Materiały do przedmiotu:

Podstawy kryptografii

Laboratorium nr 1

Wprowadzenie do narzędzi kryptograficznych Praktyczne zastosowanie algorytmów szyfrowania

Autor:

dr Michał Dolecki

Fundusze Europejskie dla Rozwoju Społecznego

Rzeczpospolita Polska

Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Spis treści

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Wprowadzenie, tematyka zajęć

Kryptografia często kojarzona jest bezpośrednio z szyframi, kluczami i algorytmami, ale zanim przejdziemy do konkretnych metod szyfrowania, musimy zrozumieć naturę przedmiotu, którym będziemy się zajmować – czyli ochroną informacji. W ramach laboratoriów przedstawione zostaną narzędzia służące do ochrony informacji w aspektach integralności, poufności, dostępności i niezaprzeczalności. Na początek skupimy się na sposobie mierzenia informacji z wykorzystaniem entropii.

Podczas dzisiejszych zajęć nauczymy się przede wszystkim mierzyć entropię. Poznamy dwa podejścia: entropię Hartleya, która mówi o pojemności informacyjnej źródła przy założeniu równych prawdopodobieństw wszystkich symboli, oraz entropię Shannona, która uwzględnia rzeczywiste, często nierównomierne rozkłady prawdopodobieństw. Obliczymy, ile informacji niosą konkretne zdarzenia, jaka jest średnia ilość informacji przypadająca na znak w różnych źródłach oraz porównamy pod tym kątem różne języki naturalne. Przekonamy się, dlaczego język polski czy angielski mają znacznie niższą entropię, niż wynikałoby to z samej liczby używanych liter – i jakie to ma konsekwencje dla kryptoanalizy.

Cel laboratorium

Student po realizacji zajęć potrafi zdefiniować i odróżnić od siebie miary entropii Hartleya oraz entropii Shannona, wskazując zakresy ich zastosowania w teorii informacji i kryptografii; student umie obliczyć ilość informacji przenoszoną przez konkretne powiadomienie przy założeniu danego rozkładu prawdopodobieństwa oraz wyznaczyć średnią ilość informacji przypadającą na jeden znak (entropię źródła) dla wskazanego zbioru komunikatów; student potrafi dokonać porównania entropii języków naturalnych z entropią źródeł o rozkładzie równomiernym i sformułować wnioski dotyczące konsekwencji redundancji języka dla bezpieczeństwa kryptograficznego (w tym podatności na ataki znające rozkład statystyczny); wreszcie student rozumie fundamentalną rolę entropii w nowoczesnych systemach kryptograficznych, potrafiąc wyjaśnić, dlaczego wysoka

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

entropia jest warunkiem koniecznym bezpieczeństwa generatorów liczb losowych oraz kluczy kryptograficznych, a także ocenić jakość źródła losowości w kontekście projektowania i analizy systemów ochrony informacji.

Instrukcja laboratoryjna

Model CIA

Model CIA (Confidentiality, Integrity, Availability) jest fundamentalną triadą realizowaną przez narzędzia kryptografii.

Poufność (Confidentiality): To najstarszy i najbardziej oczywisty cel kryptografii – zapewnienie, że informacja nie trafi w niepowołane ręce. W ramach laboratoriów, zrealizujecie ten cel, poznając narzędzia takie jak szyfry symetryczne (np. AES) i asymetryczne (np. RSA), które przekształcają czytelny tekst w zaszyfrowany, nieczytelny dla atakującego.

Integralność (Integrity): To cel polegający na zagwarantowaniu, że dane nie zostały zmodyfikowane w nieautoryzowany sposób w trakcie przechowywania lub transmisji. Jest to idealne miejsce do wprowadzenia funkcji skrótu (hash functions), które działają jak "kryptograficzny odcisk palca" dokumentu – każda, nawet najmniejsza zmiana w danych, powoduje całkowitą zmianę skrótu, co od razu sygnalizuje naruszenie integralności.

Dostępność (Availability): Ten aspekt jest często pomijany w kontekście czysto kryptograficznym, ale zawsze należy o nim pamiętać, zwłaszcza, że ataki typu DDoS dosyć popularnym zagrożeniem. Kryptografia wspiera dostępność, zapewniając, że dane są dostępne dla uprawnionych użytkowników w oczekiwanej formie. Na przykład, szyfrowanie dysku chroni dane przed utratą w przypadku kradzieży fizycznego nośnika, a mechanizmy odzyskiwania kluczy zapobiegają sytuacji, w której legalny użytkownik traci dostęp do własnych informacji.

Rozszerzenie o niezaprzeczalność

Wiele źródeł traktuje niezaprzeczalność jako czwarty, główny cel kryptografii, obok poufności, integralności i uwierzytelniania.

Niezaprzeczalność (Non-repudiation): To mechanizm, który uniemożliwia nadawcy wyparcie się wysłania wiadomości, a odbiorcy wyparcie się jej otrzymania. W laboratoriach zostanie to zobrazowane, poprzez podpisy cyfrowe (digital

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

signatures). Pokazując proces, w którym wiadomość jest podpisywana prywatnym kluczem nadawcy, a odbiorca weryfikuje podpis kluczem publicznym, udowodnicie, że tylko posiadacz klucza prywatnego mógł być autorem podpisu, co zapewnia niezbity dowód jego autorstwa.

Entropia

Entropia a siła stosowanych narzędzi kryptograficznych:

- Przy szyfrowaniu (poufność) – siła klucza zależy od jego entropii.
- Przy funkcjach skrótu (integralność) – odporność na kolizje opiera się na wysokiej entropii wyjścia.
- Przy podpisach cyfrowych (niezaprzeczalność) – bezpieczeństwo klucza prywatnego znów sprowadza się do jego entropii.

Podstawowe pojęcia ochrony danych

„Granice mojego języka oznaczają granice mojego świata" powiedział austriacki filozof Ludwig Wittgenstein. Chcąc więc poznać nową dziedzinę należy odpowiednio poszerzyć używany język. Wiedza to sztuka wprowadzania rozróżnień dla rzeczy potocznie tożsamych i ich definiowania. Zacznijmy więc od pojęcia danej i informacji. Choć w codziennej rozmowie często używamy tych słów zamiennie, w świecie teorii informacji i kryptografii różnica między nimi jest fundamentalna.

Dana (Data)

Dane to czyste fakty, symbole lub wartości, które same w sobie nie niosą konkretnego znaczenia. W informatyce to po prostu ciągi bitów (0 i 1).

Charakterystyka: Są obiektywne, statyczne i nieprzetworzone.

Przykład: Ciąg znaków 01000001. Bez kontekstu to tylko 8 bitów. 1945 to tylko liczba.

W Teorii Informacji: Dane to wartości, sygnał, który przesyłamy przez kanał komunikacyjny.

Informacja (Information)

Informacja pojawia się wtedy, gdy dane zostaną przetworzone, zinterpretowane i osadzone w kontekście. Z perspektywy Claude'a Shannona (ojca teorii informacji), informacja to redukcja niepewności.

Charakterystyka: Jest subiektywna (zależy od odbiorcy) i celowa.

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Przykład: Jeśli wiemy, że wspomniany wyżej ciąg 01000001 to kod ASCII, zamienia się on w literę „A”. To już jest informacja. Subiektywizm informacji polega na możliwości osadzenia danych w różnych kontekstach: rok 1945 albo 1945 zł.

W Teorii Informacji: Mierzymy ją za pomocą entropii. Im mniej prawdopodobne jest wystąpienie danej wiadomości, tym więcej informacji ze sobą niesie.

Przypomnienie

W tym miejscu dobrze jest sobie przypomnieć z podstawowych zagadnień informatyki zagadnienia związane z bitami:

- Pochodzenie nazwy bit
- Ilość informacji możliwą do zapisania na jednym, dwóch, czterech, ośmiu, 32 i 64 bitach
- Reprezentację liczb całkowitych bez znaku i ze znakiem

Entropia Hartleya

Zacznijmy od próby określenia ile informacji niesie ze sobą wystąpienie pewnych zjawisk na przykładach z życia codziennego, a w dalszej kolejności wprowadzimy konkretne wzory. Określmy jak liczba stanów wpływa na informację wyrażoną jako niepewność.

Załóżmy, że w mieście mamy jeden typ autobusu i codziennie dojeżdżamy nim na zajęcia albo do pracy. Przyjazd konkretnego typu autobusu na przystanek nie jest związany z żadną niepewnością, bo jest tylko jeden możliwy typ.

Pewnego dnia władze miasta zakupiły pewną liczbę nowoczesnych autobusów niskopodłogowych z klimatyzacją. Mamy już więc dwa typy autobusów. Droga na zajęcia albo do pracy wiąże się z pewną niepewnością, bo nie wiemy jaki typ autobusu przyjedzie dziś, a kiedy już pojawi się na przystanku mamy informację jakim autobusem pojedziemy.

Tak więc dla możliwego jednego stanu mamy informację równą 0, a zwiększając liczbę stanów do dwóch mamy wzrost informacji, który oznaczymy d1.

Kontynuując historię, miasto kupiło kolejną partię nowych autobusów. Mamy już więc trzy różne modele. Oczywiście niepewność związana z modelem, który dziś przyjedzie na przystanek wzrosła, ale nie jest to już taki przyrost, jak był poprzednio, gdy pojawiły się pierwsze nowoczesne pojazdy.

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Matematycznie liczba stanów wzrosła do trzech, a wzrost informacji oznaczmy jako d2. Zauważamy, że d1 > d2.

Miasto oczywiście kupuje dalej kolejne typy autobusów, liczba możliwych pojazdów, którymi podróżujemy jest coraz większa, niepewność rośnie, ale z każdym nowym modelem przyrosty są coraz mniejsze.

Dla kolejnych dodawanych stanów, mamy kolejne przyrosty d3, d4, d5, ..., dn oraz d1 > d2 > d3 > d4 > ... > dn.

Zadanie 1

Narysuj wykres funkcji, która poszczególnym liczbom stanów przyporządkowuje wartość informacji, wiedząc że przyrosty informacji są coraz mniejsze wraz ze wzrostem liczby możliwych stanów. Jaką funkcję przypomina ten wykres?

Ralph Hartley w 1928 roku zaproponował, aby do obliczania entropii użyć logarytmu o podstawie równej liczbie rozróżnialnych stanów w reprezentacji informacji, a ponieważ było to przed powstaniem maszyn bazujących na systemie dwójkowym, oryginalnie używał systemu dziesiętnego.

Dla N możliwych stanów Entropia Hartleya wyraża się wzorem:

$$H = \log_{10} N.$$

Obecnie będziemy używać jednak logarytmu o podstawie 2 do obliczania entropii Hartleya

$$H = \log_{2} N.$$

Jednostką entropii jest bit.

Można powiedzieć, że że jest to miara całkowitej pojemności nośnika informacji. Mierzy ona ilość informacji potrzebną do wskazania jednego z A możliwych, równoprawdopodobnych symboli.

Przykład

Ile bitów informacji potrzeba, aby zakodować rzut kostką? $\log_2(6) \approx 2{,}58$ bita. W praktyce należałoby przekształcić wynik funkcją ceil, co daje 3 bity.

Przykład

Ile bitów potrzeba, aby zakodować literę z 26-literowego alfabetu? $\log_2(26) \approx 4{,}7$ bita. W praktyce 5 bitów. Na ilu bitach koduje znaki komputer?

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Entropia Shannona

Nieco od innej strony do problemu ilości informacji czy też niepewności związanej z pojawianiem się kolejnego zdarzenia podszedł Shannon.

Wracając do przykładu naszego miasta i etapu, w którym postanowiono kupić pierwszą serię nowych autobusów i załóżmy że miasto dysponowało 100 starych pojazdów. W zależności od tego ile nowych autobusów zostało zakupionych, nasza niepewność co do tego jak autobus przyjedzie jest inna dla 5, a inna dla 50 nowych autobusów.

Podejście Hatleya opierało się tylko na liczbie stanów, natomiast w podejściu Shannona istotne jest też prawdopodobieństwo wystąpienia każdego stanu.

Shannon zdefiniował pojęcie informacji własnej (self-information) – to ilość informacji niesiona przez konkretne zdarzenie. Dla zdarzenia o prawdopodobieństwie wystąpienia $p(x)$ wynosi ona:

$$I(x) = -\log_2 p(x).$$

Intuicyjnie informacja własna mierzy zaskoczenie:

- zdarzenie bardzo prawdopodobne, to mało informacji,
- zdarzenie rzadkie, to dużo informacji,
- zdarzenie pewne ($p(x) = 1$), które wystąpiło, to żadna informacja $I = 0$.

Entropia Shannona to średnia informacja własna:

$$H(X) = E[I(x)] = -\sum_x p(x) \log_2 p(x),$$

co można łatwiej zapamiętać stosując nieco bardziej zwięzły zapis:

$$H = -\sum_i p_i \log\ p_i,$$

gdzie p_i jest prawdopodobieństwem zajścia i-tego zdarzenia. Informacja własna i entropia liczone są w bitach.

Entropia Hartleya jest szczególnym przypadkiem entropii Shannona dla wszystkich zdarzeń jednakowo prawdopodobnych.

Zadanie 2

Udowodnij, że dla jednakowo prawdopodobnych zdarzeń entropia Shannona jest równa entropii Hatleya.

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Ilość informacji

Entropia Hartleya jest maksymalna możliwa średnia informacja na znak dla danego alfabetu.

Entropia Shannona jest również interpretowana jako średnia ilość informacji przypadająca na znak w powiadomieniu.

Różnica między nimi to redundancja komunikatu.

Chcąc policzyć ilość informacji w całym powiadomieniu można stosować wzór:

$$I = nH,$$

gdzie n – liczba znaków w powiadomieniu, a H – entropia Shanonna dla wiadomości.

Ćwiczenia samodzielne

Zadanie 1.1

Oblicz entropię Hartleya dla alfabetu polskiego i angielskiego.

Zadanie 1.2

Źródło emituje dwa symbole A i B. Prawdopodobieństwo wystąpienia A wynosi 0,5, a B 0,5. Oblicz ilość informacji własnej niesioną przez pojawienie się A. Ile informacji niesie pojawienie się B?

Zadanie 1.3

Źródło emituje dwa symbole A i B, ale prawdopodobieństwa wynoszą: p(A)=0,9, p(B)=0,1. Oblicz ilość informacji własnej niesioną przez A i przez B. Które zdarzenie niesie ze sobą więcej informacji?

Zadanie 1.4

Dla źródła z zadania 1.3 oblicz entropię Shannona. Jaka jest średnia ilość informacji przypadająca na jeden symbol? Jaka jest redundancja tego źródła?

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Zadanie 1.5

Dla źródła binarnego (emitującego sygnały 0 albo 1) narysuj wykres zależności entropii Hartleya i Shannona w zależności od p(0).

Zadanie 1.6

Oblicz entropię Hartleya i Shannona dla komunikatu zawierającego Twoje imię i nazwisko.

Zadanie 1.7

Napisz program z graficznym interfejsem użytkownika pozwalający na wprowadzenie dowolnego tekstu i policzenie dla niego entropii Hartleya i Shannona.

Zadanie 1.8

Wybierz kilka akapitów z ulubionej książki w języku polskim i czterech innych językach np. niemiecki, angielski, inne ciekawe języki używające liter jak turecki (można użyć LLM do tłumaczenia). Korzystając z własnego programu policz entropie Shannona dla różnych języków i różnych długości fraz. Tekst polski i zestawienie w postaci tabeli:

Entropia	polski	inny język 1	inny język 2	inny język 3	inny język 4
1 zdanie					
5 zdań					
10 zdań					
50 zdań					
100 zdań					

wraz z krótkimi wnioskami z porównania prześlij jako podsumowanie laboratorium do osoby prowadzącej zajęcia.

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Pytania pomocnicze

1. Czym różnią się dane od informacji, podaj przykłady?
2. Jakie są obszary ochrony informacji i co obejmują?
3. Czym jest entropia w sensie Hatleya?
4. Czym jest entropia Shannona?

Podsumowanie

Pojęcie entropii jest podstawą do tworzenia skutecznych metod kryptografii znajdujących zastosowania w zapewnieniu ochrony informacji. Laboratoria przedstawiły czym są dane i informacje i w jaki sposób entropia – abstrakcyjna, wywodząca się z teorii informacji miara nieprzewidywalności, przekłada się na konkretne mechanizmy zabezpieczające dane w systemach teleinformatycznych. Świadome posługiwanie się entropią pozwala nie tylko projektować bezpieczne systemy, ale także rzetelnie oceniać ich rzeczywistą siłę – odróżniać pozorne zabezpieczenia od tych, które faktycznie chronią informację.

Literatura

1. D. R. Stinson, M. B. Paterson, Kryptografia. W teorii i w praktyce, PWN 2021

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Materiały do przedmiotu:

Podstawy kryptografii

Laboratorium nr 2

Tworzenie i weryfikacja podpisów cyfrowych

Autor:

dr Michał Dolecki

Fundusze Europejskie dla Rozwoju Społecznego

Rzeczpospolita Polska

Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Spis treści

Fundusze Europejskie dla Rozwoju Społecznego

Rzeczpospolita Polska

Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Wprowadzenie, tematyka zajęć

Fundamentem wszelkich działań w kryptografii jest **triada CIA** (Confidentiality, Integrity, Availability). O ile poufność skupia się na ukrywaniu danych przed niepowołanym wzrokiem, o tyle **integralność (Integrity)** odpowiada za pewność, że informacja nie została zmodyfikowana – czy to w wyniku celowego ataku, czy zwykłego błędu transmisji. Zapewnienie integralności to proces wielowarstwowy: musimy nie tylko potrafić wykryć, że doszło do zmiany choćby jednego bitu, ale w wielu przypadkach również wskazać miejsce błędu lub go naprawić. Bez mechanizmów kontrolnych nie jesteśmy w stanie ufać żadnej informacji przetwarzanej przez systemy cyfrowe.

Zanim przejdziemy do zaawansowanych mechanizmów kryptograficznych, przeanalizujemy klasyczne metody kontroli błędów, które stanowią technologiczne podwaliny dzisiejszych systemów. Rozpoczniemy od najprostszego **bitu parzystości**, który pozwala na wykrycie pojedynczych przekłamań, oraz **algorytmu CRC (Cyclic Redundancy Check)**, powszechnie stosowanego w sieciach Ethernet i systemach plików do weryfikacji spójności bloków danych. Poznamy również **kod Hamminga**, który wykracza poza samo wykrywanie, oferując możliwość automatycznej korekcji błędów (ECC) dzięki sprytnemu wykorzystaniu nadmiarowości.

Dopiero po zrozumieniu tych niskopoziomowych metod detekcji, przejdziemy do obszaru bezpieczeństwa aktywnego. Zobaczymy, jak **kryptograficzne funkcje skrótu** zamieniają dowolnie duże zbiory danych w unikalne „odciski palców", co stanowi pomost do mechanizmów **podpisu cyfrowego**. Wprowadzenie kryptografii asymetrycznej pozwoli nam rozszerzyć klasyczną triadę CIA o kluczowy fundament nowoczesnych systemów: **niezaprzeczalność (Non-repudiation)**. Dzięki niej nie tylko mamy pewność co do integralności danych, ale zyskujemy dowód, że konkretna operacja została wykonana przez posiadacza danego klucza prywatnego. Sprawia to, że nadawca nie może wyprzeć się wysłania wiadomości, co przenosi naszą dyskusję z czystej matematyki błędu w stronę cyfrowej tożsamości i wiarygodności działań w sieci.

Sugerowana kolejność wykonywania laboratorium to wykonanie zadań dotyczących integralności i zasad konstruowania funkcji skrótu, natomiast zagadnienia dotyczące podpisów najlepiej zrealizować po zagadnieniach kryptografii asymetrycznej, kiedy pojęcie kluczy i ich użycie zostanie lepiej przećwiczone.

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Cel laboratorium

Student po realizacji zajęć potrafi scharakteryzować i odróżnić od siebie podstawowe metody zabezpieczania integralności danych – od najprostszego bitu parzystości, przez wielomianowe kody CRC, aż po kody Hamminga z możliwością korekcji błędów – wskazując ich ograniczenia w kontekście ochrony przed atakami intencjonalnymi; student umie zdefiniować funkcję skrótu (funkcję hashującą) jako odwzorowanie dowolnego ciągu bitów w ciąg o ustalonej długości, wyjaśniając kluczowe własności takich funkcji (jednokierunkowość, odporność na kolizje, efekt lawinowy) oraz wskazać praktyczne zastosowania w kryptografii; wreszcie student rozumie i potrafi objaśnić schemat podpisu cyfrowego, w którym podpisywana jest nie sama wiadomość, a jej skrót kryptograficzny – wyjaśniając, dlaczego takie podejście umożliwia zapewnienie autentyczności, integralności i niezaprzeczalności wiadomości przy jednoczesnym zachowaniu wydajności obliczeniowej, a także potrafi zidentyfikować zagrożenia wynikające z zastosowania funkcji skrótu o niedostatecznej odporności kolizyjnej.

Instrukcja laboratoryjna

Bit parzystości

Bit parzystości to najprostsza, a zarazem najstarsza metoda wykrywania błędów (Error Detection Code) w transmisji danych. Jest to fundament, na którym opierają się bardziej złożone systemy. Idea polega na dodaniu do ciągu danych (np. bajta) **jednego dodatkowego bitu**, którego wartość zależy od liczby "jedynek" w tym ciągu. Dzięki temu nadawca i odbiorca umawiają się na określoną strukturę danych, którą łatwo zweryfikować.

Wyróżniamy dwa główne warianty:

1. **Parzystość (Even Parity):** Bit parzystości przyjmuje taką wartość (0 lub 1), aby łączna liczba jedynek w całym bloku (dane + bit parzystości) była **parzysta**.
2. **Nieparzystość (Odd Parity):** Bit parzystości jest ustawiany tak, aby łączna liczba jedynek była **nieparzysta**.

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Do oryginalnej wiadomości dodawane są dodatkowe bity, dlatego ten typ metod bywa nazywany kodowaniem nadmiarowym, redundantnym.

Przykład (Even Parity)

Załóżmy, że chcemy wysłać 7-bitową wiadomość: 1011001.

1. **Zliczamy jedynki:** W naszym ciągu są cztery jedynki (4).
2. **Ustalany bit:** Ponieważ 4 jest już liczbą parzystą, bit parzystości wyniesie **0**.
3. **Wysłany ciąg:** 1011001 + 0 = 10110010.

Gdyby nasza wiadomość brzmiała 1011011 (pięć jedynek), bit parzystości musiałby wynosić **1**, aby dopełnić sumę do sześciu (liczba parzysta).

Ograniczenia

Mimo swojej prostoty, bit parzystości ma istotne wady z punktu widzenia cyberbezpieczeństwa i niezawodności:

- **Wykrywa tylko błędy nieparzyste:** Jeśli podczas transmisji zmieni się **jeden** bit, odbiorca to zauważy (suma jedynek przestanie być parzysta). Jeśli jednak zmienią się **dwa** bity naraz, suma pozostanie parzysta, a błąd przejdzie niezauważony.
- **Brak możliwości naprawy:** Odbiorca wie, że dane są uszkodzone, ale nie wie, w którym miejscu. Musi poprosić o ponowne przesłanie danych.
- **Brak ochrony przed manipulacją:** Atakujący, który zna tę metodę, może celowo zmienić dwa bity w taki sposób, aby bit parzystości nadal się zgadzał. Dlatego w cyberbezpieczeństwie bit parzystości traktujemy jako ochronę przed "szumem", a nie przed "hakerem".

Kod CRC

Algorytm **CRC (Cyclic Redundancy Check)**, to lepsze narzędzie niż bit parzystości. Podczas gdy ten ostatni wykrywa tylko proste błędy, CRC potrafi zidentyfikować całe serie przekłamań (tzw. *burst errors*), co czyni go standardem w komunikacji sieciowej (np. Ethernet, Wi-Fi) i archiwizacji danych (ZIP, RAR).

Podstawy matematyczne

CRC opiera się na **dzieleniu wielomianów**, co w praktyce sprowadza się do operacji bitowych **XOR**. Zamiast traktować dane jako liczbę, traktujemy je jako współczynniki wielomianu. Na przykład ciąg bitów 1011 reprezentuje wielomian:

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

$$1 \cdot x^3 + 0 \cdot x^2 + 1 \cdot x^1 + 1 \cdot x^0$$

Aby obliczyć sumę kontrolną CRC, nadawca i odbiorca muszą najpierw uzgodnić wspólny **dzielnik**, nazywany **wielomianem generującym** (np. CRC-32).

Wyznaczanie sumy kontrolnej:

1. **Rozszerzenie danych:** Do oryginalnej wiadomości dopisujemy pewną liczbę zer (o jeden mniej niż długość dzielnika). Jest to miejsce na współczynniki wielomianu będącego resztą z dzielenia danego wielomianu przez wielomian generujący.
2. **Dzielenie modulo 2:** Dzielimy tak przygotowany ciąg przez wielomian generujący, używając operacji XOR zamiast odejmowania.
3. **Reszta z dzielenia:** Wynik tego dzielenia (reszta) to właśnie nasze **CRC**.
4. **Dołączenie CRC:** Nadawca zastępuje dopisane wcześniej zera obliczoną resztą i wysyła całość.

Odbiorca otrzymuje wiadomość wraz z doklejonym kodem CRC i wykonuje dokładnie to samo dzielenie przez ten sam wielomian generujący w celu jej weryfikacji.

- Jeśli **reszta wynosi 0**, uznaje się, że dane są nienaruszone.
- Jeśli **reszta jest różna od 0**, oznacza to, że w trakcie transmisji nastąpiło przekłamanie.

Zaletą algorytmu poza szybkością (XOR jest podstawowym rozkazem procesora) jest fakt, że jedna funkcja realizuje wyznaczanie sumy i jej weryfikację.

