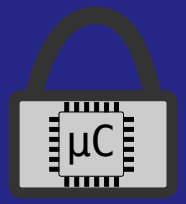
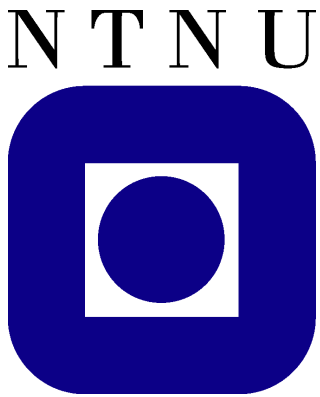
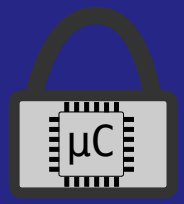


Optimizing BLUE MIDNIGHT WISH for size

Daniel Otte
(daniel.otte@rub.de)



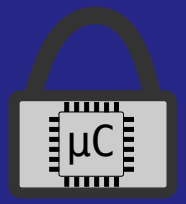
24. August 2010



shift and rotate

implement shift and rotate in dedicated functions:

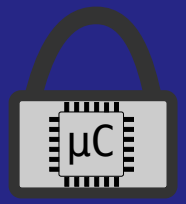
```
1  uint32_t shift32_left(uint32_t a, int8_t shift){
2    if(shift<0)
3      return(shift32_right(a, shift);
4    return a<<shift;
5  }
6
7  uint32_t shift32_right(uint32_t a, uint8_t shift){
8    return a>>shift;
9  }
10
11 uint32_t rotate32_left(uint32_t a, uint8_t rotate){
12   return (a<<rotate)|(a>>(32-rotate));
13 }
14
15 #define rotate32_right(a,n) rotate32_left((a), 32-(n))
```



memxor

xoring memory regions might come in handy

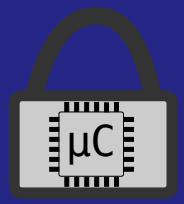
```
1  void memxor(void* dest, void* src, size_t n){
2    while(n--){
3      *((uint8_t*)dest) ^= *((uint8_t*)src);
4      dest = (uint8_t*)dest + 1;
5      src  = (uint8_t*)src  + 1;
6    }
7  }
```



definition of $S_{0..5}$

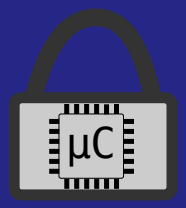
$$s_0(x) = SHR^1(x) \oplus SHL^3(x) \oplus ROTL^4(x) \oplus ROTR^{13}(x)$$
$$s_1(x) = SHR^1(x) \oplus SHL^2(x) \oplus ROTL^8(x) \oplus ROTR^9(x)$$
$$s_2(x) = SHR^2(x) \oplus SHL^1(x) \oplus ROTL^{12}(x) \oplus ROTR^7(x)$$
$$s_3(x) = SHR^2(x) \oplus SHL^2(x) \oplus ROTL^{15}(x) \oplus ROTR^3(x)$$
$$s_4(x) = SHR^1(x) \oplus x$$
$$s_5(x) = SHR^2(x) \oplus x$$

Table: Definition of the S functions



implementation of S functions


```
1  const uint8_t s_table[6][4] PROGMEM = {
2    {  1,  3,  4, 13 },
3    {  1,  2,  8,  9 },
4    {  2,  1, 12,  7 },
5    {  2,  2, 15,  3 },
6    {  1,  0,  0,  0 },
7    {  2,  0,  0,  0 },
8  };
9
10 uint32_t s32(uint32_t x, uint8_t s){
11   uint32_t r;
12   uint8_t *p = (uint8_t*)s_table+4*s;
13   r = shift32_right( x, pgm_read_byte(p++))
14     ^ shift32_left(  x, pgm_read_byte(p++))
15     ^ rotate32_left( x, pgm_read_byte(p++))
16     ^ rotate32_right(x, pgm_read_byte(p));
17   return r;
18 }
```



plain f0 function

$f_0 : \{0,1\}^{2m} \rightarrow \{0,1\}^m$

Input: Message block $M^{(i)} = (M_0^{(i)}, M_1^{(i)}, \ldots, M_{15}^{(i)})$, and the previous double pipe $H^{(i-1)} = (H_0^{(i-1)}, H_1^{(i-1)}, \ldots, H_{15}^{(i-1)})$.
Output: First part of the quadruple pipe $Q_a^{(i)} = (Q_0^{(i)}, Q_1^{(i)}, \ldots, Q_{15}^{(i)})$.

