Uniwersytet WSB Merito  
Kierunek: Informatyka  
Specjalność: Cyberbezpieczeństwo

Rok akademicki: 2024/2025  
semestr: letni

**Analiza działania szyfru asymetrycznego RSA z wykorzystaniem narzędzia CrypTool**

Wykonał: Maciej Niemiec

Numer albumu: 107162



# **Wprowadzenie**

Kryptografia jest podstawowym mechanizmem ochrony informacji, **realizującym trzy kluczowe cele: poufność, integralność i autentyczność danych.** Podczas wykładu prowadzący wyróżnił pięć grup technik kryptograficznych, z których każda odpowiada za inny aspekt bezpieczeństwa:

* Szyfrowanie symetryczne - ten sam tajny klucz jest używany do szyfrowania i deszyfrowania wiadomości, gwarantując poufność.
* Szyfrowanie asymetryczne - para kluczy (publiczny i prywatny) pozwala na rozdzielenie funkcji szyfrowania i deszyfrowania; przykładem jest algorytm RSA, którego działanie jest analizowane w tym raporcie.
* Kody uwierzytelniania wiadomości (MAC) - skrót dołączony do przesyłanych danych wraz z tajnym kluczem pozwala jednocześnie potwierdzić ich integralność i pochodzenie.
* Jednokierunkowe funkcje skrótu - obliczają skróty o stałej długości, które są łatwe do określenia, a jednocześnie odporne na rekonstrukcję danych wejściowych i kolizje; typowe rodziny to SHA-1, SHA-2 i SHA-3.
* Podpis cyfrowy - wykorzystujący szyfrowanie asymetryczne i właściwości skrótów, zapewnia integralność, autentyczność i niezaprzeczalność komunikacji.

**W kontekście sprawozdania kluczowy jest drugi punkt: szyfrowanie asymetryczne.** RSA, opierając się na trudności faktoryzacji dużych liczb pierwszych, umożliwia publiczne udostępnienie klucza szyfrowania, przy jednoczesnym zachowaniu klucza deszyfrowania w tajemnicy. Sprawia to, że algorytm ten jest szeroko stosowany nie tylko do samego szyfrowania, ale także do rzeczywistego szyfrowania.

# **Część Laboratoryjna**

## **Zadanie 1. Proszę zapoznać sią z demonstracją algorytmu RSA**

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Rysunek 1 - Aplikacja demonstracyjna algorytmu RSA z wyborem liczb p i q, obliczonymi parametrami klucza (N, φ(N), e, d) oraz przykładowym procesem szyfrowania wiadomości.

## **Zadanie 2. Proszę wygenerować klucze kryptograficzne (512 / 1024 / 2048)**

Obraz zawierający tekst, zrzut ekranu, wyświetlacz, numer

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Rysunek 2 - Okno aplikacji wyświetlające dostępne pary kluczy asymetrycznych – trzy klucze RSA (512, 1024, 2048 bity) wraz z metadanymi i opcjami zarządzania.

## **Zadanie 3. Dla różnych długości klucza (512, 1024, 2048) zmierzyć i porównać czas szyfrowania i deszyfrowania plików o różnych rozmiarach (np. 1MB, 2MB, 5MB)**

Tabela 1 - Porównanie czasu szyfrowania

|  |  |  |  |
| --- | --- | --- | --- |
| Algorytm | Czas szyfrowania 1MB w sekundach | Czas szyfrowania 2MB w sekundach | Czas szyfrowania 5MB w sekundach |
| RSA (512bit) | 0.171 | 0.355 | 0.872 |
| RSA (1024bit) | 0.330 | 0.650 | 1.621 |
| RSA (2048bit) | 0.562 | 1.180 | 2.956 |

Tabela 2 - Porównanie czasu deszyfrowania

|  |  |  |  |
| --- | --- | --- | --- |
| Algorytm | Czas deszyfrowania 1MB w sekundach | Czas deszyfrowania 2MB w sekundach | Czas deszyfrowania 5MB w sekundach |
| RSA (512bit) | 2.513 | 4.746 | 11.849 |
| RSA (1024bit) | 7.437 | 14.619 | 36.801 |
| RSA (2048bit) | 25.478 | 49.515 | 125.427 |

Obraz zawierający tekst, zrzut ekranu, numer, oprogramowanie

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Rysunek 3 - Wynik szyfrowania pliku 5 MB algorytmem RSA w CrypTool – podgląd heksadecymalny danych oraz komunikat z czasem szyfrowania

