

TLS10GS-IP Demo Instruction

Rev1.00 5-Mar-2024

This document describes the instruction to demonstrate the operation of TLS 1.3 Server 10Gbps IP Core (TLS10GS-IP) on ZCU102 Evaluation Board. In the demonstration, TLS10GS-IP are used to establish a secure connection using the Transport Layer Security protocol version 1.3 over TCP by handling TLS1.3 handshake, encrypting and decrypting data transferred between client and server. Additionally, HTTPS is chosen as the application layer to simplify the testing of data transfer between a standard client and the TLS10GSdemo.

This instruction explains the process for users to use a web browser as a client to upload or download data patterns from TLS10GSdemo, obtaining results similar to communication with the provided example node.js server, the use of the “client” application to test transfer speed between a PC and TLS10GSdemo, along with the comparison of test results between two FPGA boards.

1 Environment Setup

To operate TLS10GS-IP demo, please prepare following test environment.

- 1) FPGA development boards (ZCU102 board).
- 2) Test PC with 10 Gigabit Ethernet or connecting with 10 Gigabit Ethernet card.
- 3) 10 Gb Ethernet cable:
 - a) 10 Gb SFP+ Passive Direct Attach Cable (DAC) which has 1-m or less length
 - b) 10 Gb SFP+ Active Optical Cable (AOC)
 - c) 2x10 Gb SFP+ transceiver (10G BASE-R) with optical cable (LC to LC, Multimode)
- 4) Micro USB cable for JTAG connection connecting between ZCU102 board and Test PC.
- 5) Micro USB cable for UART connection connecting between ZCU102 board and Test PC.
- 6) Vivado tool for programming FPGA installed on Test PC.
- 7) Serial console software such as TeraTerm installed on PC. The setting on the console is Baudrate=115200, Data=8-bit, Non-parity and Stop=1.
- 8) Batch file named TLS10GSIPTest.bat” (To download these files, please visit our web site at www.design-gateway.com)

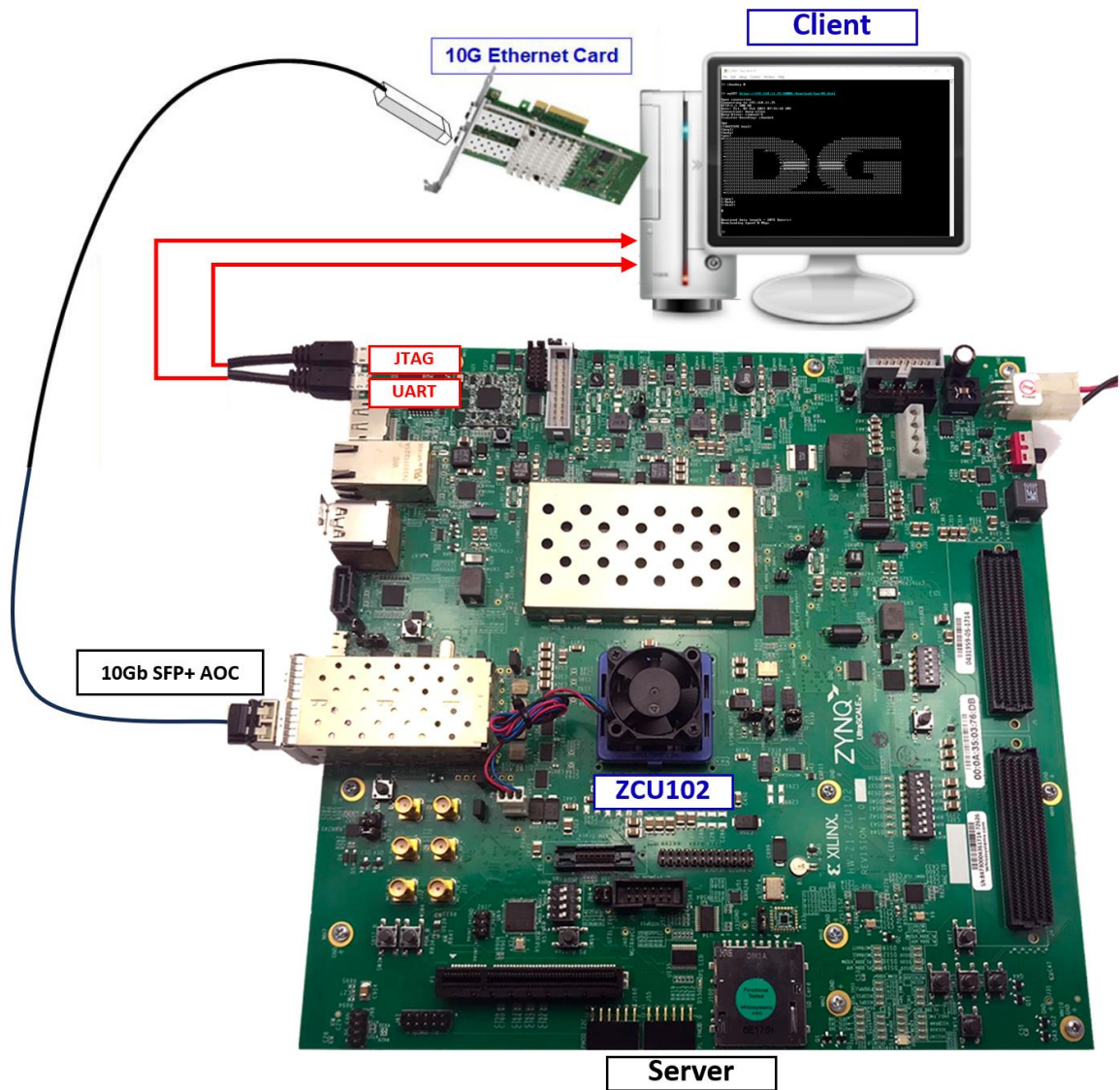


Figure 1-1 TLS10GS-IP demo environment on ZCU102 board

2 PC Setup

Before running demo, please check the network setting on PC. The example of setting 10 Gb Ethernet card is described as follows.

2.1 IP setting

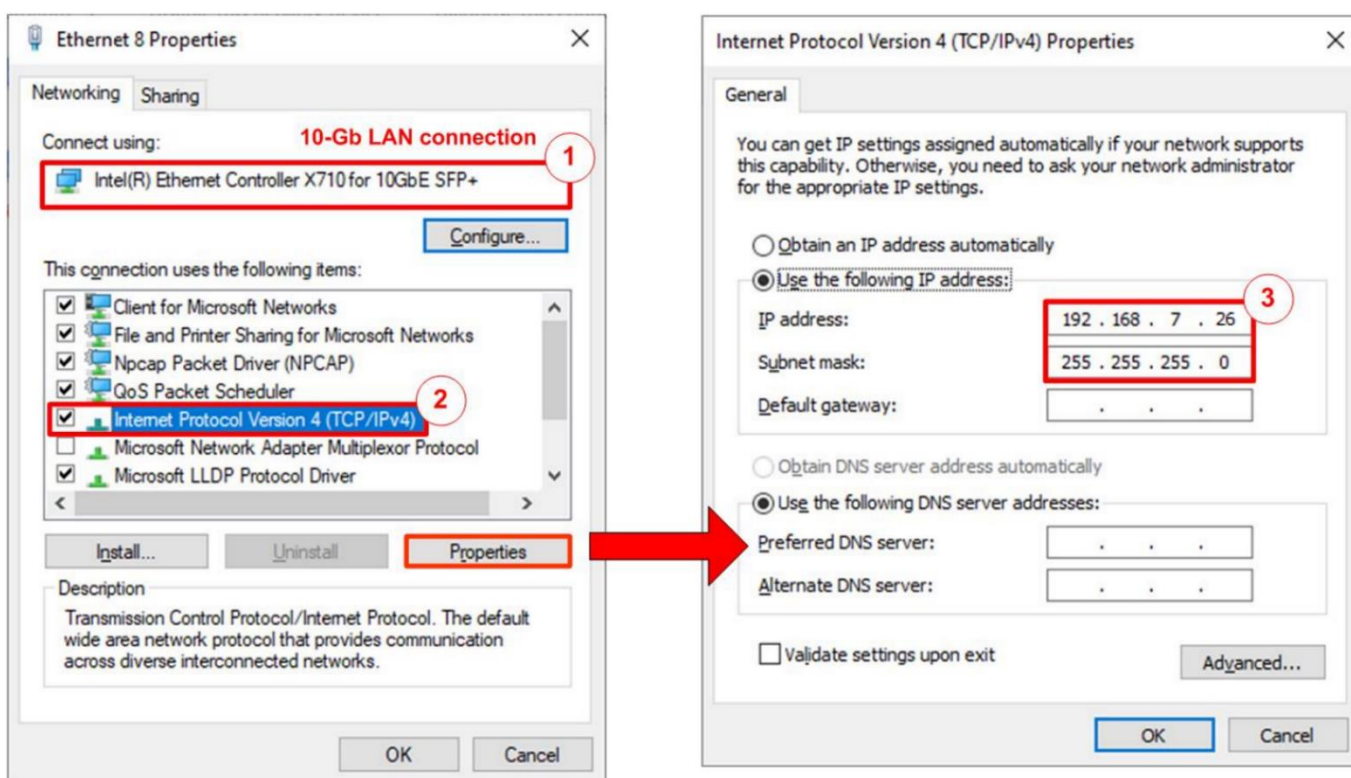


Figure 2-1 Setting IP address for PC

- 1) Open Local Area Connection Properties of 10 Gb connection, as shown in the left window of Figure 2-1.
- 2) Select "TCP/IPv4" and then click Properties.
- 3) Set IP address = 192.168.7.26 and Subnet mask = 255.255.255.0, as shown in the right window of Figure 2-1.

2.2 Speed and duplex setting

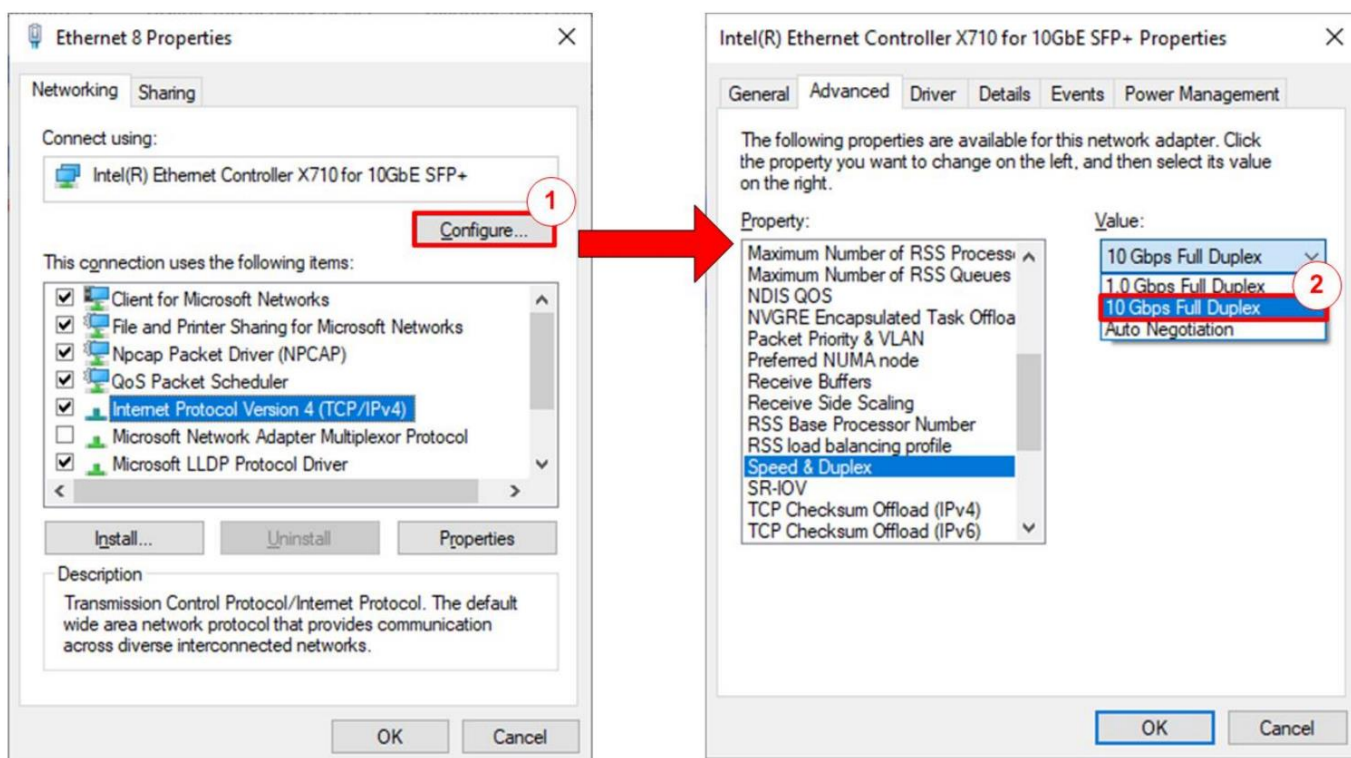


Figure 2-2 Set Link Speed = 10 Gbps

- 1) On Local Area Connection Properties window, click “Configure”, as shown in Figure 2-2.
- 2) On Advanced Tab, select “Speed and Duplex”. Set the value to “10 Gbps Full Duplex” for running 10 Gigabit transfer test, as shown in Figure 2-2.

