Spis treści

Część I

Część teoretyczna

1 Cel i zakres pracy

Celem pracy jest przedstawienie algorytmów kryptograficznych, budowy systemów wbudowanych oraz implementacja algorytmu DES na systemie wbudowanym wraz ze zmierzeniem jego wydajności w zależności od rozmiaru szyfrowanych danych. Praca składa się z części teoretycznej, będącej wprowadzeniem do zagadnień kryptograficznych oraz budowy systemów wbudowanych i części praktycznej, w której przedstawiono charakterystykę wybranego systemu wbudowanego oraz implementację algorytmu DES wraz z przedstawieniem jego wydajności. Praca została podzielono na na dwie części: część teoretyczną, która składa się z dwóch rozdziałów i część praktyczną, która zawiera również dwa rozdziały.

W pierwszej części teoretycznej w rodziale 2 opisano powszechna ataki na algorytmy kryptograficzne oraz przedstawiono sposób działania szyfrów blokowych i strumieniowych. Następnie objaśniono działanie szyfrów symetrycznych, do których należy szyfr Cezara, algorytm DES oraz IDEA. W dalszej części objaśniono działanie szyfrów asymetrycznych, do których należą dwa słynne algorytmy kryptograficzne RSA i Diffiego-Helmanna. Na zakończenie rozdziału omówiono protokoły wykorzystywane do komunikacji między dwoma urządzeniami elektronicznymi. W rodziale 3 opisane budowę systemów wbudowanych poprzez opis ich składowych komponentów tj. modulator szerokości impulsów PWM, przetwornik analogowo-cyfrowy, przetwornik cyfrowo-analogowy, wejścia/wyjścia ogólnego przeznaczenia, uniwersalny odbiornik i nadajnik asynchroniczny UART, magistrale, zegar czasu rzeczywistego, centralna jednostka obliczeniowa, JTAG, pamięci oraz szyny danych, adresowe i sterowania.

W drugiej części w rozdziale 4 przedstawiono część praktyczną pracy, która została podzielona na cztery podrozdziały. W podrozdziale 4.1 ukazano charakterystykę użytego systemu wbudowanego ESP8266. Następnie w podrozdziale 4.2 opisano założenia projektowe oraz sposób połączenia ESP8266 z środowiskiem programistycznym Arduino. W dalszej części 4.3 przedstawiono manualna implementację szyfrowania krok po kroku wiadomości z wykorzystaniem algorytmu DES. Na samym końcu w podrozdziale 4.3 przedstawiono wyniki szyfrowania dla 16 rund programu implementującego algorytm DES i porównano ich zgodność z manualnym obliczeniem dla rundy pierwszej. Pomiarów czasu szyfrowania i deszyfrowania dokonano dla trzech rozmiarów danych o wielkości 1Kb, 8Kb i 16Kb oraz przeprowadzono ich analizę.

2 Kryptografia

2.1 Wprowadzenie

W obecnych czasach dużym zainteresowaniem cieszy się bezpieczeństwo cybernetyczne, którego szczególną częścią jest kryptografia. Szczególnie ważne zastosowanie znajduje w branży informatycznych, militarnej, urzędach, grupach developerskich czy bankowości. Kryptografia pojawiła się znaczniej wcześniej niż platformy obliczeniowe, zainteresowali się nią już ludzie z czasów starożytnych, pojawiła się wraz z umiejętnością pisania. Powodem istnienia kryptografii jest bezpieczne i prywatne dostarczanie wiadomości. Znajduje szczególne zastosowanie w przypadku danych przesyłanych drogą internetową, dzięki kryptografii możliwe jest zapewnienie bezpieczeństwa cybernetycznego przesyłanych danych. W zależności od stopnia poufności informacji, którą chcemy zaszyfrować, aby niepożądane osoby jej nie odczytały można zastosować odmiennych algorytmów szyfrowania.

Kryptologia to połączenie kryptografii i kryptoanalizy. W języku greckim 'kryptos' oznacza ukryty, zaś 'logos' tłumaczone jest jako słowo. Kryptologia jest dziedziną zajmującą się ukrywaniem tekstu jawnego. Kryptografia jest dziedziną węższą od kryptologii, jest badaniem technik matematycznych związanych z bezpieczeństwem informacji. Do bezpieczeństwa danych można zaliczyć poufność informacji, uwierzytelnienie użytkowników i pochodzenia danych, a także integralność danych. Słowo kryptologia składa się z dwóch greckich słów: 'kryptos' znaczący ukryty i 'graph' oznaczający pisanie, jest to nauka o zabezpieczaniu danych. Za pomocą technik kryptograficznych możliwe jest zaszyfrowanie jawnego tekstu, w taki sposób aby niepożądana osoba nie mogła ich odczytać. Drugą gałęzią kryptologii jest kryptoanaliza, która zajmuje się analizą i możliwymi sposobami odszyfrowania kodu kryptograficznego.

2.2 Powszechne ataki

W większości metod tunelowania wykorzystano szyfrowanie, które można podzielić na symetryczne i asymetryczne. Proces szyfrowania polega na przekształceniu tekstu jawnego w tekst zaszyfrowany. Szyfrowanie asymetryczne odbywa się przy użyciu dwóch kluczy: publicznego i prywatnego. Klucz publiczny jest wykorzystywany do szyfrowania danych, zaś klucz prywatny do ich odszyfrowywania. W przypadku szyfrowania symetrycznego wykorzystywany jest jeden klucz, co prowadzi do mniejszego bezpieczeństwa szyfrowania kosztem szybszego procesu szyfrowania i odszyfrowywania danych. Długość klucza określana jest w bitach. Dla klucza o mniejszej długości proces szyfrowania i odszyfrowywania przebiega szybciej. Wadą jest mniejsze bezpieczeństwo przesyłanych danych. Powszechnie w przypadku szyfrowania asymetrycznego nie stosuje się kluczy krótszych niż 4096 bitów. VPN umożliwia uwierzytelnienie urządzenia, sprawdzenie integralności pakietów i szyfrowanie, dzięki czemu ataki są bardziej czasochłonne i trudniejsze dla atakującego. Aby przed atakami się odpowiednio chronić warto poznać najpowszechniejsze typy ataków. Wyróżniono trzy główne metody atakowania prywatnych sieci wirtualnych :

- podsłuch

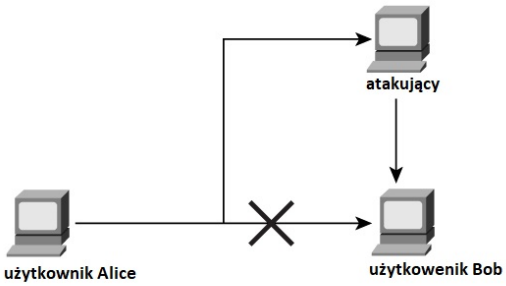
  Najpowszechniejszą metodą ataku jest podsłuchiwanie przesyłanych wiadomości przez osobę

trzecią podczas wysyłania wiadomości między dwoma osobami. Niektóre protokoły i aplikację takie jak POP,FTP, TFTP, HTTP czy TELNET są narażone na ataki metodą podsłuchu. Prywatne dane takie jak nazwa użytkownika i hasło są przesyłane w postaci tekstu między dwoma urządzeniami przy użyciu wymienionych powyżej protokołów, narażając prywatne dane na łatwe ich przechwycenie poprzez niepożądaną osobę. Głównym założeniem stworzenia protokołów POP, SMTP czy Telnet było umożliwienie komunikacji poprzez Internet z minimalnym uwierzytelnieniem użytkownika w celu zweryfikowania jego tożsamości. Głównym narzędziem do analizowania wysyłanych danych jest analizator protokołów. Pakiety wysyłane między urządzaniem źródłowym, a docelowym atakujący może przechwycić, gdy będzie miał dostęp do połączenia, które się odbywa między nimi. W celu lepszego zabezpieczenia pakietów przesyłanych przy użyciu protokołów z minimalnym uwierzytelnieniem dodano drugie uwierzytelnienie, które minimalizuje możliwości ataków przy użyciu analizatora protokołów. Przykładem z podwójnym uwierzytelnieniem będą operacje bankowe, które po podaniu loginu i hasło przy wykonywaniu danej operacji wymagają podania jednorazowego hasła, które zostało wysłane do nas elektronicznie lub drogą pocztową. Innym stosowanym rozwiązaniem jest połączenie protokołu HTTP i SSL czy VPN i szyfrowania. Najrzetelniejszym rozwiązaniem chroniącym przed podsłuchiwaniem będzie VPN połączone z szyfrowaniem, które uniemożliwi odczytanie wysyłanej wiadomości poprzez wirtualną prywatną sieć przez osoby atakujące w przypadku przechwycenia wiadomości.

- maskarada
  Ataki maskarady powszechnie znane jako podszywanie się osoby atakującej pod konkretną osobą bądź ukrywanie tożsamości. Podszywanie się pod określoną osobę wykonano poprzez zamianę informacji adresowych IP osoby atakującej na adres IP osoby, za którą chciano być postrzeganą przy pomocy specjalnych narzędzi. Atakujący nie otrzyma wiadomości, którą wysłano w ruchu powrotnym. W celu przechwycenia pakietów z ruchu powrotnego, należało by połączyć maskowanie adresu IP z routingiem w wyniku czego otrzymano by atak DoS, który przesyca sieć danymi. Pakiety danych wysyłane od nadawcy do odbiorcy mogą zostać zmodyfikowane przez osobą atakującą podczas drogi sieciowej, którą wiadomość jest transportowana. W celu wyeliminowania łatwej możliwości zmodyfikowania wysyłanych danych wprowadzono sprawdzenie autentyczności wysyłanych pakietów między dwoma urządzeniami. Sprawdzenie autentyczności pakietów umożliwia funkcja haszująca. Pakiet wysyłany do nadawcy posiada swój hasz, który jest utworzymy z użyciem klucza, który posiada odbiorca i nadawca. W celu sprawdzenia autentyczności wiadomości porównujemy hasz na dwóch urządzeniach danego pakietu. Gdy pakiet został zmodyfikowany przez atakującego hasze nie będą się zgadzały. Najczęściej spotykanymi funkcjami haszowania jest SHA i MD5 [1].

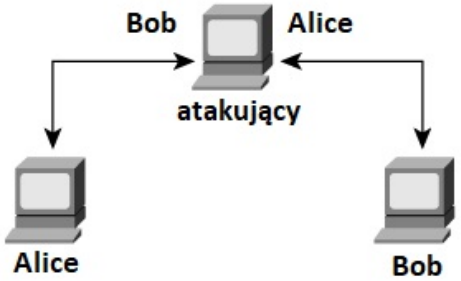
- człowiek w środku tzn. man in the middle
  Dwie najczęściej występujące metody wykorzystane do ataku to: ”powtórka ataku” oraz ataki typu ”porwanie sesji”. W przypadku sesji powtarzającej ataki osoba atakująca znajduję się między dwoma urządzeniami w celu przechwycenia pakietów informacji. Celem przechwycenia

pakietów danych jest ich późniejsze wysłanie do odbiorcy w zmienionym przez atakującego stanie np. z wirusem. Na rysunku poniżej przedstawiono schemat ataku dla sesji powtarzających ataki. W pierwszej kolejności użytkownik Alice wysyła pakiet do użytkownika Bob. Następnie atakujący przy użyciu dobranej przez niego techniki przechwytuje pakiety danych, modyfikuje ich zawartość i jako złośliwe pakiety zostają przesłane do użytkownika Bob. Na rys. 1 przedstawiono schematycznie atak wykorzystujący technikę powtarzania sesji.



Rysunek 1: Człowiek w środku - powtarzanie sesji

W atakach typu ”porwanie sesji” atakujący dodaje się do połączenia między dwoma użytkownikami w celu przejęcia komunikacji między nimi. Na rysunku poniżej pokazano schematycznie atak typu ”porwanie sesji”. Użytkownik Alice wysyła wiadomość do użytkownika Bob, jednak na drodze odczytuje ją atakujący, który przez Alice jest traktowany jako użytkownik Bob. Następnie atakujący przesyła wiadomość do prawdziwego Boba i po otrzymaniu jego odpowiedzi przesyła ją do użytkownika Alice po zmodyfikowaniu. Atakujący dołącza się do połączeń w celu znalezienie luk z zabezpieczeniach. Na rys. 2 przedstawiono schemat ataku typu człowiek w środku - porwanie sesji.



Rysunek 2: Człowiek w środku - porwanie sesji

Prywatna wirtualna sieć zapewnia trzy etapy chroniące nas przed niepożądanymi atakami w wystarczający sposób. Pierwszym z nich jest uwierzytelnienie urządzenia, które chroni przed przechwytywaniem pakietów przez zamaskowane urządzenia atakujące. Następnym krokiem

jest sprawdzenie integralności pakietów np. przy użyciu funkcji haszujących. Trzecim etapem jest zaszyfrowanie pakietów, co znacznie utrudnia przechwycenie rzeczywistych informacji [2].

2.3 Szyfry blokowe i strumieniowe

2.3.1 Szyfry blokowe

Szyfry blokowe wykorzystywane są do szyfrowania i deszyfrowania. Danymi wejściowymi do szyfrowania jest blok danych, który przy użyciu klucza szyfrującego jest przekształcany w zaszyfrowany blok danych. Odszyfrowywanie przebiega w odwrotny sposób, zaszyfrowana blokowa wiadomość przy użyciu klucza jest transformowana do odszyfrowanej blokowej wiadomości. Szyfr blokowy jest bezpieczny do momentu kiedy klucz szyfrujący pozostaje tajny. Bez znajomości klucza niemożliwe staje się rozszyfrowanie wiadomości w satysfakcjonującym nas czasie. Im bardziej klucz jest przypadkowy tym ciężej złamać algorytm. Ważnymi parametrami szyfru blokowego jest rozmiar bloku i klucza od których to zależy bezpieczeństwo algorytmu. Powszechnie stosowanymi rozmiarami bloków są 64 i 128 bitowe bloki, zazwyczaj algorytm DES ma 64 bitowy blok danych, zaś AES 128 bitowy blok. W tabeli 1 ukazano, że rozmiar 1 bajta odpowiada rozmiarowi 8 bitów.

1 bit	1 bit	1 bit	1 bit	1 bit	1 bit	1 bit	1 bit
1 bajt = 8 bitów							

Tabela 1: Rozmiar jednego bajta.

Długość zaszyfrowanego bloku musi być optymalna, im większe bloki danych tym dłuższy zaszyfrowany tekst oraz użycie pamięci. Przy szyfrowaniu wiadomości która na długość 8 bitową, zaś blok szyfru ma długość 64 bity najpierw 8 bitowa wiadomość zostanie prze konwertowana na 64 bitowy blok. Następnie 64 bitowa wiadomość zostanie zaszyfrowana przy użyciu algorytmu tworząc zaszyfrowany tekst. Do przetworzenia szyfru blokowego o rozmiarze 64 bitów potrzebujemy 64 bitów pamięci w rejestrach procesora. W dzisiejszych czasach procesory z pamięcią 64 bitową nie są kosztowne, lecz im większy chcemy utworzyć blok szyfrowy tym lepszy procesor potrzebujemy, co może mieć znaczny wpływ na wysokie koszty [3].

