Implementing a new Crypto Algorithm – MISTY1

Software: NetSim Standard v14.4, Visual Studio 2022, Wireshark

Project Download Link:

https://github.com/NetSim-TETCOS/Misty-Encryption-v14.4/archive/refs/heads/main.zip

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

1 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 a simple project of implementing a new crypto algorithm using MISTY1.

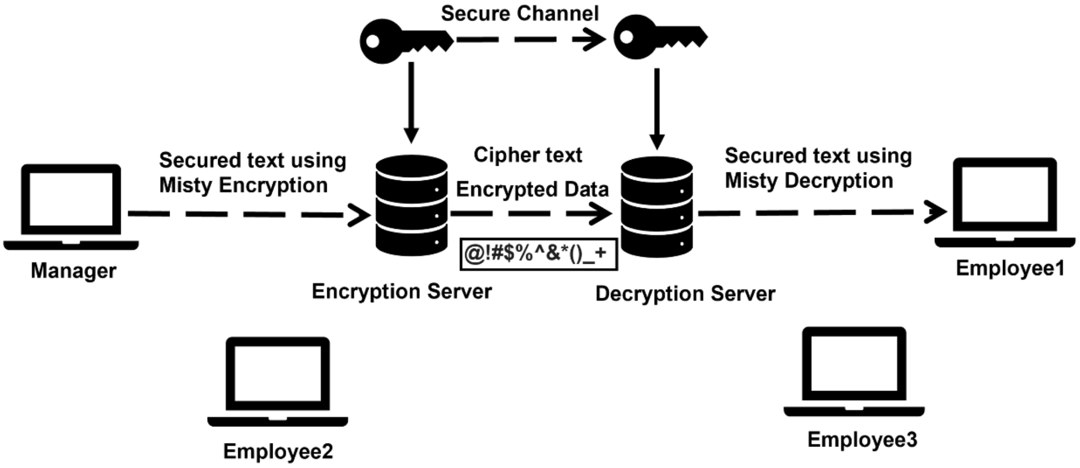


Figure 1: Misty Encryption scenario

2 Misty Encryption Overview:

The MISTY encryption algorithm encrypts a message, which is then re-encrypted with Employee1's public key, ensuring only Employee1 can decrypt it with their private key. This process secures the message, restricting access to others without the private key.

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 consists of 2 Wired Nodes and 1 Router.

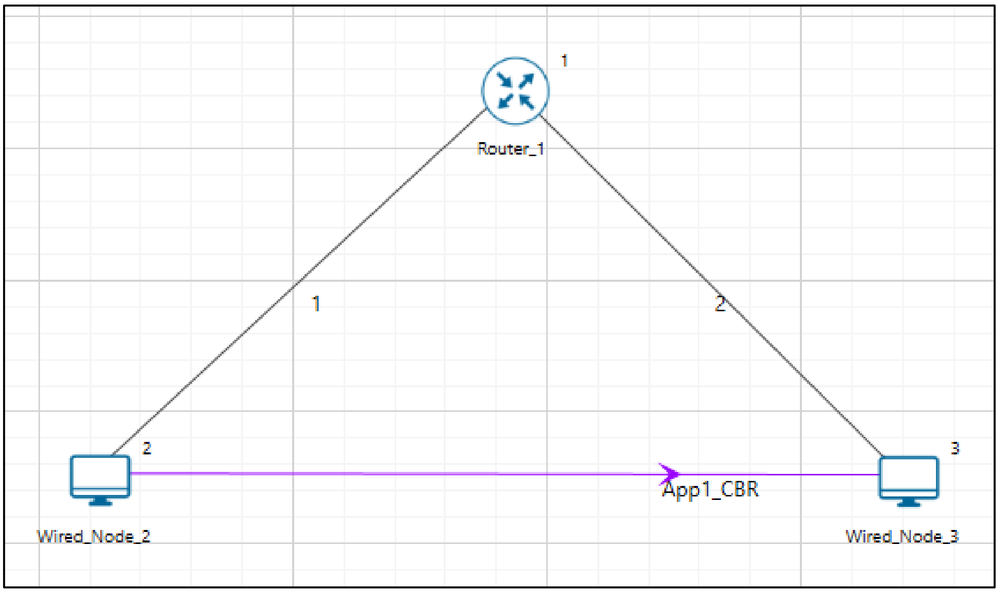
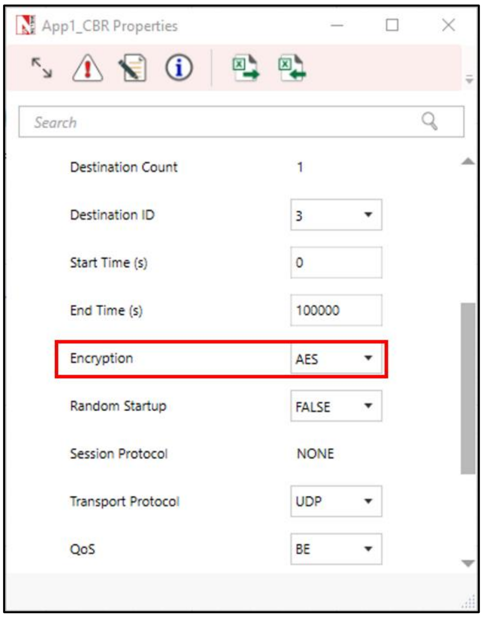


