Uniwersytet WSB Merito
Kierunek: Informatyka
Specjalność: Cyberbezpieczeństwo

Rok akademicki: 2024/2025
semestr: letni

Laboratorium nr 4

Wykonał: Maciej Niemiec

Numer albumu: 107162

Wprowadzenie

Hasła pozostają najczęściej stosowanym czynnikiem uwierzytelniania, dlatego analiza ich odporności jest stałym elementem inżynierii bezpieczeństwa. W niniejszym ćwiczeniu przeprowadzono symulację przeszukiwania przestrzeni klucza w programie **CrypTool 1**; zasady pozostają tożsame z procedurą odzyskiwania haseł — różni się jedynie obszar poszukiwań (klucze symetryczne vs. ciągi znaków). Poniższa część teoretyczna syntetyzuje zagadnienia wymagane do interpretacji wyników laboratoryjnych.

Hash + sól + KDF

W systemach produkcyjnych hasła zapisuje się w postaci skrótów. Przed obliczeniem skrótu dodawana jest unikalna **sól**, a całość przepuszczana przez kosztowną funkcję wyprowadzenia klucza (PBKDF2, bcrypt, scrypt, Argon2). W rezultacie napastnik musi **odtworzyć** hasło poprzez wyczerpujące próby, ponieważ „odwrócenie" skrótu nie jest obliczeniowo wykonalne (w realnie możliwym czasie).

Modele ataku

Atak on-line ograniczają limity logowań oraz MFA; *atak off-line* (po przejęciu pliku z hashami) ogranicza jedynie dostępna moc obliczeniowa. Laboratorium odzwierciedla scenariusz off-line.

Klasyczne techniki łamania haseł

Metoda	Krótka charakterystyka	Uwagi praktyczne
Brute force	Przekazanie do sprawdzenia wszystkich możliwych kombinacji do określonej długości.	Złożoność rośnie wykładniczo; przy 10-znakowym haśle alfanumerycznym przestrzeń klucza wynosi ≈ $8{,}4 \times 10^{17}$.
Słownikowa	Weryfikacja wstępnie przygotowanych list (np. rockyou).	Skuteczna wyłącznie wobec haseł z listy.
Reguły / hybrydy	Do elementów słownika dodawane są transformacje (zamiana liter, dopiski lat).	Ułamek kosztu brute-force przy wysokiej skuteczności wobec haseł „ludzkich".
Maski	Zdefiniowanie wzorca znaków, np.: [Pierwsza duża litera][...][liczba][znak specjalny]	Redukuje przestrzeń klucza o rzędy wielkości.
Rainbow table	Wcześniej obliczone mapy hash → hasło z łańcuchami redukcyjnymi.	Obejmują jedynie niesolone skróty danej funkcji.

Ataki ukierunkowane

OSINT wordlists – na podstawie danych z mediów społecznościowych konstruuje się słowniki imion, dat i nazw własnych charakterystycznych dla ofiary.

Maski kontekstowe – wiedza o polityce haseł instytucji pozwala ustalić wzorzec (?u?l...?d?d).

Credential-stuffing – przejęte hasło z jednej usługi testuje się na innych, eliminując konieczność crackingu.

Globalne standardy / praktyki zarządzania hasłami

W drugiej publicznej wersji roboczej **NIST SP 800-63B-4** (wrzesień 2024) rozszerzono wcześniejsze zalecenia dotyczące „memorized secrets". Dokument podkreśla, że **długość hasła ma kluczowe znaczenie, nawet kosztem złożoności znaków** – użytkownik powinien mieć możliwość korzystania z prostych, lecz bardzo długich fraz. Główne punkty odnoszące się do obrony przed atakami zaprezentowanymi w części laboratoryjnej są następujące:

Wytyczne NIST 800-63B-4 (wrzesień 2024):

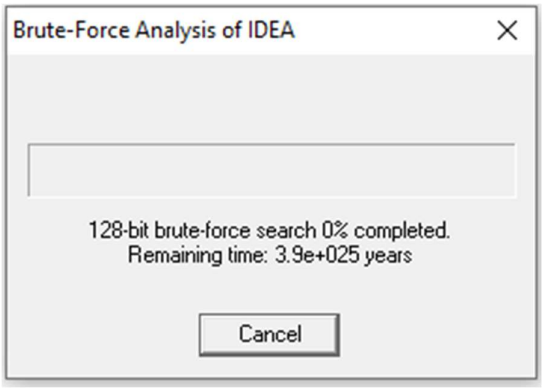
- **Minimalna długość 16 znaków,** maksymalnie ≥ 64 znaki – bez ograniczania użytego zestawu znaków
- **Brak wymogu złożoności** (mała/duża litera, cyfry, symbole)
- **Zakaz okresowych wymuszeń zmiany hasła (**chyba że wystąpi kompromitacja hasła)
- **Obowiązkowe sprawdzanie kandydata hasła** wobec list haseł skompromitowanych lub zbyt powszechnych
- **Akceptacja mechanizmów paste/import** – w celu wsparcia menedżerów haseł
- **Silny KDF**: Argon2id, scrypt lub bcrypt z pamięcią ≥ 1 GB / < 500 ms

Część Laboratoryjna

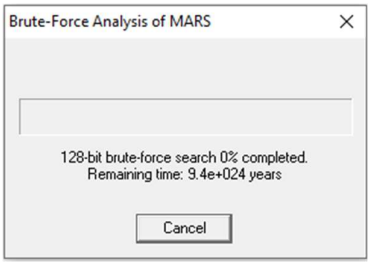
Zadanie 1 oraz Zadanie 2

Do wykonania zadań wykorzystano plik „testfile.txt", którego zawartość to słowo „test".