2.3.2 Szyfry strumieniowe

Szyfrowanie strumieniowe jest deterministyczne, co oznacza, że przy jednakowych danych wejściowych otrzymujemy taki sam wynik wyjściowy. Dzięki determinizmowi możliwe jest odszyfrowanie zaszyfrowanych strumieni bitów. Szyfrowanie strumieniowe jest podobne w działaniu do deterministycznego generatora liczb pseudolosowych, z tą różnicą że szyfrowanie strumieniowe oprócz wartości wejściowej pobiera dodatkowo klucz, który zazwyczaj ma 128 lub 256 bitów długości. Strumieniem klucza jest pseudolosowym strumieniem bitów. Szyfrowanie strumieniowe polega na przekształceniu tekstu jawnego bit po bicie na szyfrogram. Elementy tworzące szyfrowanie strumieniowe to generator strumienia bitowy oraz element dodający np. operacja XOR. Operację XOR-owania przedstawiono w tabeli 24.

p	q	p $\veebar$ q
0	0	0
0	1	1
1	0	1
1	1	0

Tabela 2: Tablica dla operacji XOR.

Proces XOR-owania przebiega następująco:

- wiadomość jako tekst jawny

01001101

- strumień klucza wytworzony przez generator strumienia klucza

00111000

- XOR-owana wiadomość (szyfrogram)

01110101

Szyfrowanie strumieniowe polega na operacji xor tekstu jawnego(P) ze strumieniem klucza(KS) w wyniku czego otrzymano zaszyfrowaną wiadomość(C).

$$C = P \oplus KS$$

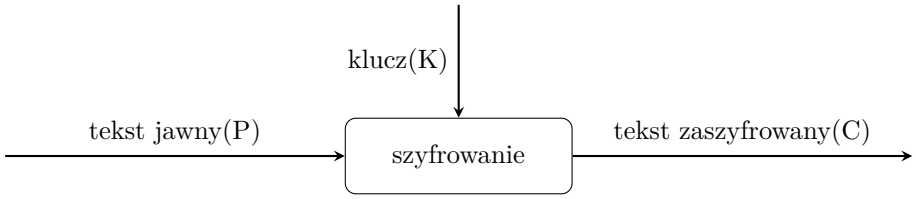
Proces deszyfrowania strumieniowego polega na operacji xor tekstu zaszyfrowanego(C) z strumieniem klucza(KS), w wyniku czego otrzymano tekst jawny(P) [4].

$$P = C \oplus KS$$

2.4 Szyfry symetryczne i asymetryczne

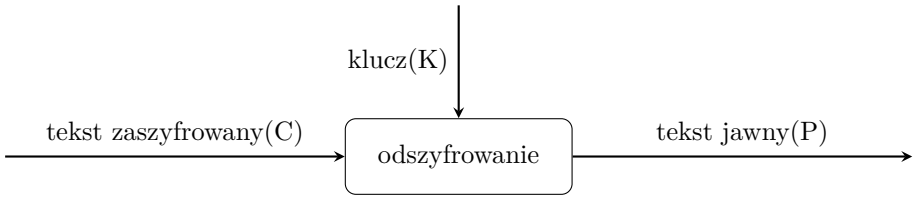
2.4.1 Szyfry symetryczne

Do szyfrów symetrycznych zaliczamy zarówno szyfry blokowe jak i strumieniowe. Szyfrowanie symetryczne używa jednego tajnego klucza do szyfrowania i odszyfrowywania wiadomości. W celu zaszyfrowania tekstu jawnego przy użyciu tajnego klucza szyfrujemy wiadomość do postaci tekstu zaszyfrowanego, co przedstawia rys. 3.



Rysunek 3: Schematyczny proces szyfrowania symetrycznego.

Proces odszyfrowywania polega na przekształceniu tekstu zaszyfrowanego przy użyciu klucza w tekst jawny. Klucz jest taki sam dla procesu szyfrowania i odszyfrowywania. Na rys. 4 ukazano schematycznie proces odszyfrowywania symetrycznego.



Rysunek 4: Schematyczny proces odszyfrowania symetrycznego.

Przed erą urządzeń obliczających używano szyfry klasyczne, które działały na literach, a nie na bitach. Do szyfrów klasycznych zaliczamy słynny szyfr Cezara.

szyfr Cezara

Szyfr Cezara zawdzięcza swoją nazwę Juliuszowi Cezarowi, który to już w czasach starożytnych był wykorzystywany przez Juliusza. Szyfr służył do szyfrowania wiadomości poprzez zastąpienie litery literą o 3 miejsca przesuniętą względem wartości początkowej np. literę A zastąpiono literą C. W tabeli 3 ukazano obrazowo podstawienie z przesunięciem o trzy litery.

podstawa	A	Ą	B	C	Ć	D	E	Ę	F	G	H	I	J	K	L	Ł	M
podstawienie	C	Ć	D	E	Ę	F	G	H	I	J	K	L	Ł	M	N	Ń	O
podstawa	N	Ń	O	Ó	P	R	S	Ś	T	U	W	X	Y	Z	Ż	Ź	
podstawienie	Ó	P	R	S	Ś	T	U	W	X	Y	Z	Ż	Ź	A	Ą	B	

Tabela 3: Szyfr Cezara z przesunięciem o trzy litery.

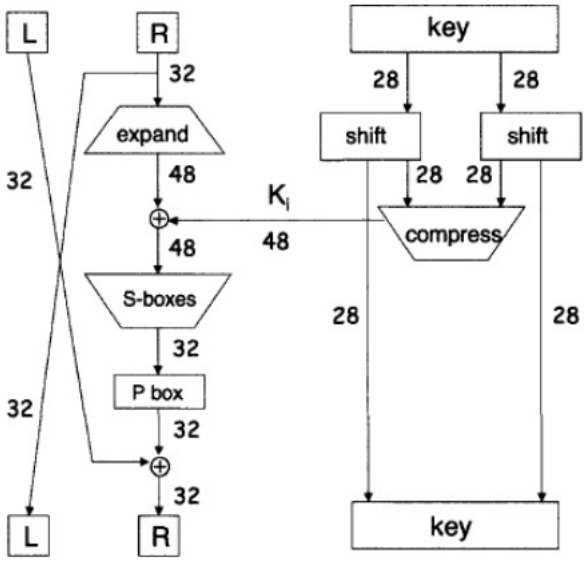
Szyfr ten nie zapewnia bezpieczeństwa, gdyż bez problemu i w krótkim czasie można przetestować wszystkie możliwe 33 opcje w przypadku języka polskiego. Szyfr ten uniemożliwia natychmiastowe zinterpretowanie wysyłanej wiadomości po jej ujrzeniu.

DES

Algorytm zwany standardem szyfrowania danych (ang. Data Encryption Standard DES) został opatentowany przez firmę IBM i rozpowszechniony w latach siedemdziesiątych do ogólnego użytku, początkowo miał nazwę Lucyfer. We wczesnych latach siedemdziesiątych nie znane były nikomu algorytmy szyfrowania do momentu opatentowania przez firmę IBM algorytmu DES i jego rozpowszechnienia. Standard szyfrowania danych jest szyfrem blokowych, co znaczy że dane są dzielone na bloki o długości 64 bitów i następnie szyfrowane. Danymi wejściowymi algorytmu jest blok tekstu jawnego o długości 64 bitów, który po przetworzeniu przez algorytm przedstawiono jako szyfrogram. Długość klucza w 64 bitowym bloku wynosi 56 bitów, gdzie co ósmy bit jest bitem parzystości. Algorytm DES składa się z 16 cykli, które są wykonywane jeden po drugim. Na schemacie XXX poglądowo pokazano schemat działania algorytmu DES. Algorytm można podzielić na poszczególne kroki:

- zamiana 64 bitowego klucza decymalnego na postać binarną i usunięcie co ósmego bitu zwanego bitem parzystości w wyniku czego otrzymano 56 bitowy binarny klucz
- permutacja 56 bitowego klucza binarnego zgodnie z tabelą 6 dla każdego z 16 cykli
- podział 56 bitowego klucza na prawą o lewą część o długości 28 bitów i przesunięcie bitowe, które jest zależne od cyklu co ukazuje tabela 7
- permutacja kompresji klucza DES zgodnie z tabelą 8 kompresująca klucz 56 bitowy do klucza 48 bitowego dla każdego z 16 cykli ukazanego na rysunku 5 jako Ki
- zamiana wiadomości tekstowej na wiadomość w postaci heksadecymalnej i binarną
- transpozycja wiadomości binarnej z tabelą permutacji początkowej
- podział wiadomości 64 bitowej na dwie części 32 bitowe: strona prawa oznaczona literą P na rysunku 5 i strona lewa oznaczona literą L na rysunku 5
- prawa cześć wiadomości ulega rozszerzeniu zgodnie z tabelą 9 permutacji rozszerzenia w wyniku czego otrzymano 48 bitową wiadomość
- operacja XOR obliczonego klucza dla danej rundy z 48 bitową wiadomością otrzymaną w wyniku permutacji rozszerzenia
- transpozycją wyniku operacji XOR z tablicą S bloków na rysunku 5 oznaczono S-boxes
- wyjście S bloku ulega transpozycji z tabelą permutacji bloku P na rysunku 5 oznaczono jako P box
- wiadomość po permutacji bloku P poddano operacji XOR z lewą częścią wiadomości w wyniku czego otrzymano 32 bitową część prawą wiadomości
- lewą częścią 32 bitowej wiadomości jest prawa część 32 bitowej wiadomości bez żadnych zmian

Dane wyjściowe wiadomości z rundy pierwszej są danymi wejściowymi dla rundy drugiej algorytmu. Kroki przedstawione powyżej należy wykonać 16 razy w calu zaszyfrowania wiadomości, która jest podzielona na bloki 64 bitowe. Na obrazku 5 zobrazowano pojedynczą rundę algorytmu DES.



Rysunek 5: Pojedyncza runda algorytmu DES [5].

W tabeli 4 ukazano reprezentację binarną liczą heksadecymalnych w systemie 64 bitowym.

0	1	2	3	4	5	6	7
0000	0001	0010	0011	0100	0101	0110	0111
8	9	A	B	C	D	E	F
1000	1001	1010	1011	1100	1101	1110	1111

Tabela 4: Reprezentacja binarna liczb heksadecymalnych.

W tabeli 5 przedstawiono tablicę dla permutacji początkowej wiadomości dla algorytmu DES. Permutacji dokonuje się poprzez zamianę miejscami bitów zgodnie z tabelą permutacji otrzymując w wyniku przekształcone dane wejściowe.

58	50	42	34	26	18	10	2	60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6	64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1	59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5	63	55	47	39	31	23	15	7

Tabela 5: Permutacja początkowa DES.

Tabela 6 ukazuje permutacje klucza 56 bitowego, który powstaje z klucza 64 bitowego w wyniku

usunięcia co ósmego bitu parzystości.

57	49	41	33	25	17	9	1	58	50	42	34	26	18
10	2	59	51	43	35	27	19	11	3	60	52	44	36
63	55	47	39	31	23	15	7	62	54	46	38	30	22
14	6	61	53	45	37	29	21	13	5	28	20	12	4

Tabela 6: Permutacja klucza DES.

Przesunięcia bitowe dwóch kluczy 28 bitowych tworzących wejściowy klucz 56 bitowy dla każdego cyklu zależą od numeru cyklu, co ukazano w tabeli 7.

cykl	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
liczba przesunięć	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Tabela 7: Liczba przesunięć klucza w zależności od cyklu.

Po operacji przesunięcia bitowego klucza 56 bitowego wykonano kompresję klucza do rozmiaru 48 bitów zgodnie z tabelą 8.

14	17	11	24	1	5	3	28	15	6	21	10
23	19	12	4	26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40	51	45	33	48
44	49	39	56	34	53	46	42	50	36	29	32

Tabela 8: Permutacja kompresji DES.

Tabela 9 jest stosowana do operacji rozszerzenia wiadomości. Wiadomość 64 bitowa zostaje podzielona na dwa 32 bitowe bloki, które następnie ulegają rozszerzeniu do dwóch 48 bitowych bloków zgodnie z tabelą 9 permutacji rozszerzenia. Rozszerzone 48 bitowe bloki zawierają powtórzenia bitów, rozszerzenie 32 do 48 bitów powoduję wystąpienie 16 bitów które nie są unikatowe. Oznacza to, że 8 bitów występuje podwójnie w szyfrowanej wiadomości.

32	1	2	3	4	5	4	5	6	7	8	9
8	9	10	11	12	13	12	13	14	15	16	17
16	17	18	19	20	21	20	21	22	23	24	25
24	25	26	27	28	29	28	29	30	31	32	1

Tabela 9: Permutacja rozszerzenia DES.

Po dokonaniu permutacji rozszerzenia wiadomości dokonano operacji xor otrzymanej wiadomości z obliczonym kluczem dla danego cyklu. Otrzymany wynik podzielono na 8 bloków 6 bitowych,

które wykorzystano do dalszych obliczeń z pomocą tabeli 10 struktury S bloków. Z każdego bloku 6 bitowego odczytano wartość pierwszego i ostatniego bitu, wartość 00 oznacza wykorzystanie wiersza pierwszego z danego bloku, 01 oznacza wiersz drugi, 10 oznacza wiersz trzeci, natomiast 11 oznacza wiersz czwarty. Wartości od bita drugiego do piątego zamieniamy na postać heksadecymalną i odczytujemy dla niej wartość z danego bloku np. jeśli pierwszy i ostatni bit to 00 to wartość odczytano z wiersza pierwszego, gdy bity między 2-5 pozycją to 1111 oznacza to ostatnią wartość z wiersza pierwszego czyli 7.

S blok 1															
14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S blok 2															
15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S blok 3															
10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S blok 4															
7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S blok 5															
2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S blok 6															
12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S blok 7															
4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S blok 8															
13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Tabela 10: Struktury S bloków.

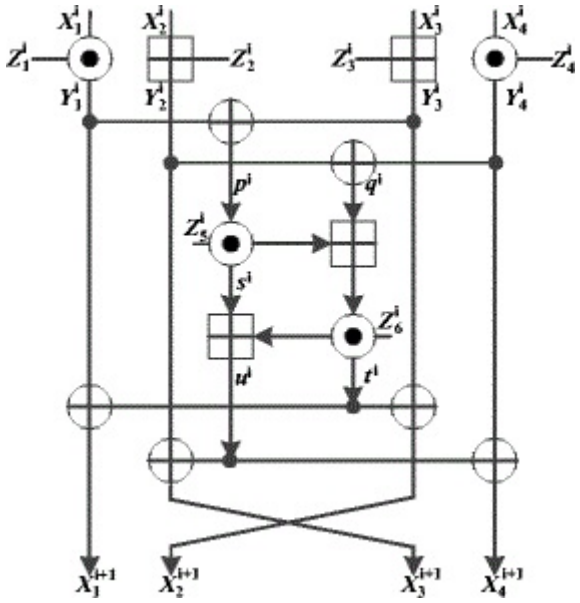
16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

Tabela 11: Permutacja bloku P.

Algorytm DES nie jest obecnie uznawany za bezpieczny przez Amerykańskie Standardy Federalne. W roku 2015 przy użyciu wysokiej wydajności FPGA możliwe było złamanie szyfru w stosunkowo krótkim czasie(4-5 dni). Dla atakujących przestała być odstraszająca cena platformy FPGA w porównaniu do zysków jakie może dać im atak. W celu przesłania klucza do drugiej osoby, z którą tworzymy bezpieczne połączenie wymaga znalezienie bezpiecznej metody przesyłu klucza. Zaletami algorytmu DES jest jego odporność na błędy, dzięki jego prostocie aplikacji. W obecnych czasach jest on nadal popularny w użytku, dzięki swojej szybkości [5].

