Protokol s koncovým šifrováním pro IEEE 802.15.4

Author: JAROMÍR BAČA

Advisor: Ing. ONDŘEJ KRAJSA, Ph.D.

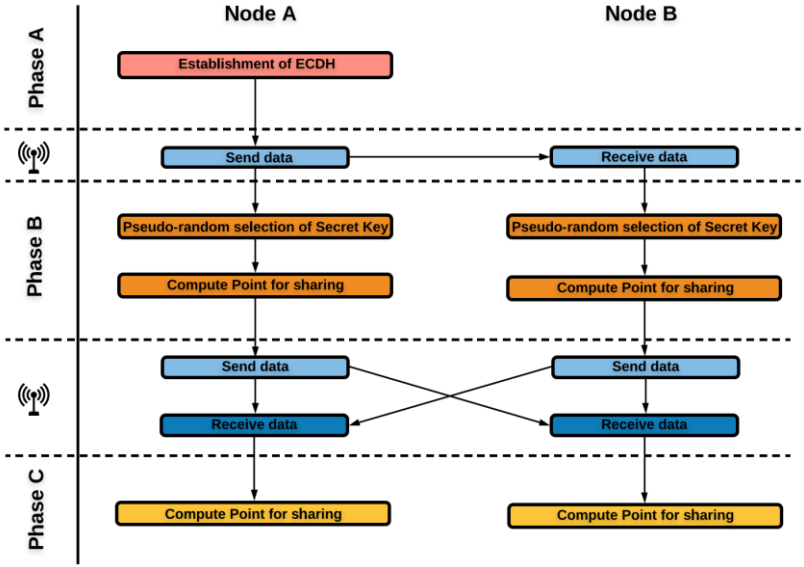
Brno, 6.1.2020

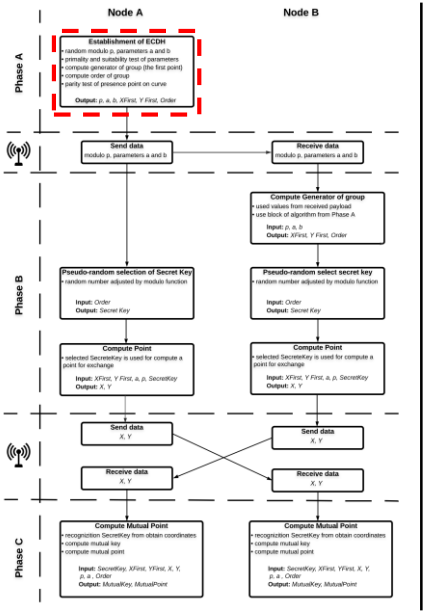
Cíle práce

- Osvojit si práci s mikrokontrolery AVR
- Nastudovat LWM stack
- Navrhnout metodu výměny klíčů mesi komunikačními body
- Implementovat navrženou metodu na do LWM stacku

Metodika výměny klíčů

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





Fáze A

Establishment of ECDH

- random modulo p, parameters a and b
- primality and suitability test of parameters
- compute generator of group (the first point)
- compute order of group
- parity test of presence point on curve

Output: *p, a, b, XFirst, Y First, Order*

Fáze A – sestavení parametrů výměny

- Pseudonáhodný výběr hodnoty p (modulo), a a b (parametry křivky)
- Kontrola správnosti vstupních hodnot
- Výpočet generátoru grupy (první bod) a řádu grupy
- Kontrola parity a důkaz, že poslední bod leží na křivce

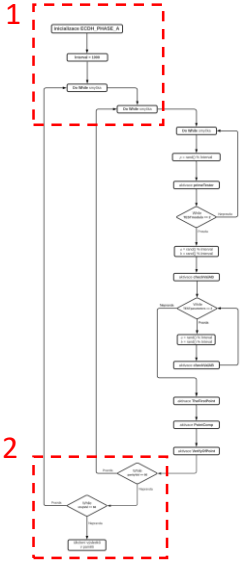
```
ECDH_PHASE_A(&mod, &a_parameter, &b_parameter, &X_first, &Y_first, &G_Order);
```