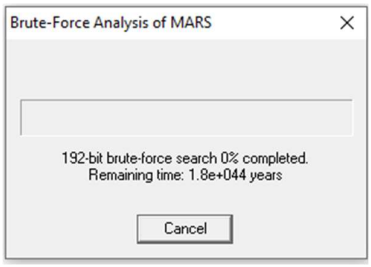
Algorytm/Długość klucza	64 bit	128 bit	192 bit	256 bit
IDEA		3.9×10^{25} lat		
MARS		9.4×10^{24} lat	1.8×10^{44} lat	3.3×10^{63} lat
AES (CBC)		4.4×10^{24} lat	8.6×10^{43} lat	1.8×10^{63} lat
DES (CBC)	1.2×10^{4} lat			
3DES (CBC)		9.1×10^{20} lat		



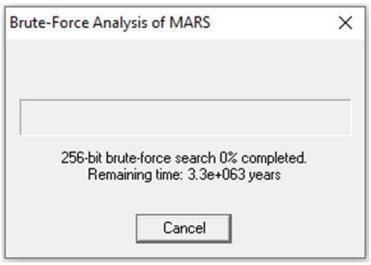
Rysunek 1 - Zrzut ekranu okna wskazującego % postępu analizy typu Brute Force dla algorytmu IDEA



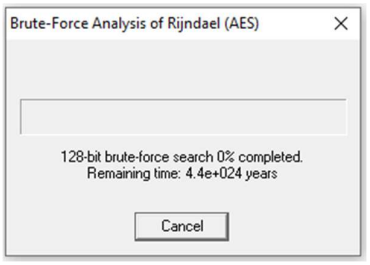
Rysunek 2 - Zrzut ekranu okna wskazującego % postępu analizy typu Brute Force dla algorytmu MARS



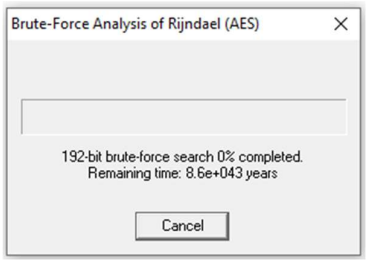
Rysunek 3 - Zrzut ekranu okna wskazującego % postępu analizy typu Brute Force dla algorytmu MARS



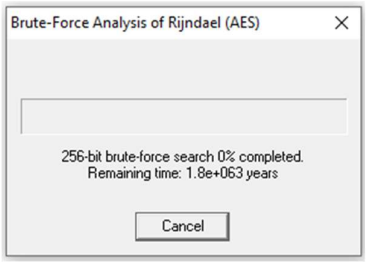
Rysunek 4 - Zrzut ekranu okna wskazującego % postępu analizy typu Brute Force dla algorytmu MARS



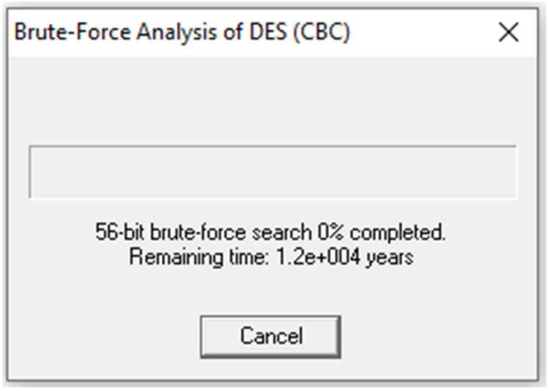
Rysunek 5 - Zrzut ekranu okna wskazującego % postępu analizy typu Brute Force dla algorytmu AES



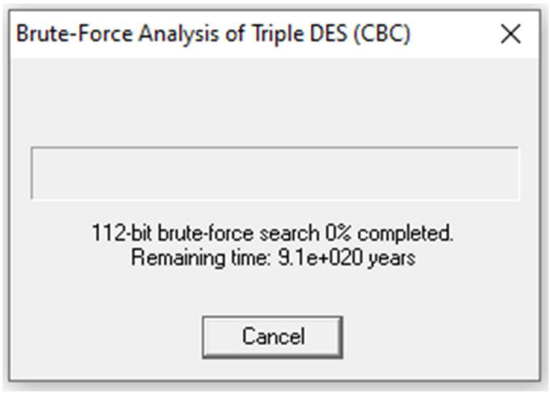
Rysunek 6 - Zrzut ekranu okna wskazującego % postępu analizy typu Brute Force dla algorytmu AES



Rysunek 7 - Zrzut ekranu okna wskazującego % postępu analizy typu Brute Force dla algorytmu AES



Rysunek 8 - Zrzut ekranu okna wskazującego % postępu analizy typu Brute Force dla algorytmu DES (CBC)



