# **Implementing a new Crypto Algorithm – MISTY1**

**Software:** NetSim Standard v13.3, Visual Studio 2022, Wireshark

**Project Download Link:**

Follow the instructions specified in the following link to download and setup the Project in NetSim:

<https://support.tetcos.com/en/support/solutions/articles/14000128666-downloading-and-setting-up-netsim-file-exchange-projects>

# **Introduction**

MISTY1 is a secret-key cryptosystem that uses a block cipher with a 128-bit key and a 64-bit block. It has a variable number of rounds, typically between 8 and 16, depending on the desired level of security.MISTY1 was developed by Mitsuru Matsui and is widely used in various applications, including secure communication, digital signatures, and authentication protocols.

Here in NETSIM we have created simple project of implementing a new crypto algorithm using MISTY1.

# **Example**

1. The **MISTY\_ENCRYPTION\_WorkSpace** comes with a sample network configuration that are already saved. To open this example, go to Your work in the Home screen of NetSim and click on the **MISTY\_ENCRYPTION\_Example** from the list of experiments.
2. The Network Scenario mainly consist of 2 Wired Nodes and 1 Router.

Chart, line chart

Description automatically generated

**Figure 1**: Network Scenario

1. Set Encryprtion Parameter as AES in the Configure Application settings

Graphical user interface, application

Description automatically generated

**Figure 2**: Application Configuration window

1. Make sure to keep the Wireshark Online in both Wired nodes(If WireShark is set to online wireshark window will open during the runtime)
2. To set Wireshark online or offline, Click on Wirednode > General >Wireshark-Capture

Graphical user interface, application

Description automatically generated

**Figure 3**: Wireshark enable window

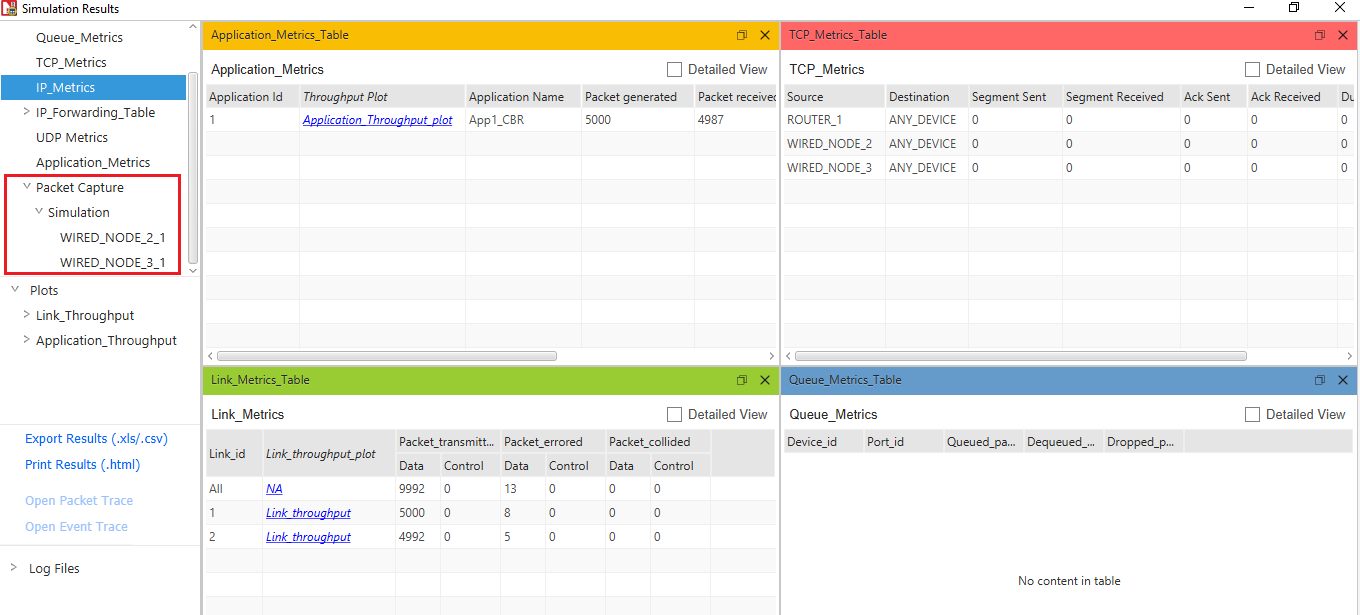
1. Run Simulation for 100 seconds.
2. Now misty1 codes will be running instead of AES256.