Vstup: žádný, aplikace si samostatně volí parametry

Výstup: modulus, a parameter, b parameter, XFirst, YFirst, GOrder

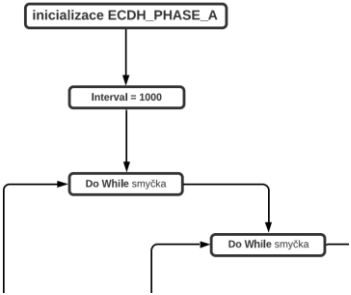
Fáze A – vývojový diagram

1

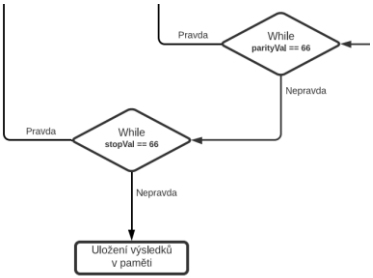


2

1

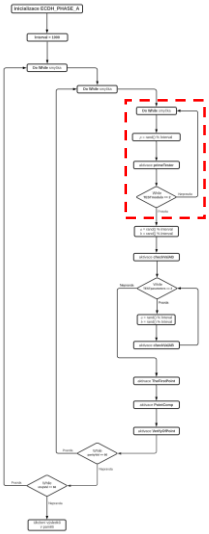


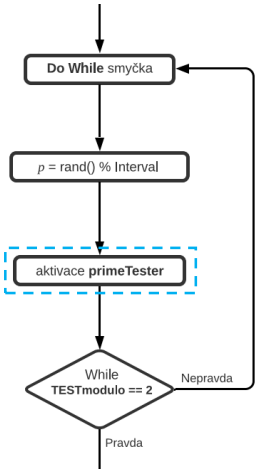
2



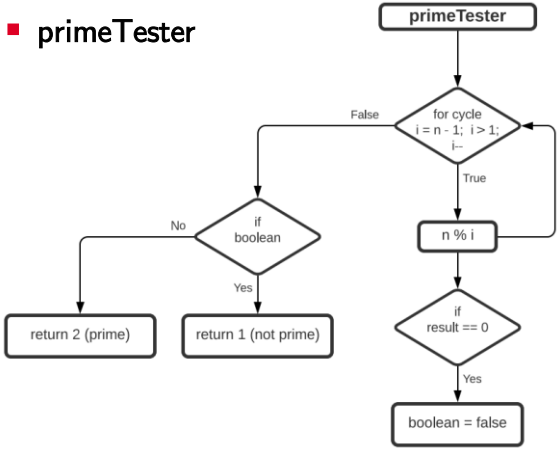
- Nastavení intervalu je součástí algoritmu
- Spuštění dvou testovacích smyček
- Test parity a test přítomnosti bodu na křivce

Fáze A – vývojový diagram (Výběr prvočísla)



