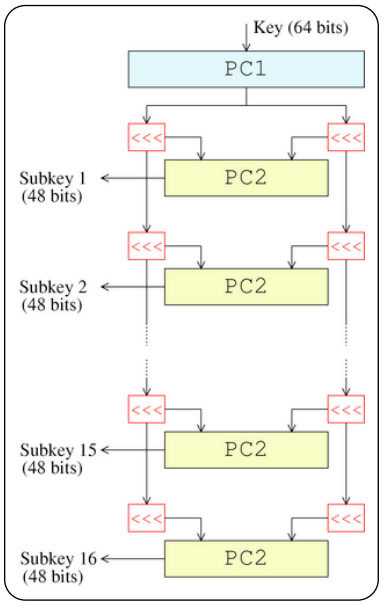
Obrázek 7: Mapování bitů v DES šifře (3. až 6. krok v postupu).

8.1.2 Tvorba rundovních klíčů

Mají délku 64 bitů, po odebrání paritních bitů (8., 16., 24., … bit) se získá 5ž bitový klíče k. Ten se pak dále dělí na 2 poloviny C_0 a D_0. Pro každou rundu platí

$$C_i = RL_i(C_{i-1})$$

$$D_i = RL_i(D_{i-1})$$



Obrázek 8: Tvorba rundovních klíčů

Délka klíče 56 bitů se rychle stala nedostatečnou. Bylo nutné vylepšení. Dvě varianty – **3DES** (použití tří šifer DES DES $\rightarrow$ DES^{-1} $\rightarrow$ DES) nebo **two key 3DES** (navíc se použijí 2 klíče).

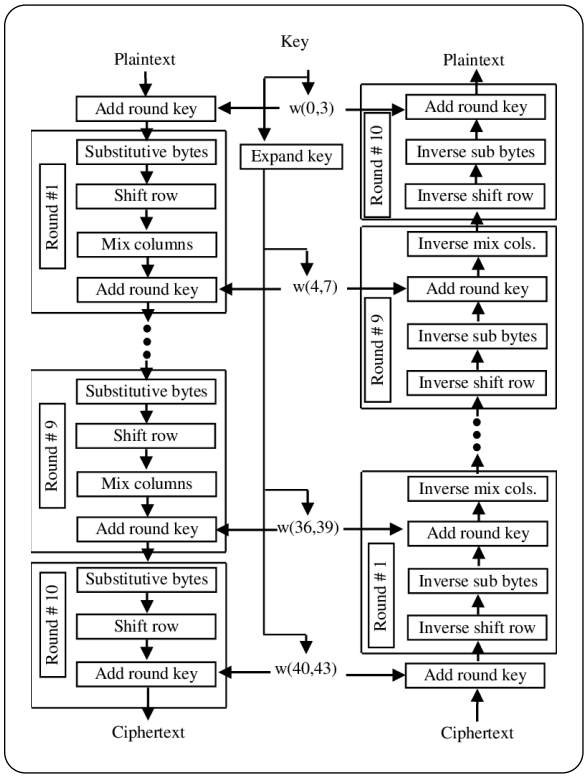
8.2 Advanced Encryption Standard (AES)

Výrazně zvětšená délka klíče a je také volitelná (128, 192, 256 bitů) a podle ní se mění počet iterací (rund; 10, 12 nebo 14). Není bitově orientovaná, ale bajtově orientovaná. Používá se polynomiální aritmetika (kvůli tomu že to jsou bajty) nad prvočíselnými tělesy. Standard AES vznikl ze šifry Rijndael od belgickcýh kryptologů. Označení podle použité délky klíče. Celým procesem šifrování prochází stavová matice S a matice rundovního klíče K typu 4×4.

Konfúze pomocí operace *SubBytes* a difúze pomocí *ShiftRows* a *MixColumns* (operace dělají přesně to co mají v názvu).

Dnešní využití je široké, např. WPA3, VPN, čipy v platebních kartách, datová úložiště, Signal, BitLocker, …

