YUKO®

Zakład Komputerowy YUKO

Witold Jureczko

44-151 Gliwice, ul. Daszyńskiego 560
internet: www.yuko.com.pl
e-mail: yuko@yuko.com.pl
tel./ fax : (+48) (32) 230-89-49

Regon: 271215331

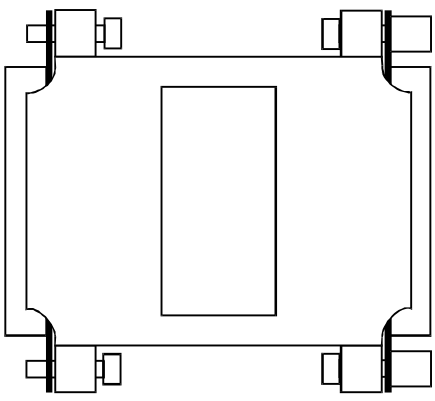
NIP: 631-010-66-35

telefony wewnętrzne, wybierane tonowo :
właściciel, sprawy techniczne - 31
dział handlowy, księgowość - 32
telefax - 40

PK-4

Sprzętowy klucz do zabezpieczania programów

Instrukcja dla programisty



Spis treści

1. Opis ogólny

Klucz PK-4 służy do zabezpieczania programów przed nieautoryzowanym użytkowaniem. System zabezpieczenia składa się z urządzenia sprzętowego (klucza) oraz zestawu modułów programowych włączanych do zabezpieczanego programu.

Klucz włączany jest do portu równoległego (drukarki) komputera. Urządzenie jest całkowicie "przezroczyste" przy standardowym wykorzystaniu portu do sterowania pracą drukarki. Drukarka może, ale nie musi być podłączona do klucza.

Podstawową właściwością klucza PK-4 jest możliwość zapisania i przechowywania w nim 64 bajtów informacji. Poprawne odczytanie zapisanych w kluczu danych jest wykorzystane do stwierdzenia legalności użycia zabezpieczanego programu. Informacja może być zapisywana w kluczu wielokrotnie i w normalnych warunkach nie ma możliwości jej przypadkowego skasowania lub zmiany, gdyż zapis może być dokonany tylko za pomocą specjalnego systemu (programu i urządzenia).

2. Zasada działania klucza PK-4

Zapis informacji w kluczu może być dokonany przez producenta klucza, zgodnie ze specyfikacją zamawiającego, lub przez użytkownika (producenta zabezpieczanego programu). YUKO gwarantuje niepowtarzalność poszczególnych egzemplarzy kluczy dostarczanych różnym użytkownikom, ale każdy użytkownik może zaprogramować lub zamówić dla siebie wiele identycznych kluczy.

Ogólnie, idea zabezpieczania programu za pomocą klucza PK-4 polega na włączeniu do niego na poziomie kodu źródłowego sekwencji powodującej odczytanie informacji zapisanej w kluczu i odpowiednią reakcję po sprawdzeniu jej poprawności. Komunikacja z kluczem możliwa jest tylko za pomocą procedur dostarczanych przez producenta wraz z kluczem.

Na dyskietce dostarczanej z kluczem znajduje się zestaw modułów programowych w postaci bibliotek typu .OBJ i .DLL umożliwiających wykorzystanie klucza praktycznie we wszystkich systemach programowania.

System zapisu (programowania) klucza składa się z urządzenia zwanego programatorem i specjalnego programu umożliwiającego zapis i testowanie klucza. Programator skonstruowany jest podobnie jak klucz PK-4 i włączany jest do portu równoległego komputera. Programowany klucz włączany jest do programatora i dopiero wtedy możliwe jest jego zaprogramowanie.

Każdy egzemplarz programatora zawiera niepowtarzalny kod gwarantujący, że zaprogramowane poprzez niego klucze nie mogą być identyczne z zaprogramowanymi w innych programatorach.

System programowania kluczy PK-4 dostarczany jest oddzielnie. Użytkownik może zakupić "czyste" klucze i samodzielnie je programować, lub zamówić klucze w pełni zaprogramowane, zgodnie ze swoimi wymaganiami.

Z punktu widzenia użytkownika, informacja zapisana w kluczu składa się z dwóch części: identyfikatora klucza i danych użytkowych. Identyfikator jest to ciąg 14 bajtów, który służy do uaktywnienia klucza i rozróżnienia poszczególnych kluczy, w przypadku, gdy do portu włączono więcej niż jeden klucz PK-4. W przypadku programowania klucza przez użytkownika, pierwsze dwa bajty identyfikatora są zdefiniowane przez używany programator, a pozostałe określone przez użytkownika. Dane użytkowe jest to ciąg 64 dowolnych bajtów określonych w fazie programowania i używanych w procedurach zabezpieczających program. **Dane użytkowe są poprawnie odczytane tylko po uaktywnieniu klucza właściwym identyfikatorem.**

Zabezpieczenie programu polega na wywołaniu z biblioteki odpowiedniej procedury z parametrem będącym identyfikatorem klucza i sprawdzeniu odpowiedzi, którą w przypadku gdy klucz jest zainstalowany w jednym z portów komputera, jest ciąg wcześniej zapisanych danych użytkowych. Dodatkowy parametr wywołania procedury używany do szyfrowania powoduje, że pomiędzy programem a procedurą obsługi klucza dane nie są przesyłane w postaci jawnej.

