

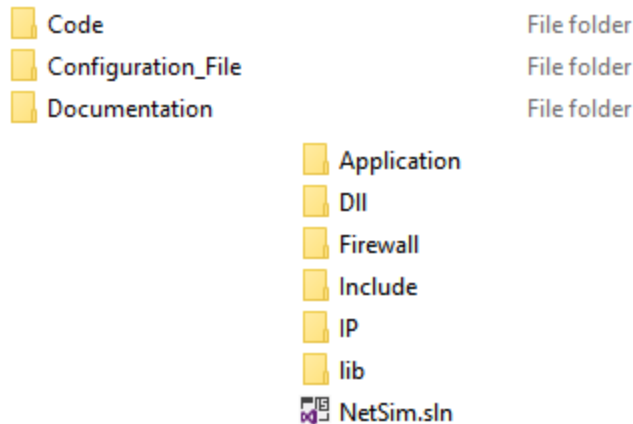
Implementing a new Crypto Algorithm – Mysty1

Software Recommended: NetSim Standard v11.0, Visual Studio 2015/2017, Wireshark

Project Download Link:

https://github.com/NetSim-TETCOS/MYSTY1_ENCRYPTION_v11.0/archive/master.zip

1. How to add a new crypto algorithm for encryption, unzip “Addition of Mysty Encryption Algorithm” project. It would look like



2. Double click on the NetSim.sln file to view the Application codes in Visual Studio 2015.

```
19 #include <string.h>
20 #include "application.h"
21 typedef unsigned long u4;
22 typedef unsigned char byte;
23 #define MISTY1_KEYSIZE 32
24 #define uint8 unsigned char
25
26 static byte s7[] =
27 {
28     0x1b, 0x32, 0x33, 0x5a, 0x3b, 0x10, 0x17, 0x54, 0x5b, 0x1a, 0x72, 0x73, 0x6b, 0x2c, 0x66, 0x49,
29     0x1f, 0x24, 0x13, 0x6c, 0x37, 0x2e, 0x3f, 0x4a, 0x5d, 0x0f, 0x40, 0x56, 0x25, 0x51, 0x1c, 0x04,
30     0x0b, 0x46, 0x20, 0x0d, 0x7b, 0x35, 0x44, 0x42, 0x2b, 0x1e, 0x41, 0x14, 0x4b, 0x79, 0x15, 0x6f,
31     0x0e, 0x55, 0x09, 0x36, 0x74, 0x0c, 0x67, 0x53, 0x28, 0x0a, 0x7e, 0x38, 0x02, 0x07, 0x60, 0x29,
32     0x19, 0x12, 0x65, 0x2f, 0x30, 0x39, 0x08, 0x68, 0x5f, 0x78, 0x2a, 0x4c, 0x64, 0x45, 0x75, 0x3d,
33     0x59, 0x48, 0x03, 0x57, 0x7c, 0x4f, 0x62, 0x3c, 0x1d, 0x21, 0x5e, 0x27, 0x6a, 0x70, 0x4d, 0x3a,
34     0x01, 0x6d, 0x6e, 0x63, 0x18, 0x77, 0x23, 0x05, 0x26, 0x76, 0x00, 0x31, 0x2d, 0x7a, 0x7f, 0x61,
35     0x50, 0x22, 0x11, 0x06, 0x47, 0x16, 0x52, 0x4e, 0x71, 0x3e, 0x69, 0x43, 0x34, 0x5c, 0x58, 0x7d
36 };
37 static u4 s9[] =
38 {
39     0x1c3, 0x0cb, 0x153, 0x19f, 0x1e3, 0x0e9, 0x0fb, 0x035, 0x181, 0x0b9, 0x117, 0x1eb, 0x133, 0x009, 0x02d, 0x0d3,
40     0x0c7, 0x14a, 0x037, 0x07e, 0x0eb, 0x164, 0x193, 0x1d8, 0x0a3, 0x11e, 0x055, 0x02c, 0x01d, 0x1a2, 0x163, 0x118,
41     0x14b, 0x152, 0x1d2, 0x00f, 0x02b, 0x030, 0x13a, 0x0e5, 0x111, 0x138, 0x18e, 0x063, 0x0e3, 0x0c8, 0x1f4, 0x01b,
42     0x001, 0x09d, 0x0f8, 0x1a0, 0x16d, 0x1f3, 0x01c, 0x146, 0x07d, 0x001, 0x082, 0x1ea, 0x183, 0x12d, 0x0f4, 0x19e,
43     0x1d3, 0x0dd, 0x1e2, 0x128, 0x1e0, 0x0ec, 0x059, 0x091, 0x011, 0x12f, 0x026, 0x0dc, 0x0b0, 0x18c, 0x10f, 0x1f7,
44     0x0e7, 0x16c, 0x0b6, 0x0f9, 0x0d0, 0x151, 0x101, 0x14c, 0x103, 0x008, 0x154, 0x12b, 0x1ae, 0x017, 0x071, 0x00c,
```

