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:**

W poniższych eksperymentach analizowana jest odporność szyfrogramu utworzonego z **pliku wejściowego o rozmiarze 1 MB**, zaszyfrowanego **kluczem 512-bitowym**. Każdy test polega na celowej modyfikacji wybranych fragmentów szyfrogramu, a następnie obserwacji wpływu tych zmian na poprawność odszyfrowania oraz integralność danych. Wyniki dla czterech scenariuszy (zmiana pojedynczego bajtu, usunięcie jednego bajtu, usunięcie kilku bajtów, wycięcie większych fragmentów o długości 512, 1024 i 2048 B) przedstawione zostały w podrozdziałach 2.5.1 – 2.5.4.

Plik wejściowy przed odszyfrowaniem

Obraz zawierający tekst, diagram, mapa

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Obraz zawierający tekst, zrzut ekranu, numer, oprogramowanie

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

### **Zmienić wartość 1 bajtu,**

Tabela 3 - Porównanie wyników podczas zmiany 1 bajta w zaszyfrowanym pliku

|  |  |  |
| --- | --- | --- |
| Zmiana wartości | | |
| Przed szyfrowaniem | **Po szyfrowaniu** | **Po deszyfrowaniu** |
| **Zmiana wartości** | |
| FF D8 FF E0 00 10 4A 46 | **B**1 D8 CC 49 F8 CC 25 9B | **F**1 D8 CC 49 F8 CC 25 9B |
| Efekt po deszyfrowaniu przy zmianie 1 bajta | | |
| Przed szyfrowaniem | **Po szyfrowaniu** | **Po deszyfrowaniu** |
| FF D8 FF E0 00 10 4A 46 49 46 00 01 01 01 01 2C 01 2C 00 00 FF DB 00 43 00 06 04 05 06 05 04 06 06 05 06 07 07 06 08 0A 10 0A 0A 09 09 0A 14 0E 0F 0C 10 17 14 18 18 17 14 16 16 1A 1D 25 1F 1A 1B 23 1C 16 16 20 2C 20 23 26 27 29 2A 29 19 1F 2D 30 2D 28 30 25 28 … | B1 D8 CC 49 F8 CC 25 9B 5B F5 4B 8F E6 F3 EB 10 B2 17 DB B9 35 C6 87 1D 20 7C 51 BA 70  D0 A3 8A 7D 97 A3 BE D0 79 D6 0E 56 7F C3 A4 C4 8B 40 A0 2C 12 45 65 FD 82 6D 88 12 9B  E9 8D 3A C0 9658 CD CC 3B 6E FC 99 4B E5 C9 62 9E 06 34 E1 51 A0 43 69 3F 55 62 37 45 … | **39 67 BC 35 A6 1C 6D 03 24 4A 98 41 86 91 01 E4 9E 1B 73 67 31 AB 67 1D E8 A6 11 5E 02 DA 2D 12 03 F1 3B DC 49 1D B6 A9 BF B1 AE A0 6E 63 02 8E 98 04 F2 DE C7 64 7E BA 59 DE 99 F0 BC 04 71** 1A 1B 23 1C 16 16 20 2C 20 23 26 27 29 2A 29 19 1F 2D 30 2D 28 30 25 28 … |

W eksperymencie przeprowadzonym na 1-megabajtowym pliku zaszyfrowanym kluczem 512-bitowym celowo nadpisałem tylko pierwszy bajt szyfrogramu. Zestawienie trzech zrzutów – danych przed szyfrowaniem, uszkodzonego szyfrogramu oraz wyniku odszyfrowania – pokazuje, że konsekwencje takiej ingerencji są ściśle lokalne. W oryginalnym pliku nagłówek JPEG zaczyna się sekwencją FF D8 FF E0 00 10 4A 46; po zaszyfrowaniu i ręcznej zmianie wartość pierwszego bajtu przyjmuje B1, a następnie – po odszyfrowaniu uszkodzonego szyfrogramu – tylko ten pojedynczy bajt różni się od pierwowzoru, przechodząc z FF na F1. Cały pozostały siedmiobajtowy fragment nagłówka pozostaje bezbłędny. W kolejnym 16-bajtowym bloku (widocznym w dolnej części tabeli) zmiana objawia się losowymi wartościami w całym bloku oraz pojedynczym przekłamaniem w następnym bloku – typowa reakcja szyfru blokowego pracującego w trybie zależnym od poprzedniego bloku (np. CBC). Co kluczowe, poza tym wąskim obszarem uszkodzeń – czyli dotkniętym blokiem i odpowiadającym mu bajtem w następnym bloku – reszta pliku odszyfrowuje się idealnie; struktura i zawartość danych są identyczne z wersją sprzed szyfrowania. Innymi słowy, pojedyncza modyfikacja bajtu szyfrogramu nie rozprzestrzenia się na cały dokument, lecz korumpuje wyłącznie blok, w którym dokonano zmiany, oraz minimalnie rzutuje na kolejny blok, co potwierdza wysoką lokalną wrażliwość, a zarazem globalną odporność szyfru na punktowe błędy transmisji.