IDEA

IDEA to Międzynarodowy Algorytm Szyfrowania Danych(z ang. International Data Encryption Algorithm). Początkowo nosił nazwę PES(z ang. Proposed Encryption Standard) opatentowany przez Xuejia Lai i Jamesa Massey w 1990 roku. Następnie został on ulepszony przed atakiem zmieniając nazwę na IPES(z ang.Improved Proposed Encryption Standard). Ostatecznie jego nazwę zmieniono na obecnie używaną IDEA w 1992 roku. Jest to algorytm blokowy, który wykorzystuje bloki 64 bitowe i klucz o długości 128 bitów. Do szyfrowania i odszyfrowywania wykorzystuje się ten sam algorytm. Algorytm ten początkowo dzieli blok 64 bitowy na 4 pod bloki 16 bitowe, na których dokonywane są operację: XOR, dodawanie modulo i mnożenie modulo. Algorytm składa się z ośmiu rund. Na rysunku 6 przedstawiono pojedyncza rundę algorytmu IDEA, gdzie pierwsza 16 bitowa część wiadomości podlega operacji mnożenia modulo z 16 bitowym kluczem rundy. Operacja modulo oznaczona przy pomocy kółka z kropką w środku $\odot$. Druga 16 bitowa część wiadomości poddano operacji dodawania modulo z 16 bitowym kluczem rundy. Operacje dodawania modulo oznaczono kwadratem z plusem w środku $\boxplus$. Operację XOR oznaczono okręgiem z plusem w środku $\oplus$. Operacje mnożenia, dodawania i xorowania są wykonywane na częściach wiadomości zgodnie z rysunkiem 6. Po operacjach xorowania, dodawania i mnożenia dokonujemy przestawienia bloku drugiego z trzecim, które są danymi wejściowymi do następnej rundy. W każdej rundzie używany jest 16 bitowy klucz dla danego pod bloku. Po zakończeniu ośmiu rund, cztery bloki są złożone do jednego bloku 64 bitowego, który jest wynikiem końcowym.



Rysunek 6: Pojedyncza runda algorytmu IDEA [6].

Dla operacji dodawania modulo potrzebne są dwie liczby, na których zostanie wykonane działanie oraz tzn. licza modulo. Operacja dodawania modulo przebiega w następujących krokach: dodanie do siebie dwóch liczb, a następnie podzielenie wyniku dodawania przez liczbę modulo. Reszta z wyniku dzielenia sumy dwóch licz przez liczbę module daje nam wynik operacji dodawania modulo. Operacje dodawania modulo oznaczono symbolem $\boxplus$. Przykładem operacji dodawania modulo jest: $2 \boxplus 5 \equiv 1 \pmod 3$. Dla operacji mnożenia modulo potrzebne są dwie liczby, na których zostanie wykonane działanie oraz tzn. liczba modulo. Operacja mnożenia modulo polega na iloczynie dwóch czynników oraz ilorazie wyniku operacji z liczbą modulo. Wynikiem jest reszta z operaji dzielenia przez liczbę modulo. Operacje mnożenia modulo oznaczono symbolem $\odot$. Przykładem operacji mnożenia modulo jest $2 \odot 7 = 2 \pmod 3$, bo 2*7 = 14, gdzie 14/3 daje resztę z dzielenia 2. Klucz o długości 128 bitów jest dzielony na osiem 16 bitowych podkluczy. Dla każdej rundy zostaje wykorzystane 6 podkluczy, które oznaczono na rysunku 6 literą Z. Do wszystkich rund, których jest 8 użyto 52 podkluczy oraz 4 podklucze użyte do końcowych przekształceń. Dla algorytmu IDEA użyto łącznie 52 podkluczy, które stworzono z 128 bitowego klucza. Generację klucza 128 bitowego do 52 podkluczy można podzielić na:

- podział 128 bitowego klucza na 8 podkluczy 16 bitowych, gdzie pierwsze 6 podkluczy wykorzystano do rundy 1, a następne 2 podklucze użyto do rundy 2 algorytmu
- przesunięcie klucza 128 bitowego w lewo o 25 bitów i ponowny podział na 8 podkluczy, gdzie pierwsze 4 podklucze użyto do rundy drugiej, a następne 4 podklucze wykorzystano w rundzie 3
- klucz jest dalej analogicznie przesuwany i dzielony, aż do momentu uzyskania 52 podkluczy

Proces deszyfrowania wiadomości również korzysta z schematu przedstawionego na rysunku 6 pojedynczej rundy algorytmu, składa się on również z 8 rund. Proces tworzenia podkluczy z klucza odbywa się na podstawie podkluczy wykorzystanych do zaszyfrowania wiadomości. Przy utworzeniu klucza pomocna jest tabela multiplikatywnej odwrotności 4 bitowej liczby binarnej przedstawionej w tabeli 12.

wartość binarna[decymalna]	multiplikatywna odwrotność[decymalna]	addytywna odwrotność[decymalna]
0000 [0]	0000 [0]	0000 [0]
0001 [1]	1111 [15]	0001 [1]
0010 [2]	1110 [14]	1001 [9]
0011 [3]	1101 [13]	0110 [6]
0100 [4]	1100 [12]	1101 [13]
0101 [5]	1011 [11]	0111 [7]
0110 [6]	1010 [10]	0011 [3]
0111 [7]	1001 [9]	0101 [5]
1000 [8]	1000 [8]	1111 [15]
1001 [9]	0111 [7]	0010 [2]
1010 [10]	0110 [6]	1100 [12]
1011 [11]	0101 [5]	1110 [14]
1100 [12]	0100 [4]	1010 [10]
1101 [13]	0011 [3]	0100 [4]
1110 [14]	0010 [2]	1011 [11]
1111 [15]	0001 [1]	1000 [8]

Tabela 12: Obliczenia multiplikatywnej i addytywnej odwrotności [8].

Multiplikatywną odwrotność dla a wyliczono według zależności x mod n, gdzie a $\in Z_n$ oraz n jest rozmiarem grupy. Do obliczenia odwrotności addytywnej wykorzystano rozszerzony algorytm Euklidesa, który oblicza liczby całkowite x i y wykorzystując zależność ax + ny = 1, gdzie a $\in Z_n$. Proces tworzenia podkluczy wykorzystanych do deszyfrowania ukazano w tabeli 13. Potęga oznaczona literą a określa multiplikatywną odwrotność danej liczby, potęga oznaczona jako m określa addytywną odwrotność podanej liczby.

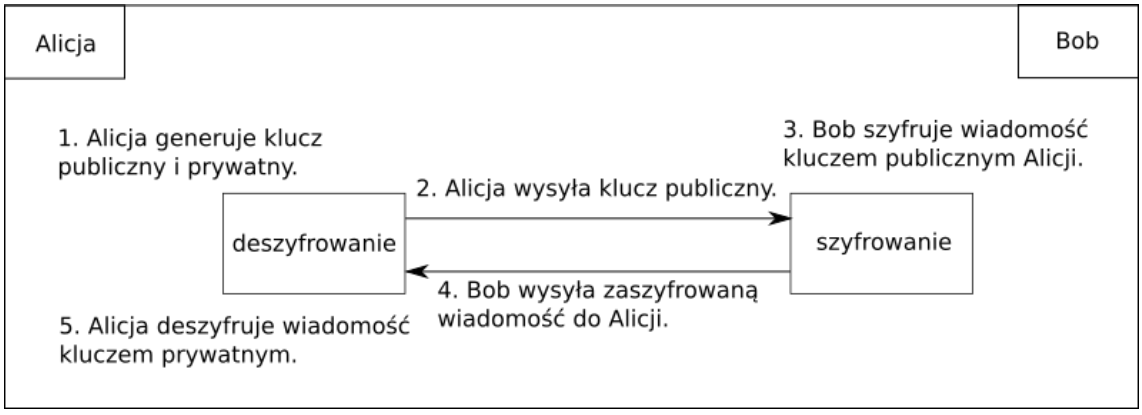
cykl	podklucze szyfrujące						podklucze deszyfrujące					
1	x_1	x_2	x_3	x_4	x_5	x_6	x^a_{49}	x^m_{50}	x^m_{51}	x^a_{52}	x_{47}	x_{48}
2	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x^a_{43}	x^m_{45}	x^m_{44}	x^a_{46}	x_{41}	x_{42}
3	x_{13}	x_{14}	x_{15}	x_{16}	x_{17}	x_{18}	x^a_{37}	x^m_{39}	x^m_{38}	x^a_{40}	x_{35}	x_{36}
4	x_{19}	x_{20}	x_{21}	x_{22}	x_{23}	x_{24}	x^a_{31}	x^m_{33}	x^m_{32}	x^a_{34}	x_{29}	x_{30}
5	x_{25}	x_{26}	x_{27}	x_{28}	x_{29}	x_{30}	x^a_{25}	x^m_{27}	x^m_{26}	x^a_{28}	x_{23}	x_{24}
6	x_{31}	x_{32}	x_{33}	x_{34}	x_{35}	x_{36}	x^a_{19}	x^m_{21}	x^m_{20}	x^a_{22}	x_{17}	x_{18}
7	x_{37}	x_{38}	x_{39}	x_{40}	x_{41}	x_{42}	x^a_{13}	x^m_{15}	x^m_{14}	x^a_{16}	x_{11}	x_{12}
8	x_{43}	x_{44}	x_{45}	x_{46}	x_{47}	x_{48}	x^a_7	x^m_9	x^m_8	x^a_{10}	x_5	x_6
dodatkowe 4 podklucze	x_{49}	x_{50}	x_{51}	x_{52}			x^a_1	x^m_2	x^m_3	x^a_4		

Tabela 13: Podklucze szyfrujące i deszyfrujące IDEA.

Jedną z podstawowych zalet algorytmu są proste operacje na bitach tj. mnożenie i dodawanie modulo oraz xorowanie. Do szyfrowania i deszyfrowania wiadomości wykorzystuje ten sam proces, natomiast podklucze są różne dla szyfrowania i odszyfrowywania. Algorytm IDEA wolniej szyfruje dane niż algorytm DES, jednak dzięki temu, że jego klucz jest dwa razy dłuższy niż w DES ataki brutalne są mniej skuteczne [6].

2.4.2 Szyfry asymetryczne

Szyfry asymetryczne charakteryzują się tym, że posiadają osobny klucz do szyfrowania i osobny deszyfrowania. W celu wyjaśnienia zasady działania szyfru asymetrycznego przedstawiono powszechnie znany sposób wysyłania wiadomości od Alicji do Boba przedstawiony schematycznie na rysunku 7.



Rysunek 7: Schematyczne szyfrowanie asymetryczne.

Do szyfrowania używany jest klucz publiczny, natomiast do odszyfrowania wiadomości użyto klucza prywatnego. Bob generuje publiczny klucz, który może być znany dla każdego, gdyż tylko Bob mający klucz prywatny może odszyfrować wiadomość zaszyfrowaną przy pomocy jego pu-

blicznego klucza. Szyfrowanie asymetryczne jest bezpieczne nawet w przypadku wystąpienia osoby trzeciej Ewy, która chciałaby podsłuchać konwersację między Bobem, a Alicją. Bob po wygenerowaniu klucza publicznego wysyła go do Alicji, aby ta mogą zaszyfrować wiadomość, którą chce do niego przesłać. W przypadku gdy Ewa przechwyci klucz publiczny Boba, nie jest w stanie odszyfrować wiadomości, gdyż nie posiada klucza prywatnego Boba, w którego posiadaniu jest tylko on. W celu przesłania wiadomości od Boba do Alicji proces wygląda tak samo. Na rysunku 7 zobrazowano szyfrowanie i deszyfrowanie z użyciem klucza publicznego i prywatnego. Alicja generuje klucz publiczny, który wysyła do Boba. Bob szyfruje wiadomość publicznym kluczem Alicji i przesyła jej zaszyfrowaną wiadomość, która Alicja odszyfruje przy użyciu jej klucza prywatnego. Obecność dwóch kluczy w algorytmie rozwiązuje problem wymiany klucza, co pozwala osobom będącym daleko od siebie w bezpieczny sposób się komunikować [7].

RSA

Algorytm RSA jest jednym z powszechniejszym algorytmów. Algorytm RSA zawdzięcza swoją nazwę trzem osobom, przez które został on opatentowany przez Rona Rivest, Adi Shamir i Leonard Adleman w 1977 roku. Nazwa algorytmu RSA pochodzi od pierwszych liter nazwisk wynalazców. Algorytm RSA bazuje na liczbach pierwszych oraz faktoryzacji dużych liczb. Proces tworzenia pary kluczy: prywatnego i publicznego przebiega w następujących krokach:

- wybór dwóch odpowiednio dużych licz pierwszych, które po przemnożeniu dadzą klucz o pożądanym rozmiarze np. 2048 bitów

  $$n = p * q$$

  ,gdzie n = wynik mnożenia liczb pierwszych, p i q liczby pierwsze.

- obliczenie funkcji Eulera według zależności

  $$m = (p-1)(q-1)$$

- określenie liczby e względnie pierwszej z m oraz spełnia zależność 1¡e¡m

- obliczenie liczby d według zależności

  $$d*e \bmod(m) = 1$$

- obliczone liczby e i d są publiczne, natomiast liczba d jest trzymana w tajemnicy

- zaszyfrowanie wiadomości c z wykorzystaniem wartości publicznych

  $$c = m^e \bmod(n)$$

- do odszyfrowania wiadomości wykorzystano liczbę prywatną d według zależności

  $$p = c^d \bmod(n)$$

Diffie-Hellman

Algorytm Diffie-Hellman został opatentowany w 1976 roku przez Whitfield Diffie i Martin Hellman. Dzięki temu algorytmowi możliwe jest przesyłanie klucza przez niezabezpieczone łącze w bezpieczny sposób. Obecnie jest on stosowany do wymiany klucza symetrycznego. Algorytm można opisać w następujących krokach:

- Alicja generuje prywatną liczbę całkowitą a
- Bob generuje prywatną liczbę całkowitą b
- wybór liczby pierwszej p
- obliczenie publicznej wartości z wykorzystaniem liczby piwerszej p, wartości g oraz prywatnej liczby a (dla Alicji) i b (dla Boba) według zależności

$$g^a \text{mod(p) i } g^b \text{mod(p)}$$

- wymiana publicznej wartości między Alicją i Bobem
- Alicja oblicza

$$g^{ab} = (g^b)^a \text{mod(p)}$$

- Bob oblicza

$$g^{ba} = (g^a)^b \text{mod(p)}$$

- Alicja i Bob posiadają wspólny tajny klucz k, gdyż

$$g^{ab} = g^{ba}$$

Algorytm Diffie-Hellmana przestanie być bezpieczny w momencie, gdy zostanie rozwiązany problem logarytmu dyskretnego. Wartościami publicznymi powiązanymi poniższą relacją są g, p, a i b.

$$g^a \equiv \text{a mod(p) oraz } g^b \equiv \text{b mod(p)}$$

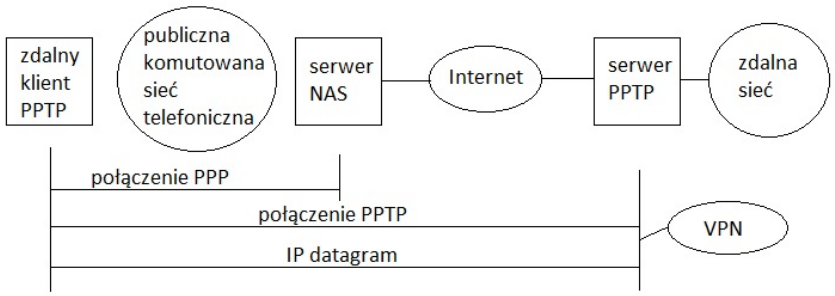
Obecnie nie istnieje żaden algorytm umożliwiający rozwiązanie problemu logarytmu dyskretnego [10].