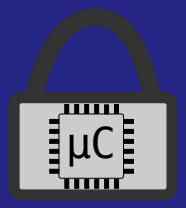
1. Bijective transform of $M^{(i)} \oplus H^{(i-1)}$:

$$
\begin{aligned}
W_0^{(i)} &= (M_5^{(i)} \oplus H_5^{(i-1)}) - (M_7^{(i)} \oplus H_7^{(i-1)}) + (M_{10}^{(i)} \oplus H_{10}^{(i-1)}) + (M_{13}^{(i)} \oplus H_{13}^{(i-1)}) + (M_{14}^{(i)} \oplus H_{14}^{(i-1)}) \\
W_1^{(i)} &= (M_6^{(i)} \oplus H_6^{(i-1)}) - (M_8^{(i)} \oplus H_8^{(i-1)}) + (M_{11}^{(i)} \oplus H_{11}^{(i-1)}) + (M_{14}^{(i)} \oplus H_{14}^{(i-1)}) - (M_{15}^{(i)} \oplus H_{15}^{(i-1)}) \\
W_2^{(i)} &= (M_0^{(i)} \oplus H_0^{(i-1)}) + (M_7^{(i)} \oplus H_7^{(i-1)}) + (M_9^{(i)} \oplus H_9^{(i-1)}) - (M_{12}^{(i)} \oplus H_{12}^{(i-1)}) + (M_{15}^{(i)} \oplus H_{15}^{(i-1)}) \\
W_3^{(i)} &= (M_0^{(i)} \oplus H_0^{(i-1)}) - (M_1^{(i)} \oplus H_1^{(i-1)}) + (M_8^{(i)} \oplus H_8^{(i-1)}) - (M_{10}^{(i)} \oplus H_{10}^{(i-1)}) + (M_{13}^{(i)} \oplus H_{13}^{(i-1)}) \\
W_4^{(i)} &= (M_1^{(i)} \oplus H_1^{(i-1)}) + (M_2^{(i)} \oplus H_2^{(i-1)}) + (M_9^{(i)} \oplus H_9^{(i-1)}) - (M_{11}^{(i)} \oplus H_{11}^{(i-1)}) - (M_{14}^{(i)} \oplus H_{14}^{(i-1)}) \\
W_5^{(i)} &= (M_3^{(i)} \oplus H_3^{(i-1)}) - (M_2^{(i)} \oplus H_2^{(i-1)}) + (M_{10}^{(i)} \oplus H_{10}^{(i-1)}) - (M_{12}^{(i)} \oplus H_{12}^{(i-1)}) + (M_{15}^{(i)} \oplus H_{15}^{(i-1)}) \\
W_6^{(i)} &= (M_4^{(i)} \oplus H_4^{(i-1)}) - (M_0^{(i)} \oplus H_0^{(i-1)}) - (M_3^{(i)} \oplus H_3^{(i-1)}) - (M_{11}^{(i)} \oplus H_{11}^{(i-1)}) + (M_{13}^{(i)} \oplus H_{13}^{(i-1)}) \\
W_7^{(i)} &= (M_1^{(i)} \oplus H_1^{(i-1)}) - (M_4^{(i)} \oplus H_4^{(i-1)}) - (M_5^{(i)} \oplus H_5^{(i-1)}) - (M_{12}^{(i)} \oplus H_{12}^{(i-1)}) - (M_{14}^{(i)} \oplus H_{14}^{(i-1)}) \\
W_8^{(i)} &= (M_2^{(i)} \oplus H_2^{(i-1)}) - (M_5^{(i)} \oplus H_5^{(i-1)}) - (M_6^{(i)} \oplus H_6^{(i-1)}) + (M_{13}^{(i)} \oplus H_{13}^{(i-1)}) - (M_{15}^{(i)} \oplus H_{15}^{(i-1)}) \\
W_9^{(i)} &= (M_0^{(i)} \oplus H_0^{(i-1)}) - (M_3^{(i)} \oplus H_3^{(i-1)}) + (M_6^{(i)} \oplus H_6^{(i-1)}) - (M_7^{(i)} \oplus H_7^{(i-1)}) + (M_{14}^{(i)} \oplus H_{14}^{(i-1)}) \\
W_{10}^{(i)} &= (M_8^{(i)} \oplus H_8^{(i-1)}) - (M_1^{(i)} \oplus H_1^{(i-1)}) - (M_4^{(i)} \oplus H_4^{(i-1)}) - (M_7^{(i)} \oplus H_7^{(i-1)}) + (M_{15}^{(i)} \oplus H_{15}^{(i-1)}) \\
W_{11}^{(i)} &= (M_8^{(i)} \oplus H_8^{(i-1)}) - (M_0^{(i)} \oplus H_0^{(i-1)}) - (M_2^{(i)} \oplus H_2^{(i-1)}) - (M_5^{(i)} \oplus H_5^{(i-1)}) + (M_9^{(i)} \oplus H_9^{(i-1)}) \\
W_{12}^{(i)} &= (M_1^{(i)} \oplus H_1^{(i-1)}) + (M_3^{(i)} \oplus H_3^{(i-1)}) - (M_6^{(i)} \oplus H_6^{(i-1)}) - (M_9^{(i)} \oplus H_9^{(i-1)}) + (M_{10}^{(i)} \oplus H_{10}^{(i-1)}) \\
W_{13}^{(i)} &= (M_2^{(i)} \oplus H_2^{(i-1)}) + (M_4^{(i)} \oplus H_4^{(i-1)}) + (M_7^{(i)} \oplus H_7^{(i-1)}) + (M_{10}^{(i)} \oplus H_{10}^{(i-1)}) + (M_{11}^{(i)} \oplus H_{11}^{(i-1)}) \\
W_{14}^{(i)} &= (M_3^{(i)} \oplus H_3^{(i-1)}) - (M_5^{(i)} \oplus H_5^{(i-1)}) + (M_8^{(i)} \oplus H_8^{(i-1)}) - (M_{11}^{(i)} \oplus H_{11}^{(i-1)}) - (M_{12}^{(i)} \oplus H_{12}^{(i-1)}) \\
W_{15}^{(i)} &= (M_{12}^{(i)} \oplus H_{12}^{(i-1)}) - (M_4^{(i)} \oplus H_4^{(i-1)}) - (M_6^{(i)} \oplus H_6^{(i-1)}) - (M_9^{(i)} \oplus H_9^{(i-1)}) + (M_{13}^{(i)} \oplus H_{13}^{(i-1)})
\end{aligned}
$$

