Protokol s koncovým šifrováním pro IEEE 802.15.4

Author: JAROMÍR BAČA

Advisor: Ing. ONDŘEJ KRAJSA, Ph.D.

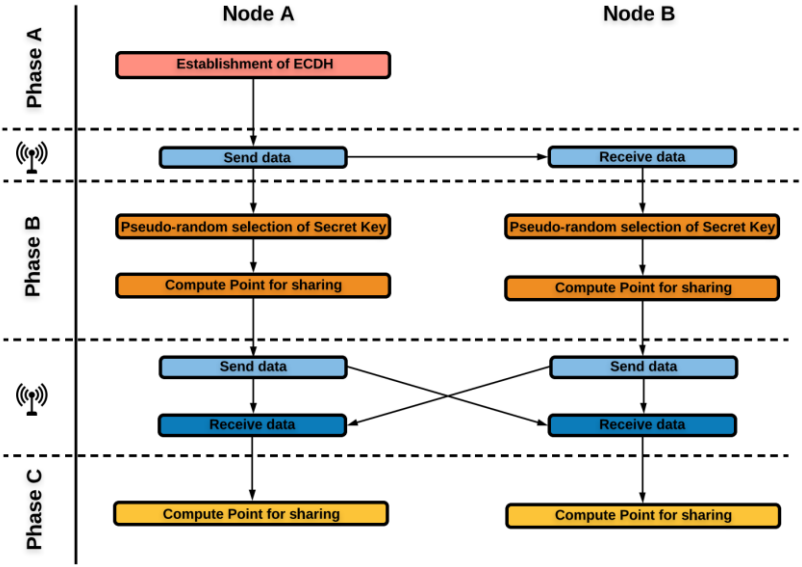
Brno, 3.9.2020

Úvod

- Metodika výměny klíčů, založeno na ECDH
- Atmel LightWeigh mesh (síťový protokol)
- Knihovna BigDigit (práce s velkými čísly)
- Vývojové desky s radio modulem deRF
- Projekt vytvořen v jazyce C

Metodika výměny klíčů

- Odesílatel, příjemce
- Inpirováno IKEv2
- Asym. krytptografie
- Eliptické křivky
- Tři fáze
- Dvě radio sekvence



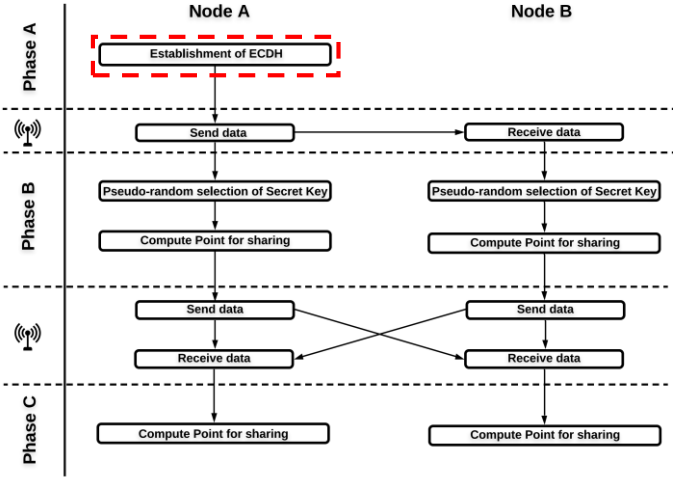
Řešení přepínání fází

- Řízení pomocí stavů
- Změna stavů na základě podmínek

```
/*- Types ----------------------
typedef enum AppState_t
{
    APP_STATE_Init,
    APP_STATE_FazeA,
    APP_STATE_FazeB,
    APP_STATE_FazeC,
    APP_STATE_END,
} AppState_t;
```

```
switch (appState)
{
case APP_STATE_Init:
{
    appInit();
    appState = APP_STATE_FazeA;

} break;

case APP_STATE_FazeA:
{
    ECDH_PHASE_A_BIGD(MOD, a_parameter, b_parameter, resultsA);
    appState = APP_STATE_FazeB;

} break;

case APP_STATE_FazeB:
{
    if(comaparsionA != 0){
        appState =  APP_STATE_FazeC;
        }
} break;

case APP_STATE_FazeC:

if(comaparsionB != 0){
    appState =  APP_STATE_END;
}

APP_STATE_END:
break;

default:
break;
}
```