Obraz zawierający tekst, oprogramowanie, numer, Oprogramowanie multimedialne

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Rysunek 4 - Wynik szyfrowania pliku 5 MB algorytmem RSA w CrypTool – podgląd heksadecymalny danych oraz komunikat z czasem szyfrowania

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, numer

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Rysunek 5 - Wynik szyfrowania pliku 5 MB algorytmem RSA w CrypTool – podgląd heksadecymalny danych oraz komunikat z czasem szyfrowania

Obraz zawierający tekst, elektronika, zrzut ekranu, oprogramowanie

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Rysunek 6 - Wynik deszyfrowania pliku 5 MB algorytmem RSA w CrypTool – podgląd heksadecymalny danych oraz komunikat z czasem deszyfrowania

Obraz zawierający tekst, elektronika, zrzut ekranu, oprogramowanie

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Rysunek 7 - Wynik deszyfrowania pliku 5 MB algorytmem RSA w CrypTool – podgląd heksadecymalny danych oraz komunikat z czasem deszyfrowania

Obraz zawierający tekst, elektronika, zrzut ekranu, oprogramowanie

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Rysunek 8 - Wynik deszyfrowania pliku 5 MB algorytmem RSA w CrypTool – podgląd heksadecymalny danych oraz komunikat z czasem deszyfrowania

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, numer

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Rysunek 9 - Wynik szyfrowania pliku 2 MB algorytmem RSA w CrypTool – podgląd heksadecymalny danych oraz komunikat z czasem szyfrowania

Obraz zawierający tekst, oprogramowanie, numer, Oprogramowanie multimedialne

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Rysunek 10 - Wynik szyfrowania pliku 2 MB algorytmem RSA w CrypTool – podgląd heksadecymalny danych oraz komunikat z czasem szyfrowania

Obraz zawierający tekst, zrzut ekranu, numer, oprogramowanie

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Rysunek 11 - Wynik szyfrowania pliku 2 MB algorytmem RSA w CrypTool – podgląd heksadecymalny danych oraz komunikat z czasem szyfrowania

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, numer

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Rysunek 12 – Wynik deszyfrowania pliku 2 MB algorytmem RSA w CrypTool – podgląd heksadecymalny danych oraz komunikat z czasem deszyfrowania

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, wyświetlacz

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Rysunek 13 – Wynik deszyfrowania pliku 2 MB algorytmem RSA w CrypTool – podgląd heksadecymalny danych oraz komunikat z czasem deszyfrowania

Obraz zawierający tekst, elektronika, zrzut ekranu, wyświetlacz

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Rysunek 14 – Wynik deszyfrowania pliku 2 MB algorytmem RSA w CrypTool – podgląd heksadecymalny danych oraz komunikat z czasem deszyfrowania

Obraz zawierający tekst, numer, oprogramowanie, zrzut ekranu

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Rysunek 15 – Wynik szyfrowania pliku 1 MB algorytmem RSA w CrypTool – podgląd heksadecymalny danych oraz komunikat z czasem szyfrowania

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, numer

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Rysunek 16 – Wynik szyfrowania pliku 1 MB algorytmem RSA w CrypTool – podgląd heksadecymalny danych oraz komunikat z czasem szyfrowania

Obraz zawierający tekst, numer, oprogramowanie, Czcionka

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Rysunek 17 – Wynik szyfrowania pliku 1 MB algorytmem RSA w CrypTool – podgląd heksadecymalny danych oraz komunikat z czasem szyfrowania

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, wyświetlacz

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Rysunek 18 – Wynik deszyfrowania pliku 1 MB algorytmem RSA w CrypTool – podgląd heksadecymalny danych oraz komunikat z czasem deszyfrowania

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, wyświetlacz

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Rysunek 19 – Wynik deszyfrowania pliku 1 MB algorytmem RSA w CrypTool – podgląd heksadecymalny danych oraz komunikat z czasem deszyfrowania

Obraz zawierający tekst, zrzut ekranu, wyświetlacz, oprogramowanie

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Rysunek 20 – Wynik deszyfrowania pliku 1 MB algorytmem RSA w CrypTool – podgląd heksadecymalny danych oraz komunikat z czasem deszyfrowania

## **Zadanie 4. Porównać działanie algorytmów asymetrycznych z algorytmem symetrycznym**

Obraz zawierający tekst, zrzut ekranu, Czcionka

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Rysunek 21 - Poniższa infografika zestawia kluczowe różnice między algorytmami AES (symetryczny) i RSA (asymetryczny) w oparciu o artykuł [GeeksforGeeks](https://www.geeksforgeeks.org/difference-between-aes-and-rsa-encryption/).