Figure 2: Network Scenario

3. Set Encryption Parameter as AES in the Application Properties.



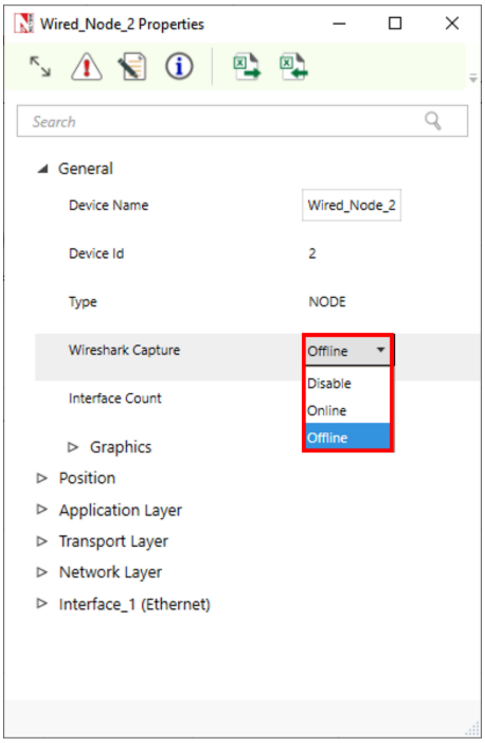
Destination Count	1
Destination ID	3
Start Time (s)	0
End Time (s)	100000
Encryption	AES
Random Startup	FALSE
Session Protocol	NONE
Transport Protocol	UDP
QoS	BE

Figure 3: Application Properties Window

4. Make sure to keep the Wireshark Online in both Wired nodes. To set Wireshark online or offline, Click on Wirednode > General > Wireshark-Capture
   - Online- Initiates a live interactive packet capture, displaying packets while running the

simulation.

- Offline- Initiates silent packet capture and generates a pcap file which can be opened using Wireshark post-simulation in the Result Dashboard Window.



Wired_Node_2 Properties

General

Device Name: Wired_Node_2

Device Id: 2

Type: NODE

Wireshark Capture: Offline

Interface Count

Figure 4: Wireshark enable window

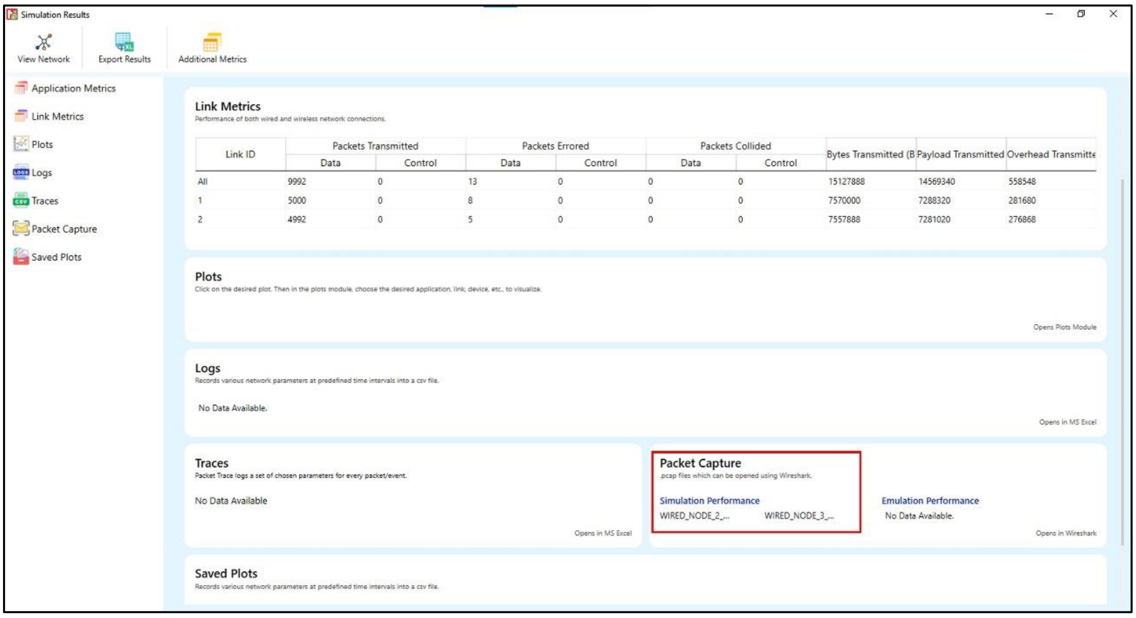
5. Run Simulation for 100 seconds.

