GYMNÁZIUM BRNO, SLOVANSKÉ NÁMĚSTÍ

Asymetrická kryptografie a šifrovací systém RSA

ROČNÍKOVÁ PRÁCE

obor Informatika

AUTOR: František Čech

VEDOUCÍ PRÁCE: Mgr. Petr Vozák

Brno 2023

Anotace

Tato práce se zabývá tématem asymetrické kryptografie, což je odvětví kryptografie, které se věnuje zabezpečené komunikaci pomocí klíčového páru. V práci je od základů popsán princip kryptografie, zabezpečené komunikace a principů šifrování i kryptografických podpisů. Je vysvětlen princip kryptosystému RSA a jeho matematické základy, především modulární aritmetika. Součástí práce je dále implementace RSA v jazyce Typescript.

Klíčová slova

šifra; RSA; prvočísla; klíčový pár; asymetrická kryptografie; modulo

Annotation

This paper deals with the topic of asymmetric cryptography, which is a branch of cryptography that deals with secure communication using a key pair. The paper describes the principles of cryptography, secure communication, encryption, and cryptographic signatures from the ground up. Furthermore, the principle of the RSA cryptosystem and its mathematical basis, especially modular arithmetic, are explained. In addition, the paper contains a Typescript implementation of RSA.

Keywords

cipher; RSA; prime numbers; key pair; asymmetric cryptography; modulo

Obsah

1 Úvod 4
2 Asymetrické šifrování 5
2.1 Obecná kryptografie 5
2.1.1 Zásady a terminologie 5
2.1.2 Notace a vlastnosti 6
2.2 Symetrie šifry 6
2.2.1 Symetrická šifra 7
2.2.2 Asymetrická šifra 7
2.2.3 Kryptografický podpis 7
2.3 Princip asymetrické kryptografie 8
2.3.1 Použití kryptografického podpisu 8
2.4 Bezpečnost 9
3 Implementace šifry RSA 9
3.1 Modulární aritmetika 10
3.1.1 Kongruence 10
3.1.2 Modulární inverze 11
3.1.3 Implementace v kódu 11
3.2 Princip RSA 12
3.2.1 Implementace v kódu 12
3.3 Vytvoření klíčů 13
3.3.1 Implementace v kódu 13
3.4 Vlastnosti 14
3.5 Bezpečnost 15
4 Závěr 16
5 Použitá literatura 17
5.1 Použité obrázky 18
6 Příloha: Matematický důkaz RSA 19
7 Příloha: Soubory Implementace RSA 19
7.1 Soubor rsa-en.ts 20
7.2 Soubor tsconfig.json: 22

1 Úvod

Asymetrická kryptografie je odvětví kryptografie věnující se způsobům, jež umožňují šifrovanou komunikaci odesílatele a příjemce pomocí klíčového páru. Asymetrické šifry jsou díky svým bezpečnostním výhodám bezesporu jedním z nejdůležitějších bezpečnostních prvků moderního světa, především online komunikace.

Autor si téma vybral z důvodu dřívějšího zájmu a důležitosti dané tématiky pro jeho budoucí studium a obor. Zaujaly ho matematické principy za dnešními asymetrickými šiframi, chtěl jim hlouběji porozumět a pokusit se algoritmy sám implementovat.

Tato práce se zabývá současnými systémy asymetrické kryptografie používanými ve výpočetní technice. Podobných šifer vzhledem k náročnosti jejich vývoje není mnoho. Byla tedy vybrána šifra RSA, jelikož je jednou z mála šifer, kterou lze popsat a jednoduše implementovat v rámci takto krátké práce. Nová látka potřebná k popisu RSA, včetně asymetrické kryptografie či modulární aritmetiky, je uváděna postupně, jelikož není součástí středoškolského učiva. Zatímco první část práce si za cíl klade srozumitelný úvod do kryptografie a objasnění pojmu asymetrického šifrování, cíl druhé části práce spočívá v popsání kryptosystému RSA a vytvoření jeho implementace v nadstavbě Typescript pro programovací jazyk Javascript.

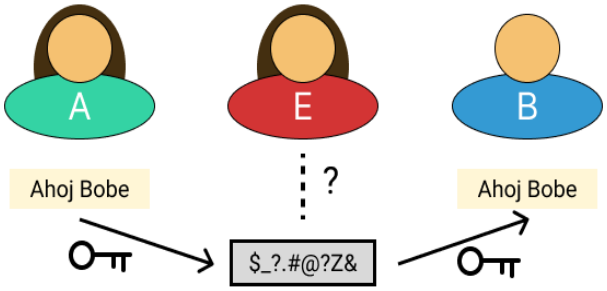
2 ASYMETRICKÉ ŠIFROVÁNÍ

V informatické bezpečnosti pojem šifrování označuje proces převodu otevřených dat na šifrovanou podobu za účelem skrytí jejich obsahu či zajištění jejich pravosti [13]. Tento princip se stal nezbytnou součástí moderní komunikace, je ale důležité správně vybrat typ používaného šifrování dle principů kryptografie.

2.1 Obecná kryptografie

Kryptografie je disciplína zahrnující zásady a způsoby šifrování a ochranu dat proti neautorizovaným použitím či změnám [3]. Pro uvedení následující teorie do kontextu použijeme analogii často používaných imaginárních postav Alice a Boba, kteří mezi sebou chtějí bezpečně předávat zprávy, a Evu (z anglického *eavesdropper* – slídil/špeh), která se jejich zprávy snaží odhalit.