2.5 Protokoły

2.5.1 VPN with Point-to-Point Tunneling Protocol (PPTP)

Protokół punkt - punt często wykorzystywany na urządzeniach z systemem operacyjnym Windows w celu utworzenia wirtualnej sieci prywatnej. Protokół PPTP został utworzony w czasach sieci

wdzwanianej tzn. dial up, jednak bez problemu można z niego korzystać również dla sieci Ethernet, Internet czy cyfrowa sieć usług zintegrowanych(ISDN). Utworzenie sicei VPN w sieci TCP/IP umożliwia bezpieczny transfer danych między zdalnym komputerem, a serwerem. Do implementacji protokołu głównie jest wykorzystywany: klient PPTP, serwer NAS i serwer PPTP. Serwer NAS bezpiecznie przechowuje dane i udostępnia je tylko wybranym użytkownikom. Na rys. 8 poniżej poglądowo przedstawiono komunikację z użyciem protokołu PPTP.



Rysunek 8: Poglądowa komunikacja z użyciem protokołu PPTP

Połączenie między zdalnym klientem, a siecią przebiega w określonych etapach:

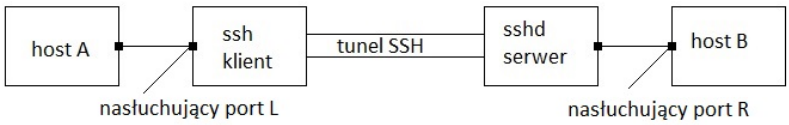
- nawiązanie połączenia z użyciem protokołu komunikacyjnego point-to-point (PPP) między dwoma węzłami sieci w sposób telefoniczny bądź przewodowy z Internetem
- połączenie PPP z serwerem PPTP, co tworzy połączenie VPN między zdalnym klientem, a serwerem

Po ustanowieniu połączenia wysyłane są dwa pakiety danych: pakiet kontrolny, który zarządza tunelem i pakiet danych. Protokół PPTP jest zaaplikowany na drugim poziomie modelu odniesienia łączenia systemów otwartych (OSI), który jest warstwą łącza danych. Jego działanie jest oparte na protokole punkt-punkt (PPP), który umożliwia korzystanie z usług internetowych. Protokół PPTP umożliwia firmom tworzenie prywatnych tuneli w publicznym internecie. Organizacje nie muszą już instalować kosztownych linii do komunikacji, dzięki protokołowi PPTP mogą w bezpieczny sposób korzystać z sieci publicznej w celu przesyłania poufnych danych. Protokół PPTP wspiera szyfrowanie i kompresję danych. Protokół PPTP nie jest wykorzystywany powszechnie ze względu na podatność na ataki [11].

2.5.2 Secure Shell (SSH)

SSH z angielskiego to Secure Shell, co znaczy bezpieczna powłoka. Protokół SSH został wprowadzony w celu ulepszenia istniejących już wcześniej protokołów takich jak telnet, ftp czy BSD r-commands(rlogin, rexec, rsh,rcp). Protokoły telnet, ftp czy rsh były wykorzystywane do transferu plików między hostem lokalnym i zdalnym. Wymienione protokoły nadal są w powszechnym

użytku, tylko tam gdzie bezpieczeństwo nie jest istotnym czynnikiem. Istotną wadą usług telnet i ftp jest brak szyfrowania i uwierzytelniania podczas przesyłania pakietów. Dane wysyłane są w sposób jawny, które mogą być w łatwiejszy sposób przechwycone przez atakującego. Atakujący może uzyskać niezaszyfrowane hasło, które następnie może wykorzystać w łatwy sposób. Wszędzie tam gdzie bezpieczeństwo danych jest ważne stosujemy protokół SSH. SSH jest protokołem warstwy transportowej, która używa TCP do przenoszenia swoich pakietów. TCP jest protokołem sterowanie transmisją między dwoma urządzeniami. SSH umożliwia bezpieczne tunelowanie między lokalnym z zdalnym urządzeniem. Na poniższym rysunku przedstawiono komunikacje lokalnego klienta z powłoką serwera przy pomocy przekierowaniu portów. Na rys. 9 ukazano schemat lokalnego przekierowania portu dla protokołu SSH.

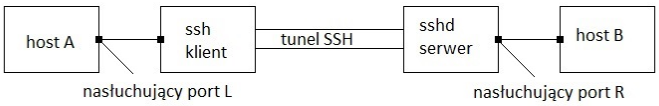


Rysunek 9: SSH lokalne przekierowywanie portu

Host A wykonuje bezpiecznie połączenie do hostu B poprzez port R. Połączenie hostu A z hostem B przebiega w następujący sposób:

- host A łączy się do porty L na kliencie SSH
- SSH przekierowuje połączenie poprzez bezpieczny tunel do serwera SSH
- serwer SSH łączy się z hostem B poprzez port R

Port L na hoście A i port R na hoście B ustawione są w trybie ciągłego nasłuchiwania dla lokalnego połączenia. Na rys. 10 przedstawiono zdalne przekierowywanie portu z hosta B do hosta.



Rysunek 10: SSH zdalne przekierowywanie portu

Zdalne przekierowanie portu przebiega w następujących krokach:

- klient SSH wysyła żądanie zdalnego przekazywania portów, co powoduje nasłuchiwanie połączeń na porcie R po stronie serwera
- przekiwrowanie połączenie przez bezpieczny tunel do klienta SSH
- klient SSH łączy się do hosta A poprzez nasłuchujący port L po jego stronie

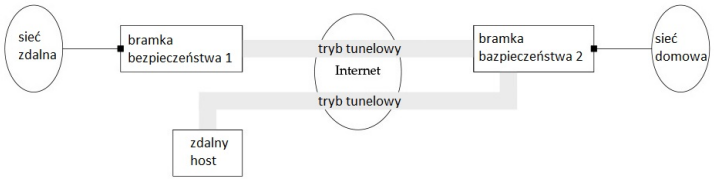
Wykorzystując zjawisko tunelowania do prywatnej sieci wirtualnej tworzymy bezpieczne połączenie poprzez tunel między dwoma sieciami [12].

2.5.3 IPsec

Protokół IPsec zawiera trzy główne protokoły:

- protokół uwierzytelnienie nagłówków (AH) - zapewnia uwierzytelnienie i integralność pakietów protokołu internetowego (IP)
- protokół bezpieczeństwa danych ESP - zapewnia te same usługi co protokół AH i dodatkowo zapewnia bezpieczeństwo danych
- protokół wymiany klucza internetowego IKE - zapewnia zarządzanie kluczem, umożliwiając bezpiecznie skojarzenie dwóch hostów

Protokoły AH i ESP mogą działać w dwóch trybach: transportowym i tunelowym. Tryb transportowy zapewnia bezpieczeństwo dla warstw powyżej datagramu IP. Jest on inicjowany głównie między dwoma stałymi hostami, połączenie nie może być nawiązana między dwoma sieciami lub między siecią i hostem. Zaś tryb tunelowy zapewnia bezpieczeństwo całego datagramu IP. Jest on stosowany do połączenia tunelowego między dwoma sieciami lub między hostem, a siecią. Na rys. 11 poniżej przedstawiono schematyczny tryb tunelowy dla protokołu IPsec.



Rysunek 11: Tryb tunelowy IPsec

Sieć zdalna jest połączona z siecią domową za pośrednictwem sieci tunelowej, bramki bezpieczeństwa 1 i bramki bezpieczeństwa 2. Funkcją bramek jest szyfrowanie, deszyfrowanie, zapobieganie ponownego odtwarzania wysyłanych pakietów i uwierzytelnianie. Z poziomu hostów bramki bezpieczeństwa i tunelowanie jest niewidoczne. Występują trzy metody uwierzytelnienie danych z udziałem protokołu bezpieczeństwa IPsec: wstępnie uzgodnione klucze, klucze prywatne i publiczne RSA oraz certyfikaty. Nagłówki uwierzytelniające dla trybu transportowego i tunelowego różnią się od siebie. Powszechny pakiet IP wchodzący w wyższą warstwę TCP składa się z: nagłówka IP, nagłówka TCP i danych użytkownika. Dla trybu transportowego nagłówek uwierzytelnienia dla IPsec składa się z: nagłówka IP, nagłówka uwierzytelniającego, nagłówka TCP i danych użytkownika. W tunelowym trybie nagłówek uwierzytelniający kopiuje część wewnętrzną nagłówka IP, która jest używana do utworzenia nowego zewnętrznego nagłówka IP. Nagłówek składa się z: zewnętrznego nagłówka IP, nagłówka uwierzytelniającego, wewnętrznego nagłówka IP, nagłówka TCP i danych użytkownika.

IPsec korzysta z protokołu Internet Key Exchange (IKE) w celu nawiązania i ustanowienie bezpiecznego połączenia między dwoma hostami lub zdalnego dostępu tunelowania VPN. Protokół IKE występuje w dwóch wersjach IKEv1 i IKEv2. Protokół IKEv1 można podzielić na dwie fazy, pierwsza w nich odpowiada za następujące funkcjonalności: algorytmy szyfrowania, algorytmy haszowania, protokół Diffiego-Hellmana oraz metodę uwierzytelniania. Faza druga IKEv1 jest głównie używana do szyfrowania w protokole IPsec. Do szyfrowania wykorzystujemy algorytmy Data Encryption Standard (DES), Triple DESo długości 168 bitów czy Advanced Encryption Standard (AES) o długości 128-256 bitów. Spośród wymienionych powyżej algorytmów DES zapewnia najniższe bezpieczeństwo danych, zaś AES o długości klucza 256 bitów największe bezpieczeństwo. Algorytmy haszujące zastosowane w protokole IPsec to: Secure Hash Algorithm (SHA) oraz Message Digest 5 (MD5). Protokół MD5 jest mniej bezpieczny niż algorytm SHA. Kiedy używamy IPsec wraz z VPN przesyłane dane będą zabezpieczone podczas przesyłania, w celu uniemożliwienia atakującemu zdobycie jawnych danych. Proces szyfrowania, który zapewnia poufność przesyłanych danych nie daje możliwości łatwej modyfikacji wrażliwych danych atakującemu.

Do stworzenia protokołu IPsec zrzeszenia bezpieczeństwa(Security Association AS) używanych jest kilka komponentów, które służą do przetwarzania ruchu tekstu jawnego, który jest chroniony i następnie przekształcany w tekst zaszyfrowany. Te same komponenty są używane do odszyfrowania danych. Protokół IPsec korzysta z trzech baz danych:

- Baza danych zasad zabezpieczeń - Security Policy Database (SPD) określa jaki ruch ma być chroniony
- Baza danych stowarzyszenia zabezpieczeń - Security Association Database (SAD) określa w jaki sposób ruch jest chroniony
- Baza danych autoryzacji - Peer Authorization Database (PAD) zapewnia mechanizm wymuszający politykę z oparciu o protokół IKE

2.5.4 Secure Socket Tunneling protocol Based on VPN

Protokół SSTP umożliwia administratorom na ustanowienie tunelów poprzez główną sieć korporacyjną na Windows Serwer. Dzięki zredukowanie wymaganych kroków technicznych do utworzenia tunelów między organizacjami zyskujemy niższy koszt administracyjny. Brak dodatkowych zmian w infrastrukturze dla protokołu SSTP, gdyż podobnie jak w protokóle HTTPS jest obsługiwana zapora sieciowa i serwery proxy. Protokół SSTP wykorzystuje protokołu SSL do transportu ruchu sieciowego. SSL jest certyfikatem odpowiedzialnym za poświadczenie wiarygodności domeny bądź jej właściciela, co zapewnia bezpieczeństw szyfrowania danych. PPP to protokół połączenie punkt-punkt. Protokołem odpowiedzialnym za sterowanie transmisją jest TCP. Proces utworzenia połączenia wykorzystującego VPN między klientem, a serwerem przebiega w następujący sposób:

- klient ustanawia połączenie TCP z serwerem SSTP pomiędzy dynamicznie zaalokowanym portem TCP po stronie klienta SSTP, a portem TCP 443 na serwerze SSTP
- klient SSTP wysyła wiadomość Witaj SSL świadczącą o chęci nawiązania połączenia SSL z serwerem SSTP
- serwer SSTP wysyła do klienta cyfrowy certyfikat
- klient SSTP sprawdza poprawność certyfikatu, wybiera metodę szyfrowania dla sesji SSL, generuje klucze i następnie szyfruje certyfikat serwera przy użyciu klucza publicznego
- klient SSTP wysyła zaszyfrowany klucz sesji SSL do serwera SSTP
- serwer SSTP odszyfrowuje zaszyfrowany klucz sesji SSL z wykorzystaniem klucza prywatnego; dalsza komunikacja klienta z serwerem jest zaszyfrowana wybraną metodą szyfrowania i odszyfrowywana kluczem sesji SSL
- klient wysyła komunikat HTTP poprzez sesję SSL do serwera
- klient ustala połączenie tunelowe z serwerem
- klient SSTP ustala połączenie PPP z serwerem SSTP, które obejmuje uwierzytelnienie danych logowanie użytkownika wraz z uwierzytelnieniem PPP i ustawieniem konfiguracji ruchu IP
- klient SSTP rozpoczyna proces ruchu sieciowego IP poprzez łącze PPP [13]

2.5.5 L2TP

Protokół tunelujący warstwy 2 (Layer 2 Tunneling Protocol L2TP) działa na warstwie łączy danych, która jest drugą w siedmio warstwowym modelu odniesienia łączenia systemów otwartych (OSI). Protokół ten nie zapewnia szyfrowania ani poufności danych, z tego powodu jest często używany z protokołem IPsec który zapewnia szyfrowanie i poufność danych. Protokół służy do komunikacji między klientem zwanym jako koncentrator dostępu L2TP(LAC) oraz serwerem nazywanym sieciowym serwerem L2TP (LNS). Protokół L2TP, który jest wykorzystywany do włączenie sieci VPN w Internecie jest rozszerzeniem protokołu PPTP. Powstał on w połączeniu protokołu firmy PPTP opracowanej przez Microsoft i L2F (przekierowywanie warstwy drugiej modelu OSI) firmy Cisco. Połączenie użytkownika domowego do sieci firmowej wykorzystujące protokół L2TP i połączenie internetowe przedstawiono w następujących krokach:

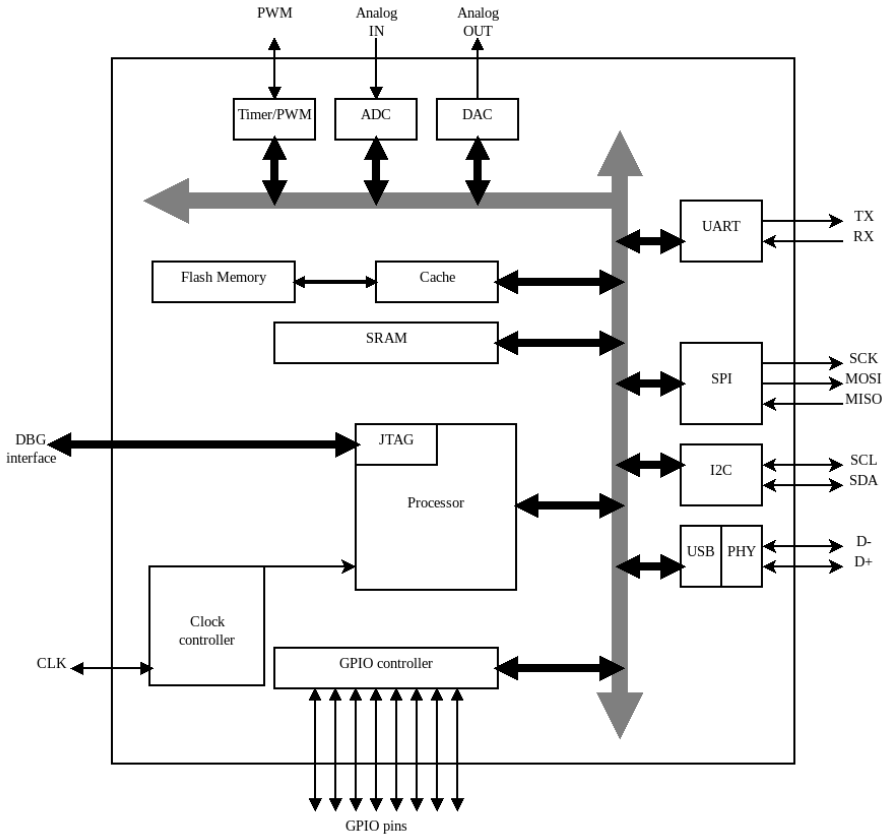
- zdalny użytkownik korzysta z analogowego połączenia telefonicznego lub łącza szerokopasmowego w celu zainicjowania połączenie PPP do dostawcy usług internetowych
- sieć LAC akceptuje połączenie co prowadzi do ustanowienia połączenia PPP
- w czasie gdy użytkownik końcowy i serwer nawiązują połączenie, koncentrator dostępu LAC rozpoczyna uwierzytelniania użytkownika metodą CHAP lub PAP
- po pomyślnie ukończonym procesie uwierzytelnienia, połączenie użytkownika z serwerem LNS zostaje pomyślnie nawiązane; w przypadku niepomyślnego procesu uwierzytelnienie klient uzyskuje dostęp do Internetu jako normalny użytkownik
- punkty końcowe tunelu (LAC i LNS) przed wysłaniem danych poprzez tunel uwierzytelniają się wzajemnie
- po utworzeniu tunelu VPN korzystającego z protokołu L2TP połączenie między użytkownikiem, a siecią korporacyjną zostaje nawiązane

Protokół L2TP jest protokołem sieci komputerowej wykorzystywanym przez dostawców internetowych do utworzenia wirtualnej sieci prywatnej między dwoma urządzeniami [14].

3 Systemy wbudowane

3.1 Budowa

W obecnych czasach wbudowane systemy zyskają coraz większą popularność poprzez stosowanie ich w przedmiotach codziennego użytku między innymi: odkurzacz automatyczny, elektryczna hulajnoga, maszyny sprzedające, systemy alarmowe, telefony komórkowe czy drukarki. W przypadku niektórych zastosowań systemów wbudowanych np.telefony komórkowe czy systemy alarmowe niezbędne jest szyfrowanie wiadomości na nich występujących. W przypadku braku bloku kryptograficznego na systemie wbudowanym jest on podatny na atak np. w przypadku włamania do mieszkania z alarmem może on być wyłączony przez hakera w zdecydowanie krótszym czasie niż w przypadku takiego systemu alarmowego, na którym dane są całkowicie szyfrowane. Systemy wbudowane nie posiadają ściśle określonych architektur, projektowane są w zależności od zapotrzebowania bądź uniwersalne. Głównym argumentem przemawiającym za systemami wbudowanymi jest ich niski koszt produkcji oraz niski pobór mocy. Na rysunku 12 przedstawiono ogólny schemat blokowy komponentów zawartych wewnątrz mikrokontrolera.



Rysunek 12: Schemat blokowy komponentów wewnątrz ogólnego mikrokontrolera [15]

.

Modulator szerokości impulsu PWM

Modulator szerokości impulsów jest wykorzystywany do regulowania sygnału napięciowego lub prądowego przy stałej amplitudzie i częstotliwości. PWM jest wykorzystywany do kontrolowanie mocy dostarczonej do urządzeń elektronicznych sterowanych przy pomocy mikrokontrolera. Wejściowy pulsujący sygnał jest generowany na sekwencyjne sygnały impulsowe w postaci fali o kształcie prostokątnym. Fala o charakterze prostokątnym posiada dwa stany: wysoki i niski, które są osiągalne poprzez przełączanie tranzystorów bądź tyrystorów w stan przewodzenia bądź zaporowy. Stan wysoki sygnały wyjściowego oznacza stan przewodzenia tranzystora, zaś stan niski oznacza stan zaporowy. Amplituda w sygnale prostokątnym jest to różnica między stanem wysoki, a stanem niskim napięcia w przypadku PWM sterowanego sygnałem napięciowym. Okres fali wyjściowej to czas trwania jego jednego cyklu, zaś częstotliwością fali jest odwrotność trwania okresu. Istotnym parametrem określającym wyjściową falę prostokątną jest również cykl roboczy, który jest stosunkiem procentowym czasu wysokiego stanu do okresu oznaczanego literą T.

Wejścia/wyjścia

Wejście analogowe ADC

Wejście analogowe zwane również jako przewrotnik sygnału analogowo-cyfrowego ADC. Przetwornik jest układem elektronicznym, który analogowy sygnał wejściowy konwertuje proporcjonalnie do cyfrowego sygnału wyjściowego. Proces ten polega na pomiarze wejściowego napięcia i jego zamiany na binarny numer wyjściowy. Jego zastosowanie można ujrzeć np. podczas mierzenia temperatury czy ważenia przedmiotów lub ludzi, gdzie sygnał analogowy jest zamieniany na cyfrowy i jego wartość jest interpretowana odpowiednio przez system wbudowany.

Wyjście analogowe DAC

Wyjście analogowe zwane jako przetwornik sygnały cyfrowo-analogowego. Przetwornik DAC jest układem elektronicznym, który konwertuje wejściowe dane binarne na analogowe dane wyjściowe. Jego zastosowanie można wykorzystać np. do sterowania silnika krokowego wyjściowym sygnałem napięciowym bądź wyświetlania godziny na zegarku analogowym.

Wejścia/Wyjścia ogólnego przeznaczenia GPIO

Wejścia- wyjścia ogólnego przeznaczenia są prostym protokołem służącym do sterowania urządzeniami korzystając z sygnałów cyfrowych. GIPO wykorzystywane jest do komunikacji między urządzeniami systemu wbudowanego. W celu skorzystania GIPO musi zostać odpowiednio skonfigurowane dla danego urządzenia.

Uniwersalny odbiornik i nadajnik asynchroniczny UART

Uniwersalny odbiornik i nadajnik asynchroniczny zwany jako UART służy do komunikowania się z systemem wbudowanym. Jest on jednym z powszechnie stosowanych protokołów komunikacyjnych. UART konwersuje równoległe dane na szeregowy strumień bitów danych. Komponent ten nie posiada zegara. Ramka wiadomości wykorzystywana do komunikacji z wykorzystaniem UART składa się z:

- bit początkowy sygnalizujący początek komunikacji
- bity danych
- bit parzystości, który jest wykorzystywany do wykrywania błędów występujących podczas komunikacji
- bit stopu, który informuje nas o końcu komunikacji

Na rysunku 12 symbolem TX(z ang. transmitted data) oznaczone są dane wyjściowe, natomiast symbolem RX(z ang. received data) oznaczono dane wejściowe. Komponent ten jest wykorzystywany np. do czytnika linii papilarnych, gdzie po zeskanowaniu odcisku dane są przesyłane poprzez komponent UART w celu dalszego ich przetworzenia.

Magistrala

Magistrale są wykorzystywane do przesyłania danych między urządzeniami występującymi w danym systemie wbudowanym. W zależności od sposobu przesyłania danych magistrale można podzielić na jednokierunkowe, gdzie przesył danych jest odbywa się tylko w jednym kierunku i dwukierunkowe, gdzie przesył danych odbywa się w dwóch kierunkach. Magistrale dwukierunkowe(ang.duplex) można podzielić na full duplex, gdzie przesyłanie danych odbywa się jednocześnie między urządzeniami oraz na half duplex, gdzie przesył danych między urządzeniami może odbywać się tylko w jednym kierunku w określonym czasie.

Szeregowy interfejs urządzeń peryferyjnych SPI

SPI służy do komunikowania się urządzeń peryferyjnych tj. klawiatura, mysz, monitor, skaner, drukarka, mikrofon, głośnik czy kamera internetowa z centralną jednostką obliczeniową. Dzięki protokołowi pełnego dupleksu umożliwia on jednoczesne wysyłanie i odbieranie danych. SPI jest szybszy od protokołu I2C, lecz wynaga większej ilości połączeń w porównaniu z I2C. Urządzenia komunikujące się między sobą określono jako master i slave. W SPI występuje jedno urządzenie, które ma charakter master i do kilkunastu urządzeń określanych jako slave. Urządzenie master jest w hierarchii wyżej od urządzenia slave, które jest jemu podporządkowane. Do przesyłania danych między urządzeniem podlegającym(slavem), a głównym(master) wykorzystana jest linia MOSI(z ang. master out, slave in). Dane przesyłane między urządzeniem głównym, a podlegającym

odbywają się na linii MISO(z ang. master in, slave out). Przesyłanie danych z urządzenia głównego do podlegającego i z urządzenia podlegającego do urządzenia głównego odbywa się w tym samym czasie, zgodnie z taktem zegara systemowego(SCK). Większość urządzeń SPI obsługuje przesył tylko 8 bitowych danych.

I^2C

Magistrala I^2 komunikuje się między urządzeniami w sposób szeregowy i dwukierunkowy. I^2C oprócz pinów służących do uziemienia i zasilania korzysta tylko z dwóch dodatkowych pinów. Dwie dodatkowe linie oznaczono jako SDA i SCL. Linie do przesyłania danych oznaczono SDA, zaś linie zegara oznaczono SCL.

USB,PHY

USB jest magistralą komunikującą się w sposób szeregowy. Służy ona do przesyłania danych między urządzeniami bądź zasilania podłączonego urządzenia. Została ona zaprojektowania w celu standaryzacji magistrali przesyłających lub zasilających urządzenia peryferyjnych tj. klawiatura, urządzenia wskazujące czy monitor. USB posiada cztery piny: zasilający, dwa piny danych(D- i D+) oraz uziemiający. PHY jest to warstwa fizyczna modelu odniesienia łączenia systemów otwartych(OSI). PHY jest odpowiedzialne za połączenie warstwy łącza do kabla miedzianego lub światłowodu.

Zegar czasu rzeczywistego RTC

Zegar czasu rzeczywistego służy do śledzenia aktualnego czasu i daty, które są wykorzystywane jako znacznik przypisujący danym plikom ich czas utworzenia czy ostatniej modyfikacji. Zegar jest zasilany własną niezależną baterią, co znaczy że działa on również gdy zasilanie nie jest doprowadzone go układu. Bateria zasilająca zegar posiada żywotność ponad 10 lat. Ważnym parametrem zegara jest jego częstotliwość. Częstotliwość zegara jest to ilość instrukcji wykonana w czasie jednej sekundy, więc im większa częstotliwość zegara, tym więcej instrukcji jest wykonanych w tej samej jednostce czasu.

Centralna jednostka obliczeniowa - procesor

Głównym komponentem składowym centralnej jednostki obliczeniowej zwanej procesorem są tranzystory, które działają w oparciu o sygnały prądowe lub napięciowe. Tranzystory są ze sobą połączone w celu utworzenia bramek logicznych tj. AND, OR, NOT, NAND, NOR, EX-OR i EX-NOR. Bramka logiczna NOT posiada jedno wejście i jedno wyjście, gdy bitem wejściowym jest 0 to na wyjściu otrzymujemy 1, natomiast gdy bitem wyjściowym jest 1 na wyjściu otrzymano wartość 0. Natomiast pozostałe bramki logiczne posiadają dwa bity wejściowe oznaczone jako A i

B oraz jeden bit wyjściowy oznaczony C. Ich zależności logiczne zostały przedstawione w poniższej tabeli 14.

A	B	C[AND]	C[NAND]	C[OR]	C[NOR]	C[EX-OR]	c[EX-NOR]
0	0	0	1	0	1	0	1
0	1	0	1	1	0	1	0
1	0	0	1	1	0	1	0
1	1	1	0	1	0	0	1

Tabela 14: Bramki logiczne.

Procesor jest stworzony w biliona takich bramek logicznych, więc im więcej tranzystorów posiada procesor tym wyższa jest jego wydajność i moc obliczeniowa. Wydajność procesora jest również zależna częstotliwości jego zegara, więc im większa częstotliwość zegara tym wyższa wydajność procesora. Liczba operacji procesora jest proporcjonalna do częstotliwości zegara. Obecnie powszechne są procesory wielordzeniowe, czyli wiele procesorów będących na jednym chipie. Pozwala to na równoległe wykonywanie instrukcji procesora przez co zwiększona jest jego wydajność. W skład głównych komponentów procesora wychodzą jednostka arytmetyczno-obliczeniowa zwana ALU, jednostka sterująca, pamięć podręczna, pamięć główna, pamięć dyskowa oraz urządzenia wejścia i wyjścia. Procesory można podzielić za względu na architektura na 32 i 64 bitowe. Procesory jedno rdzeniowe z architekturą 32 bitową wykonują operację czytania lub pisania 32 bitów w jednej jednostce czasowej, natomiast procesory z architekturą 64 bity w jednej jednostce czasowej.

JTAG

JTAG jest standardem IEEE 1149.1 do testowania połączeń oraz debugowania płytek drukowanych. Interfejs JTAG korzysta z pięciu linii sygnałowych: wejścia danych(TDI), wyjścia danych(TDO), wejścia sygnału zegarowego(TCK), wyboru trybu pracy(TMS) i zerowania(TRST). JTAG jest powszechnym interfejsem, który umożliwia komunikację zewnętrznego komputera z komponentami znajdującymi się na systemie wbudowanym.

Pamięć

Głównym zadaniem pamięci jest przechowywanie danych w postaci binarnej. Systemy wbudowane działają na dwóch stanach, gdzie stan wysoki odpowiada dodatniemu napięciowi, zaś stan niski jest reprezentowany przez napięcie zerowe. Stan wysoki oznaczamy jako 1 w notacji binarnej, zaś stan niski jako 0 w notacji binarnej. Rozmiar pamięci wyrażamy w bitach [b], bajtach [B], kilobajt [KB], megabajt [MB], gigabajt [GB] i tarabajtach [TB]. Jednostka kilobajt jest równa 1024 bajt, 1 megabajt równa się 1024 kilobajtów, 1 gigabajt równy jest 1024 megabajtów, zaś 1 terabajt równy jest 1024 gigabajtów.