2. Further bijective transform of $W_j^{(i)}, j = 0, \ldots, 15$:

$$
\begin{aligned}
&Q_0^{(i)} = s_0(W_0^{(i)}) + H_1^{(i-1)}; \quad Q_1^{(i)} = s_1(W_1^{(i)}) + H_2^{(i-1)}; \quad Q_2^{(i)} = s_2(W_2^{(i)}) + H_3^{(i-1)}; \quad Q_3^{(i)} = s_3(W_3^{(i)}) + H_4^{(i-1)}; \\
&Q_4^{(i)} = s_4(W_4^{(i)}) + H_5^{(i-1)}; \quad Q_5^{(i)} = s_0(W_5^{(i)}) + H_6^{(i-1)}; \quad Q_6^{(i)} = s_1(W_6^{(i)}) + H_7^{(i-1)}; \quad Q_7^{(i)} = s_2(W_7^{(i)}) + H_8^{(i-1)}; \\
&Q_8^{(i)} = s_3(W_8^{(i)}) + H_9^{(i-1)}; \quad Q_9^{(i)} = s_4(W_9^{(i)}) + H_{10}^{(i-1)}; \quad Q_{10}^{(i)} = s_0(W_{10}^{(i)}) + H_{11}^{(i-1)}; \quad Q_{11}^{(i)} = s_1(W_{11}^{(i)}) + H_{12}^{(i-1)}; \\
&Q_{12}^{(i)} = s_2(W_{12}^{(i)}) + H_{13}^{(i-1)}; \quad Q_{13}^{(i)} = s_3(W_{13}^{(i)}) + H_{14}^{(i-1)}; \quad Q_{14}^{(i)} = s_4(W_{14}^{(i)}) + H_{15}^{(i-1)}; \quad Q_{15}^{(i)} = s_0(W_{15}^{(i)}) + H_0^{(i-1)};
\end{aligned}
$$