3. Now expand Application Project and click misty_run.c file. This file contains the following lines of code.

```
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include "application.h"
void misty_run(char* str,int* len)
{
    int n;
    int l=*len;

    unsigned char buf[32];
    unsigned char key[32];
```

```

    for (n = 0; n < *len; n+=16, str+=16, l-=16)
    {
        /* Set the plain-text */
        memcpy( buf, str, min(16,l));

        misty1_main(buf);
        memcpy(str,buf,16);
    }

}

```

In the `misty_run()` function inside the `misty_run.c` file we pass the plain text in parts of 16 bytes each time to get it encrypted. This is done because the crypto algorithm expects a 16 byte plaintext as input. Here the variable `str` contains the packet payload and `len` corresponds to the size of payload in bytes.

4. Now double click on the `misty1.c` file present in Application project inside the solution explorer and check the following changes to it.

- a) Addition of `#include <application.h>` and `#define uint8 unsigned char` to the beginning of the file (shown in red).

```

#include <stdlib.h>
#include <string.h>
#include "application.h"
typedef unsigned long u4;
typedef unsigned char byte;
#define MISTY1_KEYSIZE 32
#define uint8 unsigned char

```

- b) Removed `inline` keyword that is present before the functions `fi()`, `fo()`, `fl()` and `flinv()`.

```

inline u4 fi( u4 fi_in, u4 fi_key) { ... }

inline u4 fo(u4 *ek, u4 fo_in, byte k) { ... }

inline u4 fl(u4 *ek, u4 fl_in, byte k) { ... }

inline u4 flinv(u4 *ek, u4 fl_in, byte k) { ... }

```

To

```

u4 fi( u4 fi_in, u4 fi_key) { ... }

u4 fo(u4 *ek, u4 fo_in, byte k) { ... }

u4 fl(u4 *ek, u4 fl_in, byte k) { ... }

u4 flinv(u4 *ek, u4 fl_in, byte k) { ... }

```

- c) Now go to the `main()` function in the file and check that line `#ifdef TESTMAIN` was removed or commented before the `main()` function and also the associated `#endif` at the end of the `main()` function.

- d) `main()` function was renamed to `unsigned char* misty1_main(uint8* input)`

```

unsigned char* misty1_main(uint8* input)
{
/*
Key:      00 11 22 33 44 55 66 77 88 99 aa bb cc dd ee ff
Plaintext: 01 23 45 67 89 ab cd ef fe dc ba 98 76 54 32 10
Ciphertext: 8b 1d a5 f5 6a b3 d0 7c 04 b6 82 40 b1 3b e9 5d
*/

u4 Key[] = {0x00112233, 0x44556677, 0x8899aabb, 0xccddeeff};
u4 Plaintext[] = {0x01234567, 0x89abcdef, 0xfedcba98, 0x76543210};
u4 Ciphertext[] = { 0x8b1da5f5, 0x6ab3d07c, 0x04b68240, 0xb13be95d};
u4 ek_e[MISTY1_KEYSIZE], ek_d[MISTY1_KEYSIZE];
u4 c[4];