Przykład

dzielnik w postaci wielomianu 1011

stopień dzielnika: 3

najwyższy możliwy stopień wielomianu będącego resztą z dzielenia: 2

liczba współczynników reszty: 3 i jest to długość sumy kontrolnej CRC

dane: 11010011101110

Wyznaczanie sumy CRC:

```
11010011101110 000      <--- 14 bitów danych + 3 wyzerowane bity wsp.
reszty
1011                    <--- 4-bitowy dzielnik CRC
01100011101110 000      <--- wynik operacji XOR
```

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

```
 1011
00111011101110 000
  1011
00010111101110 000
   1011
00000001101110 000
       1011
00000000110110 000
        1011
00000000011010 000
         1011
00000000001100 000
          1011
00000000000111 000
           101 1
00000000000010 100
            10 11
-----------------
00000000000000 010 <--- CRC
```

Weryfikacja sumy kontrolnej

```
11010011101110 010 <--- przesłany bez przekłamań ciąg 14 bitów danych + CRC
1011               <--- ustalony uprzednio, 4-bitowy dzielnik
01100011101110 010 <--- wynik operacji XOR
 1011
00111011101110 010
  1011
00010111101110 010
   1011
00000001101110 010
       1011
00000000110110 010
        1011
00000000011010 010
         1011
00000000001100 010
          1011
00000000000111 010
           101 1
00000000000010 110
            10 11
-----------------
00000000000000 000 <--- wynik operacji równy 0 oznacza poprawną transmisję
```

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Kod Hamminga

Istnieje kilka wariantów tego kodu. W laboratoriach zaprezentowany zostanie Kod Hamminga 7,4 w ujęciu macierzowym. Kod ten dla 4 bitów danych d_1, d_2, d_3, d_4 oblicza i dodaje 3 bity kontrolne p_1, p_2, p_3, które niosą informacje pozwalające na detekcję i korekcję pojedynczego błędu w bitach danych. Mamy więc do czynienia z kodem korekcyjnym (Error Correction Code).

Pierwszym krokiem jest zbudowanie **macierzy generującej**. W praktyce obliczeniowej skupimy się na macierzy nieuzupełnionej macierzą jednostkową. Dla 4 bitów danych i 3 bitów redundantnych ma ona wymiar 3x4. Budujemy ją kolumnami pamiętając o 2 zasadach:

1. kolumny nie mogą się powtarzać,
2. w każdej kolumnie muszą być przynajmniej 2 jedynki (waga Hamminga dla każdej kolumny musi być większa równa 2).

Kolejność kolumn nie ma znaczenia, ważne jest tylko aby używać tej samej macierzy dla wyznaczania sumy kontrolnej i jej weryfikacji.

Następnie do macierzy tej dołącza się macierz jednostkową. Końcowa postać przykładowej macierzy:

$$H_{3,7} = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Następnie macierz taką mnożymy przez wektor wiadomości uzupełniony o bity parzystości do wyliczenia. Mnożenie i dodawanie wykonujemy modulo 2 aby wyniki pozostały liczbami binarnymi 0 i 1.

Weryfikacja sumy kontrolnej odbywa się według prostej reguły: używając tej samej macierzy wyznaczana jest nowa suma kontrolna, a następnie otrzymany wektor sumowany jest z wektorem nowo wyliczonym (sumowanie modulo 2). Jeżeli wektor sumy jest równy 0, to wiadomość została przesłana bez przekłamań, a jeżeli nie, to odszukiwana jest kolumna, równa sumie i w niej jest błąd do poprawy. Kolumny z macierzy jednostkowej odpowiadają bitom sumy kontrolnej i te błędy nie są dla nas tak istotne.

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Funkcje skrótu

Kryptograficzne funkcje skrótu (hashujące) to "jednokierunkowe" funkcje matematyczne, które stanowią fundament podpisów cyfrowych. Podobnie jak sumy kontrolne CRC albo Hamminga mogą służyć do sprawdzenia integralności danych. Są również używane podczas autentyfikacji użytkownika w systemach komputerowych. Hasło użytkownika powinno być przekształcone funkcją skrótu zanim zostanie zapisane w bazie danych.

Funkcja skrótu H przyjmuje na wejściu wiadomość M o dowolnej długości i zwraca stały, krótki ciąg bitów h, nazywany skrótem (digest) lub "odciskiem palca" danych.

$$H(M) = h$$

W przeciwieństwie do kodu Hamminga, funkcje skrótu nie służą do naprawiania błędów, lecz do ich **bezwzględnego wykrywania**, nawet jeśli zostały wprowadzone przez inteligentnego atakującego. Dobra funkcja skrótu musi wykazywać efekt lawinowy: zmiana zaledwie jednego bitu w wiadomości wejściowej musi powodować drastyczną, nieprzewidywalną zmianę co najmniej połowy bitów w skrócie wynikowym. To sprawia, że atakujący nie może "zgadywać" brakujących części danych na podstawie podobieństwa skrótów.

Kluczowe aksjomaty (Właściwości)

Szybkość i łatwość obliczeń (Efficiency)

Obliczenie skrótu h dla dowolnej wiadomości M musi być wydajne i szybkie. W systemach bezpieczeństwa często haszujemy gigabajty danych, więc algorytm nie może być wąskim gardłem.

Odporność na wyznaczanie przeciwobrazu (Pre-image Resistance)

To tzw. **jednokierunkowość**. Mając dany skrót h, wyznaczenie oryginalnej wiadomości M musi być praktycznie niewykonalne (wymagać zbyt dużej mocy obliczeniowej).

Odporność na wyznaczanie drugiego przeciwobrazu (Second Pre-image Resistance)

Mając daną konkretną wiadomość M_1, atakujący nie może znaleźć innej wiadomości M_2 ($M_2 \neq M_1$), która dałaby dokładnie taki sam skrót h. Ta właściwość chroni przed podmianą pliku na inny o tym samym "odcisku palca".

Odporność na kolizje (Collision Resistance)

Znalezienie **jakiejkolwiek** pary dwóch różnych wiadomości M_1 i M_2, które dają ten sam skrót, musi być ekstremalnie trudne.

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Podpis wiadomości

Ten temat dobrze jest omawiać po kryptografii asymetrycznej znając już zasadę działania kryptosystemu RSA. Schemat podpisu cyfrowego w algorytmie RSA to moment, w którym łączymy wszystkie omówione wcześniej elementy: integralność (funkcje skrótu), poufność (szyfrowanie asymetryczne) oraz niezaprzeczalność.

Ten schemat opiera się na podejściu, w którym **nie podpisujemy samej treści wiadomości**, lecz jej skrót. Jest to szybsze i bezpieczniejsze.

Przygotowanie: Klucze i Funkcja Skrótu

Nadawca (Alicja) posiada **parę kluczy RSA**:

- **Klucz prywatny** (d, n)**:** Trzymany w ścisłej tajemnicy, służy do składania podpisu,
- **Klucz publiczny** (e, n)**:** Dostępny dla każdego, służy do weryfikacji podpisu.

Obie strony uzgadniają również wspólną kryptograficzną **funkcję skrótu** H (np. SHA-256).

Proces tworzenia podpisu (po stronie Nadawcy)

Alicja chce wysłać wiadomość M i podpisać ją cyfrowo:

1. **Haszowanie:** Alicja oblicza skrót wiadomości: $h = H(M)$.
2. **Podpisywanie (Szyfrowanie kluczem prywatnym):** Alicja "szyfruje" skrót h swoim kluczem prywatnym d. Wynikiem jest podpis S:
$$S = h^d (\mathrm{mod} n)$$
3. **Wysyłka:** Alicja wysyła do odbiorcy (Bolka) parę: (M, S).

Proces weryfikacji (Po stronie Odbiorcy)

Bolek otrzymuje wiadomość M' oraz podpis S' (używamy primów, bo dane mogły zostać zmienione w trakcie transmisji):

1. **Obliczenie własnego skrótu:** Bolek haszuje otrzymaną wiadomość M' tą samą funkcją: $h_{odb} = H(M')$.
2. **Odszyfrowanie podpisu (Kluczem publicznym):** Bolek używa klucza publicznego Alicji (e, n), aby odczytać skrót zawarty w podpisie:

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

$$h_{podp} = (S')^e (\mathrm{mod} n)$$

3. **Porównanie:** Bolek sprawdza, czy obliczony przez niego skrót zgadza się z tym z podpisu:

$$h_{odb} \stackrel{?}{=} h_{podp}$$

Jeżeli tak, to uznaje Alicję za autorkę wiadomości.

Ćwiczenia samodzielne

Zadanie 2.1

Policz sumę kontrolną CRC dla wiadomości będącej twoim rokiem urodzenia używając jako dzielnika d wielomianu stopnia 4. Sprawdź tę sumę kiedy nie było błędów w bitach oraz w sytuacji, kiedy bity zostały zmienione (zmień losowo jeden albo dwa bity). Czy udało się znaleźć błąd polegający na pozytywnej weryfikacji błędnych danych?

Napisz program wylicza sumę kontrolną CRC, w programie użytkownik może określić długość dzielnika, wylosować dzielnik o podanej długości lub podać go samemu. Użytkownik powinien mieć możliwość wpisania ciągu bitów do policzenia sumy. Miejsce na sumę kontrolną CRC powinno ustawiać się samo w zależności od długości dzielnika i uzupełniać zerami, przy czym użytkownik powinien mieć możliwość podania własnej wartości, dzięki czemu program będzie służył też weryfikacji sumy CRC. Gotowy program należy przesłać prowadzącemu.

Zadanie 2.2

Przedstaw działanie kodu Hamminga (generowanie macierzy, wyznaczanie bitów kontrolnych, sprawdzenie w sytuacji, kiedy nie było błędów, w sytuacji z jednym błędem na bitach danych, w sytuacji z jednym błędem na bicie parzystości oraz dla dwóch błędów) dla wiadomości:

a) m1 = 1100

b) m2 = 1011

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Zadanie 2.3

Niech będzie dana funkcja H, która dla dowolnego ciągu znaków zwraca średnią arytmetyczną z kodów ASCII 3 początkowych znaków zaokrąglaną do liczb całkowitych (krótsze wiadomości uzupełnij spacjami). Które aksjomaty funkcji skrótu spełnia funkcja H, a których nie?

Zadanie 2.4

Wygeneruj klucze RSA tak, aby n pozwalało na szyfrowanie i deszyfrowanie liter zapisanych za pomocą tabelki ASCII.

Zadanie 2.5

Użyj kluczy z zadania 1.4 i funkcji z zadania 1.3 (jako dydaktyczną wersję funkcji skrótu) do podpisania wiadomości „Witaj świecie". Zweryfikuj tożsamość nadawcy.

Pytania pomocnicze

1. Czym jest kodowanie redundantne?
2. Jak działa algorytm CRC?
3. Jak działa kod korekcyjny Hamminga 7,4?
4. Jakie są aksjomaty funkcji hashującej?
5. Jak wygląda schemat podpisu wiadomości z wykorzystaniem skrótu wiadomości?

Podsumowanie

Podsumowując dzisiejsze zajęcia, przeszliśmy pełną ścieżkę ewolucji mechanizmów kontrolnych – od czysto technicznej detekcji błędów transmisji po zaawansowane gwarancje bezpieczeństwa informacji. Zrozumienie różnicy między **bitem parzystości** czy **kodem Hamminga**, które chronią dane przed

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

przypadkowym "szumem" natury, a **kryptograficznymi funkcjami skrótu**, które stawiają opór celowej manipulacji, jest kluczowe dla każdego inżyniera cyberbezpieczeństwa. Widzimy teraz wyraźnie, że integralność w triadzie CIA to nie tylko brak przekłamań w bitach, ale przede wszystkim matematyczna pewność, że informacja dotarła do nas w formie nienaruszonej i autentycznej.

Najważniejszym wnioskiem z drugiej części laboratorium jest synergia między algorytmem **RSA** a funkcjami skrótu, która daje początek **podpisowi cyfrowemu**. Dzięki odwróceniu logiki kluczy asymetrycznych zyskaliśmy narzędzie realizujące czwarty, niezwykle istotny filar bezpieczeństwa: **niezaprzeczalność**. Potraficie już nie tylko zweryfikować, czy plik nie został zmodyfikowany, ale także dowieść ponad wszelką wątpliwość, kto jest jego autorem. Ta wiedza stanowi fundament, na którym budowane są współczesne systemy bankowości elektronicznej, e-obywatela oraz bezpiecznej komunikacji w Internecie.

Literatura

1. D. R. Stinson, M. B. Paterson, Kryptografia. W teorii i w praktyce, PWN 2021
2. A. J. Menezes , P. C. Oorschot , S. A. Vanstone, Kryptografia stosowana, WNT 2005

Fundusze Europejskie dla Rozwoju Społecznego

Rzeczpospolita Polska

Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Materiały do przedmiotu:

Podstawy kryptografii

Laboratorium nr 3

Szyfrowanie Cezara: Implementacja prostego algorytmu szyfrowania Cezara, stworzenie programu, który przesuwa litery w alfabecie o określną liczbę miejsc

Autor:

dr Michał Dolecki

Fundusze Europejskie dla Rozwoju Społecznego

Rzeczpospolita Polska

Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Spis treści

Fundusze Europejskie dla Rozwoju Społecznego

Rzeczpospolita Polska

Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Wprowadzenie, tematyka zajęć

Jednym z podstawowych szyfrów, który warto poznać jest szyfr Cezara. Jest on podstawowym szyfrem podstawieniowym. Był znany już w starożytności, ale jego wykorzystanie przez Juliusza Cezara sprawiło, że przyjął właśnie tę nazwę. Zasada działania polega na przesunięciu każdej litery o stałą liczbę pozycji w prawo lub w lewo w alfabecie. Najpopularniejsza w literaturze jest wersja z przesunięciem do przodu o trzy pozycje, chociaż historycznie Juliusz Cezar często korzystał też z przesunięcia o –1 (czyli o jedną pozycję w lewo, co odpowiada przesunięciu w prawo o 25).

Na przykładzie tego klasycznego szyfru łatwo jest pokazać w praktyce czym jest funkcja szyfrująca, deszyfrująca oraz jakie znaczenie ma klucz kryptograficzny. Poza zagadnieniami związanymi z szyfrem podstawieniowym, zaprezentowana zostanie również druga rodzina szyfrów – szyfry przestawieniowe.

Cel laboratorium

Student po realizacji zajęć potrafi samodzielnie zaimplementować oraz przeanalizować działanie klasycznego szyfru Cezara jako najprostszego przykładu kryptografii symetrycznej, wskazując jego słabości wynikające z małej przestrzeni klucza i podatności na atak brutalnej siły oraz analizę frekwencyjną; student umie precyzyjnie zdefiniować i stosować podstawowe pojęcia kryptograficzne – tekst jawny (*plaintext*), szyfrogram (*ciphertext*), funkcję szyfrującą, funkcję deszyfrującą oraz klucz kryptograficzny – rozróżniając rolę klucza w przekształcaniu tekstu jawnego w szyfrogram i odwrotnie; wreszcie student potrafi scharakteryzować oraz odróżnić od siebie dwie fundamentalne klasy szyfrów klasycznych – szyfry podstawieniowe (*substitution ciphers*), w których znaki tekstu jawnego zastępowane są innymi znakami według ustalonego klucza, oraz szyfry przestawieniowe (*transposition ciphers*), w których zmienia się jedynie kolejność znaków – i wskazać, w jaki sposób łączenie obu technik (wieloetapowe szyfrowanie) prowadzi do zwiększenia bezpieczeństwa systemu kryptograficznego.

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Instrukcja laboratoryjna

Szyfr Cezara

Szyfr podstawieniowy to metoda szyfrowania, w której każdy znak tekstu jawnego (niezaszyfrowanego) jest zastępowany innym znakiem według ustalonej reguły. W tego typu szyfrach zachowana jest kolejność znaków, zmienia się jedynie ich postać.

Szyfr Cezara jest najprostszym przykładem szyfru podstawieniowego. Działa poprzez przesunięcie każdej litery alfabetu o stałą liczbę pozycji – najczęściej w prawo. Na przykład przy przesunięciu o 3 pozycje litera A zostaje zaszyfrowana jako D, B jako E, a Z jako C (po osiągnięciu końca alfabetu wracamy do jego początku).

Funkcja szyfrująca dla szyfru Cezara może być zdefiniowana następująco:

$$E_k(x) = (x + k) \mathrm{mod}\ n,$$

gdzie x oznacza pozycję litery w alfabecie (np. A=0, B=1, ..., Z=25), k to klucz określający przesunięcie, a n to liczba liter w alfabecie.

Funkcja deszyfrująca jest działaniem odwrotnym:

$$D_k(y) = (y - k) \mathrm{mod}\ n,$$

gdzie y to pozycja litery w tekście zaszyfrowanym, k to klucz określający przesunięcie, a n to liczba liter w alfabecie.

Klucz kryptograficzny w szyfrze Cezara to liczba całkowita określająca wartość przesunięcia. W klasycznej wersji używanej przez Juliusza Cezara klucz wynosił 3, choć historycznie stosowano również inne wartości (np. przesunięcie o 1 w lewo, czyli klucz równy -1 lub 25). Ze względu na niewielką liczbę możliwych kluczy (dla alfabetu łacińskiego tylko 25) szyfr ten jest bardzo łatwy do złamania poprzez wypróbowanie wszystkich kombinacji. W bardziej wyrafinowany sposób można atakować ten szyfr analizując częstotliwość występowania liter w alfabecie użytym do stworzenia wiadomości. Na podstawie tych analiz można odgadnąć wartość przesunięcia – klucz kryptograficzny.

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Przykład

Zaszyfrujmy kłowo KRYPTOGRAFIA szyfrem Cezara.

Przyjmujemy alfabet łaciński bez polskich znaków i numerujemy litery:

$$A = 0, B = 1, \ \ldots, Z = 25$$

Szyfrowanie definiujemy jako przekształcenie:

$$E_k(x) = (x + k) mod 26$$

Słowo: **KRYPTOLOGIA**

Klucz: $k = 3$

Zamiana liter na liczby

Litera	**K**	**R**	**Y**	**P**	**T**	**O**	**L**	**O**	**G**	**I**	**A**
Numer	10	17	24	15	19	14	11	14	6	8	0

Obliczenia:

$$\begin{aligned}
10 + 3 &= 13 \\
17 + 3 &= 20 \\
24 + 3 &= 27 \equiv 1 (mod 26) \\
15 + 3 &= 18 \\
19 + 3 &= 22 \\
14 + 3 &= 17 \\
11 + 3 &= 14 \\
14 + 3 &= 17 \\
6 + 3 &= 9 \\
8 + 3 &= 11 \\
0 + 3 &= 3
\end{aligned}$$

Powrót do liter według obliczonych wartości

Numer	**13**	**20**	**1**	**18**	**22**	**17**	**14**	**17**	**9**	**11**	**3**
Litera	N	U	B	S	W	R	O	R	J	L	D

Ostateczna postać szyfrogramu:

KRYPTOLOGIA → NUBSWRORJLD

Deszyfrowanie przebiega w analogiczny sposób stosując przesunięcie o 3 pozycje w lewo.

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Szyfr przestawieniowy

Szyfr przestawieniowy (zwany również szyfrem transpozycyjnym) to metoda szyfrowania, w której zmienia się kolejność znaków w tekście jawnym, ale same znaki pozostają niezmienione. Innymi słowy, litery nie są zastępowane innymi (jak w szyfrach podstawieniowych), lecz jedynie przestawiane według ustalonego schematu. Przykładem może być zapisanie tekstu w tabeli wierszami, a następnie odczytanie go kolumnami.

Permutacja to w matematyce dowolne ustawienie elementów pewnego zbioru w określonej kolejności. W kontekście kryptografii permutację rozumiemy jako funkcję odwzorowującą zbiór pozycji na same siebie w sposób wzajemnie jednoznaczny. Formalnie permutację zbioru $\{1, 2, \ldots, n\}$ możemy zapisać jako:

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & \cdots & n \\ \sigma(1) & \sigma(2) & \sigma(3) & \cdots & \sigma(n) \end{pmatrix}$$

co oznacza, że element znajdujący się pierwotnie na pozycji i zostaje przeniesiony na pozycję $\sigma(1)$.

Permutacja odwrotna σ^{-1} to taka permutacja, która odwraca działanie permutacji σ. Oznacza to, że jeśli σ przenosi element z pozycji i na pozycję j, to σ^{-1} przenosi element z pozycji j z powrotem na pozycję i. Formalnie:

$$\sigma^{-1}(\sigma(i)) = i \text{ dla wszystkich } i$$

Szyfrowanie z użyciem permutacji polega na zastosowaniu z góry ustalonej permutacji σ do zmiany kolejności znaków w tekście jawnym. Jeśli tekst jawny zapiszemy jako ciąg znaków x_1, x_2, ..., x_n, to tekst zaszyfrowany y_1, y_2, ..., y_n powstaje według reguły:

$$y_{\sigma(i)} = x_i \text{ lub rownowaznie } y_i = x_{\sigma^{-1}(i)}$$

Innymi słowy, znak znajdujący się pierwotnie na pozycji i trafia w miejscu $\sigma(i)$ w szyfrogramie.

Deszyfrowanie wymaga zastosowania permutacji odwrotnej σ^{-1}. Aby odzyskać tekst jawny z szyfrogramu y_1, y_2, ..., y_n, wykonujemy:

$$x_i = y_{\sigma(i)} \text{ lub rownowaznie } x_{\sigma^{-1}(i)} = y_i$$

Przykład

Załóżmy, że mamy tekst jawny: "KRYPTOLOGIA" i permutację dla 5-elementowych bloków:

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 5 & 1 & 4 & 2 \end{pmatrix}$$

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Dzielimy tekst na bloki po 5 znaków:

- Blok 1: K R Y P T
- Blok 2: O L O G I
- Blok 3: A (uzupełniamy np. spacją)

Stosując permutację σ do pierwszego bloku:

- Znak z pozycji 1 (K) trafia na pozycję 3
- Znak z pozycji 2 (R) trafia na pozycję 5
- Znak z pozycji 3 (Y) trafia na pozycję 1
- Znak z pozycji 4 (P) pozostaje na pozycji 4
- Znak z pozycji 5 (T) trafia na pozycję 2

Otrzymujemy: Y T K P R

Analogicznie szyfrujemy pozostałe bloki. Do deszyfrowania użylibyśmy permutacji odwrotnej σ^{-1}, która dla podanego przykładu ma postać:

$$\sigma^{-1} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 5 & 1 & 4 & 2 \end{pmatrix}^{-1} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 5 & 1 & 4 & 2 \end{pmatrix}$$

Jak widać ta permutacja odwrotna jest identyczna z początkową permutacją, co oznacza, że jest inwolucją.

Szyfry przestawieniowe często łączy się z szyframi podstawieniowymi, tworząc bardziej złożone i nieco bezpieczniejsze systemy kryptograficzne, chociaż przy obecnych możliwościach komputerów tego typu szyfry nie stanowią żadnego problemu przy kryptoanalizie. Niemniej dla ogólnej wiedzy kryptograficznej należy je poznać.

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Ćwiczenia samodzielne

Zadanie 3.1

Zaszyfruj swoje imię Szyfrem Cezara z kluczem $k = -1$. Odszyfruj szyfrogram.

Zadanie 3.2

Zaszyfruj swoje nazwisko szyfrem przestwieniowym stosując permutację inną niż w przykładzie.

Zadanie 3.3

Napisz program umożliwiający szyfrowanie i deszyfrowanie Szyfrem Cezara z możliwością podania różnych wartości klucza.

Zadanie 3.4

Napisz program, w którym użytkownik może określić długość bloku, a wyliczona zostanie losowa permutacja i permutacja odwrotna do wylosowanej.

Gotowy program z zadania 1.4 proszę przesłać prowadzącemu jako podsumowanie pracy z laboratorium 3.

Zadanie 3.5

Napisz program który pozwala na zaszyfrowanie i odszyfrowanie wiadomości stosując szyfr przestawieniowy. Permutacja może być wylosowana w programie albo podana przez użytkownika.

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Pytania pomocnicze

1. Na czym polega szyfrowanie Szyfrem Cezara?
2. Jak rozumiesz pojęcia tekst jawny, szyfrogram, funkcja szyfrująca, funkcja deszyfrująca oraz klucz kryptograficzny?
3. Czym jest szyfr przestawieniowy?

Podsumowanie

W trakcie dzisiejszych zajęć poznaliśmy dwa fundamentalne algorytmy kryptograficzne – **szyfr Cezara** (należący do grupy szyfrów podstawieniowych) oraz **szyfr przestawieniowy** (transpozycyjny). Choć obie metody z dzisiejszej perspektywy są wyjątkowo łatwe do złamania – szyfr Cezara oferuje zaledwie 25 możliwych kluczy dla alfabetu łacińskiego, a szyfr przestawieniowy również nie stanowi wyzwania dla współczesnych metod kryptoanalizy – to ich poznanie ma istotną wartość edukacyjną.

Na tych prostych przykładach mogliśmy zaobserwować uniwersalne mechanizmy, które legły u podstaw wszystkich późniejszych, bardziej zaawansowanych systemów kryptograficznych. Wprowadzone pojęcia – **klucz kryptograficzny**, **funkcja szyfrująca**, **funkcja deszyfrująca**, a w przypadku szyfrów przestawieniowych także **permutacja** i **permutacja odwrotna** – stanowią fundament, na którym opiera się współczesna kryptografia.

Co więcej, analizując te historyczne metody, możemy lepiej zrozumieć, dlaczego współczesne systemy muszą być tak złożone. Szyfr Cezara jest podatny na atak brute-force (wypróbowanie wszystkich możliwych kluczy), a szyfr przestawieniowy – na analizę częstości występowania liter i wzorców. Te słabości doskonale ilustrują, jakimi cechami musi charakteryzować się bezpieczny algorytm: ogromną przestrzenią kluczy, odpornością na analizę statystyczną oraz złożonością obliczeniową uniemożliwiającą praktyczne złamanie.

Warto znać te podstawowe szyfry nie tylko ze względów historycznych, ale przede wszystkim dlatego, że w przystępny sposób wprowadzają w arkana myślenia kryptograficznego. Stanowią one pierwszy krok do zrozumienia, jak działają współczesne systemy zabezpieczeń – od prostego szyfrowania wiadomości po

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

zaawansowane protokoły kryptograficzne chroniące dzisiejszą komunikację cyfrową. Świadomość ich ograniczeń uczy też zdrowego sceptycyzmu i krytycznego podejścia do wszelkich rozwiązań, które deklarują "niemożliwość złamania".