Metodika výměny klíčů – Fáze A



Fáze A

Fáze A – ustanovení eliptických křivek

- Pseudonáhodný výběr hodnoty p (modulo) + kontrola pročíselnosti
- Pseudonáhodný výběr parametrů křivky a a b (asymptoty) + jejich kontrola
- Výpočet generátoru grupy (první bod) a řádu grupy
- Kontrola bodu, zda leží na křivce

$$y^2 \bmod p = (x^3 + ax + b) \bmod p$$

Výběr: p, a, b

Fáze A – Generování a kontrola prvočísla (modulo p)

- Cyklus **do-while** dokud není vybráno prvočíslo
- Pseudonáhodný výběr n-bitové čísla, příkaz **bdRandomSeed** (knihovna BigDigit)
- Příkaz **bdRabinMiller**, 10 opakování testů

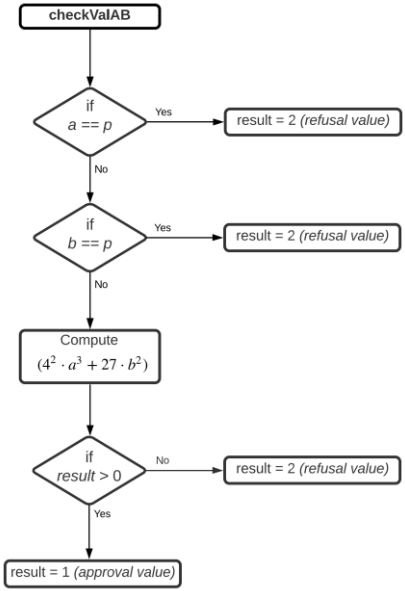
```
do
{
    // Selection of the modulo value from pseudo-random number
    bdRandomSeeded(premod, 512, (const unsigned char*)"", 0, RandomNumber);  // 512 bits key
    //bdModulo(mod, premod, cutoff);


    // Examination of modulus
    PrimeTest = bdRabinMiller(mod, 10);   // Rabin-Miller - 10 rounds iteration

}
while(PrimeTest == 0);   // This cycle runs until a value appears as 1 (prime)
```

Fáze A – Generování a kontrola asymptot

- Pseudo-random výběr asymptot ***a*** a ***b***
- Příkaz **bdRandomSeed**, n-bitové číslo
- Testovací aplikace **checkValAB**

```
// Selection of asymptos values from pseudo-random number
bdRandomSeeded(a,192, (const unsigned char*)"", 0, RandomNumber);
bdRandomSeeded(b, 192, (const unsigned char*)"", 0, RandomNumber);

checkValABBIGD(TESTvalue, mod, a, b);  // Application for test
```

$$(4 \cdot a^3 + 27 \cdot b^3) \mathrm{mod} p \neq 0$$



Fáze A – Generování a kontrola asymptot

- Použití příkazů knihovny BigDigit

```
bdPower(resultA, a, 3); // a^3
bdMultiply(resultB, AuxNumA,  resultA);  // 4 * a^3

bdPower(resultC, b, 2); // b^2
bdMultiply(resultD, AuxNumB, resultC);  // 27 * b^2

bdAdd(resultE, resultB, resultD);        // 4 * a^3 + 27 * a^3

bdModulo(resultF, resultE, mod);       // (4 * a^3 + 27 * a^3) mod p
```

$$(4 \cdot a^3 + 27 \cdot b^3) \mathrm{mod} p \neq 0$$

Fáze A – Výpočet generátoru grupy (první bod)

- Porovnávání Xové a Yové složky
- Výpočty jsou opět založeny na znalosti hdonty modulo a asymptot

y	y^2
0	0
±1	1
±2	4

$p = 5,$
$a = 1$
$\mathrm{b} = 4$

x	$x^3 + ax + b$	y^2	y	[X, Y], [X, Y]
0	4	4	±2	[0, 2], [0, 3]
1	1	1	±1	[1, 1], [1, 4]
2	4	4	±2	[2, 2], [2, 3]
3	4	4	±2	[3, 2], [3, 3]
4	-	-	-	-
∞				0

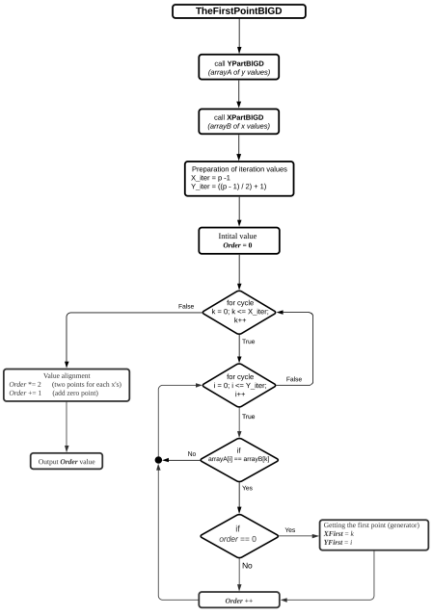
Fáze A – Výpočet generátoru grupy (první bod)