W przypadku, gdy w żadnym z portów równoległych komputera nie jest zainstalowany właściwy klucz, przesłane dane będą różne od oczekiwanych.

Dane zapisane w kluczu są dowolnym ciągiem 64 bajtów. W szczególnym przypadku może to być np. ciąg instrukcji programowych w kodzie maszynowym.

3. Interfejs programowy do klucza PK-4

Na dyskietce dostarczonej wraz z kluczem znajdują się moduły biblioteczne umożliwiające wykorzystanie klucza w zabezpieczanym programie oraz program pozwalający na zapis informacji do klucza za pomocą programatora oraz testowanie zaprogramowanych kluczy. Szczegółowy opis użycia bibliotek oraz przykładowe programy podane są w następnych punktach indywidualnie dla każdego systemu programowania. Opis systemu do programowania i testowania kluczy podano w oddzielnym rozdziale. Poniżej przedstawiono ogólne zasady wykorzystania procedury KEYPK4 służącej do komunikacji z kluczem.

W każdej z dostarczonych bibliotek dostępna jest jedna procedura o nazwie **KEYPK4**, która jest interfejsem pomiędzy programem a kluczem. Procedura ta ma trzy parametry wywołania:

KEYPK4 (KEYINF, KEYID, LRC)

- **KEYINF**: wskaźnik do obszaru pamięci, do którego procedura zwraca zaszyfrowaną informację odczytaną z klucza,
- **KEYID**: wskaźnik do obszaru pamięci, w którym podany jest identyfikator klucza,
- **LRC**: wartość 16-sto bitowego słowa szyfrującego informację odczytaną z klucza.

Uwaga: Procedura KEYPK4 przygotowana jest do przekazywania parametrów w trybie stosowanym w języku Turbo Pascal tzn. przed jej wywołaniem na stosie należy umieścić parametry w kolejności ich występowania w powyższej definicji, tzn. najpierw **KEYINF**, następnie **KEYID** i na końcu **LRC**.

Do obszaru pamięci wskazanego przez **KEYINF** procedura zapisze informację z klucza.

W obszarze pamięci wskazanym przez **KEYID** przed wywołaniem powinien być zapisany czternastobajtowy identyfikator klucza w postaci łańcucha typu "pascalowskiego" tzn. z jednobajtowym licznikiem poprzedzającym identyfikator. Wartość tego licznika jest stała i powinna wynosić 14.

Wartość **LRC** jest dowolną liczbą 16-sto bitową, która w powiązaniu z inną, stałą dla klucza liczbą tworzy kod szyfrujący informację odczytaną z klucza.

Po wykonaniu procedury **KEYPK4** w obszarze pamięci wskazanym przez **KEYINF** zapisane jest kolejno:
- wartość 64,
- 64 bajty zaszyfrowanej informacji z klucza,

Tak więc obszar wskazany przez **KEYINF** powinien mieć rozmiar 65 bajtów. W przypadku braku klucza o podanym identyfikatorze procedura **KEYPK4** zwraca przypadkowy ciąg bajtów.

Informacja odczytana z klucza przekazana jest w **KEYINF** w postaci zaszyfrowanej. Ma to na celu utrudnienie neutralizacji zabezpieczenia programu. Po wywołaniu procedury KEYPK4 otrzymany ciąg bajtów trzeba odszyfrować, aby otrzymać oryginalną informację zapisaną w kluczu.

Do odszyfrowania wykorzystywana jest 32-bitowa liczba, która składa się z dwóch 16-sto bitowych składników LRC i CRC. CRC jest stałą wartością (parametrem klucza) wynikającą z identyfikatora klucza i otrzymywana jest podczas zapisu identyfikatora, w fazie programowania klucza. Z założenia wartość CRC jest znana tylko użytkownikowi klucza tzn. programiście zabezpieczającemu program. LRC jest parametrem wywołania procedury KEYPK4. Zaleca się, aby wartość LRC była generowana przypadkowo, przy każdym wywołaniu KEYPK4 inna. W ten sposób zagwarantowane jest, że otrzymana odpowiedź będzie również przy każdym wywołaniu procedury inna. Uniemożliwi to neutralizację zabezpieczenia przez "podstawienie" za wywołanie procedury KEYPK4 innej, ponieważ zarówno do szyfrowania jak i deszyfrowania informacji z klucza konieczna jest znajomość obu składników kodu szyfrującego tzn. LRC i CRC. Z tego też powodu zrezygnowano z zamieszczenia procedury deszyfrującej w bibliotece, pozostawiając napisanie jej użytkownikowi klucza.

