Úkol 4a - Brute-force a slovníkový útok na hesla

V archivu se zadáním máte soubor **`password_database.csv.`** Na každém řádku je kombinace uživatelského jména a hesla ve formě otisku SHA-256. Předpokládejte, že nebylo použito mechanismu "solení". Uživatelská jména byla nahrazena vašimi osobními čísly. Úkolem je provést jednoduchý slovníkový útok a útok hrubou silou na tuto "ukradenou databázi" hesel.

Vaše implementace by měla obsahovat následující kroky:

1) Načtěte výše uvedený soubor a vyfiltrujte pouze záznamy s vaším osobním číslem
2) Na všechny z nich použijte nejprve slovníkový útok; zkuste využít informace o nejčastějších heslech, které lidé používají https://en.wikipedia.org/wiki/Wikipedia:10,000_most_common_passwords
3) Na zbývající hesla použijte útok hrubou silou; zkoušejte generovat různé kombinace alfanumerických znaků a vytvářet z nich otisky SHA-256 a porovnávat je s otisky v souboru.
4) Během lámání si měřte statistiky:
    a) Čas běhu programu na prolomení;
    b) Počet vygenerovaných kombinací hesel než je nalezena shoda; pokud je úspěšný slovníkový útok počet vygenerovaných kombinací je roven počtu hesel, které program musel projít v rámci slovníku.

Hash SHA-256 není třeba implementovat, využijte knihovny. Zadání máte zjednodušené. Počítejte s tím, že hesla obsahují pouze **malá písmena nebo jenom čísla, případně kombinace malých písmenek a čísel.** Velká písmena ani žádné jiné speciální znaky neuvažujte. Maximální délka hesla je **6 znaků** ⇒ tzn. delší sekvence není potřeba procházet.

Výsledkem běhu programu bude soubor CSV **`cracked_results_<os_cislo>.csv`**, kde **`<os_cislo>`** nahradíte vaším osobním číslem. Do CSV souboru vložte následující 4 údaje oddělené středníkem:
`<SHA-256hash>;<heslo>;<počet_kombinací_k_nalezení>;<čas_běhu_v_sekundách>`

Např. takto:
`9f56e761d79bfdb34304a012586cb04d16b435ef6130091a97702e559260a2f2;pass;6787196;12.09`
…
…
…

Záhlaví sloupečků nedělejte.

Pro všechny případy zajistěte, aby váš program běžel (včetně pomocných výpisů) maximálně 3 minuty pro každý "ukradený hash" (v nejhorším možném případě váš program tedy poběží 12 minut). Dávejte pozor na pomocné výpisy, které mohou čas běhu značně prodloužit. Nicméně je možné/vhodné např. logovat do souboru každých *n* vteřin, kolik různých kombinací program vygeneroval a dosavadní čas, případně počet variant, které program doposavad vyzkoušel.

Pokud se vám z nějakého důvodu nepodaří některé z hesel prolomit (např. neefektivní implementace – program běží déle než 3 minuty na jeden hash), do výsledného CSV souboru vložte k odpovídajícímu hashi řetězec **`"not_cracked".`**

Všechna hesla jsou volena tak, aby byl "rozumný" čas k jejich prolomení. 100% bodů získáte pouze za prolomení všech 4 hesel do 12 minut.

Zamyslete se nad bezpečností hesel a jejich způsobu ukládání. Proč jsou důležité takové zásady bezpečnosti jako **solení** a hlavně **minimální požadavky na délku a sílu hesla?**

Úkol 4b - Brute-force útok na Diffie-Hellman protokol

V archivu se zadáním máte soubor **`diffie_hellman_keys.csv.`** Tento soubor obsahuje následující čísla oddělená středníkem:

- délka parametru ***p*** v bitech;
- délka vygenerovaného (neznámého) privátního klíče v bitech;
- parametr ***p***
- parametr ***g***
- číslo $g^x \bmod p$, kde ***x*** je neznámý privátní klíč Alice

Soubor simuluje situaci, kdy si Alice zvolila privátní klíč *x* (veřejné parametry *p* a *g*) a poté poslala číslo $g^x \bmod p$ Bobovi (viz protokol Diffie-Hellman – Diffie-Hellman key exchange). Po této výměně by následovalo vytvoření sdíleného klíče *k* pro komunikaci.

Vaším úkolem je protokol hrubou silou "napadnout" a zjistit všechny 4 privátní klíče Alice, tedy čísla, kterými je umocňován generátor *g* v rámci modulo *p*.

Výsledkem běhu programu bude CSV soubor **`alices_private_keys_<os_cislo>.csv`** se 4 řádky a dvěma hodnotami odděleným středníkem: uhodnutý privátní klíč a čas v sekundách na jeho prolomení, tedy např. takto:

`24654;0.077`
`61555;0.219`
`3785468;20.334`
`84489545;559.306`

Zatímco privátní klíče pro první 3 řádky v soubory odhalíte velice rychle, hrubá síla pro poslední řádek by mohla trvat několik jednotek minut. Nehledě na to, omezte čas běhu programu na 15 minut. Pokud nezvládnete v této době prolomit a nalézt všechny 4 klíče nezískáte za tuto úlohu plný počet bodů. Opět pozor na pomocné výpisy, mohou zpomalovat běh programu.

Cílem je si experimentálně ověřit, že použití slabých klíčů a nedostatečně velkých čísel vede k bezpečnostnímu riziku. Uvědomte si zranitelnost kryptografických algoritmů při použití

nedostatečně dlouhého/silného klíče. Uvědomte si dále, že s rostoucí délkou klíčů neroste čas hrubé síly lineárně.

V rámci nepovinného cvičení si vyzkoušejte sami vygenerovat řádově mnohem větší čísla a vyzkoušejte na ně podobný útok.

Odevzdání

Do odevzdávacího archivu zabalte:

- váše spustitelné programy včetně zdrojových kódů
- oba soubory CSV z archivu se zadáním (**`password_database.csv`** a **`diffie_hellman_keys.csv)`**
- výsledné CSV soubory se statistikami

Doporučená literatura

Introduction to Modern Cryptography
http://staff.ustc.edu.cn/~mfy/moderncrypto/reading%20materials/Introduction_to_Modern_Cryptography.pdf
str. 309 sekce 9.3 Diffie-Hellman Key Exchange