2.3 Network properties setting

Some of network parameter setting may affect to network performance. The example of network properties setting as follows.

- 1) On “Interrupt Moderation” window, select “Disabled” to disable interrupt moderation which would minimize the latency during transferring data, as shown in Figure 2-3.

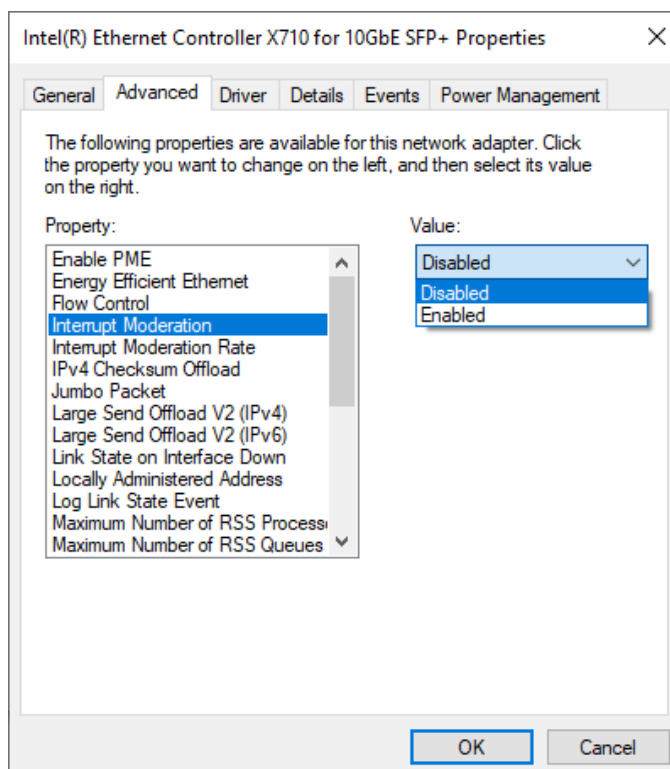


Figure 2-3 Interrupt Moderation

2) On “Interrupt Moderation Rate” window, set value to “OFF”, as shown in Figure 2-4.

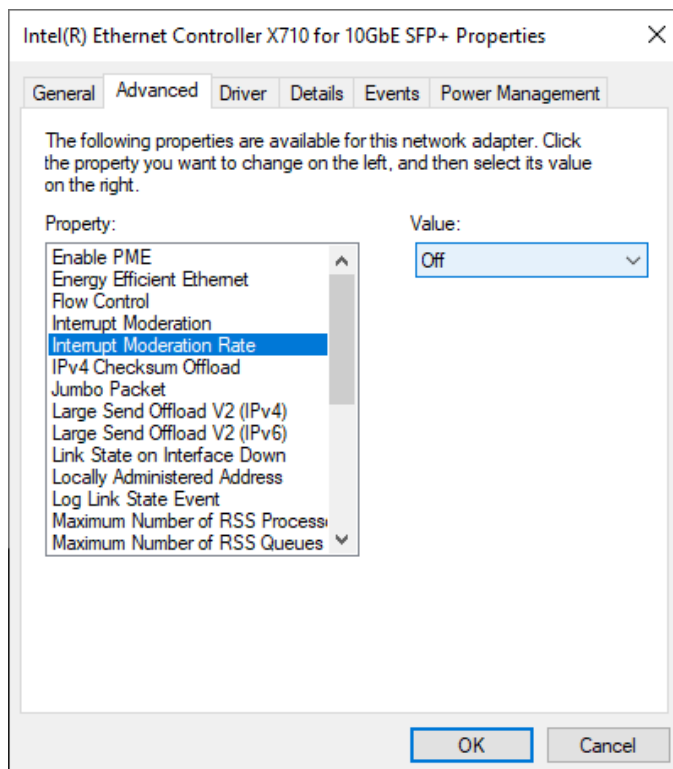


Figure 2-4 Interrupt Moderation Rate

3) On “Jumbo packet” window, set value to “9014 Bytes”, as shown in Figure 2-5.

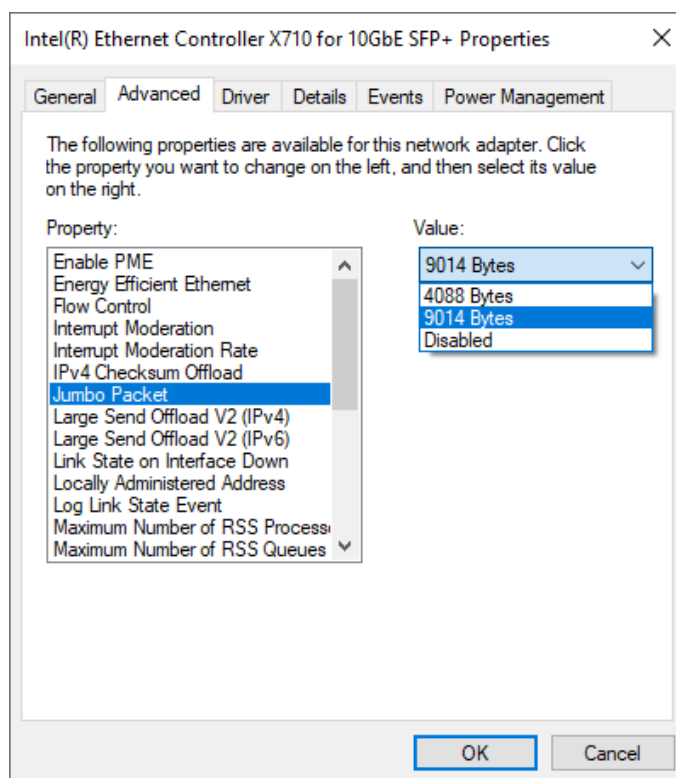


Figure 2-5 Jumbo packet

- 4) On “Receive Buffers” window, set value to the maximum value, as shown in Figure 2-6.

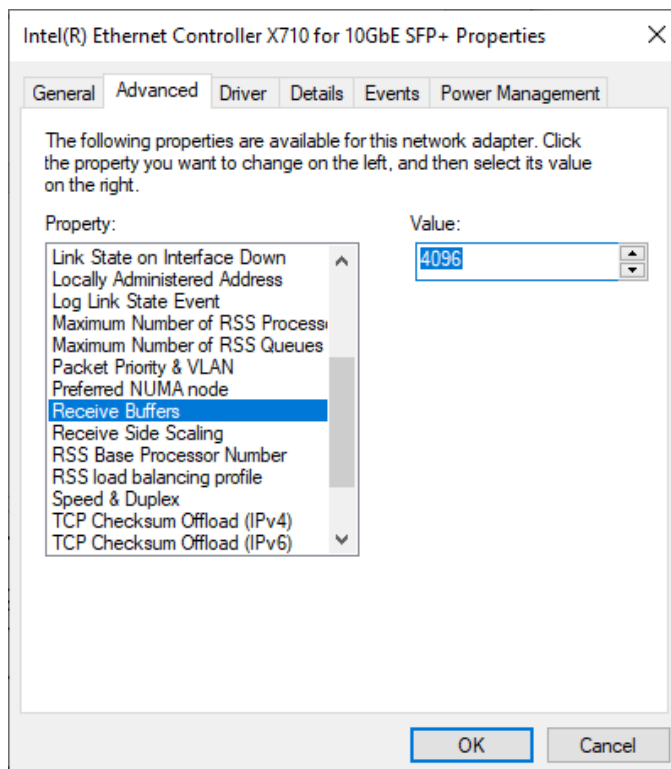


Figure 2-6 Receive Buffers

- 5) On “Transmit Buffers” window, set value to the maximum value, as shown in Figure 2-7.

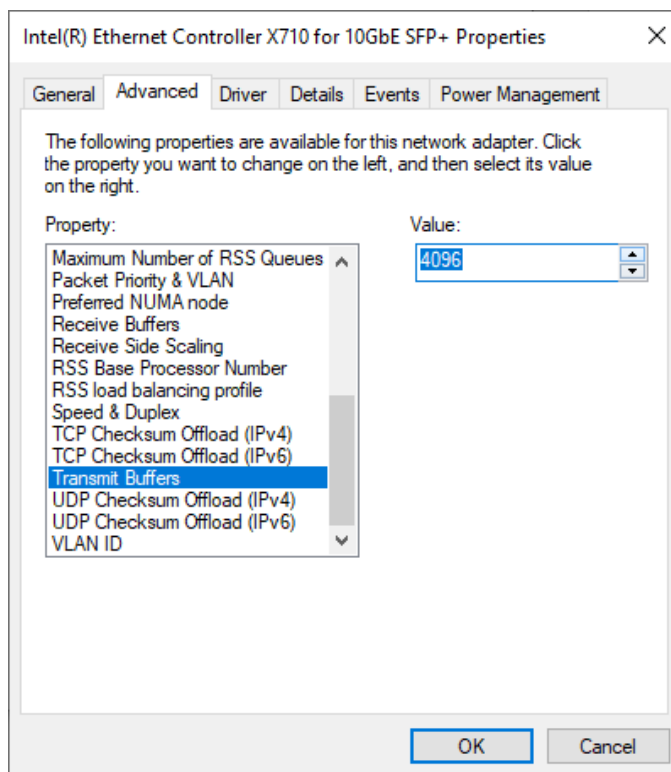


Figure 2-7 Transmit buffers

3 Client web browser

Users can use a web browser for downloading a data pattern from both Node.js server and TLS10GSDemo by GET method and uploading a data pattern to the server via POST method.

For downloading a data pattern, users can input URL in the following format,

`https://ip:port/direction/pattern/length`

| | |
|-----------|---|
| Where ip | represent server's IP address in dot-decimal notation |
| port | represent server's port number |
| direction | represent download or upload |
| pattern | represent data pattern |
| | b1: increasing binary pattern, t1: increasing text pattern, |
| | b0: decreasing binary pattern, t0: decreasing text pattern |
| length | represent data length in byte |

For example, server's IP address is 192.168.7.26, port number is 60001 and the user's URL is `https://192.168.7.26:60001/download/t1/123`. Secure connection is established, the 123-byte increasing text pattern is displayed in the web browser, as shown in Figure 3-1.



Figure 3-1 Decreasing text pattern shown in web browser

Remark

- Our tested web browser is Google Chrome version 116.0.5845.141.
- The RSA certificate used in this demonstration is self-signed, meaning it was not issued by a certification authority (CA). When attempting to access the server with a self-signed certificate, the web browser may generate a certificate unknown error and then terminate the connection. To bypass certificate errors, users can run the Chrome browser from the command prompt with the "ignore-certificate-errors" parameter. The banner "You are using an unsupported command-line flag: -- ignore-certificate-errors. Stability and security will suffer." will appear when Google Chrome has already detected the command-line option, as shown in Figure 3-3.

In case of downloading a binary pattern, a "Save as" dialog window appears. Users can save the file and view the binary data after the download process is complete.

Users can securely upload data through a web browser using uploadPage.html. The HTML page can be opened in the chrome browser by executing the following command in the command prompt: “<path to chrome.exe>” ignore-certificate-errors <path to html page>, as shown in Figure 3-2.

For example, "C:\Program Files\Google\Chrome\Application\chrome.exe" --ignore-certificate-errors D:\TLS10GS\download\client\uploadPage.html

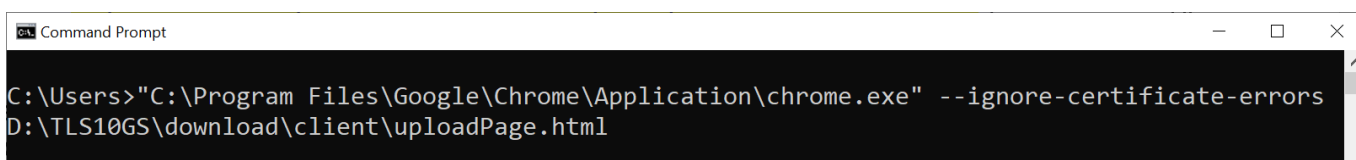


Figure 3-2 Example command line to open uploadPage.html

Upload menu is displayed in the web browser, as shown in Figure 3-3. Users can set server’s IP address, server’s port number, select the data pattern and data length. The HTML page will prepare the data and send a POST command along with the data pattern to the server when the “POST” button is pressed. Because the length of the data is greater than or equal to 16,000 bytes, only the data length and transfer speed are displayed on server console when the upload is completed, as shown in Figure 3-4.