Alice chce předat Bobovi dopis s datem tajné schůzky. Nevěří ale pošťačce Evě, která sice zásilky musí doručit, ale často si dopisy pročítá. Zamkne tedy dopis do pomyslné schránky klíčem, který dala i Bobovi. Eva tento klíč nemá a nemůže schránku otevřít, aby se dozvěděla, co je uvnitř. Ale Bob ji může odemknout klíčem od Alice a číst si dopis s pocitem, že byl poslán bezpečně. V kryptografii odpovídá „zamykání krabičky“ procesu šifrování. Namísto schránky by Alice použila nějakou šifru, kterou by dopis změnila tak, že by byl po cestě nečitelný.

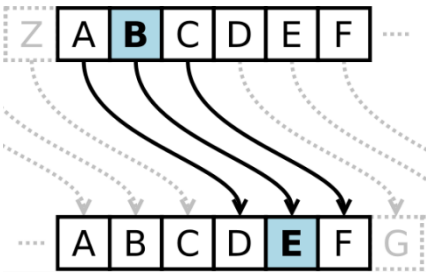


Obrázek 1: Znázornění analogie Alice a Boba

2.1.1 Zásady a terminologie

Obecně kryptografie pracuje s daty, tedy jednotkami informací, jež mohou být interpretovány jako čísla, text, binární řetězec apod [7]. Původní formu čitelné zprávy (*message*), tedy srozumitelných dat označujeme jako prostý či otevřený text (*plaintext*). Jejich zašifrovanou, tedy nečitelnou a nesrozumitelnou podobu pak jako šifrovaný text (*ciphertext*) [12]. Proces převádění prostého textu na text šifrovaný nazýváme zašifrování (*encryption*), opakem pak rozumíme dešifrování (*decryption*), tedy převod šifrovaného textu zpět na text otevřený. Šifrování se často mylně zaměňuje za kódování (*encoding*), tedy obecný převod formátu dat, například převod písmen v textu na čísla jejich pozic v abecedě. Aby šifra dosáhla svého účelu je nutné, aby šlo dešifrování provést pouze s určitou tajnou znalostí. Pro bezpečnost je důležité, aby tato znalost nebyl samotný postup šifry, protože jeho odhalení by samo o sobě navždy oslabilo její bezpečnost. Tajnou znalost proto určují data, která můžeme při každé komunikaci měnit, označovaná jako klíč (*key*).

Pro příklad lze uvést jednoduchou Caesarovu šifru. Zašifrování definujeme jako posunutí každého písmeny zprávy o tři pozice dopředu v abecedě ($AHOJ \rightarrow DKRM$). Dešifrování pak obdobně s posunutím dozadu ($DKRM \rightarrow AHOJ$). Tuto šifru jako jednu z prvních doložených použití kryptografie používal slavný Julius Caesar pro komunikování tajných zpráv svým velitelům [6]. Z dnešního pohledu šlo o šifru prolomitelnou zcela primitivně, ale frekvenční analýza (metoda pro prolomení tohoto typu šifer) nebyla vynalezena po další stovky let. Caesar však musel spoléhat na to, že se o šifře jeho nepřítel nedozví. Jakmile totiž její princip někdo odhalí, je hned schopen rozluštit všechny zprávy. Rozhodně to není typ šifry, na jejímž základě by šel vytvořit kryptografický standard pro světovou komunikaci.



Obrázek 2: Schéma Caesarovy šifry

Moderní šifry proto následují tzv. Kerckhoffsův princip, pravidla jejich fungování jsou veřejně známa a mohou být podrobena analýze kryptoanalytiků[1]. Rozhodujícím faktorem pro prolomení šifry tak není kolik lidí o ní ví, ale bezpečnost šifry samotné a použitý klíč. Prolomením šifry je myšlena možnost zjištění původního otevřeného textu ze šifrovaného textu bez znalosti klíče.

2.1.2 Notace a vlastnosti

Bezrozměrná data jsou označována proměnnými s velkým prvním písmenem jejich anglického názvu (např. zpráva P – pro *plaintext*). Podobně pak procedury zašifrování a dešifrování v notaci obvykle odpovídají funkcím E a D s případným subscriptem. Pro funkční kryptosystém existují určité vlastnosti, tedy podmínky, které musí splňovat:

a) Zašifrováním otevřeného textu získáme šifrovaný text C, ten je nečitelný a lze ho poslat i přes odposlouchávaný komunikační kanál.

$$E(P) = C$$

b) Provést D v reálném čase je možné pouze pro příjemce zprávy, který disponuje správným šifrovacím klíčem. D je inverzní operací pro E a vrací původní zprávu:

$$D(C) = P = D\big(E(P)\big)$$

2.2 Symetrie šifry

Šifry, kterými se kryptografie zabývá, dělíme na symetrické a asymetrické. Liší se vlastnostmi, a především použitím v reálném světě. Symetrii šifry určuje typ klíče, který se používá pro zašifrování a dešifrování.

[1] Kryptoanalýza je disciplína zabývající se luštěním a lámáním šifer, úzce související s kryptografií. Spolu tvoří disciplínu kryptologie. [11]

2.2.1 Symetrická šifra

Symetrická šifra používá stejný „symetrický“ klíč k na straně odesílatele (Alice) i příjemce (Boba). Definice zašifrování i dešifrování jsou proto identické.

$$E_A = E_B, \;\; D_A = D_B$$

V tomto případě tedy musí E i D vždy zůstat tajné, aby nedošlo k vyzrazení klíče k, což je jediná tajná hodnota, na které obě tyto funkce závisí. Pro symetrické šifry obecně platí, že jsou méně výpočetně náročné než asymetrické [12]. Symetrické šifrovací programy jsou tak schopné rychle zpracovat velká množství dat.