3 Results and discussion

After simulation, observe the Captured Packets in the already opened Wireshark window.

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

In result dashboard scroll down to Packet capture > Simulation Performance



Link Metrics

Performance of both wired and wireless network connections.

Link ID	Packets Transmitted		Packets Errored		Packets Collided		Bytes Transmitted (B	Payload Transmitted	Overhead Transmitte
	Data	Control	Data	Control	Data	Control			
All	9992	0	13	0	0	0	15127888	14569340	558548
1	5000	0	8	0	0	0	7570000	7288320	281680
2	4992	0	5	0	0	0	7557888	7281020	276868

Plots

Click on the desired plot. Then in the plots module, choose the desired application, link, device, etc., to visualize.

Opens Plots Module

Logs

Records various network parameters at predefined time intervals into a csv file.

No Data Available.

Opens in MS Excel

Traces

Packet Trace logs a set of chosen parameters for every packet/event.

No Data Available

Opens in MS Excel

Packet Capture

.pcap files which can be opened using Wireshark.

Simulation Performance

WIRED_NODE_2_... WIRED_NODE_3_...

Emulation Performance

No Data Available.

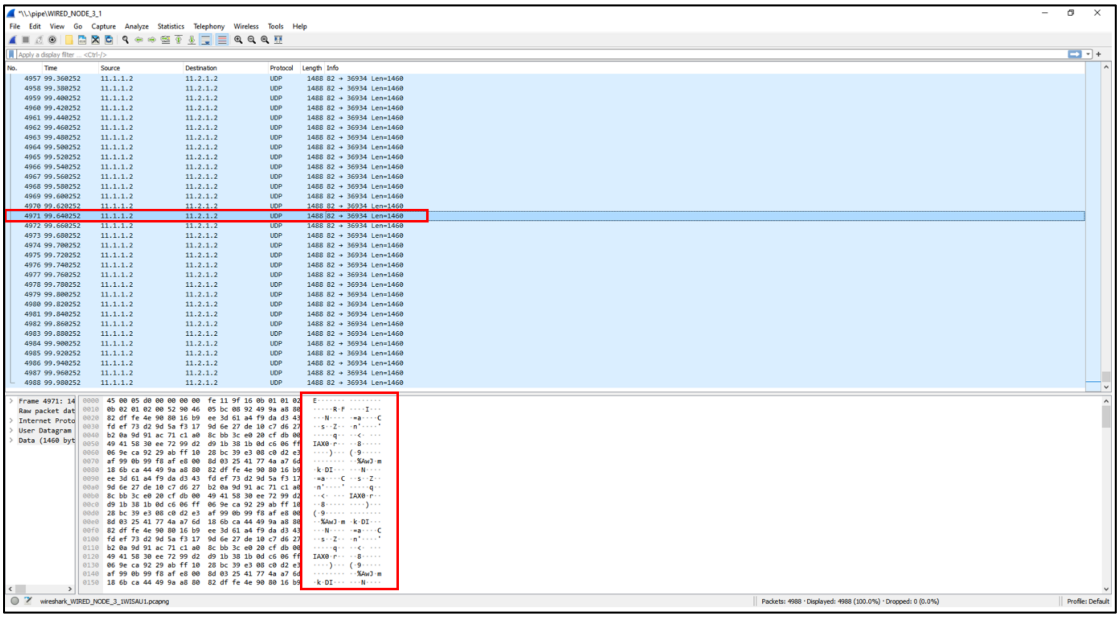
Opens in Wireshark

Saved Plots

Records various network parameters at predefined time intervals into a csv file.