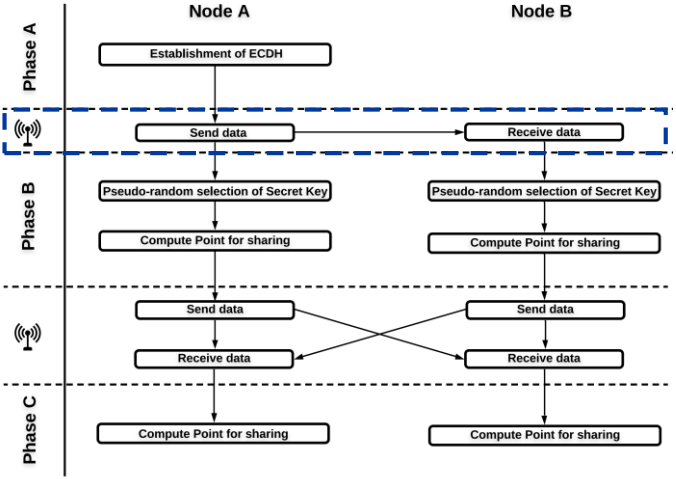
- Yová složka

y	y^2
0	0
±1	**1**
±2	4

- Xová složka

x	$x^3 + ax + b$
0	4
1	**1**
2	4
3	4
4	-
∞	





Odeslání a příjem dat A->B

Odeslání dat A->B

- Realizováno pomocí LWM protokolu
- Využívá knihovny **BigDigit**
- Obsah globální proměnné se přenese do pole
- Konfigurace payloadu
- Nastavení endpointů
- Nutnost mazat nepoužívané bloky (úspora paměti)

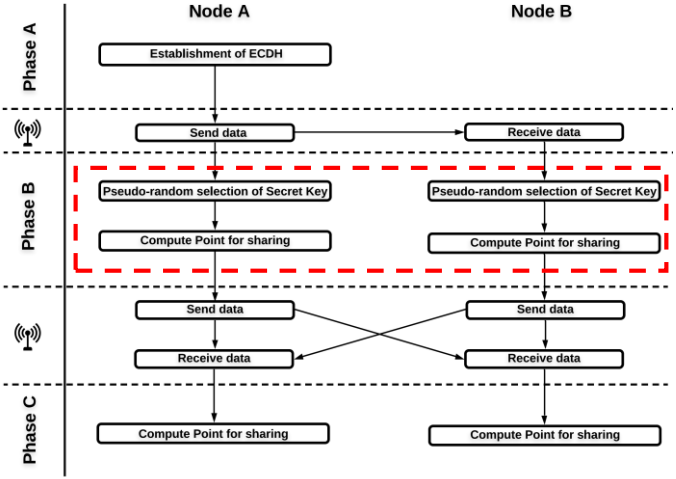
```
/*********************************************************************
 *********************************************************************
static void SendDataToBB(void)
{
    // varible for stored data as array
    BB data[6];

    for(uint8_t i = 0; i <= 5; i++){

        data[i] = bdNew();              // block activation
    }

    data[0] = MOD;          // MOD
    data[1] = a_parameter;  // A_parametr
    data[2] = b_parameter;  // b_parametr
    data[3] = resultsA[0];  // Xf
    data[4] = resultsA[1];  // Yf
    data[5] = resultsA[2];  // Order


    appDataReq.dstAddr = 0xFFFF;
    appDataReq.dstEndpoint = 2;
    appDataReq.srcEndpoint = 1;
    //appDataReq.options = NWK_OPT_ENABLE_SECURITY;
    appDataReq.data = data;
    appDataReq.size = 7;
    appDataReq.confirm = appDataConf;
    NWK_DataReq(&appDataReq);


    appDataReqBusy = true;


    for(uint8_t i = 0; i <= 5; i++){

        bdFree(&data[i]);           // block deactivation
    }
}
```