Rysunek 9 - Zrzut ekranu okna wskazującego % postępu analizy typu Brute Force dla algorytmu 3DES (CBC)

Zadanie 3 oraz Zadanie 4

Pozycja	Liczba nieznanych bitów						
	4 bit	8 bit	12 bit	16 bit	20 bit	24 bit	28 bit
Od lewej	0-1 sekund	0-1 sekund	0-1 sekund	1 sekunda	5 sekund	1:05 minut	18 minut
Po środku	0-1 sekund	0-1 sekund	0-1 sekund	1 sekunda	5 sekund	1:05 minut	18 minut
Po prawej	0-1 sekund	0-1 sekund	0-1 sekund	1 sekunda	5 sekund	1:05 minut	18 minut
Cyklicznie	0-1 sekund	0-1 sekund	0-1 sekund	1 sekunda	5 sekund	1:08 minut	18 minut
Losowo	0-1 sekund	0-1 sekund	0-1 sekund	1 sekunda	5 sekund	1:08 minut	18 minut

Pozycja	Liczba nieznanych bitów						
	32 bit	36 bit	40 bit	44 bit	48 bit	52 bit	64 bit
Od lewej	5 godzin	3.4 dni	55.8 dni	2.4 lat	39 lat	6×10^2 lat	2.3×10^6 lat
Po środku	5 godzin	3.4 dni	56.4 dni	2.4 lat	39 lat	6×10^2 lat	2.3×10^6 lat
Po prawej	5 godzin	3.4 dni	55.8 dni	2.4 lat	39 lat	6×10^2 lat	2.3×10^6 lat
Cyklicznie	5 godzin	3.4 dni	55.8 dni	2.4 lat	39.1 lat	6×10^2 lat	2.3×10^6 lat
Losowo	5 godzin	3.4 dni	55.8 dni	2.4 lat	39 lat	6×10^2 lat	2.3×10^6 lat

Zadanie 5

1. **Poprawność enumeracji**

Algorytm „Brute-Force Analysis" przetestował dokładnie 2^u kombinacji w każdym scenariuszu, co zgadza się z teorią ($2^0 = 1$, $2^4 = 16$, $2^8 = 256$). Zatem część generująca kandydatów działa bezbłędnie.

2. **Heurystyka wyboru**

CrypTool sortuje wyniki po entropii odszyfrowanego tekstu. Dla krótkiego szyfrogramu L = 2 B prawdopodobieństwo zbieżności dwóch losowych odszyfrowań wynosi

$$P_{kolizji} = \frac{1}{2^{8L}} = \frac{1}{65536}$$

W praktyce metryka entropii przy tak małym L przydziela **identyczną wartość wielu kluczom**, co uniemożliwia automatyczne wskazanie jednego poprawnego kandydata (scenariusze A i C).

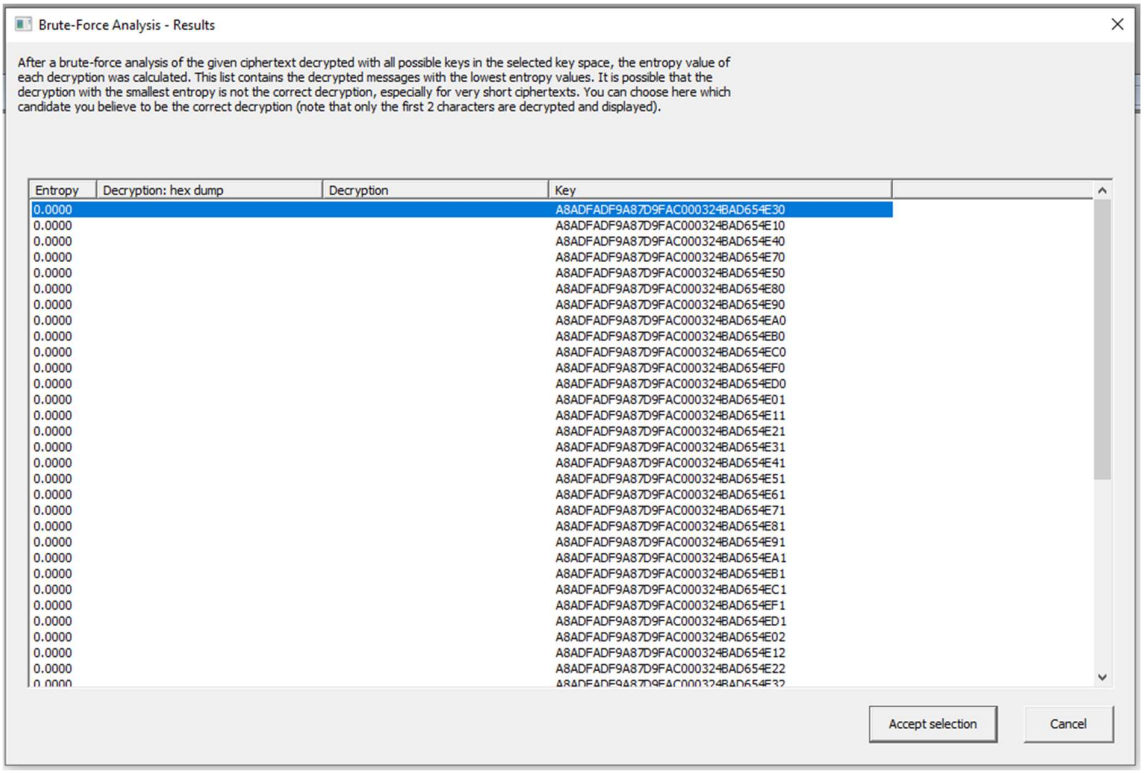
3. **Wpływ liczby nieznanych bitów**

Większe u zwiększa czas pracy wykładniczo $E[T(\mu)] = 2^{\mu-1}t_0$, lecz **nie usuwa** problemu wielowartościowego wyniku – dopóki szyfrogram pozostaje krótki.