2.2.2 Asymetrická šifra

Asymetrické šifry, nebo též šifry veřejného klíče, se vyznačují použitím dvou různých šifrovacích klíčů. Jeden pro šifrování na straně odesílatele, druhý pro šifrování na straně příjemce. Šifrovací klíč je veřejný a poslat zašifrovanou zprávu může kdokoli, ale dešifrovat ji může pouze příjemce se správným dešifrovacím klíčem, který je soukromý. Aby pak byla zajištěna vlastnost b) z kapitoly 2.1.2, musí platit:

c) Funkce D nejde zjistit z E, je tajná i přesto, že E je veřejně známa a jednoduše proveditelná

Pokud jsou splněny podmínky a), b) a c), pak splňuje E vlastnosti „trap door“ funkce [11], tedy jednosměrná funkce. Funkci k ní inverzní nelze technicky provést za pouhé znalosti jejího výstupu [4]. Právě jednosměrné funkce jsou jedním z nezbytných principů pro bezpečnost asymetrických kryptosystémů.

2.2.3 Kryptografický podpis

Kryptografické systémy používající veřejné klíče jsou také velice užitečné díky tomu, že umožňují ověřování totožnosti odesílatele zprávy. Technologie kryptografických certifikátů, zajišťující i bezpečnost https protokolu závisí právě na kryptografických podpisech. Aby bylo možné kryptografií implementovat tuto funkci, musí kryptosystém splňovat ještě jednu důležitou podmínku [8]:

d) Provedení procedur v opačném pořadí stále vede k původnímu otevřenému textu:

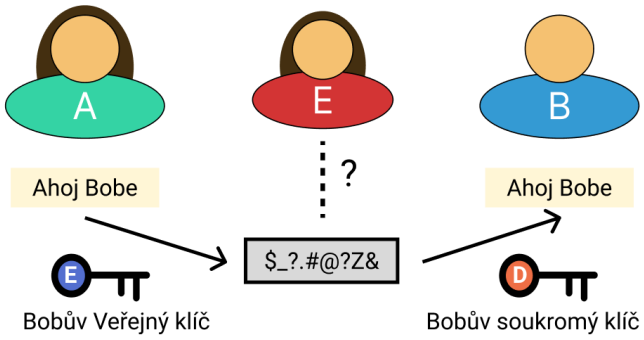
$$E\big(D(P)\big) = P = D\big(E(P)\big)$$

Tato podmínka má důležitou implikaci, že každý otevřený text je zároveň šifrovaným textem pro další otevřený text. Mají i stejný typ dat, jelikož argumenty a výstupy funkcí E a D musí být též stejné. Množiny možných otevřených a šifrovaných textů se tedy rovnají a pro P i C lze použít jednotné označení M [2].

2.3 Princip asymetrické kryptografie

Asymetrický kryptosystém implementuje E a D tak, aby byly splněny podmínky a) až c) z předchozích kapitol. Existuje několik asymetrických kryptosystémů, mimo RSA např. ElGamal, Diffie-Hellman, či novější ECC (kryptografie nad eliptickými křivkami). Používání asymetrické šifry však vypadá jinak než příklad v kapitole 2.1, což byla zjednodušená ukázka symetrické šifry.

Výměna zprávy pomocí asymetrické šifry začíná vytvořením tzv. klíčového páru. Bob vygeneruje pár dvou matematicky propojených klíčů. První klíč je veřejný a Bob jej může volně rozeslat komukoli na veřejné síti, tento klíč umožňuje provedení E. Druhý klíč je soukromý a Bob jej musí udržet v naprosté tajnosti, tento klíč pak umožňuje provedení D. Alice či kdokoli jiný pak může poslat Bobovi bezpečně zašifrovanou zprávu, kterou může dešifrovat jen Bob, díky podmínce b). Přestože je veřejný klíč dostupný všem, dešifrovat zprávu může jen vlastník soukromého klíče, tedy Bob, díky podmínce c).



Obrázek 3: Obrázek analogie Alice a Boba s použitím asymetrické šifry

$$D_B\big(E_B(M)\big) = M$$

Pokud by chtěl Bob odpovědět, musel by znát veřejný klíč Alice, nebo by mu Alice mohla bezpečně poslat klíč pro nějakou symetrickou šifru, kterou by komunikovali. Právě druhý způsob použití je ten nejčastěji využívaný. Vzhledem k menší rychlosti a efektivitě asymetrických šifer se používají pouze k ustanovení symetrického klíče pro jinou, rychlejší šifru [9].

2.3.1 Použití kryptografického podpisu

V kapitole 2.2.3 byly stanoveny podmínky které musí kryptosystém splňovat pro implementaci kryptografického podpisu. Kryptografický podpis je používán k autorizaci. Jak např. ověřit, že zprávu, která přišla od Boba, skutečně poslal on? Bob může použít svůj soukromý klíč a použít $D_B(M)$ na unikátní M a poslat výsledný C i M. Jakýkoli příjemce pak díky podmínce d) může ověřit že

$$E_B(C) = M \Leftarrow E_B\big(D_B(M)\big)$$

Tato rovnost bude platit pouze pokud byl k zašifrování použit správný soukromý klíč, tedy můžeme ověřit, že odesílatel je správný.

2.4 Bezpečnost

Bezpečnost kryptografických systémů závisí na splnění podmínky b) a tím pádem především podmínky c) pro asymetrické šifry. Obě závisí na tom, že díky nějakému nevyřešitelnému problému pro útočníka nelze zjistit *E*. Některé kryptografické systémy jsou podloženy matematickými důkazy pro obtížnost těchto problémů, jiné spoléhají na tom, že se doposud kryptologické komunitě šifru nepodařilo prolomit. Existuje mnoho druhů útoků na šifru [1]:

1) Útok hrubou silou: Při tomto typu útoku útočník zkouší všechny možné klíče, dokud nenajde správný klíč. Tento přístup je časově náročný, ale může být úspěšný u slabých šifer s malými velikostmi klíčů.