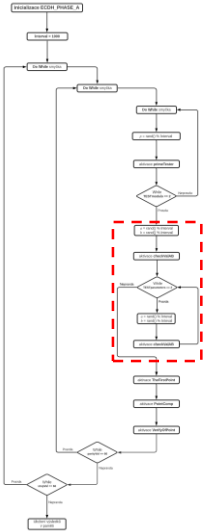


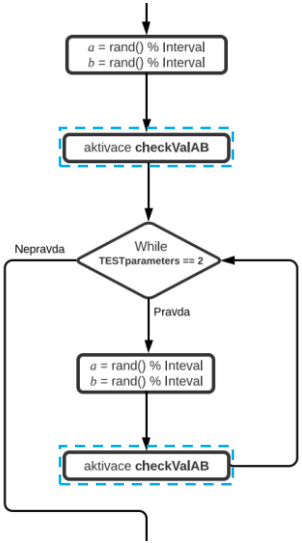
- primeTester

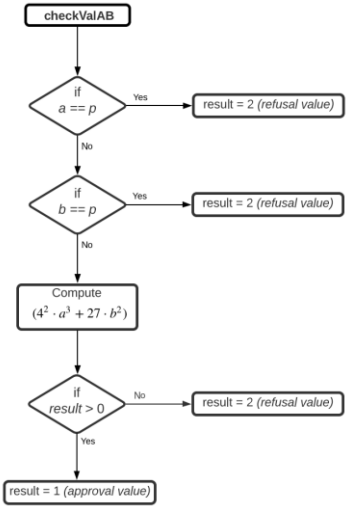


Příklad:
4 mod3 = 1
4 mod2 = 0 => beze zbytku => 4 není prvočíslo

Fáze A – vývojový diagram (Výběr parametrů)

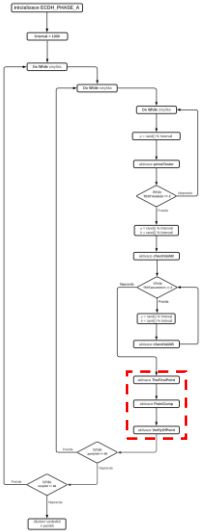


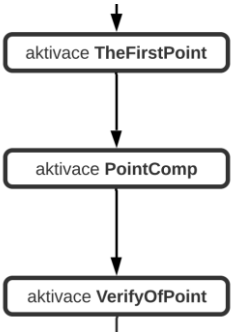




$$(4 \cdot a^3 + 27 \cdot b^2) \bmod p$$

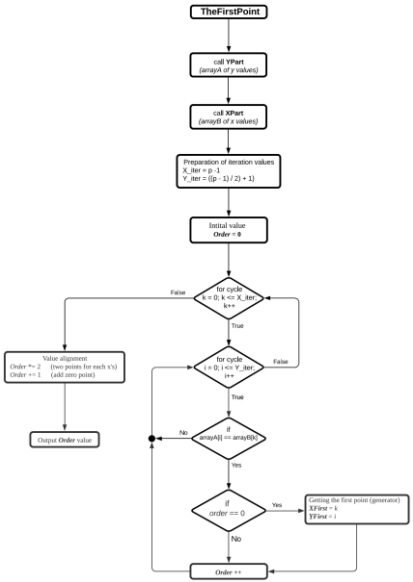
Fáze A – vývojový diagram





- TheFirstPoint
  - Výpočet generátoru grupy
  - Výpočet řádu grupy (počet bodů)
- PointComp
  - Výpočet všech bodů
  - Test parity
- VerifyOfPoint
  - Test přítomnosti bodu na křivce

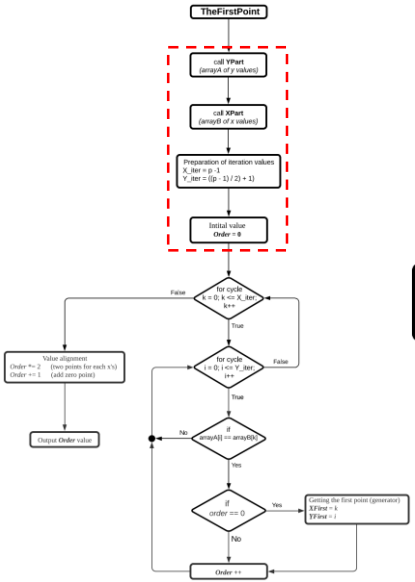
Fáze A – blok TheFistPoint (vývojový diagram)