Literatura

1. D. R. Stinson, M. B. Paterson, Kryptografia. W teorii i w praktyce, PWN 2021
2. A. J. Menezes , P. C. Oorschot , S. A. Vanstone, Kryptografia stosowana, WNT 2005

Fundusze Europejskie dla Rozwoju Społecznego

Rzeczpospolita Polska

Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Materiały do przedmiotu:

Podstawy kryptografii

Laboratorium nr 4

Zarządzanie kluczami w praktyce

Autor:

dr Michał Dolecki

Fundusze Europejskie dla Rozwoju Społecznego

Rzeczpospolita Polska

Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Spis treści

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Wprowadzenie, tematyka zajęć

Współczesna kryptografia symetryczna i asymetryczna opiera się na fundamentalnym założeniu, że bezpieczeństwo systemu zależy od zachowania tajności kluczy, a nie od utrzymywania w sekrecie samych algorytmów szyfrowania. W praktyce inżynierskiej oznacza to, że klucze kryptograficzne muszą być nie tylko generowane w sposób losowy i nieprzewidywalny, ale również odpowiednio przechowywane, dystrybuowane, rotowane i ostatecznie unieważniane. Nawet najsilniejszy algorytm szyfrowania, taki jak AES czy RSA, staje się bezużyteczny, jeśli klucz zostanie skompromitowany, zapisany w niebezpiecznym miejscu lub przypadkowo usunięty. Dlatego zarządzanie kluczami (ang. key management) stanowi jeden z najważniejszych aspektów praktycznej kryptografii, często bardziej krytyczny niż sam dobór algorytmów. W rzeczywistych systemach informatycznych klucze są przechowywane w dedykowanych sprzętowych modułach bezpieczeństwa (HSM), w zabezpieczonych plikach konfiguracyjnych, w certyfikatach X.509, a w najprostszych zastosowaniach – w plikach chronionych odpowiednimi uprawnieniami dostępu.

Platforma .NET oferuje bogaty zestaw klas w przestrzeni nazw System.Security.Cryptography, które w znaczący sposób upraszczają implementację mechanizmów zarządzania kluczami. Dla kryptografii symetrycznej (np. AES) klucz jest zazwyczaj liczbą binarną o określonej długości, którą można wygenerować za pomocą generatora liczb losowych (RandomNumberGenerator), zapisać do pliku w postaci bajtów, a następnie odczytać i wykorzystać do szyfrowania oraz deszyfrowania danych. Dla kryptografii asymetrycznej (np. RSA) sytuacja jest bardziej złożona – operujemy na parach kluczy: kluczu publicznym przeznaczonym do dystrybucji i kluczu prywatnym, który musi pozostać poufny. W .NET klasy takie jak RSA udostępniają metody ExportRSAPrivateKey() i ExportRSAPublicKey() do eksportu kluczy w formacie binarnym PKCS#1 lub PKCS#8, a także ImportRSAPrivateKey() i ImportRSAPublicKey() do ich importowania. Dodatkowo platforma wspiera format PEM (np. ExportRSAPrivateKeyPem()), który jest powszechnie stosowany w systemach Unixowych i ułatwia wymianę kluczy między różnymi platformami.

Podczas dzisiejszych zajęć laboratoryjnych przejdziemy przez pełen cykl życia kluczy kryptograficznych w praktyce. Nauczymy się generować klucze

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

symetryczne AES i asymetryczne RSA z wykorzystaniem standardowych bibliotek C#, zapisywać je do plików w formacie binarnym oraz PEM, a następnie odczytywać je w niezależnych procesach w celu szyfrowania i deszyfrowania danych. Szczególną uwagę poświęcimy praktycznym aspektom wymiany kluczy. Celem ćwiczeń jest nie tylko nabycie umiejętności technicznych, ale przede wszystkim zrozumienie, że samo szyfrowanie to dopiero początek – prawdziwe bezpieczeństwo wymaga starannego zarządzania całym ekosystemem kluczy kryptograficznych.

Cel laboratorium

Celem zajęć jest wykształcenie u studentów praktycznych umiejętności samodzielnego zarządzania kluczami kryptograficznymi w środowisku programistycznym C# z wykorzystaniem standardowych bibliotek platformy .NET. Po ukończeniu laboratorium student potrafi wygenerować klucze symetryczne AES oraz pary kluczy asymetrycznych RSA, zapisać je w formacie binarnym i PEM do plików, a następnie odczytać je w niezależnych procesach w celu przeprowadzenia operacji szyfrowania i deszyfrowania danych. Student zdobywa również świadomość praktycznych wyzwań związanych z bezpiecznym przechowywaniem i dystrybucją kluczy, w tym zagrożeń wynikających z przesyłania kluczy pocztą elektroniczną czy przechowywania ich na nośnikach zewnętrznych, co przygotowuje go do świadomego projektowania mechanizmów bezpieczeństwa w rzeczywistych systemach informatycznych.

Instrukcja laboratoryjna

Zarządzanie kluczami AES w C#

Advanced Encryption Standard (AES) jest symetrycznym algorytmem blokowym, który stał się standardem szyfrowania w zastosowaniach rządowych i komercyjnych na całym świecie. W kryptografii symetrycznej ten sam klucz

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

używany jest zarówno do szyfrowania, jak i do deszyfrowania danych, co czyni klucz najcenniejszym elementem całego systemu – jego kompromitacja oznacza kompromitację wszystkich zaszyfrowanych nim danych. W platformie .NET implementację AES zapewnia klasa Aes w przestrzeni nazw System.Security.Cryptography, która oferuje wygodne metody generowania losowych kluczy i wektorów inicjujących (IV). Klucz AES może mieć długość 128, 192 lub 256 bitów (odpowiednio 16, 24 lub 32 bajty), przy czym dłuższy klucz zapewnia wyższy poziom bezpieczeństwa kosztem nieznacznie większego obciążenia obliczeniowego. Wektor inicjujący (IV) jest dodatkową wartością losową, która zapewnia, że szyfrowanie tych samych danych tym samym kluczem da za każdym razem inny rezultat, uniemożliwiając ataki oparte na analizie wzorców.

```
using System;
using System.Security.Cryptography;
using System.Text;
using System.Linq;

class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("=== GENEROWANIE KLUCZA AES ===\n");

        // Tworzymy nową instancję klasy AES z domyślnymi parametrami
        // Domyślnie: klucz 256 bitów (32 bajty), tryb CBC, padding PKCS7
        using (Aes aes = Aes.Create())
        {
            // Generowanie klucza i IV następuje automatycznie przy
tworzeniu obiektu
            // Możemy jednak wymusić konkretny rozmiar klucza
            aes.KeySize = 256; // 128, 192 lub 256 bitów

            Console.WriteLine($"Algorytm:
{aes.ToString().Split('.').Last()}");
            Console.WriteLine($"Rozmiar klucza: {aes.KeySize} bitów
({aes.Key.Length} bajtów)");
            Console.WriteLine($"Rozmiar bloku: {aes.BlockSize} bitów");
            Console.WriteLine($"Tryb: {aes.Mode}");
            Console.WriteLine($"Padding: {aes.Padding}\n");

            // Pobieramy wygenerowany klucz i IV
            byte[] key = aes.Key;
```

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

```
            byte[] iv = aes.IV;

            // Wyświetlenie klucza w różnych formatach

            // Format 1: Bajty szesnastkowo (hex) - najczęściej używany w
programowaniu
            Console.WriteLine("Klucz (hex):");
            Console.WriteLine(BitConverter.ToString(key).Replace("-", "
"));
            Console.WriteLine();

            // Format 2: Base64 - wygodny do przechowywania w plikach
tekstowych i przesyłania
            Console.WriteLine("Klucz (Base64):");
            Console.WriteLine(Convert.ToBase64String(key));
            Console.WriteLine();

            // Format 3: Tablica bajtów w składni C# - przydatne do
wklejenia do kodu
            Console.WriteLine("Klucz (tablica bajtów C#):");
            Console.WriteLine("byte[] key = new byte[] { " + string.Join(",
", key) + " };");
            Console.WriteLine();

            // Analogicznie dla IV
            Console.WriteLine("Wektor inicjujący (IV) - hex:");
            Console.WriteLine(BitConverter.ToString(iv).Replace("-", " "));
            Console.WriteLine();

            Console.WriteLine("IV (Base64):");
            Console.WriteLine(Convert.ToBase64String(iv));
            Console.WriteLine();

            // Prosta wizualizacja losowości - wyświetlenie pierwszych 16
bajtów jako tekst
            Console.WriteLine("Klucz jako tekst (tylko dla ilustracji -
zwykle nieczytelny):");
            string keyAsText = Encoding.ASCII.GetString(key).Replace("\0",
"\\0");
            Console.WriteLine(keyAsText);
        }

        Console.WriteLine("\nNaciśnij Enter, aby zakończyć...");
        Console.ReadLine();
    }
}
```


Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Klasa Aes.Create() zwraca nową instancję implementacji AES.
Właściwości Key i IV są automatycznie wypełniane losowymi wartościami przy tworzeniu obiektu. Format hex (szesnastkowy) jest powszechnie używany w dokumentacji i narzędziach debugujących, natomiast Base64 jest bardziej kompaktowy i często stosowany w plikach konfiguracyjnych oraz przy przesyłaniu kluczy przez sieć.

Zapis klucza AES do pliku w różnych formatach

```
using System;
using System.IO;
using System.Security.Cryptography;
using System.Text;

class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("=== ZAPIS KLUCZA AES DO PLIKU ===\n");

        using (Aes aes = Aes.Create())
        {
            aes.KeySize = 256;
            byte[] key = aes.Key;
            byte[] iv = aes.IV;

            Console.WriteLine("Wygenerowano nowy klucz AES-256");
            Console.WriteLine($"Klucz (hex):
{BitConverter.ToString(key).Replace("-", "")}");
            Console.WriteLine($"IV (hex):
{BitConverter.ToString(iv).Replace("-", "")}\n");

            // Format 1: Zapis binarny (surowy)
            string fileBinaryKey = "aes_key.bin";
            string fileBinaryIV = "aes_iv.bin";

            File.WriteAllBytes(fileBinaryKey, key);
            File.WriteAllBytes(fileBinaryIV, iv);
```

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

```
            Console.WriteLine($"Zapisano klucz binarnie: {fileBinaryKey}
({new FileInfo(fileBinaryKey).Length} bajtów)");
            Console.WriteLine($"Zapisano IV binarnie: {fileBinaryIV} ({new
FileInfo(fileBinaryIV).Length} bajtów)");

            // Format 2: Zapis jako tekst hex
            string fileHexKey = "aes_key_hex.txt";
            string fileHexIV = "aes_iv_hex.txt";

            string keyHex = BitConverter.ToString(key).Replace("-", "");
            string ivHex = BitConverter.ToString(iv).Replace("-", "");

            File.WriteAllText(fileHexKey, keyHex);
            File.WriteAllText(fileHexIV, ivHex);

            Console.WriteLine($"Zapisano klucz hex: {fileHexKey}");
            Console.WriteLine($"Zapisano IV hex: {fileHexIV}");

            // Format 3: Zapis jako Base64
            string fileBase64Key = "aes_key_base64.txt";
            string fileBase64IV = "aes_iv_base64.txt";

            string keyBase64 = Convert.ToBase64String(key);
            string ivBase64 = Convert.ToBase64String(iv);

            File.WriteAllText(fileBase64Key, keyBase64);
            File.WriteAllText(fileBase64IV, ivBase64);

            Console.WriteLine($"Zapisano klucz Base64: {fileBase64Key}");
            Console.WriteLine($"Zapisano IV Base64: {fileBase64IV}");

            // Format 4: Zapis jako plik PEM (dla AES nie jest standardem,
ale pokazujemy koncepcję)
            string filePemKey = "aes_key.pem";
            string filePemIV = "aes_iv.pem";

            // PEM dla klucza symetrycznego to nieoficjalnie Base64 z
nagłówkiem
            string pemKey = $"-----BEGIN AES KEY-----\n{keyBase64}\n-----
END AES KEY-----";
            string pemIV = $"-----BEGIN AES IV-----\n{ivBase64}\n-----END
AES IV-----";

            File.WriteAllText(filePemKey, pemKey);
            File.WriteAllText(filePemIV, pemIV);
```


Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

```
            Console.WriteLine($"Zapisano klucz w formacie PEM-like:
{filePemKey}");
            Console.WriteLine($"Zapisano IV w formacie PEM-like:
{filePemIV}");

            // Wyświetlenie zawartości pierwszego pliku dla weryfikacji
            Console.WriteLine("\nZawartość pliku Base64 (pierwsze 50
znaków):");
            string content = File.ReadAllText(fileBase64Key);
            Console.WriteLine(content.Substring(0, Math.Min(50,
content.Length)) + "...");
        }

        Console.WriteLine("\nNaciśnij Enter, aby kontynuować...");
        Console.ReadLine();

        // Wyświetlenie listy utworzonych plików
        Console.WriteLine("\nUtworzone pliki w katalogu aplikacji:");
        foreach (string file in Directory.GetFiles(".", "aes_*"))
        {
            FileInfo fi = new FileInfo(file);
            Console.WriteLine($"{fi.Name,-20} {fi.Length,8} bajtów");
        }

        Console.WriteLine("\nNaciśnij Enter, aby zakończyć...");
        Console.ReadLine();
    }
}
```

Zapis binarny (File.WriteAllBytes) jest najbardziej kompaktowy i wydajny, ale plik jest nieczytelny dla człowieka. Format hex jest czytelny i łatwy do ręcznego skopiowania, ale zajmuje dwa razy więcej miejsca (jeden bajt zapisany jako dwa znaki hex). Base64 jest kompromisem – zajmuje około 33% więcej miejsca niż format binarny, ale jest tekstowy i łatwy do przesyłania. Format PEM (Privacy-Enhanced Mail) jest standardem dla kluczy asymetrycznych, ale dla AES nie ma oficjalnej specyfikacji – w przykładzie pokazujemy jedynie konwencję z dodaniem nagłówków.

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Odczytywanie klucza AES z plików w różnych formatach

```
using System;
using System.IO;
using System.Security.Cryptography;
using System.Text;
using System.Text.RegularExpressions;

class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("=== ODCZYT KLUCZA AES Z PLIKÓW ===\n");

        // Najpierw generujemy i zapisujemy klucze testowe
        byte[] originalKey;
        byte[] originalIV;

        using (Aes aes = Aes.Create())
        {
            aes.KeySize = 256;
            originalKey = aes.Key;
            originalIV = aes.IV;

            // Zapis w różnych formatach
            File.WriteAllBytes("test_key.bin", originalKey);
            File.WriteAllText("test_key_hex.txt",
BitConverter.ToString(originalKey).Replace("-", ""));
            File.WriteAllText("test_key_base64.txt",
Convert.ToBase64String(originalKey));
            File.WriteAllText("test_key_pem.txt", $"-----BEGIN AES KEY-----
\n{Convert.ToBase64String(originalKey)}\n-----END AES KEY-----");

            Console.WriteLine("Utworzono pliki testowe z kluczem.\n");
        }

        // Odczyt z pliku binarnego
        Console.WriteLine("--- Odczyt z pliku binarnego ---");
        byte[] keyFromBin = File.ReadAllBytes("test_key.bin");

        Console.WriteLine($"Odczytano {keyFromBin.Length} bajtów");
        Console.WriteLine($"Klucz (hex):
{BitConverter.ToString(keyFromBin).Replace("-", " ")}");
```

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

```
        Console.WriteLine($"Czy zgadza się z oryginałem?
{CompareByteArrays(originalKey, keyFromBin)}\n");

        // Odczyt z pliku hex
        Console.WriteLine("--- Odczyt z pliku hex ---");
        string hexContent = File.ReadAllText("test_key_hex.txt").Trim();
        byte[] keyFromHex = HexStringToByteArray(hexContent);

        Console.WriteLine($"Odczytano tekst hex: {hexContent.Substring(0,
32)}...");
        Console.WriteLine($"Konwertowano na {keyFromHex.Length} bajtów");
        Console.WriteLine($"Czy zgadza się z oryginałem?
{CompareByteArrays(originalKey, keyFromHex)}\n");

        // Odczyt z pliku Base64
        Console.WriteLine("--- Odczyt z pliku Base64 ---");
        string base64Content =
File.ReadAllText("test_key_base64.txt").Trim();
        byte[] keyFromBase64 = Convert.FromBase64String(base64Content);

        Console.WriteLine($"Odczytano tekst Base64:
{base64Content.Substring(0, 20)}...");
        Console.WriteLine($"Konwertowano na {keyFromBase64.Length}
bajtów");
        Console.WriteLine($"Czy zgadza się z oryginałem?
{CompareByteArrays(originalKey, keyFromBase64)}\n");

        // Odczyt z pliku PEM (wyciągnięcie Base64 spomiędzy nagłówków)
        Console.WriteLine("--- Odczyt z pliku PEM ---");
        string pemContent = File.ReadAllText("test_key_pem.txt");
        string extractedBase64 = ExtractBase64FromPem(pemContent);
        byte[] keyFromPem = Convert.FromBase64String(extractedBase64);

        Console.WriteLine($"Odczytano plik PEM");
        Console.WriteLine($"Wyciągnięto Base64:
{extractedBase64.Substring(0, 20)}...");
        Console.WriteLine($"Konwertowano na {keyFromPem.Length} bajtów");
        Console.WriteLine($"Czy zgadza się z oryginałem?
{CompareByteArrays(originalKey, keyFromPem)}\n");

        // Tworzenie obiektu AES z odczytanego klucza
        Console.WriteLine("--- Tworzenie instancji AES z odczytanego klucza
---");

        using (Aes aes = Aes.Create())
        {
```


Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

```
            // Ustawiamy klucz i IV (IV też musimy skądś wziąć - tu dla
uproszczenia generujemy nowy)
            aes.Key = keyFromBase64; // Używamy klucza odczytanego z Base64
            aes.GenerateIV(); // Generujemy nowy IV (w praktyce IV też
byśmy odczytali z pliku)

            Console.WriteLine("Utworzono obiekt AES z odczytanym
kluczem:");
            Console.WriteLine($"Rozmiar klucza: {aes.KeySize} bitów");
            Console.WriteLine($"Klucz (hex):
{BitConverter.ToString(aes.Key).Replace("-", " ")}");
            Console.WriteLine($"Nowy IV (hex):
{BitConverter.ToString(aes.IV).Replace("-", " ")}");

            // Sprawdzenie czy klucz został poprawnie załadowany
            Console.WriteLine($"Klucz w obiekcie AES zgadza się z
oryginałem? {CompareByteArrays(originalKey, aes.Key)}");
        }

        Console.WriteLine("\nNaciśnij Enter, aby usunąć pliki testowe...");
        Console.ReadLine();

        // Sprzątanie - usunięcie plików testowych
        foreach (string file in Directory.GetFiles(".", "test_key*"))
        {
            File.Delete(file);
            Console.WriteLine($"Usunięto: {file}");
        }

        Console.WriteLine("\n=== ZAKOŃCZONO ===");
        Console.ReadLine();
    }

    // Metoda pomocnicza: porównanie dwóch tablic bajtów
    static bool CompareByteArrays(byte[] a, byte[] b)
    {
        if (a.Length != b.Length) return false;
        for (int i = 0; i < a.Length; i++)
        {
            if (a[i] != b[i]) return false;
        }
        return true;
    }

    // Metoda pomocnicza: konwersja stringa hex na tablicę bajtów
    static byte[] HexStringToByteArray(string hex)
```


Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

```
    {
        // Usuwamy ewentualne myślniki i białe znaki
        hex = Regex.Replace(hex, @"[-\s]", "");

        int length = hex.Length;
        byte[] bytes = new byte[length / 2];

        for (int i = 0; i < length; i += 2)
        {
            bytes[i / 2] = Convert.ToByte(hex.Substring(i, 2), 16);
        }

        return bytes;
    }

    // Metoda pomocnicza: wyciągnięcie Base64 z formatu PEM
    static string ExtractBase64FromPem(string pem)
    {
        // Szukamy tekstu między znacznikami BEGIN i END
        var match = Regex.Match(pem, @"-----BEGIN.*?-----\s*(.*?)\s*-----
END.*?-----", RegexOptions.Singleline);
        if (match.Success)
        {
            return match.Groups[1].Value.Trim();
        }

        // Jeśli nie znaleziono formatu PEM, zwracamy cały tekst
        return pem.Trim();
    }
}
```

Szyfrowanie i deszyfrowanie AES w C# na przykładzie "Hello World!!!"

```
using System;
using System.IO;
using System.Security.Cryptography;
using System.Text;

class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("=== SZYFROWANIE I DESZYFROWANIE AES ===\n");
```

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

```
        // Wiadomość do zaszyfrowania
        string originalMessage = "Hello World!!!";
        Console.WriteLine($"Oryginalna wiadomość: {originalMessage}\n");

        // Krok 1: Generujemy klucz i IV
        using (Aes aes = Aes.Create())
        {
            aes.KeySize = 256;
            aes.GenerateKey();
            aes.GenerateIV();

            byte[] key = aes.Key;
            byte[] iv = aes.IV;

            Console.WriteLine($"Klucz (hex):
{BitConverter.ToString(key).Replace("-", " ")}");
            Console.WriteLine($"IV (hex):
{BitConverter.ToString(iv).Replace("-", " ")}\n");

            // Krok 2: Szyfrowanie
            byte[] encryptedData = EncryptStringToBytes(originalMessage,
key, iv);
            Console.WriteLine($"Zaszyfrowane dane (hex):
{BitConverter.ToString(encryptedData).Replace("-", " ")}");
            Console.WriteLine($"Zaszyfrowane dane (Base64):
{Convert.ToBase64String(encryptedData)}\n");

            // Krok 3: Deszyfrowanie
            string decryptedMessage = DecryptStringFromBytes(encryptedData,
key, iv);
            Console.WriteLine($"Odszyfrowana wiadomość:
{decryptedMessage}");

            // Weryfikacja
            Console.WriteLine($"\nCzy oryginał i odszyfrowane są
identyczne? {originalMessage == decryptedMessage}");
        }

        Console.WriteLine("\nNaciśnij Enter, aby zakończyć...");
        Console.ReadLine();
    }

    static byte[] EncryptStringToBytes(string plainText, byte[] key, byte[]
iv)
    {
```


Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

```
        // Sprawdzenie parametrów
        if (plainText == null || plainText.Length <= 0)
            throw new ArgumentNullException(nameof(plainText));
        if (key == null || key.Length <= 0)
            throw new ArgumentNullException(nameof(key));
        if (iv == null || iv.Length <= 0)
            throw new ArgumentNullException(nameof(iv));

        byte[] encrypted;

        // Tworzymy obiekt AES z podanym kluczem i IV
        using (Aes aes = Aes.Create())
        {
            aes.Key = key;
            aes.IV = iv;

            // Tworzymy szyfrator
            ICryptoTransform encryptor = aes.CreateEncryptor(aes.Key,
aes.IV);

            // Tworzymy strumień pamięci do przechowania zaszyfrowanych
danych
            using (MemoryStream msEncrypt = new MemoryStream())
            {
                using (CryptoStream csEncrypt = new CryptoStream(msEncrypt,
encryptor, CryptoStreamMode.Write))
                {
                    using (StreamWriter swEncrypt = new
StreamWriter(csEncrypt))
                    {
                        // Zapisz dane do strumienia kryptograficznego
                        swEncrypt.Write(plainText);
                    }
                    encrypted = msEncrypt.ToArray();
                }
            }
        }

        return encrypted;
    }

    static string DecryptStringFromBytes(byte[] cipherText, byte[] key,
byte[] iv)
    {
        // Sprawdzenie parametrów
        if (cipherText == null || cipherText.Length <= 0)
```


Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

```
            throw new ArgumentNullException(nameof(cipherText));
        if (key == null || key.Length <= 0)
            throw new ArgumentNullException(nameof(key));
        if (iv == null || iv.Length <= 0)
            throw new ArgumentNullException(nameof(iv));

        string plaintext = null;

        // Tworzymy obiekt AES z podanym kluczem i IV
        using (Aes aes = Aes.Create())
        {
            aes.Key = key;
            aes.IV = iv;

            // Tworzymy deszyfrator
            ICryptoTransform decryptor = aes.CreateDecryptor(aes.Key,
aes.IV);

            // Tworzymy strumień pamięci z zaszyfrowanymi danymi
            using (MemoryStream msDecrypt = new MemoryStream(cipherText))
            {
                using (CryptoStream csDecrypt = new CryptoStream(msDecrypt,
decryptor, CryptoStreamMode.Read))
                {
                    using (StreamReader srDecrypt = new
StreamReader(csDecrypt))
                    {
                        // Odczytaj odszyfrowane dane
                        plaintext = srDecrypt.ReadToEnd();
                    }
                }
            }
        }

        return plaintext;
    }
}
```

Wektor inicjujący (IV) - zawsze używaj losowego IV dla każdej operacji szyfrowania. W powyższym przykładzie IV jest stały (zapisany w pliku), co w praktyce jest błędem - IV powinien być unikalny dla każdej szyfrowanej wiadomości i najczęściej jest dołączany na początku zaszyfrowanych danych.

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Zarządzanie kluczami - w praktyce klucze AES rzadko są przechowywane w plikach binarnych. Częściej używa się:

- Zabezpieczonych plików konfiguracyjnych z ograniczonymi uprawnieniami
- Magazynów kluczy (Windows Certificate Store)
- Sprzętowych modułów bezpieczeństwa (HSM)
- Usług zarządzania kluczami (Azure Key Vault, AWS KMS)

Ćwiczenia samodzielne

Zadanie 4.1

Zapoznaj się z klasą RSA i znajdującymi się w niej metodami do eksportu i importu kluczy. Powtórz operacje generowania, zapisu i wczytywania kluczy z przykładu AES z kluczami RSA, pamiętając, że w kryptosystemie asymetrycznym mamy parę kluczy. Dodatkowo dla RSA mamy wbudowane wsparcie dla formatu PEM:

```
using (RSA rsa = RSA.Create(2048))
{
    // Eksport klucza prywatnego w formacie PEM (PKCS#1)
    string privateKeyPem = rsa.ExportRSAPrivateKeyPem();
    // Zapisuje:
    // -----BEGIN RSA PRIVATE KEY-----
    // MIIEpAIBAAKCAQEA... (base64)
    // -----END RSA PRIVATE KEY-----

    // Eksport klucza publicznego w formacie PEM (X.509
SubjectPublicKeyInfo)
    string publicKeyPem = rsa.ExportSubjectPublicKeyInfoPem();
    // Zapisuje:
    // -----BEGIN PUBLIC KEY-----
    // MIIBCgKCAQEA... (base64)
    // -----END PUBLIC KEY-----
}
```

Zadanie 4.2

Napisz program, który pozwala na otwarcie klucza publicznego i prywatnego ze wskazanego pliku, a następnie zaszyfrowanie i odszyfrowanie wiadomości tekstowej.