2) Útok se znalostí otevřeného textu nebo také s vybraným otevřeným textem: Při tomto typu útoku má útočník přístup jak k šifrovanému textu, tak k odpovídajícímu otevřenému textu. Analýzou známých dvojic prostého textu a šifrovaného textu může útočník odvodit informace o klíči nebo dokonce o celém šifrovacím algoritmu. Rozdíly mezi šifrovaným a otevřeným textem se zabývá i útok typu diferenciální kryptoanalýzy.

3) Útoky typu Man-in-the-middle: Při tomto typu útoku útočník zachycuje komunikaci mezi dvěma stranami a přenáší mezi nimi zprávy tam a zpět, čímž efektivně odposlouchává konverzaci. To může útočníkovi umožnit zachytit a upravit zprávy nebo získat přístup k citlivým informacím. V předchozí analogii by např. Eva dala Alici svůj veřejný klíč a vydávala by se za Boba. Zprávu by dešifrovala, případně i upravila a Bobovi by poslala zašifrovanou jeho veřejným klíčem.

4) Útoky postranním kanálem: Při tomto typu útoku útočník využívá slabiny ve fyzické implementaci šifrovacího algoritmu, jako je spotřeba energie nebo elektromagnetické emise k odvození informací o klíči nebo prostém textu.

5) Matematický útok: Slabiny v matematickém principu fungování šifry mohou být použity k jejím prolomení.

3 IMPLEMENTACE ŠIFRY RSA

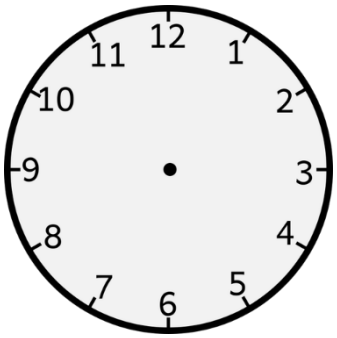
V roce 1977 vydali Ron Rivest, Adi Shamir a Len Adleman práci „A Method for Obtaining Digital Signatures and Public-Key Cryptosystems“ [10], ve které odhalili první funkční asymetrický kryptosystém, RSA. Šlo o první, dodnes široce využívanou asymetrickou šifru a bezpochyby jeden z nejdůležitějších objevů v historii moderní komunikace [9]. V této práci se drželi podnětu z práce „New Directions in Cryptography“ od autorů Whitfield Diffie a Martin E. Hellman [2] vydané o rok dříve, ve které byl návrh, jak by měl takový kryptosystém fungovat, konkrétní systém však vymysleli až autoři RSA.

Tato kapitola se zabývá matematickou podstatou kryptosystému RSA a její následné implementaci v Javascriptu. Do teorie jsou proto průběžně prokládány praktické ukázky ale i výstřižky Javascript kódu použitého ve finální implementaci. Tato kapitola se nevěnuje základním znalostem jazyka Javascript potřebným k programování.

3.1 Modulární aritmetika

Na modulární aritmetice závisí celý systém RSA. Základním operátorem modulární aritmetiky je operátor mod, který vyjadřuje zbytek po celočíselném dělení. Jak například vyčíslit hodnotu výrazu $23\ mod\ 9$? Číslo 9 je v tomto případě pomyslný dělitel čísla 23. Celočíselným dělením získáme tento výsledek:

$$\frac{23}{9} = 2, zbytek\ 5 \ \Rightarrow\ 23\ mod\ 9 = 5$$



Obrázek 4: Analogové hodiny

Tento princip není využíván pouze v kryptografii a počítačové technice, nýbrž i v každodenním všedním životě. To lze nejlépe ilustrovat na analogových hodinách. Pro příklad uvažujme hodinovou ručičku v poloze 10 hodin. Posuneme-li ji o 5 hodin, shledáme, že je v poloze 3 hodiny, neboť $(10 + 5)\ mod\ 12 = 3$. Ručička pro minuty či sekundy pak funguje modulo 60.[2] Pro lepší porozumění následující teorii je tak možné si každý výpočet modulo x představit jako hodiny s x pozicemi. Jediným rozdílem by byl fakt, že na pozici s hodnotou modulu (např. dvanáct) by měla být nula, aby hodnota odpovídala výpočtu, např. $12\ mod\ 12 = 0$.

3.1.1 Kongruence

Kongruence je zvláštní typ ekvivalence úzce spojený s modulární aritmetikou. Modulární aritmetika operuje na množině celých čísel Z_n vzniklé ze zbytků celočíselného dělení modulem n pro všechna celá čísla. Pro účely popsání algoritmu RSA však budeme převážně uvažovat pouze nezáporná celá čísla, tedy patřící do $N \cup \{0\}$.

Celá čísla a a b jsou kongruentní modulo n, pokud mají stejný zbytek po celočíselném dělení n, tedy n dělí jejich rozdíl:

$$a \equiv b\ (mod\ n) \quad \Leftrightarrow \quad (a - b) = kn \quad \Leftrightarrow \quad (a\ mod\ n) = (b\ mod\ n)$$

Z této definice lze odvodit řadu dalších vlastností. Pro celá čísla a, b, c, d, n a případné libovolné k platí následující pravidla kongruence [5]:

1) Pro kongruenci platí reflexivita, tedy $a \equiv a\ (mod\ n)$
2) Pro kongruenci platí symetrie, tedy pokud $a \equiv b\ (mod\ n)$, pak $b \equiv a\ (mod\ n)$
3) Kongruence je tranzitivní, vzájemná kongruence mezi kongruentními čísly musí platit mezi všemi z nich:

$$a \equiv b\ (mod\ n)\ \wedge\ b \equiv c\ (mod\ n) \quad \Rightarrow \quad a \equiv c\ (mod\ n)$$

[2] Caesarova šifra zmíněná v kapitole 2.1.1 posouvá pozice písmen modulo 26, po překročení Z (index 25) jde zpět na A (index 0).