Figure 5: NetSim result dashboard.

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


*\\.\pipe\WIRED_NODE_3_1
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No.  Time       Source    Destination Protocol Length Info
4957 99.360252  11.1.1.2  11.2.1.2    UDP      1488   82 → 36934 Len=1460
4958 99.380252  11.1.1.2  11.2.1.2    UDP      1488   82 → 36934 Len=1460
4959 99.400252  11.1.1.2  11.2.1.2    UDP      1488   82 → 36934 Len=1460
4960 99.420252  11.1.1.2  11.2.1.2    UDP      1488   82 → 36934 Len=1460
4961 99.440252  11.1.1.2  11.2.1.2    UDP      1488   82 → 36934 Len=1460
4962 99.460252  11.1.1.2  11.2.1.2    UDP      1488   82 → 36934 Len=1460
4963 99.480252  11.1.1.2  11.2.1.2    UDP      1488   82 → 36934 Len=1460
4964 99.500252  11.1.1.2  11.2.1.2    UDP      1488   82 → 36934 Len=1460
4965 99.520252  11.1.1.2  11.2.1.2    UDP      1488   82 → 36934 Len=1460
4966 99.540252  11.1.1.2  11.2.1.2    UDP      1488   82 → 36934 Len=1460
4967 99.560252  11.1.1.2  11.2.1.2    UDP      1488   82 → 36934 Len=1460
4968 99.580252  11.1.1.2  11.2.1.2    UDP      1488   82 → 36934 Len=1460
4969 99.600252  11.1.1.2  11.2.1.2    UDP      1488   82 → 36934 Len=1460
4970 99.620252  11.1.1.2  11.2.1.2    UDP      1488   82 → 36934 Len=1460
4971 99.640252  11.1.1.2  11.2.1.2    UDP      1488   82 → 36934 Len=1460
4972 99.660252  11.1.1.2  11.2.1.2    UDP      1488   82 → 36934 Len=1460
4973 99.680252  11.1.1.2  11.2.1.2    UDP      1488   82 → 36934 Len=1460
4974 99.700252  11.1.1.2  11.2.1.2    UDP      1488   82 → 36934 Len=1460
4975 99.720252  11.1.1.2  11.2.1.2    UDP      1488   82 → 36934 Len=1460
4976 99.740252  11.1.1.2  11.2.1.2    UDP      1488   82 → 36934 Len=1460
4977 99.760252  11.1.1.2  11.2.1.2    UDP      1488   82 → 36934 Len=1460
4978 99.780252  11.1.1.2  11.2.1.2    UDP      1488   82 → 36934 Len=1460
4979 99.800252  11.1.1.2  11.2.1.2    UDP      1488   82 → 36934 Len=1460
4980 99.820252  11.1.1.2  11.2.1.2    UDP      1488   82 → 36934 Len=1460
4981 99.840252  11.1.1.2  11.2.1.2    UDP      1488   82 → 36934 Len=1460
4982 99.860252  11.1.1.2  11.2.1.2    UDP      1488   82 → 36934 Len=1460
4983 99.880252  11.1.1.2  11.2.1.2    UDP      1488   82 → 36934 Len=1460
4984 99.900252  11.1.1.2  11.2.1.2    UDP      1488   82 → 36934 Len=1460
4985 99.920252  11.1.1.2  11.2.1.2    UDP      1488   82 → 36934 Len=1460
4986 99.940252  11.1.1.2  11.2.1.2    UDP      1488   82 → 36934 Len=1460
4987 99.960252  11.1.1.2  11.2.1.2    UDP      1488   82 → 36934 Len=1460
4988 99.980252  11.1.1.2  11.2.1.2    UDP      1488   82 → 36934 Len=1460
```

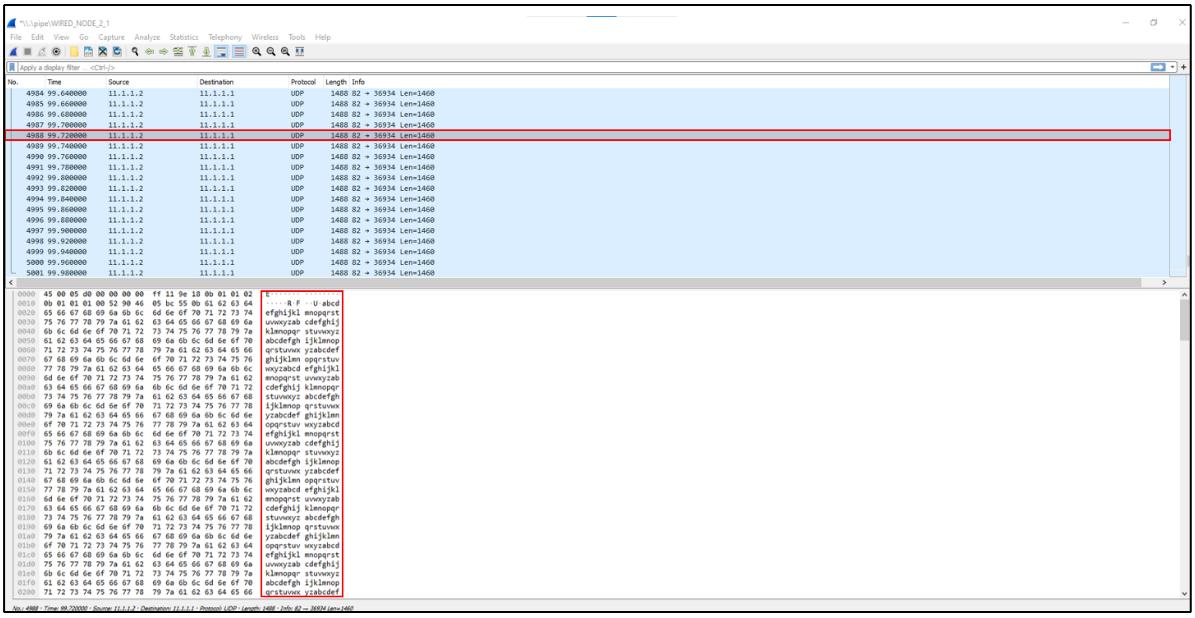
```
> Frame 4971: 14
  Raw packet dat
> Internet Proto
> User Datagram
> Data (1460 byt
```