SRAM

SRAM jest to pamięć statyczna o swobodnym dostępie. Jest to pamięć statyczne, gdyż dane zawarte na niej są tak długo jak długo jest zasilany układ. Pamięć SRAM jest pamięcią ulotną. Magistrale danych i adresów łączą bezpośrednio procesor z pamięcią SRAM. Pamięć SRAM jest szybsza od pamięci DRAM w operacjach odczytu i zapisu danych, co jest następstwem jej większej ceny. Pamięć SRAM często jest wykorzystywana jako pamięć podręczna lub pamięć wewnętrzna procesora ze względu na swoją szybkość. Jedną z wad pamięci SRAM jest to, że wymaga ona więcej mocy do działania przez co wytwarza więcej ciepła i wymaga lepszego chłodzenia w porównaniu w pamięcią DRAM.

Pamięć podręczna procesora Cache

Procesor systemu wbudowanego nieustanie czyta i zapisuje informacje z jego pamięci podręcznej. Głównymi cechami pamięci podręcznej jest mały rozmiar, tymczasowość i ich szybkość. Pamięć podręczną dzielimy na dwa poziomy. Pamięć podręczna pierwszego poziomu oznaczona L1 znajduje się na procesorze i jest mniejsza od pamięci poziomu drugiego. Pamięć podręczna poziomu drugiego oznaczona L2 znajduję się między procesorem, a pamięcią główną i jest większa od pamięci L1. W niektórych przypadkach pamięć pierwszego L1 i drugiego poziomu L2 jest umieszczona razem w pamięci procesora oraz jest dodawany poziom trzeci L3 pamięci podręcznej. Poziomy pamięci podręcznej opisuje zależność L1<L2<L3.

Pamięć flash

Pamięć flash jest pamięcią nieulotną, co znaczy, że po odłączeniu zasilania znajdujące się na niej dane nie zostaną utracone. Pamięci flash są powszechnie stosowane w kartach pamięci, pamięciach USB czy dyskach SSD.

Szyny danych, adresowe i sterowania

Wszystkie wewnętrzne komponenty znajdujące się na systemie wbudowanym do komunikacji używają trzech szyn. Szyna danych jest odpowiedzialna za przesyłanie danych między nimi, szyna adresowa odpowiedzialna jest za wysyłanie adresów, natomiast szyna sterowania jest odpowiedzialna za wysyłanie instrukcji między komponentami [15].

Część II

Część praktyczna

4 Aplikacja algorytmów kryptograficznych na systemie wbudowanym

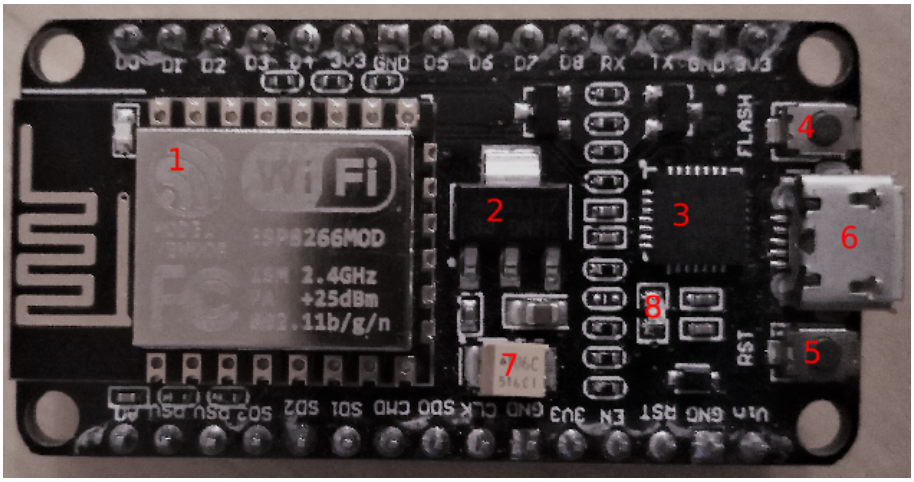
4.1 Charakterystyka ESP8266

Mikrokontroler składa się z płytki NodeMCU v2 oraz układ ESP8266. Układ ESP8266 z wbudowanym 32 bitowym układem RISC(zredukowany zestaw instrukcji) z takowanym zegarem 80MHz połączono 22 pinami do płytki. Głównymi komponentami moduł WiFi ESP8266 NodeMCU v2 są :

- moduł Wi-Fi ESP8266MOD, 2,4GHz, 802.11b/g/n
- stabilizator napięcia AMS1117 3.3 DN711
- konwerter USB-UART umożliwiający programowanie poprzez USB przy wykorzystaniu środowiska Arduino
- pamięć Flash 4MB
- 10 portów GIPO, które mogą obsługiwać magistrale PWN, I2C i l-wire
- wbudowane 10-bitowe ADC
- 10-bitowy przetwornik analogowo-cyfrowy
- antena PCB
- dwa przełączniki stanowe
- wejście mikro USB
- kondensator tantalu, dioda niebieska [16]

Do projektu wykorzystano płytkę NodeMCU v2 ESP8266, której przód przedstawiono na rysunku 13, zaś tył na rysunku 14. Numery na rysunku 13 oznaczają:
1. moduł Wi-Fi ESP8266MOD 2,4GHz, 802.11b/g/n
2. stabilizator napięcia AMS1117 3.3 DN711
3. most USB-UART
4. przełącznik stanowy Flash
5. przełącznik stanowy RST
6. wejście mikro USB
7. kondensator tantalu 106C 516C1

8. dioda niebieska 0603



Rysunek 13: Przód mikrokontrolera NodeMCU v2 ESP8266.

Rysunek 14: Przód mikrokontrolera NodeMCU v2 ESP8266.

4.2 Połączenie ESP8266 z Arduino

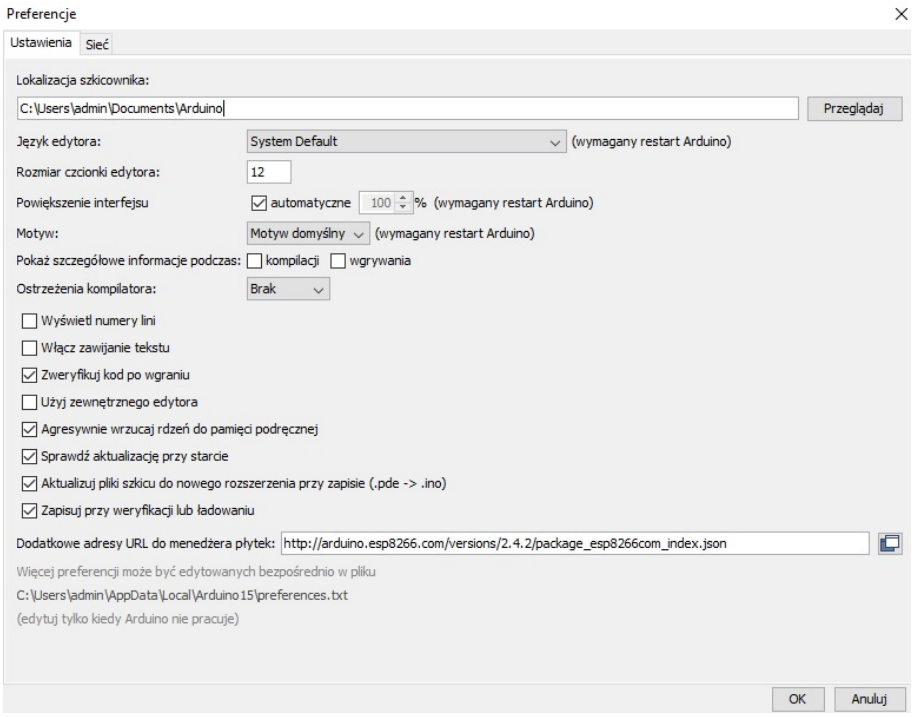
Założenia projektowe:

- wykorzystanie mikrokontrolera ESP8266 NodeMCU v2
- wybrano język programowania C
- środowisko programistyczne Arduino uruchomione na systemie operacyjnym windows 10

- implementacja algorytmu DES na mikrokontrolerze
- pomiar czasu potrzebnego na szyfrowanie algorytmem DES
- pomiar czasu potrzebnego na deszyfrowanie algorytmem DES

Podłączenie mikrokontrolera do środowiska programistycznego Arduino przebiegło w następujących krokach:

- pobranie i instalacja środowiska Arduino 1.8.7 IDE
- konfiguracja środowiska Arduino do obsługi ESP 8266 przez otwarcie okna dialogowego plik → preferencje i w miejscu "Dodatkowe adresy URL do menedżera płytek" umieszczenie adresu $http://arduino.esp8266.com/stable/package_esp8266com_index.json$ jak ukazano na rysunku 15 znajdującym się poniżej
- instalacja paczki dla płytki ESP8266 poprzez otwarcie okna dialogowego narzędzia → Płytka → Menadżer płytek ... → wpisanie esp8266 w polu wyszukiwania i instalacja paczki
- konfiguracja płytki ESP826
  - Narzędzia → Płytka → NodeMCU 1.0 (ESP-12E Module)
  - Narzędzia → Flash Size → 4M (3M SPIFFS)
  - Narzędzia → CPU Frequency → 80 Mhz
  - Narzędzia → Upload Speed → 921600
  - Narzędzia → Port → COM3



Rysunek 15: Konfiguracja środowiska Arduino dla płytki ESP8266.

4.3 Manualna implementacja algorytmu DES

W podrozdziale 2.4.1 zostały opisane teoretyczne kroki działania algorytmu DES, które zostały poniżej zobrazowane dla rundy pierwszej. W tabeli 15 ukazano wiadomość w postaci decymalnej i heksadecymalnej oraz heksadecymalny klucz, które zostały użyte do przedstawienie działania algorytmu DES.

wiadomość	witaj :)
wiadomość w postaci heksadecymalnej	77 69 74 61 6A 20 3A 29
klucz	43 23 66 A3 6B BB 53 C1

Tabela 15: Dane wejściowe algorytmu DES.

Jednym z pierwszych kroków jest zamiana klucza heksadecymalnego do postaci binarnej co zostało ukazane w tabeli 16.

klucz	4	3	2	3	6	6	A	3
klucz w postaci binarnej	0100	001**1**	0010	001**1**	0110	011**0**	1010	001**1**
numer bitu	1-4	5-8	9-12	13-16	17-20	21-24	25-28	29-32
klucz	6	B	B	B	5	3	C	1
klucz w postaci binarnej	0110	101**1**	1011	101**1**	0101	001**1**	1100	000**1**
numer bitu	33-36	37-40	41-44	45-48	49-52	53-56	57-60	61-64

Tabela 16: Klucz w postaci binarnej.

Po przekształceniu heksadecymalnego klucza do postaci binarnej usuwamy bit parzystości z 64 bitowego bloku czyli 8, 16, 24, 32, 40, 48, 56 i 64 bit. W tabeli 16 bit parzystości oznaczono pogrubioną kursywą. Następnie z wykorzystaniem tabeli 6 dokonujemy przestawienie bitów klucza, odszukano bit 57 klucza, który wynosi 0 i wpisano do nowo utworzonej tabeli 17 w lewym górnym rogu. W tak przedstawiony sposób dokonano translacji dla dalszych bitów klucza, a wynik przedstawiono w tabeli 17.

permutacja klucza	57	49	41	33	25	17	9	1	58	50	42	34	26	18		
numer bitu	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
klucz binarny	0	1	0	0	0	0	1	1	0	0	1	0	0	0	1	1
klucz po permutacji	1	0	1	0	1	0	0	0	1	1	0	1	0	1		
permutacja klucza	10	2	59	51	43	35	27	19	11	3	60	52	44	36		
numer bitu	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
klucz binarny	0	1	1	0	0	1	1	0	1	0	1	0	0	0	1	1
klucz po permutacji	0	1	0	0	1	1	1	1	1	0	0	1	1	0		
permutacja klucza	63	55	47	39	31	23	15	7	62	54	46	38	30	22		
numer bitu	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
klucz binarny	0	1	1	0	1	0	1	1	1	0	1	1	1	0	1	1
klucz po permutacji	0	1	1	1	1	1	1	1	0	0	0	0	0	1		
permutacja klucza	14	6	61	53	45	37	29	21	13	5	28	20	12	4		
numer bitu	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
klucz binarny	0	1	0	1	0	0	1	1	1	1	0	0	0	0	0	1
klucz po permutacji	0	0	0	0	1	1	0	0	0	0	0	0	0	0		

Tabela 17: Klucz w postaci binarnej.

Klucz po permutacji z postaci 64 bitowej redukuje się do postaci 56 bitowej poprzez usunięcie bitów parzystości i dzieli na dwie części: prawą i lewą. W tabeli 18 w górnym wierszy ukazane jest pierwsze 28 bitów klucza tzn. lewa część L oraz w dolnym wierszu przedstawiono następne 28 bitów klucza, które zwane są prawą częścią klucza P.

1010 1000 1101 0101 0011 1110 0110
0111 1111 0000 0100 0011 0000 0000

Tabela 18: Obliczony klucz w postaci 56 bitowej.

Klucz został podzielony na dwie osobne części, gdyż każda z nich podlega osobnym dalszym operacjom. W tabeli 19 ukazano przesuniecie bitowe o 1 klucza w lewo wykonane na lewej i prawej części klucza.

klucz	0101	0001	1010	1010	0111	1100	1101
numer bitu	1-4	5-8	9-12	13-16	17-20	21-24	25-28
klucz	1111	1110	0000	1000	0110	0000	0000
numer bitu	29-32	33-36	37-40	41-44	45-48	49-52	53-56

Tabela 19: Przesunięcie bitowe obliczonego klucza w postaci 56 bitowej.

Ostatnim etapem obliczania klucza dla rundy pierwszej jest permutacja kompresji klucza DES zgodnie z tabelą 8 kompresji klucza oraz przedstawienie wyniku w postaci heksadecymalnej co ukazano w tabeli 20.

permutacja kompresji DES	14 17 11 24	1 5 3 28	15 6 21 10	23 19 12 4	26 8 16 7	27 20 13 2
bitowy klucz 56bitowy po permutacji	0010	0001	1010	0101	1100	0111
heksadecymalny klucz 56bitowy po permutacji	2	1	A	5	C	7
permutacja kompresji DES	41 52 31 37	47 55 30 40	51 45 33 48	44 49 39 56	34 53 46 42	50 36 29 32
bitowy klucz 56bitowy po permutacji	1010	1010	0010	0000	1010	0011
heksadecymalny klucz 56bitowy po permutacji	A	A	2	0	A	3

Tabela 20: Permutacja klucza 56 bitowego.

Kluczem obliczonym dla pierwszej rundy to 21A5C7AA20A3.

Następnym krokiem algorytmu jest przetworzenie w odpowiedni sposób jawnej wiadomości, którą przedstawiono w postaci heksadecymalnej i decymalnej w tabeli 21.

wiadomość heksadecymalna	7	7	6	9	7	4	6	1
wiadomość binarna	0111	0111	0110	1001	0111	0100	0110	0001
numer bitu	1-4	5-8	9-12	13-16	17-20	21-24	25-28	29-32
wiadomość heksadecymalna	6	A	2	0	3	A	2	9
wiadomość binarna	0110	1010	0010	0000	0011	1010	0010	1001
numer bitu	33-36	37-40	41-44	44-48	49-52	53-56	57-60	61-64

Tabela 21: Wiadomość przedstawiona w postaci binarnej i heksadecymalnej.