Fundusze Europejskie dla Rozwoju Społecznego | Rzeczpospolita Polska | Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Zadanie 4.3

Wygeneruj klucze RSA w formacie PEM i zapisz w plikach na swoim komputerze. Udostępnij w Teamsach swój klucz publiczny innym osobom z grupy. Pobierz klucze 2 osób z grupy i wyślij do nich zaszyfrowane wiadomości. Po odebraniu wiadomości od innych osób odszyfruj je używając swojego klucza prywatnego. Jeżeli uda się odczytać tekst odpisz tekstem jawnym „Odebrano”.

Pytania pomocnicze

1. W jakich formatach można zapisywać klucze kryptograficzne?
2. Jaka biblioteka C# zawiera algorytmy AES i RSA?

Podsumowanie

Dzisiejsze zajęcia przeprowadziły nas przez pełen cykl życia kluczy kryptograficznych – od ich generacji, poprzez praktyczne zastosowanie, aż po programistyczną implementację. Głównym zadaniem było przeniesienie wiedzy o kluczach na grunt programistyczny – w środowisku C# z wykorzystaniem wbudowanych bibliotek System.Security.Cryptography samodzielnie zaimplementowaliśmy generację kluczy AES i RSA. Kluczowym wnioskiem jest zasada, że nawet najsilniejsze algorytmy nie zapewnią bezpieczeństwa, jeśli zawiedzie ich implementacja lub zarządzanie kluczami – to one są fundamentem, na którym buduje się zaufanie w cyfrowym świecie.

Literatura

1. Dokumentacja klasy AES w C#

   https://learn.microsoft.com/en-us/dotnet/api/system.security.cryptography.aes

2. Dokumentacja klasy RSA w C#

   https://learn.microsoft.com/en-us/dotnet/api/system.security.cryptography.rsa

Fundusze Europejskie dla Rozwoju Społecznego | Rzeczpospolita Polska | Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Materiały do przedmiotu:

Podstawy kryptografii

Laboratorium nr 5

Analiza ataków na systemy kryptograficzne

Autor:

dr Michał Dolecki

Fundusze Europejskie dla Rozwoju Społecznego

Rzeczpospolita Polska

Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Spis treści

Fundusze Europejskie dla Rozwoju Społecznego

Rzeczpospolita Polska

Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Wprowadzenie, tematyka zajęć

Wprowadzenie do niniejszych laboratoriów stanowi kontynuację zagadnień związanych z analizą bezpieczeństwa systemów kryptograficznych, ze szczególnym uwzględnieniem metod oceny jakości szyfrogramów oraz technik łamania prostych algorytmów symetrycznych. W trakcie zajęć studenci skupią się na praktycznym zastosowaniu miar teoretyczno-informacyjnych, w szczególności entropii Shannona, która pozwala na ilościową ocenę losowości danych wyjściowych i stanowi pierwsze, często decydujące narzędzie diagnostyczne w procesie kryptoanalizy. Poprzez implementację w języku C# studenci nauczą się wyznaczać entropię dla szyfrogramów na poziomie bajtów, co umożliwi im odróżnienie danych o wysokim stopniu przypadkowości od tych, które zachowują strukturalne regularności mogące stanowić słabość systemu kryptograficznego. Wiedza ta stanowi niezbędne przygotowanie do dalszych etapów analizy, gdzie statystyka języka naturalnego staje się kluczowym narzędziem ataku.

W drugiej części laboratoriów studenci przeprowadzą praktyczne ataki na dwa klasyczne szyfry – Cezara oraz przestawieniowy z ustaloną permutacją. W przypadku szyfru Cezara porównane zostaną dwie strategie kryptoanalityczne: atak brute-force wykorzystujący małą przestrzeń kluczy oraz atak oparty na analizie częstości występowania liter, który pozwala na automatyczną identyfikację poprawnego przesunięcia bez konieczności przeglądania wszystkich możliwości. Następnie, w ramach ataku na szyfr przestawieniowy, studenci zmierzą się z problemem odtworzenia nieznanej permutacji bez zmiany wartości poszczególnych znaków. Zastosowana zostanie metoda analizy częstości bigramów i trigramów w połączeniu z algorytmem wspinaczkowym (hill climbing), który poprzez sekwencyjne, losowe modyfikacje kolejności kolumn i ocenę uzyskiwanych tekstów jawnych pozwoli na efektywne odtworzenie struktury permutacji nawet przy dłuższych blokach, gdzie przeszukiwanie wyczerpujące staje się niewykonalne. Celem tych ćwiczeń jest nie tylko praktyczne złamanie omawianych szyfrów, ale przede wszystkim zrozumienie, w jaki sposób wiedza o statystyce języka oraz odpowiednio zaprojektowane algorytmy przeszukiwania mogą zastąpić nieefektywne metody brute-force, stanowiąc fundament współczesnej kryptoanalizy.

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Cel laboratorium

Celem laboratoriów jest wykształcenie u studentów umiejętności praktycznej analizy słabości systemów kryptograficznych poprzez zastosowanie metod statystycznych oraz algorytmów przeszukiwania, z jednoczesnym ugruntowaniem przekonania, że o bezpieczeństwie szyfru decyduje nie tyle złożoność samego algorytmu, ile odporność na ustrukturyzowane ataki wykorzystujące właściwości języka naturalnego oraz ograniczenia przestrzeni kluczy. Poprzez sekwencyjne przechodzenie od prostych metod brute-force, przez analizę częstości liter, aż po heurystyczne algorytmy wspinaczkowe oparte na bigramach, studenci nie tylko przyswajają konkretne techniki kryptoanalityczne, ale przede wszystkim rozwijają myślenie algorytmiczne i uczą się krytycznej oceny jakości implementacji kryptograficznych – kompetencje kluczowe zarówno dla przyszłych projektantów systemów bezpieczeństwa, jak i specjalistów zajmujących się testowaniem podatności rozwiązań informatycznych.

Instrukcja laboratoryjna

Ocena jakości szyfru za pomocą pomiaru entropii stanowi jeden z podstawowych, a zarazem najszybciej dostępnych wskaźników w procesie analizy kryptosystemu. W praktyce laboratoryjnej, operując na danych w postaci binarnej lub bajtowej, kluczowe jest zrozumienie, że entropię najczęściej wyznacza się dla bajtów (8-bitowych bloków), a nie dla pojedynczych bitów. Wybór ten wynika z natury współczesnych algorytmów kryptograficznych, które operują na bajtach jako podstawowej jednostce przetwarzania – zarówno w szyfrach blokowych (jak AES), jak i podczas analizy szyfrogramów przechowywanych w plikach. Liczenie entropii na poziomie bitów mogłoby być mylące, ponieważ idealny szyfrogram powinien wykazywać równomierny rozkład bitów, jednak to właśnie analiza na poziomie bajtów lepiej odzwierciedla potencjalne wzorce strukturalne, takie jak powtarzające się bloki w trybie ECB czy niejednorodności wynikające z nieprawidłowego paddingu. Wartość entropii wyznaczona dla bajtów pozwala wychwycić subtelne anomalie, które przy agregacji bitowej uległyby uśrednieniu.

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

W teorii informacji najczęściej stosuje się miarę entropii Shannona, która dla danego zbioru danych wyraża średnią niepewność związaną z przewidzeniem kolejnego symbolu. Entropia Hartleya (logarytm z liczby możliwych stanów) ma znaczenie głównie teoretyczne przy założeniu równomiernego rozkładu – w rzeczywistych szyfrogramach to właśnie entropia Shannona dostarcza informacji o faktycznej losowości i stopniu uporządkowania danych. Wartość ta przyjmuje maksimum (8 bitów na bajt) w przypadku idealnego rozkładu równomiernego wszystkich 256 wartości bajtowych, co jest pożądaną cechą wysokiej jakości szyfrogramu. Im niższa entropia, tym większy stopień regularności, a co za tym idzie – większe prawdopodobieństwo istnienia korelacji statystycznych, które mogą zostać wykorzystane przez atakującego. W kontekście analizy kryptosystemu pomiar entropii Shannona stanowi więc pierwszą linię obrony przed implementacją wadliwych algorytmów lub błędów konfiguracyjnych, takich jak użycie trybu ECB, brak losowego wektora inicjującego czy stosowanie tzw. textbook RSA bez odpowiedniego paddingu.

Znaczenie pomiaru entropii na początku analizy kryptosystemu jest kluczowe z dwóch powodów. Po pierwsze, pozwala na szybkie odróżnienie szyfrogramu o wysokiej jakości kryptograficznej od danych, które jedynie pozornie są zaszyfrowane – na przykład wynikających z kompresji, kodowania czy prostych przekształceń bitowych. W praktyce laboratoryjnej, gdzie studenci implementują własne warianty szyfrów, entropia stanowi obiektywny wskaźnik, który może ujawnić niezamierzoną deterministyczność lub niewystarczające mieszanie bitów w algorytmie. Po drugie, pomiar ten dostarcza kontekstu dla dalszych, bardziej zaawansowanych ataków – jeśli szyfrogram charakteryzuje się entropią zbliżoną do teoretycznego maksimum, analityk musi sięgnąć po bardziej wyrafinowane metody (np. analizę czasową, ataki z wyborem tekstu jawnego lub szyfrogramu). Natomiast jeśli entropia jest istotnie niższa, sugeruje to fundamentalne słabości, które często można wykorzystać bezpośrednio – jak w przypadku klasycznych szyfrów podstawieniowych, gdzie entropia tekstu zaszyfrowanego pozostaje na poziomie języka naturalnego. W ten sposób entropia Shannona dla bajtów pełni rolę nie tylko narzędzia diagnostycznego, ale również drogowskazu kierującego dalszym tokiem analizy bezpieczeństwa analizowanego systemu kryptograficznego.

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Ataki na Szyfr Cezara

Atak brutalny

Atak brutalny (brute-force) na szyfr Cezara jest najprostszym z możliwych ataków kryptanalitycznych, który wykorzystuje fundamentalną słabość tego algorytmu – jego skrajnie małą przestrzeń kluczy. Szyfr Cezara polega na przesunięciu każdej litery alfabetu o stałą liczbę pozycji, co w przypadku współczesnego alfabetu łacińskiego (bez rozróżniania wielkości liter i bez uwzględniania znaków diakrytyzowanych) daje zaledwie 25 możliwych, niezerowych kluczy. Taka liczba kombinacji czyni atak wyczerpujący nie tylko teoretycznie wykonalnym, ale również praktycznie trywialnym – nawet ręczne wykonanie go dla krótkich tekstów zajmuje zaledwie kilkanaście minut, a w implementacji programistycznej w języku C# jest kwestią milisekund.

W praktyce laboratoryjnej atak brute-force na szyfr Cezara realizuje się poprzez iteracyjne stosowanie odwrotnego przesunięcia dla każdego z 25 możliwych kluczy. Dla każdej próby generowany jest kandydat na tekst jawny, który następnie podlega ocenie pod kątem poprawności językowej. Kluczowym elementem implementacji jest tutaj umiejętność automatycznej weryfikacji, które z odszyfrowanych tekstów stanowią spójny komunikat w języku naturalnym, a które są jedynie przypadkowym zbiorem znaków. W najprostszej wersji wystarczy porównanie odszyfrowanego tekstu ze słownikiem najczęściej występujących wyrazów lub analiza częstości występowania liter i porównanie z modelem języka polskiego lub angielskiego. W kontekście przedmiotu "Podstawy kryptografii" atak ten pełni funkcję dydaktyczną.

Atak statystyczny

Atak z wykorzystaniem statystyki liter stanowi bardziej wyrafinowaną metodę kryptoanalizy szyfru Cezara niż atak brute-force, opierającą się na fundamentalnej właściwości języków naturalnych – niejednorodnym rozkładzie występowania poszczególnych liter w tekście. W języku polskim, podobnie jak w innych językach europejskich, litery pojawiają się z różną częstotliwością: przykładowo litera "a" występuje najczęściej, następnie "e", "i", "o", "n", podczas gdy litery takie jak "q", "v", "x" należą do rzadkości. Szyfr Cezara, będący przesunięciem liter, zachowuje ten rozkład – zmienia jedynie etykiety poszczególnych liter, ale nie zmienia samego faktu, że pewne litery w tekście zaszyfrowanym będą występować częściej

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

niż inne. Innymi słowy, histogram częstości liter w szyfrogramie jest identyczny w kształcie jak histogram języka jawnego, tyle że przesunięty o wartość klucza.

W praktyce laboratoryjnej atak ten przeprowadza się poprzez wyznaczenie rozkładu częstości występowania wszystkich liter w przechwyconym szyfrogramie. Następnie analizator porównuje uzyskany histogram z wzorcowym rozkładem częstości liter dla danego języka naturalnego (np. polskiego lub angielskiego), który może być wcześniej przygotowany na podstawie korpusu tekstów referencyjnych. Kluczowym etapem jest znalezienie takiego przesunięcia, które najlepiej dopasowuje histogram szyfrogramu do histogramu wzorcowego – najczęściej poprzez identyfikację litery o największej częstości w szyfrogramie i założenie, że odpowiada ona literze "a" (lub innej najbardziej typowej dla danego języka) w tekście jawnym. Różnica pozycji między tymi literami w alfabecie wskazuje bezpośrednio na wartość klucza. Metoda ta jest szczególnie skuteczna przy dłuższych tekstach, gdzie fluktuacje statystyczne są mniej znaczące, a rozkład częstości stabilizuje się w okolicach wartości charakterystycznych dla danego języka.

Wartość dydaktyczna tego ataku w ramach przedmiotu "Podstawy kryptografii" jest wielowymiarowa. Po pierwsze, studenci uczą się, że bezpieczeństwo szyfru nie może opierać się wyłącznie na przesłonięciu struktury języka, jeśli transformacja nie zaciera jednocześnie jego własności statystycznych. Po drugie, atak ten wprowadza koncepcję analizy częstości jako narzędzia kryptanalizy, które znajduje zastosowanie nie tylko w przypadku szyfru Cezara, ale również szerszej klasy szyfrów podstawieniowych.

Ataki na Szyfr Przestawieniowy

Atak z użyciem analizy częstości bigramów i trigramów

Taki atak na szyfr przestawieniowy (permutacyjny) stanowi zaawansowaną metodę kryptoanalizy, która wykracza poza proste badanie rozkładu pojedynczych liter. Szyfr przestawieniowy, w przeciwieństwie do szyfru Cezara, nie zmienia wartości poszczególnych znaków – jedynie zmienia ich kolejność zgodnie z ustaloną permutacją. Oznacza to, że rozkład częstości pojedynczych liter w szyfrogramie jest identyczny jak w tekście jawnym, co czyni analizę unigramów całkowicie bezużyteczną do złamania tego szyfru. Atakujący musi zatem sięgnąć po wyższe rzędy statystyk – bigramy (dwie kolejne litery) i trigramy (trzy kolejne litery) – które niosą informację o strukturze sekwencji w języku naturalnym. Ponieważ

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

permutacja rozbija naturalne pary i trójki liter charakterystyczne dla danego języka, analiza częstości bigramów i trigramów pozwala na odtworzenie oryginalnego porządku znaków poprzez identyfikację, które fragmenty szyfrogramu po odpowiednim przekształceniu tworzą znane, często występujące wzorce językowe.

W praktyce laboratoryjnej atak ten realizuje się poprzez kilkuetapową procedurę. Pierwszym krokiem jest zebranie statystyk referencyjnych – najczęściej występujących bigramów i trigramów w języku polskim lub angielskim, takich jak "st", "cz", "ni", "nie", "ego", "ych" itp. Następnie, dysponując szyfrogramem o znanej długości i zakładając, że permutacja działa na blokach o ustalonej długości (co jest typowe dla tego typu szyfrów), atakujący generuje wszystkie możliwe permutacje kolumn (dla małych bloków) lub stosuje algorytmy heurystyczne dla większych bloków. Kluczowym elementem jest ocena jakości każdej testowanej permutacji poprzez złożenie odszyfrowanego tekstu i obliczenie, jak często występują w nim bigramy i trigramy zgodne z modelem języka naturalnego. Im więcej wysokoczęstościowych bigramów i trigramów pojawia się w potencjalnym tekście jawnym, tym większe prawdopodobieństwo, że testowana permutacja jest poprawna. W bardziej zaawansowanych implementacjach wykorzystuje się również analizę wzajemnych odległości między wystąpieniami tych samych trigramów, co pozwala na wnioskowanie o strukturze permutacji nawet bez bezpośredniego iterowania po wszystkich możliwych kluczach.

W metodzie sekwencyjnego konstruowania permutacji, znanej również jako *metoda zachłanna* lub *algorytm z nawrotami* (ang. *hill climbing*), atakujący postępuje w następujący sposób:

1. **Przygotowanie kolumn** – po ustaleniu (lub założeniu) długości bloku n, dzieli szyfrogram na n kolumn. Każda kolumna to ciąg znaków, które w oryginalnym tekście jawnym występowały na tej samej pozycji w kolejnych blokach, ale w szyfrogramie zostały rozrzucone zgodnie z nieznaną permutacją.
2. **Wybór punktu startowego** – algorytm rozpoczyna od dowolnego ułożenia kolumn, na przykład w kolejności naturalnej (1,2,3,...,n) lub losowej. Następnie oblicza dla tego ułożenia łączną ocenę statystyczną – najczęściej sumę częstości (lub logarytmów prawdopodobieństw) wszystkich bigramów występujących w tekście złożonym z kolumn ułożonych w tej kolejności.

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

3. **Modyfikacje i ocena** – algorytm iteracyjnie próbuje ulepszyć bieżące ułożenie, dokonując lokalnych zmian. Najczęściej stosowaną operacją jest **zamiana miejscami dwóch kolumn** lub **przesunięcie kolumny w inne miejsce** w sekwencji. Po każdej takiej modyfikacji algorytm ponownie składa tekst z kolumn w nowej kolejności i oblicza jego ocenę bigramową.
4. **Akceptacja zmian** – jeśli nowe ułożenie daje wyższą ocenę (więcej zgodnych z językiem bigramów), algorytm je przyjmuje i kontynuuje poszukiwania od tego punktu. Jeśli nie, odrzuca zmianę i próbuje innej modyfikacji.

Wartość dydaktyczna tego ataku w kontekście przedmiotu "Podstawy kryptografii" jest szczególnie istotna z kilku powodów. Po pierwsze, studenci poznają fundamentalne rozróżnienie między szyframi podstawieniowymi a przestawieniowymi – pierwsze zmieniają tożsamość znaków, drugie jedynie ich kolejność, co ma kluczowe znaczenie dla metodologii ataku. Po drugie, atak na bigramy i trigramy wprowadza koncepcję analizy wyższego rzędu, która jest niezbędna przy badaniu bardziej złożonych systemów kryptograficznych. Po trzecie, studenci mogą zaobserwować zależność między długością bloku permutacji a trudnością ataku – dla małych bloków (np. 5–10 znaków) możliwe jest przeszukanie wyczerpujące, podczas gdy dla większych bloków konieczne staje się stosowanie metod probabilistycznych i heurystycznych. Wreszcie, w kontekście wcześniejszych rozważań o entropii, atak ten doskonale ilustruje ograniczenie entropii Shannona jako miary jakości szyfru – ponieważ szyfr przestawieniowy nie zmienia entropii tekstu (pozostaje ona na poziomie języka naturalnego), atakujący musi sięgnąć po bardziej zaawansowane narzędzia statystyczne, które ujawniają zaburzoną strukturę sekwencji, mimo że pojedyncze znaki zachowują swoje naturalne rozkłady.

Ćwiczenia samodzielne

Zadanie 5.1

Przygotuj zestawy po 10 tekstów jawnych i odpowiadających im szyfrogramów dla Szyfru Cezara, Szyfru przestawieniowego dla bloku długości 10 znaków, AES i RSA. Użyj różnej długości tekstów od najkrótszych jednozdaniowych do kilkuzdaniowych. Do wykonania tego zadania użyj swoich programów z

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

poprzednich laboratoriów. Następnie policz entropię Shannona dla tekstów jawnych i szyfrogramów. Zapisz klucze używane w szyfrach oraz permutację.

Zadanie 5.2

Napisz program, który przeprowadza atak brute-force na Szyfr Cezara. Sprawdź jego działanie na wybranych szyfrogramach z zadnia 5.1.

Zadanie 5.3

Napisz program, który przeprowadzi atak statystyczny na Szyfr Cezara. Wyświetl statystykę liter w szyfrogramie, wyznacz 3 najbardziej prawdopodobne wartości klucza i sprawdź czy uda się złamać szyfrogram. Sprawdź dla wyznaczonych w zadaniu 5.1 szyfrogramów o różnej długości.

Zadanie 5.4

Napisz program, który dla danego pliku tekstowego wyznaczy częstotliwość bigramów (dwuliterowych ciągów) w formie tablicy, gdzie wiersze odpowiadają pierwszej literze z bigramu, a kolumny drugiej. Zapisz wynikową tablicę do pliku. Podaj 10 częstszych bigramów w języku polskim i angielskim. Możesz użyć tekstu książki z Laboratorium 1 do wyznaczenia tej wartości.

Zadanie 5.5

Napisz program realizujący metodę sekwencyjnego konstruowania permutacji do łamania szyfru przestawieniowego. Załóżmy, że atakujący zna długość permutacji wynoszącą n=10. Jako funkcję oceny wykorzystaj sumę logarytmów częstotliwości digramów. Jako warunek końca przyjmij brak poprawy wyniku przez 1000 iteracji. Podaj ustaloną permutację, liczbę iteracji do jej wyznaczenia (bez 1000 iteracji bez poprawy) oraz odszyfrowany tekst. Na szyfrogramach z zadnia 5.1 sprawdź skuteczność metody.

Pytania pomocnicze

1. Jakie znasz sposoby ataku na Szyfr Cezara?
2. Jakie znasz sposoby ataku na Szyfr Przestawieniowy?
3. Jak działa algorytm sekwencyjnego budowania permutacji?

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Podsumowanie

Podczas dwugodzinnych laboratoriów z analizy ataków na systemy kryptograficzne studenci zdobyli praktyczną wiedzę z zakresu oceny jakości szyfrogramów oraz metod łamania wybranych algorytmów. Poprzez pomiar entropii Shannona dla bajtów nauczyli się odróżniać losowe szyfrogramy wysokiej jakości od tych, które zachowują strukturalne regularności – co stanowi niezbędny pierwszy krok w każdej analizie kryptosystemu. Następnie, w ramach ataków na szyfr Cezara, studenci porównali metodę brute-force z analizą częstości liter, dostrzegając zarówno ograniczenia wynikające z małej przestrzeni kluczy, jak i znaczenie statystyki języka w automatyzacji kryptoanalizy. W kolejnym ćwiczeniu, dotyczącym szyfru przestawieniowego z nieznaną permutacją, studenci zaimplementowali algorytm hill climbing wykorzystujący ocenę bigramową do sekwencyjnego konstruowania poprawnego ułożenia kolumn, co unaoczniło, jak w przypadku większych długości bloku (np. n=10) konieczne jest stosowanie heurystyk zamiast przeszukiwania wyczerpującego. Całość zajęć podkreśliła fundamentalne znaczenie analizy statystycznej w procesie kryptoanalizy oraz pokazała, że nawet proste techniki – entropia, bigramy czy algorytmy wspinaczkowe – mogą być skutecznymi narzędziami w ocenie i łamaniu systemów kryptograficznych, stanowiąc przy tym solidną podstawę do zrozumienia bardziej zaawansowanych metod ataków na współczesne algorytmy, takie jak AES czy RSA.

Literatura

1. D. R. Stinson, M. B. Paterson, Kryptografia. W teorii i w praktyce, PWN 2021
2. A. J. Menezes , P. C. Oorschot , S. A. Vanstone, Kryptografia stosowana, WNT 2005

Fundusze Europejskie dla Rozwoju Społecznego

Rzeczpospolita Polska

Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Materiały do przedmiotu:

Podstawy kryptografii

Laboratorium nr 6

Kryptografia w sieciach

Autor:

dr Michał Dolecki

Fundusze Europejskie dla Rozwoju Społecznego

Rzeczpospolita Polska

Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Spis treści

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Wprowadzenie, tematyka zajęć

Rejestracja nowego użytkownika w serwisie internetowym to proces, który na pozór wydaje się prosty – użytkownik podaje nazwę i hasło, a system tworzy konto. Jednak za tą pozorną prostotą kryje się jeden z najbardziej krytycznych obszarów bezpieczeństwa aplikacji: sposób przechowywania haseł w bazie danych. Przechowywanie haseł w postaci jawnego tekstu stanowi katastrofalny błąd bezpieczeństwa – w przypadku wycieku bazy danych atakujący uzyskuje bezpośredni dostęp do wszystkich kont użytkowników, co często prowadzi do kompromitacji nie tylko danego serwisu, ale także innych platform, na których użytkownicy stosują te same hasła. Z tego powodu współczesne systemy nigdy nie przechowują samych haseł, a jedynie ich jednokierunkowe reprezentacje – hasze. Haszowanie jest operacją matematyczną, która przekształca hasło o dowolnej długości w ciąg o stałej długości, przy czym proces ten jest praktycznie nieodwracalny. Dzięki temu nawet w przypadku wycieku bazy danych atakujący widzi jedynie ciągi znaków, z których nie jest w stanie odtworzyć oryginalnych haseł w rozsądnym czasie.