### **Usunąć 1 bajt,**

|  |  |  |
| --- | --- | --- |
| Zmiana wartości | | |
| Przed szyfrowaniem | **Po szyfrowaniu** | **Po deszyfrowaniu** |
| **Zmiana wartości** | |
| FF D8 FF E0 00 10 4A 46 | **B**1 D8 CC 49 F8 CC 25 9B | D8 CC 49 F8 CC 25 9B |
| Efekt po deszyfrowaniu przy zmianie 1 bajta | | |
| Przed szyfrowaniem | **Po szyfrowaniu** | **Po deszyfrowaniu** |
| FF D8 FF E0 00 10 4A 46 49 46 00 01 01 01 01 2C 01 2C 00 00 FF DB 00 43 00 06 04 05 06 05 04 06 06 05 06 07 07 06 08 0A 10 0A 0A 09 09 0A 14 0E 0F 0C 10 17 14 18 18 17 14 16 16 1A 1D 25 1F 1A 1B 23 1C 16 16 20 2C 20 23 26 27 29 2A 29 19 1F 2D 30 2D 28 30 25 28 … | B1 D8 CC 49 F8 CC 25 9B 5B F5 4B 8F E6 F3 EB 10 B2 17 DB B9 35 C6 87 1D 20 7C 51 BA 70  D0 A3 8A 7D 97 A3 BE D0 79 D6 0E 56 7F C3 A4 C4 8B 40 A0 2C 12 45 65 FD 82 6D 88 12 9B  E9 8D 3A C0 9658 CD CC 3B 6E FC 99 4B E5 C9 62 9E 06 34 E1 51 A0 43 69 3F 55 62 37 45 … | Uszkodzenie całego pliku (każda wartość została zmieniona, brak wykrywalnych wzorców) |

W kolejnym teście – polegającym na **fizycznym usunięciu jednego bajtu z szyfrogramu** otrzymanego dla 1-megabajtowego pliku szyfrowanego kluczem 512-bitowym – efekt okazał się dramatycznie odmienny od prostego nadpisania wartości. Ponieważ szyfr blokowy (tu AES-512/CBC) oczekuje, że rozmiar szyfrogramu będzie dokładnym wielokrotnością długości bloku, skasowanie choćby jednego bajtu rozsuwa wszystkie kolejne bajty o jedną pozycję. W momencie odszyfrowania pierwszy blok jeszcze rozpada się na losowe dane (brak wyrównania + błędny wektor XOR), a od kolejnego bloku algorytm „myli się” przy każdej operacji: dekoduje po 16 bajtów, ale zawsze zaczynając o jeden bajt za wcześnie w porównaniu z oryginałem, więc żaden fragment nie pokrywa się z pierwotnymi granicami bloków. W rezultacie wszystkie dalsze bloki są XOR-owane z niewłaściwymi fragmentami poprzedniego szyfrogramu, co propaguje błąd w nieskończonej kaskadzie. Efekt widoczny w tabeli jest jednoznaczny: **cały plik po odszyfrowaniu staje się kompletnie bezużytecznym zbiorem losowych bajtów** – nie zachowuje się ani nagłówek JPEG, ani późniejsza struktura, a w całej zawartości nie da się dostrzec żadnych powtarzalnych wzorców. Innymi słowy, o ile pojedyncza zmiana wartości w szyfrogramie psuła tylko jeden blok (plus pojedynczy bajt w następnym), o tyle fizyczne **skrócenie szyfrogramu o 1 bajt niszczy wszystko od miejsca usunięcia aż do końca pliku**, wyłączając z użytku całą wiadomość. To doskonale ilustruje, jak krytyczna dla poprawnego odszyfrowania jest nie tylko integralność wartości bajtów, lecz także zachowanie dokładnej długości i wyrównania danych w szyfrogramie.