Wartości LRC i CRC, traktowane jako słowa 16-sto bitowe, tworzą 32-bitową kombinację, w której poszczególne bity oznaczone są jako B31..B0, przy czym LRC stanowi bardziej znaczącą część tzn. bity B31..B16. W celu otrzymania oryginalnego ciągu zapisanego w kluczu, przeprowadzić następujące operacje:

Założyć wartości początkowe X= 0, CRC właściwe dla danego klucza, LRC równe parametrowi ostatniego wywołania procedury KEYPK4.
Dla każdego bajtu otrzymanego w wyniku wywołania procedury KEYPK4 wykonać następującą sekwencję operacji:

1 - sprawdzić parzystość ilości jedynek na pozycjach B31, B18, B10, B4 i B1 LRC,CRC;
2 - dla parzystej ilości jedynek X = X, dla nieparzystej X = X XOR 1. X może przyjmować wartości 0 i 1;
3 - przesunąć 32-dwu bitową wartość LRC,CRC o jedno miejsce w lewo;
4 - wpisać X na pozycję B0 tak otrzymanej wartości LRC,CRC;
5 - najmniej znaczący bajt nowej, 32-bitowej wartości LRC,CRC użyć w operacji różnicy symetrycznej (XOR) z bajtem otrzymanym w wyniku wywołania procedury KEYPK4. Otrzymany wynik jest równy oryginalnej wartości zapisanej w kluczu.

Powtarzając 1..4 dla każdego bajtu otrzymuje się oryginalny ciąg zapisany w kluczu. Należy zwrócić uwagę, że po każdym wywołaniu KEYPK4 dla pierwszego deszyfrowanego bajtu wykorzystuje się początkowe wartości LRC, CRC i X, a dla kolejnych bajtów

zmodyfikowane w wyniku działania algorytmu deszyfrującego.

Do deszyfracji można użyć przykładowej procedury zamieszczonej na dyskietce, lub utworzyć własną bazując na opisanym algorytmie.

Procedura KeyPK4 sprawdza obecności klucza w wszystkich dostępnych portach komputera i nie jest istotne w którym porcie klucz jest zainstalowany.

W poniższym programie napisanym w języku Turbo Pascal zademonstrowano przykład użycia procedury KEYPK4. Zdefiniowana tam procedura CodeStr realizuje algorytm deszyfracji opisany wyżej. W tym przykładzie identyfikator klucza i ciąg odczytanych danych traktowane są jako obiekt typu *String*, co znacznie upraszcza operacje. Parametry procedury KEYPK4 są tak zdefiniowane, że takie podstawienie jest możliwe. Należy zwrócić uwagę, że obiekty typu String mogą zawierać znaki z przedziału [Chr(0)..Chr(255)], czyli praktycznie elementy typu Byte.

```
Program PK4Test;

Const
 KeyIdent = 'Klucz testowy '; {14 bajtowy identyfikator klucza}
 CRC = 46598;           {Parametr  tego  klucza  otrzymany  w  czasie
                              programowania; połowa kodu szyfrującego}
Var InfString : String;

Procedure KeyPK4(Var KeyInf: String; KeyId: String; CRC : Word); Far;External;
                              {KEYPK4 może być tutaj tak zdefiniowana,
                              uwzględniając wcześniejszy opis}
{$L PK4.OBJ}                  {Dołączenie KEYPK4 z biblioteki PK4.OBJ}

Procedure CodeStr (Var S:String; LRC,CRC: Word);

{Deszyfrowanie łańcucha S 32-bitowym kodem szyfrującym LRC, CRC
zgodnie z opisanym algorytmem.
Wynik w S. Oryginalne wartości LRC i CRC nie zostają zmienione,
ponieważ te parametry są przekazywane przez wartość!}

Var I,X     : Byte;
Begin
 X := 0;
 For I := 1 To Length(S) Do
     Begin                                  {Badanie parzystości}
      If LRC And $8000 <> 0 Then Inc (X);   {B31}
      If LRC And $0004 <> 0 Then Inc (X);   {B18}
      IF CRC And $0400 <> 0 Then Inc (X);   {B10}
      If CRC And $0010 <> 0 Then Inc (X);   {B4}
      If CRC And $0002 <> 0 Then Inc (X);   {B1}
      X := X And $01;                       {Wynik badania parzystości}
      LRC := LRC SHL 1;                     {Przesunięcie części LRC w lewo}
      If CRC And $8000 <> 0 Then LRC := LRC Or 1;
      CRC := (CRC SHL 1) Or X;              {Przesunięcie części CRC w lewo
                                            i wpisanie do B0}
      S[I] := Chr (Ord(S[I]) XOR Lo(CRC)); {Odszyfrowany I-ty znak}
     End;
End;

Begin
 Writeln ('Test klucza: ', KeyIdent);
 KeyPK4(InfString, KeyIdent, 12345);
 CodeStr (InfString, 12345,CRC);
 Writeln ('1: ',InfString);
 {I powtórna próba odczytu, inna wartość LRC, powinien być ten sam wynik po
 CodeStr,jeżeli klucz o identyfikatorze KeyIdent jest zainstalowany}
 KeyPK4 (InfString,KeyIdent,45321);
 CodeStr (InfString,45321,CRC);
 Writeln ('2: ',InfString);
End.
```

Przykłady dla innych systemów programowania umieszczone są na dyskietce dostarczanej z kluczem.