4) Pro kongruenci platí základní ekvivalentní úpravy aritmetických operací včetně vzájemného sčítání, odčítání a násobení; pokud $a \equiv b\ (mod\ n)$ a $c \equiv d\ (mod\ n)$, potom platí:

$$a + c \equiv b + d\ (mod\ n)$$

$$ac \equiv bd\ (mod\ n)$$

$$a^k \equiv b^k\ (mod\ n)$$

Každé celé číslo je kongruentní alespoň s jedním číslem v množině $Z_n = \{\ 0, 1, \ldots, |n| - 1\ \}$, označované jako celá čísla modulo *n*. Množina všech kongruentních čísel s *a* modulo *n* se nazývá třída ekvivalence[3] *a*; vznikne definicí $\{\ kn + a \mid k \in Z\ \}$.

3.1.2 Modulární inverze

V předchozí podkapitole byly definovány operace včetně sčítání, odčítání, násobení i umocňování. Ale „klasické“ dělení[4], stejně jako pro celá čísla, není definováno. Definujeme však modulární multiplikativní inverzi. Uvažujme $a \in Z_n$. Pro prvek *a* lze definovat prvek inverzní prvek $x \in Z_n$, který splňuje

$$a \cdot x \equiv 1\ (mod\ n)$$

Avšak ne pro všechna *a* existuje inverzní prvek. Existuje pouze pokud je číslo *a* nesoudělné s modulem *n* [9], tedy

$$nsd(a, n) = 1$$

Pokud je tato podmínka splněna, číslo *a* je invertibilní [5] a jeho inverzní prvek se označuje a^{-1}. Proces nalezení inverzního prvku lze provést vyzkoušením všech čísel od 1 do *n*, jde však urychlit použitím tzv. Euklidova algoritmu [9].

3.1.3 Implementace v kódu

V Javascriptu je implementován operátor modulo pomocí znaku procenta %. Ukázka:

```
let remainder = 11 % 4
console.log(remainder)   //vypíše hodnotu 3
```

Dále je součástí kódu funkce `modular_power` která vrací výsledek modulární exponenciace, ale používá algoritmus, který přesahuje rozsah této práce. Stejně tak je součástí kódu funkce `modular_inversion`, která vrací výsledek modulární inverze, ale také používá algoritmus, který přesahuje rozsah této práce.

[3] Třída ekvivalence se také někdy v literatuře označuje jako zbytková třída, což indikuje, že každá zbytková třída odpovídá právě jednomu nejmenšímu zbytku, tedy číslu z celých čísel modulo *n*.

[4] Dělení lze v modulární aritmetice definovat jako násobení inverzním prvkem (např. vydělení čísla *a* číslem *b*), tedy: $\frac{a}{b} \equiv a \cdot b^{-1}\ (mod\ n)$ a je pak definované pouze za podmínky, že je *b* invertibilní modulo *n*.

3.2 Princip RSA

RSA je asymetrická šifra a musí mít veřejný a soukromý klíč. Oba klíče jsou reprezentovány dvojicí čísel – veřejný či soukromý šifrovací exponent a *n*:

$$Veřejný\ klíč = (e, n)$$
$$Soukromý\ klíč = (d, n)$$

Proces získání těchto matematicky provázaných klíčů je vysvětlen v následující kapitole. Důležitá vlastnost však je, že pro zprávu *m* platí:

$$m^{e \cdot d} \equiv m\ (mod\ n)$$

Důkaz této vlastnosti lze nalézt jako přílohu v kapitole 6. V RSA jsou procedury *E* a *D* definovány jako exponenciace modulo *n* [9]:

$$E(M) = M^e\ mod\ n$$
$$D(M) = M^d\ mod\ n$$

Následujme analogii Alice a Boba z kapitoly 2.3. Po vytvoření klíčového páru Bob pošle Alici (e, n) a ponechá (d, n) v tajnosti. Pokud chce Alice poslat Bobovi celočíselnou zprávu $m \in Z_n$, pošle zašifrovanou formu *c*:

$$c = E_B(m) = m^e\ mod\ n$$

Bob šifrovanou zprávu obdrží a provede *D*.

$$D_B(c) = c^d\ mod\ n = m$$

Získá tím originální zprávu *m*. To platí, protože *m* je nejmenším zbytkem:

$$c^d \equiv m^{e \cdot d} \equiv m\ (mod\ n) \quad \wedge \quad m \in Z_n \quad \Rightarrow \quad c^d\ mod\ n = m$$

3.2.1 Implementace v kódu

V kódu byla vytvořena třída `RSAKey`, která reprezentuje jeden klíč. Má hodnotová pole pro exponent a *n*.

```
export class RSAKey {

	private exponent: number
	public mod: number

	constructor(exponent: number, mod: number){

		this.exponent = exponent
		this.mod = mod
	}
```

```
	process(message: number): number { //encrypt = decrypt

		return modular_power(message, this.exponent, this.mod)
	}

}
```

3.3 Vytvoření klíčů

Následujme postup z původní práce RSA z roku 1977. Určení soukromého a veřejného klíče pro funkce *E* a *D* je prováděno následujícím způsobem [10]:

1) Náhodný výběr dvou odlišných velkých prvočísel p a q

2) Výpočet modulu $n = pq$ a hodnoty Eulerovy funkce

$$\varphi(n) = (p-1)(q-1)$$

Eulerova funkce φ je použita pouze v tomto kroku, její definici lze nalézt v příloze v kapitole 6.