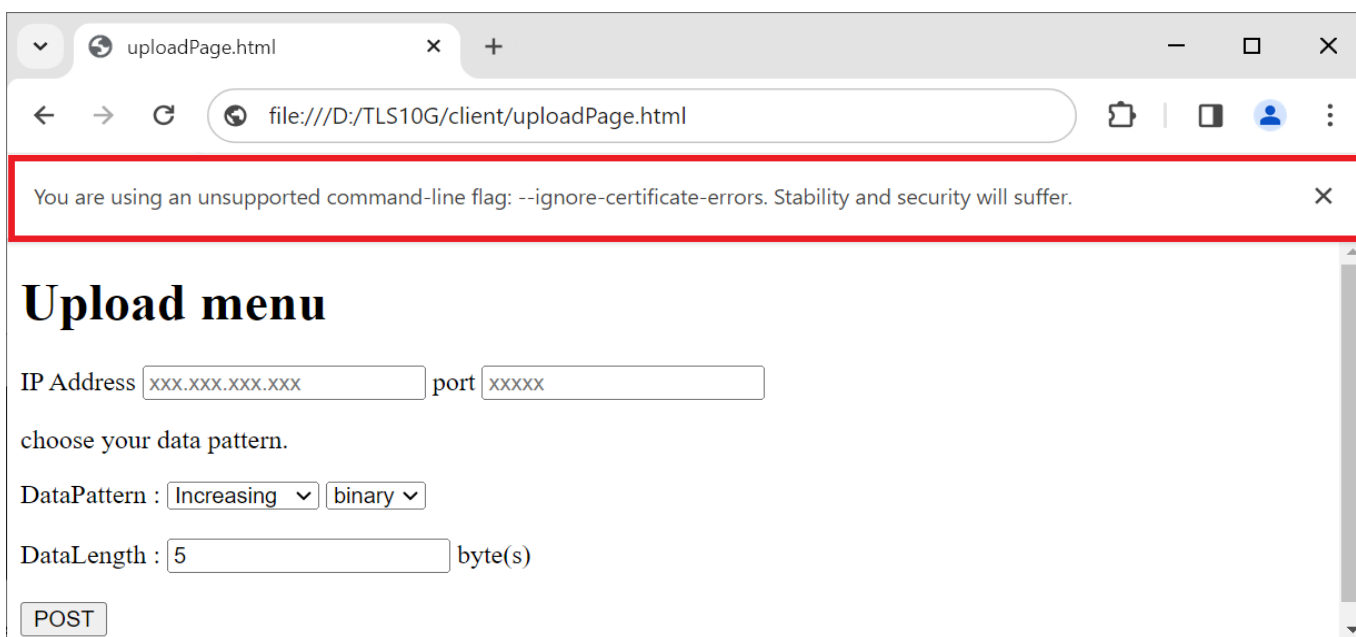


Figure 3-3 Secured upload page

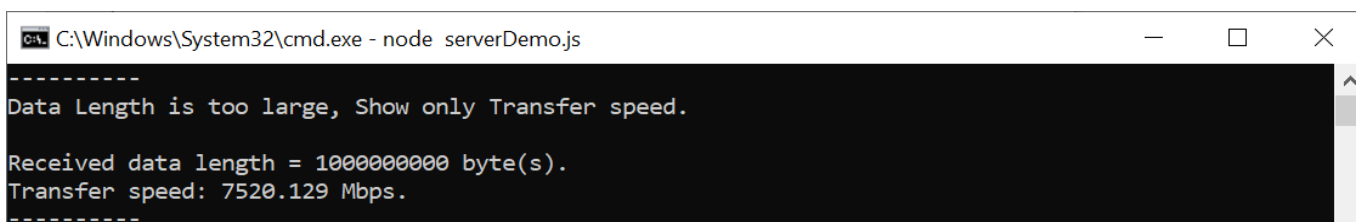


Figure 3-4 Server’s console when client upload large data

4 Test software on PC

“client” application is designed to run on PC as an example client for testing performance of TLS10GS-IP via ethernet. When starting client application, client’s IP address of network interface is displayed on application console, as shown in Figure 4-1.

```

-----
Client's IP Address :
[0] Ethernet 6
    192.168.7.26
[1] Ethernet 7
    169.254.87.186
[2] Ethernet
    192.168.11.26
[3] VirtualBox Host-Only Network
    192.168.56.1
[4] Loopback Pseudo-Interface 1
    127.0.0.1
-----
>>

```

Figure 4-1 Client application console

Users can use client application to download, upload data and test full duplex by using the following command.

4.1 Download data

command> myGET protocol://ip:port/download/pattern/length

This command simulates GET method of HTTP to download data from the server. Users can input URL and then received data is displayed on the serial console. “client” application will receive TxData from server and count the number of received data to validate whether the number of received data is matched the value form URL. For optimal data transfer speed, the received data remains undecrypted and unverified during the transfer process. Upon completion of data transfer, the last block of data will be decrypted and verified. Then the transfer speed is displayed on server console. If the data length is less than 16 kB, the received data, including the HTTP header, is also displayed, as shown in Figure 4-2.

```

-----
>> myGET https://192.168.7.25:60001/download/t1/10
connecting to 192.168.7.25
HTTP/1.1 200 OK
Connection: close
Date: Thu, 15 Feb 2024 12:02:54 GMT
Transfer-Encoding: chunked

A
0123456789
0

Received data: 10 Byte success.
Thoughtput 2.09 Mbps
-----

```

Figure 4-2 Client application console when downloading data from server

4.2 Upload data

command> myPOST protocol://ip:port/upload/pattern/length

This command simulates POST method of HTTP to upload data to the server. Users can indicate data pattern and data length in URL. “client” application will prepare the encrypted data pattern corresponding to data pattern from URL and send to server continuously. The transfer speed will be displayed on server application console, as shown in Figure 4-3.

```
-----
>> myPOST https://192.168.7.25:60001/upload/t1/100000000
connecting to 192.168.7.25
Thoughtput 9209.10 Mbps
-----
```

Figure 4-3 Client application console when uploading data to server

4.3 Full duplex test

command> myFull duplex protocol://ip:port/fullduplex/pattern/length

This command is used to transfer data between client and server in full duplex mode. It simulates POST method of HTTP with the fullduplex URL that request data pattern from server and also upload data pattern to the server. Users can indicate data pattern and data length in URL. After transmitting and receiving data are done, data length and transfer speed are displayed, as shown in Figure 4-4.

```
-----
>> myFull duplex https://192.168.7.25:60001/fullduplex/b1/1073741823
connecting to 192.168.7.25
Expected transmitted data length : 1073741823 Byte
Generate Data: 1073741823 Byte Success!
Expected received data length : 1073741823 Byte
Sending data is done.
Received data: 1073741823 Byte success.
-----
Send total data: 1079509084 Byte Success!
Thoughtput 3913.04 Mbps

Recv total data: 1073741943 Byte Success!
Thoughtput 3893.17 Mbps
-----
```

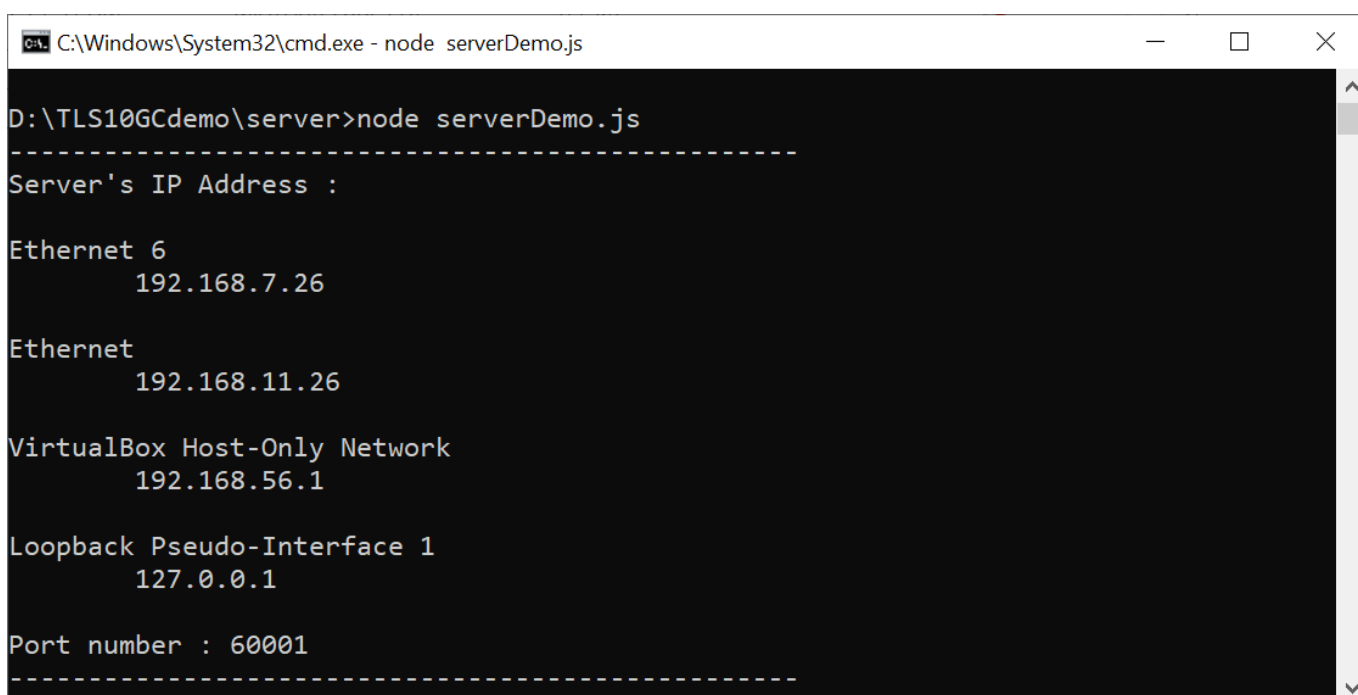
Figure 4-4 Client application console when full-duplex testing

5 Node.js server

In this demonstration, a sample server is created using Node.js. The server opens port 60001 for HTTPs connection. The required files for running the server are provided in server folder which contains the file as follow,

- 1) serverDemo.js for running server.
- 2) key.pem and cert.pem as a sample RSA key information and server's certificate.

When serverDemo.js is executed, IP address and port number of server are displayed on console, as shown in Figure 5-1.

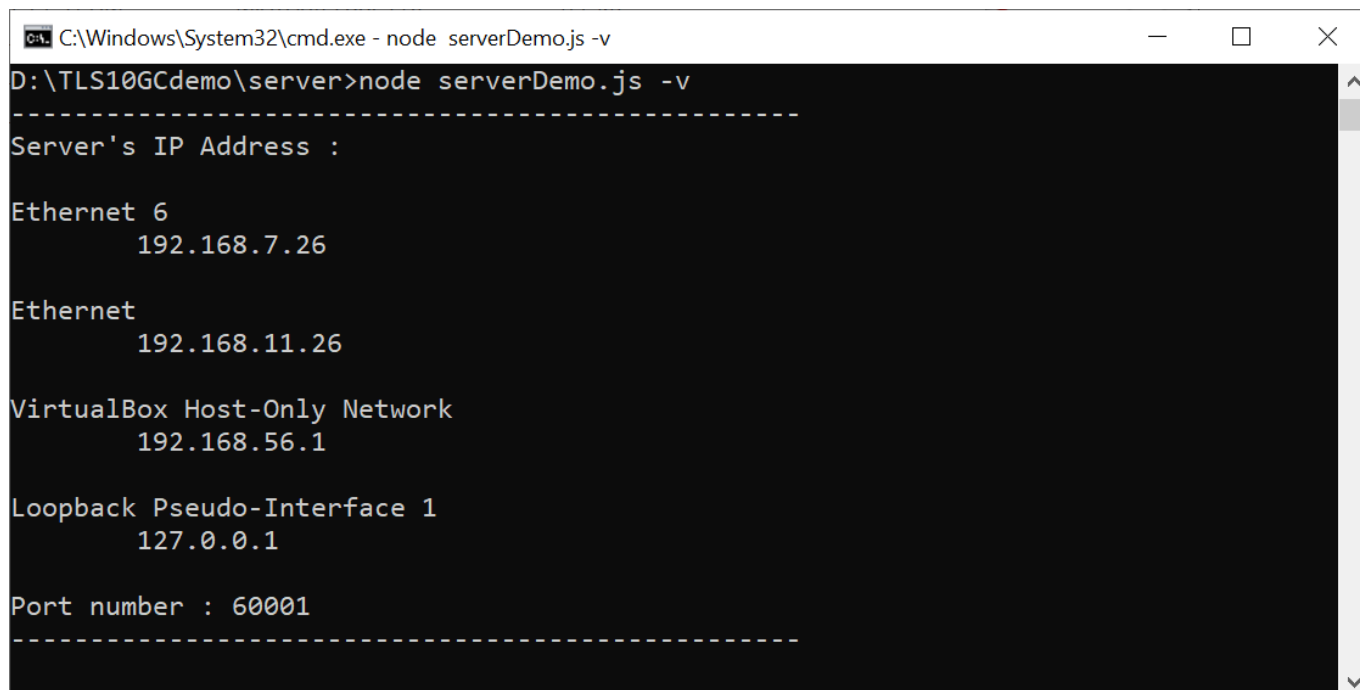


```

C:\Windows\System32\cmd.exe - node serverDemo.js
D:\TLS10GCdemo\server>node serverDemo.js
-----
Server's IP Address :
Ethernet 6
    192.168.7.26
Ethernet
    192.168.11.26
VirtualBox Host-Only Network
    192.168.56.1
Loopback Pseudo-Interface 1
    127.0.0.1
Port number : 60001
-----
  