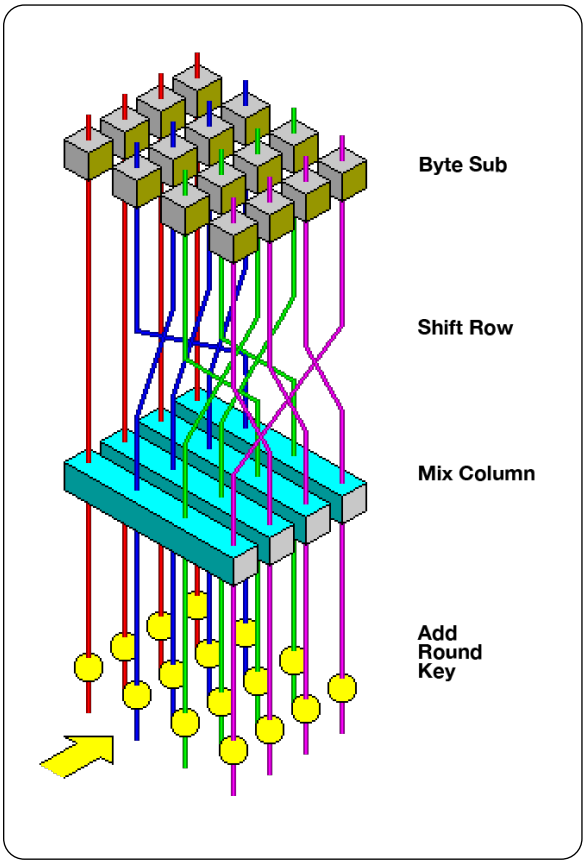
Pokud AES potřebuje hledat inverzní prvky z polynomiální aritmetiky kouká se do tzv. *lookup table*, kde jsou vypočítané a stačí je načíst.



Obrázek 9: Schéma šifry AES.

Rundovní funkce se skládá ze 4 operací:

1. **SubBytes** – jasně definovaná operace, nelineární, realizuje konfúzi, provádí se ve 2 krocích (počítání inverze s mod a afinní transformace)
2. **ShiftRows** – realizuje difúzi, je invertibilní, rotace řádků (i-tý řádek se rotuje o $(i-1)$ pozic doleva)
3. **MicColumns** – realizuje difúzi, je invertibilní, zajišťuje vzájemnou interakci řádků během každé rundy
4. **AddRound Key** – XOR stavové matice rundovního klíče (po bajtech)



Obrázek 10: Operace rundovní funkce u AES.

Poznámka 32

Polynomiální aritmetika umožní rychlé, reverzibilní a bezepčné operace. Celé je to „matematicky průhledné“.

```
// šifrování
AddRoundKey(S, K_0)
for i = 1, 2, ..., 9 do
  SubBytes(S)
  ShiftRows(S)
  MixColumns(S)
  AddRoundKey(S, K_i)
end for
SubBytes(S)
ShiftRows(S)
AddRoundKey(S, K_10)
```
Typst

```
// dešifrování
AddRoundKey(S, K_10)
InversibleShiftRows(S)
InversibleSubBytes(S)
for i = 9, 8, ..., 1 do
  AddRoundKey(S, K_i)
  InversbileMixColumns(S)
  InversibleShiftRows(S)
```
Typst

```
9     InversibleSubBytes(S)
10  end for
11  AddRoundKey(S, K_0)
```

8.2.1 Šifra Rijndael

Je to iterační a bloková šifra (převzato z DES pro posílení konfúze i difúze). Používá substituční bloky. Je bajtově orientovaná. Jako ireducibilní polynom[5] je použit $x^8 + x^4 + x^3 + x + 1$ a posloupnost bitů 100011011 (číslo v bitu udává zda se x na danou mocninu v polynomu nachází či nikoliv).

8.3 Polynomiální aritmetika

- Okruh jako algebraická struktura
- Pole jako algebraická struktura

[Opět část rozšiřitelná z jiných předmětů.]

9 Asymetrické šifrování

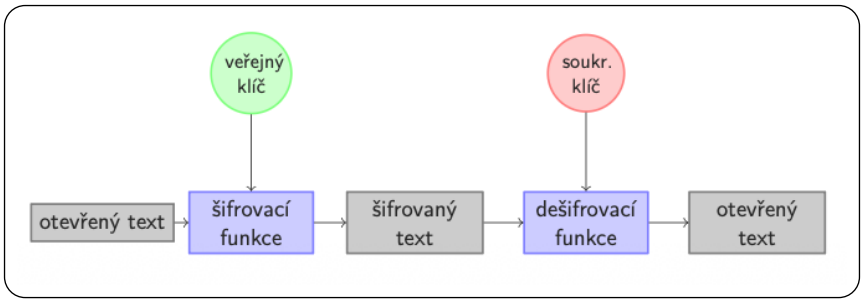
Na začátek zopakování věcí jak jsou rozebrané na začátku obecně o šifrování. U symetrického šifrování jsou 2 problémy – distribuce klíče a příliš velké množství klíčů. Problém můžeme vyřešit bezpečnou výměnou klíče ⇒ koncepčně dostaneme jiný systém šiforvání. V tomto příapdě nejsou klíče pro šifrování a dešifrování stejné:

- šifrovací klíč … **veřejný klíč**
- dešfirovací klíče … **soukromý klíč** (držen v tajnosti)

Příklady: šifrování založené na diskrétním logaritmu, zavazadlovém problému nebo eliptických křivkách, RSA, …

Postup

1. Příjemce vytvoří soukromý i veřejný klíč
2. Soukromý klíče uschová a veřejný zveřejní
3. Odesílatel zašifruje zprávu pomocí veřejného klíče
4. Zpráva je poslána
5. Příjemce ji rozšifruje pomopcí soukromého klíče



Obrázek 11: Asymetrické šifrování

9.1 Bezpečná výměna klíče

Máme několik několik předpokladů. Alice vlastní zámek A, který umí zamknout a odemknout pouze ona. Bob vlastní zámek B, který umí zamknout a odemknout pouze on.