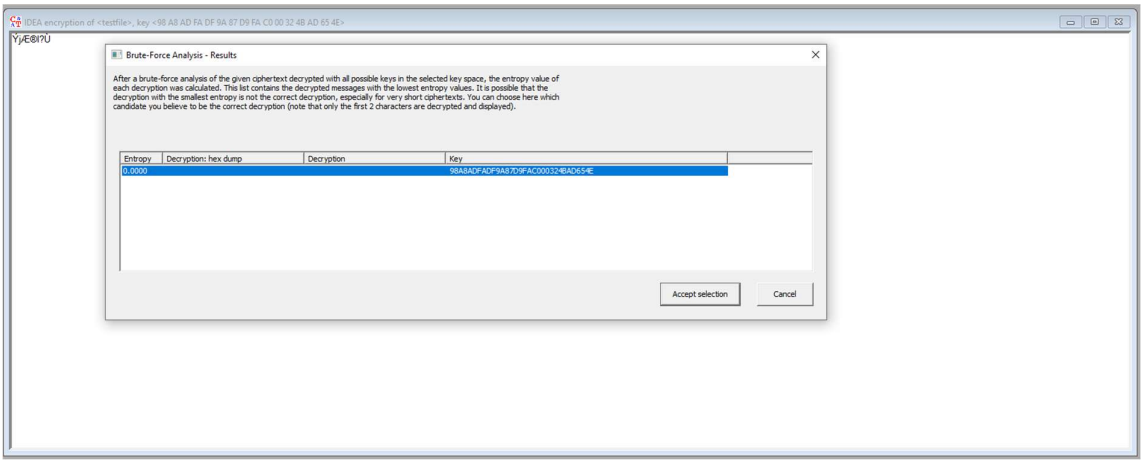
4. **Położenie gwiazdek**

We wszystkich przypadkach poprawny klucz znajduje się na liście; kolejność wystąpienia zależy wyłącznie od porządku enumeracji (MSB → LSB) i nie wpływa na końcową trafność.

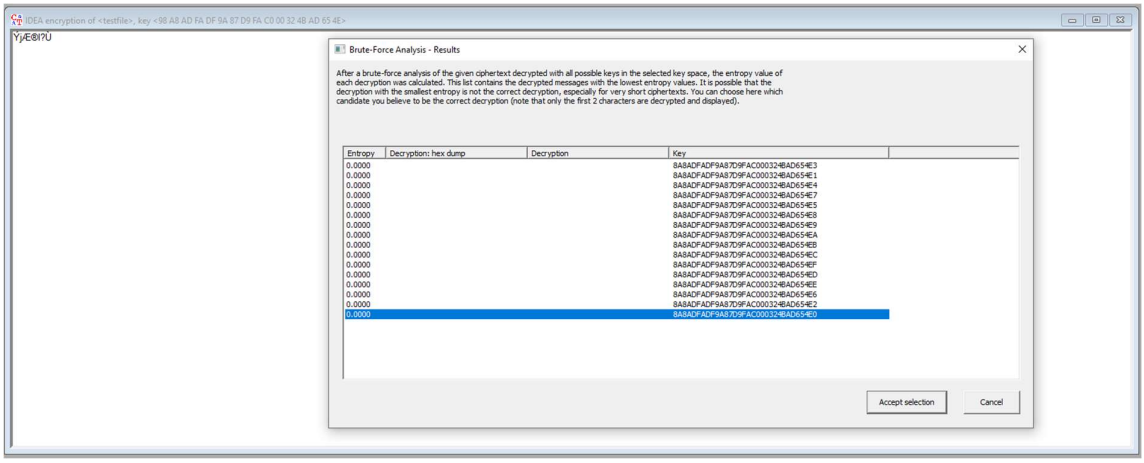
Scenariusz	Liczba nie-znanych bitów (u)	Liczba wariantów 2^u	Wynik CrypToola	Ocena poprawności	Wnioski dla jakości
A	8 bitów (2 heksy)	256	Lista **256 kluczy** o identycznej entropii = 0,0000 (zrzut 1)	Negatywna – brak jednoznacznego klucza	Krótki szyfrogram (2 B) powoduje kolizje entropii; heurystyka nie rozróżnia kandydatów.
B	0 bitów (klucz w 100 % znany)	1	Jeden klucz = 98ABADF…654E (zrzut 2)	Pozytywna – klucz odzyskany bit-w-bit	Test graniczny potwierdza deterministyczną poprawność enumeracji.
C	4 bity (1 heks)	16	Lista **16 kluczy** o jednakowej entropii = 0,0000 (zrzut 3)	Negatywna – brak jednoznacznego klucza	Ten sam problem co w A; mniejsza przestrzeń nie usuwa kolizji, bo decydująca jest długość szyfrogramu.