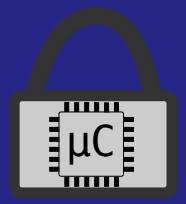
Table: Definition of **f0**



first part rearranged

$$T_j^{(i)} = M_j^{(i)} \oplus H_j^{(i-1)}$$

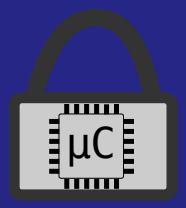
$$\begin{aligned}
W_0^{(i)} &= +T_5^{(i)} -T_7^{(i)} +T_{10}^{(i)} +T_{13}^{(i)} +T_{14}^{(i)} \\
W_1^{(i)} &= +T_6^{(i)} -T_8^{(i)} +T_{11}^{(i)} +T_{14}^{(i)} -T_{15}^{(i)} \\
W_2^{(i)} &= +T_7^{(i)} +T_9^{(i)} -T_{12}^{(i)} +T_{15}^{(i)} +T_0^{(i)} \\
W_3^{(i)} &= +T_8^{(i)} -T_{10}^{(i)} +T_{13}^{(i)} +T_0^{(i)} -T_1^{(i)} \\
W_4^{(i)} &= +T_9^{(i)} -T_{11}^{(i)} -T_{14}^{(i)} +T_1^{(i)} +T_2^{(i)} \\
W_5^{(i)} &= +T_{10}^{(i)} -T_{12}^{(i)} +T_{15}^{(i)} -T_2^{(i)} +T_3^{(i)} \\
W_6^{(i)} &= -T_{11}^{(i)} +T_{13}^{(i)} -T_0^{(i)} -T_3^{(i)} +T_4^{(i)} \\
W_7^{(i)} &= -T_{12}^{(i)} -T_{14}^{(i)} +T_1^{(i)} -T_4^{(i)} -T_5^{(i)} \\
W_8^{(i)} &= +T_{13}^{(i)} -T_{15}^{(i)} +T_2^{(i)} -T_5^{(i)} -T_6^{(i)} \\
W_9^{(i)} &= +T_{14}^{(i)} +T_0^{(i)} -T_3^{(i)} +T_6^{(i)} -T_7^{(i)} \\
W_{10}^{(i)} &= +T_{15}^{(i)} -T_1^{(i)} -T_4^{(i)} -T_7^{(i)} +T_8^{(i)} \\
W_{11}^{(i)} &= -T_0^{(i)} -T_2^{(i)} -T_5^{(i)} +T_8^{(i)} +T_9^{(i)} \\
W_{12}^{(i)} &= +T_1^{(i)} +T_3^{(i)} -T_6^{(i)} -T_9^{(i)} +T_{10}^{(i)} \\
W_{13}^{(i)} &= +T_2^{(i)} +T_4^{(i)} +T_7^{(i)} +T_{10}^{(i)} +T_{11}^{(i)} \\
W_{14}^{(i)} &= +T_3^{(i)} -T_5^{(i)} +T_8^{(i)} -T_{11}^{(i)} -T_{12}^{(i)} \\
W_{15}^{(i)} &= -T_4^{(i)} -T_6^{(i)} -T_9^{(i)} +T_{12}^{(i)} +T_{13}^{(i)}
\end{aligned}$$



sign translation

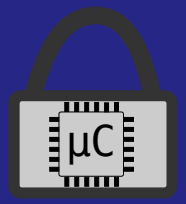
						0x0311	0xDDB3	0x2A79	0x07AA	0x51C2
0 (msb):	+	-	+	+	+	0	1	0	0	0
1 :	+	-	+	+	-	0	1	0	0	1
2 :	+	+	-	+	+	0	0	1	0	0
3 :	+	-	+	+	-	0	1	0	0	1
4 :	+	-	-	+	+	0	1	1	0	0
5 :	+	-	+	-	+	0	1	0	1	0
6 :	-	+	-	-	+	1	0	1	1	0
7 :	-	-	+	-	-	1	1	0	1	1
8 :	+	-	+	-	-	0	1	0	1	1
9 :	+	+	-	+	-	0	0	1	0	1
10 :	+	-	-	-	+	0	1	1	1	0
11 :	-	-	-	+	+	1	1	1	0	0
12 :	+	+	-	-	+	0	0	1	1	0
13 :	+	+	+	+	+	0	0	0	0	0
14 :	+	-	+	-	-	0	1	0	1	1
15 (lsb):	-	-	-	+	+	1	1	1	0	0