4. Implementacja zabezpieczeń w programach

4.1. Asembler

Procedura KEYPK4 z biblioteki PK4.OBJ dostępna jest w programie napisanym w asemblerze np. w następujący sposób:

```
stos_pr SEGMENT PARA STACK 'STOS'          ;Rezerwacja stosu
        db 12 dup ( 'Yuko 1995-03-15 ' )
        dw 0
stos_pr ENDS

CODE    SEGMENT WORD PUBLIC 'CODE'

KeyPK4 PROTO far pascal inf:DWord, id:DWord, lrc:Word

KeyId                          ;Identyfikator klucza
          db 14                ;Licznik = 14
;             12345678901234
          db 'Klucz testowy ' ;Spacja na końcu jest istotna!
InfStr    db 65 dup ( ? )      ;Obszar ma informację z klucza

start: ASSUME cs:CODE, ds:CODE, es:CODE, ss:stos_pr
       push cs
       pop  ds
       push ds                 ;Przekazanie parametrów na stos
       mov  ax, offset InfStr ;Obszar na informację
       push ax
       push ds
       mov  ax, offset KeyId  ;Identyfikator
       push ax
       mov  ax, 12345          ;LRC
       push ax
       call far ptr KeyPK4
CODE   ENDS
       END start
```

Po wykonaniu tego programu w obszarze rozpoczynającym się od **InfStr** umieszczona będzie zaszyfrowana informacja z klucza o identyfikatorze zapisanym od **KeyId**. Pierwszy bajt w tych obszarach ma wartość licznika, odpowiednio: 64 i 14.

4.2. C

Program wyprowadzający na ekran informację z klucza o identyfikatorze "Klucz testowy " napisany w języku Microsoft C może być np. następujący.

```
/*    ====================================================================
      =                Program TPK4    ( Microsoft C )                   =
      =                Zakład Komputerowy YUKO, 1995                     =
      =          Przykład użycia funkcji  KeyPK4 z PK4.OBJ               =
      ====================================================================
*/
#include <stdio.h>

/*Poniższa deklaracja KeyPK4 uwzględnia sposób przekazywania parametrów*/
extern void far _pascal
       KeyPK4(char *InfStr, char *KeyName, unsigned int crc );

char      _pascal KeyId[ 14+2 ];   /*Nazwa klucza*/
char      _pascal InfStr[ 66 ];

/*=========================================================================
 Procedura deszyfracji zakodowanej informacji odczytanej z klucza przez KeyPK4
 Jest to odpowiednik procedury opisanej w dokumentacji
 =========================================================================
*/
void CodeStr( char *s, unsigned int LRC, unsigned int CRC) {
  int i, x;
  x = 0;
  for ( i = 1; i <= s[ 0 ]; i++ ) {
    if ((LRC & 0x8000) != 0) x++;
    if ((LRC & 0x0004) != 0) x++;
    if ((CRC & 0x0400) != 0) x++;
    if ((CRC & 0x0010) != 0) x++;
    if ((CRC & 0x0002) != 0) x++;
    x &= 0x01;
    LRC = LRC << 1;
    if ((CRC & 0x8000) != 0) LRC++;
    CRC = (CRC << 1) | x;
    s[i] = s[ i ] ^ (char)CRC;
  }
}
void main( void ) {

 strcpy( KeyId, " Klucz testowy " );
 KeyId[ 0 ] = 14;
 printf("\n Test klucza PK4 ");
 printf("<%s>\n",KeyId+1);
 printf(" ____________________________________________________________________ \n");
 KeyPK4( InfStr, KeyId, 12345 );                /*LRC=12345, dowolne*/
 CodeStr( InfStr, 12345, (unsigned int)46598 ); /*LRC takie same */
 InfStr[ 65 ] = 0;                              /*Znacznik końca lancucha*/
 printf("\n Informacja odczytana z klucza: \n");
 printf("<12345678901234567890123456789012345678901234567890123456789012345678901234>\n");
 /* Poniższy wydruk będzie poprawny, pod warunkiem, ze w informacji z klucza
    nie będzie bajtów o wartości < 32 */
 printf("<%s>\n",InfStr+1);
}
```

4.3. Turbo Pascal

Przykład programu napisanego w języku Turbo Pascal wyprowadzającego na ekran treść informacji zapisanej w kluczu o identyfikatorze podanym w **KeyId**.

```
Program TPK4;  {Należy kompilować bez opcji kompilatora Strict var-strings
     ================================================================
     =             Program TPK4    ( Turbo Pascal 7.0 )              =
     =          Przykład użycia funkcji  KeyPK4 z PK4.OBJ            =
     ================================================================
}
Const
  KeyId = 'Klucz testowy ';        {Identyfikator klucza}
  CRC = 46598;                     {Parametr CRC dla tego klucza}
Var
  Inf : String [64];               {Informacja z klucza}

Procedure KeyPK4(Var KInf: String; KeyId: String; LRC : Word ); Far;External;
{$L PK4.OBJ}               {Dołączenie KEYPK4 z biblioteki PK4.OBJ}

Procedure CodeStr (Var S:String; LRC,CRC: Word);

{Deszyfrowanie łańcucha S 32-bitowym kodem szyfrującym LRC, CRC
zgodnie z opisanym algorytmem.
Wynik w S. Oryginalne wartości LRC i CRC nie zostają zmienione,
ponieważ te parametry są przekazywane przez wartość!}

Var I,X     : Byte;
Begin
 X := 0;
 For I := 1 To Length(S) Do
     Begin                                    {Badanie parzystości}
      If LRC And $8000 <> 0 Then Inc (X);     {B31}
      If LRC And $0004 <> 0 Then Inc (X);     {B18}
      IF CRC And $0400 <> 0 Then Inc (X);     {B10}
      If CRC And $0010 <> 0 Then Inc (X);     {B4}
      If CRC And $0002 <> 0 Then Inc (X);     {B1}
      X := X And $01;                         {Wynik badania parzystości}
      LRC := LRC SHL 1;                       {Przesunięcie części LRC w lewo}
      If CRC And $8000 <> 0 Then LRC := LRC Or 1;
      CRC := (CRC SHL 1) Or X;                {Przesunięcie części CRC w lewo
                                               i wpisanie do B0}
      S[I] := Chr (Ord(S[I]) XOR Lo(CRC)); {Oryginalny znak}
     End;
End;

Function KeyInf: String;

Var LRC : Word;

Begin
 LRC := Random ($FFFF);          {Przypadkowa wartość LRC}
 KeyPk4 (Inf,KeyId,LRC);         {W Inf  zaszyfrowana informacja z klucza.}
 CodeStr (Inf,LRC,CRC);          {Odszyfrowanie}
 KeyInf := Inf;
End;

Begin
 Randomize;
 Writeln ('Test klucza: ',KeyId);
 Writeln ('Odczytano:');
 Writeln (KeyInf);
End.
```