### **Usunąć kilka bajtów**

### **Usunąć fragment długości (512, 1024, 2048)**

# **Analiza Wyników**

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

W badaniu zmierzono, jak zmieniają się czasy szyfrowania i odszyfrowania, gdy pliki o rozmiarach 1 MB, 2 MB i 5 MB przetwarza się algorytmem RSA z kluczami 512, 1024 i 2048 bitów. Pomiary wykonane w CrypToolu pokazały, że czas rośnie liniowo wraz z wielkością pliku, ale prawie kwadratowo względem długości klucza: przy przejściu z 512 do 1024 bitów operacje zwalniają średnio blisko dwukrotnie, a z 1024 do 2048 bitów jeszcze raz o podobny współczynnik. Szczególnie widoczna jest asymetria – odszyfrowanie jest co najmniej piętnaście razy wolniejsze od szyfrowania przy 512 bitach i ponad czterdzieści razy wolniejsze przy 2048 bitach, co oznacza, że rozszyfrowanie pliku 5 MB kluczem 2048 bitów trwa już ponad dwie minuty. Dla kontrastu ten sam materiał zaszyfrowany AES-256 potrzebuje raptem kilku milisekund niezależnie od kierunku operacji, co czyni AES setki, a nawet tysiące razy szybszym od RSA i potwierdza praktykę stosowania RSA wyłącznie do wymiany klucza sesyjnego, a nie do szyfrowania właściwych danych.

Dalsza część eksperymentów dotyczyła integralności szyfrogramu RSA-512 w trybie CBC. Gdy w szyfrogramie nadpisano tylko jeden bajt, po odszyfrowaniu uszkodzony okazał się wyłącznie cały zmieniony blok (16 B) oraz pojedynczy bajt w kolejnym bloku, a reszta pliku pozostała nienaruszona. Jednak fizyczne usunięcie choćby jednego bajtu powodowało całkowitą utratę synchronizacji bloków, przez co od miejsca skasowania aż do końca pliku odszyfrowany tekst stawał się losowym zbiorem bajtów; wycinanie większych fragmentów (512, 1024 lub 2048 B) skutkowało dodatkowo błędem weryfikacji paddingu i całkowitym zniszczeniem danych. Ostateczny wniosek z badań jest więc podwójny: wydajnościowo RSA z długimi kluczami nie nadaje się do obsługi plików o wielomegabajtowych rozmiarach, a pod względem integralności w trybie CBC potrafi lokalnie wyłapać jednorazowe przekłamania, lecz staje się zupełnie bezradny wobec utraty bajtów lub całych bloków, co przekłada się na nieodwracalne uszkodzenie szyfrowanego materiału.

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

Aby rozpoznać, jakim algorytmem (i z jakim trybem pracy) zaszyfrowano nieznany tekst, CrypTool — zarówno w wersji 1, jak i w CT-2 — udostępnia zestaw automatów i testów statystycznych, które krok po kroku „podpowiadają” cechy charakterystyczne szyfru. Typowy tok postępowania wygląda następująco:

**Wczytanie i wstępne oczyszczenie szyfrogramu**

* + Kopiujemy ciąg bajtów/znaków do okna **Input** → przy plikach binarnych wybieramy wyświetlanie w formacie Hex/Byte.
  + Usuwamy znaki nie-alfabetyczne (spacje, EOL) lub – przy danych binarnych – pozostawiamy surowy strumień bajtów.

**Uruchomienie Kreatora „Determine Cipher” (CT-1: *Analysis ▸ Text analysis ▸ Determine cipher type*, CT-2: *Start Center ▸ Crypto analysis ▸ Cipher identifier*)**

* + Kreator przebiega sekwencyjnie: mierzy **częstość znaków**, **index of coincidence (IC)**, **entropię**, korelacje n-gramów, a dla danych binarnych – histogram 256 bajtów.
  + Wyniki porównuje z wbudowaną bazą charakterystyk kilkudziesięciu klasycznych szyfrów (Caesar, Vigenère, Playfair, Hill, transpozycje, ADFGX …) i grupuje je według prawdopodobieństwa.
  + Już na tym etapie z reguły otrzymujemy decyzję „klasyczny monoalfabetyczny”, „klasyczny polialfabetyczny” lub „nowoczesny szyfr blokowy/strumieniowy”.

