Implementing a new Crypto Algorithm – MISTY1

Software: NetSim Standard v14.1, Visual Studio 2022, Wireshark

Project Download Link:

https://github.com/NetSim-TETCOS/Misty-Encryption-v14.1/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

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.

Real-World Context:
To safeguard confidential communications between managers and employees in a corporate environment, MISTY encryption can be employed, effectively preventing unauthorized access or third employee access to sensitive messages.

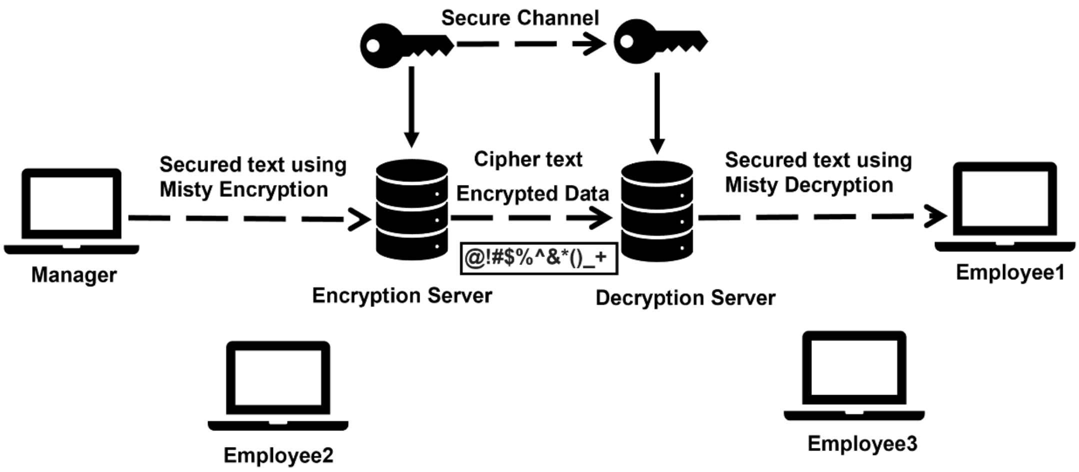


Figure 1:Misty Encryption in Real World

Misty Encryption Overview:

- The manager uses the MISTY encryption algorithm to encrypt the message. This will scramble the message so that it is unreadable to anyone who does not have the decryption key.
- The Manager sends the encrypted message to the encryption server.
- The encryption server re-encrypts the message using the Employee1's public key.
- The encryption server sends the re-encrypted message to the decryption server.
- The decryption server decrypts the message using the Employee1's private key.

- This process ensures that the message is secure and that only Employee1 can read it.
- Other employees in the company or the outside resources cannot access the message because they do not have the employee's 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 consist of 2 Wired Nodes and 1 Router.