4.4. Windows 3.x

Aplikacje przeznaczone do pracy pod kontrolą systemu Windows 3.xx mogą być zabezpieczane kluczem PK4 za pomocą funkcji KeyPK4 z pliku PK4.DLL (indeks=1) dostępnego na dyskietce dystrybucyjnej. Jej działanie oraz funkcje parametrów wywołania są zgodne z wcześniejszym opisem procedury KeyPK4 z biblioteki PK4.OBJ:

KEYPK4 (KEYINF, KEYID, LRC)

KEYINF: wskaźnik do obszaru pamięci, do którego procedura zwraca zaszyfrowaną informację odczytaną z klucza, rozmiar: 65 bajtów. Po wykonaniu funkcji w obszarze zapisane jest kolejno:
wartość 64,
64 bajty zaszyfrowanej informacji odczytanej z klucza.

KEYID: wskaźnik do obszaru pamięci, w którym podany jest identyfikator klucza. Przed wywołaniem funkcji należy w tym obszarze w kolejnych bajtach umieścić:
wartość 14,
14 bajtów będących identyfikatorem klucza.

LRC: Dowolna wartość 16-sto bitowa słowa kodującego informację odczytaną z klucza.

Otrzymana informacja z klucza jest zaszyfrowana w sposób identyczny z opisanym wcześniej.

W poniższym przykładzie przedstawiono aplikację napisaną w języku Borland Pascal 7, która wyprowadza na ekran okienko typu MessageBox zawierające identyfikator klucza i treść odczytanej z niego informacji (o ile taki klucz jest zainstalowany). Dla uproszczenia przyjęto, że zarówno identyfikator, jak i informacja są ciągami znaków o kodach mniejszych niż 32. To ograniczenie narzucone jest jedynie przez sposób działania funkcji MessageBox, i nie występuje w przypadku, gdy te ciągi nie są wyprowadzane na ekran.

```
Program DemoPk4;

{     ================================================================
      =             Program DemoPK4    ( Borland Pascal 7.0 )          =
      =                Zakład Komputerowy YUKO, 1995                   =
      =          Przykład użycia funkcji  KeyPK4 z PK4.DLL             =
      ================================================================
}

{$X+} {Extended syntax}
{$V-} {Bez Strict var checing}

Uses winAPI,Strings;
Const
      KeyId ='Klucz testowy ';     {Identyfikator klucza}
      CRC= 46598;                  {Wartsc CRC dla tego klucza}
Var
      InfStr : String[65];         {Obszar na informację z klucza, ze względu
                                    na standard  API musi być typu PChar,
                                    dlatego jest 65, patrz dalej}
      WS : Array [0..40] Of Char; {Obszar pomocniczy}
      PIS : Pchar;
      LRC : Word;

Procedure KeyPK4 (Var IS: String; KS:String;Code:Word); FAR; External 'pk4';
{Deklaracja KeyPK4 z PK4.DLL}

Procedure Decode (Var Inf:String;LRC,CRC:Word);
{Dekodowanie S algorymem opisanym w dokumentacji }
Var I,X :Byte;
    R,K : Word;
Begin
 R := LRC;
 K := CRC;
 X := 0;
 For I := 1 To Length (Inf) Do
      Begin
       If R And $8000 <> 0 Then Inc (X); {B31}
       If R And $0004 <> 0 Then Inc (X); {B18}
       IF K And $0400 <> 0 Then Inc (X); {B10}
       IF K And $0010 <> 0 Then Inc (X); {B04}
```

```
      If K And $0002 <> 0 Then Inc (X); {B01}
      X := X And $01;
      R := R SHL 1;
      If K And $8000 <> 0 Then Inc (R);
      K := (K SHL 1) Or X;
      Inf[I] := Chr (Ord(Inf[I]) XOR Lo(K));
     End;
End;

Begin
 Randomize;
 LRC := Random ($FFFF);                    {Przypadkowa wartosc LRC}
 KeyPK4 (InfStr,KeyId,LRC);                {Odczyt informacji z klucza}
 Decode (InfStr,LRC,CRC);                  {Deszyfracja, LRC jak w KeyPK4}

 {Przygotowanie parametrów dla funkcji MessageBox}
 PIS := @InfStr[1];                        {Zamiana InfStr na Pchar}
 InfStr[65] := Chr(0);                     {Koniec tekstu typu PChar. Zakładamy,
                                            że w InfStr nie występuje znak = Chr(0)}

 StrCopy (WS,'Informacja z klucza:   '); {Naglowek dla MessageBox}
 StrCat (WS,KeyId);
 MessageBox (0,PIS, WS, mb_Ok+mb_IconInformation);
End.
```