Table: sign translation



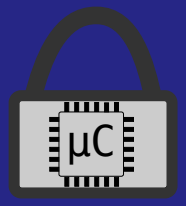
example C code


```
1  const uint16_t sign_table[] =
2    { 0x0311, 0xDDB3, 0x2A79,
3      0x07AA, 0x51C2 };
4  const uint8_t offset_table[] =
5    { 4, 6, 9, 12, 13};
6
7  void bmw32_f0(uint32_t* q,
8       uint32_t* h, uint32_t* m){
9    uint8_t i,row,column;
10   int8_t s_select;
11   uint16_t sign_reg;
12
13   /* xor m into h */
14   memxor(h, m, 64);
15   /* set q array to zero */
16   memset(q, 0, 4*16);
17   column = 4;
18   do{
19     i = 15;
20     row = offset_table[column];
21     sign_reg = hack_table[column];
22     do{
23       if(sign_reg&1){
24         q[i] -= h[row&15];
25       }else{
26         q[i] += h[row&15];
27       }
28       --row;
29       sign_reg >>= 1;
30     }while(i-- != 0);
31   }while(column-- != 0);
32   /* xor m into h again */
33   memxor(h, m, 64);
34   i = 15;
35   s_select = 0;
36   do{
37     q[i] = s32(q[i], s_select--)
38            + h[(i+1)&15];
39     if( s_select == -1){
40       s_select = 4;
41     }
42   }while(i-- != 0);
43 }
```



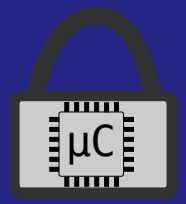
optimizations employed for f1

- use dedicated function for `AddElement`



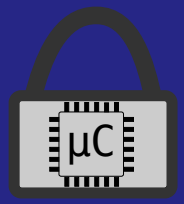
optimizations employed for f1

- use dedicated function for `AddElement`
- implement r-functions as immediate rotates



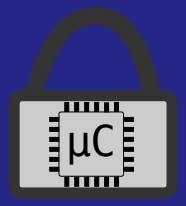
f1 basic function


```
1  void bmw_small_f1(uint32_t* q, const void* m, const void* h){
2  uint8_t i;
3  q[16] = bmw_small_expand1(0, q, m, h);
4  q[17] = bmw_small_expand1(1, q, m, h);
5  for(i=2; i<16; ++i){
6  q[16+i] = bmw_small_expand2(i, q, m, h);
7  }
8  }
```



bmw_small_expand1 function

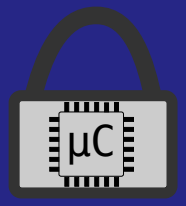

```
1  uint32_t bmw_small_expand1(uint8_t j, const uint32_t* q,
2              const void* m, const void* h){
3  uint32_t r;
4  uint8_t i;
5  r = expand_base(j, m) ^ ((uint32_t*)h)[(j+7)&0xf];
6  for(i=0; i<16; ++i){
7  r += s32(q[j+i], (i+1)%4);
8  }
9  return r;
10 }
```



bmw_small_expand2 function

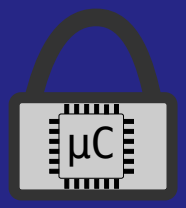
Daniel Otte (daniel.otte@rub.de)


```
1  uint32_t bmw_small_expand2(uint8_t j, const uint32_t* q,
2                 const void* m, const void* h){
3  uint8_t rotates[] = { 3, 7, 13, 16, 19, 23, 27};
4  uint32_t r;
5  uint8_t i;
6  r = expand_base(j, m) ^ ((uint32_t*)h)[(j+7)&0xf];
7  for(i=0; i<14; i+=2){
8  r += q[j+i];
9  r += rotate32_left(q[j+i+1], rotates[i/2]);
10 }
11 r += s32(q[j+14], 4);
12 r += s32(q[j+15], 5);
13 return r;
14 }
```