[5] To znamená, že nejde rozložit na součin jednodušších polynomů.

9.1.1 Jednoduché řešení

Jako první nás napadné jednoduché řešení popsané v následujícím postupu.

1. Alice uzamkne zprávu zámkem A
2. Takto zamčenou zprávu pošle Bobovi
3. Bob tuto zprávu uzamkne zámkem B
4. Tuto (drvkarát zamčenou) zpráv pošle zpět Alici
5. Alice odemkne zámek A
6. Zprávu, která je zamčená nyní pouze zámkem B pošle Bobovi
7. Bob odemkne zámkem B a dostane původní zprávu

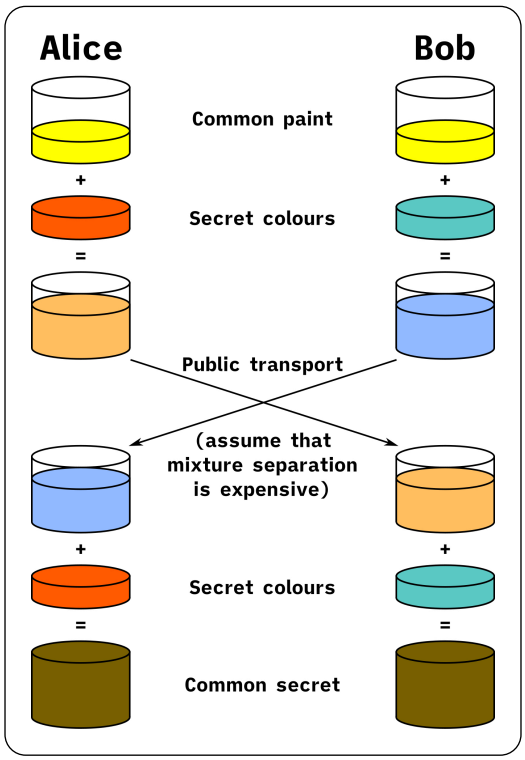
Tento postup má ale jeden háček, neboť funguje díky vztahu

$$e(\cdot, k_A) \circ e(\cdot, k_B) \circ d(\cdot, k_A) \circ d(\cdot, k_B) = \mathrm{id}$$

který se nazývá komutativita, ten ale **neplatí obecně**. Proto musíme najít lepší a funkčí řešení.

9.1.2 Diffie-Hellmanova výměna klíče

Myšlenka tohot postupu je založená na míchání barev (viz obrázek). Tím záasdním krokem je mícháni společné a tajné barvy. Pozor ale tento proces je pouze **jednosměrný**.



Obrázek 12: Myšlenka výměny klíče podle Diffieho a Hellmana.

Převedeno do matematiky hledáme funkci f, která bude taky jednosměrná. Čili vyčíslení funkce bude výpočetně snadné, ale výpočet inverze bude obtížný (pokud znám jen y, tak získat $f(x)$ je obtížné). Pro nás bude takovou vhodnou funkcí umocnění v modulární aritmetice.

Hodnoty $g^x \pmod p$ **Poznámka 33**

Hodnoty výrazu $g^x \pmod p$ se chovají pseudonáhodně.

Postup *(Všechna vybíraná čísla jsou z prvočíselného tělesa.)*

1. Alice a Bob si zvolí velká veřejná prvočísla p a g (obě $\in GF^*(p)$)
2. Oba vyberou náhodně tajná čísla a a b
3. Alice vypočítá $A \equiv g^a \pmod p$ a Bob $B \equiv g^B \pmod p$
4. Alice a Bob si vyřejně vymění A a B
5. Alice vypočítá $A' \equiv B^a \pmod p$ a Bob $B' \equiv A^b \pmod p$
6. Potom platí $A' \equiv B' \pmod p$ a toto číslo se může použít jako tajný klíč.

9.1.3 Diffie-Hellmanův problém

Ukazuje a zajišťuje bezpečnost této výměny. *[TBA]*

9.2 Generování bezpečných prvočísel

Prvočíslo p nazveme jako bezpečné pokud má tvar $p = 2q + 1$, kde q je také prvočíslo a nazývá se *Sophie Germainové prvočíslo*. Je to kvůli tomu, že plno útoků je založených (a efektvních) pokud číslo $p - 1$ jde rozložit na plno malých prvočinitelů. Tímto požadavkem se tomuto vyhneme, protože q je obrovské.