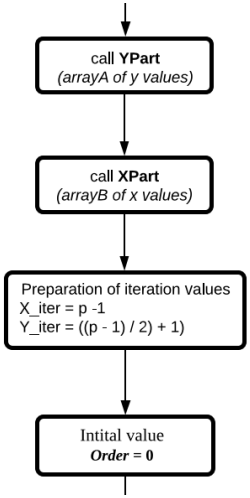


- TheFirstPoint
  - Výpočet generátoru grupy
  - Výpočet řádu grupy (počet bodů)

x	$x^3 + ax + b$	y	y^2	[X, Y], [X, Y]
0	6	-	-	-
1	1	1	1	[1, 1], [1, 6]
2	2	3	2	[2, 3], [2, 4]
3	1	1	1	[3, 1], [3, 6]
4	4	2	4	[4, 2], [4, 5]
5	3	-	-	-
6	4	2	4	[6, 2], [6, 5]
∞				0

Fáze A – blok TheFistPoint (vývojový diagram)



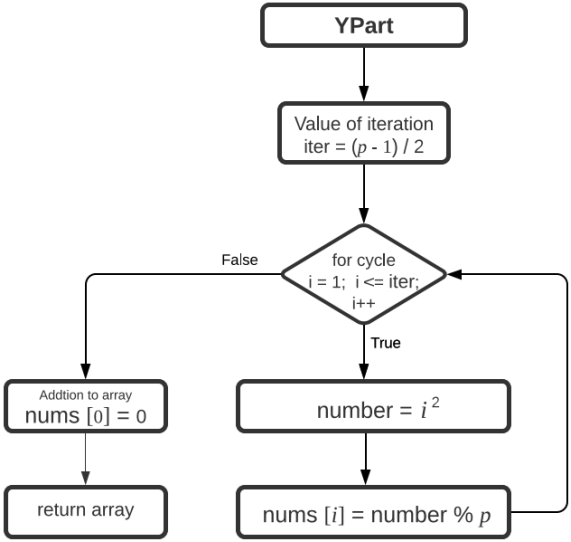


- Výpočet tabulky s Y hodnotami
- Výpočet tabulky s X hodnotami
- Nastavení počáteční hodnoty řádu grupy

Fáze A – blok TheFistPoint (tabulka Y)

- Příklad s hodnotou $p = 7$
- Výpočet tabulky s Y hodnotami

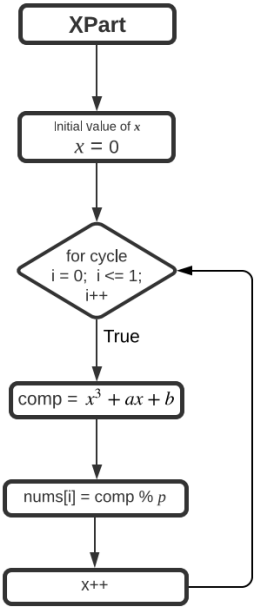
y	y^2
0	0
±1	1
±2	4
±3	2



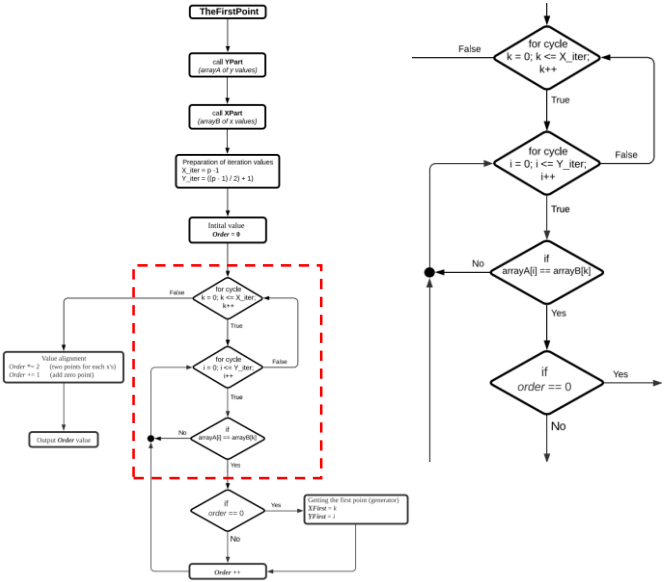
Fáze A – blok TheFistPoint (tabulka X)