# **Results and discussion**

After simulation Open Metrics window and observe the result.

If Wireshark option is set to offline, then the capture files can be accessed from the results dashboard.

Click on Packet capture > Simulation



**Figure 4:** NetSim result dashboard.

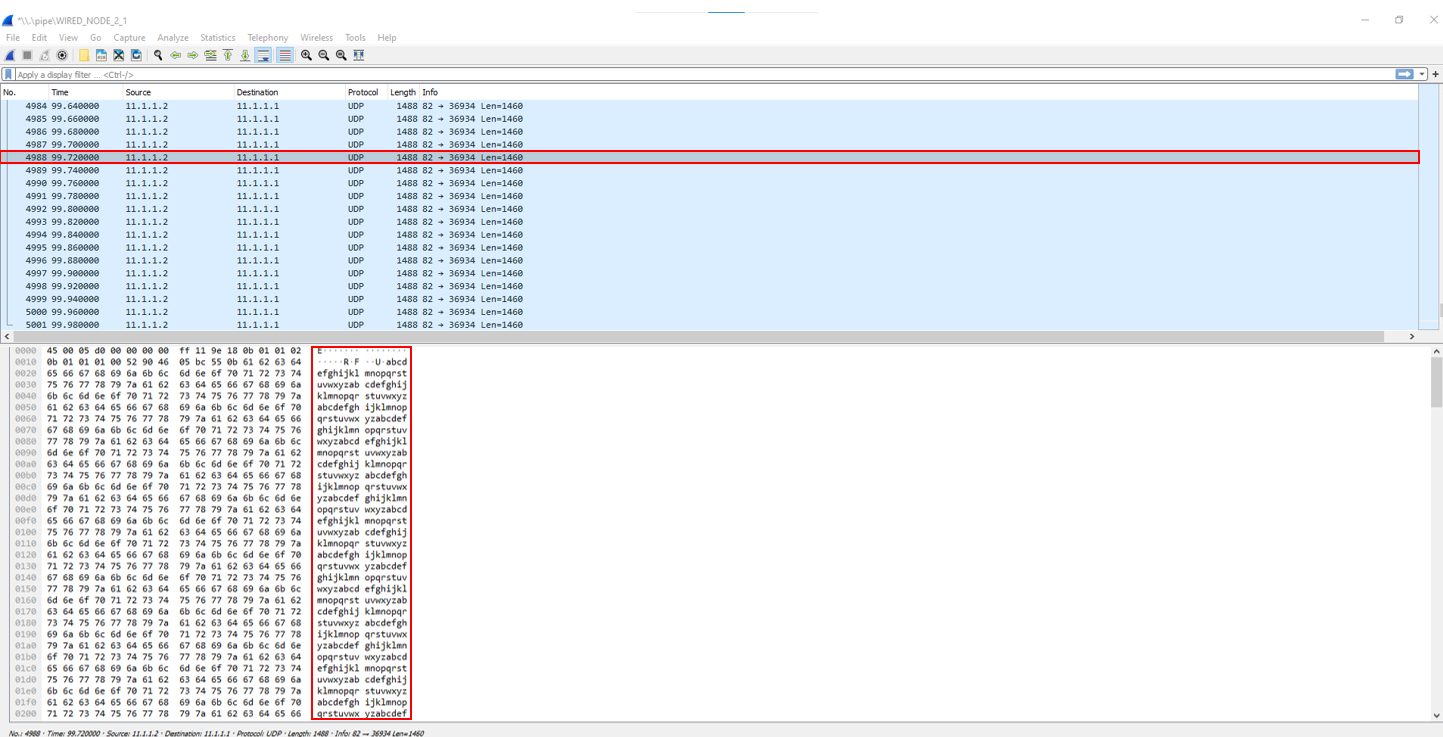
You can see the encrypted payload by double clicking on any packet in wireshark window

Table

Description automatically generated

**Figure 5:** Wireshark window with data encrypted

If you want to see the Normal payload before encrypted by Misty encryption code in wireshark window as shown in below screenshot, It is Possible By setting encryprtion parameter as None in the Configure Application settings



**Figure 6:** Wireshark window before encryption

# **Appendix: NetSim source code modifications**

**Added code in misty\_run.c, within Application project**

#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 accepts a 16-byte plaintext as input. Here the variable str contains the packet payload and len corresponds to the size of payload in bytes.