4.5. Windows 95

Dla aplikacji 32 bitowych, pracujących w środowisku Windows 95 należy użyć funkcji KEYPK4 z pliku WPK4_32.DLL. Plik ten dostarczany jest na dyskietce dystrybucyjnej, gdzie również są zamieszczone przykłady aplikacji dla Windows 95. Interfejs programowy do funkcji KEYPK4 z biblioteki WPK4_32.DLL jest identyczny jak dla wersji 16-to bitowej

5. Uwagi dotyczące zabezpieczania oprogramowania

Niewątpliwie najważniejszym etapem zabezpieczania programu za pomocą urządzenia takiego jak klucz PK4 jest właściwe zamaskowanie w programie procedur testujących obecność klucza. Wynika to z faktu, że zneutralizowanie tych procedur pozwala następnie na proste powielanie w dowolnych ilościach nielegalnych kopii odbezpieczonego programu. Natomiast ewentualne odtworzenie zabezpieczenia sprzętowego tzn. klucza, powoduje znacznie mniejsze straty producenta oprogramowania, ponieważ wymaga stosunkowo kosztownych zabiegów wyprodukowania klucza do każdej nielegalnej kopii programu. W większości przypadków proceder taki jest niopłacalny dla potencjalnych piratów.

W celu utrudnienia neutralizacji zabezpieczeń zalecane jest przestrzeganie następujących zasad:

- Test obecności klucza powinien wystąpić więcej niż jeden raz.

- Należy unikać prostych skoków według wartości znacznika wskazującego poprawność testu klucza. Skuteczniejszym rozwiązaniem w przypadku klucza PK4 jest np. użycie odczytanej informacji jako wartości początkowych istotnych zmiennych sterujących działaniem programu. Możliwe jest również wykorzystanie tej informacji jako fragmentu kodu programu. Aby ustrzec się w takich rozwiązaniach przed niebezpieczeństwem nieokreślonego działania programu w przypadku braku właściwego klucza, można dodatkowo sprawdzać poprawność odczytanej informacji np. przez włączenie do niej sumy kontrolnej.

- Wskazane jest uniemożliwienie poprawnego działania programu zachowanego po pomyślnym przebiegu testu klucza. Można to uzyskać np. przez cykliczne testowanie obecności klucza.
- Celowe jest wprowadzenie utrudnienia śledzenia działania programu pod kontrolą narzędzi typu debuger.

- Nie należy udostępniać dwóch kopii tego samego programu różniących się jedynie informacją zapisaną w kluczu. Celowe jest raczej zabezpieczenie różnych kopii tej samej wersji programu identycznymi kluczami.

- Należy generować różne wartości kodu szyfrującego LRC (np. tworzone przypadkowo) przy każdorazowym wywołaniu procedury KeyPK4.

Powyższe uwagi będą niewątpliwie jedynie inspiracją dla programisty i zostaną uzupełnione o własne doświadczenia, przemyślenia i inwencję twórcy zabezpieczającego swój produkt i stanowiące jego tajemnicę zawodową.

W przypadku sprzedaży jednemu użytkownikowi kilku kopii tego samego programu wszystkie one powinny być zabezpieczone tym samym kluczem. Ułatwia to zarówno użytkowanie programu jak i dystrybucję jego nowych wersji.

Należy zwrócić uwagę, że informacja zapisana w kluczu może być użyta w szerszym znaczeniu np. do określenia jakie funkcje zabezpieczonego programu są udostępnione użytkownikowi.

Ze swej strony YUKO gwarantuje niepowtarzalność identyfikatorów kluczy chyba, że użytkownik zamówi wiele kluczy o tych samych identyfikatorach. Identyfikatory dostarczonych użytkownikowi kluczy

oraz inne szczegóły, w tym niniejszą dokumentację, YUKO traktuje jako informacje poufne zobowiązując się do niudostępniania ich osobom trzecim i oczekując równocześnie takiego samego podejścia ze strony użytkowników. Leży to w dobrze pojętym interesie obu stron.

6. Programowanie i testowanie kluczy PK4

6.1. Program TESTPK4

Na dyskietce dostarczonej wraz z kluczem PK4 znajduje się program o nazwie TESTPK4.EXE, który służy do programowania i testowania kluczy PK4. Program jest sterowany za pomocą menu zawierającego następujące pozycje:

- <Pliki>: Operacje związane z bazą danych i zakończenie działania programu
- <Port>: Określenie używanego przez program portu równoległego,
- <Programowanie klucza>: Zapisanie do pamięci klucza identyfikatora i/lub informacji użytkowej oraz aktualizacja bazy danych,
- <Testy klucza i portu>: Testowanie poprawności działania klucza.