3) Náhodný výběr soukromého šifrovacího exponentu d z rozmezí $< 1; \varphi(n) >$, takovým způsobem, aby byl s $\varphi(n)$ nesoudělný, tedy $nsd\big(d, \varphi(n)\big) = 1$

4) Výpočet veřejného šifrovacího exponentu e pomocí inverze d modulo $\varphi(n)$, tak aby:

$$ed \equiv 1 \left(mod\ \varphi(n)\right)$$

Nahlédneme-li však do novější literatury [1] moderních implementací, najdeme mnohé úpravy této originální definice. Vzhledem k tomu, že provedení modulární inverze není možné bez znalosti p a q, většina implementací prvně zvolí e a následně vypočítá d modulární inverzí. S tímto postupem můžeme standardizovat e na jednu hodnotu, nejčastější je např. 65537. V tomto případě je potřeba zvolit p a q takovým způsobem, aby e bylo stále nesoudělné s $\varphi(n)$.

3.3.1 Implementace v kódu

V kódu byla vytvořena třída `RSAKeyPair`, která reprezentuje klíčový pár. Má hodnotová pole `privateKey` a `publicKey` pro soukromý a veřejný klíč.

```
export class RSAKeyPair {

	public publicKey: RSAKey
	public privateKey: RSAKey

	static generate(prime1: number = 0, prime2: number = 0, exponent: number =
7){

		let e = exponent
```

```
            if(prime1 % e == 1 || prime2 % e == 1) //assert modular invertibility
under phi(n)
                  throw new Error("Exponent must be coprime with totient of n")

            while(!prime1 || prime1 % e == 1)
                  prime1 = random_prime()

            while(!prime2 || prime2 % e == 1 || prime2 == prime1) // prevent
factorization to n = p*p
                  prime2 = random_prime()

            let p = prime1,
                  q = prime2

            let n = p * q
            let totient = (p - 1) * (q - 1)
            let d = modular_inversion(e, totient)

            return new this(new RSAKey(e, n), new RSAKey(d, n))
      }

      constructor(publicKey: RSAKey, privateKey: RSAKey){

            this.publicKey = publicKey
            this.privateKey = privateKey
      }

}
```

Hodnoty p a q jsou vybrány z vygenerovaného seznamu prvočísel o velikosti jednoho bytu, tedy menších než číslo 256. Pro tento náhodný výběr je definována funkce `random_prime()`.

3.4 Vlastnosti

Není je třeba matematicky podložit požadované vlastnosti asymetrické šifry z kapitoly 2.2.2 pomocí pravidel modulární aritmetiky z kapitoly 3.1.

a) RSA splňuje podmínku a) protože zašifrováním získáme šifrovaný text *c*.

b) Pro provedení *D* je třeba znát soukromý šifrovací exponent *d*, RSA tedy splňuje i podmínku b). Dešifrování *c* vede k původnímu zprávě.

c) Celá bezpečnost RSA závisí na podmínce c). Exponent *d* by stále šel zjistit z *e* pouze modulární inverzí modulo $\varphi(n)$, ale k vypočítání této hodnoty je potřeba znát p a q. Musí to proto být velká prvočísla, aby veřejné n nešlo jednoduše faktorizovat. Faktorizací rozumíme rozklad celého čísla na prvočíselné činitele. Na problém faktorizace dodnes neexistuje algoritmus dostatečně efektivní, aby faktorizoval číslo o délce např. 2048 bitů za dostatečně krátkou dobu, aby šlo považovat šifru RSA za prolomenou [9].

d) RSA splňuje i podmínku d), jelikož násobení e a d v exponentu je komutativní, je tedy jedno v jakém pořadí je provedeno.

$$m^{e \cdot d} = (m^e)^d = (m^d)^e$$

$$D\big(E(M)\big) = E\big(D(M)\big)$$

3.5 Bezpečnost

Bezpečnost kryptosystému RSA není dokonalá. Při implementaci této šifry pro komerční použití je nutné se vyvarovat možných útoků, především těch nejrozšířenějších z kapitoly 2.4. Je důležité poznamenat, že implementace, která je součástí této práce, není bezpečná, je vytvořena právě tak, aby na ni byly tyto útoky aplikovatelné.

V útoku hrubou silou útočník zkouší všechna možná d, dokud nenarazí na správnou hodnotu. Tomuto útoku mohou podlehnout implementace s krátkou délkou klíče. Pokud je klíč dostatečně dlouhý, vyzkoušení všech možných d není prakticky možné.

Mezi nejdiskutovanější útoky patří matematický útok faktorizace n. Při úspěšném rozkladu na $n = pq$ může útočník vypočítat hodnotu Eulerovy funkce $\varphi(n) = (p-1)(q-1)$, provést modulární inverzi e a získat d. Jak však již bylo řečeno, efektivní způsob faktorizace pro ohrožení bezpečnosti RSA doposud neexistuje [9]. Další matematické útoky budou vždy alespoň stejně výpočetně náročné jako faktorizace n, a mohou tedy být vyřešeny zvětšením délky klíčů [8].

Mnohé implementace RSA jsou zranitelné vůči útokům postranním kanálem, specificky časovacím útokům. Provedení výpočtů modulární aritmetiky je pro člověka nepostřehnutelně rychlé, ale pro velká čísla se mohou objevit časové prodlevy, než je výpočet dokončen. Velikost čísel lze pak odvodit dle rychlosti výpočtů. Pokud má útočník fyzický přístup k zařízení, může také velikost odhadnout dle tepla vydaného procesorem, jelikož tyto výpočty procesor značně zatěžují. Prevence spočívá v implementaci výpočtu takovým způsobem, že proces vždy trvá stejnou dobu.

Existují různé útoky s více vybranými šifrovanými texty, které využívají vlastností modulární aritmetiky k výpočtu soukromého klíče, otevřených textů, nebo jejich částí. Těmto útokům se dá však předejít tím způsobem, že je každá zpráva doplněna o unikátní náhodnou hodnotu. Dnešní implementace využívají systému PKCS#1, který chrání RSA před těmito útoky [1].

