SLOVENSKÁ TECHINCKÁ UNIVERZITA V BRATISLAVE

FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

ŠIFROVANIE

Škodlivý kód, jeho formy a identifikácia

PRINCÍPY INFORMAČNEJ BEZPEČNOSTI

Samuel Bubán

ID: 102879

30.3.2021

PRENOS INFORMÁCII

História

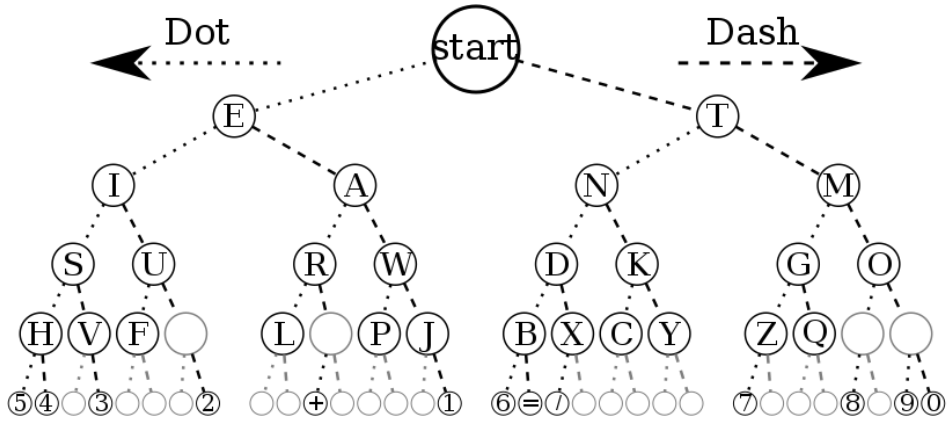
Ľudia mali už oddávna potrebu utajiť informácie, aby ich mohol čítať iba odosielateľ a prijímateľ. Šifrovanie slúži viacerým účelom, ale hlavne sa využívalo a stále využíva v armáde, na zašifrovanie nadchádzajúcich útokov, postupov ...

V dnešnej dobe sa šifrovanie nevyužíva iba na prenos armádnych alebo vysoko úradných informácii, ale aj na prenos osobných údajov, ich ukladanie, a ochrana pred ukradnutím. Týmto sa zabezpečuje, že k údajom má prístup iba odosielateľ a prijímateľ, a teda počas prenosu ich nemôže niekto ukradnúť.

Kódovanie

Kódovanie sa od šifrovania líši tým, že kľúč na odkódovanie správy je verejne známy. To znamená, že hocikto, kto sa k danej informácii dostane ju vie rozlúštiť, pretože si môže vyhľadať, akým spôsobom bola zakódovaná.

Príklad kódovania je Morseova abeceda, ASCII tabuľka, alebo aj ukladanie súborov do .rar archívov. Pre všetky tieto spôsoby je spoločné, že sa dajú bez problémov prekonvertovať na bežný text (s využitím znalostí daného spôsobu kódovania).



Obrázok 1 Morse code tree

Šifrovanie

Pri šifrovaní kľúč na rozlúštenie správy nie je verejne známy (nemal by byť). Preto informáciu dokáže získať len odosielateľ (niekedy ani ten) a prijímateľ.

Existujú dve skupiny šifier. Symetrické a asymetrické. Symetrické šifry používajú rovnaký kľúč na zašifrovanie aj na odšifrovanie správy. Asymetrické šifry používajú jeden kľúč na

zašifrovanie, a druhý, ktorý je známy iba prijímateľovi je použitý na odšifrovanie. Tieto dva kľúče spolu súvisia, pretože sú matematicky prepojené, ale súkromný kľúč (ten, ktorý použije prijímateľ správy na odšifrovanie) sa nedá vypočítať z verejného kľúča.