```

- e) Commented the declaration of Ciphertext, Modify the declaration of Plaintext variable, as shown below:

```

u4 Key[] = {0x00112233, 0x44556677, 0x8899aabb, 0xccddeeff};
u4 Plaintext[4];
//u4 Ciphertext[] = { 0x8b1da5f5, 0x6ab3d07c, 0x04b68240, 0xb13be95d};
u4 ek_e[MISTY1_KEYSIZE], ek_d[MISTY1_KEYSIZE];
u4 c[4];

```

- f) Now check the commented lines starting from misty1_keyinit() to misty1_key_destroy() as shown below:

```

misty1.c - Application (Global Scope)
283  /* misty1_keyinit(ek_e,Key);
284  misty1_encrypt_block(ek_e,&Plaintext[0],&c[0]);
285  misty1_encrypt_block(ek_e,&Plaintext[2],&c[2]);
286
287  if (!memcmp(c,Ciphertext,4 * sizeof(u4))) {
288  printf("Encryption OK\n");
289  }
290  else {
291  printf("Encryption failed[0x%08lx 0x%08lx 0x%08lx 0x%08lx]\n",
292  c[0],c[1],c[2],c[3]);
293  exit(1);
294  }
295
296  misty1_keyinit(ek_d,Key);
297
298  if (memcmp(ek_e,ek_d,MISTY1_KEYSIZE*sizeof(u4))) {
299  printf("Internal Error keysch is wrong\n");
300  exit(1);
301  }
302
303  misty1_decrypt_block(ek_d,&Ciphertext[0],&c[0]);
304  misty1_decrypt_block(ek_d,&Ciphertext[2],&c[2]);
305
306
307  if (!memcmp(c,Plaintext,4 * sizeof(u4))) {
308  printf("Decryption OK\n");
309  }
310  else {
311
312  printf("Decryption failed[0x%08lx 0x%08lx 0x%08lx 0x%08lx]\n",
313  c[0],c[1],c[2],c[3]);
314  exit(1);
315  }
316  */

```

- g) Addition of the following lines of code just above the misty1_key_destroy(ek_e); statement as shown below:

```

// Memcpy is used to equate input which is Char to Plaintext

```

```

// which is Unsigned Long

memcpy(Plaintext,input,2*sizeof(u4));
memcpy(&Plaintext[2],&input[8],2*sizeof(u4));

misty1_keyinit(ek_e,Key);
misty1_encrypt_block(ek_e,Plaintext,&c[0]);
misty1_encrypt_block(ek_e,&Plaintext[2],&c[2]);

memcpy(input,c,2*sizeof(u4));
memcpy(&input[8],&c[2],2*sizeof(u4));

```

```

// Mmcpy is used to equate input which is Char to Plaintext
// which is Unsigned Long

memcpy(Plaintext,input,2*sizeof(u4));
memcpy(&Plaintext[2],&input[8],2*sizeof(u4));

misty1_keyinit(ek_e,Key);
misty1_encrypt_block(ek_e,Plaintext,&c[0]);
misty1_encrypt_block(ek_e,&Plaintext[2],&c[2]);

memcpy(input,c,2*sizeof(u4));
memcpy(&input[8],&c[2],2*sizeof(u4));

misty1_key_destroy(ek_e);
misty1_key_destroy(ek_d);
memset(Key,0,4 * sizeof(u4));

```

- h) Inside the mysty1_main function the above codes were modified to ensure that the plaintext is properly initialized with the 16 bytes of payload received, for the encryption to happen.
- i) Here, memcpy() is done initially to equate input received as which is char, to the plain text which is unsigned long.

```

memcpy(Plaintext,input,2*sizeof(u4));
memcpy(&Plaintext[2],&input[8],2*sizeof(u4));

```

- j) After the calls to misty1_encrypt_block() memcpy() is done to equate the encrypted cipher text back to the input.

```

memcpy(input,c,2*sizeof(u4));
memcpy(&input[8],&c[2],2*sizeof(u4));