Co se týče budoucnosti, pokud by byla šifra RSA prolomena např. kvantovým počítačem, musela by ji nahradit kvantově odolná šifra. Zatím největší hrozbou je však faktorizace. Na začátku 20. století byly široce používány klíče o délce 1024 bitů, ale v rámci projektu RSA factoring challenge byly prolomeny klíče až o délce 768 bitů (přes 200 dekadických číslic) [9]. Šifra RSA je dodnes bezpečnou a široce používanou šifrou, ale zda bude bezpečná i v budoucnu nelze říci s jistotou.

4 ZÁVĚR

Látka potřebná k popsání RSA je nad rámec středoškolského učiva. Bylo proto potřeba popsat základy kryptografie a modulární aritmetiky zcela od základů. Jak bylo v této práci dokázáno, RSA je asymetrický kryptosystém s univerzálním použitím včetně bezpečné výměny zpráv, šifrovacích klíčů i kryptografických podpisů. RSA je jednou z nejjednodušších široce používaných šifer, i tak byl v této práci prostor pouze pro její základy.

Hlavní dva soubory implementace jsou přiloženy jako příloha, ale celý Git repositář lze najít na adrese https://github.com/Franatrtur/rsa_demo, výsledný NPM archív pak na adrese https://www.npmjs.com/package/@franatrtur/rsa_demo. Oba repositáře jsou volně dostupné k použití, hodnocení a veřejným pull request žádostem. Omezující je ovšem samotný jazyk Javascript, který limituje čísla na délku 32 bitů. JavaScript má datový primitiv pro větší čísla, ale implementace by byla složitější. Tato implementace není určena k bezpečnostnímu použití, je pouze demonstrativní.

Jako podnět pro další práce by mohla posloužit implementace plného algoritmu pracujícího s čísly o délkách srovnatelných s komerčním použitím, např. 2048 bitů. S tím také úzce souvisí algoritmy pro hledání velkých prvočísel (Miller-Rabinův algoritmus, Fermatův test, atd.).

5 POUŽITÁ LITERATURA

[1] AL HASIB, Abdullah a Abul Ahsan Md. Mahmudul HAQUE. *A Comparative Study of the Performance and Security Issues of AES and RSA Cryptography* [online]. Finland: Helsinki University of Technology, 2008, s. 505-510 [cit. 2023-01-04]. Dostupné z: https://www.academia.edu/12439308/A_Comparative_Study_of_the_Performance_and_Security_Issues_of_AES_and_RSA_Cryptography

[2] DIFFIE, Whitfield a Martin E. HELLMAN. *New Directions in Cryptography*. 1976.

[3] JIRÁSEK, Petr, Luděk NOVÁK a Josef POŽÁR. *Výkladový slovník kybernetické bezpečnosti: Cyber security glossary*. 2., aktualiz. vyd. Praha: Policejní akademie ČR v Praze, 2013. ISBN 978-80-7251-397-0.

[4] KEJÍKOVÁ, Petra. *Šifrování a teorie čísel* [online]. Brno, 2016 [cit. 2023-01-24]. Dostupné z: https://is.muni.cz/th/oae8s/. Bakalářská práce. Masarykova univerzita, Pedagogická fakulta. Vedoucí práce Karel LEPKA.

[5] KLEINOVÁ, Hana. *Algoritmus RSA a problém faktorizace v praxi* [online]. Brno, 2008 [cit. 2023-01-04]. Dostupné z: https://is.muni.cz/th/uctoh/. Diplomová práce. Masarykova univerzita v Brně Fakulta informatiky. Vedoucí práce Doc. RNDr. Václav Matyáš, M.Sc., Ph.D.

[6] KOLLER, Alexandr. *Implementace historických kryptosystémů* [online]. Brno, 2006 [cit. 2023-01-24]. Dostupné z: https://is.muni.cz/th/zfxea/. Bakalářská práce. Masarykova univerzita, Fakulta informatiky. Vedoucí práce Josef ŠPROJCAR.

[7] MENEZES, Alfred J., Paul C.van OORSCHOT a Scott A. VANSTONE. *Handbook of applied cryptography*. Boca Raton: CRC Press, 1997. ISBN 08-493-8523-7.

[8] MILANOV, Evgeny. *The RSA Algorithm* [online]. 2009. Dostupné také z: https://sites.math.washington.edu/~morrow/336_09/papers/Yevgeny.pdf

[9] OULEHLA, Milan a Roman JAŠEK. *Moderní kryptografie*. [Praha]: IFP Publishing, 2017. ISBN 978-80-87383-67-4.

[10] RIVEST, Ron, L. ADLEMAN a Leonard SHAMIR. *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*. 1977.

[11] SHIREY, R. *Internet Security Glossary: Version 2* [online]. In: . 2007 [cit. 2023-01-04]. Dostupné z: https://datatracker.ietf.org/doc/html/rfc4949

[12] SKALA, Martin. *Zabezpečení paketového zrcadla s využitím asymetrické kryptografie* [online]. Brno, 2010 [cit. 2023-01-04]. Dostupné z: https://is.muni.cz/th/ixz4i/. Bakalářská práce. Masarykova univerzita, Fakulta informatiky. Vedoucí práce Doc. RNDr. Eva Hladká, Ph.D.