SYMETRICKÉ ŠIFROVANIE

Caesarova šifra

Asi celosvetovo najznámejší spôsob šifrovania je Caesarova šifra. Jedná sa o najjednoduchšiu šifru, ktorá sa dá aj jednoducho prelomiť. Jedná sa o symetrické šifrovanie, preto má len jeden kľúč. Caesarova šifra funguje na princípe, že sa písmená abecedy zapíšu za sebou do radu, a následne sa znaky posunú o 3 doľava (D -> A). Takto sa prejde postupne cez celý text, a výsledok vyzerá ako úplne iný jazyk.

```
Toto je ukážkový text, ktorý bol zašifrovaný pomocou Caesarovej šifry. Jediný rozdiel
od pôvodnej šifry je, že namiesto znakov abecedy boli použité znaky ASCII, takže
sa dajú zašifrovať aj znaky slovenskej abecedy - mäkčene, dĺžne ...
```

Obrázok 2 Obyčajný text

```
QlqlgbrhÀžÂ»hlsÀºqbuq)hqloÀº_liw^Âžfcols^kÀºmljl`lr@^bp^olsbgÂžfcov+GbafkÀºolwafbi
lamÀ±slakbgÂžfcovgb)Â»bk^jfbpqlwk^hls^_b`bav_lifmlrÂ»fqÀ¦wk^hv>P@FF)q^hÂ»b
p^a^gÀ·w^Âžfcols^Â¢^gwk^hvpilsbkphbg^_b`bav*jÀ¡hÁŠbkb)aÁ·Â»kb+++
```

Obrázok 3 Zašifrovaný text cez ASCII

```
Qlql gb rhxwhlsv qbuq, hqlov yli wxpfcolsxkv mljlzlr Zxbpxolsbg pfcov. Gbafkv olwafbi
la mlslakbg pfcov gb, wb kxjfbpql wkxhls xybzbav ylif mlrwfqb wkxhv XPZFF, qxhwb
px axgr wxpfcolsxq xg wkxhv pilsbkphbg xybzbav - jxhzbkb, aiwkb ...
```

Obrázok 4 Zašifrovaný text cez originál Caesarovu šifru

Tu sú tri obrázky, ako príklad Caesarovej šifry. Prvý obrázok je len ukážkový text. Druhý je zašifrovaný pomocou programu, ktorý som vytvoril. Tento program len posunie aktuálny znak o niekoľko symbolov dopredu (-3 pre Caesarovu šifru) na základe ASCII tabuľky. To znamená, že aj medzery sú zašifrované. Tretí obrázok je výsledok po použití pôvodnej Caesarovej šifry, ktorá posúvala iba samotné znaky (takže slová mali stále rovnaký počet písmen ako nezašifrovaný text).

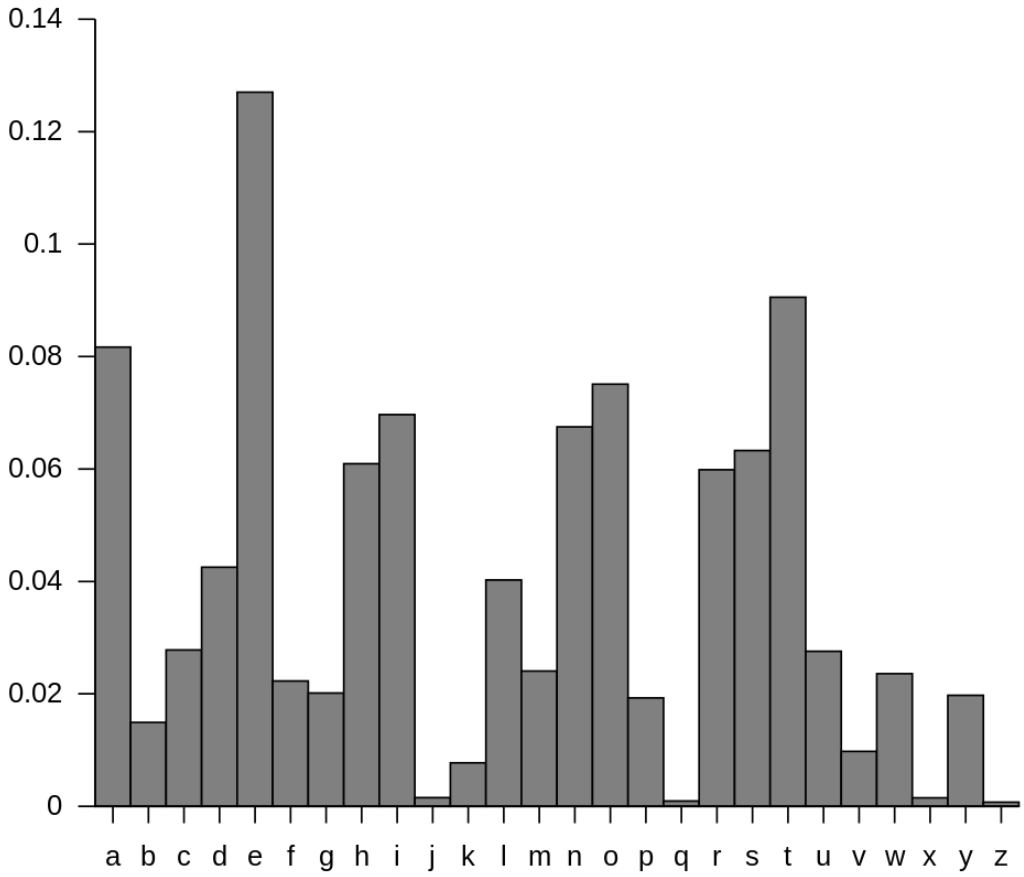
Problém Caesarovej šifry ale je, že rovnaký znak bude v danom texte vždy prepísaný ako iný, ale stále rovnaký symbol (D bude stále A). Preto je možné Caesarovu šifru prelomiť pomocou