9.2.1 OpenSSL jako praktická ukázka

[TBA]

9.2.2 Odbočka k modulární aritmetice

Pokud je p prvočíslo, pak existuje $g \in GF^*(p)$, tak že každý prvek z toho je možné vyjádřit jako jeho mocninu. Tomuto g říkáme **generátor** (případně primitivní kořen).

Dále platí, že $GF^*(p)$ má $\varphi(p-1)$ generátorů.

Prvočíselné pole $GF(p)$, kde $GF^*(p)$ znamená, že je bez nulového prvku (jde o $*$).

Pracujeme s **problémem diskrétního logaritmu**, kde jde o nalezení specifického exponentu.

$$g^x \equiv h \pmod p$$

V praxi se spíše pracuje s obecnější verzí, kde g nemusí být generátor.

9.3 Šiforvání založené na zavazadlovém problému

Existuje varianta tohoto progblému, která je řešitelná v polynomickém čase. Důležitou roli tu hraje **superrostoucí posloupnost**, což je taková posloupnost, kde prvek musí být větší než součet všech předchozích.

Super rostoucí posloupnost **Definice 34**

Posloupnost (a_n) je super rostoucí jestliže každý další prvek je větší než součet všech předchozích. Např.: $a_1 = 2, a_2 = 5, a_3 = 12, a_4 = 30$.

$$a_n > \sum_{i=1}^{n-1} a_i$$

Příklad 35

Máme množinu $\{27, 3, 6, 2, 52, 13\}$, která by tvořila superrostoucí posloupnost. Celková hmotnost $S = 70$.

1. Jelikož $S > 52$, tak 52 je součástí zavazadla (řešení).
2. Vznikne nová posloupnost $\{2, 3, 6, 13, 27\}$ i nová celková hmotnost $S = 70 - 52 = 18$
3. Protože $S < 27$, tak 27 nebude součástí zavazadla
4. Vznikne opět nová posloupnost $\{2, 3, 6, 13\}$ i „nová“ celková hmotnost $S = 18$
5. ⋮

V konečném důsledku se algoritmus zastaví (protože $S = 0$) a odpoví na danou otázku i ukáže co bude součástí zavazadla.

Z pohledu šifer je problém následující:

- **zpráva** je jako výběr položek do zavazadla (b_i)
- **soukromý klíč** je superrostoucí posloupnost
- **veřejný klíč** je obtížná posloupnost vypočítaná ze superrostoucí pomocí

$$T(M_i) = M_i \cdot m \bmod n$$

- **kryptogram** je celková hmotnost zavazadla.

Šifrovací proces

1. Bob zvolí libovolnou superrostoucí posloupnost a čísla m, n splňující podmínky (soukromý klíč)
2. Bob z ní vypočítá obtížnou posloupnost (veřejný klíč)
3. Alice rozdělí otevřený text (binární tvar) na bloky, kde počet bitů každého bloku odpovídá počtu čísel v posloupnosti
4. Alice zašifruje každý blok pomocí veřejného klíče a odešle Bobovi

Dešifrovací proces

1. Bob zná soukromý klíč (superrostoucí posloupnost a m, n)
2. Bob vypočítá inverzní prvek m^{-1} k m v $\langle \mathbb{Z}_n, \cdot \rangle$
3. Bob každé číslo kryptogramu vynásobí číslem m^{-1} modulo n (tyto čísla představují hmotnosti položek zavazadla)
4. Nakonec Bob vyřeší v polynomickém čase zavazadlový problém pro superrosstoucí zavazadlo

Poznámka 36

Slabinou tohoto přístupu je, že šifrovací funkce je lineární. Vztah tedy může být popsán pomocí lineárních rovnice, pro jejichž řešení máme dobré metody.

9.4 Zpřesnění matematických pojmů

Zanedbatelná funkce **Definice 37**

Funkce $\epsilon : N \to [0, 1]$ se nazývá zanedbatelnou funkcí, jestliže pro každé $c \in N$ existuje $n_0 \in N$ takové, že pro všechna $n > n_0$ platí

$$\epsilon(n) < \frac{1}{n_c}$$

Taková funkce tedy velmi rychle klesá k nule. Událostí vyskytující se za *zanedbatelnou pravděpodobností* můžeme ignorovat.