```

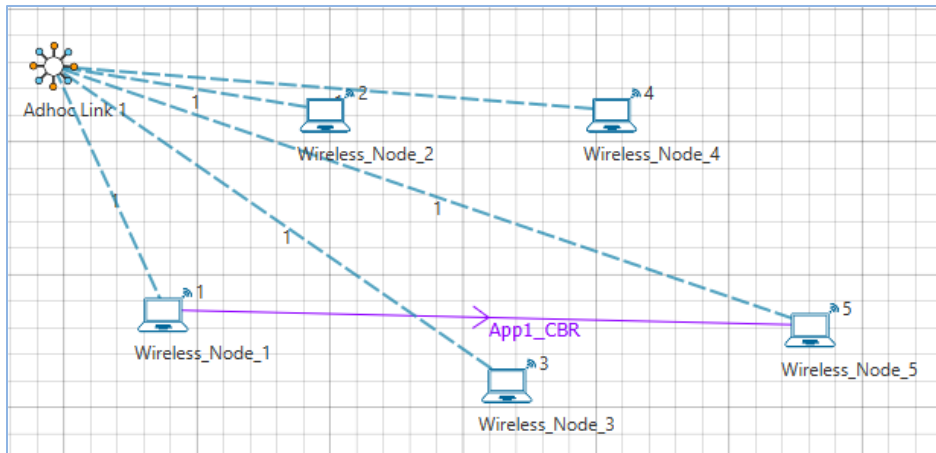
5. Now double click on the application.c file and make a call to mysty_run() function instead of the call to aes256, inside the copy_payload() function as shown below (changes are marked in red):

```

if(info->encryption==Encryption_TEA)
    encryptBlock(real,payload,&key);
else if(info->encryption==Encryption_AES)
{
    misty_run(real,payload);
    //aes256(real,payload);
}
else if(info->encryption==Encryption_DES)
    des(real,payload);

```

6. Rebuild Application project and replace the libApplication.dll file in NetSim bin folder i.e. "C:\Program Files\NetSim Standard\bin".
7. Open Configuration.netsim file from the zip and make sure that AES encryption is selected in the application properties.



8. Also Wireshark option has to be set to either Online or Offline in any of the nodes where AES256 encryption is enabled.
9. Now mysty1 codes will be running instead of AES256.
10. You can see the encrypted payload in Wireshark either during simulation if online is set or after the simulation if offline is set.
11. Setting Wireshark to either online or offline will give you Packet Capture metrics where links to .pcap files are provided. The number of links available depends on the number of nodes in which Wireshark is enabled.

The screenshot shows the Wireshark interface for a packet capture file named 'WIRELESS_NODE_1_1.pcap'. The main window displays a list of captured packets. The selected packet (No. 5) is a UDP packet from source IP 11.1.1.1 to destination IP 11.1.1.5. The packet details pane shows the following structure:

- Logical-Link Control
- Internet Protocol Version 4, Src: 11.1.1.1, Dst: 11.1.1.5
- User Datagram Protocol, Src Port: 82, Dst Port: 36934
- Data (1460 bytes)

The packet bytes pane shows the raw data in hexadecimal and ASCII. A red dashed box highlights the encrypted payload, which appears as a sequence of random-looking characters in the ASCII column.

No.	Time	Source	Destination	Protocol	Length	Info
4	5.004794		06:0c:d0:dc:53:64 (- 802.11)	14	Acknowledgement, Flags=.....	
5	5.005509	11.1.1.1	11.1.1.5	UDP	1528	82 → 36934 Len=1460
6	5.007135		90:7f:b3:8d:da:5a (- 802.11)	14	Acknowledgement, Flags=.....	
7	5.020530	11.1.1.1	11.1.1.5	UDP	1528	82 → 36934 Len=1460
8	5.022155		90:7f:b3:8d:da:5a (- 802.11)	14	Acknowledgement, Flags=.....	
9	5.040150	11.1.1.1	11.1.1.5	UDP	1528	82 → 36934 Len=1460
10	5.041775		90:7f:b3:8d:da:5a (- 802.11)	14	Acknowledgement, Flags=.....	
11	5.060250	11.1.1.1	11.1.1.5	UDP	1528	82 → 36934 Len=1460
12	5.061875		90:7f:b3:8d:da:5a (- 802.11)	14	Acknowledgement, Flags=.....	
13	5.080150	11.1.1.1	11.1.1.5	UDP	1528	82 → 36934 Len=1460
14	5.081775		90:7f:b3:8d:da:5a (- 802.11)	14	Acknowledgement, Flags=.....	