W programie TESTPK4 konieczne jest określenie używanego portu równoległego. Standardowo przyjęty jest LPT1, ale w przypadku wykorzystania innego portu należy wybrać właściwy w menu <Port>.

> ✯ **Uwaga** ✯
>
> Programator i klucz mogą być włączane **tylko do portu równoległego** (złącze 25-cio stykowe żeńskie zainstalowane na tylnej ściance komputera). Włączenie klucza lub programatora do portu szeregowego może spowodować trwałe uszkodzenie tych urządzeń.
>
> W czasie działania programu można bez obawy uszkodzenia dołączać do portu równoległego komputera programator i klucze bez konieczności wyłączania zasilania komputera. Należy jednakże zachować ostrożność przy dołączaniu kabla drukarki. Wymagane jest, by drukarka i komputer zasilane były z tego samego punktu przyłączeniowego wyposażonego w poprawnie zainstalowany obwód zera energetycznego.

Aktualnie dostępne funkcje programu zależą od tego, czy w używanym porcie jest zainstalowany programator. Przy braku programatora nie jest dostępna pozycja menu <Programowanie klucza>. Sprawdzenie obecności programatora odbywa się na po uruchomieniu programu oraz przy każdej zmianie używanego portu równoległego.

U dołu ekranu wyświetlane jest okno, w którym podane są aktualne informacje o najważniejszych parametrach programu: identyfikatorze klucza, używanym porcie i identyfikatorze programatora, itp.

W programie zaimplementowana jest prosta baza danych umożliwiająca rejestrację zaprogramowanych kluczy. W bazie danych zanotowane są identyfikatory kluczy oraz daty ich zaprogramowania. Przewidziane jest również pole na krótki komentarz. Jeżeli baza danych jest używana, informacje o kluczach przechowywane są w pliku o nazwie PK4.DBY w bieżącej kartotece.

6.2. Programowanie kluczy

Operacja programowania polega na wpisaniu do pamięci klucza nowego identyfikatora i/lub informacji użytkowej. Klucz PK4 może być wielokrotnie programowany. Nie można programować dwóch lub więcej kluczy równocześnie. Po zaprogramowaniu klucza wskazane jest odnotowanie tego w bazie danych.

Przed przystąpieniem do programowania należy w jednym z portów równoległych komputera zainstalować programator. Po uruchomieniu programu TESTPK4 w oknie informacyjnym w dolnej części ekranu wyświetlany jest między innymi dwuznakowy identyfikator programatora, tzn. pierwsze dwa bajty identyfikatora klucza wymuszane przez ten programator. W przypadku, gdy wyświetlana jest informacja o braku programatora należy poprzez menu <Port> zlokalizować port, w którym zainstalowano programator. Po każdej zmianie portu program sprawdza, czy programator jest w nim zainstalowany i odpowiednio modyfikuje informacje wyświetlane w oknie. W przypadku braku programatora menu <Programowanie klucza> będzie niedostępne.

Programowany klucz należy podłączyć do programatora. Jeżeli przewiduje się wykorzystanie drukarki w czasie programowania, np. do testowania klucza, kabel drukarki należy podłączyć do klucza.

Uwaga: Drukarka przyłączona bezpośrednio do programatora nie będzie działała poprawnie!

Przed programowaniem klucza należy zdefiniować jego identyfikator i informację zapisywaną do klucza, jeśli taka operacja jest przewidziana. Oddzielne pozycje menu pozwalają na niezależny zapis do klucza identyfikatora i informacji lub łączne wykonanie obu tych operacji. Po poprawnym zaprogramowaniu identyfikatora na ekran wyprowadzona jest wartość składnika CRC kodu szyfrującego. Liczbę tą należy zanotować, gdyż jest ona niezbędna do deszyfracji informacji oczytanej z tego klucza w programie zabezpieczającym (patrz p.3 niniejszej dokumentacji). Wartość CRC jest jednakowa dla wszystkich egzemplarzy klucza o tych samych identyfikatorach.

Należy zaznaczyć, że zapis informacji użytkowej do klucza jest możliwy tylko po poprawnym określeniu jego identyfikatora.

Do określenia parametrów klucza tzn. identyfikatora i informacji używane jest okno edycyjne umożliwiające wpisanie tych parametrów jako łańcucha znaków lub wartości szesnastkowych. Przełączanie pomiędzy obiema sposobami edycji następuje za pomocą klawisza <Tab>. Klawiszem <Enter> zatwierdzane są wprowadzone zmiany, a klawiszem <Esc> anulowane. Odczytana w menu <Test klucza i portu> informacja zapisana w kluczu jest zapamiętana w buforze programu aż do jej zmiany lub zakończenia działania programu. W ten sposób można uniknąć każdorazowego jej wprowadzania.

Zaimplementowana w programie prosta baza danych umożliwia przechowywanie informacji o zaprogramowanych kluczach. Każdy rekord zapisany w bazie danych zawiera:

- numer seryjny klucza,
- identyfikator,
- wartośc kodu szyfrującego,
- kod użytkownika,
- data i czas ostatniego zapisu do klucza,
- informację użytkową zapisaną w kluczu.