frekvenčnej analýzy. Prvýkrát použitá bola v deviatom storočí (prvýkrát o čom sa zachoval záznam).

Druhý spôsob, ako sa dá Caesarova šifra prelomiť je brute force. Spravil som tiež program na otestovanie, ktorý vytvorí 256 kópii textu s rôznym posunutím. Program dokáže výsledok vytvoriť okamžite, takže Caesarova šifra nie je vôbec bezpečná (jeden zo súborov obsahuje pôvodný text), a to dokonca aj bez znalosti kľúča.

Frekvenčná analýza

Každý jazyk má špecifické využívanie niektorých znakov. Napríklad v slovenčine sa častejšie používajú samohlásky ako spoluhlásky, v angličtine takisto. Preto ak útočník pozná, v akom jazyku bol pôvodný text písaný, a text je dostatočne dlhý, vie spraviť pomerne dobrý odhad výsledného textu. Najskôr urobí histogram (všetky znaky a ich výskyt v texte), následne ich skúsi namapovať na originálny text podľa početnosti.



Obrázok 5 Rozloženie znakov v Angličtine

Vigenère cipher

Vigenere-ova šifra na rozdiel od Caesarovej využíva ako kľúč nejaké slovo. Tým sa zabezpečí, že raz bude znak zapísaný nejak, a neskôr inak. Tento kľúč sa posúva s každým znakom, a keď už je použitý posledný znak z kľúča, ide sa od začiatku.

Pri tejto šifre je možné využiť aj viacnásobné zašifrovanie s viacerými kľúčmi. Pri Caesarovej šifre by z toho bol len výsledok rovnaký ako po jednom zašifrovaní s iným kľúčom. Táto šifra je aj odolná voči frekvenčnej analýze, nakoľko sa na zakódovanie jedného písmena môžu použiť rôzne znaky vzhľadom na to, kde v texte sa tento znak nachádza.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Obrázok 6 Vigenere-ova tabuľka

ASYMETRICKÉ ŠIFROVANIE

Na rozdiel od symetrického šifrovania využíva asymetrické až dva kľúče. Jeden je verejne známy, a používa sa na zašifrovanie súborov, a druhý je súkromný, pomocou ktorého sa údaje odšifrujú.

RSA algoritmus

Jeden z možných algoritmov na šifrovanie údajov pomocou viacerých kľúčov je RSA algoritmus. Tento algoritmus využíva prvočísla na určenie public a private kľúča. Vstupný text sa šifruje pomocou dvoch funkcii – mocnina a modulo. Vstupný text sa umocní na dané číslo (verejne známe) a na výsledok sa použije funkcia modulo, ktorá vráti zvyšok po delení druhým, tiež verejne známym číslom n. Takto vznikne zašifrovaný text, ktorý sa môže poslať druhej strane.

Na odšifrovanie sa použije druhý kľúč, a rovnaký postup, teda text sa umocní na hodnotu privátneho kľúča, a následne sa znovu použije modulo funkcia (tej istej verejnej hodnoty n). Výsledkom je odšifrovaná správa.