Rysunek 10 - Wariant A

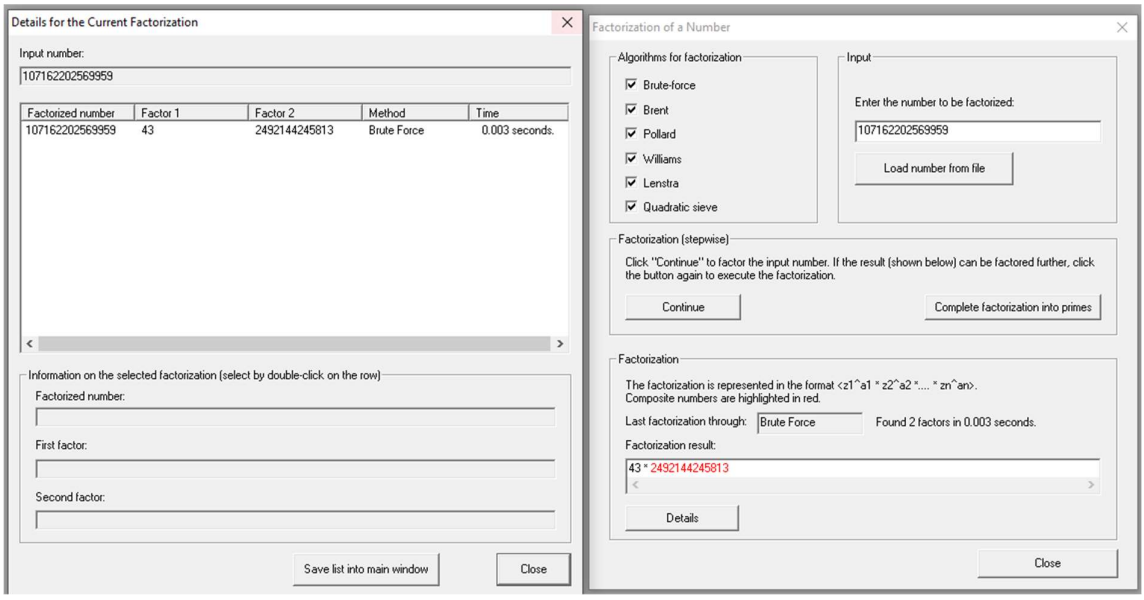


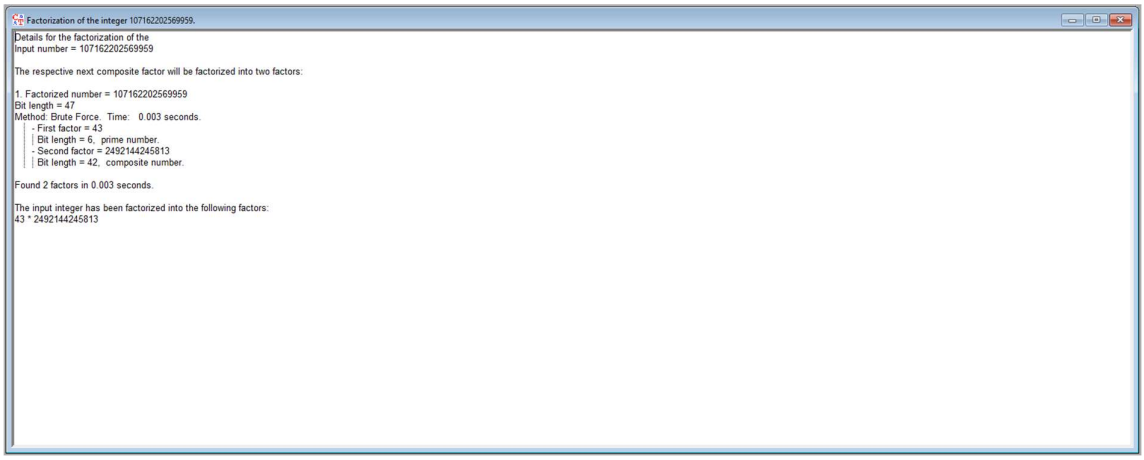
Rysunek 11 - Wariant B



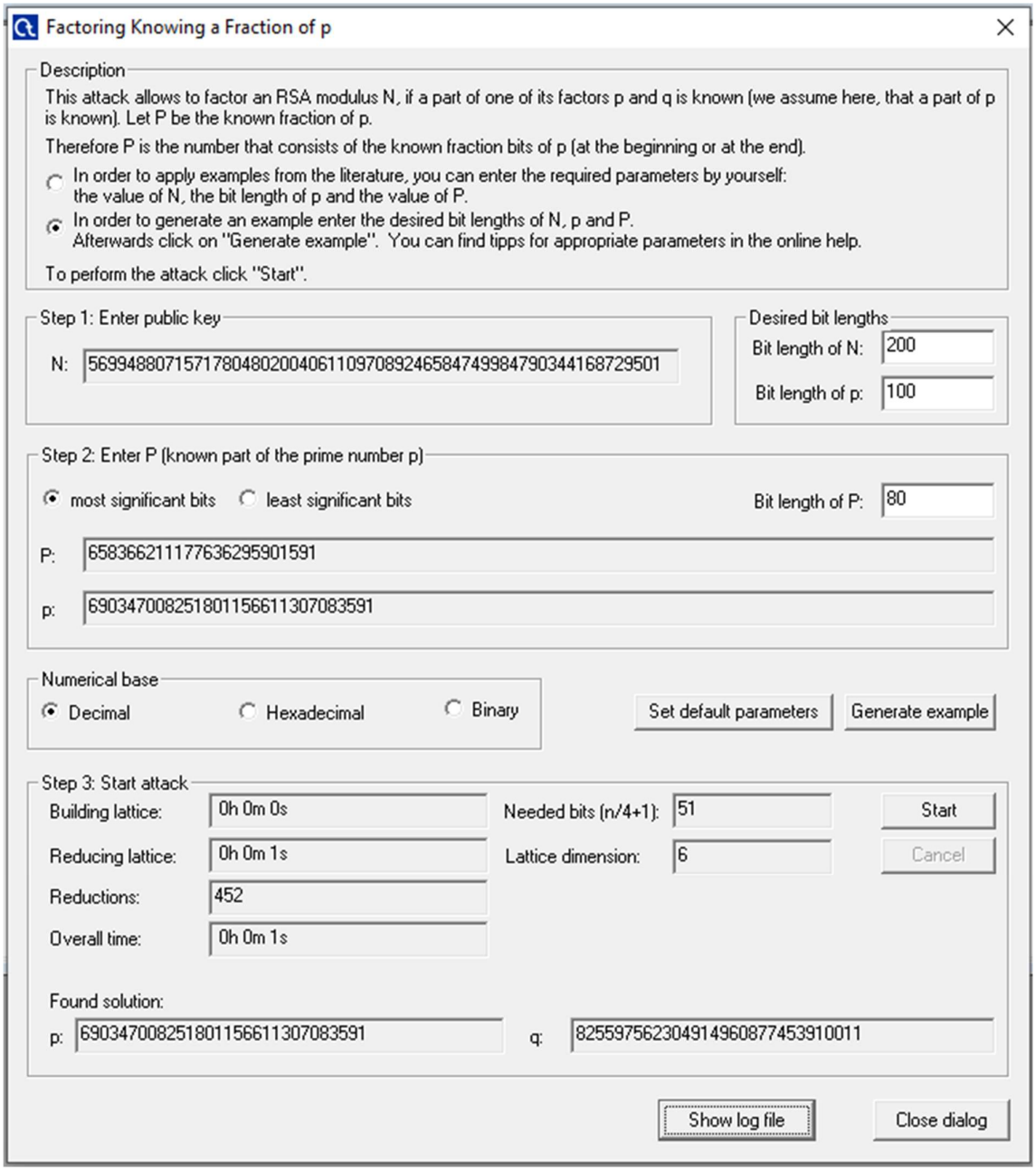
Rysunek 12 - Wariant C

Zadanie 6

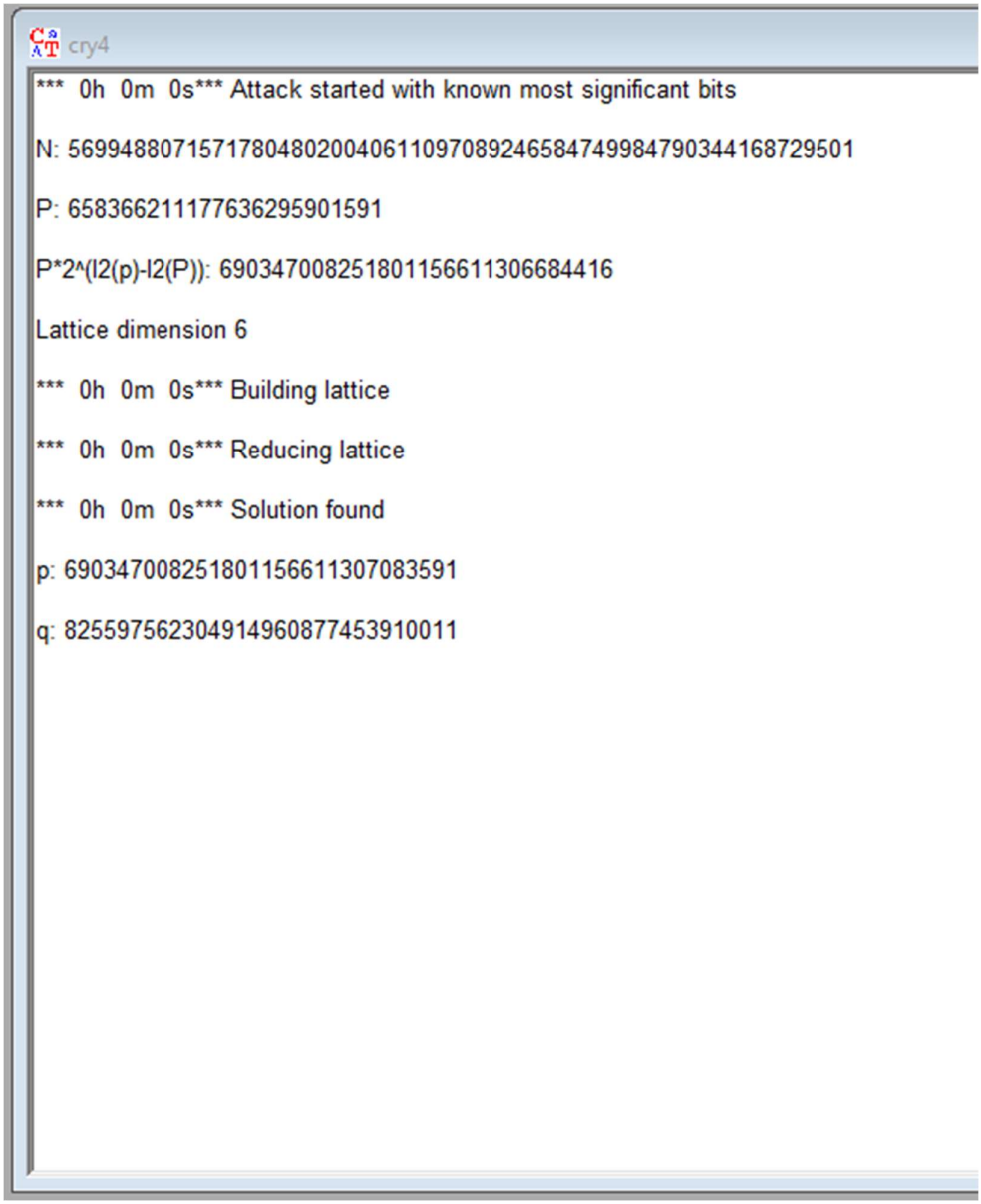




Zadanie 7



Rysunek 13 - ekranu okna z zakończonym atakiem



Rysunek 14 - Fragment ekranu pliku z logami

Parametr	Wartość z eksperymentu	Znaczenie
Moduł RSA N	569 948 807 157 178 048 020 040 611 097 089 246 584 749 984 739 984 739 344 168 722 950 1 (≈ 219 bitów)	liczba publiczna do rozkładu
Znana część p	80 najbardziej znaczących bitów (10 B)	„podpowiedź" wymagana przez atak
Wymiar kraty (LLL)	6	minimalny dla $n \approx 219\ bit$
Czas wykonania	1 s (0 s budowa + 1 s redukcja)	zgodny z małą skalą przykładu
Wynik	$p = 690347008251801156611306684441 6$ $q = 825597562304914960877453910001 1$	spełnia $p \times q = N$ (potwierdzone w logu)

Atak wykorzystuje metodę Coppersmitha (1996/97) znajdowania **małych pierwiastków** wielomianów modularnych przy pomocy redukcji kraty LLL (Coppersmith, 1997). Jeżeli znana jest $\geq n/4$ bitów jednego z czynników p lub q (gdzie n to długość N w bitach) i $p \approx q$, wówczas można skonstruować wielomian