Kluczowym uzupełnieniem samego haszowania jest stosowanie **soli** (ang. *salt*) – losowego, unikalnego ciągu znaków generowanego dla każdego użytkownika osobno i dołączanego do hasła przed procesem haszowania. Sól pełni dwie zasadnicze funkcje. Po pierwsze, sprawia, że dwa identyczne hasła od różnych użytkowników dają zupełnie różne hasze, co uniemożliwia atakującemu wykorzystanie tęczowych tablic (ang. *rainbow tables*) – prekompilowanych słowników zawierających hasze dla milionów często używanych haseł. Po drugie, sól znacząco utrudnia ataki równoległe, ponieważ atakujący musi atakować każde hasło osobno, zamiast stosować uniwersalną tablicę dla całej bazy danych. W praktyce sól jest przechowywana w bazie danych obok hasha, a proces weryfikacji hasła podczas logowania polega na odczytaniu soli, dołączeniu jej do podanego hasła, ponownym obliczeniu hasha i porównaniu z wartością przechowywaną w bazie.

W architekturze klient-serwer, gdzie aplikacja kliencka napisana w C# komunikuje się z serwerem PHP i bazą danych MariaDB, istotnym elementem staje się również zarządzanie stanem uwierzytelnienia po zalogowaniu. Ponieważ protokół HTTP jest z natury bezstanowy, po udanym logowaniu serwer musi przekazać klientowi dowód autoryzacji, który będzie dołączany do kolejnych żądań. Jednym z

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

najpopularniejszych rozwiązań są **tokeny JWT** (JSON Web Tokens) – samodzielne, podpisane cyfrowo struktury danych zawierające informacje o użytkowniku oraz metadane, takie jak czas ważności tokena. JWT jest podpisywany przez serwer przy użyciu klucza tajnego, co uniemożliwia klientowi jego modyfikację, a jego struktura pozwala na weryfikację autentyczności i integralności bez konieczności przechowywania stanu sesji po stronie serwera. W niniejszym laboratorium studenci zrealizują pełny przepływ uwierzytelniania – od bezpiecznej rejestracji z haszowaniem i solą, przez logowanie i generowanie tokena JWT, aż po dostęp do chronionych zasobów API – integrując w praktyce wiedzę z zakresu kryptografii, bezpieczeństwa aplikacji oraz programowania w środowisku C# i PHP z wykorzystaniem bazy danych MariaDB.

Cel laboratorium

Celem laboratorium jest nabycie przez studentów umiejętności projektowania i implementacji bezpiecznego systemu uwierzytelniania w architekturze klient-serwer, ze szczególnym uwzględnieniem kryptograficznych metod ochrony danych wrażliwych. Poprzez praktyczną realizację rejestracji i logowania użytkowników z wykorzystaniem bazy danych MariaDB, warstwy pośredniczącej w języku PHP oraz aplikacji klienckiej w C#, studenci poznają kluczowe mechanizmy bezpieczeństwa – jednokierunkowe haszowanie haseł z zastosowaniem soli zapobiegające ujawnieniu poświadczeń w przypadku wycieku bazy danych, oraz zarządzanie sesją opartą na tokenach JWT umożliwiającą bezpieczną autoryzację kolejnych żądań. W efekcie studenci nie tylko zdobędą praktyczne umiejętności implementacyjne, ale także zrozumieją, dlaczego przechowywanie haseł w postaci jawnej stanowi krytyczną podatność oraz jakie mechanizmy kryptograficzne i architektoniczne są niezbędne do zbudowania odpornego na typowe ataki systemu uwierzytelniania.

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Baza danych

Centralnym elementem każdego systemu uwierzytelniania jest baza danych przechowująca poświadczenia użytkowników. W najprostszym, a zarazem najczęściej spotykanym wariancie projektowym, tworzy się pojedynczą tabelę users, która zawiera kolumny przechowujące unikalny identyfikator użytkownika, jego login (lub adres e-mail), a także dane niezbędne do weryfikacji hasła – przede wszystkim sól oraz hash. W tym modelu sól, będąca losowym ciągiem generowanym indywidualnie dla każdego użytkownika w momencie rejestracji, przechowywana jest bezpośrednio obok hasha w tym samym wierszu tabeli. Takie rozwiązanie jest poprawne pod względem bezpieczeństwa, ponieważ sól nie wymaga szczególnej ochrony – jej rola polega wyłącznie na zapewnieniu unikalności wejścia do funkcji haszującej, a nie na byciu tajemnicą. Podczas logowania aplikacja odczytuje z bazy zarówno sól, jak i hash, a następnie oblicza hash z połączenia soli i podanego hasła, porównując wynik z wartością przechowywaną. Prostota tego podejścia sprawia, że jest ono wygodne w implementacji i wystarczające dla większości zastosowań, pod warunkiem że funkcja haszująca jest odpowiednio dobrana (np. bcrypt lub Argon2) i charakteryzuje się kosztem obliczeniowym utrudniającym ataki brute-force.

Alternatywnym, bardziej zaawansowanym modelem jest rozdzielenie danych uwierzytelniających na dwie tabele powiązane relacją jeden do jednego. W tym podejściu tabela users zawiera wyłącznie dane publiczne lub identyfikacyjne, takie jak login, adres e-mail oraz identyfikator, natomiast wydzielona tabela credentials (lub auth_data) przechowuje sól i hash, łącząc się z tabelą users za pomocą klucza obcego. Taka separacja niesie ze sobą kilka korzyści z perspektywy bezpieczeństwa i zarządzania danymi. Po pierwsze, umożliwia precyzyjniejszą kontrolę dostępu na poziomie bazy danych – można nadać różne uprawnienia dla aplikacji uwierzytelniającej (która ma prawo odczytu i zapisu w tabeli z danymi uwierzytelniającymi) oraz dla innych modułów systemu, które nie powinny mieć dostępu do wrażliwych danych kryptograficznych. Po drugie, w przypadku konieczności zmiany algorytmu haszowania lub dodania dodatkowych parametrów (np. kosztu obliczeniowego, wersji algorytmu), struktura staje się bardziej elastyczna, a modyfikacje nie wymuszają zmian w głównej tabeli użytkowników. Po trzecie, separacja ta ułatwia zachowanie zgodności z zasadą

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

minimalnego przywileju (least privilege) oraz ogranicza ryzyko przypadkowego ujawnienia danych uwierzytelniających poprzez mniej zabezpieczone zapytania lub widoki bazy danych. Choć dla prostych aplikacji różnica w poziomie bezpieczeństwa może wydawać się marginalna, stosowanie tej praktyki odzwierciedla dobre nawyki projektowe, które w bardziej rozbudowanych systemach mają istotne znaczenie dla ochrony krytycznych danych.

W projektowanej bazie danych dla systemu uwierzytelniania proponuje się zastosowanie modelu z dwiema tabelami powiązanymi relacją jeden do jednego, co zapewnia lepszą separację odpowiedzialności oraz bardziej precyzyjną kontrolę dostępu do danych wrażliwych. Główna tabela users przeznaczona jest do przechowywania informacji o charakterze publicznym lub administracyjnym. Znajdą się w niej następujące pola: id – całkowitoliczbowy identyfikator o charakterze klucza głównego, automatycznie inkrementowany, stanowiący jednoznaczny identyfikator każdego użytkownika w systemie; username – łańcuch znaków o maksymalnej długości 50 bajtów, zdefiniowany jako unikalny (UNIQUE), służący jako nazwa użytkownika wykorzystywana podczas logowania; email – opcjonalne pole łańcuchowe o długości do 100 bajtów, również unikalne, umożliwiające kontakt z użytkownikiem oraz ewentualne mechanizmy odzyskiwania dostępu; created_at – znacznik czasu (timestamp) rejestrujący moment utworzenia konta, wypełniany automatycznie wartością bieżącą; oraz is_active – logiczna flaga (boolean) wskazująca, czy konto jest aktywne, co umożliwia czasową blokadę użytkownika bez usuwania jego danych z systemu.

Druga tabela, nazwana credentials, przechowuje wyłącznie dane związane z uwierzytelnianiem kryptograficznym i pozostaje w ścisłej relacji jeden do jednego z tabelą users. W tabeli tej kluczem głównym jest pole user_id, które jednocześnie pełni rolę klucza obcego wskazującego na users.id – takie rozwiązanie zapewnia, że każdy użytkownik może mieć co najwyżej jeden zestaw danych uwierzytelniających, a jednocześnie eliminuje konieczność stosowania osobnego, sztucznego identyfikatora. Kluczowym polem w tej tabeli jest password_hash – łańcuch znaków o długości do 255 bajtów, przechowujący wynik funkcji haszującej zastosowanej do połączenia soli i hasła użytkownika. W przypadku wykorzystania algorytmów takich jak bcrypt, które same w sobie zawierają sól w strukturze hasha, pole to może przechowywać kompletny ciąg wyjściowy funkcji. Alternatywnie, przy stosowaniu prostszych funkcji (np. SHA-256) konieczne jest wydzielenie pola salt – łańcucha o długości 32–64 bajtów, zawierającego unikalny, losowo wygenerowany ciąg znaków w postaci heksadecymalnej lub zakodowanej w Base64. Dodatkowo, w

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

tabeli credentials można przewidzieć pola hash_algorithm – przechowujące identyfikator zastosowanej funkcji haszującej (np. "bcrypt", "sha256") wraz z jej parametrami (koszt, rozmiar pamięci, równoległość), co umożliwia w przyszłości migrację na silniejsze algorytmy bez konieczności jednoczesnej zmiany dla wszystkich użytkowników; oraz last_changed – znacznik czasu rejestrujący ostatnią modyfikację danych uwierzytelniających, przydatny z punktu widzenia polityki rotacji haseł i audytu bezpieczeństwa. Taka struktura zapewnia pełną elastyczność w zakresie zarządzania danymi kryptograficznymi, jednocześnie respektując zasadę minimalnego dostępu – aplikacja odpowiedzialna za uwierzytelnianie może mieć pełny dostęp do tabeli credentials, podczas gdy inne moduły systemu operują wyłącznie na tabeli users, nigdy nie mając bezpośredniego kontaktu z danymi wrażliwymi.

Serwer

Skoro na poprzednich przedmiotach wykorzystywany był XAMPP do pracy z bazą danych pozostaniemy w tej grupie technologii i przygotujemy serwer w języku PHP. Proszę jednak nie skupiać się na tym języku samym w sobie, a na zastosowanych mechanizmach związanych z kryptografią.

Warstwa serwerowa realizowana w języku PHP stanowi kluczowy element architektury, pełniąc rolę pośrednika między aplikacją kliencką w C# a bazą danych MariaDB. Serwer udostępnia zestaw endpointów REST API, które przyjmują żądania w formacie JSON i zwracają odpowiedzi w analogicznej strukturze. Pierwszym z nich jest endpoint rejestracji dostępny pod ścieżką POST /api/register.php, który przyjmuje dane użytkownika – nazwę użytkownika, adres e-mail oraz hasło. Po otrzymaniu żądania serwer przeprowadza walidację poprawności adresu e-mail (sprawdzając jego format oraz unikalność w bazie danych) oraz spełnianie przez hasło minimalnych wymagań bezpieczeństwa (długość, złożoność). W przypadku pozytywnej weryfikacji serwer generuje kryptograficznie bezpieczną sól o odpowiedniej długości, a następnie oblicza hash hasła przy użyciu algorytmu bcrypt (poprzez wbudowaną funkcję password_hash(), która automatycznie osadza sól w wyniku) lub alternatywnie SHA-256 z ręcznym dołączeniem soli. Po zapisaniu rekordu w tabelach users i credentials (z zachowaniem transakcyjności i relacji jeden do jednego) serwer zwraca kod 201 Created wraz z komunikatem potwierdzającym rejestrację, bez ujawniania jakichkolwiek danych uwierzytelniających.

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Endpoint logowania POST /api/login.php realizuje proces weryfikacji poświadczeń i generowania tokena JWT. Po przyjęciu nazwy użytkownika (lub adresu e-mail) oraz hasła serwer odczytuje z bazy danych odpowiadający mu wpis, pobierając sól i hash. Następnie oblicza hash dla podanego hasła z użyciem przechowywanej soli i porównuje go z wartością w bazie – przy zastosowaniu bcrypt funkcja password_verify() dokonuje tego porównania automatycznie, obsługując także weryfikację soli. Po potwierdzeniu poprawności poświadczeń serwer generuje token JWT zawierający w swoim payloadzie identyfikator użytkownika, nazwę użytkownika oraz znacznik czasu ważności (ustalony na przykład na 24 godziny). Token jest podpisywany przy użyciu tajnego klucza przechowywanego poza głównym kodem źródłowym (np. w zmiennej środowiskowej) algorytmem HS256. Odpowiedź zwracana do klienta zawiera wyłącznie wygenerowany token – żadne dane uwierzytelniające ani informacje o użytkowniku nie są przekazywane w ciele odpowiedzi poza tym tokenem. Trzeci endpoint GET /api/user.php stanowi przykład zasobu chronionego – wymaga dołączenia tokena JWT w nagłówku Authorization: Bearer <token>. Po odebraniu żądania serwer weryfikuje poprawność podpisu tokena, sprawdza jego ważność czasową, a następnie odczytuje z bazy danych adres e-mail powiązany z identyfikatorem użytkownika zawartym w tokenie. Odpowiedź zawiera adres e-mail w formacie JSON, demonstrując w ten sposób, że klient po pomyślnym uwierzytelnieniu ma dostęp do chronionych zasobów. Wszystkie endpointy implementują mechanizmy zabezpieczające przed atakami typu SQL Injection poprzez wykorzystanie przygotowanych zapytań (prepared statements) z użyciem rozszerzenia MySQLi lub PDO, a także stosują odpowiednie nagłówki CORS umożliwiające komunikację z aplikacją kliencką w C#.

Konfiguracja – config.php

Definiując połączenie do bazy danych należy podać odpowiednie parametry, to samo dla secret w JWT.

```
<?php
// config.php - Konfiguracja połączenia z bazą danych i klucza JWT

// Ustawienia bazy danych
define('DB_HOST', 'localhost');
define('DB_NAME', 'auth_system');
define('DB_USER', 'root');
define('DB_PASS', '');
```

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

```
// Klucz tajny do podpisywania tokenów JWT (w produkcji należy przechowywać
poza kodem, np. w zmiennej środowiskowej)
define('JWT_SECRET', 'your-secret-key-minimum-32-characters-long');

// Funkcja zwracająca połączenie z bazą danych (PDO)
function getDBConnection() {
    try {
        $dsn = 'mysql:host=' . DB_HOST . ';dbname=' . DB_NAME .
';charset=utf8mb4';
        $pdo = new PDO($dsn, DB_USER, DB_PASS);
        $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        $pdo->setAttribute(PDO::ATTR_DEFAULT_FETCH_MODE, PDO::FETCH_ASSOC);
        return $pdo;
    } catch (PDOException $e) {
        // W przypadku błędu połączenia zwracamy kod 500
        http_response_code(500);
        echo json_encode(['error' => 'Database connection failed']);
        exit;
    }
}

// Funkcja pomocnicza do generowania odpowiedzi JSON
function jsonResponse($data, $statusCode = 200) {
    http_response_code($statusCode);
    header('Content-Type: application/json');
    echo json_encode($data);
    exit;
}

// Ustawienie nagłówków CORS (dla komunikacji z aplikacją C#)
header('Access-Control-Allow-Origin: *');
header('Access-Control-Allow-Methods: GET, POST, OPTIONS');
header('Access-Control-Allow-Headers: Content-Type, Authorization');
header('Content-Type: application/json');

// Obsługa preflight request (OPTIONS)
if ($_SERVER['REQUEST_METHOD'] === 'OPTIONS') {
    http_response_code(200);
    exit;
}
?>
```

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Rejestracja użytkownika – register.php

```
<?php
// register.php - Endpoint rejestracji nowego użytkownika
// Metoda: POST
// Parametry (JSON): username, email, password

require_once 'config.php';

// Akceptujemy tylko metodę POST
if ($_SERVER['REQUEST_METHOD'] !== 'POST') {
    jsonResponse(['error' => 'Method not allowed'], 405);
}

// Odczytujemy dane wejściowe
$input = json_decode(file_get_contents('php://input'), true);

// Walidacja obecności wymaganych pól
if (!isset($input['username']) || !isset($input['email']) ||
!isset($input['password'])) {
    jsonResponse(['error' => 'Missing required fields: username, email,
password'], 400);
}

$username = trim($input['username']);
$email = trim($input['email']);
$password = $input['password'];

// Walidacja formatu adresu e-mail
if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
    jsonResponse(['error' => 'Invalid email format'], 400);
}

// Walidacja długości i złożoności hasła (przykładowe minimalne wymagania)
if (strlen($password) < 8) {
    jsonResponse(['error' => 'Password must be at least 8 characters
long'], 400);
}

$pdo = getDBConnection();

// Sprawdzamy, czy użytkownik o podanym username lub email już istnieje
$stmt = $pdo->prepare('SELECT id FROM users WHERE username = :username OR
email = :email');
$stmt->execute([':username' => $username, ':email' => $email]);
```

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

```
if ($stmt->fetch()) {
    jsonResponse(['error' => 'Username or email already exists'], 409);
}

// Generujemy hash hasła przy użyciu bcrypt (funkcja automatycznie dodaje
sól)
// Koszt 12 jest dobrym kompromisem między bezpieczeństwem a wydajnością
$passwordHash = password_hash($password, PASSWORD_BCRYPT, ['cost' => 12]);

// Rozpoczynamy transakcję - zapisujemy dane w dwóch tabelach
try {
    $pdo->beginTransaction();

    // 1. Wstawiamy dane użytkownika do tabeli users
    $stmt = $pdo->prepare('
        INSERT INTO users (username, email, created_at, is_active)
        VALUES (:username, :email, NOW(), 1)
    ');
    $stmt->execute([
        ':username' => $username,
        ':email' => $email
    ]);

    $userId = $pdo->lastInsertId();

    // 2. Wstawiamy dane uwierzytelniające do tabeli credentials
    // Uwaga: bcrypt przechowuje sól w samym hashu, więc osobne pole salt
nie jest potrzebne
    $stmt = $pdo->prepare('
        INSERT INTO credentials (user_id, password_hash, hash_algorithm,
last_changed)
        VALUES (:user_id, :password_hash, "bcrypt", NOW())
    ');
    $stmt->execute([
        ':user_id' => $userId,
        ':password_hash' => $passwordHash
    ]);

    $pdo->commit();

    // Rejestracja zakończona sukcesem - nie zwracamy żadnych danych
uwierzytelniających
    jsonResponse(['message' => 'User registered successfully'], 201);

} catch (Exception $e) {
    $pdo->rollBack();
```


Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

```
    jsonResponse(['error' => 'Registration failed: ' . $e->getMessage()],
500);
}
?>
```

Logowanie i generowanie JWT – login.php

```
<?php
// login.php - Endpoint logowania i generowania tokena JWT
// Metoda: POST
// Parametry (JSON): username (lub email), password

require_once 'config.php';

// Akceptujemy tylko metodę POST
if ($_SERVER['REQUEST_METHOD'] !== 'POST') {
    jsonResponse(['error' => 'Method not allowed'], 405);
}

$input = json_decode(file_get_contents('php://input'), true);

if (!isset($input['username']) || !isset($input['password'])) {
    jsonResponse(['error' => 'Missing username or password'], 400);
}

$login = trim($input['username']);  // może być username lub email
$password = $input['password'];

$pdo = getDBConnection();

// Pobieramy dane użytkownika wraz z danymi uwierzytelniającymi
// Łączymy tabele users i credentials po user_id
$stmt = $pdo->prepare('
    SELECT u.id, u.username, u.email, u.is_active, c.password_hash
    FROM users u
    INNER JOIN credentials c ON u.id = c.user_id
    WHERE u.username = :login OR u.email = :login
');
$stmt->execute([':login' => $login]);
$user = $stmt->fetch();

// Sprawdzamy, czy użytkownik istnieje
if (!$user) {
    jsonResponse(['error' => 'Invalid credentials'], 401);
```

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

```
}

// Sprawdzamy, czy konto jest aktywne
if (!$user['is_active']) {
    jsonResponse(['error' => 'Account is disabled'], 403);
}

// Weryfikujemy hasło przy użyciu password_verify (obsługuje bcrypt
automatycznie)
if (!password_verify($password, $user['password_hash'])) {
    jsonResponse(['error' => 'Invalid credentials'], 401);
}

// Generujemy token JWT
// Payload tokena zawiera dane identyfikujące użytkownika
$issuedAt = time();
$expirationTime = $issuedAt + (24 * 60 * 60); // Token ważny 24 godziny

$payload = [
    'user_id' => $user['id'],
    'username' => $user['username'],
    'iat' => $issuedAt,      // issued at - czas wystawienia
    'exp' => $expirationTime  // expiration time - czas wygaśnięcia
];

// Nagłówek JWT (algorytm HS256, typ JWT)
$header = json_encode(['alg' => 'HS256', 'typ' => 'JWT']);

// Zakodowanie nagłówka i payloadu w Base64URL
$base64UrlHeader = str_replace(['+', '/', '='], ['-', '_', ''],
base64_encode($header));
$base64UrlPayload = str_replace(['+', '/', '='], ['-', '_', ''],
base64_encode(json_encode($payload)));

// Wygenerowanie podpisu
$signature = hash_hmac('sha256', $base64UrlHeader . '.' .
$base64UrlPayload, JWT_SECRET, true);
$base64UrlSignature = str_replace(['+', '/', '='], ['-', '_', ''],
base64_encode($signature));

// Złożenie tokena
$jwt = $base64UrlHeader . '.' . $base64UrlPayload . '.' .
$base64UrlSignature;

// Zwracamy token w odpowiedzi
jsonResponse([
```


Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

```
    'token' => $jwt,
    'expires_in' => 24 * 60 * 60  // informacja dla klienta o czasie
ważności w sekundach
]);
?>
```

Chroniony endpoint – pobieranie e-maila – user.php

```
<?php
// user.php - Chroniony endpoint zwracający adres e-mail zalogowanego
użytkownika
// Metoda: GET
// Wymaga nagłówka: Authorization: Bearer <token>

require_once 'config.php';

// Akceptujemy tylko metodę GET
if ($_SERVER['REQUEST_METHOD'] !== 'GET') {
    jsonResponse(['error' => 'Method not allowed'], 405);
}

// Pobieramy nagłówek Authorization
$authHeader = $_SERVER['HTTP_AUTHORIZATION'] ??
$_SERVER['REDIRECT_HTTP_AUTHORIZATION'] ?? '';

if (empty($authHeader)) {
    jsonResponse(['error' => 'Authorization header missing'], 401);
}

// Sprawdzamy format: "Bearer <token>"
if (!preg_match('/^Bearer\s+(.*)$/i', $authHeader, $matches)) {
    jsonResponse(['error' => 'Invalid authorization header format.
Expected: Bearer <token>'], 401);
}

$token = $matches[1];

// Walidacja tokena JWT
$tokenParts = explode('.', $token);
if (count($tokenParts) !== 3) {
    jsonResponse(['error' => 'Invalid token format'], 401);
}
```


Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

```
list($base64UrlHeader, $base64UrlPayload, $base64UrlSignature) =
$tokenParts;

// Weryfikacja podpisu
$signature = base64_decode(str_replace(['-', '_'], ['+', '/'],
$base64UrlSignature));
$expectedSignature = hash_hmac('sha256', $base64UrlHeader . '.' .
$base64UrlPayload, JWT_SECRET, true);

if (!hash_equals($signature, $expectedSignature)) {
    jsonResponse(['error' => 'Invalid token signature'], 401);
}

// Dekodowanie payloadu
$payloadJson = base64_decode(str_replace(['-', '_'], ['+', '/'],
$base64UrlPayload));
$payload = json_decode($payloadJson, true);

// Sprawdzenie ważności czasowej tokena
$currentTime = time();
if (isset($payload['exp']) && $payload['exp'] < $currentTime) {
    jsonResponse(['error' => 'Token expired'], 401);
}

// Sprawdzenie, czy token zawiera wymagane dane
if (!isset($payload['user_id'])) {
    jsonResponse(['error' => 'Invalid token payload'], 401);
}

$userId = $payload['user_id'];

// Pobieramy adres e-mail z bazy danych
$pdo = getDBConnection();
$stmt = $pdo->prepare('SELECT email FROM users WHERE id = :id AND is_active
= 1');
$stmt->execute([':id' => $userId]);
$user = $stmt->fetch();

if (!$user) {
    jsonResponse(['error' => 'User not found or inactive'], 404);
}

// Zwracamy adres e-mail (zasób chroniony)
jsonResponse([
    'email' => $user['email'],
    'message' => 'Access granted to protected resource'
```


Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

```
]);
?>
```

Struktura bazy danych – przed uruchomieniem skryptów należy utworzyć bazę danych i tabele zgodnie z wcześniejszym opisem (tabela users i credentials z relacją 1:1).

Klucz JWT – w pliku config.php zdefiniowano stałą JWT_SECRET. W środowisku produkcyjnym klucz powinien być przechowywany poza kodem źródłowym, na przykład w pliku .env lub zmiennej środowiskowej.

Bezpieczeństwo – wszystkie zapytania do bazy danych wykorzystują przygotowane instrukcje (prepared statements), co eliminuje ryzyko ataków SQL Injection.

CORS – nagłówki CORS zostały skonfigurowane w pliku config.php tak, aby umożliwić komunikację z aplikacją kliencką w C#. W środowisku produkcyjnym warto zawęzić Access-Control-Allow-Origin do konkretnego adresu.

Błędy – skrypty zwracają kody HTTP zgodne z konwencjami REST (200 OK, 201 Created, 400 Bad Request, 401 Unauthorized, 403 Forbidden, 404 Not Found, 405 Method Not Allowed, 409 Conflict, 500 Internal Server Error).