Jednosměrná funkce **Definice 38**

V polynomickém čase vyčíslitelná funkce f je jednosměrnou funkcí, jestliže pro každý pravděpodobnostní polynomický algoritmus A existuje zanedbatelná funkce ϵ taková, že pro všechna n, x, y taková, že $|x| = n, y = f(x)$, platí

$$P(A(y) = x) < \epsilon(n)$$

Neboli každý pravděpodobnostní polynomický algoritmus, který vypočítá z obrazu y správný vzor x' uspěje se **zanedbatelnou pravděpodobností**.

Domněnka **Theorem 39**

Domníváme se, že existuje alespoň jedna jednsoměrná funkce.

Pokud by se tato doměnka potvrdila, znamenalo by to $P \neq NP$. Ale máme několik kandidátů, které se zatím nepdoařilo invertovat. Jako příklad slouží 3 funkce:

- umocnění funkce $f(x) = g^x \bmod p$ pro daný generátor g a provčíslo p. Invertování takovéto funkce by znamenalo vyřešit *Problém diskrétního logaritmu (DLP)*
- funkce $f(x) = A \cdot B$, kde $|x| = n$ kóduje čísla A a B jejichž délka je $\frac{n}{2}$. Invertování by znamenalo vyřešit *Integer Factorization problem (IFP)*.
- nakonec funkce $f(x) = x^e \bmod n$, kde n je velké složené číslo a e je nesoudělné s $\varphi(n)$. Vyřešení tohot by znamenalo vyřešit *problém RSA*.

9.5 Algoritmus RSA

Pojmenována podle tvůrců. Jedná se o asymetrcikou šifru považovanou za velmi bezpečnou. Používá se pro bezpečnou výměnu klíčů při symetrickém šifrování (tz.v hybridní šifrování, protokoly SSL a TSL) či digitální podpis.

9.5.1 Generování soukromého a veřejného klíče RSA

1. Zvolí se dvě různá provčísla p a q (cca stejně velká, obvykle alespoň 1024 až 3072 bitů)
2. Vypočítá se součin $n = p \cdot q$ (platí $\varphi(n) = (p-1) \cdot (q-1)$)
3. Náhodně se zvolí $e \in \{1, 2, ..., \varphi(n) - 1\}$ tak, aby

   $$\gcd(e, \varphi(n)) = 1$$

   (e … veřejný exponent, často je 3)
4. Pomocí rozšířeného Euklidova algortimu vypočteme inverzi $d = e - 1 \bmod \varphi(n)$
5. Pak platí
   - $k_e = \langle e, n \rangle$ … veřejný klíč
   - $k_d = d$ … soukromý klíč
   - čísla p, q se mohou odložit, ale nikdy se nesmí zveřejnit

9.5.2 Šifrování a dešifrování

Text $x \in M$ se rozdělí na číselné bloky x_i, tak aby $x_i < n$. Vztahy jsou pak následující:

$$e(x_i, k_e) = x_i^e \bmod n$$

$$d(y_i, k_d)$$

Příklad 40

V prezentaci na slajdech 41 a 42 přesně detailně rozepsaná ukázka.

Věta **Theorem 41**

Algoritmus RSA pracuje správně, tzn. platí

$$d(e(x, k_e), k_d) = x$$

pro libovbolný blok x otevřeného textu.

K této větě je i důkaz (slajdy 43-45). □

9.5.3 Praktické aspekty a využití

Praktické využití algoritmu RSA rozebráno v následujících příkladech.

Generování soukromého klíče
Pro výpočet modulo n potřebujeme 2 velká prvočísla ($n = p \cdot q$). Pokud má mít n 1024 bitů, pak p i q musí mít po 512 bitech. Náhodně generujeme číslo délky, kterou chceme a provedem test provčíselnosti. Mohou nasta 2 problémy:

1. Kolik čísel musíme v průměru vygenerovat, abychom narazili na prvočíslo?(není pravděpodobnost objevení se prvočísla příliš malá?)
2. Jak efektivně provést test prvočíslenosti?

Problém pravděpodobnosti objevení prvočísla
S délkou prvočísel ubývá. Platí, že náhodně vygenerované číslo p mezi 1 a N je prvočíslem s pravděpodobností cca $\frac{1}{\ln N}$. Pokud testujeme pouze lichá čísla, pak vzorec upravíme na $\frac{2}{\ln N}$. Pro n délky oněch 1024 bitů by tedy platilo $\frac{2}{\ln 2^{512}} \approx \frac{1}{177}$

Problém test prvočíselnosti
Existuje několik algoritmů, které můžeme zvolit. Existují i deterministické pracující v polynomickém čase. V praxi se najčastěji využívá algoritmus typu Monte Carlo, který však nevrací vždy správnou odpověď ⇒ spouští se vícekrát čímž se snižuje pravděpodobnost nesprávného výsledku.