- Příklad s hodnotami $p = 7$, a $= 1$, b $= 6$
- Výpočet tabulky s X hodnotami

x	$x^3 + ax + b$
0	6
1	1
2	2
3	1
4	4
5	3
6	4
∞	



Fáze A – blok TheFistPoint (hledání bodů)



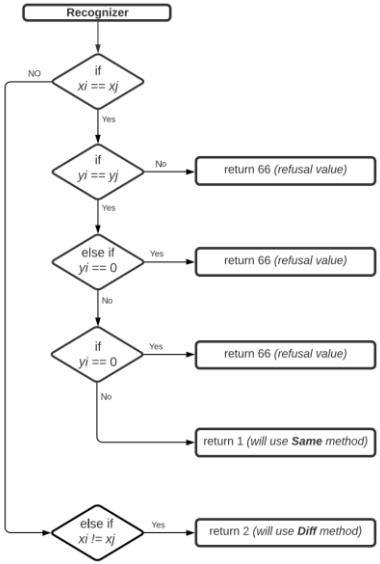
- Porovnání obou tabulek
- Shoda = nalezený bod
- Na pozadí běží počítání počtu bodů

Fáze A – blok TheFistPoint (hledání bodů)

- Ukázka části algoritmu, která porovnáním tabulek X a Y získá generátor grupy
- V místě, kde je nalezena shoda, jsou zaznamenány hodnoty iterací (souřadnice generátoru)
- S každou shodou je inkrementálně zvýšena hodnota řády grupy

```
for(int k = 0; k <= X_iter; k++)  // Give X's
    {
        for(int i = 0; i <= Y_iter; i++)  // Give Y's
        {
            if (poleA[i] == poleB[k])
            {
                if(order == 0)
                {

                    *Xfirst = k;
                    *Yfirst = i;
                }
```

Fáze A – blok PointComp (součet bodů)



P + P

LambdaSame
Application based on formula:

$$\lambda = \frac{3x^2+a}{2b} \bmod p$$

XSame
Application based on formula:

$$x_k = (\lambda^2 - 2x_i) \bmod p$$

YSame
Application based on formula:

$$y_k = (\lambda \cdot (x_i - x_k) - y_i) \bmod p$$

P + Q

LambdaDiff
Application based on formula:

$$\lambda = \frac{y_j - y_i}{x_j - x_i} \bmod p$$

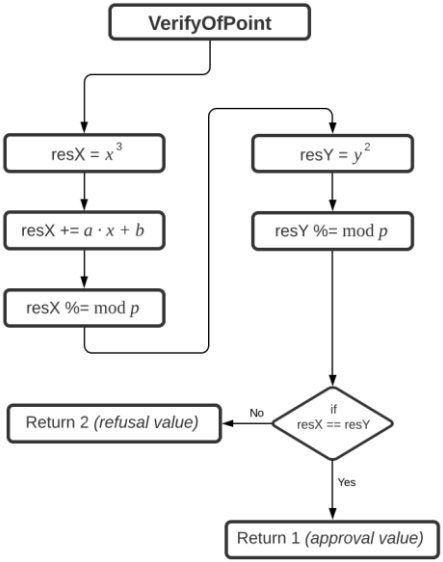
XDiff
Application based on formula:

$$x_k = (\lambda^2 - x_i - x_j) \bmod p$$

YDiff
Application based on formula:

$$y_k = (\lambda \cdot (x_i - x_k) - y_i) \bmod p$$

Fáze A – blok VerifyOfPoint (test přítomnosti bodu)