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Aplikacja klienta

Struktura projektu

```
AuthClient/
├── Program.cs          – Główna pętla programu i interfejs użytkownika
├── Models/
│   └── AuthModels.cs   – Klasy do serializacji/deserializacji JSON
├── Services/
│   └── AuthService.cs  – Logika komunikacji z API
└── Utils/
    └── TokenStorage.cs – Zarządzanie przechowywaniem tokena (w pamięci)
```

Modele danych – Models/AuthModels.cs

```
using System.Text.Json.Serialization;

namespace AuthClient.Models
{
    // Model żądania rejestracji
    public class RegisterRequest
    {
        [JsonPropertyName("username")]
        public string Username { get; set; } = string.Empty;

        [JsonPropertyName("email")]
        public string Email { get; set; } = string.Empty;

        [JsonPropertyName("password")]
        public string Password { get; set; } = string.Empty;
    }

    // Model żądania logowania
    public class LoginRequest
    {
        [JsonPropertyName("username")]
        public string Username { get; set; } = string.Empty;

        [JsonPropertyName("password")]
        public string Password { get; set; } = string.Empty;
    }

    // Model odpowiedzi logowania
    public class LoginResponse
    {
```

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

```
        [JsonPropertyName("token")]
        public string Token { get; set; } = string.Empty;

        [JsonPropertyName("expires_in")]
        public int ExpiresIn { get; set; }
    }

    // Model odpowiedzi dla chronionego endpointu user.php
    public class UserResponse
    {
        [JsonPropertyName("email")]
        public string Email { get; set; } = string.Empty;

        [JsonPropertyName("message")]
        public string Message { get; set; } = string.Empty;
    }

    // Uniwersalna odpowiedź błędu (używana przez wszystkie endpointy)
    public class ErrorResponse
    {
        [JsonPropertyName("error")]
        public string Error { get; set; } = string.Empty;
    }

    // Odpowiedź rejestracji
    public class RegisterResponse
    {
        [JsonPropertyName("message")]
        public string Message { get; set; } = string.Empty;
    }
}
```

Przechowywanie tokena – Utils/TokenStorage.cs

```
namespace AuthClient.Utils
{
    // Klasa statyczna do przechowywania tokena JWT w pamięci aplikacji
    // W aplikacji konsolowej token jest przechowywany w zmiennej
statycznej
    public static class TokenStorage
    {
        private static string? _token;

        public static string? Token
```


Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

```
        {
            get => _token;
            set => _token = value;
        }

        public static bool IsAuthenticated =>
!string.IsNullOrEmpty(_token);

        public static void Clear()
        {
            _token = null;
        }
    }
}
```

Serwis komunikacji z API – Services/AuthService.cs

```
using System;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Text;
using System.Text.Json;
using System.Threading.Tasks;
using AuthClient.Models;
using AuthClient.Utils;

namespace AuthClient.Services
{
    public class AuthService
    {
        private readonly HttpClient _httpClient;
        private readonly string _baseUrl;

        public AuthService(string baseUrl)
        {
            _baseUrl = baseUrl.EndsWith('/') ? baseUrl : baseUrl + "/";
            _httpClient = new HttpClient();
            _httpClient.BaseAddress = new Uri(_baseUrl);
            _httpClient.DefaultRequestHeaders.Accept.Add(
                new MediaTypeWithQualityHeaderValue("application/json"));
        }

        /// <summary>
        /// Rejestracja nowego użytkownika
```


Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

```
        /// </summary>
        /// <returns>True jeśli rejestracja powiodła się, false w przypadku
błędu</returns>
        public async Task<(bool success, string message)>
RegisterAsync(string username, string email, string password)
        {
            try
            {
                var request = new RegisterRequest
                {
                    Username = username,
                    Email = email,
                    Password = password
                };

                var json = JsonSerializer.Serialize(request);
                var content = new StringContent(json, Encoding.UTF8,
"application/json");

                var response = await _httpClient.PostAsync("register.php",
content);
                var responseContent = await
response.Content.ReadAsStringAsync();

                if (response.IsSuccessStatusCode)
                {
                    var registerResponse =
JsonSerializer.Deserialize<RegisterResponse>(responseContent,
                        new JsonSerializerOptions {
PropertyNameCaseInsensitive = true });
                    return (true, registerResponse?.Message ??
"Registration successful");
                }
                else
                {
                    var error =
JsonSerializer.Deserialize<ErrorResponse>(responseContent,
                        new JsonSerializerOptions {
PropertyNameCaseInsensitive = true });
                    return (false, error?.Error ?? $"HTTP Error:
{response.StatusCode}");
                }
            }
            catch (Exception ex)
            {
                return (false, $"Connection error: {ex.Message}");
```

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

```
            }
        }

        /// <summary>
        /// Logowanie użytkownika – pobiera token JWT i zapisuje go w
pamięci
        /// </summary>
        /// <returns>True jeśli logowanie powiodło się, false w przypadku
błędu</returns>
        public async Task<(bool success, string message)> LoginAsync(string
username, string password)
        {
            try
            {
                var request = new LoginRequest
                {
                    Username = username,
                    Password = password
                };

                var json = JsonSerializer.Serialize(request);
                var content = new StringContent(json, Encoding.UTF8,
"application/json");

                var response = await _httpClient.PostAsync("login.php",
content);
                var responseContent = await
response.Content.ReadAsStringAsync();

                if (response.IsSuccessStatusCode)
                {
                    var loginResponse =
JsonSerializer.Deserialize<LoginResponse>(responseContent,
                        new JsonSerializerOptions {
PropertyNameCaseInsensitive = true });

                    if (loginResponse != null &&
!string.IsNullOrEmpty(loginResponse.Token))
                    {
                        // Zapisujemy token w pamięci
                        TokenStorage.Token = loginResponse.Token;
                        return (true, $"Login successful. Token expires in
{loginResponse.ExpiresIn} seconds.");
                    }
                    return (false, "Invalid response from server");
                }
```


Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

```
                else
                {
                    var error =
JsonSerializer.Deserialize<ErrorResponse>(responseContent,
                        new JsonSerializerOptions {
PropertyNameCaseInsensitive = true });
                    return (false, error?.Error ?? $"HTTP Error:
{response.StatusCode}");
                }
            }
            catch (Exception ex)
            {
                return (false, $"Connection error: {ex.Message}");
            }
        }

        /// <summary>
        /// Pobiera adres e-mail zalogowanego użytkownika (endpoint
chroniony)
        /// </summary>
        /// <returns>Adres e-mail lub komunikat błędu</returns>
        public async Task<(bool success, string email, string message)>
GetUserEmailAsync()
        {
            // Sprawdzamy, czy token istnieje
            if (string.IsNullOrEmpty(TokenStorage.Token))
            {
                return (false, string.Empty, "Not authenticated. Please
login first.");
            }

            try
            {
                // Dodajemy token do nagłówka Authorization
                _httpClient.DefaultRequestHeaders.Authorization =
                    new AuthenticationHeaderValue("Bearer",
TokenStorage.Token);

                var response = await _httpClient.GetAsync("user.php");
                var responseContent = await
response.Content.ReadAsStringAsync();

                if (response.IsSuccessStatusCode)
                {
                    var userResponse =
JsonSerializer.Deserialize<UserResponse>(responseContent,
```


Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

```
                        new JsonSerializerOptions {
PropertyNameCaseInsensitive = true });

                    return (true, userResponse?.Email ?? string.Empty,
userResponse?.Message ?? "Success");
                }
                else
                {
                    var error =
JsonSerializer.Deserialize<ErrorResponse>(responseContent,
                        new JsonSerializerOptions {
PropertyNameCaseInsensitive = true });

                    // Jeśli token jest nieprawidłowy lub wygasł, czyścimy
go
                    if (response.StatusCode ==
System.Net.HttpStatusCode.Unauthorized)
                    {
                        TokenStorage.Clear();
                        return (false, string.Empty, "Session expired.
Please login again.");
                    }

                    return (false, string.Empty, error?.Error ?? $"HTTP
Error: {response.StatusCode}");
                }
            }
            catch (Exception ex)
            {
                return (false, string.Empty, $"Connection error:
{ex.Message}");
            }
            finally
            {
                // Usuwamy nagłówek Authorization po zakończeniu żądania
(dobra praktyka)
                _httpClient.DefaultRequestHeaders.Authorization = null;
            }
        }

        /// <summary>
        /// Wylogowanie - czyści token z pamięci
        /// </summary>
        public void Logout()
        {
            TokenStorage.Clear();
```


Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

```
        }

        /// <summary>
        /// Sprawdza, czy użytkownik jest aktualnie zalogowany
        /// </summary>
        public bool IsAuthenticated() => TokenStorage.IsAuthenticated;
    }
}
```

Główny program – Program.cs

```
using System;
using System.Threading.Tasks;
using AuthClient.Services;

namespace AuthClient
{
    class Program
    {
        // Adres serwera – należy dostosować do lokalnej konfiguracji XAMPP
        private const string ServerUrl = "http://localhost/auth_api/";

        static async Task Main(string[] args)
        {
            Console.WriteLine("=== Auth Client - System Uwierzytelniania
===\n");

            var authService = new AuthService(ServerUrl);
            bool exit = false;

            while (!exit)
            {
                // Wyświetlenie menu w zależności od stanu uwierzytelnienia
                if (authService.IsAuthenticated())
                {
                    Console.WriteLine("\n--- ZALOGOWANY ---");
                    Console.WriteLine("1. Pobierz mój adres e-mail");
                    Console.WriteLine("2. Wyloguj");
                    Console.WriteLine("3. Wyjdź");
                }
                else
                {
                    Console.WriteLine("\n--- NIEZALOGOWANY ---");
                    Console.WriteLine("1. Rejestracja");
```


Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

```
                Console.WriteLine("2. Logowanie");
                Console.WriteLine("3. Wyjdź");
            }

            Console.Write("\nWybierz opcję: ");
            string? choice = Console.ReadLine();

            switch (choice)
            {
                case "1":
                    if (authService.IsAuthenticated())
                        await GetUserEmailAsync(authService);
                    else
                        await RegisterAsync(authService);
                    break;
                case "2":
                    if (authService.IsAuthenticated())
                        Logout(authService);
                    else
                        await LoginAsync(authService);
                    break;
                case "3":
                    exit = true;
                    Console.WriteLine("Do widzenia!");
                    break;
                default:
                    Console.WriteLine("Nieprawidłowa opcja. Spróbuj
ponownie.");
                    break;
            }
        }
    }

    static async Task RegisterAsync(AuthService authService)
    {
        Console.WriteLine("\n--- REJESTRACJA NOWEGO UŻYTKOWNIKA ---");

        Console.Write("Podaj nazwę użytkownika: ");
        string? username = Console.ReadLine();

        Console.Write("Podaj adres e-mail: ");
        string? email = Console.ReadLine();

        Console.Write("Podaj hasło (minimum 8 znaków): ");
        string? password = ReadPassword(); // Maskowanie hasła
```


Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

```
            if (string.IsNullOrWhiteSpace(username) ||
string.IsNullOrWhiteSpace(email) || string.IsNullOrWhiteSpace(password))
            {
                Console.WriteLine("Błąd: Wszystkie pola są wymagane.");
                return;
            }

            Console.WriteLine("\nRejestracja w toku...");

            var (success, message) = await
authService.RegisterAsync(username, email, password);

            if (success)
            {
                Console.ForegroundColor = ConsoleColor.Green;
                Console.WriteLine($"Sukces: {message}");
            }
            else
            {
                Console.ForegroundColor = ConsoleColor.Red;
                Console.WriteLine($"Błąd: {message}");
            }
            Console.ResetColor();

            Console.WriteLine("\nNaciśnij dowolny klawisz, aby
kontynuować...");
            Console.ReadKey();
        }

        static async Task LoginAsync(AuthService authService)
        {
            Console.WriteLine("\n--- LOGOWANIE ---");

            Console.Write("Podaj nazwę użytkownika lub adres e-mail: ");
            string? username = Console.ReadLine();

            Console.Write("Podaj hasło: ");
            string? password = ReadPassword();

            if (string.IsNullOrWhiteSpace(username) ||
string.IsNullOrWhiteSpace(password))
            {
                Console.WriteLine("Błąd: Nazwa użytkownika i hasło są
wymagane.");
                return;
            }
```


Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

```
            Console.WriteLine("\nLogowanie w toku...");

            var (success, message) = await authService.LoginAsync(username,
password);

            if (success)
            {
                Console.ForegroundColor = ConsoleColor.Green;
                Console.WriteLine($"Sukces: {message}");
                Console.WriteLine("Token został zapisany w pamięci.");
            }
            else
            {
                Console.ForegroundColor = ConsoleColor.Red;
                Console.WriteLine($"Błąd: {message}");
            }
            Console.ResetColor();

            Console.WriteLine("\nNaciśnij dowolny klawisz, aby
kontynuować...");
            Console.ReadKey();
        }

        static async Task GetUserEmailAsync(AuthService authService)
        {
            Console.WriteLine("\n--- POBIERANIE DANYCH UŻYTKOWNIKA ---");
            Console.WriteLine("Wysyłanie żądania z tokenem JWT...");

            var (success, email, message) = await
authService.GetUserEmailAsync();

            if (success)
            {
                Console.ForegroundColor = ConsoleColor.Green;
                Console.WriteLine($"Sukces: {message}");
                Console.WriteLine($"Adres e-mail zalogowanego użytkownika:
{email}");
            }
            else
            {
                Console.ForegroundColor = ConsoleColor.Red;
                Console.WriteLine($"Błąd: {message}");
            }
            Console.ResetColor();
```


Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

```
            Console.WriteLine("\nNaciśnij dowolny klawisz, aby
kontynuować...");
            Console.ReadKey();
        }

        static void Logout(AuthService authService)
        {
            authService.Logout();
            Console.ForegroundColor = ConsoleColor.Yellow;
            Console.WriteLine("Wylogowano pomyślnie. Token został usunięty
z pamięci.");
            Console.ResetColor();

            Console.WriteLine("\nNaciśnij dowolny klawisz, aby
kontynuować...");
            Console.ReadKey();
        }

        /// <summary>
        /// Wczytuje hasło z konsoli bez wyświetlania znaków (maskowanie)
        /// </summary>
        static string ReadPassword()
        {
            string password = string.Empty;
            ConsoleKeyInfo key;

            do
            {
                key = Console.ReadKey(true);

                // Backspace – usuwa ostatni znak
                if (key.Key == ConsoleKey.Backspace && password.Length > 0)
                {
                    password = password[0..^1];
                    Console.Write("\b \b");
                }
                // Normalny znak – dodaje do hasła
                else if (!char.IsControl(key.KeyChar))
                {
                    password += key.KeyChar;
                    Console.Write("*");
                }
            }
            while (key.Key != ConsoleKey.Enter);

            Console.WriteLine();
```


Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

```
            return password;
        }
    }
}
```

Maskowanie hasła – funkcja ReadPassword() demonstruje, jak bezpiecznie wczytywać hasła w aplikacjach konsolowych.

Obsługa błędów – aplikacja przechwytuje wyjątki sieciowe i błędy HTTP, zwracając czytelne komunikaty.

Zarządzanie tokenem – token jest przechowywany w pamięci (nie na dysku), co jest bezpieczniejsze niż zapis w pliku tekstowym.

Nagłówki HTTP – dodawanie tokena w nagłówku Authorization pokazuje standardowy sposób przekazywania poświadczeń w REST API.

Ćwiczenia samodzielne

Zadanie 6.1

Stwórz w MariaDB bazę danych auth_system według wskazówek z instrukcji.

Zadanie 6.2

Skonfiguruj serwer PHP:

1. Umieść pliki config.php, register.php, login.php, user.php w katalogu dostępnym przez serwer XAMPP (np. htdocs/auth_api/).
2. W pliku config.php dostosuj dane połączenia z bazą (nazwa użytkownika, hasło).

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Zadanie 6.3

Konfiguracja klienta C#:

1. Utwórz nowy projekt konsolowy w Visual Studio.
2. Dodaj klasy zgodnie z powyższą strukturą.
3. W Program.cs dostosować stałą ServerUrl do adresu, pod którym dostępny jest serwer PHP (np. http://localhost/auth_api/).

Zadanie 6.4

Przetestuj działanie aplikacji:

1. Uruchom aplikację konsolową.
2. Zarejestruj nowego użytkownika. Sprawdź za pomocą phpMyAdmin jak wykonała się ta operacja.
3. Zaloguj się – token zostanie zapisany w pamięci aplikacji.
4. Zmodyfikuj kod, tak aby otrzymany token z odpowiednim komunikatem został wyświetlony na konsolę.
5. Spróbuj zalogować się ze złym hasłem.
6. Pobierz adres e-mail z chronionego endpointu.
7. Wyloguj się i spróbuj ponownie pobrać e-mail (powinien zwrócić błąd autoryzacji).

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Pytania pomocnicze

1. Omów proces autoryzacji użytkownika w serwisie: rejestracja i logowanie.
2. Wyjaśnij znaczenie hashowania haseł w bazie danych.
3. Opisz mechanizm soli, warianty jej tworzenia i zapisywania w bazie danych.
4. Czym jest JWT?

Podsumowanie

Przeprowadzone laboratorium pozwoliło studentom na praktyczne zaprojektowanie i zaimplementowanie kompletnego systemu uwierzytelniania w architekturze klient-serwer, integrującego wiedzę z zakresu kryptografii, bezpieczeństwa aplikacji oraz programowania w środowiskach PHP i C#. Uczestnicy zajęć zdobyli umiejętność bezpiecznego przechowywania danych uwierzytelniających w bazie danych MariaDB z zastosowaniem jednokierunkowego haszowania haseł z użyciem soli (bcrypt), co skutecznie zabezpiecza przed ujawnieniem poświadczeń w przypadku wycieku bazy danych. Ponadto studenci zrealizowali mechanizm uwierzytelniania oparty na tokenach JWT, które po poprawnym logowaniu są generowane po stronie serwera, przekazywane do aplikacji klienckiej w C#, a następnie wykorzystywane do autoryzacji żądań do chronionych zasobów REST API. Dzięki temu ćwiczeniu studenci nie tylko opanowali praktyczne aspekty implementacji bezpiecznego logowania, ale także zrozumieli fundamentalne zasady ochrony danych wrażliwych – od izolacji tabel z danymi kryptograficznymi, przez stosowanie przygotowanych zapytań SQL zapobiegających iniekcjom, aż po prawidłowe zarządzanie sesją i tokenami w aplikacji klienckiej. Uzyskane kompetencje stanowią solidną podstawę do projektowania bezpiecznych systemów informatycznych w rzeczywistych środowiskach produkcyjnych, gdzie ochrona tożsamości użytkowników jest jednym z kluczowych wymagań stawianych współczesnym aplikacjom sieciowym.

Literatura

1. Jacek Ross, Bezpieczne programowanie. Aplikacje hakeroodporne, Helion 2009

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Materiały do przedmiotu:

Podstawy kryptografii

Laboratorium nr 7

Szyfrowanie plików za pomocą narzędzi online

Autor:

dr Michał Dolecki

Fundusze Europejskie dla Rozwoju Społecznego

Rzeczpospolita Polska

Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Spis treści

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Wprowadzenie, tematyka zajęć

Współczesna kryptografia asymetryczna i symetryczna stanowią fundament bezpieczeństwa danych w systemach teleinformatycznych, jednak z perspektywy inżynierskiej równie istotna jest umiejętność praktycznego zastosowania tych mechanizmów w codziennej pracy z danymi. W ramach dzisiejszych zajęć skoncentrujemy się na zagadnieniu szyfrowania plików w stanie spoczynku (*encryption at rest*) – jednym z kluczowych elementów ochrony poufności informacji. Należy podkreślić fundamentalną zasadę, że bezpieczeństwo kryptograficzne nigdy nie powinno opierać się wyłącznie na zaufaniu do pojedynczego narzędzia czy usługi, lecz na zrozumieniu stosowanych algorytmów, zarządzaniu kluczami oraz świadomym modelowaniu zagrożeń. W praktyce inżynierskiej oznacza to konieczność odróżniania sytuacji, w których wystarczające jest szyfrowanie pojedynczych archiwów (np. przy przesyłaniu plików), od przypadków wymagających pełnego szyfrowania wolumenów czy całych systemów.

W warstwie aplikacyjnej zaprezentowane zostały zarówno narzędzia lokalne, jak i usługi online, z pełną świadomością różnic w modelu bezpieczeństwa. W ekosystemie Windows naturalnym punktem startowym jest wykorzystanie powszechnie dostępnego programu 7-Zip, który implementuje szyfrowanie AES-256 dla archiwów ZIP i 7z – jest to przykład kryptografii symetrycznej, gdzie klucz pochodny od hasła podlega wielokrotnej funkcji skrótu. Kolejnym, istotnie bardziej zaawansowanym narzędziem jest VeraCrypt, umożliwiający tworzenie zaszyfrowanych kontenerów oraz szyfrowanie całych partycji w trybie *full-disk encryption*. W przeciwieństwie do archiwów, VeraCrypt stosuje mechanizmy *deniability* (możliwość zaprzeczenia) oraz wieloetapowe uwierzytelnianie, co ukazuje studentom, jak poziom bezpieczeństwa wzrasta wraz ze złożonością rozwiązania kosztem wygody użytkowania.

Usługi online wprowadzają dodatkowy wymiar analizy bezpieczeństwa – kwestię zaufania do strony trzeciej, bezpieczeństwa transmisji danych oraz polityki zarządzania kluczami. W trakcie laboratoriów przeanalizujemy wybrane, sprawdzone narzędzia webowe, które implementują model *client-side encryption* (szyfrowanie po stronie klienta przed przesłaniem danych na serwer), co minimalizuje ryzyko nieautoryzowanego dostępu ze strony operatora usługi. Zwrócimy szczególną uwagę na identyfikację algorytmów kryptograficznych, metodę generowania kluczy oraz fakt, czy narzędzie udostępnia swój kod źródłowy

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

do audytu. Ta część zajęć ma na celu wykształcenie u studentów krytycznego podejścia do rozwiązań online – kluczowej kompetencji w pracy współczesnego specjalisty ds. bezpieczeństwa, który musi równoważyć wymagania funkcjonalne z ryzykiem wynikającym z utraty kontroli nad kluczami kryptograficznymi.

Cel laboratorium

Po zakończeniu dwugodzinnych zajęć student potrafi samodzielnie dobrać odpowiednią metodę szyfrowania plików w zależności od kontekstu operacyjnego – wykorzystując narzędzia lokalne (7-Zip, VeraCrypt) oraz wybrane usługi online z szyfrowaniem po stronie klienta – ze szczególnym uwzględnieniem identyfikacji stosowanych algorytmów kryptograficznych, mechanizmów pochodnych klucza oraz modelu zaufania wobec dostawcy rozwiązania; student świadomie rozróżnia bezpieczeństwo archiwum szyfrowanego od szyfrowania wolumenowego, potrafi oszacować ryzyko wynikające z transferu danych do usług online oraz wykazuje umiejętność weryfikacji, czy wybrane narzędzie realizuje szyfrowanie lokalnie przed transmisją, co stanowi podstawę do świadomego podejmowania decyzji inżynierskich w zakresie ochrony danych w stanie spoczynku.

Instrukcja laboratoryjna

Tworzenie zaszyfrowanych archiwów ZIP w programie 7-Zip

Cel zadania

Celem jest nabycie przez studenta umiejętności praktycznego zastosowania kryptografii symetrycznej w ochronie danych w stanie spoczynku z wykorzystaniem powszechnie dostępnego narzędzia 7-Zip. Student po wykonaniu ćwiczenia potrafi samodzielnie utworzyć zaszyfrowane archiwum ZIP z zastosowaniem algorytmu AES-256, świadomie skonfigurować parametry szyfrowania w zależności od wymaganego poziomu bezpieczeństwa oraz zrozumieć znaczenie bezpiecznego zarządzania hasłem w kontekście przesyłania zaszyfrowanych danych za pośrednictwem poczty elektronicznej i innych kanałów komunikacji.

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Wymagania wstępne

- Zainstalowany program 7-Zip (dostępny do pobrania z oficjalnej strony https://www.7-zip.org)
- System operacyjny Windows
- Dowolny plik tekstowy lub folder z dokumentami do zaszyfrowania

Wprowadzenie teoretyczne

7-Zip wykorzystuje algorytm AES-256 (Advanced Encryption Standard z kluczem 256-bitowym) do szyfrowania archiwów, który jest uznawany za bezpieczny nawet w zastosowaniach profesjonalnych i nie został oficjalnie skompromitowany. W przypadku formatu ZIP, 7-Zip implementuje szyfrowanie AES-256, co stanowi istotne udoskonalenie w stosunku do starszego, podatnego na ataki algorytmu ZipCrypto stosowanego domyślnie w wielu innych narzędziach. Należy podkreślić fundamentalną zasadę: skuteczność szyfrowania nie zależy wyłącznie od algorytmu, lecz przede wszystkim od siły hasła – musi ono charakteryzować się odpowiednią entropią (długość minimum 12-15 znaków, zawierać małe i wielkie litery, cyfry oraz znaki specjalne).

W kontekście bezpieczeństwa transmisji danych istotne jest, aby hasło do zaszyfrowanego archiwum przekazywać odbiorcy innym kanałem komunikacji niż sam plik – nigdy w tej samej wiadomości e-mail ani w ramach tego samego komunikatora . Dodatkowo, w przypadku najwyższych wymagań poufności, zaleca się stosowanie formatu 7z zamiast ZIP, który umożliwia również szyfrowanie nazw plików, co uniemożliwia potencjalnemu atakującemu zapoznanie się nawet ze strukturą zaszyfrowanych danych .

Tworzenie zaszyfrowanego archiwum ZIP

Krok 1. W Eksploratorze plików Windows zlokalizuj plik lub folder przeznaczony do zaszyfrowania.

Krok 2. Kliknij prawym przyciskiem myszy na wybrany obiekt, a następnie z menu kontekstowego wybierz opcję **7-Zip** → **Dodaj do archiwum...** .

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Krok 3. W oknie dialogowym *Dodaj do archiwum* dokonaj następującej konfiguracji :

Pole/opcja	Ustawienie	Uzasadnienie
Format archiwum	ZIP	Zapewnia kompatybilność – archiwum można rozpakować bez instalacji 7-Zip
Poziom kompresji	Normalny (lub dowolny)	Bez wpływu na bezpieczeństwo szyfrowania
Metoda szyfrowania	AES-256	Najsilniejszy dostępny algorytm

Krok 4. W sekcji **Szyfrowanie** u dołu okna:

- W polu *Wprowadź hasło* wpisz silne hasło
- W polu *Wprowadź hasło ponownie* powtórz hasło

Uwaga krytyczna: W przypadku formatu ZIP opcja *Szyfruj nazwy plików* jest niedostępna – jest to ograniczenie specyfikacji formatu ZIP. Jeśli wymagane jest również ukrycie nazw plików, należy wybrać format 7z .