9.5.4 Bezpečnost RSA

Bezpečnost RSA je založena na předpokladu, že problém faktorizace IFP je pro velké moduly obtížný. S přesnostní však nevíme do jaké třídy spadá (pravděpodobně NP-complete, ale určitě NP a co-NP).

9.5.5 Faktorizace Pollardova $p - 1$ metoda

B-power smooth čísla **Definice 42**

Tak nazveme číslo n jestliže jsou mocniny prvočísel v jeho prvočíselném rozkladu menší než B.

Např. $n = 21600$ je 33-power smooth, protože $n = 2^5 \cdot 3^3 \cdot 5^2$ a všechna tato čísla jou menší než 33.

Předpokládejme, že číslo $n = p \cdot q$ a $p - 1$ je B-power smooth a $q - 1$ není B-power smooth. Pak platí $(p-1)|B!$ a současně je nepravděpodobné, že $(q-1)|B!$. Vypočítáme a jako $a \equiv 2^{B!} (\mathrm{mod}\, n)$. A protože $n = p \cdot q$, paltí $a \equiv 2^{B!} (\mathrm{mod}\, p)$ a $a \equiv 2^{B!} (\mathrm{mod}\, q)$.

9.5.6 Faktorizace rozdíl druhých mocnin

9.5.7 Faktorizace další metody

1. Continued Fraction Method
2. Quadratic Sieve
3. Eliptic Curve Method
4. Number Fiedl Sieve

čsílo kratší než 256 bitů může být v roce 2025 faktorizováno na obyčejném osobním počítači. Obvykle používaná délka module je 1024 až 3072 bitů.

9.5.8 Útok pomocí postranního kanálu

Při implementaci vznikají postranní kanály, které lze využít k útoku aniž bychom museli provádět kryptoanalýzu. Např. časový postranní kanál nebo chybvý postranní kanál.

Bleichenbacherův útok
Útočník odychtí kryptogram c. Náhodnými konstantami s_i modifikuje c na řadu čísel tvaru $c_i = c \cdot s_i^e \bmod$, které zašle příjemci.Příjemce c_i dešifruje a zkontroluje, zdali je formátu 02. Pokud není, pošle chybové hlášení a útočník zopakuje postup. Jakmile se útočník trefí do zprávy formátu 02, ví že zprávu odhalil.

9.6 Testy prvočíselnosti

Pomocí těchto testů (algoritmických postupů) jsme schopni ověřit zda se jedná o prvočíslo. Testy mají různé výhody, nevýhody a specifika.

9.6.1 Fermatův test

Založen na *Malé Fermatově větě*. Ale dá se použít jen pro testování složenosti daného čísla ⇒ není úplně vhodný pro test prvočíselnosti. Důležitý pojem tzv. **Carmichaelova čísla**, což jsou čísla, která tímto testem projdou, ale jsou to čísla složená (naštěstí je jich poměrně málo).

Malá Fermatova věta **Theorem 43**

Pro každé prvočíslo p a každé celé číslo a platí

$$a^p \equiv a (\mathrm{mod}\, p)$$

Tzn. číslo $(a^p - a)$ je dělitelné prvočíslem p. Je založena na Eulerově větě.

Eulerova věta **Theorem 44**

Pro každé přirozené číslo n a přirozené číslo a nesoudělné s n platí

$$a^{\varphi(n)} \equiv 1 (\mathrm{mod}\, n)$$

Idea **Definice 45**

Základní myšlenka je, že pokud najdeme $a \in \mathbb{Z}_n^*$, pro které neplatí $a^{n-1} \equiv 1 (\mathrm{mod}\, n)$, pak a je svědkem složenosti n.

$\mathbb{Z}_n^*$ obsahuje všechna celá čísla a která splňují:
- leží v rozsahu $1 \leq a < n$
- jsou nesoudělná s n

Pokud bychom postup opakovali můžeme pravděpodobnost, s kterou známe výsledek zpřesňovat.

9.6.2 Miller-Rabinův test

Tento test nemá *Carmichaelova čísla*. Založen na následující větě.

Věta **Theorem 46**

Nechť p je prvočíslo, pro které platí

$$p - 1 = 2^k q$$

kde q je liché číslo. Dále nechť a je číslo takové, že $\gcd(a, p) = 1$. Pak platí jedno z následujících tvrzení
1. $a^q \equiv 1 (\mathrm{mod}\, p)$
2. jedno z čísel $a^q, a^{2q}, a^{4q}, ..., a^{2^{k-1}q}$ je kongurentní s $-1 \,\mathrm{mod}\, p$

Při tomto testu se opět vychází z obměněné implikace. Nechť n je liché číslo, pro které platí $n - 1 = 2^k q$, kde q je liché číslo. Číslo a takové, že $\gcd(a, n) = 1$ se nazývá *svědek složenosti* čísla n, jestliže současně platí tyto dvě podmínky:
1. $a^q \not\equiv 1 (\mathrm{mod}\, n)$
2. $a^{2^i} q \not\equiv -1 (\mathrm{mod}\, n)$ pro všechna $i = 0, 1, ..., k - 1$