```
0000  45 00 05 d0 00 00 00 00  fe 11 9f 16 0b 01 01 02   E·······  ········
0010  0b 02 01 02 00 52 90 46  05 bc 08 92 49 9a a8 80   ·····R·F  ····I···
0020  82 df fe 4e 90 80 16 b9  ee 3d 61 a4 f9 da d3 43   ···N····  ·=a····C
0030  fd ef 73 d2 9d 5a f3 17  9d 6e 27 de 10 c7 d6 27   ··s··Z··  ·n'····'
0040  b2 0a 9d 91 ac 71 c1 a0  8c bb 3c e0 20 cf db 00   ·····q··  ··<· ···
0050  49 41 58 30 ee 72 99 d2  d9 1b 38 1b 0d c6 06 ff   IAX0·r··  ··8·····
0060  06 9e ca 92 29 ab ff 10  28 bc 39 e3 08 c0 d2 e3   ····)···  (·9·····
0070  af 99 0b 99 f8 af e8 00  8d 03 25 41 77 4a a7 6d   ········  ··%AwJ·m
0080  18 6b ca 44 49 9a a8 80  82 df fe 4e 90 80 16 b9   ·k·DI···  ···N····
0090  ee 3d 61 a4 f9 da d3 43  fd ef 73 d2 9d 5a f3 17   ·=a····C  ··s··Z··
00a0  9d 6e 27 de 10 c7 d6 27  b2 0a 9d 91 ac 71 c1 a0   ·n'····'  ·····q··
00b0  8c bb 3c e0 20 cf db 00  49 41 58 30 ee 72 99 d2   ··<· ···  IAX0·r··
00c0  d9 1b 38 1b 0d c6 06 ff  06 9e ca 92 29 ab ff 10   ··8·····  ····)···
00d0  28 bc 39 e3 08 c0 d2 e3  af 99 0b 99 f8 af e8 00   (·9·····  ········
00e0  8d 03 25 41 77 4a a7 6d  18 6b ca 44 49 9a a8 80   ··%AwJ·m  ·k·DI···
00f0  82 df fe 4e 90 80 16 b9  ee 3d 61 a4 f9 da d3 43   ···N····  ·=a····C
0100  fd ef 73 d2 9d 5a f3 17  9d 6e 27 de 10 c7 d6 27   ··s··Z··  ·n'····'
0110  b2 0a 9d 91 ac 71 c1 a0  8c bb 3c e0 20 cf db 00   ·····q··  ··<· ···
0120  49 41 58 30 ee 72 99 d2  d9 1b 38 1b 0d c6 06 ff   IAX0·r··  ··8·····
0130  06 9e ca 92 29 ab ff 10  28 bc 39 e3 08 c0 d2 e3   ····)···  (·9·····
0140  af 99 0b 99 f8 af e8 00  8d 03 25 41 77 4a a7 6d   ········  ··%AwJ·m
0150  18 6b ca 44 49 9a a8 80  82 df fe 4e 90 80 16 b9   ·k·DI···  ···N····
```

wireshark_WIRED_NODE_3_1WISAU1.pcapng | Packets: 4988 · Displayed: 4988 (100.0%) · Dropped: 0 (0.0%) | Profile: Default

Figure 6: 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 Encryption parameter as None in the Application Properties as shown in Figure 3.



*\\.\pipe\WIRED_NODE_2_1
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No.   Time       Source     Destination  Protocol  Length  Info
4984 99.640000   11.1.1.2   11.1.1.1     UDP       1488 82 → 36934 Len=1460
4985 99.660000   11.1.1.2   11.1.1.1     UDP       1488 82 → 36934 Len=1460
4986 99.680000   11.1.1.2   11.1.1.1     UDP       1488 82 → 36934 Len=1460
4987 99.700000   11.1.1.2   11.1.1.1     UDP       1488 82 → 36934 Len=1460
4988 99.720000   11.1.1.2   11.1.1.1     UDP       1488 82 → 36934 Len=1460
4989 99.740000   11.1.1.2   11.1.1.1     UDP       1488 82 → 36934 Len=1460
4990 99.760000   11.1.1.2   11.1.1.1     UDP       1488 82 → 36934 Len=1460
4991 99.780000   11.1.1.2   11.1.1.1     UDP       1488 82 → 36934 Len=1460
4992 99.800000   11.1.1.2   11.1.1.1     UDP       1488 82 → 36934 Len=1460
4993 99.820000   11.1.1.2   11.1.1.1     UDP       1488 82 → 36934 Len=1460
4994 99.840000   11.1.1.2   11.1.1.1     UDP       1488 82 → 36934 Len=1460
4995 99.860000   11.1.1.2   11.1.1.1     UDP       1488 82 → 36934 Len=1460