Bod $[5, 1], p = 17, a = 2, b = 2$

X: $x^3 + ax + b \bmod p$

Y: $y^2 \bmod p$

X: $5^3 + 2 \cdot 5 + 2 \bmod 17 = \mathbf{1}$

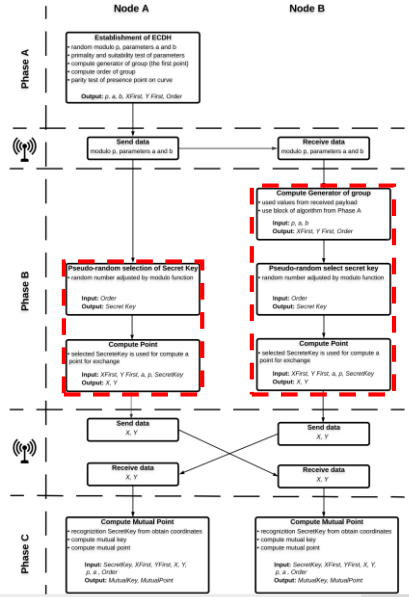
Y: $1^2 \bmod 17 = \mathbf{1}$

Shoda, bod leží na křivce

Fáze A – test aplikace

```
Modulo value is 5
Selected parameters are 1 and 6
Body jsou [   0,   1] [ 0, 4]
Body jsou [   2,   1] [ 2, 4]
Body jsou [   3,   1] [ 3, 4]
Body jsou [   4,   2] [ 4, 3]

The first point is [0, 1] and order of group is 9

Bod: [4, 2]

Bod: [2, 1]

Bod: [3, 4]

Bod: [3, 1]

Bod: [2, 4]

Bod: [4, 3]

Bod: [0, 4]

 Value X of the last point in group: 0
 Value Y of the last point in group: 4
 Result of parity test: 1
 Result of non-zero test: 1

Selected imput values are: mod = 5, parametry jsou 1 a 6
The first point has these coordinates [0, 1], Order is 9
```

- Testováno s deaktivovaným pseudonáhodným výběrem
- Aplikace (na pozadí) vypočítá celou grupu
- Pokud jsou výsledky obou testů rovny 1, jsou hodnoty podstoupeny k dalším fázím procesu



Fáze B

Pseudo-random selection of Secret Key
• random number adjusted by modulo function

Input: *Order*
Output: *Secret Key*

Compute Point
• selected SecreteKey is used for compute a point for exchange

Input: *XFirst, Y First, a, p, SecretKey*
Output: *X, Y*

Compute Generator of group
• used values from received payload
• use block of algorithm from Phase A

Input: *p, a, b*
Output: *XFirst, Y First, Order*

Pseudo-random select secret key
• random number adjusted by modulo function

Input: *Order*
Output: *Secret Key*

Compute Point
• selected SecreteKey is used for compute a point for exchange

Input: *XFirst, Y First, a, p, SecretKey*
Output: *X, Y*

Fáze B – úvod

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

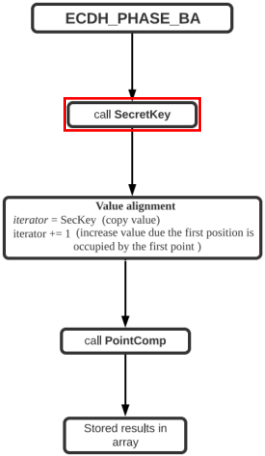
```
ECDH_PHASE_BA(&Order, &Xf, &Yf, &mod, &a, container);
```

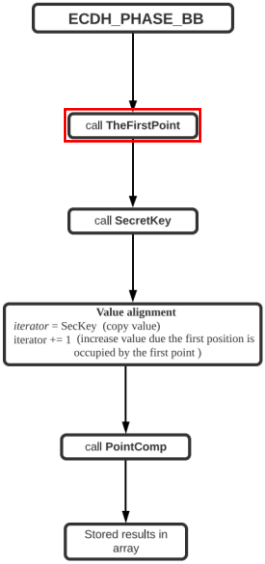
```
ECDH_PHASE_BB(&mod, &a, &b, container);
```