```

Figure 5-1 Server console when serverDemo.js is executed

By default, serverDemo.js does not verify data to optimize transfer speed. However, users can enable the data verification feature by including the “-v” parameter when executing serverDemo.js, as shown in Figure 5-2.



```

C:\Windows\System32\cmd.exe - node serverDemo.js -v
D:\TLS10GCdemo\server>node serverDemo.js -v
-----
Server's IP Address :

Ethernet 6
    192.168.7.26

Ethernet
    192.168.11.26

VirtualBox Host-Only Network
    192.168.56.1

Loopback Pseudo-Interface 1
    127.0.0.1

Port number : 60001
-----

```

Figure 5-2 Server console when enabling verifying data

In case of client cannot access node.js server, please check firewall setting as below,

- 1) Go to Windows Defender Firewall with Advanced Security
- 2) Click on “Inbound Rules”
- 3) Search for “Node.js JavaScript Runtime” and open its properties
- 4) Go to “Protocols and Ports” tab and set Protocol type = TCP, Local port = Specific Ports that server on PC open. By default, the sample node.js server opens port 60001. Local port number is set to 60001, as shown in Figure 5-3.
- 5) Go to “Advanced” tab and mark the profile boxes that match the network profile of ethernet card, as shown in Figure 5-4.

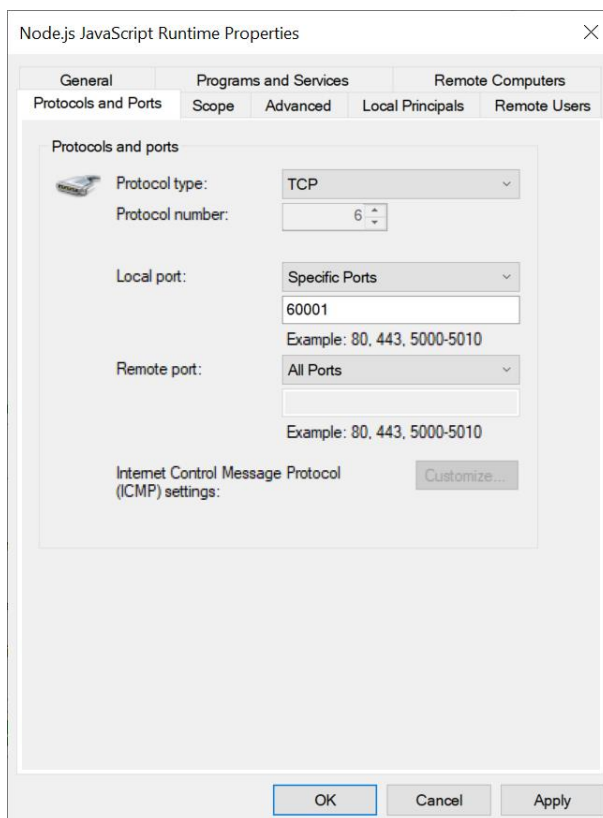


Figure 5-3 Protocols and Ports setting

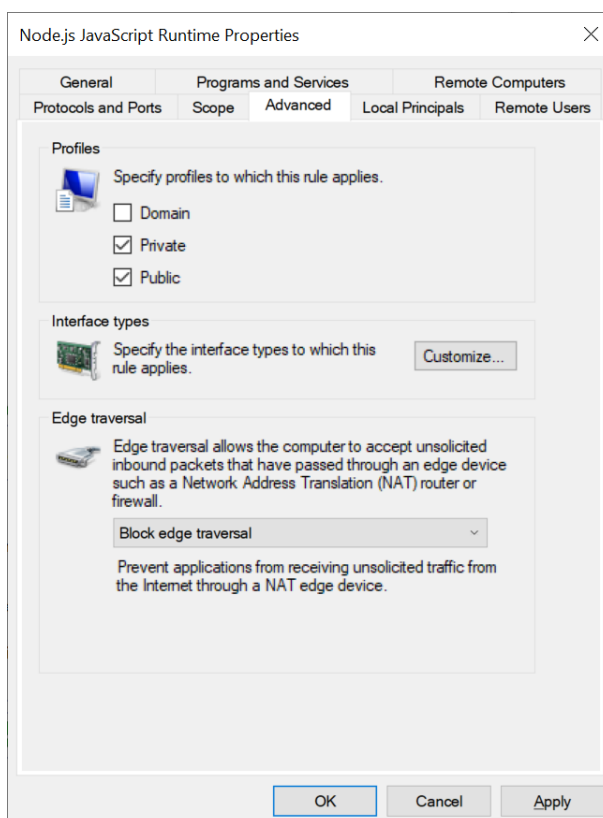


Figure 5-4 Advanced setting

Clients can download data patterns by sending a GET command with URL.

For downloading data pattern, there are 4 data patterns which are increasing binary, decreasing binary, increasing text and decreasing text pattern. When a server receives a GET request, the data pattern and the length of requested data are displayed on the server console, as shown in Figure 5-5.

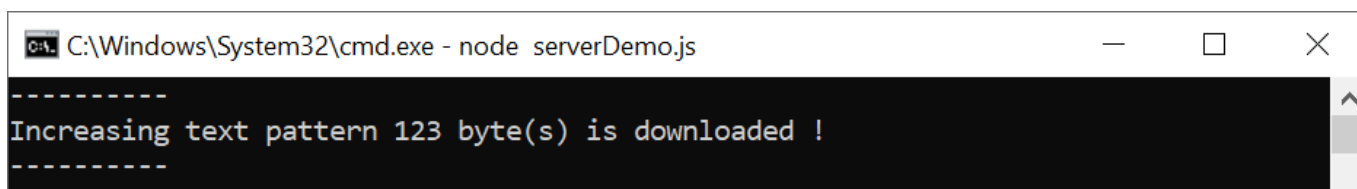


Figure 5-5 Server console when client download data pattern

Clients can upload data to the server by sending a POST command followed by the uploaded data. After the transfer is completed, the received data, its length and the transfer speed are displayed on the server console, as shown in Figure 5-6. If data length exceeds 16 kB, only the data length and transfer speed are displayed on the server console.

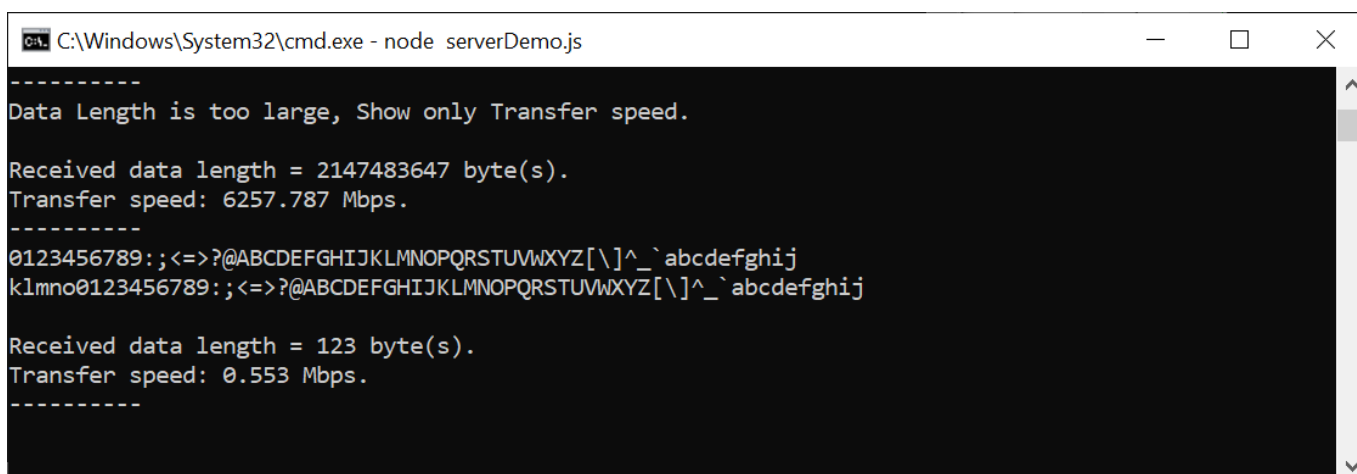
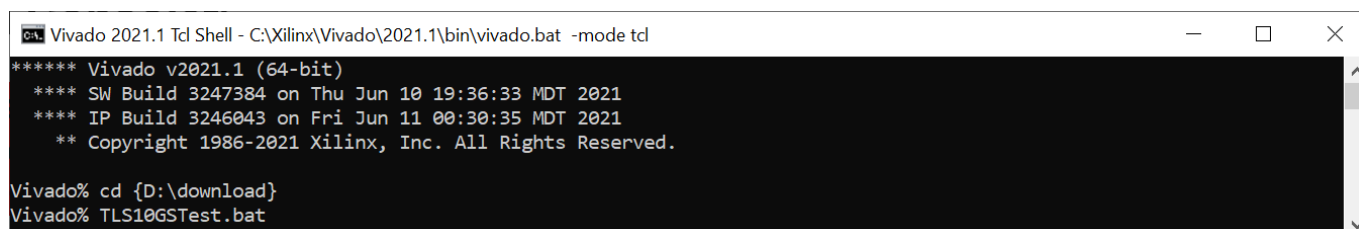


Figure 5-6 Server console when client upload data

6 Board setup

- 1) Make sure power switch is off and connect power supply to FPGA development board.
- 2) Connect two USB cables between FPGA board and PC via micro-USB ports.
- 3) Power on system.
- 4) Download configuration file and firmware to FPGA board by following step,
 - a) open Vivado TCL shell.
 - b) change current directory to download folder which includes demo configuration file.
 - c) Type “TLS10GSTest.bat”, as shown in Figure 6-1.



```

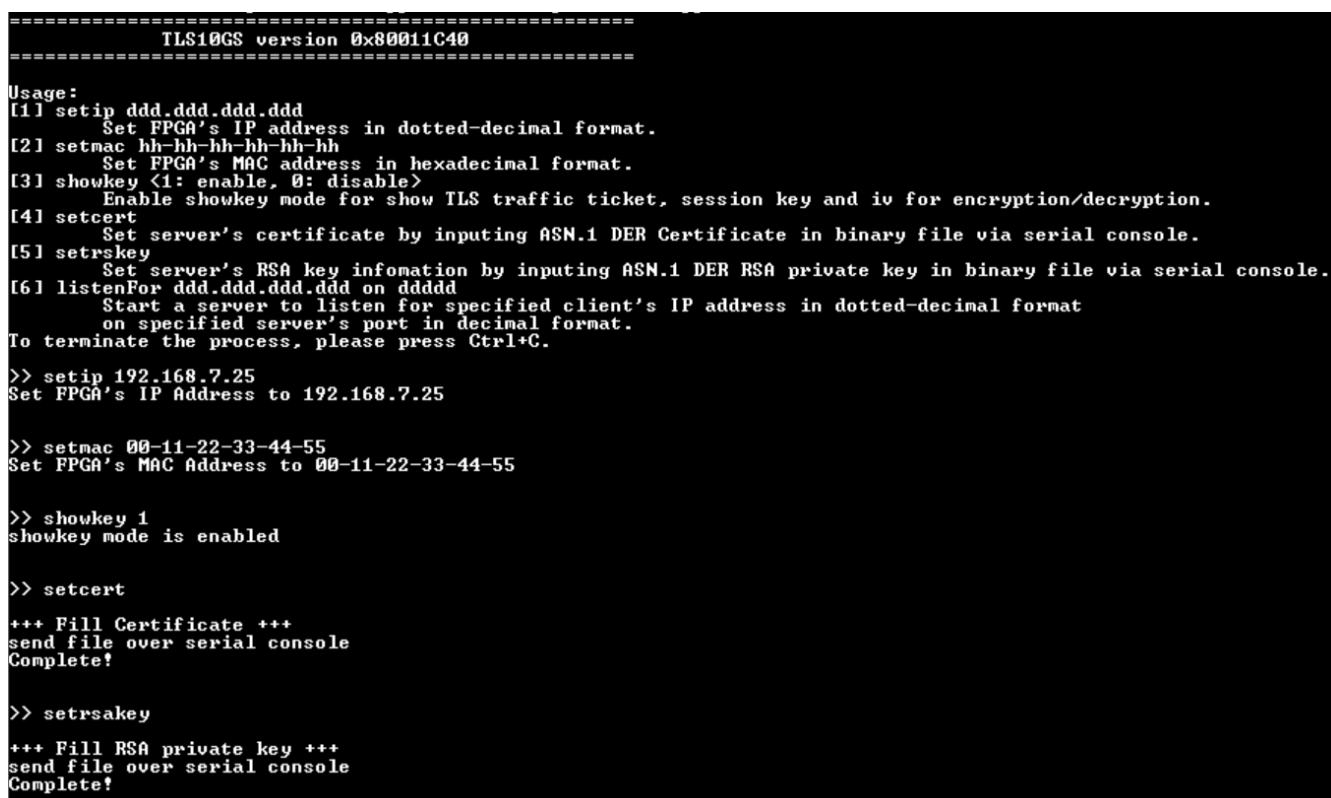
c:\ Vivado 2021.1 Tcl Shell - C:\Xilinx\Vivado\2021.1\bin\vivado.bat -mode tcl
***** Vivado v2021.1 (64-bit)
**** SW Build 3247384 on Thu Jun 10 19:36:33 MDT 2021
**** IP Build 3246043 on Fri Jun 11 00:30:35 MDT 2021
** Copyright 1986-2021 Xilinx, Inc. All Rights Reserved.

Vivado% cd {D:\download}
Vivado% TLS10GSTest.bat
  