4996 99.880000   11.1.1.2   11.1.1.1     UDP       1488 82 → 36934 Len=1460
4997 99.900000   11.1.1.2   11.1.1.1     UDP       1488 82 → 36934 Len=1460
4998 99.920000   11.1.1.2   11.1.1.1     UDP       1488 82 → 36934 Len=1460
4999 99.940000   11.1.1.2   11.1.1.1     UDP       1488 82 → 36934 Len=1460
5000 99.960000   11.1.1.2   11.1.1.1     UDP       1488 82 → 36934 Len=1460
5001 99.980000   11.1.1.2   11.1.1.1     UDP       1488 82 → 36934 Len=1460
```

```
0000  45 00 05 d0 00 00 00 00  ff 11 9e 18 0b 01 01 02   E······· ········
0010  0b 01 01 01 00 52 90 46  05 bc 55 0b 61 62 63 64   ·····R·F ··U·abcd
0020  65 66 67 68 69 6a 6b 6c  6d 6e 6f 70 71 72 73 74   efghijkl mnopqrst
0030  75 76 77 78 79 7a 61 62  63 64 65 66 67 68 69 6a   uvwxyzab cdefghij
0040  6b 6c 6d 6e 6f 70 71 72  73 74 75 76 77 78 79 7a   klmnopqr stuvwxyz
0050  61 62 63 64 65 66 67 68  69 6a 6b 6c 6d 6e 6f 70   abcdefgh ijklmnop
0060  71 72 73 74 75 76 77 78  79 7a 61 62 63 64 65 66   qrstuvwx yzabcdef
0070  67 68 69 6a 6b 6c 6d 6e  6f 70 71 72 73 74 75 76   ghijklmn opqrstuv
0080  77 78 79 7a 61 62 63 64  65 66 67 68 69 6a 6b 6c   wxyzabcd efghijkl
0090  6d 6e 6f 70 71 72 73 74  75 76 77 78 79 7a 61 62   mnopqrst uvwxyzab
00a0  63 64 65 66 67 68 69 6a  6b 6c 6d 6e 6f 70 71 72   cdefghij klmnopqr
00b0  73 74 75 76 77 78 79 7a  61 62 63 64 65 66 67 68   stuvwxyz abcdefgh
00c0  69 6a 6b 6c 6d 6e 6f 70  71 72 73 74 75 76 77 78   ijklmnop qrstuvwx
00d0  79 7a 61 62 63 64 65 66  67 68 69 6a 6b 6c 6d 6e   yzabcdef ghijklmn
00e0  6f 70 71 72 73 74 75 76  77 78 79 7a 61 62 63 64   opqrstuv wxyzabcd
00f0  65 66 67 68 69 6a 6b 6c  6d 6e 6f 70 71 72 73 74   efghijkl mnopqrst
0100  75 76 77 78 79 7a 61 62  63 64 65 66 67 68 69 6a   uvwxyzab cdefghij
0110  6b 6c 6d 6e 6f 70 71 72  73 74 75 76 77 78 79 7a   klmnopqr stuvwxyz
0120  61 62 63 64 65 66 67 68  69 6a 6b 6c 6d 6e 6f 70   abcdefgh ijklmnop
0130  71 72 73 74 75 76 77 78  79 7a 61 62 63 64 65 66   qrstuvwx yzabcdef
0140  67 68 69 6a 6b 6c 6d 6e  6f 70 71 72 73 74 75 76   ghijklmn opqrstuv
0150  77 78 79 7a 61 62 63 64  65 66 67 68 69 6a 6b 6c   wxyzabcd efghijkl
0160  6d 6e 6f 70 71 72 73 74  75 76 77 78 79 7a 61 62   mnopqrst uvwxyzab
0170  63 64 65 66 67 68 69 6a  6b 6c 6d 6e 6f 70 71 72   cdefghij klmnopqr
0180  73 74 75 76 77 78 79 7a  61 62 63 64 65 66 67 68   stuvwxyz abcdefgh
0190  69 6a 6b 6c 6d 6e 6f 70  71 72 73 74 75 76 77 78   ijklmnop qrstuvwx
01a0  79 7a 61 62 63 64 65 66  67 68 69 6a 6b 6c 6d 6e   yzabcdef ghijklmn
01b0  6f 70 71 72 73 74 75 76  77 78 79 7a 61 62 63 64   opqrstuv wxyzabcd
01c0  65 66 67 68 69 6a 6b 6c  6d 6e 6f 70 71 72 73 74   efghijkl mnopqrst
01d0  75 76 77 78 79 7a 61 62  63 64 65 66 67 68 69 6a   uvwxyzab cdefghij
01e0  6b 6c 6d 6e 6f 70 71 72  73 74 75 76 77 78 79 7a   klmnopqr stuvwxyz
01f0  61 62 63 64 65 66 67 68  69 6a 6b 6c 6d 6e 6f 70   abcdefgh ijklmnop
0200  71 72 73 74 75 76 77 78  79 7a 61 62 63 64 65 66   qrstuvwx yzabcdef
```