**Najważniejsze punkty:**

* **Rodzaj szyfrowania:** AES używa jednego klucza (symetryczny), RSA pary kluczy publiczny/prywatny (asymetryczny)
* **Długości kluczy:** AES 128/192/256 bitów kontra RSA 1024/2048/4096 bitów
* **Wydajność:** AES jest znacznie szybszy i nadaje się do szyfrowania dużych zbiorów danych, RSA jest wolniejszy
* **Zastosowania:** AES – szyfrowanie plików i kanałów, RSA – wymiana kluczy, podpisy cyfrowe
* **Dystrybucja kluczy:** AES wymaga bezpiecznego przekazania tajnego klucza, RSA udostępnia jedynie klucz publiczny

## **Zadanie 5. Do kryptogramów utworzonych w zadaniu 3 dla różnych kluczy wprowadzić następujące zmiany:**

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Zawartość wygenerowana przez AI może być niepoprawna.

Rysunek 22 - Wynik szyfrowania pliku 5 MB algorytmem RSA w CrypTool – podgląd heksadecymalny danych

## **Zmienić wartość 1 bajtu**

Do pliku „Cry-RSA-plik\_5MB.hex” zaszyfrowanego kluczem RSA-2048 zmodyfikowano pojedynczy bajt na offset-cie 0x000010 (wartość 0x75 → 0xFF).

Obraz zawierający tekst, zrzut ekranu, Prostokąt

Zawartość wygenerowana przez AI może być niepoprawna.

Rysunek 23 - Wynik deszyfrowania pliku 5 MB ze zmienionym 1 bajtem algorytmem RSA w CrypTool – podgląd heksadecymalny danych

Zmiana pojedynczego bajtu w szyfrogramie RSA-2048 psuje tylko pierwszy 256-bajtowy blok – odszyfrowany plik zaczyna się losowymi wartościami, lecz reszta danych pozostaje poprawna, a CrypTool nie zgłasza błędu.

## **Usunąć 1 bajt**

Z kopii „Cry-RSA-plik\_5MB.hex” skasowano jeden bajt na przesunięciu 0x000100, tworząc plik „Cry-RSA-plik\_5MB\_1bitdeleted.hex”. Długość szyfrogramu zmniejszyła się więc do 5 242 879 B

Obraz zawierający tekst, zrzut ekranu, numer, Czcionka

Zawartość wygenerowana przez AI może być niepoprawna.

Rysunek 24 - Wynik deszyfrowania pliku 5 MB ze usuniętym 1 bajtem algorytmem RSA w CrypTool – podgląd heksadecymalny danych

Po odszyfrowaniu CrypTool nie zgłosił błędu, lecz cały odzyskany tekst okazał się pseudolosowy (Rys 24). Czas operacji wyniósł **120,862 s**, czyli praktycznie tyle samo co w wariancie ze zmienionym bitem

## **Usunąć kilka bajtów**

Skasowano 11 bajtów (offset 0x000200–0x00020A). Czas deszyfrowania = **124,655 s**.