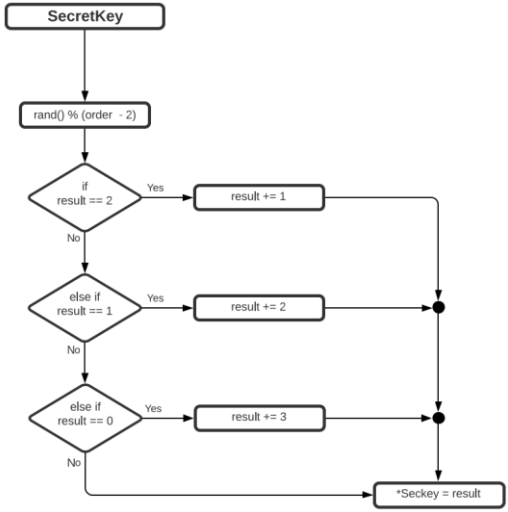
Vstup (subverze BA): řád, generátor grupy [X, Y], modulus, parametetry a, b
Vstup (subverze BB): modulus, parametetry a, b

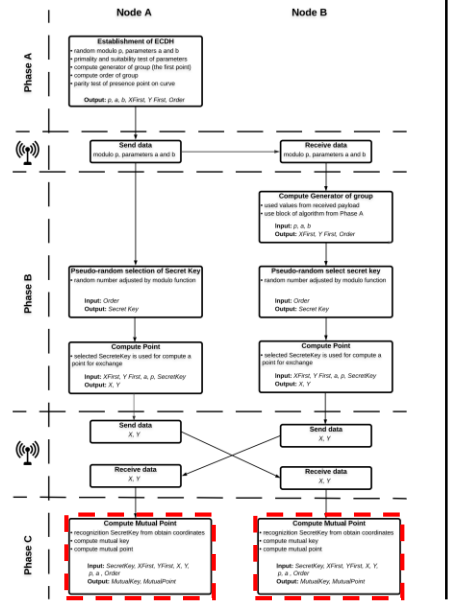
Výstup: pole obsahující souřadnice sdíleného bodu a tajný klíč

Fáze B – vývojový diagram a blok SecretKey









Fáze C

Compute Mutual Point

- recognizition SecretKey from obtain coordinates
- compute mutual key
- compute mutual point

Input: *SecretKey, XFirst, YFirst, X, Y, p, a , Order*
Output: *MutualKey, MutualPoint*

Fáze C – úvod

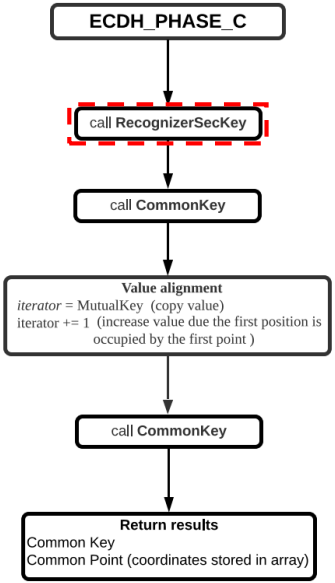
- Ze znalosti vlastního tajného klíče a obdrženého bodu vypočítá společný klíč
- Aplikace z obdrženého bodu rozpozná tajný klíč protistrany
- Ze získaného společného klíče se spočítá společný bod

```
void ECDH_PHASE_C(&MSKey, &XF, &YF, &XO, &YO, &mod, &a, &order, &MutKEY, container);
```

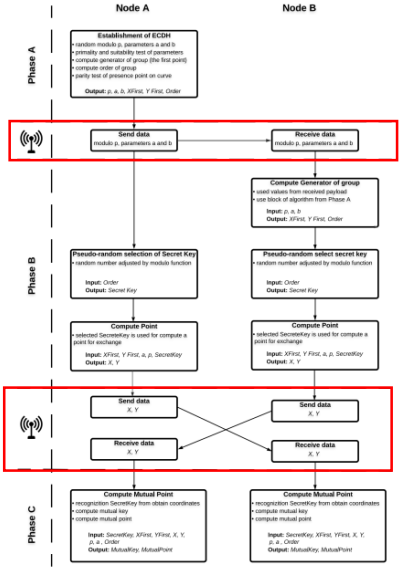
Vstup (subverze BA):
Vlastní tajný klíč, generátor grupy [X, Y], získaný bod [X, Y], modulus, parametr a