No.: 4988 · Time: 99.720000 · Source: 11.1.1.2 · Destination: 11.1.1.1 · Protocol: UDP · Length: 1488 · Info: 82 → 36934 Len=1460

Figure 7: Wireshark window without 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);
        }
}
```

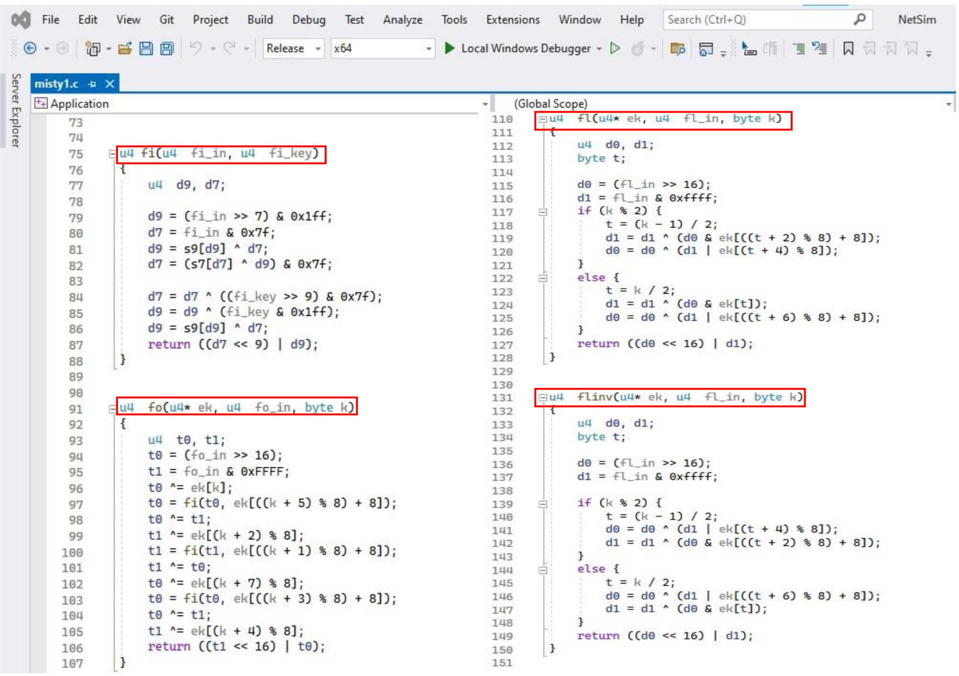
In the misty_run() function inside the misty_run.c file we pass the plain text in parts of 16 bytes each time to get it encrypted. This is done because the crypto algorithm accepts a 16-byte plaintext as input. Here the variable str contains the packet payload and len corresponds to the size of payload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

Added code in misty1.c, within Application project

A. 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
```