Příjem dat A->B

- Realizováno pomocí LWM protokolu
- Využívá knihovny **BigDigit**
- Přenesení obdržených dat do proměnných
- Spuštění **aplikace pro fázi B**
- Kontrola výstupu => změna stavu/fáze

```
196 static bool SendFromAToB_BB(NWK_DataInd_t *ind)  // Version BA
197 {
198
199     // Activate block
200     MOD = bdNew();
201     a_parameter = bdNew();
202     b_parameter = bdNew();
203     resultsBB[0] = bdNew();
204     resultsBB[1] = bdNew();
205     ZEROB = bdNew();
206
207     bdSetZero(ZEROB);
208
209     MOD = ind->data[0];
210     a_parameter = ind->data[1];
211     a_parameter = ind->data[2];
212
213     ECDH_PHASE_BB_BIGD(MOD, a_parameter, b_parameter, resultsBB);
214
215     comaparsionA = bdCompare(resultsBB[1], ZEROB);  // Variable filling control
216
217     // Deatcivate block
218     bdFree(&MOD);
219     bdFree(&a_parameter);
220     bdFree(&b_parameter);
221
222 }
223
```

Metodika výměny klíčů – Fáze A

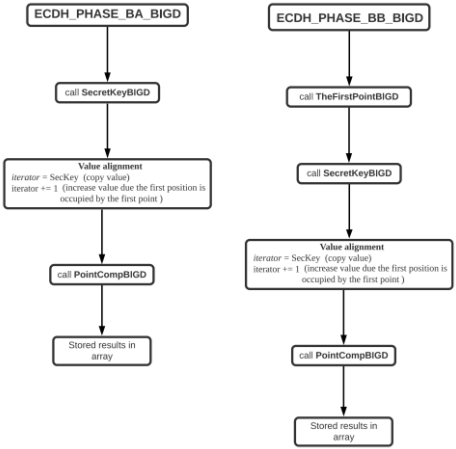


Fáze B

Fáze B – úvod

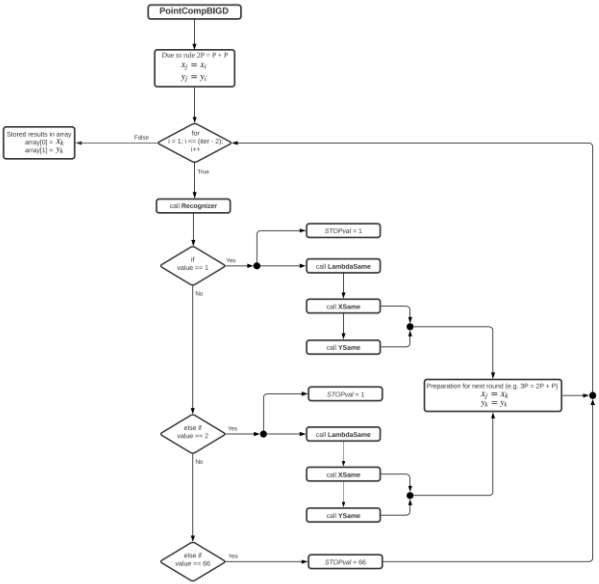
- Rozdělena na dvě subverze
- Příjemce si ze znalosti ***p***, ***a***, ***b*** si sám vypočítá generátor
- Pseudonáhodný výběr hodnoty tajného klíče, pomocí příkazy **bdRandomSeed**
- Výpočet bodu pro sdílení

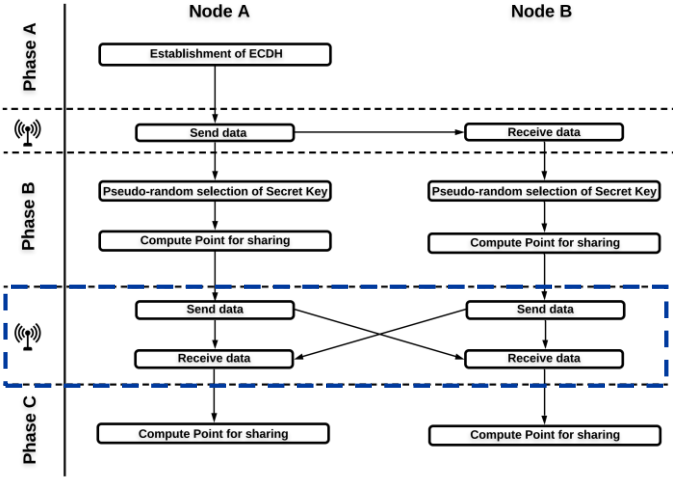
$$Public\ key = Secret\ key * G$$



Fáze B – vývojový diagram a blok SecretKey

- Výpočet bodu pro sdílení
- Tajný klíč (počet iterací)
- Generator (první bod grupy)





Odeslání a příjem dat B->C

Odeslání dat B->C

- Realizováno pomocí LWM protokolu
- Využívá knihovny **BigDigit**
- Data čerpána z globálních proměnných
- Konfigurace payloadu
- Nastavení endpointů
- Nutnost mazat nepoužívané bloky (úspora paměti)