Výstup: pole obsahující společný klíč, resp. společný bod

Fáze C – vývojový diagram a blok RecognizerSecKey

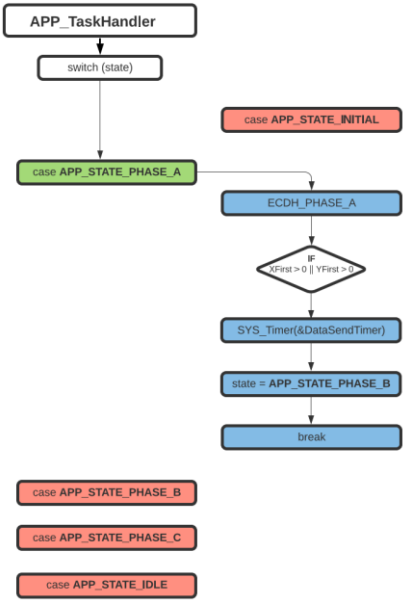


Implementace v rámci LWM STACKU



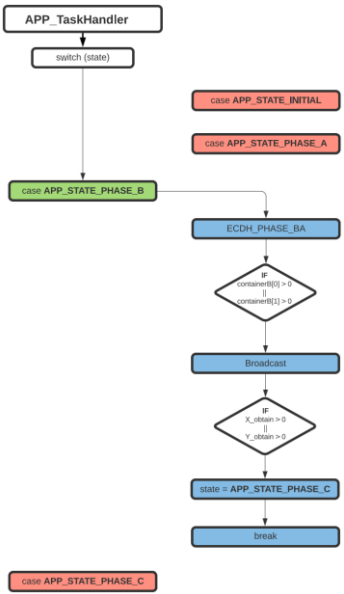
- Řešení bezdrátového přeposílání dat je řešeno na základě předchystaného LWM stacku
- V rámci implementace řešeno přepínání jednotlivých fází a spouštění vysílacích relací

Implementace v rámci LWM STACKU



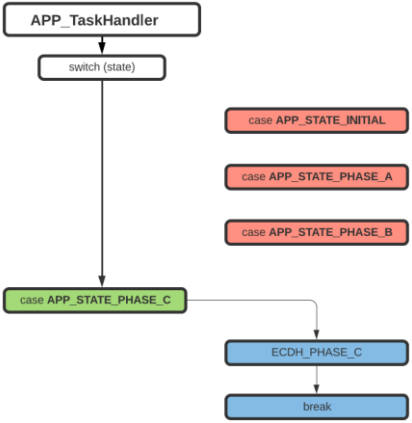
- Aktivace Fáze A, kde proběhne výběr vstupních parametrů a výpočtu generátoru grupy
- Pokud je splněna podmínka **if**, tedy že X nebo Y jsou větší než nula, znamená to, že první fáze úspěšně proběhla
- Pomocí **SYS_Timer**, který ovládá interval odesílání dat, jsou odeslány data protistraně
- V posledním příkazu je řešeno přepnutí programu na další fázi

Implementace v rámci LWM STACKU



- Aktivace Fáze BA, kde proběhne volba tajného klíče a následně výpočet bodu pro sdílení
- Pokud je splněna podmínka **if**, tedy že X nebo Y jsou větší než nula, znamená to, že **fáze BA** úspěšně proběhla
- Poté jsou souřadnice bodu X a Y odeslány protisttaně
- Pokud je splněna další podmínka **if**, znamená to, že byly obdrženy data od protistrany a algoritmus se může přepnout na třetí fázy

Implementace v rámci LWM STACKU



- Aktivace Fáze BA, kde výpočet společného klíče
- Do této fáze je implementován blok, který má za úkol zvýšit bitovou velikost vypočítaného společného klíče
- V této fázi je klíč předán do bloku AES, který jej použije k šifrování další komunikace

Děkuji za pozornost!