expand_base function


```
1  uint32_t expand_base(uint8_t j, const void* m){
2  return (    rotate32_left(((uint32_t*)m)[j&0xf],
3                                ((j+0)&0xf)+1   )
4         + rotate32_left(((uint32_t*)m)[(j+3)&0xf],
5                                ((j+3)&0xf)+1   )
6         - rotate32_left(((uint32_t*)m)[(j+10)&0xf],
7                                ((j+10)&0xf)+1  )
8         + pgm_read_dword_near(&(k_lut[j])));
9  }
```



f2 function

Folding $f_2 : \{0,1\}^{3m} \rightarrow \{0,1\}^m$

Input: Message block $M^{(i)} = (M_0^{(i)}, M_1^{(i)}, \ldots, M_{15}^{(i)})$,
quadruple pipe $Q^{(i)} = (Q_0^{(i)}, Q_1^{(i)}, \ldots, Q_{15}^{(i)}, Q_{16}^{(i)}, \ldots, Q_{31}^{(i)})$.
Output: New double pipe $H^{(i)} = (H_0^{(i)}, H_1^{(i)}, \ldots, H_{15}^{(i)})$.

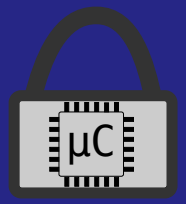
1. Compute the cumulative temporary variables XL and XH.

$$XL = Q_{16}^{(i)} \oplus Q_{17}^{(i)} \oplus \ldots \oplus Q_{23}^{(i)}$$
$$XH = XL \oplus Q_{24}^{(i)} \oplus Q_{25}^{(i)} \oplus \ldots \oplus Q_{31}^{(i)}$$

2. Compute the new double pipe $H^{(i)}$:

$$\begin{aligned}
H_0^{(i)} &= \left(SHL^5(XH) \oplus SHR^5(Q_{16}^{(i)}) \oplus M_0^{(i)}\right) + \left(XL \oplus Q_0^{(i)} \oplus Q_{24}^{(i)}\right) \\
H_1^{(i)} &= \left(SHR^7(XH) \oplus SHL^8(Q_{17}^{(i)}) \oplus M_1^{(i)}\right) + \left(XL \oplus Q_1^{(i)} \oplus Q_{25}^{(i)}\right) \\
H_2^{(i)} &= \left(SHR^5(XH) \oplus SHL^5(Q_{18}^{(i)}) \oplus M_2^{(i)}\right) + \left(XL \oplus Q_2^{(i)} \oplus Q_{26}^{(i)}\right) \\
H_3^{(i)} &= \left(SHR^1(XH) \oplus SHL^5(Q_{19}^{(i)}) \oplus M_3^{(i)}\right) + \left(XL \oplus Q_3^{(i)} \oplus Q_{27}^{(i)}\right) \\
H_4^{(i)} &= \left(SHR^3(XH) \oplus Q_{20}^{(i)} \oplus M_4^{(i)}\right) + \left(XL \oplus Q_4^{(i)} \oplus Q_{28}^{(i)}\right) \\
H_5^{(i)} &= \left(SHL^6(XH) \oplus SHR^6(Q_{21}^{(i)}) \oplus M_5^{(i)}\right) + \left(XL \oplus Q_5^{(i)} \oplus Q_{29}^{(i)}\right) \\
H_6^{(i)} &= \left(SHR^4(XH) \oplus SHL^6(Q_{22}^{(i)}) \oplus M_6^{(i)}\right) + \left(XL \oplus Q_6^{(i)} \oplus Q_{30}^{(i)}\right) \\
H_7^{(i)} &= \left(SHR^{11}(XH) \oplus SHL^2(Q_{23}^{(i)}) \oplus M_7^{(i)}\right) + \left(XL \oplus Q_7^{(i)} \oplus Q_{31}^{(i)}\right) \\
H_8^{(i)} &= ROTL^9(H_4^{(i)}) + \left(XH \oplus Q_{24}^{(i)} \oplus M_8^{(i)}\right) + \left(SHL^8(XL) \oplus Q_8^{(i)} \oplus Q_{23}^{(i)}\right) \\
H_9^{(i)} &= ROTL^{10}(H_5^{(i)}) + \left(XH \oplus Q_{25}^{(i)} \oplus M_9^{(i)}\right) + \left(SHR^6(XL) \oplus Q_9^{(i)} \oplus Q_{16}^{(i)}\right) \\
H_{10}^{(i)} &= ROTL^{11}(H_6^{(i)}) + \left(XH \oplus Q_{26}^{(i)} \oplus M_{10}^{(i)}\right) + \left(SHL^6(XL) \oplus Q_{10}^{(i)} \oplus Q_{17}^{(i)}\right) \\
H_{11}^{(i)} &= ROTL^{12}(H_7^{(i)}) + \left(XH \oplus Q_{27}^{(i)} \oplus M_{11}^{(i)}\right) + \left(SHL^4(XL) \oplus Q_{11}^{(i)} \oplus Q_{18}^{(i)}\right) \\
H_{12}^{(i)} &= ROTL^{13}(H_0^{(i)}) + \left(XH \oplus Q_{28}^{(i)} \oplus M_{12}^{(i)}\right) + \left(SHR^3(XL) \oplus Q_{12}^{(i)} \oplus Q_{19}^{(i)}\right) \\
H_{13}^{(i)} &= ROTL^{14}(H_1^{(i)}) + \left(XH \oplus Q_{29}^{(i)} \oplus M_{13}^{(i)}\right) + \left(SHR^4(XL) \oplus Q_{13}^{(i)} \oplus Q_{20}^{(i)}\right) \\
H_{14}^{(i)} &= ROTL^{15}(H_2^{(i)}) + \left(XH \oplus Q_{30}^{(i)} \oplus M_{14}^{(i)}\right) + \left(SHR^7(XL) \oplus Q_{14}^{(i)} \oplus Q_{21}^{(i)}\right) \\
H_{15}^{(i)} &= ROTL^{16}(H_3^{(i)}) + \left(XH \oplus Q_{31}^{(i)} \oplus M_{15}^{(i)}\right) + \left(SHR^2(XL) \oplus Q_{15}^{(i)} \oplus Q_{22}^{(i)}\right)
\end{aligned}$$