9.6.3 Rychlé umocnění

Výhodou symetrických šifer je, že se počítá s malými čísly. Modulo n je obvykle veliké číslo. Pro e délky 1024 bitů je pro výpočet $x^e \,\mathrm{mod}\, n$ potřeba provést 2^{1024} násobení (odhadovaný počet atomů ve vesmíru je 2^{300}). Toto je zcela zásdaní, neboť bez rychlého umocnění by nebylo šifrování RSA použitelné. Pro urychlení se používá kombinace násobení (`MUL`) a výpočtu druhé mocniny (`SQ`).

$$x^9 : x \overset{\text{SQ}}{\frown}\!\!\rightarrow x^2 \overset{\text{SQ}}{\frown}\!\!\rightarrow x^4 \overset{\text{SQ}}{\frown}\!\!\rightarrow x^8 \overset{\text{MUL}}{\frown}\!\!\rightarrow x^9$$

Obecně se díváme na číslo tak, že má binární zápis exponentu. `MUL` pak vkládá na nejméně významnou pozici jedničku a `SQ` posouvá 1 v binárním zápise doleva a na nejméně významnou pozici vkládá 0.

$$x^{26} = x^{11010} : x^1 \overset{\text{SQ}}{\frown}\!\!\rightarrow x^{10} \overset{\text{MUL}}{\frown}\!\!\rightarrow x^{11} \overset{\text{SQ}}{\frown}\!\!\rightarrow x^{110} ...$$

Složitost rychlého umocnění

Počet `SQ` operací je roven délce bitového zápisu exponentu e (`#SQ` $= t$). Počet operací `MUL` je roven *Hammingově váze* e (Hammingově vzdálenosti e od nuly). Pro průměrný počet `MUL` operací platí `#MUL` $= 0{,}5t$ a platí tedy

$$\#\text{SQ} + \#\text{MUL} = 1,5t$$

Složitost tohoto algoritmu je tedy linerání.

Pokud by měl exponent 1024 bitů, pak při přímém umocnění by se jednalo cca o 10^{300} operací `MUL`, zatímco při použití rychlého umocnění je to 1516 operací `MUL` a `SQ`.

9.7 Šifrování založené na eliptických křivkách (ECC)

Stačí menší veliksot klíčů (256 bitů), což přináší nižší výpočetní náročnost. Ale šifrování je velmi silně bezpečné.

Eliptická křivka **Definice 47**

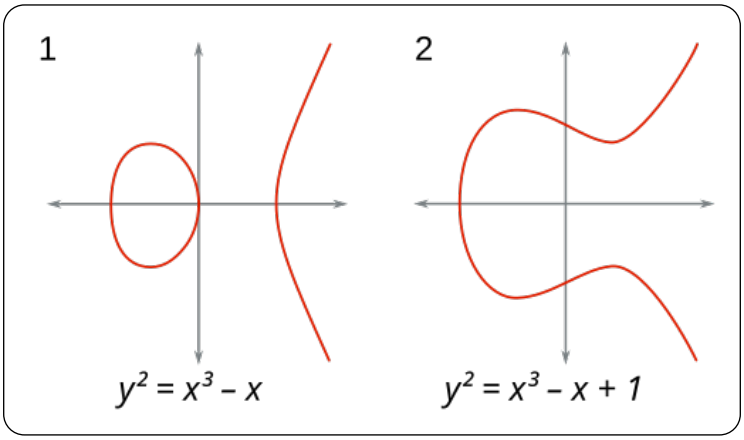
Je definováno jako

$$y^2 = x^3 + ax + b$$

kde

$$4a^3 + 27b^2 \neq 0$$

Navíc definujeme i **nevlastní bod** $\boldsymbol{O}$ (bod ležící v nekonečnu libovolné vertikální přímce). Operace sčítání bodů lze definovat jak geometricky, tak algebraicky nad prvočíselnými poli.



Obrázek 13: Eliptické křivky.

Pro součty platí několik vlastností:

$$P + O = O + P = P$$

$$P + (-P) = 0$$

$$(P + Q) + R = P + (Q + R)$$

$$P + Q = Q + P$$

Bezpečnost ECC je založená na obtížnosti problému jak určit n pro daný bod P a násobek $Q = n \cdot P$.