```

Figure 6-1 Example command script for download configuration file

7 Serial Console

Users can set the parameters and start a server to listen for specified client’s IP address on specified server’s port and respond the supported request from a client by using the following command. The TLS10GSdemo commands and their usage will be displayed, as shown in Figure 7-1. Detailed information about each command is described in topic 8.



```

=====
          TLS10GS version 0x80011C40
=====
Usage:
[1] setip ddd.ddd.ddd.ddd
    Set FPGA's IP address in dotted-decimal format.
[2] setmac hh-hh-hh-hh-hh-hh
    Set FPGA's MAC address in hexadecimal format.
[3] showkey <1: enable, 0: disable>
    Enable showkey mode for show TLS traffic ticket, session key and iv for encryption/decryption.
[4] setcert
    Set server's certificate by inputing ASN.1 DER Certificate in binary file via serial console.
[5] setrsa
    Set server's RSA key information by inputing ASN.1 DER RSA private key in binary file via serial console.
[6] listenfor ddd.ddd.ddd.ddd on dddd
    Start a server to listen for specified client's IP address in dotted-decimal format
    on specified server's port in decimal format.
To terminate the process, please press Ctrl+C.

>> setip 192.168.7.25
Set FPGA's IP Address to 192.168.7.25

>> setmac 00-11-22-33-44-55
Set FPGA's MAC Address to 00-11-22-33-44-55

>> showkey 1
showkey mode is enabled

>> setcert
+++ Fill Certificate +++
send file over serial console
Complete!

>> setrsa
+++ Fill RSA private key +++
send file over serial console
Complete!
  
```

Figure 7-1 Serial console

8 Command detail

8.1 Set FPGA's IP Address

```
command> setip ddd.ddd.ddd.ddd
```

This command is used to set FPGA's IP address in dotted-decimal format. The default FPGA's IP address is 192.168.7.25. Users can input setip command following by valid IP address, as shown in Figure 7-1.

8.2 Set FPGA's MAC address

```
command> setmac hh-hh-hh-hh-hh-hh
```

This command is used to set FPGA's MAC address in hexadecimal format. The default FPGA's MAC address is 00-11-22-33-44-55.

8.3 Enable showkey mode

```
command> showkey <1: enable, 0: disable>
```

This command is used to enable showkey mode. When showkey mode is enabled, the TLS traffic ticket for encryption/decryption is displayed on the serial console, as shown in Figure 8-1. Users can use the TLS traffic ticket as (Pre)-Master-Secret log file for Wireshark* to decrypt transferred data between client and server.

*Wireshark, a network packet analyzer tool used for network troubleshooting, analysis, and security purposes.

```
>> listenfor 192.168.7.26 on 60001
Server listening on 192.168.7.26 port 60001

Wait Open connection ...
Connected from 192.168.7.26
=====
Traffic Secret
-----
CLIENT_HANDSHAKE_TRAFFIC_SECRET CB34ECB1E78163BA1C38C6DACB196A6DFFA21A8D9912EC18A2EF6283024DECE7 5DC2EDAD0214D6B84
F9E0B6E17783F366CE7B4DD6508920FC7AE77AD0E7792212F915AC73F551BF91B0706407A9EBC81
SERVER_HANDSHAKE_TRAFFIC_SECRET CB34ECB1E78163BA1C38C6DACB196A6DFFA21A8D9912EC18A2EF6283024DECE7 2FD635F4B8B514237
74A55B07DF3440E9C5E2B95B167A30F73529482E92FCC5E297F10190EA2720DD24DBB84714AE816
CLIENT_TRAFFIC_SECRET_0 CB34ECB1E78163BA1C38C6DACB196A6DFFA21A8D9912EC18A2EF6283024DECE7 4EB89986783DD2F038787CE5
9D088FFFA181D99166B496520921AFE58A212EBE22630BCE77E7888FCE044CCC6CF9BE1
SERVER_TRAFFIC_SECRET_0 CB34ECB1E78163BA1C38C6DACB196A6DFFA21A8D9912EC18A2EF6283024DECE7 FBE00A3390839E9805ECCA39A
4770604100B20156C53E7C70A4BBF0CFC70C6657E90F11984BFBCD9F507A7A7A5E3BE8B
=====

Transmitting...

Close connection
Connection is closed
=====

Transmitted data length = 999999999 Byte(s)
Transmit Speed 9328 Mbps

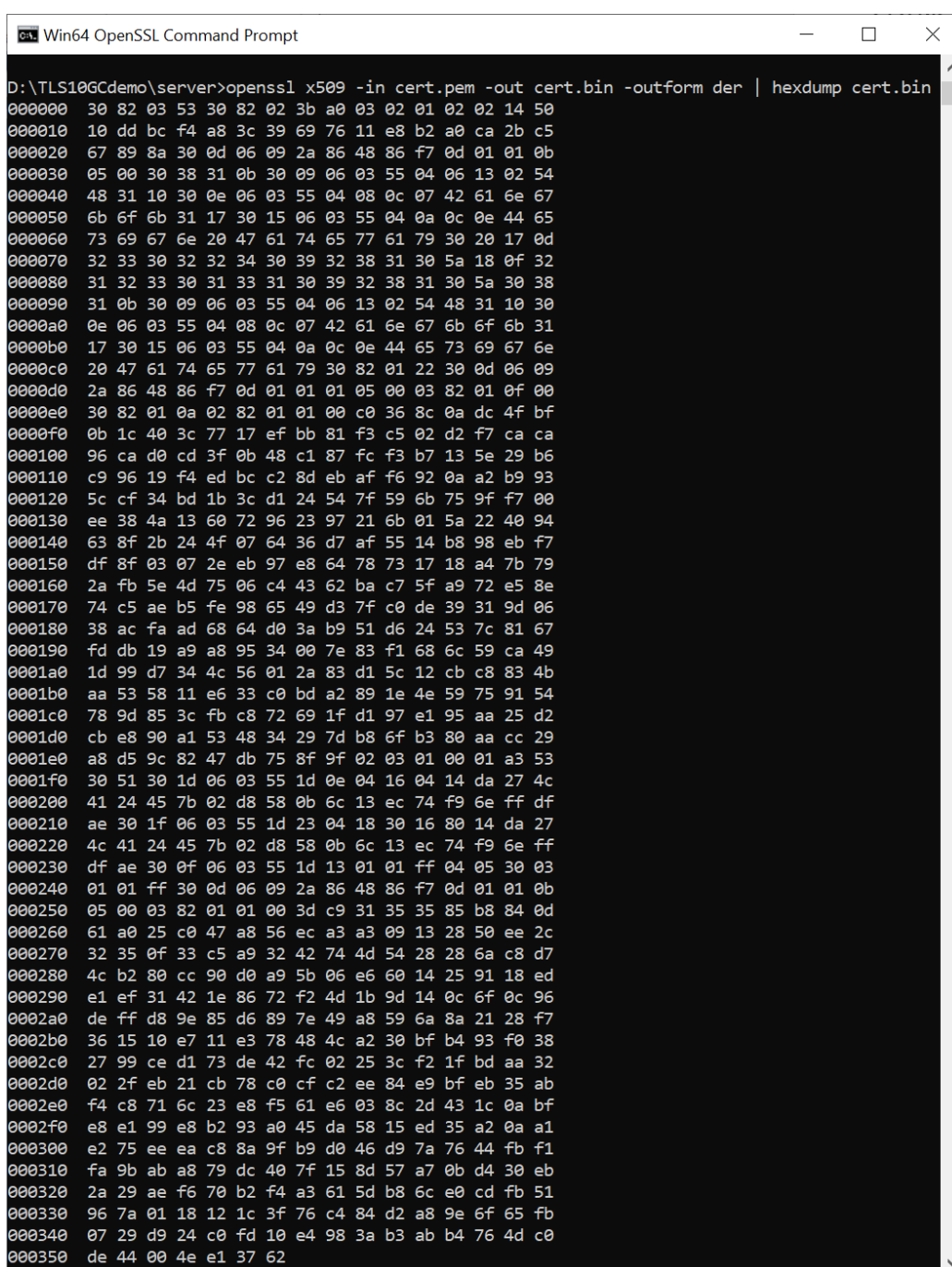
Wait Open connection ...
```

Figure 8-1 Serial console when showkey mode is enabled

8.4 Set certificate

command> setcert

This command is used to set the server's certificate, which must be valid before starting a server. The supported certificate format is ASN.1 DER in a binary file. A certificate in PEM format can be converted to ASN.1 DER in a binary file using the openssl command: `openssl x509 -in cert.pem -out cert.bin -outform der`, as shown in Figure 8-2. Users can send the binary certificate file (cert.bin) via the serial console as Figure 8-3. In this demonstration, the maximum certificate size is limited to 8 kB.



```

Win64 OpenSSL Command Prompt
D:\TLS10GCdemo\server>openssl x509 -in cert.pem -out cert.bin -outform der | hexdump cert.bin
000000 30 82 03 53 30 82 02 3b a0 03 02 01 02 02 14 50
000010 10 dd bc f4 a8 3c 39 69 76 11 e8 b2 a0 ca 2b c5
000020 67 89 8a 30 0d 06 09 2a 86 48 86 f7 0d 01 01 0b
000030 05 00 30 38 31 0b 30 09 06 03 55 04 06 13 02 54
000040 48 31 10 30 0e 06 03 55 04 08 0c 07 42 61 6e 67
000050 6b 6f 6b 31 17 30 15 06 03 55 04 0a 0c 0e 44 65
000060 73 69 67 6e 20 47 61 74 65 77 61 79 30 20 17 0d
000070 32 33 30 32 32 34 30 39 32 38 31 30 5a 18 0f 32
000080 31 32 33 30 31 33 31 30 39 32 38 31 30 5a 30 38
000090 31 0b 30 09 06 03 55 04 06 13 02 54 48 31 10 30
0000a0 0e 06 03 55 04 08 0c 07 42 61 6e 67 6b 6f 6b 31
0000b0 17 30 15 06 03 55 04 0a 0c 0e 44 65 73 69 67 6e
0000c0 20 47 61 74 65 77 61 79 30 82 01 22 30 0d 06 09
0000d0 2a 86 48 86 f7 0d 01 01 01 05 00 03 82 01 0f 00
0000e0 30 82 01 0a 02 82 01 01 00 c0 36 8c 0a dc 4f bf
0000f0 0b 1c 40 3c 77 17 ef bb 81 f3 c5 02 d2 f7 ca ca
000100 96 ca d0 cd 3f 0b 48 c1 87 fc f3 b7 13 5e 29 b6
000110 c9 96 19 f4 ed bc c2 8d eb af f6 92 0a a2 b9 93
000120 5c cf 34 bd 1b 3c d1 24 54 7f 59 6b 75 9f f7 00
000130 ee 38 4a 13 60 72 96 23 97 21 6b 01 5a 22 40 94
000140 63 8f 2b 24 4f 07 64 36 d7 af 55 14 b8 98 eb f7
000150 df 8f 03 07 2e eb 97 e8 64 78 73 17 18 a4 7b 79
000160 2a fb 5e 4d 75 06 c4 43 62 ba c7 5f a9 72 e5 8e
000170 74 c5 ae b5 fe 98 65 49 d3 7f c0 de 39 31 9d 06
000180 38 ac fa ad 68 64 d0 3a b9 51 d6 24 53 7c 81 67
000190 fd db 19 a9 a8 95 34 00 7e 83 f1 68 6c 59 ca 49
0001a0 1d 99 d7 34 4c 56 01 2a 83 d1 5c 12 cb c8 83 4b
0001b0 aa 53 58 11 e6 33 c0 bd a2 89 1e 4e 59 75 91 54
0001c0 78 9d 85 3c fb c8 72 69 1f d1 97 e1 95 aa 25 d2
0001d0 cb e8 90 a1 53 48 34 29 7d b8 6f b3 80 aa cc 29
0001e0 a8 d5 9c 82 47 db 75 8f 9f 02 03 01 00 01 a3 53
0001f0 30 51 30 1d 06 03 55 1d 0e 04 16 04 14 da 27 4c
000200 41 24 45 7b 02 d8 58 0b 6c 13 ec 74 f9 6e ff df
000210 ae 30 1f 06 03 55 1d 23 04 18 30 16 80 14 da 27
000220 4c 41 24 45 7b 02 d8 58 0b 6c 13 ec 74 f9 6e ff
000230 df ae 30 0f 06 03 55 1d 13 01 01 ff 04 05 30 03
000240 01 01 ff 30 0d 06 09 2a 86 48 86 f7 0d 01 01 0b
000250 05 00 03 82 01 01 00 3d c9 31 35 35 85 b8 84 0d
000260 61 a0 25 c0 47 a8 56 ec a3 a3 09 13 28 50 ee 2c
000270 32 35 0f 33 c5 a9 32 42 74 4d 54 28 28 6a c8 d7
000280 4c b2 80 cc 90 d0 a9 5b 06 e6 60 14 25 91 18 ed
000290 e1 ef 31 42 1e 86 72 f2 4d 1b 9d 14 0c 6f 0c 96
0002a0 de ff d8 9e 85 d6 89 7e 49 a8 59 6a 8a 21 28 f7
0002b0 36 15 10 e7 11 e3 78 48 4c a2 30 bf b4 93 f0 38
0002c0 27 99 ce d1 73 de 42 fc 02 25 3c f2 1f bd aa 32
0002d0 02 2f eb 21 cb 78 c0 cf c2 ee 84 e9 bf eb 35 ab
0002e0 f4 c8 71 6c 23 e8 f5 61 e6 03 8c 2d 43 1c 0a bf
0002f0 e8 e1 99 e8 b2 93 a0 45 da 58 15 ed 35 a2 0a a1
000300 e2 75 ee ea c8 8a 9f b9 d0 46 d9 7a 76 44 fb f1
000310 fa 9b ab a8 79 dc 40 7f 15 8d 57 a7 0b d4 30 eb
000320 2a 29 ae f6 70 b2 f4 a3 61 5d b8 6c e0 cd fb 51
000330 96 7a 01 18 12 1c 3f 76 c4 84 d2 a8 9e 6f 65 fb
000340 07 29 d9 24 c0 fd 10 e4 98 3a b3 ab b4 76 4d c0
000350 de 44 00 4e e1 37 62
  