Poszczególne rekordy umieszczone są w bazie danych w kolejności wzrastających numerów seryjnych kluczy. Numer seryjny zawiera do ośmiu cyfr jest przez producenta i uwidoczniony na obudowie. Jest on unikalny dla każdego wyprodukowanego egzemplarza. Zalecane jest używanie tego numeru w bazie danych. Numer seryjny nie jest pamiętany w pamięci klucza. Numery seryjne w bazie danych nie mogą się powtarzać. Odpowiednia pozycja menu pozwala na określenie numeru seryjnego klucza.

Z każdym zanotowanym w bazie danych kluczem związana jest szesnastoznakowa informacja o dowolnej treści, nazwana tutaj "kodem użytkownika", która może być użyta np. do identyfikacji użytkownika tego klucza. podobnie jak numer seryjny informacja ta występuje tylko w bazie danych i nie jest pamiętana w kluczu.

Zalecana jest aktualizacja bazy danych (pozycja menu <Aktualizacja bazy danych>
po każdym zapisaniu do klucza nowego identyfikatora lub informacji użytkowej. Zapis aktualnego stanu bazy danych umożliwia odpowiednia pozycja w menu <Pliki>. Dostępna jest również pozycja menu <Przegląd bazy danych>. W czasie przeglądania zawartości bazy danych możliwe jest wydrukowanie na drukarce treści wskazanego rekordu, usunięcie rekordu z bazy danych oraz wczytanie rekordu w celu np. wykorzystania zawartych w nim informacji do programowania innych kluczy.

Pozycja menu <Raport programowania> umożliwia wydrukowanie wszyskich informacji związanych z programowanym kluczem. Wydruk ten zawiera 8 linii tekstu i nie jest zakończony komendą wysunięcia strony, tak, że w przypadku użycia drukarki laserowej, aby uzyskać wydruk konieczne jest wykonanie odpowiedniej operacji z pulpitu drukarki.

W czasie programowania klucza kontrolowana jest poprawność wykonania tej operacji, ale niezależnie od tego można uaktywnić z poziomu menu głównego pozycję <Testy klucza i portu>, co pozwala na niezależne sprawdzenie poprawności działania klucza (patrz p. "Testowanie kluczy").

6.3. Testowanie kluczy

Klucz o dowolnym, znanym identyfikatorze może być przetestowany w programie TESTPK4 po uaktywnieniu menu <Testy klucza i portu>. Nie można testować kluczy o nieznanych identyfikatorach oraz nie ma możliwości odczytania identyfikatora z klucza.

Sprawdzenie poprawności działania zaprogramowanych kluczy może odbywać się w dwóch wariantach: z programatorem lub bez.

W przypadku zainstalowania programatora można testować jedynie klucze o identyfikatorach określonych w menu <Programowanie klucza>, natomiast przy braku programatora, w menu <Testy klucza i portu> dostępna jest pozycja pozwalająca na określenie dowolnego identyfikatora klucza. Tak więc obecność programatora ogranicza zakres identyfikatorów kluczy możliwych do przetestowania. Jeżeli istnieje konieczność sprawdzenia poprawności działania klucza o identyfikatorze innym niż to wynika z posiadanego programatora, klucz należy włączyć do portu równoległego komputera bez pośrednictwa programatora, ale wtedy nie będzie dostępne menu <Programowanie klucza>

Poszczególne pozycje menu <Testy klucza i portu> realizują:

<Identyfikator klucza>:
: Określenie identyfikatora testowanego klucza w sposób podany w p. "Programowanie kluczy". Pozycja ta nie występuje, jeżeli jest zainstalowany programator.

<Test klucza>:
: Sprawdzenie obecności klucza o podanym identyfikatorze w wybranym porcie oraz kontrola technicznej poprawności zapisanej informacji. Test przeprowadzany jest dwudziestokrotnie.

<Informacja z klucza>:
: Odczyt informacji użytkowej z klucza o podanym identyfikatorze i wyprowadzenie jej w postaci odszyfrowanej na ekran.

<Test portu i drukarki>:
: Kontrola wybranego portu równoległego oraz popraw-ności przesyłania informacji poprzez klucz do drukarki.

Pozycja <Informacja z klucza> może być wykorzystana do kopiowania informacji z zaprogramowanego wcześniej klucza do nowo programowanego.

Pozycja <Test portu i drukarki> testuje aktualnie używany port i poprawność działania drukarki podłączonej poprzez klucz. Do tego testu nie jest potrzebna znajomość identyfikatora klucza. Sprawdzenie poprawności działania podłączonej drukarki wymaga wizualnej kontroli otrzymanego wydruku. Na drukarkę wyprowadzany jest tekst zawierający:

- identyfikator klucza,
- 4 linie tekstu.

W pierwszej z tych linii znajdują się znaki o kodach z zakresu [32..95]. każda następna linia zawiera te same znaki ale z przesunięciem o jeden znak w każdej linii. Na końcu tekstu nie ma komendy "Wysuń stronę", tak, że w przypadku niektórych drukarek, np. laserowych w celu uzyskania wydruku należy spowodować wysunięcie strony za pomocą odpowiedniej operacji z pulpitu drukarki.

Do testowania kluczy w środowisku Windows 95 można użyć programu o nazwie **WTEST.EXE** dostarczanego na dyskietce dystrybucyjnej.