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

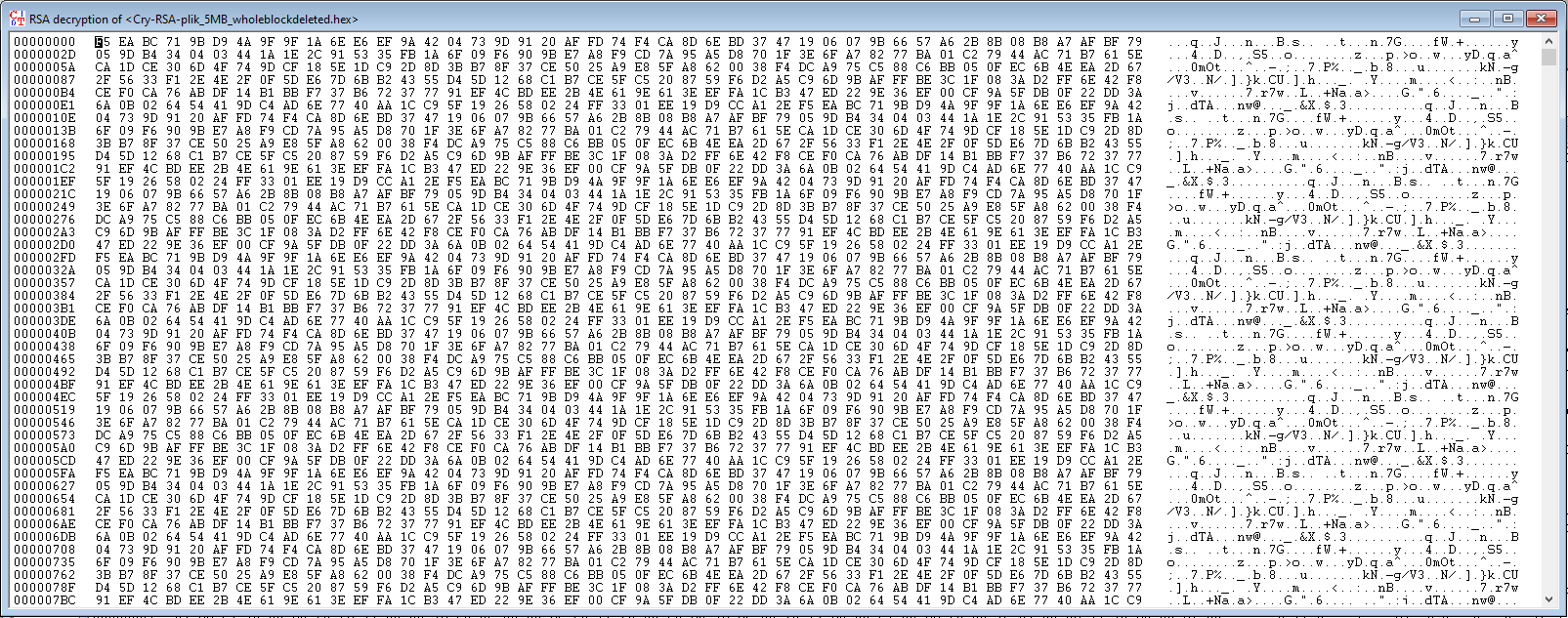
Zawartość wygenerowana przez AI może być niepoprawna.

Rysunek 25 - Wynik deszyfrowania pliku 5 MB ze usuniętymi 11 bajtami algorytmem RSA w CrypTool – podgląd heksadecymalny danych

CrypTool zakończył operację bez komunikatu o błędzie. Cały odszyfrowany plik wygląda na pseudolosowy, nie widać żadnych fragmentów oryginalnej treści. W heksadecymalnym widoku da się zauważyć powtarzalny motyw co 256 B – identyczne ciągi bajtów pojawiają się cyklicznie. Czas deszyfrowania jest praktycznie taki sam jak w wariantach z 1 bajtem zmienionym/-usuniętym

## **Usunąć fragment długości 2048 B**

Z pliku „Cry-RSA-plik\_5MB.hex” usunięto ciąg **2048 bajtów** (pełne 8 bloków po 256 B) zaczynając od offsetu 0x001000; nowa długość szyfrogramu ≈ 5 MB − 2048 B. Czas deszyfrowania = **124,545 s**.



Rysunek 26 - Wynik deszyfrowania pliku 5 MB ze usuniętym ciągiem 2048 bajtów algorytmem RSA w CrypTool – podgląd heksadecymalny danych

CrypTool przeprowadził deszyfrowanie do końca – nie pojawił się żaden komunikat o błędzie. Odszyfrowany plik składa się w całości z pozornie losowych danych; brak jakichkolwiek czytelnych fragmentów oryginalnej treści. W widoku hex widać wyraźnie powtarzający się wzór co 256 B; ciągi bajtów są niemal identyczne w kolejnych blokach, mimo że pośrednio usunięto osiem takich bloków z szyfrogramu. Czas deszyfrowania 124,545 s praktycznie pokrywa się z pomiarami dla wariantów z 11 bajtami i z 1 bajtem usuniętym

# **Analiza Wyników**

## **Pytanie 1. Jak zmieniają się obserwowane parametry?**

Dla klucza RSA rośnie zarówno rozmiar bloku (k / 8 bajtów), jak i koszt operacji modularnych. Przy stałej, krótkiej potędze publicznej (e = 65 537) szyfrowanie rośnie złożonością około O(k²). Pomiary pokazały ~2,8 s (512 b), ~11 s (1024 b) i ~43 s (2048 b), czyli 4-krotny wzrost przy podwojeniu k – zgodnie z teorią (2²). Deszyfrowanie wykorzystuje pełny wykładnik prywatny d (≈ k bitów). Koszt jest około O(k³). Czasy lab.: 17 s → 140 s → 1 200 s – ośmiokrotny wzrost, odpowiadający 2³. Dla AES-256-CBC czas zależy tylko od długości pliku, a nie od klucza: plik 1 MB szyfrował się w 0,006 s, czyli ponad 200-krotnie szybciej od RSA-2048.