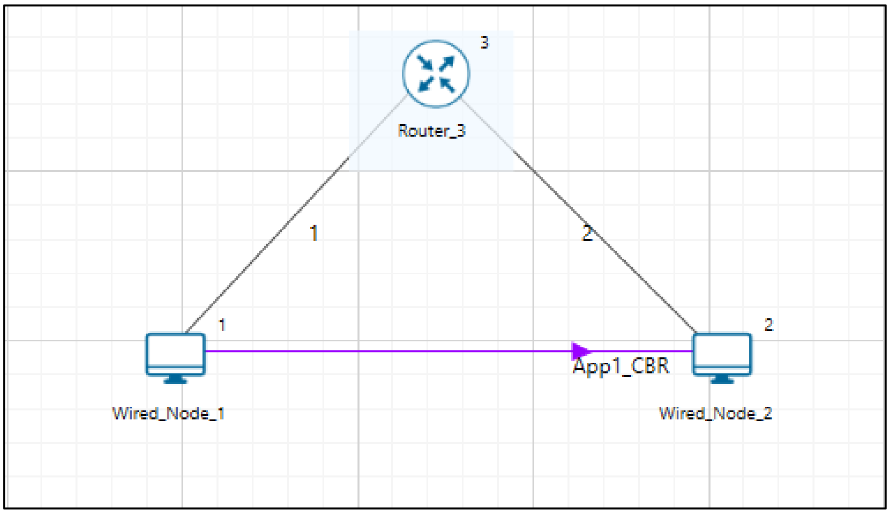


Figure 2: Network Scenario

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

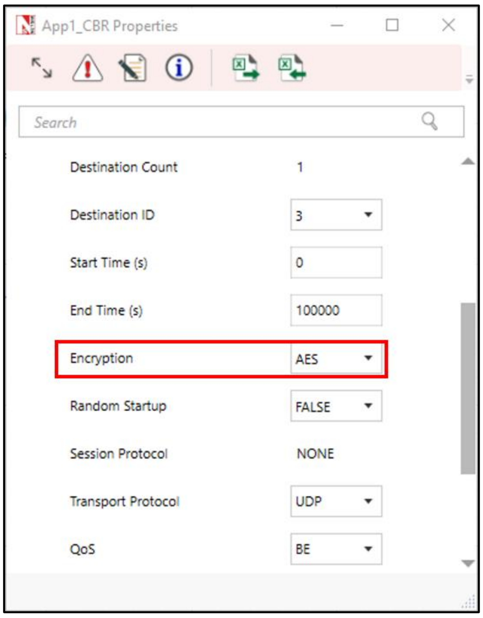


Figure 3: Application Properties Window

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 generate a pcap file which can be opened using Wireshark post-simulation in the Result Dashboard Window.

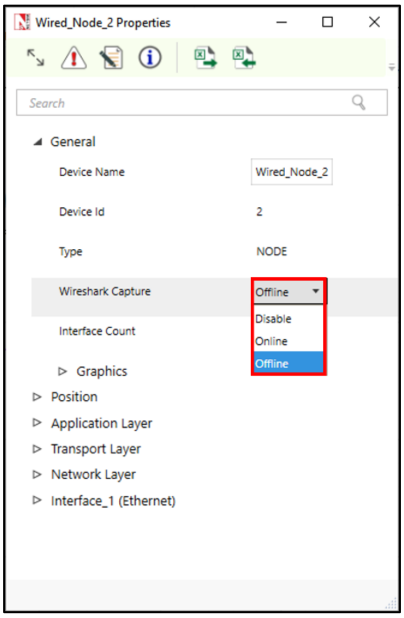


Figure 4: Wireshark enable window

5. Run Simulation for 100 seconds.

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.
Click on Packet capture > Simulation