[13] ŠENOVSKÝ, Pavel. VŠB – Technická univerzita Ostrava, Fakulta bezpečnostního inženýrství. *Bezpečnostní informatika 1* [online]. Skripta. 8. vydání. Ostrava, 2017 [cit. 2023-01-04]. Dostupné z: https://fbiweb.vsb.cz/~sen76/data/uploads/skripta/bi1_8ed_fin.pdf

5.1 Použité obrázky

1. Vlastní tvorba autora: *Znázornění analogie Alice a Boba*

2. CEPHEUS. *Caesar3.svg: Caesar cipher with a shift of 3*. In: Wikimedia Commons [online]. 2006 [cit. 2023-01-22]. Dostupné z: https://commons.wikimedia.org/wiki/File:Caesar3.svg

3. Vlastní tvorba autora: *Obrázek analogie Alice a Boba s použitím asymetrické šifry*

4. *Purzen_Clock_face_web.png: Vektorový obrázek nástěnné hodiny s čísly. Černobílý obrázek hodin bez ruky*. In: Publicdomainvectors [online]. 2015 [cit. 2023-01-17]. Dostupné z: https://publicdomainvectors.org/cs/volnych-vektoru/Vektorov%C3%A9-grafiky-n%C3%A1st%C4%9Bnn%C3%A9-hodiny-s-%C4%8D%C3%ADsly/12539.html

6 PŘÍLOHA: MATEMATICKÝ DŮKAZ RSA

Cílem je dokázat integritu $D\big(E(M)\big) = M$. Definice pro E a D jsou:

$$E\colon\ m^e \bmod n$$
$$D\colon\ m^d \bmod n$$

Je tedy třeba dokázat, že $m^{e\cdot d} \equiv m\ (mod\ n)$

Definice: Pro celé číslo n, Eulerova funkce $\varphi(n)$ vrací počet celých čísel z $\{1, \dots, n\}$ nesoudělných s n.

Eulerova věta: Pro $a, n \in Z$ kde $nsd(a, n) = 1$ platí:

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

To jsou potřebné nástroje k důkazu za využití vlastností modulární aritmetiky:

1) Začneme Eulerovou větou pro nesoudělné n a zprávu $m \in Z_n$, dále libovolné $k \in Z$:

$$m^{\varphi(n)} \equiv 1\ (mod\ n)$$
$$m^{k\cdot\varphi(n)} \equiv 1^k \equiv 1\ (mod\ n)$$
$$m^{k\cdot\varphi(n)+1} \equiv m\ (mod\ n)$$

2) Tato kongruence je už velmi podobná tomu, co potřebujeme dokázat. Je jen třeba upravit exponent. Jak víme, e a d jsou navzájem modulárně multiplikativně inverzní dle $\varphi(n)$:

$$e \cdot d \equiv 1\ \big(mod\ \varphi(n)\big)$$
$$\Rightarrow\quad e \cdot d = 1 + k \cdot \varphi(n)$$

3) Zjišťujeme, že se rovnají. To platí, protože k v 1) je **každé** k, zatímco v 2) je to **nějaké** k. Získáváme finální kongruenci:

$$m^{e\cdot d} \equiv m\ (mod\ n)$$

Důkaz je kompletní.

7 PŘÍLOHA: SOUBORY IMPLEMENTACE RSA

Celý repositář se všemi soubory, které jsou součástí této implementace, lze najít na adrese https://github.com/Franatrtur/rsa_demo. Zde jsou přiloženy jen základní soubory, bez testů a metadat.

7.1 Soubor rsa-en.ts

```
let PRIMES: Array<number> = [2] //small PRIMES (less than 2^8) - constant

//array initialization
outer: for(let n = 3; n < 256; n += 2){

	for(let factor = 1; factor < PRIMES.length; factor++)
		if(n % PRIMES[factor] === 0)
			continue outer

	PRIMES.push(n)
}

export const Primes = PRIMES.slice()

function random_prime(): number{

	// only select from the bigger half of primes
	return PRIMES[PRIMES.length / 2 + Math.floor(Math.random() * (PRIMES.length
/ 2))]
}

//code design from: https://www.geeksforgeeks.org/multiplicative-inverse-under-
modulo-m/#tablist3-tab7
function modular_inversion(num: number, modulus: number): number{ //num != 1

	let mod0 = modulus
	let y = 0, x = 1
	let quot, temp

	while(num > 1){

		quot = Math.floor(num / modulus)
		temp = modulus

		//as euclids algorigm
		modulus = num % modulus
		num = temp
		temp = y

		y = x - quot * y
		x = temp
	}

	return x < 0 ? x + mod0 : x
}


function modular_power(base: number, exponent: number, modulus: number): number {

	let result = 1

	while(exponent){

		if((exponent & 1) == 1)
			result = (result * base) % modulus
```

```
		exponent >>= 1 //next bit
		base = (base ** 2) % modulus
	}

	return result
}


export class RSAKey {

	private exponent: number
	public mod: number

	constructor(exponent: number, mod: number){

		this.exponent = exponent
		this.mod = mod
	}

	process(message: number): number { //encrypt = decrypt

		return modular_power(message, this.exponent, this.mod)
	}

}


export class RSAKeyPair {

	public publicKey: RSAKey
	public privateKey: RSAKey

	static generate(prime1: number = 0, prime2: number = 0, exponent: number =
7){

		let e = exponent

		if(prime1 % e == 1 || prime2 % e == 1) //assert modular invertibility
under phi(n)
			throw new Error("Exponent must be coprime with totient of n")

		while(!prime1 || prime1 % e == 1)
			prime1 = random_prime()

		while(!prime2 || prime2 % e == 1 || prime2 == prime1) // prevent
factorization to n = p*p
			prime2 = random_prime()

		let p = prime1,
			q = prime2

		let n = p * q
		let totient = (p - 1) * (q - 1)
		let d = modular_inversion(e, totient)

		return new this(new RSAKey(e, n), new RSAKey(d, n))
	}
```

```
	constructor(publicKey: RSAKey, privateKey: RSAKey){

		this.publicKey = publicKey
		this.privateKey = privateKey
	}

}
```

7.2 Soubor tsconfig.json

```
{

	"compilerOptions" : {

		"target": "ES6",

		"lib": ["ES6", "DOM", "ESNext"],

		"declaration": true,

		"removeComments": false,

		"watch": false

	}

}
```