Kryptografia, Lista 2

Adrian Mucha, Politechnika Wrocławska, WPPT 17/05/2020

AES vs. DES

AES	DES
AES: Advanced Encryption Standard	DES: Data Encryption Standard
Klucze mają długość 128, 192 lub 256 bitów	Długość klucza to 56 bitów.
Liczba rund w zależności od klucza: 10(128-bitów), 12(192-bitów) lub 14(256-bitów)	16 rund identycznych operacji
Struktura opiera się o sieć substytucyjno-permutacyjną.	Struktura opiera się o sieci Feistela.
AES jest bezpieczniejszy niż DES i jest światowym standardem.	DES może być łatwo złamany i ma wiele znanych słabości. Istnieje 3DES który jest bezpieczniejszy niż DES.
Koduje 128 bitów tekstu jawnego.	Koduje 64 bity tekstu jawnego.
Brak znanych ataków crypto-analitycznych prócz ataków side channel.	Znane ataki: Brute-force, Linear crypt-analysis oraz Differential crypt-analysis.

Tryby AES

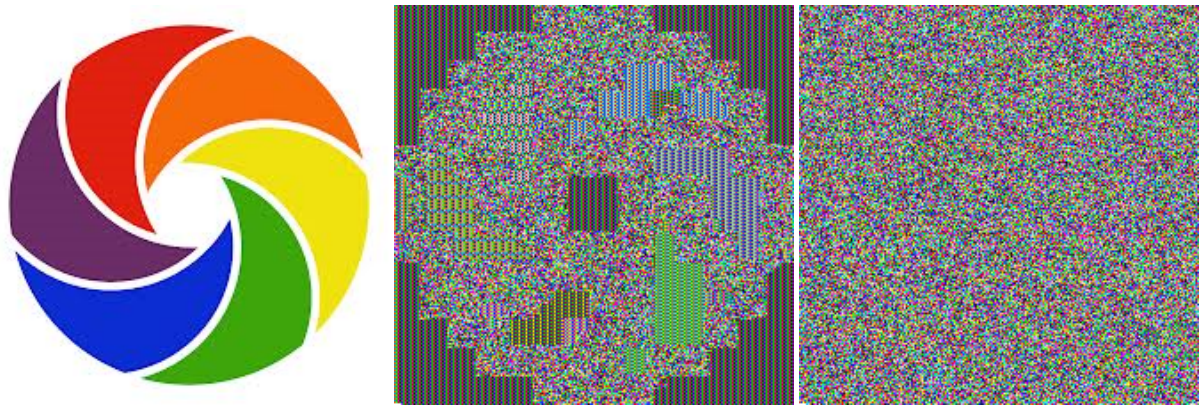
AES obsługuje różne tryby operowania na danych, które posiadają różne właściwości i stopnie bezpieczeństwa oraz szybkości działania czy możliwości pracy równoległej na wielu blokach. Tabela 1 przedstawia zalety i wady różnych trybów operowania.

Słabość w trybie ECB

Przykład słabości szyfrowania w trybie ECB. Gołym okiem widać powtarzające się szyfrogramy oraz wzorce na obrazku 1.

Bezpieczeństwo CBC

Tryb CBC jest wystawiony na *atak z wybranym tekstem jawnym* jeżeli wektor inicjalizujący `IV` nie będzie wybierany losowo przez nadawcę. Jeżeli adwersarz potrafi przewidzieć `IV` to będzie mógł zastosować wyżej wymieniony atak. Atak przedstawia obrazek 2.



(a) Zdjęcie oryginalne (b) ECB (c) CBC

Figure 1: Obrazy przedstawiają różnice w kodowaniu przykładowego obrazu za pomocą ECB oraz CBC. Można zauważyć, że tryb ECB zawiera powtórzenia w szyfrogramach jeżeli w tekście jawnym również takowe się znajdują.