```

Figure 8-2 Certificate information from openssl command

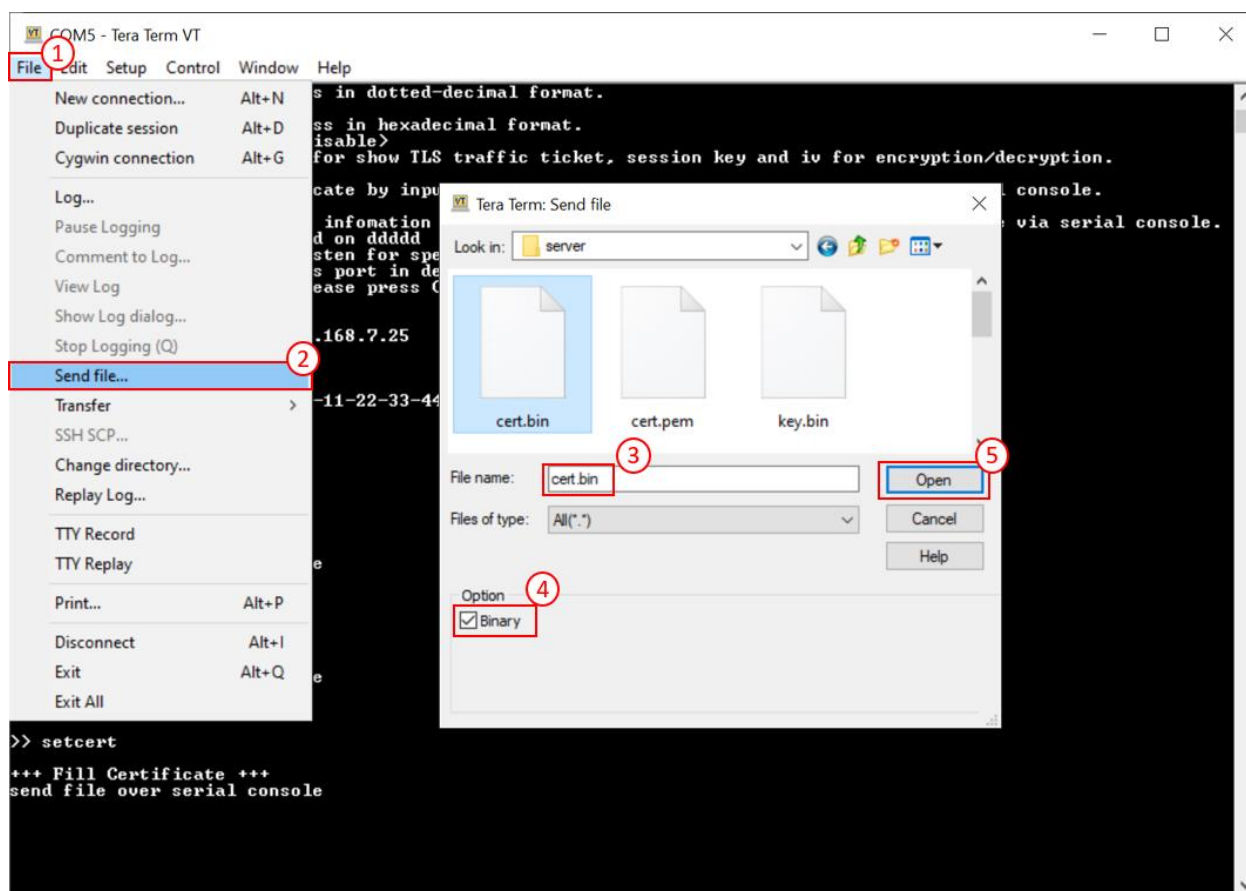


Figure 8-3 Example binary file transfer

8.5 Set RSA key information

command> setrsakey

This command is used to set RSA key information, which must be valid before starting a server. The supported RSA key format is ASN.1 DER in a binary file. An RSA key in PEM format can be converted to ASN.1 DER in a binary file using the openssl command: `openssl rsa -in key.pem -out key.bin -outform der`, as shown in Figure 8-4. Users can send the binary RSA key file (key.bin) via the serial console. In this demonstration, the maximum RSA key size is limited to 2 kB.

```

Win64 OpenSSL Command Prompt
D:\TLS10GCdemo\server>openssl rsa -in key.pem -out key.bin -outform der | hexdump key.bin
writing RSA key
000000 30 82 04 be 02 01 00 30 0d 06 09 2a 86 48 86 f7
000010 0d 01 01 01 05 00 04 82 04 a8 30 82 04 a4 02 01
000020 00 02 82 01 01 00 c0 36 8c 0a dc 4f bf 0b 1c 40
000030 3c 77 17 ef bb 81 f3 c5 02 d2 f7 ca ca 96 ca d0
000040 cd 3f 0b 48 c1 87 fc f3 b7 13 5e 29 b6 c9 96 19
000050 f4 ed bc c2 8d eb af f6 92 0a a2 b9 93 5c cf 34
000060 bd 1b 3c d1 24 54 7f 59 6b 75 9f f7 00 ee 38 4a
000070 13 60 72 96 23 97 21 6b 01 5a 22 40 94 63 8f 2b
000080 24 4f 07 64 36 d7 af 55 14 b8 98 eb f7 df 8f 03
000090 07 2e eb 97 e8 64 78 73 17 18 a4 7b 79 2a fb 5e
0000a0 4d 75 06 c4 43 62 ba c7 5f a9 72 e5 8e 74 c5 ae
0000b0 b5 fe 98 65 49 d3 7f c0 de 39 31 9d 06 38 ac fa
0000c0 ad 68 64 d0 3a b9 51 d6 24 53 7c 81 67 fd db 19
0000d0 a9 a8 95 34 00 7e 83 f1 68 6c 59 ca 49 1d 99 d7
0000e0 34 4c 56 01 2a 83 d1 5c 12 cb c8 83 4b aa 53 58
0000f0 11 e6 33 c0 bd a2 89 1e 4e 59 75 91 54 78 9d 85
000100 3c fb c8 72 69 1f d1 97 e1 95 aa 25 d2 cb e8 90
000110 a1 53 48 34 29 7d b8 6f b3 80 aa cc 29 a8 d5 9c
000120 82 47 db 75 8f 9f 02 03 01 00 01 02 82 01 00 2d
000130 c6 0f ad 9a 6f a7 48 47 0f 09 17 37 6d 10 d3 4e
000140 b1 11 0e 1a 92 81 92 4d 74 52 1c 7c 5c 74 32 25
000150 4c 08 c2 24 ff 7c 17 1f 96 c8 dc 40 c2 78 37 b3
000160 6c dd b4 88 b1 f6 e4 f8 37 4f fd 87 8b 2a c2 b0
000170 9d 23 d9 1c 22 1f 67 9b a2 10 61 3c 88 82 ab 3f
000180 fb 12 94 5b 69 d3 ac ad f0 91 31 fc c9 a1 c1 d5
000190 70 46 81 fb 70 de 53 af e4 01 b6 eb c5 fe 42 c6
0001a0 e8 69 8c a5 c6 fd c0 fd a1 a4 82 84 fd 02 2a bc
0001b0 33 08 62 39 2c 22 32 2c aa 3d 24 5b 19 5d 94 6e
0001c0 6f aa 14 31 36 e2 e2 62 40 54 04 d7 0c 42 eb 00
0001d0 75 80 76 ba 20 2d 80 f3 65 34 3c 67 ec bf 42 27
0001e0 da 2f 36 d3 7c 56 2f 38 d8 fd a4 6a c8 87 7e ae
0001f0 09 6a 74 a4 05 38 74 12 37 27 26 c4 de 35 0e 3b
000200 b1 c8 75 e6 c1 17 55 f1 10 4d fd 48 5a a0 73 eb
000210 93 d3 0d 81 3e ad 00 92 a2 9f 31 3f ad c9 43 7a
000220 af d7 70 ba 07 5e cb 7d c8 db 33 5b bc 13 91 02
000230 81 81 00 e1 b3 db c9 71 98 0b 2b 75 bb 11 9c bd
000240 22 73 89 3c 22 51 5d b3 ed 8c fe fb af 3c a0 e5
000250 56 12 a5 46 57 4f 88 a4 a5 5a 4f f0 49 23 13 f0
000260 97 32 8c c6 66 f2 c9 b4 82 1a f7 f0 aa b2 d5 c0
000270 e8 50 67 87 bb 2f a2 cc 58 51 cb d1 45 46 91 82
000280 43 00 dc b1 4d f5 b4 c6 da 8a 96 65 bb d5 e6 52
000290 ee 60 35 24 fa 94 db 13 39 7c 96 46 6a 99 9b ea
0002a0 3f 50 d0 83 2c 25 71 de 9d c2 fb 8c d0 0c c4 f9
0002b0 08 58 29 02 81 81 00 da 03 d7 3f 2a 62 dd 05 85
0002c0 7b e7 75 d9 00 86 82 7e 35 7b 31 9e 25 3e a5 6c
0002d0 af a4 33 19 90 77 0d ce 3d 1f 5b 2d d4 27 3f d6
0002e0 cc ec 51 5b 0f 36 2f b1 3b ac e1 4f 43 32 fe 42
0002f0 b3 93 55 ef 04 c5 31 81 15 c8 5f 8c b6 64 94 e5
000300 b7 a3 29 52 e1 30 7c ef ae 07 63 76 06 96 54 2d
000310 be 7e 6e 92 02 52 ea e7 46 80 6c e6 8a 35 e8 7a
000320 e8 dd d7 80 9d 7c 4a 87 6f e8 00 80 c2 57 79 42
000330 e3 4e fa 33 40 c2 87 02 81 81 00 9c 91 df 7b 0b
000340 e1 14 86 8e 82 3a 02 88 35 d8 fe 2f 88 02 f7 c4
000350 b4 9a e5 db 84 c1 c3 3f b4 88 f4 bc 2a 1f 53 44
000360 1c 2c dd 5d 6b ee f8 8b 22 e7 ff 3e 36 f6 5f b4
000370 67 b8 fb 9c a9 5d ab e8 c9 7f d5 82 13 f9 44 af
000380 0a e9 9b 41 4e 14 59 26 8b 02 93 16 30 65 ad 85
000390 70 df 48 db c4 04 eb 65 46 55 d9 28 10 e8 a8 5c
0003a0 da b9 31 aa 21 92 f3 d4 f9 1d 3f b8 6f 2c 7e a4
0003b0 96 be 47 30 74 b7 17 01 46 a7 99 02 81 81 00 97
0003c0 74 03 9c 45 fd d8 3d 75 b5 d5 dd f0 9a 84 d7 32
0003d0 86 44 c6 fb 6e 34 4f 07 6a 1d 4f c2 7a b1 ba 4d
0003e0 83 f8 bc 86 e1 d3 42 6e 1e 7e 2d 26 6d 32 df 7e
0003f0 e8 4d f9 57 ee ff 05 d3 a0 9c c2 1e 01 da 5b c1
000400 a9 38 41 e8 a6 ec c8 e3 ac e7 14 56 17 4a 70 00

```

Figure 8-4 RSA key information from openssl command

8.6 Start a server

Command> listenFor ddd.ddd.ddd.ddd on dddd

This command is used for starting a server to listen for a specified client's IP address on a specified server's port. Users can input the client's IP address that the server is listening for and server's port number. When a connection is established from the specified client's IP address, the server will respond to incoming data corresponding to supported request from the client as follows.