Następnym krokiem było dokonanie obliczenia permutacji początkowej wiadomości zgodnie z tabelą 5 co ukazano w tabeli 22.

wiadomość binarna po permutacji	0001	1111	0100	0101	0000	0101	1000	1011
wiadomość heksadecymalna po permutacji	1	F	4	5	0	5	8	B
wiadomość binarna po permutacji	0000	0000	1111	1111	1101	0010	0101	0001
wiadomość heksadecymalna po permutacji	0	0	F	F	D	2	5	1

Tabela 22: Transpozycja wiadomości w postaci binarnej z tabelą permutacji początkowej.

Lewa strona wiadomości to L = 1F45058B, prawa strona P = 00FFD251. Dla rundy pierwszej algorytmu lewą stroną wiadomości zostaje prawa część wiadomości, natomiast prawą część wiadomości obliczamy następująco:

- prawa część wiadomości podlega transpozycji zgodnie z tabelą 9 permutacji rozszerzenia,
- operacja XOR wiadomości otrzymanej w wyniku transpozycji z obliczonym kluczem to EF108A230793,
- prawa część wiadomości podlega transpozycji zgodnie z tabelą 9 permutacji rozszerzenia dając wynik przedstawiony w tabeli 23.

prawa część wiadomości	00FFD251
	0000 0000 1111 1111 1101 0010 0101 0001
permutacja rozszerzenia	10000000 00010111 11111111 11101010 01000010 10100010

Tabela 23: Permutacja rozszerzenia prawej części wiadomości.

W tabeli 24 przestawiono wynik operacji xor zmodyfikowanej wiadomości z kluczem.

wiadomość	10000000 00010111 11111111 11101010 01000010 10100010
klucz	00100001 10100101 11000111 10101010 00100000 10100011
XOR	10100001 10110010 00111000 01000000 01100010 00000001

Tabela 24: XOR wiadomości po permutacji rozszerzenia z kluczem.

Następnym krokiem szyfrowania wiadomości było dokonanie podstawienia w S-blokach zgodnie z tabelą 10 s bloków. Wynik operacji XOR z tabeli 24 podzielono na S1 = 101000, S2 = 011011, S3 = 001000, S4 = 111000, S5 = 010000, S6 = 000110, S7 = 001000 i S8 = 000001. Wartości S bloków odczytano z użyciem tabeli 10. Pierwszy w ostatni bit bloku oznacza rząd z tabeli 10 S-bloków, wartości 00 oznaczają wiersz 0, 01 oznaczają wiersz 1, 10 oznaczają wiersz 2, zaś 11 oznaczają wiersz 3. Dla wartości S1 pierwszy i ostatni bit bloku S1 = 10, co znaczy korzystanie z wiersza 2 i bloku S1. Następnie cztery bity między 1-5 bitem zmieniono na wartość decymalną. Dla S1 postać binarna 0100 wynosi 4 w postaci decymalnej, z tabeli 10 odczytujemy z wiersza 1, kolumny 4 wartość 14, więc S1 = 14. W sposób analogiczny odczytano pozostałe wartości S-bloków, które przedstawiono w tabeli 25.

decymalnie	14	9	6	5	8	15	15	1
heksadecymalnie	D	9	6	5	8	F	F	1
binarnie	1110	1001	0110	0101	1000	1111	1111	0001
numer bitu	1-4	5-8	9-12	13-16	17-20	21-24	25-28	29-32

Tabela 25: Wiadomość po operacji na S blokach.

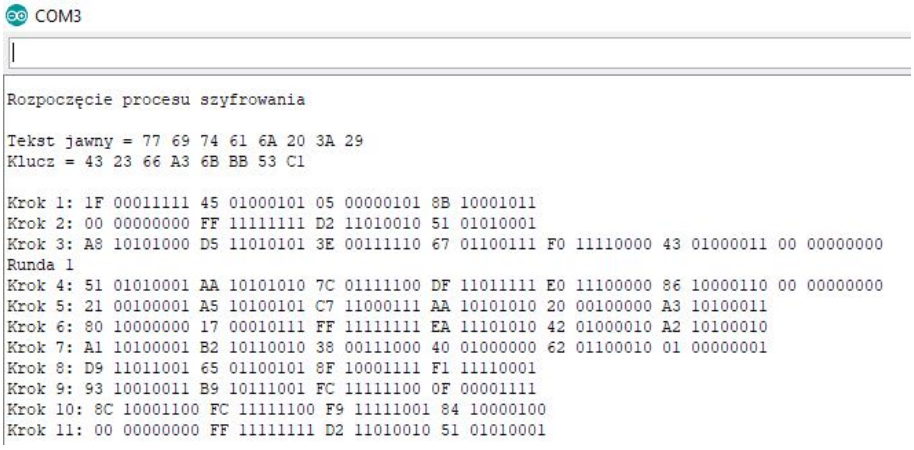
Jednym w ostatnich kroków szyfrowania wiadomości jest dokonanie transpozycji wyjścia S bloku z tabelą permutacji P-bloku 11 w wyniku czego otrzymano: 1001 0011 1011 1001 1111 1100 0000 1111. Otrzymaną wartość poddano operacji XOR z lewą połową wiadomości (1F45058B), co przedstawiono jako (0001 1111 0100 0101 0000 0101 1000 1011) $\oplus$ (1001 0011 1011 1001 1111 1100 0000 1111) = 1000 1100 1111 1100 1111 1001 1000 0100, co w postaci heksadecymalnej daje wartość 8CFCF984. Wartość prawej strony wiadomości dla rundy pierwszej wynosi 8CFCF984. Lewą częścią wynikową wiadomości jest niezmodyfikowana prawa część 32 bitowa wiadomości czyli 00FFD251.

4.4 Implementacja algorytmu DES na ESP8266

Algorytm DES zaimplementowano zgodnie z krokami przedstawionymi w manualnym obliczaniu szyfrowania algorytmem DES, którą szczegółowo przedstawionego na rysunku 5. W celu poprawienia czytelności wyniku wypisywanego na monitor portu szeregowego poszczególne etapy pojedynczej rundy DES oznaczono jako krok 1, krok 2 itd. Na rysunku 16 ukazano poszczególne kroki wyliczane z wykorzystaniem algorytmu DES. Tekst jawny oraz klucz wykorzystano taki sam jak w przykładzie manualnej implementacji DES. Poszczególne kroki oznaczają:

- Krok 1: Permutacja początkowa prawej części 32 bitowej wiadomości.
- Krok 2: Permutacja początkowa lewej części 32 bitowej wiadomości.
- Krok 3: Klucz po operacji permutacji klucza.
- Krok 4: Przesunięcie bitowe klucza 56 bitowego.

- Krok 5: Permutacja kompresji klucza 48 bitowego.
- Krok 6: Permutacja rozszerzenia prawej części 32 bitowej wiadomości w wyniku, której otrzymano 48 bitową wiadomość.
- Krok 7: Operacja XOR klucza danej rundy i prawej części 48 bitowej wiadomości.
- Krok 8: Operacje na S blokach.
- Krok 9: Operacja na P bloku.
- Krok 10: Permutacja początkowa prawej części 32 bitowej wiadomości.
- Krok 11: Permutacja początkowa lewej części 32 bitowej wiadomości.

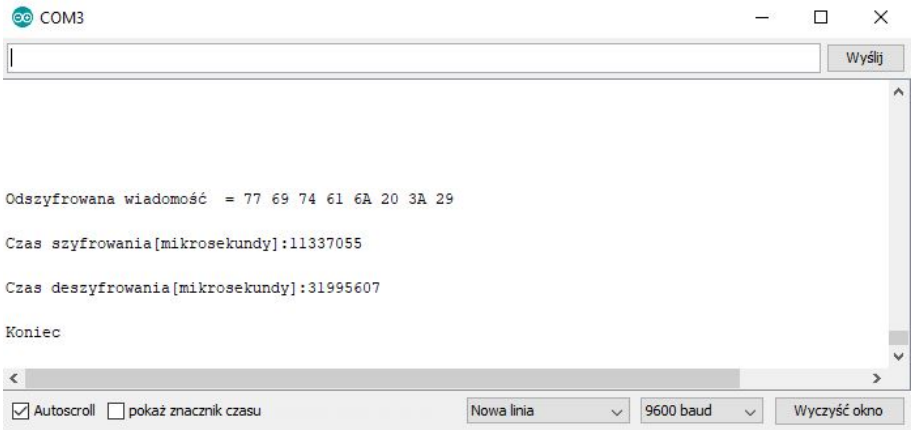


Rysunek 16: Wynik szyfrowania algorytmem DES dla rundy 1.

Porównując wynik implementacji programu w środowisku Arduino na płytce NodeMCU v2 ESP8266 wraz z manualnym obliczeniem stwierdzono zgodność wyników dla rundy pierwszej algorytmu DES, co oznacza poprawną implementację algorytmu na mikrokontrolerze. Następnie dokonano pomiaru czasu szyfrowania i deszyfrowania wiadomości dla trzech prędkości szybkości transmisji danych, która jest określana w bitach na sekundę. Pomiarów dokonano dla szybkości transmisji: 9600, 19200 i 115200 bitów na sekundę. Szybkość transmisji danych mikrokontrolera określono poleceniem "Serial.begin(9600);", zaś w celu odczytania wyników na monitorze portu szeregowego ustawiono 9600. Pomiaru czasu szyfrowania i deszyfrowania wykonano z wykorzystaniem funkcji micros(), która zwraca czas wyrażony w mikrosekundach wykonywania programu. W celu obliczenia czasu szyfrowania i deszyfrowania obliczono różnicę czasu jako:

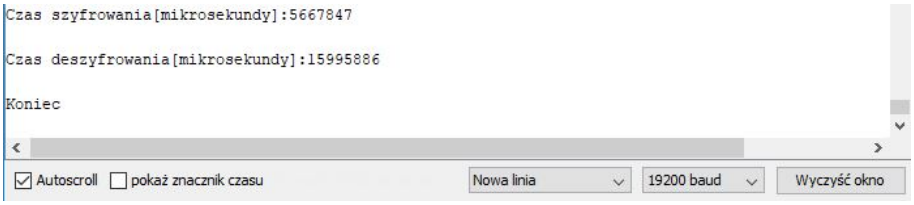
```
unsigned long poczatek = micros();
szyfrowanie(tekst_jawny, klucz);
unsigned long koniec = micros();
unsigned long czas_szyfrowania = koniec - poczatek;
```

Pierwszym etapem było zmierzenie czasu szyfrowania i deszyfrowania dla przypadku, gdy dane obliczeń były wypisywane na monitor portu szeregowego. Na rysunku 17, 18 i 19 przedstawiono wyniki czasów w przypadku wypisywania poszczególnych kroków na konsole. Na rysunku 17 przedstawiono wyniki czasu szyfrowania i deszyfrowania przy prędkości transmisji 9600 bitów na sekundę dla algorytmu wypisującego poszczególne kroki obliczeń na monitor portu szeregowego.



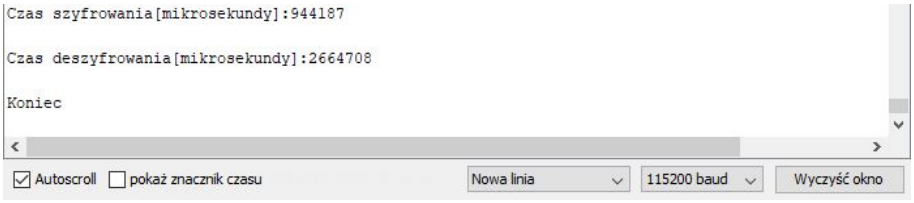
Rysunek 17: Czas szyfrowania i deszyfrowania dla prędkości transmisji danych 9600.

Na rysunku 18 przedstawiono obliczony czas potrzebny do szyfrowania i deszyfrowania algorytmem DES dla prędkości transmisji danych 19200 bitów na sekundę dla algorytmu wypisującego poszczególne kroki obliczeń na monitor portu szeregowego.



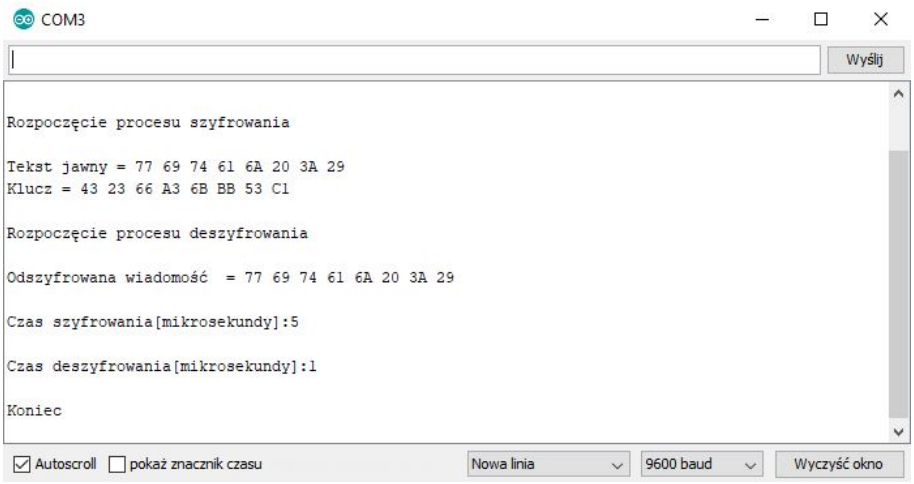
Rysunek 18: Czas szyfrowania i deszyfrowania dla prędkości transmisji danych 19200.

Na rysunku 19 przedstawiono obliczony czas potrzebny do szyfrowania i deszyfrowania algorytmem DES dla prędkości transmisji danych 115200 bitów na sekundę dla algorytmu wypisującego poszczególne kroki obliczeń na monitor portu szeregowego.



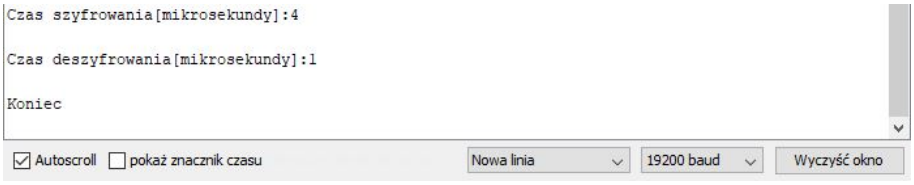
Rysunek 19: Czas szyfrowania i deszyfrowania dla prędkości transmisji danych 115200.

Drugim etapem było wyeliminowanie wypisywania na monitor portu szeregowego obliczeń poszczególnych kroków dla algorytmu DES i obliczenie rzeczywistego czasu zużytego na szyfrowania i deszyfrowanie wiadomości. Na rysunku 20, 21 i 22 przedstawiono wynik czasu obliczeń dla algorytmu DES dla różnych prędkości przesyłu transmisji danych. Na rysunku 20 przedstawiono obliczone czas potrzebny do szyfrowania i deszyfrowania algorytmem DES dla prędkości transmisji danych 9600 bitów na sekundę.