Postup šifrování

1. Alice a Bob se dohodnou na prvočíslu p, eliptické křivce E a na bodu $P \in E$
2. Bob vygeneruje klíče (soukromý n a veřejný $Q = n \cdot P$)
3. Alice si zvolí tajné číslo k (efemérní klíč)

4. Alice azširuje zprávu M:

$$C_1 = k \cdot P$$

$$C_2 = M + k \cdot Q$$

5. vznikne kryptogram $\langle C_1, C_2 \rangle$

Postup dešifrování

1. Bob vypočítá $M = C_2 - n \cdot C_1$ *(ano je to kompletní)*

10 Digitální podpis

Jedná se o připojení identifikačích údajů autora k dokumentu. Zaručuje autentičnsot zprávy a obvykle i integritu podepsaného dokumentu. Využívá se asymetrického šifrování, kryptografické hashovací funkce a digitálních certifikátů.

Hlavní myšlenka je založena na obrácení použití veřejného a soukromého klíče. Tento přístup se však v praxi nepoužívá, protože by trval moc dlouho.

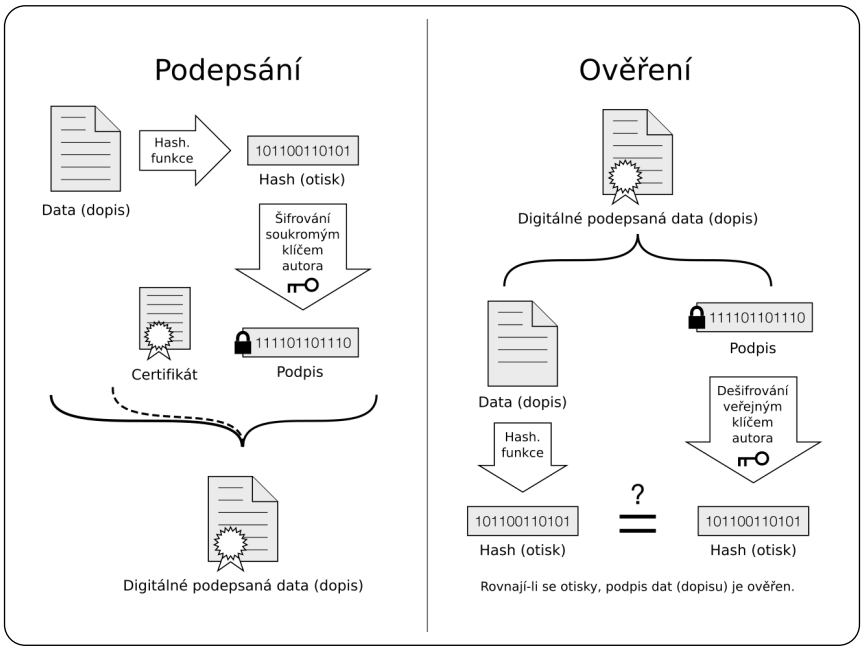
U odeslané zprávy se spočítá hash, který umožní ověřit její integirtu a urychlí celý prcoes (šifruje se pouze hash).

Digitální certifikát **Definice 48**

Digitální certifikát je elektronický dokument obsahující:

- veřejný klíč
- identifikační údaj o majiteli klíče
- informaci o certifikační autoritě

Cetrifikační autorita vytvoří digitální certifikát tak, že ověří údaje o majiteli veřejného klíče a veřejný klíč digitálně podepíše. Tento platný podpis zaručí, že s certifikátem nebylo od vydání manipulováno. Pokud tedy věříme autoritě, můžeme věřit i certifikátu jím podepsaném.



Obrázek 14: Podepsání a oveření digitálního podpisu

11 Zero knowledge proofs

Motivační příklad. Jak dokázat barvoslepému člvoěku, která tužka ze dvou má jakou barvu, když jsou až na onu barvu identické.

V tomto přístupu jsou dvě strany – prover a verifier. Ti spolu interagují. Prover má přesvědčit verifiera o pravdivosti tvrzení, aniž by mu sdělil přímý důkaz tohoto tvrzení.

11.1 Fiat-Shamirův identifikační protokol

1. Bezpečnostní server (verifier) vygeneruje RSA modul $n = p \cdot q$
2. n zveřejní, p a q zahodí
3. uživatel (prover) náhodně vygeneruje tajné číslo s, dále vypočítá číslo $v = s^2 \bmod n$, které pošle serveru
4. později se chce uživatel identifikovat důkazem, že zná s, aniž by s sdělil

Jeden cyklus protokolu:

1. uživatel náhodně zvolí r; dále vypočítá $x = r^2 \bmod n$, které pošle serveru
2. server náhodně zvolí číslo $b \in \{0, 1\}$, které pošle uživateli
3. uživatel vypočítá číslo $y = (r \cdot sb) \bmod n$, které pošle serveru
4. server ověří $y^2 \equiv x \cdot vb (\bmod n)$
5. pokud kongruence platí, pak server akceptuje cyklus