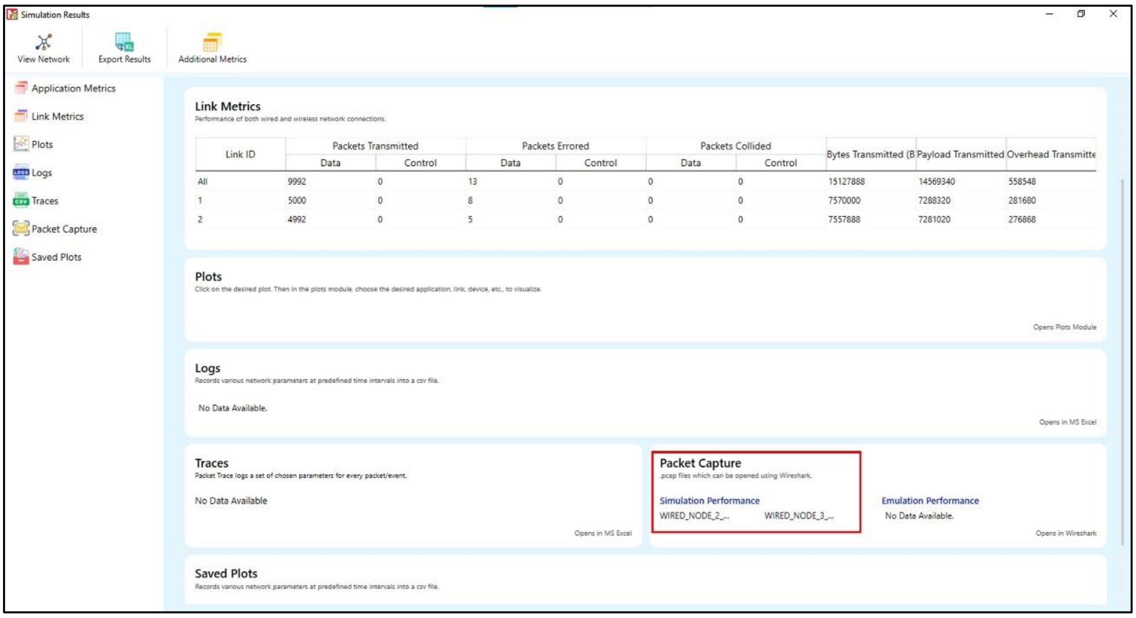


Figure 5: NetSim result dashboard.

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

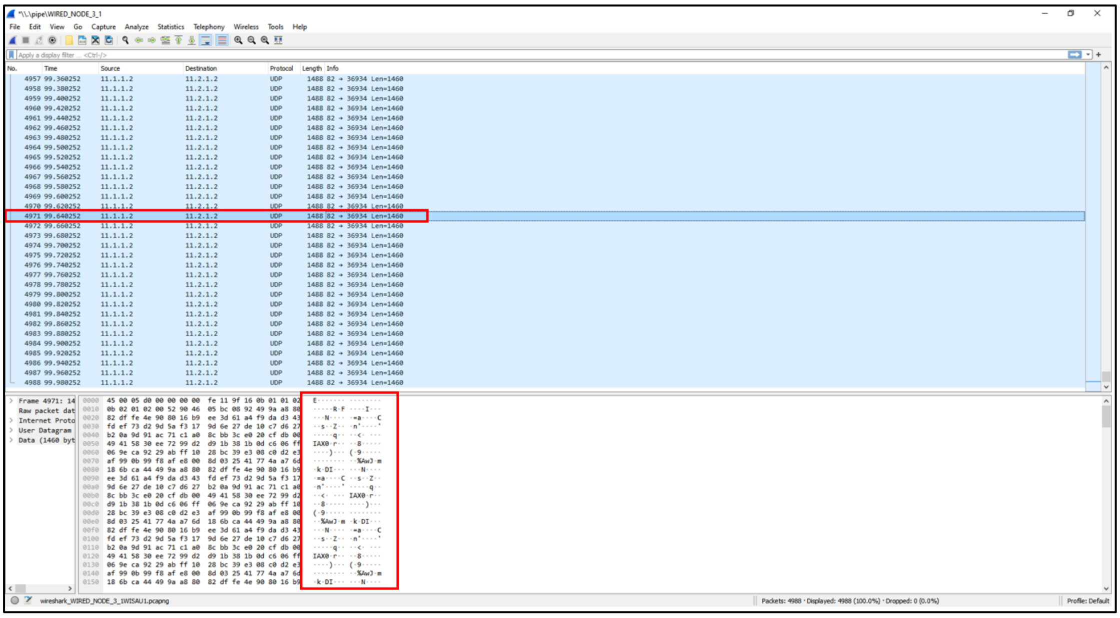


Figure 6: Wireshark window with data encrypted

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.