**Added code in misty1.c, within Application project**

1. Addition of #include<application.h> and #define uint8 unsigned char to the beginning of the misty1.c file

#include <stdlib.h>

#include <string.h>

#include "application.h"

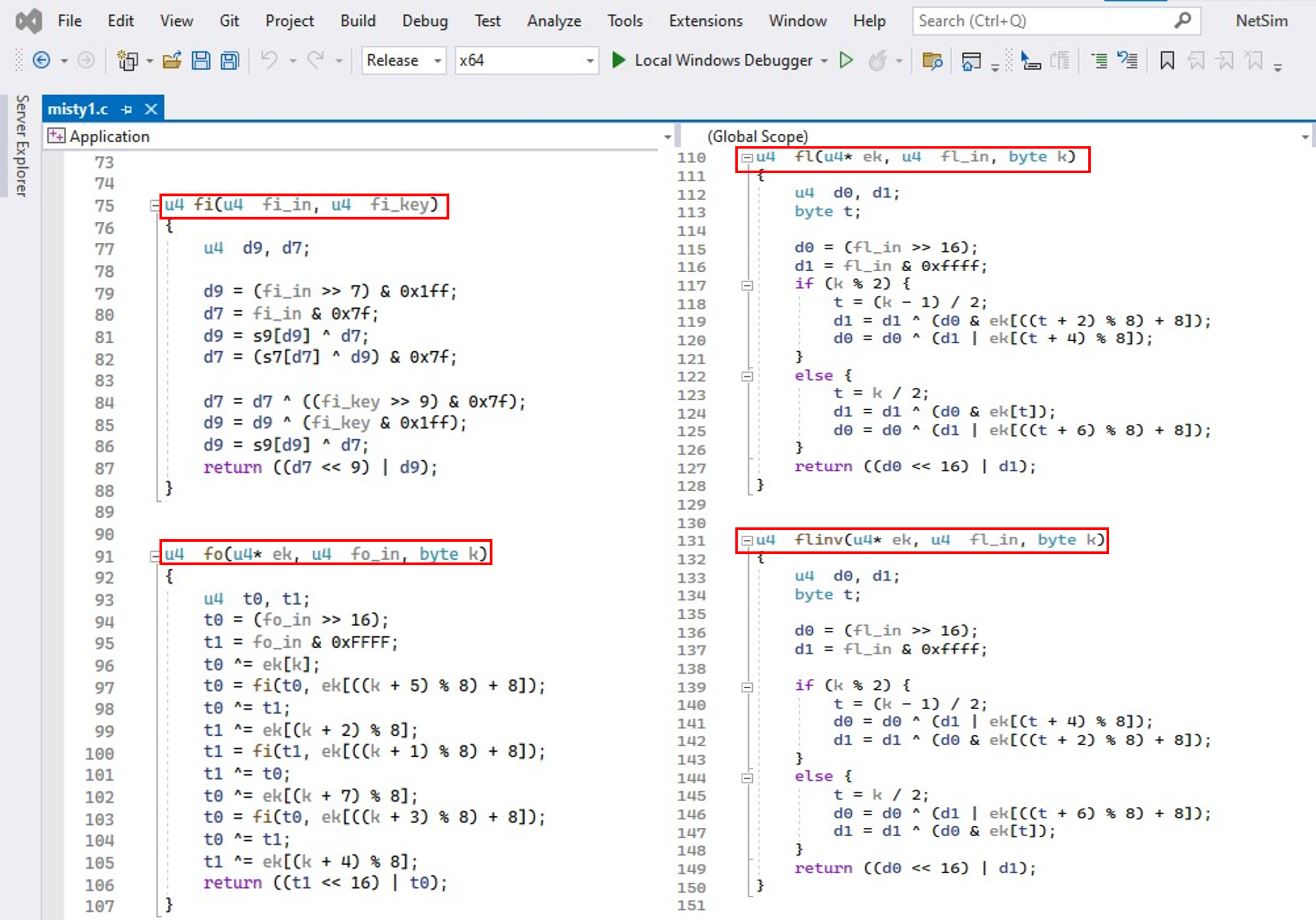
typedef unsigned long u4;

typedef unsigned char byte;