Krok 5. Kliknij przycisk **OK**. Po zakończeniu procesu w lokalizacji źródłowej pojawi się plik o rozszerzeniu .zip zabezpieczony hasłem .

Otwieranie zaszyfrowanego archiwum

Dwukrotne kliknięcie na plik ZIP spowoduje wyświetlenie okna z prośbą o podanie hasła. Po jego wprowadzeniu można przeglądać zawartość archiwum, kopiować pliki lub je wypakować . Wypakowanie można również wykonać przez menu kontekstowe: kliknij prawym przyciskiem na archiwum → **7-Zip** → **Wypakuj tutaj** (lub *Wypakuj do ...*) .

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Szyfrowanie plików i katalogów za pomocą VeraCrypt Portable

Cel zadania

Celem jest nabycie przez studenta umiejętności praktycznego zastosowania kryptografii symetrycznej w ochronie danych z wykorzystaniem narzędzia VeraCrypt w wersji przenośnej. Student po wykonaniu ćwiczenia potrafi samodzielnie utworzyć zaszyfrowany kontener plikowy (file container) w środowisku Windows bez konieczności instalacji oprogramowania, zamontować go jako wirtualny dysk, przetransferować do niego wrażliwe dane oraz bezpiecznie odmontować kontener. Student rozumie również różnicę między szyfrowaniem pojedynczego archiwum (7-Zip) a szyfrowaniem wolumenowym oferowanym przez VeraCrypt, w tym korzyści wynikające z szyfrowania w locie (on-the-fly encryption) oraz znaczenie bezpiecznego zarządzania hasłem.

Wymagania wstępne

- System operacyjny Windows 10 lub 11
- Program VeraCrypt ze strony https://veracrypt.io/en/Downloads.html
- Brak konieczności posiadania praw administratora (wersja portable nie wymaga instalacji, a do działania wymaga jedynie uprawnień użytkownika – w przypadku braku uprawnień admina niektóre zaawansowane funkcje mogą być ograniczone)
- Pamięć USB (opcjonalnie, do przenoszenia narzędzia i kontenerów)

Wprowadzenie teoretyczne – VeraCrypt i tryb portable

VeraCrypt to wieloplatformowe, otwartoźródłowe narzędzie do szyfrowania dysków, będące następcą popularnego programu TrueCrypt. W odróżnieniu od szyfrowania archiwów w 7-Zip, VeraCrypt tworzy tzw. kontenery plikowe – pojedyncze pliki, które po zamontowaniu (mount) stają się w systemie operacyjnym wirtualnymi dyskami. Wszystkie dane zapisywane na takim dysku są automatycznie szyfrowane w locie (on-the-fly encryption), a odczytywane – deszyfrowane wyłącznie w pamięci RAM . Oznacza to, że użytkownik nie musi ręcznie szyfrować poszczególnych plików – wystarczy skopiować je do zamontowanego kontenera.

Wersja **VeraCrypt Portable** to specjalna edycja programu, która nie wymaga instalacji – może być uruchamiana bezpośrednio z pamięci USB lub folderu na dysku. W środowisku pracowni komputerowej jest to rozwiązanie idealne, ponieważ:

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

- nie wymaga uprawnień administratora (w podstawowym zakresie – tworzenie i montowanie kontenerów plikowych)
- nie pozostawia śladów w rejestrze systemowym (z pewnymi ograniczeniami opisanymi w dokumentacji)
- umożliwia studentom zabranie narzędzia na własnym pendrive i korzystanie z niego na dowolnym komputerze

VeraCrypt wykorzystuje zaawansowane algorytmy kryptograficzne, w tym AES-256, Serpent, Twofish oraz ich kaskady, a także funkcje skrótu SHA-512, Whirlpool i BLAKE2s. Domyślnie zalecanym wyborem dla kontenerów plikowych jest algorytm AES z funkcją skrótu SHA-512 .

Część praktyczna – przygotowanie VeraCrypt Portable

Pobranie i przygotowanie narzędzia

Krok 1. Pobierz wersję portable VeraCrypt.

Krok 2. Wypakuj archiwum:

- Kliknij prawym przyciskiem myszy na pobrany plik .zip i wybierz opcję **Wypakuj wszystko...**
- Jako lokalizację docelową wskaż folder na dysku lub bezpośrednio na pamięci USB

Krok 3. Uruchom program:

- Otwórz folder, do którego wypakowano pliki
- Dwukrotnie kliknij plik VeraCrypt-x64.exe – program uruchomi się bez procesu instalacji

Tworzenie zaszyfrowanego kontenera plikowego

Krok 1. W głównym oknie VeraCrypt kliknij przycisk menu Wolumeny i **Create Volume** (Utwórz wolumin).

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Krok 2. W oknie Kreatora tworzenia woluminu wybierz opcję **Create an encrypted file container** (Utwórz zaszyfrowany plik) – jest to opcja domyślnie zaznaczona. Kliknij **Next** (Dalej) .

Krok 3. Wybierz typ woluminu: **Standard VeraCrypt volume** (Standardowy wolumin VeraCrypt). Kliknij **Next** .

Krok 4. Wybierz lokalizację i nazwę pliku kontenera:

- Kliknij przycisk **Select File** (Wybierz plik)
- W oknie dialogowym wskaż folder, w którym ma zostać utworzony kontener (np. C:\Users\Student\Documents lub na pamięci USB)
- W polu *File name* (Nazwa pliku) wpisz nazwę kontenera, np. **MojeDane.hc** (rozszerzenie .hc jest konwencją, można użyć dowolnego rozszerzenia zwłaszcza nic niemówiącego .log albo .dat)
- Kliknij **Save** (Zapisz)
- W oknie kreatora kliknij **Next**

Uwaga krytyczna: Jeśli wybierzesz istniejący plik, zostanie on **nadpisany** i utracony – VeraCrypt nie szyfruje istniejących plików, lecz tworzy nowy plik kontenera.

Krok 5. Wybór algorytmów szyfrowania:

- Pozostaw ustawienia domyślne: **AES** (algorytm szyfrowania) oraz **SHA-512** (algorytm haszujący), dla własnego użytku można korzystać z kilku algorytmów działających kaskadowo.
- Kliknij **Next**

Krok 6. Określenie rozmiaru kontenera:

- Wprowadź żądany rozmiar w polu liczbowym (np. **100** dla 100 MB lub **500** dla 500 MB)
- Upewnij się, że jednostką są **MB** (megabajty) lub **GB** (gigabajty)
- Kliknij **Next**

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Krok 7. Ustawienie hasła:

- W polu *Password* (Hasło) wprowadź silne hasło (minimum 12-15 znaków, zawierające małe i wielkie litery, cyfry oraz znaki specjalne)
- W polu *Confirm* (Potwierdź) wprowadź hasło ponownie
- Kliknij **Next**

Wskazówka: VeraCrypt zaleca hasła dłuższe niż 20 znaków i ostrzega przy krótszych hasłach .

Krok 8. Formatowanie kontenera:

- Wybierz system plików: zalecany **exFAT** (umożliwia przechowywanie plików większych niż 4 GB i działa w systemach Windows, macOS i Linux) lub **NTFS** (tylko Windows)
- Przesuwaj myszą losowo po oknie kreatora, aż pasek *Randomness* (Losowość) zmieni kolor z czerwonego na zielony – zwiększa to entropię kluczy szyfrowania
- Kliknij przycisk **Format**

Krok 9. Po zakończeniu formatowania pojawi się okno potwierdzenia. Kliknij **OK**, a następnie **Exit** (Zakończ) .

Część praktyczna – montowanie i używanie kontenera

Krok 1. W głównym oknie VeraCrypt wybierz wolną literę dysku z listy (np. **C:** nie jest dostępne – wybierz np. **M:** lub **X:**) .

Krok 2. Kliknij przycisk **Select File** (Wybierz plik) i wskaż utworzony wcześniej plik kontenera (np. MojeDane.hc) .

Krok 3. Kliknij przycisk **Mount** (Zamontuj) .

Krok 4. W oknie dialogowym wprowadź hasło ustawione podczas tworzenia kontenera. W polu *PRF* pozostaw opcję *autodetection* (automatyczne wykrywanie). Kliknij **OK** .

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Krok 5. Po poprawnym zamontowaniu w głównym oknie VeraCrypt pojawi się informacja o zamontowanym woluminie, a w systemie (Eksplorator plików – *Ten komputer*) pojawi się nowy dysk o wybranej literze .

Krok 6. Używaj zamontowanego dysku jak zwykłego dysku:

- Kopiuj pliki i foldery do dysku – są one automatycznie szyfrowane
- Otwieraj pliki bezpośrednio z dysku – są one automatycznie deszyfrowane w pamięci RAM

Krok 7. Po zakończeniu pracy odmontuj kontener:

- W głównym oknie VeraCrypt zaznacz zamontowany wolumin na liście
- Kliknij przycisk **Unmount** (Odmontuj) lub **Dismount All** (Odmontuj wszystkie)

Ważne: Po zamknięciu komputera lub restarcie kontener jest automatycznie odmontowywany. Aby ponownie uzyskać dostęp do danych, należy powtórzyć czynności montowania .

Zalecenia bezpieczeństwa

1. **Bezpieczeństwo hasła** – hasło do kontenera powinno być unikalne, nieużywane w innych usługach. VeraCrypt nie oferuje mechanizmu odzyskiwania hasła – utrata hasła oznacza trwałą utratę danych .
2. **Przechowywanie kontenera** – plik kontenera można dowolnie przenosić, kopiować i przesyłać. Należy jednak pamiętać, że sam plik nie jest zabezpieczony przed skopiowaniem – bezpieczeństwo zapewnia wyłącznie hasło.
3. **Kopia zapasowa** – regularnie wykonuj kopię zapasową pliku kontenera na innym nośniku. Uszkodzenie pliku kontenera (np. awaria dysku) uniemożliwia odzyskanie danych.
4. **Prawa administratora** – wersja portable może wymagać uprawnień administratora w niektórych scenariuszach (np. przy montowaniu woluminów z określonymi opcjami). W przypadku braku uprawnień kreator tworzenia kontenera i montowanie powinny działać poprawnie .

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Szyfrowanie plików za pomocą narzędzi online

Cel zadania

Celem jest nabycie przez studenta umiejętności krytycznej oceny i praktycznego zastosowania internetowych narzędzi do szyfrowania plików. Student po wykonaniu ćwiczenia potrafi odróżnić usługi realizujące szyfrowanie po stronie klienta (*client-side encryption*) od tych, które przechowują klucze na serwerze, rozumie znaczenie fragmentu URL (części po znaku #) w kontekście bezpieczeństwa transmisji klucza oraz potrafi świadomie korzystać z wybranych, sprawdzonych rozwiązań online do bezpiecznego przesyłania plików. Student rozwija również umiejętność analizy modelu zaufania wobec usługodawcy oraz identyfikacji kluczowych mechanizmów kryptograficznych stosowanych w narzędziach webowych.

Wymagania wstępne

- Komputer z dostępem do Internetu i przeglądarką internetową (zalecane: Firefox, Chrome, Edge)
- Konto e-mail (do testowania mechanizmów wysyłki)
- Podstawowa znajomość obsługi przeglądarki i menedżera plików
- Dwa dowolne pliki testowe (np. dokument tekstowy, obraz) o łącznym rozmiarze poniżej 5 MB

Wprowadzenie teoretyczne – szyfrowanie w chmurze i model zaufania

Narzędzia do szyfrowania plików online można podzielić na dwie zasadnicze kategorie ze względu na miejsce wykonywania operacji kryptograficznych. W przypadku rozwiązań z szyfrowaniem po stronie klienta (*client-side encryption*), dane są szyfrowane bezpośrednio w przeglądarce użytkownika przed wysłaniem ich na serwer – serwer otrzymuje wyłącznie zaszyfrowany *blob* danych, nie mając fizycznej możliwości odczytania ich treści. Klucz deszyfrujący w takich rozwiązaniach często znajduje się we fragmencie URL (część po znaku #), który nigdy nie jest przesyłany do serwera, co stanowi realizację zasady *zero-knowledge*. Z kolei usługi, które nie stosują tego mechanizmu, przechowują klucze na swoich serwerach, co oznacza, że operator usługi – lub osoba, która przejęłaby kontrolę nad serwerem – może potencjalnie odczytać dane użytkownika.

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Drugim kluczowym aspektem jest otwartość kodu źródłowego. Narzędzia, które udostępniają swój kod do audytu (open source), umożliwiają niezależnym specjalistom ds. bezpieczeństwa weryfikację, czy deklarowane mechanizmy szyfrowania są faktycznie implementowane zgodnie z najlepszymi praktykami. W przypadku narzędzi komercyjnych o zamkniętym kodzie źródłowym użytkownik zmuszony jest do polegania wyłącznie na deklaracjach dostawcy. W trakcie niniejszego ćwiczenia studenci zapoznają się z reprezentatywnymi przykładami obu typów rozwiązań, kładąc szczególny nacisk na identyfikację mechanizmów kryptograficznych i świadome zarządzanie kluczami.

Działanie prostej strony do szyfrowania plików można prześledzić na przykładzie

https://www.devglan.com/online-tools/file-encryption-decryption

W ramach zajęć przejdziemy przez szyfrowanie i deszyfrowanie pliku tekstowego.

Krok 1. Utwórz plik tekstowy z krótkim tekstem np. aala ma kotaa.

Krok 2. Otwórz stronę do szyfrowania

Krok 3. Wyślij swój plik przez formularz. Podaj własny ciąg do wygenerowania klucza. Po chwili wygenerowany zostanie zaszyfrowany plik, który można pobrać. Plik ma rozszerzenie .enc. Dodatkowo można skopiować klucz w postaci Base64.

Dla pliku o treści:

aala ma kotaa

Wygenerowano szyfrogram:

#ű- mrŢ2r-9Řâ————————————x-?T’|Açß™————————————
é«¸hyż[aş”ŘÜ'„ćSö»ô÷öŽsŤK)ăçĚ,‘fŢQ…cz!

Klucz w Base64

l/seiG0Rct4yci052OIDeB4/VJJ/fBBB59+ZBOmrsmh5v1thupQF2NwnhOZT9rv09/aO
c41LKePnF8wskWbeUYVjeiE=

Krok 4. Na stronie można również odszyfrować plik. Należy podać zaszyfrowany plik .enc w formularzu i ponownie podać własny ciąg do wygenerowania klucza (musi być identyczny jak w **Kroku 3.**)

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Ćwiczenia samodzielne

Zadanie 7.1

Stwórz 20MB kontener plikowy na dysku lub na pendrive, zamontuj go, wgraj niepusty plik txt oraz jakąś grafikę z internetu, odmontuj kontener, zamontuj go ponownie jako dysk z inną literą i odtwórz zawartość.

Zadanie 7.2

1. Utwórz **dwa** oddzielne kontenery plikowe w dowolnej lokalizacji:
    - Kontener A: rozmiar 20 MB, hasło: KontenerA202X! (lub inne zgodne z zasadami silnego hasła)
    - Kontener B: rozmiar 30 MB, hasło: KontenerB202X! (inne niż dla kontenera A)
2. Zamontuj oba kontenery na dwóch różnych wolnych literach dysków (np. V: i W:).
3. Skopiuj do każdego kontenera inne pliki (np. obrazy, dokumenty tekstowe).
4. Odmontuj **tylko** jeden z kontenerów.
5. Sprawdź, czy drugi kontener pozostaje dostępny.
6. Odmontuj wszystkie kontenery za pomocą przycisku **Dismount All**.
7. Zamontuj je ponownie i sprawdź zawartość.

Zadanie 7.3

Zaszyfruj plik przez narzędzie online, zapisz klucz użyty do szyfrowania. Użyj własnego programu do deszyfrowania wiadomości.

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Pytania pomocnicze

1. Jaki algorytm szyfrowania stosuje 7-Zip do zabezpieczania archiwów ZIP i jakie ma to znaczenie dla bezpieczeństwa danych?
2. Jakie są ograniczenia formatu ZIP w porównaniu do formatu 7z w kontekście szyfrowania?
3. Czym różni się model szyfrowania oferowany przez VeraCrypt od szyfrowania archiwum w 7-Zip?
4. Jakie są zalety korzystania z wersji portable VeraCrypt w środowisku pracowni komputerowej?
5. Przedstaw ryzyka związane z szyfrowaniem plików za pomocą narzędzi on-line.

Podsumowanie

W ramach zajęć studenci przeszli kompleksowe wprowadzenie do praktycznych metod szyfrowania plików, rozpoczynając od lokalnych narzędzi dostępnych w systemie Windows (7-Zip z szyfrowaniem AES-256 dla archiwów ZIP), poprzez zaawansowane rozwiązania oparte na kontenerach, aż po świadome korzystanie z narzędzi online implementujących model zero-knowledge z szyfrowaniem po stronie klienta i kluczem we fragmencie URL. Dzięki temu uzyskali praktyczną umiejętność doboru odpowiedniej metody szyfrowania w zależności od kontekstu operacyjnego – od szybkiej ochrony pojedynczych plików przeznaczonych do przesłania pocztą elektroniczną, przez tworzenie zaszyfrowanych wolumenów do długoterminowego przechowywania danych, aż po bezpieczne przesyłanie poufnych informacji z wykorzystaniem mechanizmów samodestrukcji.

Literatura

1. Program 7-zip do obsługi pakowania i szyfrowania plików

   https://www.7-zip.org/faq.html

2. Program veracrypt do szyfrowania katalogów i partycji

   https://veracrypt.io/en/Documentation.html

Fundusze Europejskie dla Rozwoju Społecznego | Rzeczpospolita Polska | Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Materiały do przedmiotu:

Podstawy kryptografii

Laboratorium nr 8

Szyfrowanie i deszyfrowanie wiadomości z użyciem algorytmu ROT13

Autor:

Dr inż. Marek B.Horyński

Fundusze Europejskie dla Rozwoju Społecznego

Rzeczpospolita Polska

Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Spis treści

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Wprowadzenie, tematyka zajęć

Instrukcja laboratoryjna: Szyfrowanie i deszyfrowanie wiadomości z użyciem algorytmu ROT13

Jednym z fundamentów historycznej kryptografii są **szyfry podstawieniowe**. Ich zasada opiera się na zamianie (podstawieniu) znaków tekstu jawnego na inne znaki według ściśle określonego klucza. Najbardziej znanym przedstawicielem tej grupy jest **Szyf Cezara**, który polega na przesunięciu liter alfabetu o stałą liczbę pozycji.

ROT13 (Rotate by 13 places) to specyficzny wariant szyfru Cezara, w którym przesunięcie wynosi dokładnie 13 pozycji. W standardowym alfabecie łacińskim (26 liter) powoduje to unikalną właściwość: ten sam algorytm służy zarówno do szyfrowania, jak i deszyfrowania wiadomości.

Cechy charakterystyczne ROT13:

- **Inwolucja:** Funkcja szyfrująca jest swoją własną odwrotnością: $f(f(x)) = x$.
- **Symetryczność:** Nie wymaga osobnego klucza do odczytu wiadomości.
- **Obfuskacja:** Współcześnie nie jest traktowany jako bezpieczny szyfr, a jedynie jako metoda ukrywania treści (np. spoilerów na forach internetowych).

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Cel laboratorium

Głównym celem zajęć jest praktyczne zapoznanie się z mechanizmami kryptografii klasycznej poprzez:

1. Zrozumienie operacji matematycznych na kodach ASCII.
2. Implementację algorytmu obsługującego duże i małe litery.
3. Analizę zachowania algorytmu dla znaków spoza alfabetu łacińskiego.
4. Zrozumienie pojęcia „bezpieczeństwa przez niejasność" (security through obscurity).

Analiza matematyczno-algorytmiczna

Podstawą działania ROT13 jest arytmetyka modularna. Dla litery o pozycji x w alfabecie (gdzie A=0, B=1,..., Z=25), wzór na szyfrowanie wygląda następująco:

$$E(x) = (x + 13) \pmod{26}$$

Ponieważ 13 + 13 = 26 ≡0 mod 26, ponowne nałożenie szyfru przywraca pierwotną wartość:

$$D(E(x)) = (x + 13 + 13) \pmod{26} = x \pmod{26}$$

Mapowanie ASCII

W implementacji programistycznej musimy uwzględnić kody ASCII:

1. Wielkie litery: 'A' (65) do 'Z' (90)
2. Małe litery: 'a' (97) do 'z' (122)

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Zadanie 1: Szkielet programu

Należy przygotować funkcję rot13(text), która będzie iterować po każdym znaku ciągu wejściowego.

Logika transformacji dla pojedynczego znaku (pseudokod):

1. Sprawdź, czy znak jest literą.
2. Jeśli tak:

- Określ bazę (65 dla wielkich, 97 dla małych liter).
- Oblicz nową pozycję: (kod_ascii - baza + 13) % 26 + baza.

3. Jeśli nie (cyfra, spacja, interpunkcja):

- Pozostaw znak bez zmian.

Zadanie 2: Implementacja w Pythonie

Przykładowy fragment kodu realizujący transformację:

```
def rot13(text):
    result = ""
    for char in text:
        if 'a' <= char <= 'z':
            # Przesunięcie dla małych liter
            result += chr((ord(char) - 97 + 13) % 26 + 97)
        elif 'A' <= char <= 'Z':
            # Przesunięcie dla wielkich liter
            result += chr((ord(char) - 65 + 13) % 26 + 65)
        else:
            # Znaki specjalne zostają bez zmian
            result += char
    return result

# Testowanie
wiadomosc = "Hello World!"
zaszyfrowana = rot13(wiadomosc)
print(f"Zaszyfrowana: {zaszyfrowana}")
print(f"Odszyfrowana: {rot13(zaszyfrowana)}")
```

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Ćwiczenia samodzielne

1. **Obsługa plików:** Rozszerz program tak, aby wczytywał tekst z pliku .txt, szyfrował go i zapisywał wynik do nowego pliku.
2. **Własny przesuw:** Zmodyfikuj funkcję tak, aby przyjmowała drugi parametr n (przesunięcie). Sprawdź, czy dla n=13 program nadal działa dwukierunkowo.
3. **Analiza statystyczna:** Spróbuj odpowiedzieć na pytanie: dlaczego ROT13 jest podatny na analizę częstotliwościową występowania liter?

Pytania pomocnicze

1. Dlaczego ROT13 nie wymaga osobnej funkcji deszyfrującej?
2. Co się stanie, jeśli spróbujemy zaszyfrować polskie znaki diakrytyczne (np. 'ą', 'ęż')? Jak zmodyfikować algorytm, by je obsługiwał?
3. Czy zamiana 13 na 10 w algorytmie zachowa właściwość inwolucji? Uzasadnij.

Podsumowanie

ROT13 to doskonały przykład dydaktyczny, pokazujący jak proste operacje matematyczne mogą służyć do transformacji danych. Choć nie zapewnia on realnego bezpieczeństwa przed hakerami, jest powszechnie stosowany do ukrywania treści przed przypadkowym odczytaniem. Zrozumienie jego działania jest wstępem do nauki o nowoczesnych szyfrach blokowych i strumieniowych.

Literatura

1. Singh S., Księga szyfrów, Albatros, Warszawa 2001.
2. Schneier B., Kryptografia dla praktyków, Helion, Gliwice 2002.
3. Dokumentacja Python: ord(), chr(), operacje na stringach.

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Materiały do przedmiotu:

Podstawy kryptografii

Laboratorium nr 9

Szyfrowanie i deszyfrowanie wiadomości przy użyciu alfabetu Morse'a

Autor:

Dr inż. Marek B.Horyński

Fundusze Europejskie dla Rozwoju Społecznego

Rzeczpospolita Polska

Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Spis treści

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Wprowadzenie, tematyka zajęć

Instrukcja laboratoryjna: Szyfrowanie i deszyfrowanie wiadomości przy użyciu alfabetu Morse’a

Kod Morse’a to jedna z najstarszych i najbardziej rozpoznawalnych metod kodowania informacji, stworzona w celu przesyłania wiadomości tekstowych za pomocą serii impulsów elektrycznych (krótkich i długich). Choć współcześnie wyparty przez systemy cyfrowe, kod Morse’a pozostaje doskonałym modelem do nauki o **protokołach transmisji**, **strukturach danych** oraz **integralności sygnału**.

W przeciwieństwie do szyfrów podstawieniowych takich jak ROT13, kod Morse'a jest systemem kodowania o **zmiennej długości**. Każda litera alfabetu odpowiada unikalnej sekwencji kropek (sygnał krótki) i kresek (sygnał długi). Kluczowym elementem poprawnej interpretacji kodu Morse'a nie są jednak same symbole, ale **czas i separacja** (odstępy między znakami oraz słowami).

Cechy charakterystyczne kodu Morse'a:

- **Binarność sygnału:** Oparty na dwóch stanach (on/off), ale zróżnicowanych czasowo.
- **Standard ITU:** Międzynarodowy standard (ITU-R M.1677-1) definiuje precyzyjnie stosunki długości trwania sygnałów (kropka = 1 jednostka, kreska = 3 jednostki).
- **Hierarchiczna struktura:** Możliwość reprezentacji w formie drzewa binarnego, co optymalizuje proces dekodowania.

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Cel laboratorium

Podczas zajęć laboratoryjnych studenci osiągną następujące cele dydaktyczne:

1. **Budowa dwukierunkowego konwertera:** Zaprojektowanie i implementacja systemu konwersji (Tekst ↔ Kod Morse'a).
2. **Zrozumienie struktur danych:** Zastosowanie słowników (*dictionary*) w języku Python jako efektywnej metody mapowania znaków.
3. **Parsowanie ciągów znaków:** Nauka dzielenia i łączenia danych przy użyciu separatorów (metody .split() i .join()).
4. **Obsługa wyjątków:** Implementacja mechanizmów radzenia sobie z nieznanymi znakami i różną wielkością liter.