8.6.1 GET /download/pattern/length

This request is responded to by transmitting a data pattern and length based on parameters within the request. After the transmission is complete, the data length and transmit speed are displayed, as shown in Figure 8-5.

```
>> listenfor 192.168.7.26 on 60001
Server listening on 192.168.7.26 port 60001

Wait Open connection ...
Connected from 192.168.7.26
=====
Transmitting...

Close connection
Connection is closed
=====
Transmitted data length = 2147483647 Byte(s)
Transmit Speed 9272 Mbps

Wait Open connection ...
```

Figure 8-5 Serial console when transmitting large data

8.6.2 POST /upload/pattern/length

This request is responded to by receiving and verifying the data pattern and length based on parameters within the request. If the received data matches the expected data, the total length of the received data and the transfer speed are displayed on serial console.

If the received data length exceeds 16kB, "Data Length is too large, Show only Transfer speed" is displayed instead of the received data, as shown in Figure 8-6.

```
>> listenfor 192.168.7.26 on 60001
Server listening on 192.168.7.26 port 60001

Wait Open connection ...
Connected from 192.168.7.26
=====
Receiving...

Data Length is too large, Show only Transfer speed

Close connection
Connection is closed
=====
Received data length = 2147483647 Byte(s)
Receive Speed 9192 Mbps
```

Figure 8-6 Serial console when receiving large data

8.6.3 POST /fullduplex/pattern/length

This request is responded to by transferring data between the client and server in full duplex mode. The server receives and verifies the data pattern and also transmits data to the client based on parameters within the request. After transmitting and receiving data are completed, the data length and transfer speed are displayed, as shown in Figure 8-7.

```

>> listenfor 192.168.7.26 on 60001
Server listening on 192.168.7.26 port 60001

Wait Open connection ...
Connected from 192.168.7.26
=====

Transferring...

Close connection
Connection is closed
=====

Transmitted data length = 2147483647 Byte(s)
Transmit Speed 6672 Mbps
-----

Received data length = 2147483647 Byte(s)
Receive Speed 4744 Mbps

Wait Open connection ...

```

Figure 8-7 Serial console when full duplex mode is tested

9 Test setup when using 2 FPGA boards

9.1 Environment setup when using 2 FPGA boards

To operate TLS10GS-IP demo with TLS10GC-IP demo, please prepare following test environment.

- 1) FPGA development boards (ZCU102 as a server and ZCU106 as a client).
- 2) 10 Gb Ethernet cable:
 - a) 10 Gb SFP+ Passive Direct Attach Cable (DAC) which has 1-m or less length
 - b) 10 Gb SFP+ Active Optical Cable (AOC)
 - c) 2x10 Gb SFP+ transceiver (10G BASE-R) with optical cable (LC to LC, Multimode)
- 3) Micro USB cable for JTAG connection connecting between FPGA board and Test PC.
- 4) 2 Micro USB cable for UART connection connecting between ZCU102 board and Test PC and between ZCU106 board and Test PC.
- 5) Vivado tool for programming FPGA installed on Test PC.
- 6) Serial console software such as TeraTerm installed on PC. The setting on the console is Baudrate=115200, Data=8-bit, Non-parity and Stop=1.
- 7) Batch file named "TLS10GSIPTest.bat" and "TLS10GCIPTest.bat" (To download these files, please visit our web site at www.design-gateway.com)

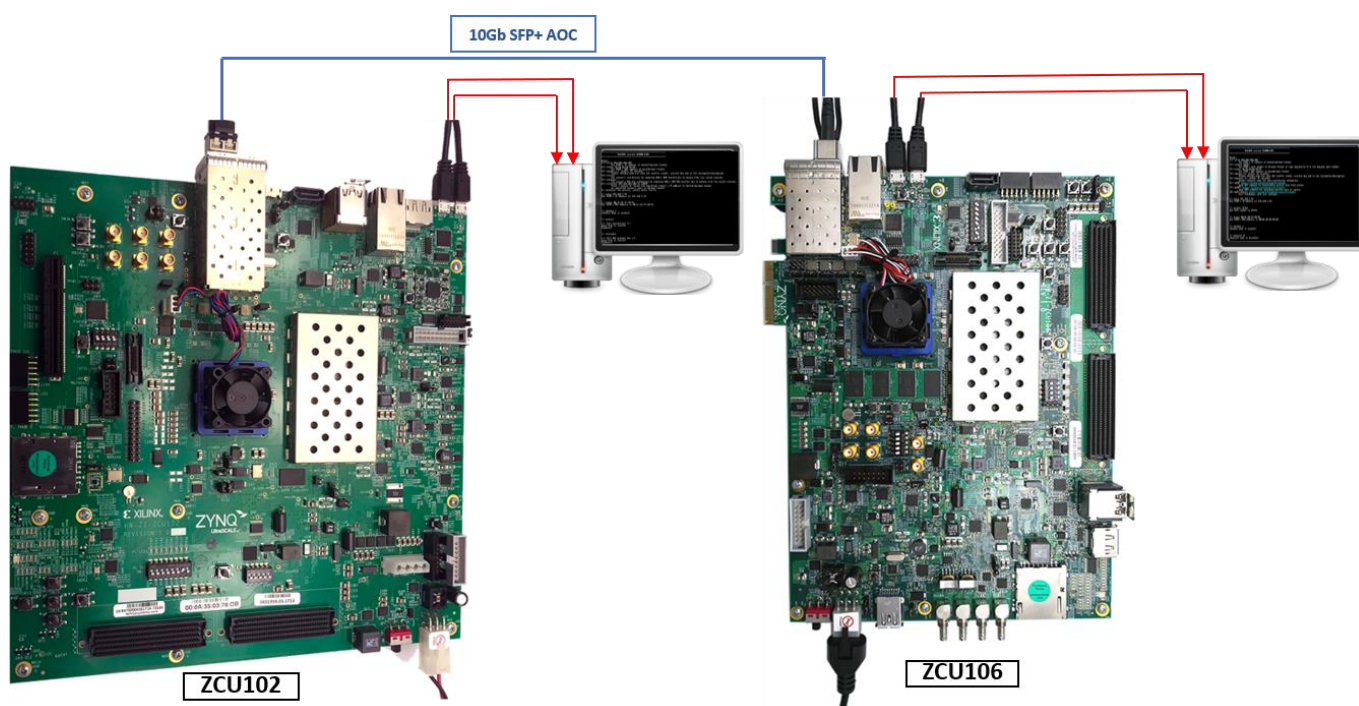


Figure 9-1 TLS10GS-IP demo environment when using 2 FPGA boards

Follow step 1)-8) of Topic 6 Board setup to prepare FPGA boards for running the demo. Run "TLS10GSTest.bat" to download configuration file and firmware to ZCU102 board as a server and run "TLS10GCTest.bat" to download configuration file and firmware to ZCU106 board as a client. The details of supported commands and their usage for TLS10GC-IP demo is described in the following link.

<https://www.dgway.com/products/IP/TLS-IP/TLS10GCIP-instruction-xilinx-en/>

9.2 Test sequence

9.2.1 Set parameters and start a server

- 1) Set network parameters of each FPGA board: IP address, port number, and mac address.
- 2) Set server's certificate and RSA key information via serial console of server.
- 3) Start a server by entering the following command in server's console:
listenFor <client's IP address> on <server's port number>, as shown in Figure 9-2.

| TLS10GS | TLS10GC |
|---|---|
| <pre> ===== TLS10GS version 0x80011C40 ===== Usage: [1] setip ddd.ddd.ddd.ddd Set FPGA's IP address in dotted-decimal format. [2] setmac hh-hh-hh-hh-hh-hh Set FPGA's MAC address in hexadecimal format. [3] showkey <1: enable, 0: disable> Enable showkey mode for show TLS traffic ticket, session key and iv for encryption/decryption. [4] setcert Set server's certificate by inputing ASN.1 DER Certificate in bin ary file via serial console. [5] setrsakey Set server's RSA key infomation by inputing ASN.1 DER RSA private key in binary file via serial console. [6] listenFor ddd.ddd.ddd.ddd on dddd Start a server to listen for specified client's IP address in dot ted-decimal format on specified server's port in decimal format. To terminate the process, please press Ctrl+C. >> setip 192.168.7.25 Set FPGA's IP Address to 192.168.7.25 >> setmac 00-11-22-33-44-55 Set FPGA's MAC Address to 00-11-22-33-44-55 >> setcert +++ Fill Certificate +++ send file over serial console Complete! >> setrsakey +++ Fill RSA private key +++ send file over serial console Complete! >> listenFor 192.168.7.42 on 60001 Server listening on 192.168.7.42 port 60001 Wait Open connection ... </pre> | <pre> ===== TLS10GC version 0x80011841 ===== Usage: [1] setip ddd.ddd.ddd.ddd Set FPGA's IP address in dotted-decimal format. [2] setport dddd Set FPGA's port number in decimal format or type dynamic/d/~d to set dynamic port number. [3] setmac hh-hh-hh-hh-hh-hh Set FPGA's MAC address in hexadecimal format. [4] showkey <1: enable, 0: disable> Enable showkey mode for show TLS traffic ticket, session key and iv for encryption/decryption. [5] showcert <1: enable, 0: disable> Enable showcert mode for show certificate infomation. [6] myGET https://ip:port/download/pattern/length Send GET command for downloading pattern data from server. [7] myPOST https://ip:port/upload/pattern/length Send POST command for uploading pattern data to server. [8] myFullduplex https://ip:port/fullduplex/pattern/length Test fullduplex with test software. >> setip 192.168.7.42 Set FPGA's IP Address to 192.168.7.42 >> setmac 00-01-02-03-04-05 Set FPGA's MAC Address to 00-01-02-03-04-05 >> setport 60000 Set Port number to 60000 >> </pre> |

Figure 9-2 Server and client console when parameters are set

9.2.2 Transmit data test (Server to client)

Enter the command myGET protocol://ip:port/download/pattern/length through the client's console to request the data pattern from TLS10GS demo. Once the data transfer is complete, the transfer results and speed will be presented on both the client's and server's consoles, as shown in Figure 9-3.

| TLS10GS | TLS10GC |
|--|--|
| <pre> >> listenFor 192.168.7.42 on 60001 Server listening on 192.168.7.42 port 60001 Wait Open connection ... Connected from 192.168.7.42 ===== Transmitting... Close connection Connection is closed ===== Transmitted data length = 1073741824 Byte(s) Transmit Speed 9360 Mbps Wait Open connection ... </pre> | <pre> >> myGET https://192.168.7.25:60001/download/t0/1073741824 Open connection Connecting to 192.168.7.25 ===== Downloading... Data Length is too large, Show only Transfer speed Close connection Connection closed ===== Received data length = 1073741824 Byte(s) downloading Speed 9360 Mbps </pre> |

Figure 9-3 Server and client console when transfer data from server to client

9.2.3 Receive data test (Client to server)

Enter the command myPOST protocol://ip:port/upload/pattern/length through client's console to transmit the data pattern from TLS10GC demo to TLS10GS demo. Once the data transfer is complete, the transfer results and speed will be presented on both the client's and server's consoles, as shown in Figure 9-4.

| TLS10GS | TLS10GC |
|--|--|
| <pre> Wait Open connection ... Connected from 192.168.7.42 ===== Receiving... Data Length is too large, Show only Transfer speed Close connection Connection is closed ===== Received data length = 2147483647 Byte(s) Receive Speed 9360 Mbps Wait Open connection ... </pre> | <pre> >> myPOST https://192.168.7.25:60001/upload/t1/2147483647 Open connection Connecting to 192.168.7.25 ===== Uploading... Close connection Connection closed ===== Transmitted data length = 2147483647 Byte(s) Uploading Speed 9360 Mbps </pre> |

Figure 9-4 Server and client console when transfer data from client to server

9.2.4 Full duplex test

Enter the command myFull duplex protocol://ip:port/fullduplex/pattern/length through client's console to test transferring data in full duplex mode between TLS10GS demo and TLS10GC demo. Once the data transfer is complete, the transfer results and speed will be presented on both the client's and server's consoles, as shown in Figure 9-5.

| TLS10GS | TLS10GC |
|--|--|
| <pre> Wait Open connection ... Connected from 192.168.7.42 ===== Transferring... Close connection Connection is closed ===== Transmitted data length = 999999999 Byte(s) Transmit Speed 9360 Mbps ----- Received data length = 999999999 Byte(s) Receive Speed 9360 Mbps Wait Open connection ... </pre> | <pre> >> myFull duplex https://192.168.7.25:60001/fullduplex/h1/999999999 Open connection Connecting to 192.168.7.25 ===== Waiting... Close connection Connection closed ===== Transmitted data length = 999999999 Byte(s) Uploading Speed 9360 Mbps ----- Received data length = 999999999 Byte(s) downloading Speed 9360 Mbps </pre> |

Figure 9-5 Server and client console when transfer data in full duplex mode

10 Test results

This demonstration, TLS10GSdemo is showcased for its ability to function as a secure server. The HTTPS protocol is chosen as the application layer to demonstrate that TLS10GS-IP can implement TLS1.3 to secure HTTP communication. The subsequent section details the test results when transferring data between each component, covering 2 main aspects: functionality testing and performance testing.