#define MISTY1\_KEYSIZE 32

#define uint8 unsigned char

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



1. Now go to the main() function in the file and check the line #ifdef TESTMAIN was removed or commented before the main() function and also check the associated #endif at the end of the main() function.
2. main() function was renamed to unsigned char\* misty1\_main(uint8\* input)

//#ifdef TESTMAIN

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[4];

// u4 Ciphertext[]= { 0x8b1da5f5, 0x6ab3d07c, 0x04b68240, 0xb13be95d};

u4 ek\_e[MISTY1\_KEYSIZE], ek\_d[MISTY1\_KEYSIZE];

u4 c[4];

/\* misty1\_keyinit(ek\_e,Key);

misty1\_encrypt\_block(ek\_e,&Plaintext[0],&c[0]);

misty1\_encrypt\_block(ek\_e,&Plaintext[2],&c[2]);

if (!memcmp(c,Ciphertext,4 \* sizeof(u4))) {

printf("Encryption OK\n");

}

else {

printf("Encryption failed[0x%08lx 0x%08lx 0x%08lx 0x%08lx]\n",

c[0],c[1],c[2],c[3]);

exit(1);

}

misty1\_keyinit(ek\_d,Key);

if (memcmp(ek\_e,ek\_d,MISTY1\_KEYSIZE\*sizeof(u4))) {

printf("Internal Error keysch is wrong\n");

exit(1);

}

misty1\_decrypt\_block(ek\_d,&Ciphertext[0],&c[0]);

misty1\_decrypt\_block(ek\_d,&Ciphertext[2],&c[2]);

if (!memcmp(c,Plaintext,4 \* sizeof(u4))) {

printf("Decryption OK\n");

}

else {

printf("Decryption failed[0x%08lx 0x%08lx 0x%08lx 0x%08lx]\n",

c[0],c[1],c[2],c[3]);

exit(1);

}

\*/

1. Commented the declaration of Cipher text, 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];

1. Now check the commented lines starting from misty1\_keyinit() to misty1\_key\_destroy() as shown below:

/\* misty1\_keyinit(ek\_e,Key);

misty1\_encrypt\_block(ek\_e,&Plaintext[0],&c[0]);

misty1\_encrypt\_block(ek\_e,&Plaintext[2],&c[2]);

if (!memcmp(c,Ciphertext,4 \* sizeof(u4))) {

printf("Encryption OK\n");

}

else {

printf("Encryption failed[0x%08lx 0x%08lx 0x%08lx 0x%08lx]\n",

c[0],c[1],c[2],c[3]);

exit(1);

}

misty1\_keyinit(ek\_d,Key);

if (memcmp(ek\_e,ek\_d,MISTY1\_KEYSIZE\*sizeof(u4))) {

printf("Internal Error keysch is wrong\n");

exit(1);

}

misty1\_decrypt\_block(ek\_d,&Ciphertext[0],&c[0]);

misty1\_decrypt\_block(ek\_d,&Ciphertext[2],&c[2]);

if (!memcmp(c,Plaintext,4 \* sizeof(u4))) {

printf("Decryption OK\n");

}

else {

printf("Decryption failed[0x%08lx 0x%08lx 0x%08lx 0x%08lx]\n",

c[0],c[1],c[2],c[3]);

exit(1);

}

\*/

1. 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));

misty1\_key\_destroy(ek\_e);

misty1\_key\_destroy(ek\_d);

memset(Key, 0, 4 \* sizeof(u4));

1. Inside the misty1\_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
2. 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));

1. 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));

1. Now double click on the application.c file and make a call to misty\_run() function instead of the call to aes256, inside the copy\_payload() function.

void copy\_payload(UINT8 real[],NetSim\_PACKET\* packet,unsigned int\* payload, ptrAPPLICATION\_INFO info)

{

u\_short i;

uint32\_t key = 16;

if (payload)

{

for (i = 0; i < \*payload; i++)

{

if (info->encryption == Encryption\_XOR)

real[i] = xor\_encrypt('a' + i % 26, 16);

else

real[i] = 'a' + i % 26;

}

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);

}

}