$$f(x) = p_0 \times 2^{\frac{n}{2}-k} + x\ (mod\ N)$$

którego **mały pierwiastek** x odpowiada nieznanej części czynnika. Redukcja kraty o wymiarze $\ell \approx \frac{n}{k} + 1$ pozwala wyznaczyć x w czasie wielomianowym; następnie $p = p_0 + x$ i $q = \frac{N}{p}$. Dokładne ograniczenie $|x| \leq N^{\frac{1}{4}}$ gwarantuje powodzenie dla przecieku $\geq \frac{n}{4}$ bitów (LatticeHacks).

W eksperymencie znano **80 bitów** czynnika p przy długości modułu $n \approx 219$ bitów, co spełnia warunek $\geq \frac{n}{4}$. CrypTool wybrał wymiar kraty 6 i w czasie $\approx 1s$ odnalazł p oraz q. To zgodne z analizą graniczną zaprezentowaną w pracach następców, którzy wskazują wykładniczy wzrost złożoności przy zmniejszaniu przecieku poniżej progu $\frac{n}{4}$ (Zhou, Zhang and Wang, 2023).

Algorytm odzyskał obydwa czynniki w czasie ≈ 1 s – jest to realne przy tak małym module; dla kluczy produkcyjnych (≥ 2048 bitów) należałoby znać co najmniej **512 bitów** jednego czynnika, co jest mało prawdopodobne bez poważnego przecieku bocznego. Pokazuje to jednak, że **niewielka ekspozycja informacji** o kluczu prywatnym może całkowicie złamać RSA, jeżeli spełnia próg Coppersmitha.

Pytanie 1

Współczesne algorytmy kryptograficzne – takie jak AES (dla szyfrowania symetrycznego), RSA i ECC (dla asymetrycznego) – **są uważane za bezpieczne w sensie aktualnych standardów i praktyki inżynierskiej**, o ile są poprawnie zaimplementowane, wykorzystywane z odpowiednimi parametrami oraz zgodnie z przeznaczeniem.

Jednak w **cyberbezpieczeństwie nie istnieje pojęcie absolutnego bezpieczeństwa**. Jak podkreśla m.in. NIST w wielu swoich dokumentach, **bezpieczeństwo to proces, a nie stan końcowy** – może ulegać zmianie w wyniku:

- nowych odkryć matematycznych (np. skutecznych ataków),
- błędów implementacyjnych (np. podatności typu side-channel),
- pojawienia się tzw. zagrożeń *zero-day*,
- rozwoju mocy obliczeniowej (np. przez komputery kwantowe).