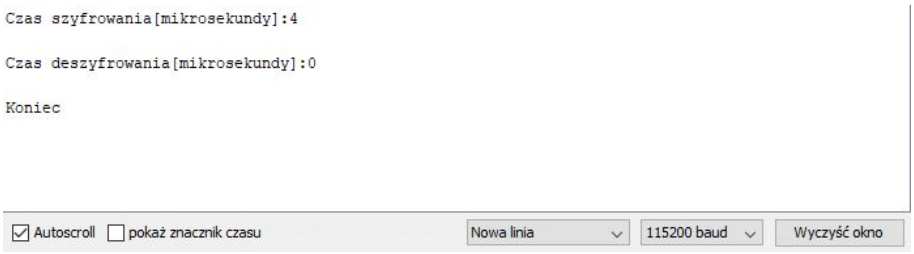
Rysunek 20: Czas szyfrowania i deszyfrowania dla prędkości transmisji danych 9600.

Na rysunku 21 przedstawiono obliczone czas potrzebny do szyfrowania i deszyfrowania algorytmem DES dla prędkości transmisji danych 19200 bitów na sekundę.



Rysunek 21: Czas szyfrowania i deszyfrowania dla prędkości transmisji danych 19200.

Na rysunku 22 przedstawiono obliczone czas potrzebny do szyfrowania i deszyfrowania algorytmem DES dla prędkości transmisji danych 115200 bitów na sekundę.



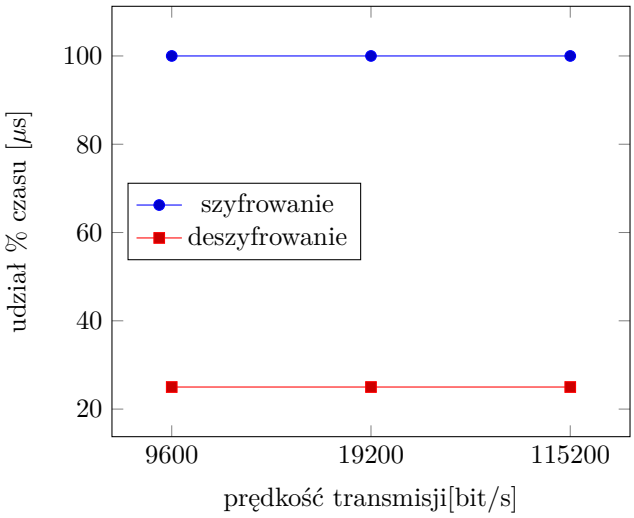
Rysunek 22: Czas szyfrowania i deszyfrowania dla prędkości transmisji danych 115200.

W tabeli 26 zobrazowano czas szyfrowania i deszyfrowania wiadomości 64 bitowej wyrażony w mikro sekundach. Szyfrowanie i deszyfrowanie wiadomości z wypisywaniem poszczególnych kroków na konsole konsumuje znacznie więcej czasu potrzebnego na poszczególne operacje. Możliwość podglądu poszczególnych kroków podczas procesu szyfrowania i deszyfrowania może być przydatna w celu wykrycia błędu występującego podczas wykonywania programu poprzez analizę otrzymanych wyników. Gdy zależy nam na wydajności oprogramowania szyfrującego należy unikać zbędnego wypisywania danych na konsolę, gdyż konsumuje to dużo czasu przez co wydajność oprogramowania maleje.

prędkość transmisji [bit/s]	9600	19200	115200
szyfrowanie z wypisywaniem na konsole [μs]	11337054	5667847	944187
deszyfrowanie z wypisywaniem na konsole [μs]	31995607	15995886	2664708
szyfrowanie [μs]	4	4	4
deszyfrowanie [μs]	1	1	0

Tabela 26: Czas przesyłu danych przy wykorzystaniu algorytmu DES.

Na rysunku 23 przedstawiono procentowy czas potrzebny do zaszyfrowania i odszyfrowania wiadomości 64 bitowej wykorzystując klucz 64 bitowy.



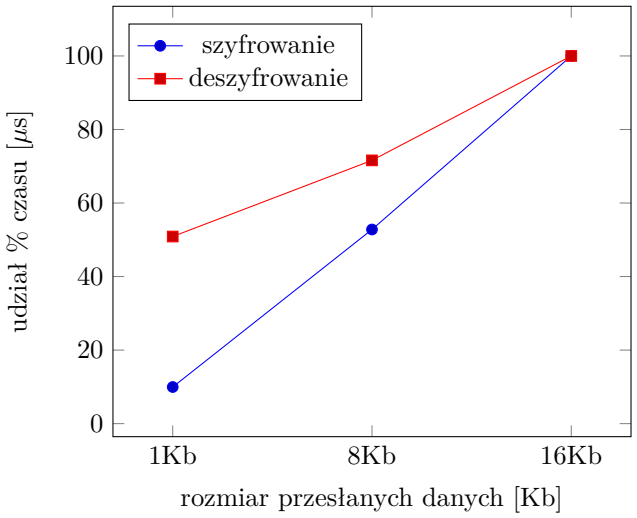
Rysunek 23: Procentowa prędkość przesyłu danych w zależności od prędkości transmisji.

Bit jest najmniejszą jednostką pamięci komputera oznaczoną literą b. Rozmiary pamięci komputera przedstawiono jako wielokrotności bitów i bajtów. Wielokrotność bita przedstawiono jako: 1Kb(kilobit) = 1024 bitów, 1Mb = 1048576 bitów, 1Gb = 1073741824 bitów itd. Wielokrotność bajta zaprezentowano jako 1KB(kilobajt) = 8192 bitów, 1MB = 8388608 bitów, 1GB = 8589934592 bitów itd. Dokonano pomiarów czasu szyfrowania i deszyfrowania danych o rozmiarach 1Kb, 8Kb i 16Kb dla trzech prędkości transmisji danych 9600, 19200 i 115200 bitów/s, które przedstawiono w tabeli 27.

prędkość transmisji [bit/s]	9600	19200	115200
szyfrowanie 1Kb[μs]	169	188	116
deszyfrowanie 1Kb[μs]	1718	113	101
szyfrowanie 8Kb[μs]	896	888	816
deszyfrowanie 8Kb[μs]	2417	813	801
szyfrowanie 16Kb[μs]	1697	1688	1616
deszyfrowanie 16Kb[μs]	3376	1613	1600

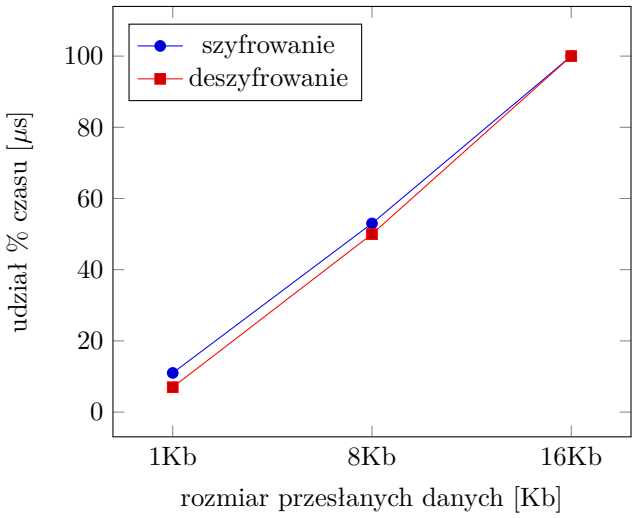
Tabela 27: Czas przesyłu danych o rozmiarze 1Kb, 8Kb i 16Kb.

Na rysunku 24 przedstawiono czas szyfrowania i deszyfrowania wiadomości o rozmiarze 1Kb, 8Kb i 16Kb dla prędkości transmisji danych 9600bitów/s.



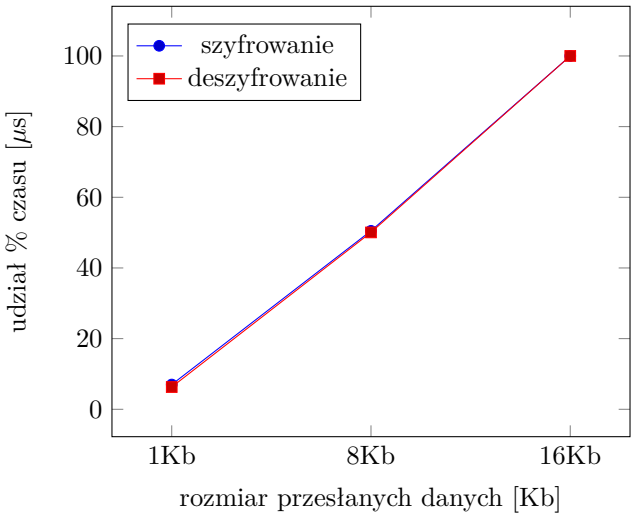
Rysunek 24: Czas szyfrowania i deszyfrowania w zależności od rozmiaru danych dla prędkości transmisji 9600.

Na rysunku 25 przedstawiono czas szyfrowania i deszyfrowania wiadomości o rozmiarze 1Kb, 8Kb i 16Kb dla prędkości transmisji danych 9600bitów/s.



Rysunek 25: Czas szyfrowania i deszyfrowania w zależności od rozmiaru danych dla prędkości transmisji 19200.

Na rysunku 26 przedstawiono czas szyfrowania i deszyfrowania wiadomości o rozmiarze 1Kb, 8Kb i 16Kb dla prędkości transmisji danych 9600bitów/s.



Rysunek 26: Czas szyfrowania i deszyfrowania w zależności od rozmiaru danych dla prędkości transmisji 115200.

Analizując rysunek 24, 25 i 26 spostrzeżono, że im większe prędkość transmisji danych tym bardziej czas szyfrowania jest zbliżony do czasu deszyfrowania danych. Dla najmniejszej prędkości przesyłu danych dostępnej w oprogramowaniu Arduino czas deszyfrowania wiadomości dla 1Kb i 8Kb jest większy niż czas potrzebny do zaszyfrowania tej wiadomości. Szyfrowanie DES wykorzystuje taki sam algorytm do deszyfrowania, w związku z czym czas szyfrowania i deszyfrowania powinien być zbliżony. Jak widać na rysunku 25 i 26 czas dla szyfrowania i deszyfrowania jest zbliżony, co świadczy o poprawnej implementacji algorytmu DES. Wydajność algorytmu można polepszyć poprzez wyeliminowanie z kodu zbędnych operacji.

5 Wnioski

Celem pracy był opis algorytmów kryptograficznych, budowy systemów wbudowanych, implementacja algorytmu DES na module ESP8266 oraz pomiar czasu szyfrowania i deszyfrowania w zależności od rozmiary danych. W prezentowanej przeze mnie pracy magisterskiej jako element wprowadzający do kryptografii opisano działanie szyfrów strumieniowych, blokowych, symetrycznych i asymetrycznych. W części praktycznej niniejszej pracy dokonano implementacji algorytmu DES, który jest szyfrem symetrycznym, co oznacza, że istnieje jeden klucz do szyfrowania i odszyfrowywania przesyłanej wiadomości. Mikrokontrolerem wykorzystany do implementacji algorytmu była płytka ESP8266 NodeMCU v2, na którą wgrano oprogramowanie za pośrednictwem kabla mikro USB i środowiska programistycznego Arduino. Pierwszym etapem było przedstawienie manualnej implementacji algorytmu DES dla pierwszej rundy w celu weryfikacji otrzymanych danych dla implementacji algorytmu na mikrokontrolerze. Stwierdzono poprawność implementacji algorytmu DES na ESP8266 po analizie wyników dla poszczególnych kroków rundy pierwszej wypisanych na monitorze portu szeregowego wraz z manualnymi obliczeniami. Otrzymane wyniki obliczeń manualnych są takie same jak obliczenia wykonane przy wykorzystaniu algorytmu na mikrokontrolerze ESP8266, co potwierdza poprawność implementacji.

Drugim etapem części praktycznej pracy był pomiar i porównanie czasów szyfrowania i deszyfrowania wiadomości w zależności od rozmiaru przesyłanych danych oraz prędkości transmisji programu. Pomiarów czas szyfrowania i deszyfrowania wiadomości dokonano dla danych o rozmiarze 1 kilobit, 8 kilobitów i 16 kilobitów oraz dla trzech prędkości transmisji: 9600, 19200 i 115200 bitów/s. Otrzymane dane pozwalają stwierdzić, że dla większej prędkości transmisji danych czas szyfrowania i deszyfrowania danych jest niższy, co prowadzi do lepszej wydajności algorytmu. Dla wyższej prędkości transmisji czas szyfrowania i deszyfrowania jest zbliżony, co jest pożądanym zjawiskiem gdyż proces deszyfrowania przebiega przy wykorzystaniu takich samych operacji jak proces szyfrowania, lecz w odwrotnej kolejności. Proces szyfrowania i deszyfrowania również zwiększa się wprost proporcjonalnie do rozmiaru szyfrowanych danych. Obliczając ilość potrzebnego czasu do zaszyfrowania danych o rozmiarze 1Gb według 1000000000*116/1000=160000000[μs],otrzymujemy 1min. 56sek. W celu zaszyfrowania pliku o rozmiarze 1Gb potrzebujemy ok 2min., co jest czasem zadowalającym dla użytkowników domowych. Implementację algorytmu DES na ESP8266 oraz pomiar wydajności szyfrowania danych zrealizowano zgodnie z celem przedstawionym na początku pracy.

Literatura

[1] J. Forshaw, *Attacking Network Protocols*, 08.2017, chapter 4

[2] , E. Cole, *Hackers Beware*, 08.2001, chapter Man-in-the-Middle Attack Against Key Exchange

[3] , Y. Li, Yu Sasaki, K. Sakiyama, *Security of Block Ciphers*, Wiley-IEEE Press, 04.2016, chapter 1

[4] , S. Bose, A. Kumar, *Cryptography and Network Security* Pearson Education India, 03.2016, chapter 5

[5] J. Wiley, *Information Security: Principles and Practice,2nd Edition*, M. Stamp, 05.2011, chapter 3

[6] Tingting CuiHuaifeng ChenLong WenMeiqin Wang, *Cryptography and Communications: Statistical integral attack on CAST-256 and IDEA*, January 2018, Volume 10

[7] W. Mao, *Modern Cryptography: Theory and Practice* Prentice Hall, 07.2003, chapter 8. Encryption — Asymmetric Techniques

[8] N. Hoffman, *Article in Cryptologia: A simplified IDEA algorithm*, Department of Mathematics, Northern Kentucky University, March 2007, str 143-151

[9] F.M. Groom, K. Groom and S.S. Jones *Network and Data Security for Non-Engineers*, Auerbach Publications, 08.2016, chapter 8

[10] C. Chebbi *Advanced Infrastructure Penetration Testing*, Packt Publishing, 02.2018

[11] D. Stoddard i T.M. Thomas, *Cover image for Network Security First-Step, Second Edition Network Security First-Step, Second Edition*, Cisco Press, 12.2011, chapter 6

[12] A. Lockhart, *Network Security Hacks*, O'Reilly Media Inc., 04.2004, chapter 6

[13] J. Savill, *The Complete Guide to Windows Server 2008*, Addison-Wesley Professional, 10.2008, chapter 8

[14] S. Malik, *Cover image for Network Security Principles and Practices Network Security Principles and Practices*, Cisco Press, 10.2002

[15] D. Lacamera *Embedded Systems Architecture*, Packt Publishing, 05.2018

[16] Espressif Systems, *ESP8266-DevKitC Getting Started Guide* Espressif Systems, 2018

Spis rysunków