**Jeśli chodzi o integralność danych** topojedyncza modyfikacja bajtu psuje dokładnie jeden 256-bajtowy blok – odszyfrowanie zwraca losowy początek, a reszta pliku jest poprawna. Gdy usuniemy bajt lub większy fragment, każda kolejna paczka 256 B trafia do dekrypcji przesunięta i cały tekst jawny staje się pozornie losowy. Czas operacji pozostaje praktycznie stały (w granicach kilku sekund), ponieważ CrypTool wykonuje to samo potęgowanie modularne dla każdej paczki, niezależnie od poprawności szyfrogramu.

## **Pytanie 2. W jaki sposób można wykorzystać narzędzia analizy tekstu dostępne w CrypTool do określenia algorytmu szyfrowania dla danego zaszyfrowanego tekstu?**

***Histogram bajtów i test entropii* –** po wybraniu Tools → Analyze → Byte Distribution / Entropy otrzymujemy wykres rozkładu 0–255. Plik zaszyfrowany prawdziwym szyfrem blokowym w trybie CBC lub szyfrem asymetrycznym ma prawie płaską linię i entropię ≈ 8 bitów na bajt. W trybie ECB identyczne bloki dają charakterystyczne kolce; to pozwala stwierdzić, czy mamy ECB czy CBC/GCM.

***Block Length Test* –** funkcja Analyze → Block Length Determination bada podzielność szyfrogramu. Jeżeli długość zawsze jest wielokrotnością 256 B i nigdy nie pojawia się nagłówek IV, mamy mocną poszlakę na RSA-2048, przy AES widzimy wielokrotność 16 B oraz losowe pierwsze 16 B (IV).

***CryptoScope / N-gram Analysis* –** tu można zobaczyć stałe prefiksy. Szyfrogram RSA z paddingiem PKCS #1 zaczyna się zwykle od 0x00 02, a plik zakodowany AES-CBC ma zupełnie losowy początek.

***Indeks koincydencji i autokorelacja* –** menu „Analysis → Classical Tests” ujawnia, czy tekst pozostaje w jakimś języku (IC≈0,065) – co wskazywałoby na szyfr klasyczny – czy ma losowy rozkład (IC≈0,038). Dzięki temu można odróżnić np. zlepiony plik ZIP (entropia wysoka, ale blok 16 B) od tekstu zakodowanego Vigenère.

## **Scenariusz użycia?**

1. **Histogram bajtów i entropia**  
   W CrypTool uruchamiamy Tools → Analyze → Byte Distribution. Jeżeli słupki dla wartości 0-255 są prawie równe, a entropia ≈ 8 bit/B, tekst wygląda jak biały szum. Odrzucamy więc klasyczne szyfry (np. Vigenère), bo te zostawiają nieregularny histogram.
2. **Test długości bloku**  
   Tools → Analyze → Determine Block Length.  
   • długość pliku = wielokrotność **256 B** → wskazuje na RSA-2048 (1024 b dałoby 128 B, 512 b - 64 B).  
   • długość = wielokrotność **16 B** → sugeruje szyfr blokowy AES lub 3DES.  
   • brak stałej wielokrotności → raczej szyfr strumieniowy (ChaCha20, RC4).
3. **Obecność losowego IV**  
   Przy AES-CBC / AES-GCM pierwsze 16 B szyfrogramu to losowy IV – będzie inny dla dwóch plików zaszyfrowanych tym samym kluczem. RSA IV-ki nie używa, więc początek dwóch szyfrogramów tego samego pliku jest identyczny.

**Wniosek praktyczny**  
Histogram płaski + wielokrotność 256 B + brak IV ⇒ szyfrogram najpewniej pochodzi z RSA-2048.  
Wielokrotność 16 B + zmienny pierwszy blok ⇒ szyfrowanie AES-CBC / AES-GCM.