Výpočet e, d, n

RSA algoritmus používa tieto tri premenné: e – encryption, hodnota, na ktorú je umocnený pôvodný text. d – decryption, hodnota, na ktorú je umocnený zašifrovaný text. n – hodnota, ktorá sa používa na modulo.

Na začiatok sa zvolia dve prvočísla (na profesionálne šifrovanie sa používajú veľké čísla, ktoré môžu zaberať až 1024 bitov každé, čiže maximálna hodnota je 2 ^ 1024). Tieto dve prvočísla sa vynásobia, a získa sa tak hodnota n. Keďže obe čísla majú maximálnu veľkosť 2 ^ 1024, tak výsledok ich súčinu je číslo s maximálnou hodnotou 2 ^ 2048. Niekto by si ale mohol povedať, že nakoľko je n produkt dvoch prvočísel, ktoré sa používajú na vygenerovanie kľúčov (aj súkromného aj verejného), tak stačí získať tieto prvočísla a môžeme získať aj súkromný kľúč, a teda môžeme dešifrovať správu.

Problém ale je, že pri veľkých číslach je jednoduché získať násobok dvoch čísel, ale naspäť to už nie je jednoduché, a trvalo by to neskutočne dlho prísť na to, aké dve čísla sú korene pre generovanie kľúčov. Práve kvôli tomuto je šifra bezpečná.

Na ukážku, maximálna hodnota 2 ^ 1024:

179769313486231590772930519078902473361797697894230657273430081157732675805500963132708477322407536021120113879871393357658789768814416622492847430639474124377767893424865485276302219601246094119453082952085005768838150682342462881473913110540827237163350510684586298239947245938479716304835356329624224137216

Maximálna hodnota 2 ^ 2048:

32317006071311007300714876688669951960444102669715484032130345427524655138867890893197201411522913463688717960921898019494119559150490921095088152386448283120630877367300996091750197750389652106796057638384067568276792218642619756161838094338476170470581645852036305042887575891541065808607552399123930385521914333389668342420684974786564569494856176035326322058077805659331026192708460314150258592864177116725943603718461857357598351152301645904403697613233287231227125684710820209725157101726931323469678542580656697935045997268352998638215525166389437335543602135433229604645318478604952148193555853611059596230656

Myslím, že teraz je už asi jasnejšie, prečo je tento spôsob šifrovania považovaný za bezpečný. Vyššie uvedené čísla sa používajú štandardne, ale je možné použiť aj väčšie hodnoty (prvočísla 2 ^ 2048 a výsledok 2 ^ 4096).

Od týchto dvoch prvočísel sa ďalej odpočíta jednotka, a vynásobia sa navzájom. $(p - 1)(q - 1)$ Toto je výsledok funkcie $\Phi(\mathrm{n})$ a používa sa neskôr pri výpočte kľúčov. Táto hodnota tiež určuje, koľko existuje nesúdeliteľných čísel s hodnotou n.

Teraz treba vybrať číslo e. Podmienky sú, že musí byť väčšie ako 1 a menšie ako $\Phi(\mathrm{n})$, a tiež musí byť nesúdeliteľné s číslami n a $\Phi(\mathrm{n})$. Táto hodnota je väčšinou nejaké malé prvočíslo (3, 5, 7, 11 ...).

Posledné čo treba získať je číslo d. Podmienka, ktorá musí pre toto číslo platiť je:

$$e * d \; mod \; \Phi(\mathrm{n}) = 1$$

Takýto vzorec ale nie je jednoduché vypočítať postupným dosadzovaním (viac ako miliardy a miliardy možností), preto sa použije Rozšírený Euklidov algoritmus. Presný postup ako vypočítať hodnotu d sa nachádza v zdroji 3, kde je tento algoritmus výborne a podrobnejšie vyjadrený a vysvetlený.

RÝCHLOSŤ ALGORITMOV

Caesarova šifra

Nakoľko výpočet výsledných hodnôt pre symetrické šifrovanie je jednoduchý, aj čas na výpočet je veľmi nízky. Zašifrovanie a odšifrovanie prebehne v zlomku sekundy, teda môžeme povedať aj okamžite.