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



```
u4 fi(u4  fi_in, u4  fi_key)
{
    u4  d9, d7;

    d9 = (fi_in >> 7) & 0x1ff;
    d7 = fi_in & 0x7f;
    d9 = s9[d9] ^ d7;
    d7 = (s7[d7] ^ d9) & 0x7f;

    d7 = d7 ^ ((fi_key >> 9) & 0x7f);
    d9 = d9 ^ (fi_key & 0x1ff);
    d9 = s9[d9] ^ d7;
    return ((d7 << 9) | d9);
}


u4  fo(u4* ek, u4  fo_in, byte k)
{
    u4  t0, t1;
    t0 = (fo_in >> 16);
    t1 = fo_in & 0xFFFF;
    t0 ^= ek[k];
    t0 = fi(t0, ek[((k + 5) % 8) + 8]);
    t0 ^= t1;
    t1 ^= ek[(k + 2) % 8];
    t1 = fi(t1, ek[((k + 1) % 8) + 8]);
    t1 ^= t0;
    t0 ^= ek[(k + 7) % 8];
    t0 = fi(t0, ek[((k + 3) % 8) + 8]);
    t0 ^= t1;
    t1 ^= ek[(k + 4) % 8];
    return ((t1 << 16) | t0);
}
```

```
u4  fl(u4* ek, u4  fl_in, byte k)
{
    u4  d0, d1;
    byte t;

    d0 = (fl_in >> 16);
    d1 = fl_in & 0xffff;
    if (k % 2) {
        t = (k - 1) / 2;
        d1 = d1 ^ (d0 & ek[((t + 2) % 8) + 8]);
        d0 = d0 ^ (d1 | ek[(t + 4) % 8]);
    }
    else {
        t = k / 2;
        d1 = d1 ^ (d0 & ek[t]);
        d0 = d0 ^ (d1 | ek[((t + 6) % 8) + 8]);
    }
    return ((d0 << 16) | d1);
}


u4  flinv(u4* ek, u4  fl_in, byte k)
{
    u4  d0, d1;
    byte t;

    d0 = (fl_in >> 16);
    d1 = fl_in & 0xffff;

    if (k % 2) {
        t = (k - 1) / 2;
        d0 = d0 ^ (d1 | ek[(t + 4) % 8]);
        d1 = d1 ^ (d0 & ek[((t + 2) % 8) + 8]);
    }
    else {
        t = k / 2;
        d0 = d0 ^ (d1 | ek[((t + 6) % 8) + 8]);
        d1 = d1 ^ (d0 & ek[t]);
    }
    return ((d0 << 16) | d1);
}
```

C. 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.

D. 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);
 }
 */
```

E. 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];
```

F. 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);
}
*/
```

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));

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

H. 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

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));
```

K. 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);
}
}
```