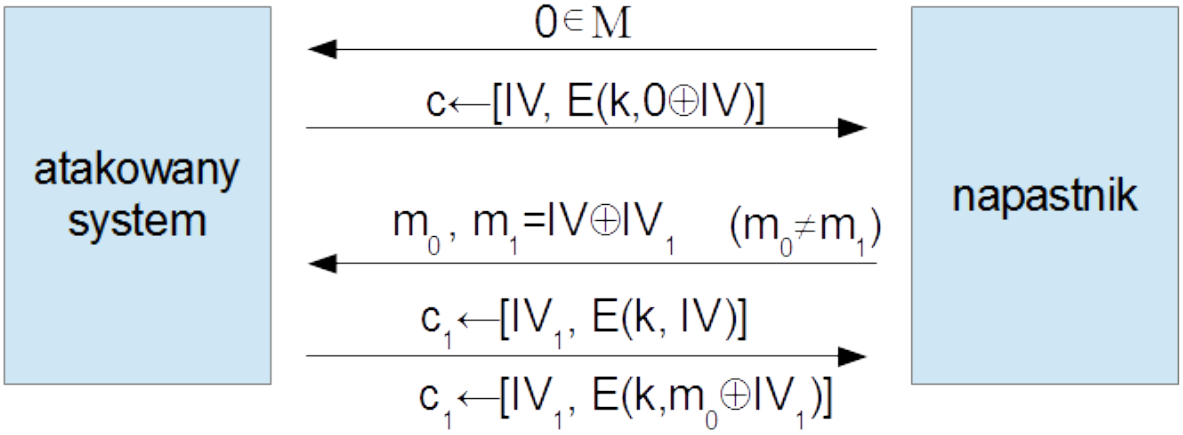


Figure 2: Atak z wybranym tekstem jawnym (CPA) na tryb szyfrowania CBC.

Tryb	Zalety	Wady
ECB	• Prosty • Szybki • Równoległy	• Powtórzenia tekstu jawnego będą widoczne w szyfrze • Uszkodzony szyfr będzie mieć wpływ na tekst jawny • Brak odporności na *replay attacks* • Nie powinno się go używać
CBC	• Równoległe odszyfrowywanie • Powtórzenia nie będą widoczne w szyfrze	• Brak równoległego szyfrowania • Uszkodzony blok wpływa na kolejne bloki
CFB	• Brak wyrównania (no padding) • Równoległe odszyfrowywanie	• Brak równoległego szyfrowania • Brak odporności na *replay attack* • Uszkodzony blok wpływa na kolejne bloki
OFB	• Brak wyrównania (no padding) • Szyfrowanie i deszyfrowanie używa tego samego schematu • Uszkodzony blok nie wpływa na inne	• Brak równoległego szyfrowania • Adwersarz może zmienić uszkodzić część szyfru by zmienić tekst jawny
CTR	• Brak wyrównania (no padding) • Równoległe szyfrowanie i deszyfrowanie • Szyfrowanie i deszyfrowanie używa tego samego schematu • Uszkodzony blok nie wpływa na inne	• Adwersarz może zmienić uszkodzić część szyfru by zmienić tekst jawny

Table 1: Wady i zalety różnych trybów AES.

Zad 1 & 2

Program szyfrujący/deszyfrujący oraz keystore

Program `index.ts` korzysta z mechanizmów szyfrujących dostarczanych przez `OpenSSL`. W szczególności używa dostępnych trybów `AES` (`CBC, OFB, CTR`) o długości klucza 256 bitów. Do celów demonstracyjnych `IV` jest stałe $(0...0)$ - nie powinno się tak robić.

Pliki są przetwarzane strumieniowo dzięki czemu możemy przetwarzać dane dowolnej wielkości bez potrzeby ładowania całości do pamięci. Przed szyfrowaniem pliki są dodatkowo kompresowane za pomocą `Gzip`. Zaletą programu jest to, że operuje na plikach tymczasowych i na końcu zastępuje szyfrowany plik, przez co unikamy możliwości uszkodzenia oryginalnego pliku gdyby wystąpił błąd podczas pracy programu i całość podmieniana jest już z gotowym zaszyfrowanym plikiem.

Program używa biblioteki opensource `key-store` do obsługi keystore'a i przechowuje klucze w pliku `store.jsks`. Plik `store.ts` jest odpowiedzialny za odczyt/zapis keystore'a. Nowe klucze można dodawać za pomocą skryptu `newKey.ts`.

Program obsługuje argumenty linii komend, które można wyświetlić za pomocą `--help`.

Uruchamianie

Wymagane środowisko `Node.js` w wersji `10 >=`.

1. `npm install` - instaluje zależności
2. `npm run encrypt -- [lista argumentów]`

Oracle & challenge

Plik `oracle.ts` można uruchomić jako serwer nasłuchujący na porcie 3000 na zapytania typu `POST`, który jako odpowiedź zwraca zaszyfrowane wiadomości. `IV` jest celowo zwiększane o 1 z każdą zaszyfrowaną wiadomością.

Skrypt `zad2.ts` jest atakiem CPA wykorzystującym *„lukę w systemie"* dzięki której jest w stanie przewidywać następne `IV` i z powodzeniem rozróżnić wiadomości zakodowane w challenge'u. Atak został przedstawiony na schemacie 2.

Uruchamianie

Wymagane środowisko `Node.js` w wersji `10 >=`.

1. `npm install` - instaluje zależności
2. `npm start` - uruchamia wyrocznię oraz challenge
3. `npm run crack` - uruchamia skrypt wygrywający atak CPA na `AES-256-CBC` gdy `IV` jest przewidywalne