RSA algoritmus

Tento algoritmus pracuje s obrovskými číslami. To tiež znamená, že môže vyžadovať väčšie množstvo pamäte, ak nie je správne optimalizovaný. Python ale ponúka pri výpočte mocnín aj možnosť zadať číslo, ktoré sa má použiť ako modulo na výsledok. Toto urýchli výpočet až o niekoľko rádov (aj 1000x – bez optimalizácie som výsledok nedostal ani za dlhší čas).

Ja som svoju implementáciu spravil v programovacom jazyku Python. Na výpočet kľúčov som použil dve 9-ciferné prvočísla (takže ani zďaleka nie tak veľké, ako sa používajú štandardne). Tieto prvočísla som sám vypočítal v druhom programe pomocou Erathostenovho sita. Vypočítal som všetky prvočísla menšie ako 1 000 000 000. Celý výpočet trval menej ako 10 minút, a využil asi 20 GB pamäte RAM. Výsledný súbor, ktorý obsahuje všetky prvočísla menšie ako jedna miliarda má celkovú veľkosť 0.5 GB, preto ho nemôžem priložiť.

Z týchto dvoch ručne zvolených prvočísel som pomocou druhého programu vypočítal hodnoty e, n, d. Výpočet bol urýchlený Rozšíreným Euklidovým algoritmom, a výsledné hodnoty som dostal okamžite.

Na zašifrovanie nejakého súboru ho môj program najprv načíta celý ako binárne čísla, a spracuje ho po 4 bajtoch. Tým sa zabezpečí, že frekvenčná analýza nemôže prelomiť túto šifru. Zašifrovanie väčšieho súboru (konkrétne súboru s prvočíslami, ktorý má 0.5 GB) trvalo 5 minút, a výsledný súbor mal dvojnásobnú veľkosť.

Odšifrovanie tohto súboru trvalo 35 minút, nakoľko pri odšifrovaní sa pracuje s väčšou mocninou. Na príklad prikladám hodnoty, ktoré som použil.

e = 3 d = 666665802666906259 n = 999998706000358093

RANSOMWARE

Program

Nakoľko sa v tomto projekte venujem hlavne ransomware-u, rozhodol som sa, že svoj program rozšírim tak, že po spustení sa zašifrujú všetky súbory obsiahnuté v danom priečinku. Na vygenerovanie kľúčov z prvočísel slúži **RSAKeyGenerator.py**, na zašifrovanie slúži **RSAEncrypt.py** a na odšifrovanie slúži **RSADecrypt.py**.

Tento program (ktorý teraz obsahuje verejný kľúč e, a hodnotu n) by mohol byť použitý na zašifrovanie údajov niekoho iného (po tom, ako by sa urobil export na napríklad .exe aplikáciu). Po zaplatení výkupného by útočník poslal druhý program na odblokovanie zašifrovaných súborov (v ktorom sa nachádza privátny kľúč).

Jediné, na čo by si útočník musel dať pozor je, aby pre každý cieľ použil nanovo vygenerovaný pár kľúčov, pretože ináč by stačilo, aby jedna z obetí zaplatila výkupné, získala tak privátny kľúč, a ostatní by si mohli údaje odblokovať tiež.

ZÁVER

Bezpečnosť používania

V dnešnej dobe je nutné na zašifrovanie textu používať pomerne veľké kľúče (čím sa zabezpečí, že v blízkej ani dlhšej dobe nebudú prelomené). Problém ale môže prísť s nástupom kvantových počítačov, pretože tie by mali byť na prelomenie šifier ako stvorené. Bežné počítače ale takýmto bonusom nedisponujú, preto je zatiaľ bezpečné posielať svoje súkromné údaje ak sú zašifrované pomocou asymetrickej šifry.

ZDROJE

1. Eddie Woo: The RSA Encryption Algorithm 1 4.11.2014
   https://www.youtube.com/watch?v=4zahvcJ9glg

2. Eddie Woo: The RSA Encryption Algorithm 2 4.11.2014
   https://www.youtube.com/watch?v=oOcTVTpUsPQ

3. Anthony Vance: How the RSA algorithm works 14.10.2014
   https://www.youtube.com/watch?v=Z8M2BTscoD4

4. Caesar cipher
   https://www.dcode.fr/caesar-cipher

5. Frequency analysis Wikipedia
   https://en.wikipedia.org/wiki/Frequency_analysis