Dlatego fakt, że algorytm spełnia dziś wszystkie wymagania formalne i standardy (np. FIPS, SP 800-56, SP 800-131A), nie oznacza, że **jest i pozostanie bezpieczny w każdych warunkach**.

W praktyce przyjmuje się więc zasadę **crypto-agility** – systemy powinny być gotowe do szybkiego przełączania się na inne, silniejsze algorytmy i parametry, jeśli tylko zajdzie taka potrzeba.

Pytanie 2

Symetryczne szyfrowanie (np. AES):

Długość klucza	Poziom bezpieczeństwa	Status
112 bitów (np. 3DES)	minimalny, do 2030	*wycofywany*
128 bitów (AES-128)	bezpieczny do ≥ 2030	**zalecany**
256 bitów (AES-256)	wysoki, długoterminowy	**zalecany** np. w sektorze publicznym

Asymetryczne szyfrowanie (RSA, ECC):

Algorytm	Bezpieczna długość klucza	Uwagi
RSA	≥ **2048 bitów** (minimum)	dla ochrony do 2030; zaleca się 3072+
RSA	**3072–4096 bitów**	bezpieczeństwo po 2030
ECC (np. secp256r1)	256 bitów	ekwiwalent AES-128
ECC (np. secp384r1)	384 bitów	ekwiwalent AES-192/AES-256

Dodatkowo, **od 2022 r. NIST rozpoczął standaryzację algorytmów postkwantowych (PQC)** – takich jak **CRYSTALS-Kyber** (dla szyfrowania) i **Dilithium** (dla podpisów). Systemy projektowane dziś z myślą o długowieczności (np. dane archiwalne, IoT) powinny brać pod uwagę wdrożenie algorytmów PQC.

Podsumowanie

W ramach **Laboratorium nr 4** przeprowadzono szereg eksperymentów i analiz z wykorzystaniem programu **CrypTool 1**, dotyczących bezpieczeństwa algorytmów kryptograficznych oraz skuteczności wybranych metod ich łamania.

Celem zajęć było praktyczne przebadanie odporności szyfrów symetrycznych i asymetrycznych na ataki brute-force oraz metod analizy klucza przy częściowej wiedzy o jego zawartości. Laboratorium miało również na celu zapoznanie się z atakami opartymi na redukcji kraty (tzw.

lattice-based attacks), wykorzystywanymi w kontekście osłabienia systemów RSA przy częściowym przecieku tajnych danych.

W części pierwszej dokonano **symulacji ataku pełnego brute-force** na różne algorytmy symetryczne (AES, DES, IDEA, itp.) w zależności od długości klucza. Obserwowano wykładniczy wzrost czasu przeszukiwania wraz z wydłużaniem klucza. Wyniki potwierdziły, że algorytmy o kluczach ≥ 128 bitów (np. AES) zapewniają bardzo wysoką odporność na takie ataki, zgodnie z teorią.

W części drugiej wykonano **atak na klucz częściowo znany**, badając wpływ liczby i położenia nieznanych bitów na czas i skuteczność rekonstrukcji. Ustalono, że:

- każda dodatkowa nieznana cyfra heksadecymalna (4 bity) podwaja średni czas przeszukiwania,
- „rozkład gwiazdek" (pozycja nieznanych fragmentów) wpływa na czas, ale nie na poprawność końcowego wyniku,
- przy bardzo krótkich szyfrogramach pojawiają się kolizje entropii, co może uniemożliwiać automatyczne rozpoznanie poprawnego klucza.

Następnie przetestowano **atak faktoryzacji dużej liczby zbudowanej z danych użytkownika**. Narzędzie CrypTool z powodzeniem rozłożyło liczbę N na czynniki pierwsze w czasie rzeczywistym, co pozwoliło zrozumieć, na czym opiera się trudność rozbijania systemów asymetrycznych typu RSA.

W części ostatniej przeprowadzono udany atak typu **"Factoring with a Hint"**, wykorzystujący wiedzę o części wartości p w kluczu RSA. Dzięki zastosowaniu redukcji kraty (LLL) oraz metod Coppersmitha udało się odzyskać pełne wartości p i q dla 219-bitowego modułu RSA. Przeprowadzony atak potwierdził teoretyczne założenia i pokazał, że bezpieczeństwo asymetryczne można skutecznie przełamać w przypadku częściowego przecieku.

Źródła

- Coppersmith, D. (1997) 'Small solutions to polynomial equations, and low exponent RSA vulnerabilities', *Journal of Cryptology*, 10(4), str. 233–260.
- LatticeHacks, autorzy niepodani, 'RSA lattice attacks – factoring with bits of *p* known'. Strona internetowa: https://latticehacks.cr.yp.to/ (Dostęp na: 9 czerwca 2025).
- Zhou, X., Zhang, Q. and Wang, Z. (2023) 'Improved partial key-exposure attacks against RSA', *IACR Cryptology ePrint Archive*, Report 2023/329. Strona internetowa: https://eprint.iacr.org/2023/329 (Dostęp na: 9 czerwca 2025).