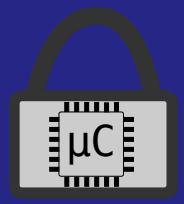
(1) (2) (3) (4) (5) (6) (7)



C implementation

Basic facts of our small C implementation (using avr-gcc):

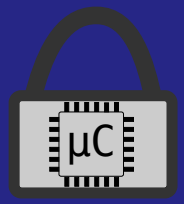
- codesize: 4446 bytes



C implementation

Basic facts of our small C implementation (using avr-gcc):

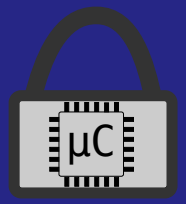
- codesize: 4446 bytes
- stacksize: 289 bytes



C implementation

Basic facts of our small C implementation (using avr-gcc):

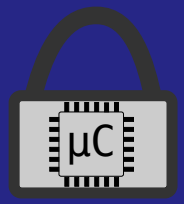
- codesize: 4446 bytes
- stacksize: 289 bytes
- speed (infinite long message): 1720.16 cycles/byte



ASM implementation

Basic facts of our small assembly implementation (calling convention from avr-gcc):

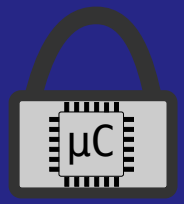
- codesize: 1536 bytes



ASM implementation

Basic facts of our small assembly implementation (calling convention from avr-gcc):

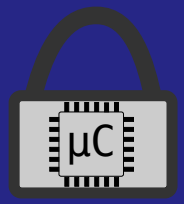
- codesize: 1536 bytes
- stacksize: 246 bytes



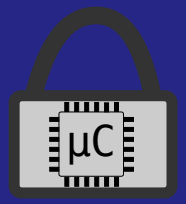
ASM implementation

Basic facts of our small assembly implementation (calling convention from avr-gcc):

- codesize: 1536 bytes
- stacksize: 246 bytes
- speed (infinite long message): 826.95 cycles/byte



- BMW Specification: http://www.q2s.ntnu.no/blue_midnight_wish/start
- AVR-Crypto-Lib: http://www.das-labor.org/wiki/AVR-Crypto-Lib/en
- Source Code: http://avrcryptolib.das-labor.org/trac/browser/bmw



EOP

Thank you for your attention.