## **Pytanie 3. W jaki sposób można wykorzystać narzędzia analizy tekstu dostępne w programie CryptTool do ustalenia hasła używanego do szyfrowania?**

***Automatic Analysis*** *–* moduł Analyze → Automatic Analysis iteruje po kluczach dla szyfrów klasycznych (Caesar, Vigenère, Playfair). Każdy kandydat odszyfrowuje tekst, a algorytm sprawdza, czy wskaźniki statystyczne (IC, chi-kwadrat, rozkład bigramów) zbliżają się do języka naturalnego. Klucz z najwyższym score zostaje zwrócony użytkownikowi.

***Brute-Force / Dictionary Attack*** *–* dla haseł symetrycznych (AES z passphrase, ZIP, PDF). Użytkownik definiuje zakres znaków lub podaje plik słownika; CrypTool generuje hasła, szyfruje lub deszyfruje pierwsze kilkaset bajtów i porównuje z regułą rozpoznawania (np. nagłówek „PK\003\004” w ZIP). Znaleziony kandydat jest prezentowany wraz z czasem trwania ataku.

***KeySearcher / DES Challenge*** *–* plugin skanuje przestrzeń 2⁵⁶ kluczy DES na GPU lub klastrze, sprawdzając zadany warunek (CRC, magic bytes). Ta sama metoda działa na 3DES w trybie meet-in-the-middle (2 × 2⁵⁶ zamiast 2¹¹²).

***RSA Factorization Attack*** *–* dla kluczy RSA ≤ 1024 b. Moduł „Number Theory → Factor n” implementuje GNFS i Pollard Rho; po znalezieniu czynników p,q program wylicza φ(n), klucz prywatny d ≡ e⁻¹ (mod φ(n)) i odszyfrowuje wiadomość.

W każdym przypadku CrypTool skraca czas testów przez „wczesne odrzucanie”, gdy odszyfrowanie fragmentu nie daje poprawnego nagłówka, kandydat klucza jest z miejsca odrzucany bez dalszego przetwarzania dużych danych. Dzięki temu słownik 5 mln haseł można przetestować w ciągu minut, a pełne 2³² kluczy RC4 w akceptowalnym czasie na współczesnym CPU + GPU.

# **Podsumowanie**

Podczas zajęć przeprowadzono kompletny cykl pracy z kryptografią asymetryczną i symetryczną – od wygenerowania kluczy, przez pomiary wydajności, aż po test odporności szyfrogramów na uszkodzenia.

**Klucze RSA 512 b / 1024 b / 2048 b.**  
Czas szyfrowania rósł zgodnie z ≈ O(k²): podwojenie długości klucza powodowało 4-krotny wzrost czasu. Deszyfrowanie rosło z ≈ O(k³) (8-krotny wzrost). Plik 5 MB szyfrował się kluczem 2048 b ~ 43 s, a odszyfrowywał ~ 20 min, co potwierdza teorię.

**AES-256-CBC (ten sam plik 5 MB).**  
Szyfrowanie trwało ~ 6 ms, deszyfrowanie ~ 5 ms – odpowiednio **> 200×** i **> 1000×** szybciej niż RSA-2048. Wynik potwierdza praktyczną zasadę: *RSA służy do wymiany klucza sesji, a masowe dane szyfruje się algorytmem symetrycznym*.

**Odporność RSA na naruszenie integralności.**

* Bit-flip jednego bajtu uszkodził tylko pierwszy 256-bajtowy blok; pozostała część plaintextu pozostała poprawna.
* Usunięcie 1, 11 lub 2048 B sprawiło, że **cały** odszyfrowany plik stał się pseudolosowy, a CrypTool nie zgłosił błędu.

Wniosek: sam RSA nie zabezpiecza integralności – potrzebny jest podpis cyfrowy lub MAC.

* **Rekomendowany model produkcyjny.**  
  W praktycznych systemach stosuje się układ hybrydowy: RSA (lub ECDH) wyłącznie do bezpiecznej wymiany klucza sesyjnego, natomiast cała komunikacja realizowana jest szybkim szyfrem blokowym z kontrolą integralności (np. AES-GCM).

Podsumowując, laboratorium potwierdziło zarówno teoretyczne zależności złożoności RSA, jak i przewagę wydajnościową oraz funkcjonalną nowoczesnych szyfrów symetrycznych w roli szyfrowania danych.