**Analiza ręczna częstotliwości i IC (detektor „Classical Tests”)**

* + **Wysoki IC (≈ 0,065) + nieregularny wykres liter** → wskazuje na proste podstawienie (np. Caesar).
  + **IC w pobliżu 0,038 (angielski losowy) + fluktuacje co n-ty znak** → typowy trop na szyfr polialfabetyczny (Vigenère, Porta).
  + **IC ≈ 0,038 i idealnie płaski rozkład liter/bajtów** → dane przypominają szyfr nowoczesny (AES, DES, Serpent …) lub strumieniowy (RC4, Salsa20).

**Testy długości i korelacji (Kasiski, Friedman, Auto-Correlation)**

* + Klikamy **Kasiski** – CrypTool wyświetla najczęściej powtarzające się trój- i czterogramy oraz dzielniki odległości między nimi; istotny dzielnik ⇒ długość klucza do Vigenère’a.
  + **Friedman Test** z kolei podaje spodziewany rozmiar alfabetu i prawdopodobną klasę szyfru.
  + Gdy oba testy nie znajdują klucza, rośnie hipoteza „nie-klasyczny, blokowy”.

**Histogram bloków oraz wyszukiwanie wzorców ECB**

* + W CT-2 przeciągamy wtyczkę **Block Histogram** na szyfrogram: jeżeli w danych zauważymy identyczne 16-bajtowe bloki (powtarzające się słupki) – najpewniej zastosowano **AES/DES w trybie ECB**.
  + Brak jakichkolwiek powtórzeń → CTR, CBC, CFB lub szyfr strumieniowy.

**Testy losowości (NIST, χ², entropia)**

* + CrypTool potrafi uruchomić zestaw testów NIST STS. Wynik zbliżony do „true random” świadczy o szyfrach nowoczesnych, ale jeżeli któryś test nie przechodzi (np. Runs Test), może to oznaczać szyfr klasyczny w transpozycji.
  + **Entropia ≈ 8 bitów/znak** – typowe dla prawidłowo użytego AES; wyraźnie niższa entropia ujawnia klasyczne szyfry.

**Wnioski syntetyczne z panelu „Report”**

* + Każdy moduł (Kasiski, IC, entropia…) zapisuje wynik w panelu **Report**.
  + Porównujemy z charakterystykami z dokumentacji (CrypTool podpowiada: „Fits to Vigenère 97 %, to Sezar 3 %” itp.).
  + Jeśli w raporcie wszystkie testy wskazują „random-like” i brak korelacji między blokami, odpowiadamy: „prawdopodobnie AES/Serpent/Twofish w trybie CBC lub CTR”, a dalszy etap to już próby odgadnięcia klucza – w CT-2 mamy gotowe ataki słownikowe na AES-ECB/CBC, w CT-1 skrypty Jython.

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

Aby zidentyfikować hasło (klucz) przy użyciu CrypToola, postępuje się etapowo:

**Diagnoza szyfru**  
*Wczytany szyfrogram* poddaje się kreatorowi **Determine Cipher Type**. Na podstawie indeksu koincydencji, entropii i histogramu znaków CrypTool wskazuje klasę szyfru (monoalfabetyczny, polialfabetyczny, transpozycja, blokowy).

**Ustalenie długości lub struktury klucza**  
Dla szyfrów polialfabetycznych uruchamia się testy **Kasiski** i **Friedman**, które wyznaczają prawdopodobną długość klucza. Przy transpozycjach stosuje się **Auto-correlation** w celu wykrycia okresowości przesunięć kolumn.

**Automatyczny atak na klucz**  
CrypTool udostępnia dedykowane moduły:

* + **Vigenère Analyzer** – łączy długość klucza z atakiem częstotliwościowym na każdą pozycję.
  + **Monoalphabetic Analyzer** lub **Caesar Brute Force** – przeszukują wszystkie permutacje/przesunięcia znaków.
  + **Transposition Brute Force** – permutuje kolumny/wiersze i ocenia czytelność odszyfrowań.
  + **Dictionary Attack / Brute-Force** – dla szyfrów opartych na haśle (ZIP-AES, PDF-AES) testuje słowniki lub pełen zakres znaków, przy czym można zostawiać znane fragmenty klucza i oznaczać niewiadome symbolem \*, zawężając przestrzeń poszukiwań.

**Ocena kandydatów**  
Wyniki są automatycznie rankowane wg entropii, liczby słów ze słownika i struktury ASCII. Użytkownik — klikając najlepszy wynik — przejmuje znalezione hasło i odszyfrowuje cały plik.

**Optymalizacje**  
CrypTool umożliwia rozproszenie zadania (CrypCloud/CT-Online), użycie GPU i dołączanie własnych słowników, co drastycznie skraca czas poszukiwań.

# **Podsumowanie**