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Poniżej znajduje się przykładowa implementacja konwertera w języku Python, wykorzystująca strukturę słownika do przechowywania mapowania znaków.

```
# Słownik mapujący znaki na kod Morse'a
MORSE_CODE_DICT = {
    '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': '--..', '1': '.----', '2': '..---', '3': '...--',
    '4': '....-', '5': '.....', '6': '-....', '7': '--...',
    '8': '---..', '9': '----.', '0': '-----', ', ': '--..--',
    '.': '.-.-.-', '?': '..--..', '/': '-..-.', '-': '-....-',
    '(': '-.--.', ')': '-.--.-', ' ': '/'
}

def encrypt(message):
    """Konwertuje tekst jawny na kod Morse'a"""
    cipher = []
    for char in message.upper():
        if char in MORSE_CODE_DICT:
            cipher.append(MORSE_CODE_DICT[char])
        else:
            cipher.append('?')
    return ' '.join(cipher)

def decrypt(message):
    """Konwertuje kod Morse'a na tekst jawny"""
    REVERSE_DICT = {value: key for key, value in MORSE_CODE_DICT.items()}
    words = message.split(' ')
```

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

```
decipher = ""
for symbol in words:
    if symbol in REVERSE_DICT:
        decipher += REVERSE_DICT[symbol]
    else:
        decipher += '?'
return decipher
```

Ćwiczenia samodzielne

1. Optymalizacja dekodowania: Zaimplementuj funkcję decrypt tak, aby odwrócony słownik był generowany tylko raz (poza funkcją).
2. Obsługa polskich znaków: Rozszerz słownik MORSE_CODE_DICT o polskie znaki diakrytyczne (np. Ą, Ć, Ę).
3. Walidator wejścia: Napisz funkcję sprawdzającą poprawność kodu Morse'a (tylko dozwolone znaki).

Pytania pomocnicze

1. Dlaczego w kodzie Morse'a separatory są kluczowe dla integralności danych?
2. Jaka jest różnica w wydajności między słownikiem a listą przy dużej ilości danych?
3. Jak struktura drzewa binarnego ułatwia dekodowanie sygnałów o zmiennej długości?

Podsumowanie

Kod Morse'a uczy poprawnego zarządzania danymi i mapowania informacji. Choć algorytm ten nie zapewnia poufności, stanowi solidny fundament pod zrozumienie protokołów cyfrowych.

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Literatura

1. International Telecommunication Union (ITU). (2009). *Recommendation ITU-R M.1677-1*.
2. Cormen, T. H., et al. (2022). *Introduction to Algorithms*. MIT Press.
3. Van Rossum, G. *PEP 8 – Style Guide for Python Code*.

Fundusze Europejskie dla Rozwoju Społecznego

Rzeczpospolita Polska

Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Materiały do przedmiotu:

Podstawy kryptografii

Laboratorium nr 10

Szyfrowanie wiadomości przy użyciu szyfru Polibiusza

Autor:

Dr inż. Marek B.Horyński

Fundusze Europejskie dla Rozwoju Społecznego

Rzeczpospolita Polska

Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Spis treści

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Wprowadzenie, tematyka zajęć

Instrukcja laboratoryjna: Szyfrowanie wiadomości przy użyciu szyfru Polibiusza

Szyfr Polibiusza, znany również jako kwadrat Polibiusza, to antyczny szyfr podstawieniowy wynaleziony w Grecji przez historyka Polibiusza. W odróżnieniu od szyfru Cezara, który operuje na przesunięciu pojedynczych liter, szyfr Polibiusza zamienia każdą literę na parę współrzędnych (liczb), co stanowi wczesny przykład kodowania frakcyjnego.

Podstawą algorytmu jest kwadrat o wymiarach 5x5, w którym umieszcza się litery alfabetu. Ponieważ alfabet łaciński posiada 26 liter, a kwadrat oferuje tylko 25 pól, standardowo litery 'I' oraz 'J' traktuje się jako jeden znak (zajmują to samo pole).

Cechy charakterystyczne szyfru Polibiusza:

- Frakcjonowanie: Rozbicie jednego znaku na dwa mniejsze elementy (wiersz i kolumna).
- Numeryczność: Tekst jawny zostaje przekształcony w ciąg cyfr, co ułatwiało historyczną transmisję sygnałową (np. za pomocą pochodni).
- Determinizm: Każdej literze przypisany jest unikalny (w ramach danego klucza) zestaw współrzędnych.

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Cel laboratorium

Podczas zajęć laboratoryjnych studenci zrealizują następujące cele dydaktyczne:

1. Zaprojektowanie struktury kwadratu: Implementacja macierzy 5x5 w języku Python.
2. Opracowanie funkcji enkodującej: Stworzenie algorytmu zamiany znaków na pary współrzędnych.
3. Opracowanie funkcji dekodującej: Implementacja mechanizmu odczytu znaków z macierzy na podstawie podanych indeksów.
4. Analiza wydajnościowa: Porównanie czasu wyszukiwania w macierzy z wykorzystaniem słowników pomocniczych.

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Poniżej znajduje się implementacja szyfru w języku Python.

```
# Implementacja kwadratu Polibiusza (standard 5x5)
# Uwaga: Litery I oraz J dzielą to samo pole (2,4)
POLY_SQUARE = [
    ['A', 'B', 'C', 'D', 'E'],
    ['F', 'G', 'H', 'I', 'K'],
    ['L', 'M', 'N', 'O', 'P'],
    ['Q', 'R', 'S', 'T', 'U'],
    ['V', 'W', 'X', 'Y', 'Z']
]
def encrypt_polybius(text):
    """Szyfruje tekst jawny na współrzędne Polibiusza"""
    text = text.upper().replace('J', 'I')
    result = ""
    for char in text:
        for r in range(5):
            if char in POLY_SQUARE[r]:
                c = POLY_SQUARE[r].index(char)
                result += f"{r+1}{c+1} "
    return result.strip()
```

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

```
def decrypt_polybius(cipher):
    """Deszyfruje współrzędne na tekst jawny"""
    result = ""
    coords = cipher.split()
    for coord in coords:
        row = int(coord[0]) - 1
        col = int(coord[1]) - 1
        result += POLY_SQUARE[row][col]
    return result
# Przykład użycia
msg = "POLIBIUSZ"
encrypted = encrypt_polybius(msg)
print(f"Szyfr: {encrypted}")
print(f"Tekst: {decrypt_polybius(encrypted)}")
```

Ćwiczenia samodzielne

1. Rozszerzenie o polskie znaki: Zaimplementuj kwadrat o wymiarach 6x6, który pomieści polskie znaki diakrytyczne oraz cyfry 0-9.
2. Kluczowanie macierzy: Napisz funkcję, która generuje kwadrat Polibiusza na podstawie zadanego słowa-klucza (unikalne litery klucza wpisywane są jako pierwsze).
3. Analiza częstotliwości: Sprawdź, czy częstotliwość występowania par cyfr w szyfrogramie odpowiada częstotliwości liter w języku polskim.

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Pytania pomocnicze

1. Dlaczego w klasycznym kwadracie Polibiusza litery 'I' oraz 'J' są łączone? Jakie to niesie konsekwencje dla procesu deszyfrowania?
2. W jaki sposób szyfr Polibiusza ułatwiał komunikację przy użyciu sygnałów wizualnych (np. pochodni)?
3. Czy szyfr Polibiusza sam w sobie zapewnia wysoki poziom bezpieczeństwa? Uzasadnij odpowiedź.

Podsumowanie

Szyfr Polibiusza uczy studentów operowania na strukturach dwuwymiarowych i pokazuje, jak informacja tekstowa może zostać zredukowana do postaci numerycznej bez utraty jej znaczenia. Jest to kluczowy krok w zrozumieniu systemów frakcjonujących.

Literatura

1. International Telecommunication Union (ITU). (2009). *Recommendation ITU-R M.1677-1: International Morse Code*. Geneva: ITU.
2. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). *Introduction to Algorithms* (4th ed.). MIT Press.
3. Kahn, D. (1996). *The Codebreakers: The Comprehensive History of Secret Communication from Antiquity to the Internet*.

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Materiały do przedmiotu:

Podstawy kryptografii

Laboratorium nr 11

Szyfrowanie wiadomości przy użyciu szyfru Atbash

Autor:

Dr inż. Marek B.Horyński

Fundusze Europejskie dla Rozwoju Społecznego

Rzeczpospolita Polska

Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Spis treści

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Wprowadzenie, tematyka zajęć

Instrukcja laboratoryjna: Szyfrowanie wiadomości przy użyciu szyfru Atbash

Szyfr Atbash to jeden z najstarszych znanych szyfrów podstawieniowych, wywodzący się z biblijnej tradycji hebrajskiej. Jego nazwa pochodzi od pierwszych i ostatnich liter alfabetu hebrajskiego: Aleph, Taw, Beth i Shin.

Zasada działania jest niezwykle prosta, ale elegancka: polega na odwróceniu alfabetu. Pierwsza litera alfabetu (A) jest zastępowana przez ostatnią (Z), druga (B) przez przedostatnią (Y), i tak dalej.

Kluczowe cechy szyfru Atbash:

- Inwolucja: Podobnie jak ROT13, Atbash posiada właściwość f(f(x))=x. Oznacza to, że ponowne zaszyfrowanie kryptogramu tym samym algorytmem przywraca tekst jawny.
- Szyfr monoalfabetyczny: Każdej literze tekstu jawnego odpowiada zawsze ta sama litera szyfru.
- Brak klucza zewnętrznego: Bezpieczeństwo szyfru opiera się wyłącznie na poufności algorytmu (co w dzisiejszych czasach czyni go metodą czysto edukacyjną, a nie zabezpieczającą).

Cel laboratorium

Podczas zajęć student zrealizuje następujące cele:

- Zrozumienie mechaniki inwolucji: Analiza matematyczna i logiczna algorytmów samo-odwracalnych.
- Implementacja algorytmu: Stworzenie funkcjonalnego programu w Pythonie obsługującego transformację ASCII.
- Analiza kryptoanalityczna: Zrozumienie podatności szyfrów monoalfabetycznych na analizę częstotliwościową.

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

- Analiza wydajnościowa: Porównanie czasu wyszukiwania w macierzy z wykorzystaniem słowników pomocniczych.

Instrukcja laboratoryjna

Algorytm matematyczny

Dla alfabetu łacińskiego o długości L=26, jeśli przypiszemy literze A wartość 0, a literze Z wartość 25, transformację Atbash dla danej litery x można zapisać wzorem:

E(x)=(L−1)−x

Implementacja w języku Python

Poniższy kod demonstruje, jak zaimplementować Atbash z zachowaniem wielkości liter i pominięciem znaków specjalnych:

Python

```
def atbash(text):
    result = ""
    for char in text:
        if char.isalpha():
            # Określenie punktu startowego ASCII (A=65 lub a=97)
            start = ord('A') if char.isupper() else ord('a')
            # Obliczenie nowej pozycji: (25 - obecna_pozycja)
            new_char = chr(start + (25 - (ord(char) - start)))
            result += new_char
        else:
```


Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

```
        # Znaki niebędące literami pozostają bez zmian
        result += char
    return result
# Testowanie algorytmu
wiadomosc = "Kryptografia 2026"
zaszyfrowana = atbash(wiadomosc)
print(f"Tekst jawny: {wiadomosc}")
print(f"Kryptogram:   {zaszyfrowana}")
print(f"Odszyfrowanie: {atbash(zaszyfrowana)}")
```

Ćwiczenia samodzielne

1. Atbash a liczby: Zmodyfikuj funkcję tak, aby szyfrowała również cyfry (0 zamień na 9, 1 na 8 itd.).
2. Obsługa plików: Napisz skrypt, który wczyta tajną wiadomość z pliku sekret.txt i wygeneruje jego "lustrzane odbicie" w pliku wynik.txt.
3. Kryptoanaliza: Znajdź w sieci tekst zaszyfrowany Atbash i spróbuj go odszyfrować ręcznie, używając przygotowanej przez siebie tabeli podstawień.

Pytania pomocnicze

1. Czy Szyfr Atbash posiada klucz? Jeśli tak, to jaki?
2. Wyjaśnij, dlaczego Atbash jest uznawany za szczególny przypadek Szyfru Afinicznego (ax+b(modm)). Podaj wartości a i b.
3. Dlaczego analiza częstotliwościowa występowania liter w języku polskim pozwala złamać Atbash w kilka sekund?

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Podsumowanie

Szyfr Atbash, choć trywialny z punktu widzenia nowoczesnej kryptografii, stanowi doskonały punkt wyjścia do nauki o **symetrii algorytmicznej**. Pokazuje, że bezpieczeństwo nie może polegać na samym ukryciu metody (Security through obscurity), lecz musi opierać się na tajnym kluczu o dużej entropii.

Literatura

1. Kahn D., The Codebreakers: The Story of Secret Writing, Scribner 1996.
2. Singh S., Księga Szyfrów, Albatros 2001.
3. Dokumentacja Python.

Fundusze Europejskie dla Rozwoju Społecznego

Rzeczpospolita Polska

Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Materiały do przedmiotu:

Podstawy kryptografii

Laboratorium nr 12

Kryptografia w chmurze. Analiza metod zabezpieczania danych w chmurze z wykorzystaniem szyfrowania

Autor:

Dr inż. Marek B.Horyński

Fundusze Europejskie dla Rozwoju Społecznego

Rzeczpospolita Polska

Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Spis treści

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Wprowadzenie, tematyka zajęć

Instrukcja laboratoryjna: Kryptografia w chmurze. Analiza metod zabezpieczania danych w chmurze z wykorzystaniem szyfrowania

W dobie powszechnej migracji danych do chmury (Cloud Computing), kryptografia staje się jedyną skuteczną barierą chroniącą poufność informacji w środowiskach współdzielonych. Podstawowym wyzwaniem nie jest już sam wybór algorytmu (standardem pozostaje AES-256), lecz **architektura zarządzania kluczami** (Key Management).

Wyróżniamy trzy stany danych wymagające zabezpieczenia:

1. **Data at Rest (Dane w spoczynku):** Dane zapisane na dyskach w centrach danych dostawcy chmurowego.
2. **Data in Transit (Dane w ruchu):** Dane przesyłane między użytkownikiem a chmurą (zabezpieczane protokołem TLS).
3. **Data in Use (Dane w użyciu):** Najtrudniejszy do zabezpieczenia stan, wymagający zaawansowanych technik takich jak **Szyfrowanie Homomorficzne** lub izolowane enklawy (Confidential Computing).

Cel laboratorium

Podczas zajęć student zrealizuje następujące cele dydaktyczne:

- Zrozumienie modeli odpowiedzialności: Podział ról między dostawcą chmury (CSP) a klientem w zakresie kryptografii.
- Symulacja zarządzania kluczami (KMS): Implementacja mechanizmu rotacji kluczy i separacji uprawnień.
- Implementacja Envelope Encryption: Zastosowanie techniki szyfrowania kopertowego, która jest standardem w AWS, Azure i GCP.

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Zadanie: Implementacja Szyfrowania Kopertowego (Envelope Encryption)

Szyfrowanie kopertowe polega na szyfrowaniu danych kluczem danych (DEK), który z kolei jest szyfrowany kluczem nadrzędnym (MK/KMS).

Python

```
import cryptography

from cryptography.fernet import Fernet


# 1. Symulacja KMS (Klucz Główny - Master Key)

# W rzeczywistości ten klucz nigdy nie opuszcza modułu HSM/KMS

master_key = Fernet.generate_key()

kms = Fernet(master_key)


def encrypt_cloud_data(data):

    # 2. Generowanie unikalnego klucza danych (Data Encryption Key)

    dek = Fernet.generate_key()

    f_dek = Fernet(dek)


    # 3. Szyfrowanie danych kluczem DEK

    encrypted_data = f_dek.encrypt(data.encode())


    # 4. Szyfrowanie klucza DEK kluczem nadrzędnym (Envelope)
```

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

```
    encrypted_dek = kms.encrypt(dek)

    return encrypted_data, encrypted_dek
# Przykładowe użycie
secret_info = "Poufne dane w chmurze 2026"
data_blob, wrapped_key = encrypt_cloud_data(secret_info)
print(f"Zaszyfrowane dane (Blob): {data_blob[:20]}...")
print(f"Zaszyfrowany klucz (Wrapped DEK): {wrapped_key[:20]}...")
```

Ćwiczenia samodzielne

1. Mechanizm Rotacji: Zmodyfikuj powyższy kod tak, aby system obsługiwał dwa klucze nadrzędne (stary i nowy) i pozwalał na re-encapsulation bez deszyfrowania danych źródłowych.
2. Analiza Metadanych: Przygotuj strukturę pliku (np. JSON), która będzie przechowywać zaszyfrowane dane wraz z ID klucza nadrzędnego i datą ważności.
3. Symulacja wycieku: Co się stanie, jeśli atakujący przejmie data_blob, ale nie ma dostępu do kms? Uzasadnij odpowiedź.

Pytania pomocnicze

1. Czym różni się Client-Side Encryption od Server-Side Encryption w kontekście bezpieczeństwa prawnego (np. RODO)?
2. Dlaczego w chmurze rzadko szyfruje się duże zbiory danych bezpośrednio kluczem nadrzędnym (Master Key)?
3. Czym jest Cloud Hardware Security Module (HSM) i jaką rolę pełni w certyfikacji FIPS 140-2?

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Podsumowanie

Kryptografia w chmurze to proces ciągłego balansowania między bezpieczeństwem a wydajnością. Implementacja szyfrowania kopertowego pozwala na bezpieczne składowanie ogromnych wolumenów danych przy zachowaniu centralnej kontroli nad dostępem poprzez usługi zarządzania kluczami..

Literatura

1. Cloud Security Alliance (CSA), *Security Guidance for Critical Areas of Focus in Cloud Computing*.
2. NIST Special Publication 800-144, *Guidelines on Security and Privacy in Public Cloud Computing*.
3. Hu V. C., *Guide to General Server Security*, NIST.

Fundusze Europejskie dla Rozwoju Społecznego

Rzeczpospolita Polska

Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Materiały do przedmiotu:

Podstawy kryptografii

Laboratorium nr 13

Kryptografia w mediach społecznościowych. Zastosowania w zabezpieczaniu danych użytkowników

Autor:

Dr inż. Marek B.Horyński

Fundusze Europejskie dla Rozwoju Społecznego

Rzeczpospolita Polska

Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Spis treści

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Wprowadzenie, tematyka zajęć

Instrukcja laboratoryjna: Kryptografia w mediach społecznościowych. Zastosowania w zabezpieczaniu danych użytkowników

Media społecznościowe i komunikatory (Messenger, WhatsApp, Signal) przetwarzają ogromne ilości wrażliwych danych. Kluczowym wyzwaniem jest ochrona wiadomości nie tylko przed osobami trzecimi, ale również przed samym dostawcą usługi.

W tradycyjnym modelu szyfrowania (at-transit), serwer pośredniczący posiada dostęp do kluczy deszyfrujących, co umożliwia mu odczytanie treści. Rozwiązaniem tego problemu jest Szyfrowanie End-to-End (E2EE). W tym modelu klucze kryptograficzne są generowane i przechowywane wyłącznie na urządzeniach końcowych użytkowników.

Kluczowe technologie w social mediach:

- Perfect Forward Secrecy (PFS): Mechanizm gwarantujący, że kompromitacja jednego klucza sesji nie pozwoli na odszyfrowanie wiadomości z przeszłości.
- Protokół Signal: Standard de facto dla bezpiecznej komunikacji, wykorzystujący algorytm Double Ratchet.
- Haszowanie haseł: Wykorzystanie funkcji typu Argon2 lub bcrypt do bezpiecznego przechowywania danych uwierzytelniających.

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Cel laboratorium

Podczas zajęć student zrealizuje następujące cele:

1. Zrozumienie różnicy między szyfrowaniem symetrycznym a asymetrycznym w kontekście wymiany kluczy.
2. Implementacja uproszczonego protokołu wymiany wiadomości z wykorzystaniem kluczy publicznych i prywatnych.
3. Analiza mechanizmu "bezpiecznych kodów" (numerów bezpieczeństwa) używanych do weryfikacji tożsamości rozmówcy.

Analiza modelu End-to-End Encryption (E2EE)

W modelu E2EE proces wygląda następująco:

1. Użytkownik A generuje parę kluczy: publiczny (dostępny dla wszystkich) i prywatny (tylko na telefonie).
2. Użytkownik B pobiera klucz publiczny Użytkownika A z serwera.
3. Użytkownik B szyfruje wiadomość kluczem publicznym A.
4. Tylko klucz prywatny A (którego serwer nie posiada) może odszyfrować tę wiadomość.

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

Zadanie 1: Symulacja bezpiecznej wymiany wiadomości (Python)

1. Koncepcja teoretyczna

W mediach społecznościowych proces ten nazywamy Szyfrowaniem Kopertowym (Envelope Encryption):

1. Nadawca generuje jednorazowy, losowy klucz symetryczny (Session Key).
2. Nadawca szyfruje treść wiadomości kluczem sesji (algorytm AES – bardzo szybki).
3. Nadawca szyfruje klucz sesji kluczem publicznym odbiorcy (algorytm RSA – zapewnia poufność).
4. Odbiorca deszyfruje "kopertę" (klucz sesji) swoim kluczem prywatnym RSA.
5. Odbiorca używa odzyskanego klucza sesji do odczytania treści wiadomości.

2. Implementacja w Pythonie

Do wykonania zadania wymagana jest biblioteka cryptography.

Python

```
import os
from cryptography.hazmat.primitives.asymmetric import rsa, padding
from cryptography.hazmat.primitives import hashes, serialization
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes


# --- ETAP 1: PRZYGOTOWANIE INFRASTRUKTURY (Użytkownik A - Odbiorca) ---
# Odbiorca generuje klucze raz i udostępnia klucz publiczny
private_key_A = rsa.generate_private_key(public_exponent=65537, key_size=2048)
public_key_A = private_key_A.public_key()


# --- ETAP 2: SZYFROWANIE (Użytkownik B - Nadawca) ---
```

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

```
message_to_send = "Tajna wiadomość z laboratorium: Protokół hybrydowy działa!".encode()

# A. Generowanie losowego klucza symetrycznego AES (256-bit) i wektora inicjalizacyjnego (IV)
session_key = os.urandom(32)
iv = os.urandom(16)

# B. Szyfrowanie wiadomości algorytmem symetrycznym (AES-CBC)
cipher = Cipher(algorithms.AES(session_key), modes.CBC(iv))
encryptor = cipher.encryptor()
# Padding (dopełnienie) wiadomości do wielokrotności 16 bajtów
pad_len = 16 - (len(message_to_send) % 16)
padded_message = message_to_send + bytes([pad_len] * pad_len)
ciphertext = encryptor.update(padded_message) + encryptor.finalize()

# C. Szyfrowanie klucza sesji kluczem publicznym odbiorcy (RSA)
encrypted_session_key = public_key_A.encrypt(
    session_key,
    padding.OAEP(
        mgf=padding.MGF1(algorithm=hashes.SHA256()),
        algorithm=hashes.SHA256(),
        label=None
    )
)
print(f"--- WYSYŁKA PRZEZ SERWER ---")
print(f"Zaszyfrowany klucz (RSA): {encrypted_session_key.hex()[:40]}...")
print(f"Zaszyfrowana treść (AES): {ciphertext.hex()[:40]}...")
```


Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00

```
# --- ETAP 3: DESZYFROWANIE (Użytkownik A - Odbiorca) ---
# A. Odbiorca deszyfruje klucz sesji swoim kluczem prywatnym
decrypted_session_key = private_key_A.decrypt(
    encrypted_session_key,
    padding.OAEP(
        mgf=padding.MGF1(algorithm=hashes.SHA256()),
        algorithm=hashes.SHA256(),
        label=None
    )
)

# B. Odbiorca używa klucza sesji do odszyfrowania treści wiadomości
cipher_dec = Cipher(algorithms.AES(decrypted_session_key), modes.CBC(iv))
decryptor = cipher_dec.decryptor()
decrypted_padded_msg = decryptor.update(ciphertext) + decryptor.finalize()

# Usunięcie paddingu
unpad_len = decrypted_padded_msg[-1]
final_message = decrypted_padded_msg[:-unpad_len]

print(f"\n--- ODBIÓR ---")
print(f"Odszyfrowana wiadomość: {final_message.decode()}")
```

Ćwiczenia samodzielne

1. Atak Man-in-the-Middle (MitM): Opisz, co by się stało, gdyby serwer podstawił własny klucz publiczny zamiast klucza Użytkownika A. Jak

aplikacje społecznościowe (np. WhatsApp) pozwalają użytkownikom wykryć taką sytuację?
2. Efemeryczność: Zaimplementuj mechanizm, który po każdej wiadomości generuje nowy klucz sesji (uproszczona idea Ratchetingu).
3. Zabezpieczanie mediów: Czy szyfrowanie asymetryczne (RSA) nadaje się do przesyłania dużych zdjęć w social mediach? Zaproponuj rozwiązanie hybrydowe (szyfrowanie zdjęcia kluczem AES i szyfrowanie klucza AES kluczem RSA).

Pytania pomocnicze

1. Dlaczego Facebook Messenger oferuje szyfrowanie E2EE tylko w "tajnych konwersacjach", a nie domyślnie (w starszych wersjach)?
2. Jakie zagrożenie dla kryptografii w social mediach stanowi tzw. "metadane" (kto, z kim i kiedy rozmawia), nawet jeśli treść jest zaszyfrowana?
3. Czym jest odzyskiwanie konta w kontekście E2EE? Dlaczego reset hasła często wiąże się z utratą dostępu do starych, zaszyfrowanych wiadomości?

Podsumowanie

Kryptografia w mediach społecznościowych to nie tylko algorytmy, ale przede wszystkim walka o zaufanie użytkownika. Standard E2EE sprawia, że dostawcy usług stają się jedynie "ślepymi kurierami", co znacząco podnosi poziom prywatności, ale jednocześnie stawia wyzwania przed organami ścigania.

Literatura

1. Signal Protocol Documentation: *The Double Ratchet Algorithm*.
2. Anderson R., *Security Engineering: A Guide to Building Dependable Distributed Systems*.
3. Zalecenia ENISA dotyczące bezpieczeństwa komunikatorów internetowych.

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Projekt współfinansowany ze środków Unii Europejskiej w ramach: FUNDUSZE EUROPEJSKIE DLA ROZWOJU SPOŁECZNEGO 2021-2027 (FERS) "POLLUB z nami nowoczesne technologie" FERS.01.05-IP.08-0319/23-00