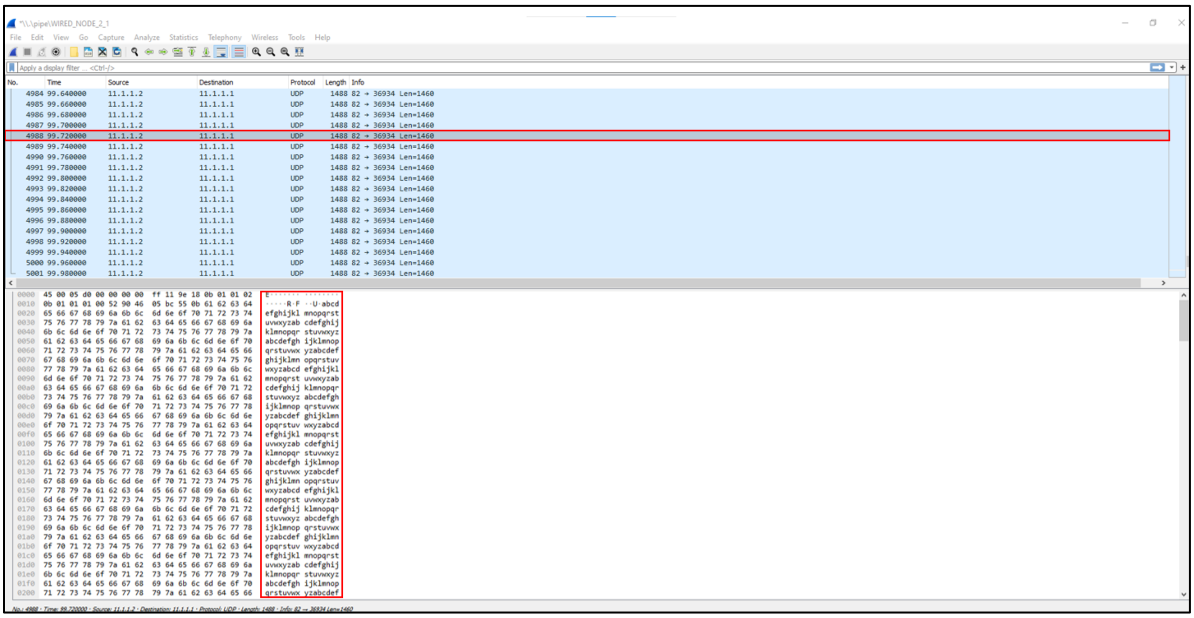


Figure 7: Wireshark window without encryption

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);
	}
}
```

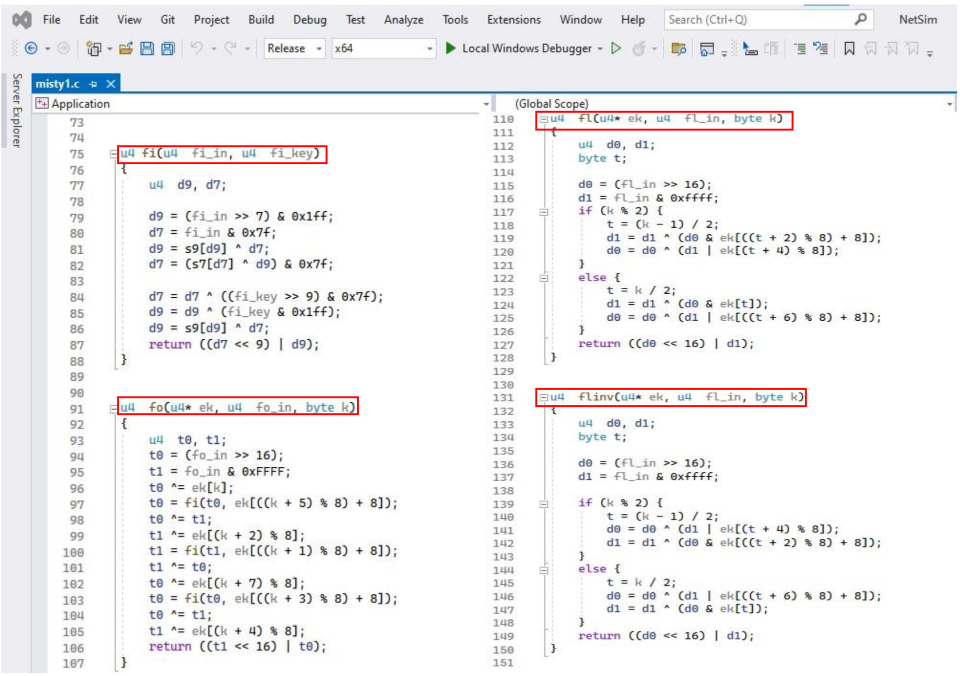
In the misty_run() function inside the misty_run.c file we pass the plain text in parts of 16 bytes each time to get it encrypted. This is done because the crypto algorithm accepts a 16-byte plaintext as input. Here the variable str contains the packet payload and len corresponds to the size of payload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



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);
}
}
```