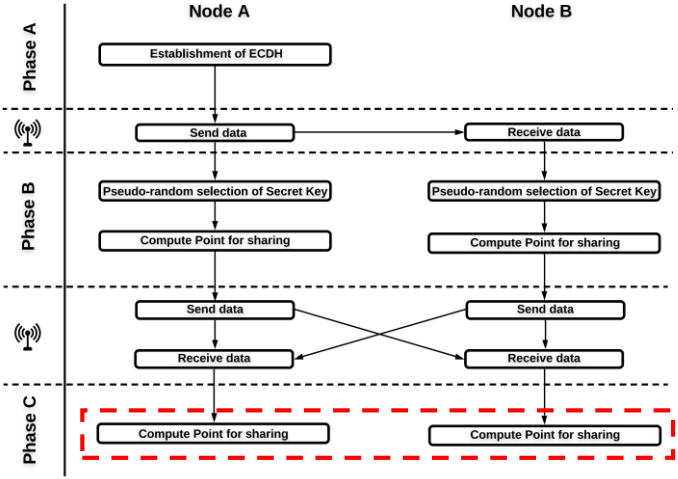
```
144
145 static void SendDataToC(void)
146 {
147     BB data[3];
148
149     for(uint8_t i = 0; i <= 2; i++){
150
151         data[i] = bdNew();              // block DEactivation
152     }
153
154
155     data[0] = resultsBA;  // X
156     data[1] = resultsBA;  // Y
157
158
159     appDataReq.dstAddr = 0xFFFF;
160     appDataReq.dstEndpoint = 3;
161     appDataReq.srcEndpoint = 2;
162     //appDataReq.options = NWK_OPT_ENABLE_SECURITY;
163     appDataReq.data = data;
164     appDataReq.size = 4;
165     appDataReq.confirm = appDataConf;
166     NWK_DataReq(&appDataReq);
167
168
169     appDataReqBusy = true;
170
171
172     for(uint8_t i = 0; i <= 2; i++){
173
174         bdFree(&data[i]);           // block DEactivation
175     }
176
```

Příjem dat B->C

- Realizováno pomocí LWM protokolu
- Využívá knihovny **BigDigit**
- Přenesení obdržených dat do proměnných
- Spuštění **aplikace pro fázi C**
- Kontrola výstupu => změna stavu/fáze

```
25
26  static bool ExchangePoint(NWK_DataInd_t *ind)
27  {
28      Xm = bdNew();
29      Ym = bdNew();
30      Xo = bdNew();
31      Yo = bdNew();
32      MOD = bdNew();
33      a_parameter = bdNew();
34      ZEROB = bdNew();
35
36      bdSetZero(ZEROB);
37
38      KEY = bdNew();
39
40
41      ECDH_PHASE_C_BIGD(Xm, Ym, Xo, Yo, MOD, a_parameter, KEY);
42
43
44      comaparsionB = bdCompare(KEY, ZEROB);
45
46      bdConvToDecimal(KEY, s, sizeof(s));
47      key = atoi(s);
48
49
50      bdFree(&Xm);
51      bdFree(&Ym);
52      bdFree(&Xo);
53      bdFree(&Yo);
54      bdFree(&MOD);
55      bdFree(&a_parameter);
56
57      bdFree(&KEY);
58
59  }
60
61
```

Metodika výměny klíčů – Fáze A

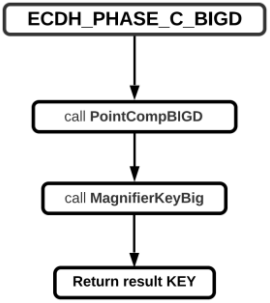


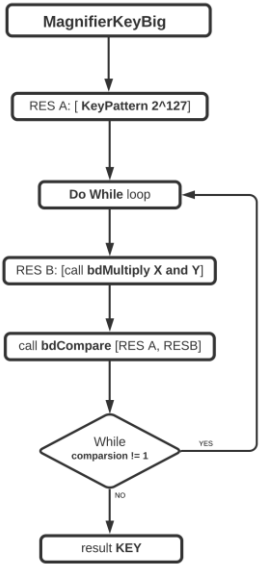
Fáze C

Fáze C – úvod

- Ze znalosti vlastního tajného klíče a obdrženého bodu se vypočítá společný bod
- Souřadnice se v aplikaci magnifierKeyBig upraví na 128bitový klíč
- Výstup se uloží do proměnné v LWM Stacku (jako klíč k AES)

```
void NWK_SetSecurityKey(uint8_t *key);
```





Děkuji za pozornost!