10.1 Functionality testing

TLS10GSdemo is designed to respond HTTPs clients, illustrating that TLS10GS-IP can handle TLS1.3 connection similar to the example node.js server. As shown in Figure 10-1 and Figure 10-2, a web browser requests a data pattern via the GET command and displays the received data from TLS10GSdemo on the browser, producing the same result as when requesting from node.js server.

In the case of uploading data, TLS10GSdemo is capable of receiving a data pattern from the web browser and displaying the TLS payload, as shown in Figure 10-4. This mirrors the process where the node.js server receives data from the web browser, as shown in Figure 10-3.

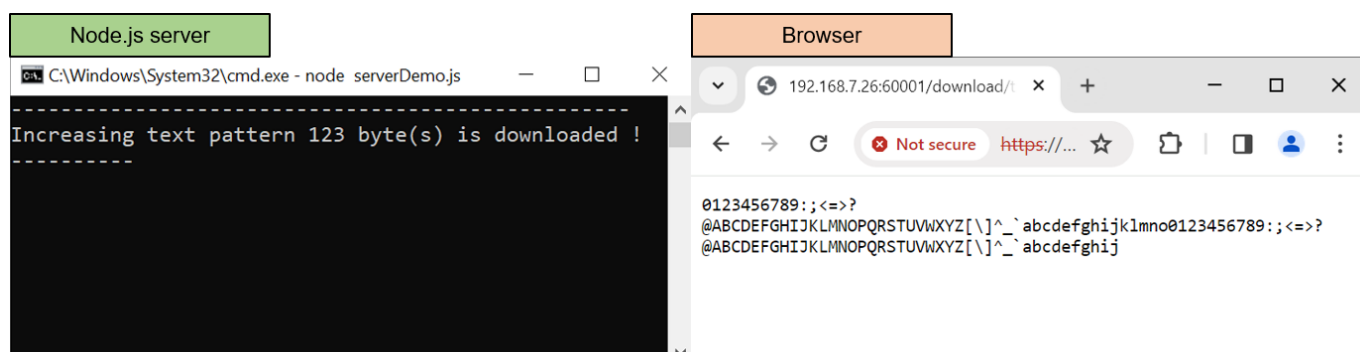


Figure 10-1 Test results when web browser download data from node.js server

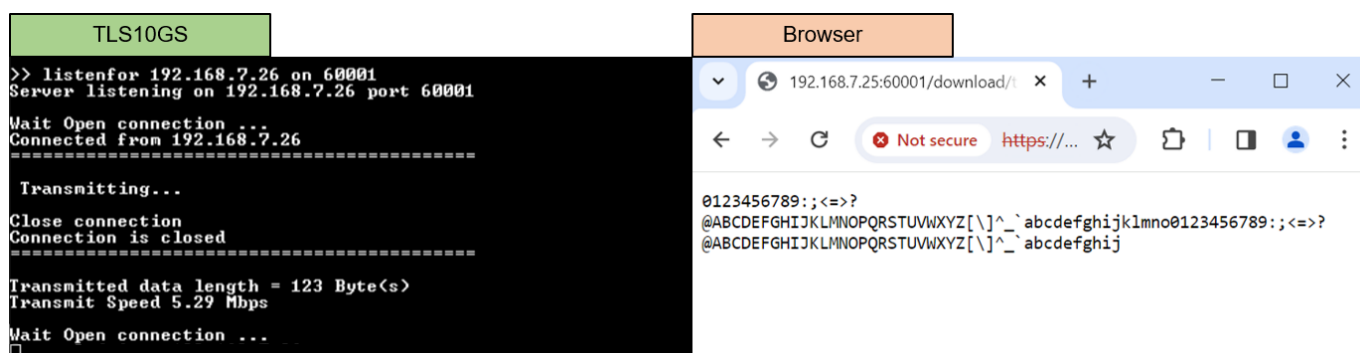


Figure 10-2 Test results when web browser download data from TLS10GSdemo

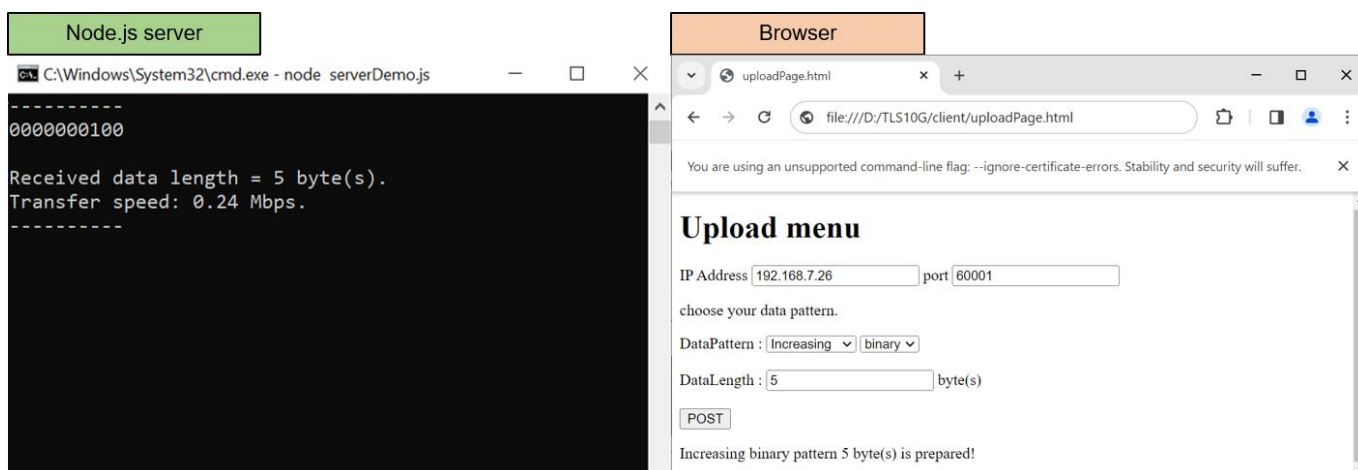


Figure 10-3 Test results when web browser upload data to node.js server

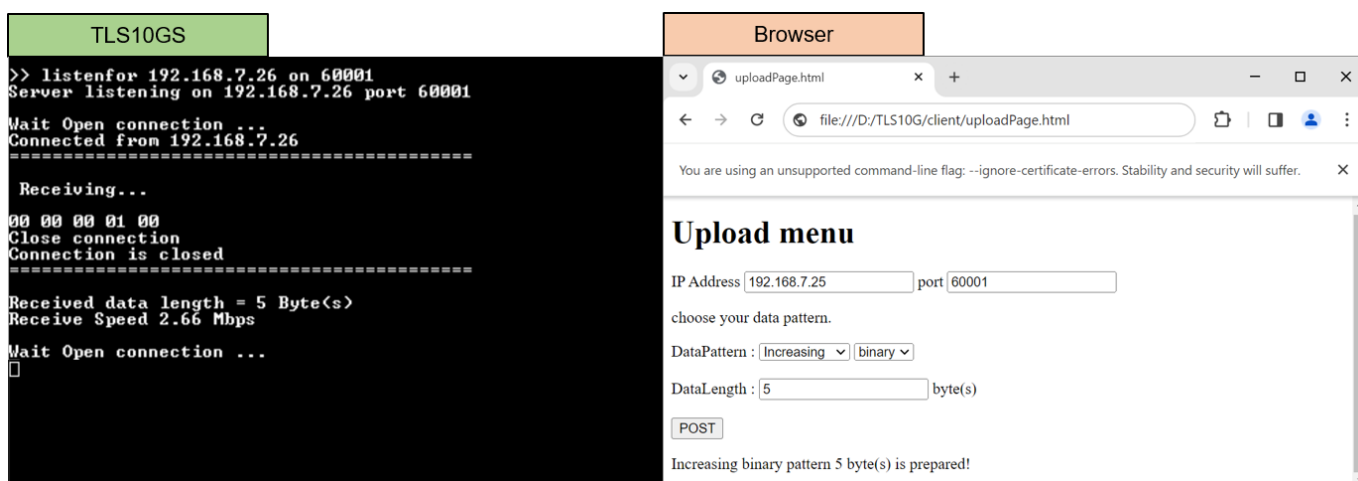


Figure 10-4 Test results when web browser upload data to TLS10GSdemo

10.2 Performance testing

When users use a web browser as a client to communicate with TLS10GSdemo, the CPU manages encryption/decryption during data transfer, causing a decrease in transfer speed. To achieve the maximum throughput, the "client" application is used instead.

"Client" application is designed to encrypt data before transmission through the network in Tx mode, decrypt the last data block and verify the total received data size in Rx mode. As shown in Figure 10-5 and Figure 10-6, the transfer speed is nearly by 10 Gbps and the utilization of the Intel i7 CPU is approximately 100%, as monitored by the PC's task manager. This indicates that if the CPU is tasked with encrypting/decrypting data while transferring data through the network, the transfer speed will be reduced.

| TLS10GS | "client" application |
|--|---|
| <pre> >> listenfor 192.168.7.26 on 60001 Server listening on 192.168.7.26 port 60001 Wait Open connection ... Connected from 192.168.7.26 ===== Transmitting... Close connection Connection is closed ===== Transmitted data length = 2147483647 Byte(s) Transmit Speed 9272 Mbps </pre> | <pre> ----- >> myGET https://192.168.7.25:60001/download/t1/2147483647 connecting to 192.168.7.25 Received data: 2147483647 Byte success. Thoughtput 9280.53 Mbps ----- >> </pre> |

Figure 10-5 Test results when "client" application download data from TLS10GSdemo

| TLS10GS | "client" application |
|---|--|
| <pre> Wait Open connection ... Connected from 192.168.7.26 ===== Receiving... Data Length is too large, Show only Transfer speed Close connection Connection is closed ===== Received data length = 2147483647 Byte(s) Receive Speed 8784 Mbps Wait Open connection ... </pre> | <pre> ----- >> myPOST https://192.168.7.25:60001/upload/t1/2147483647 connecting to 192.168.7.25 Thoughtput 8938.13 Mbps ----- >> </pre> |

Figure 10-6 Test results when "client" application upload data from TLS10GSdemo

For full duplex mode, the transfer speed between “client” application and TLS10GSdemo decreases, as shown in Figure 10-7. This suggests that the CPU cannot handle both receiving and transmitting task in a secure connection to maintain the 10 Gbps throughput.

In the testing scenario between two FPGA boards, where TLS10GSdemo acts as the server and TLS10GCdemo as the client, cryptographic tasks, including encryption/decryption, are entirely offloaded to hardware. As shown in Figure 10-8, the throughput increases to 9360 Mbps, representing the maximum throughput achievable with TCP/IP in this demonstration.

| TLS10GS | “client” application |
|---|---|
| <pre> >> listenFor 192.168.7.42 on 60001 Server listening on 192.168.7.42 port 60001 Wait Open connection ... Connected from 192.168.7.42 ===== Transferring... Close connection Connection is closed ===== Transmitted data length = 2147483640 Byte(s) Transmit Speed 7968 Mbps ----- Received data length = 2147483640 Byte(s) Receive Speed 5648 Mbps Wait Open connection ... </pre> | <pre> >> myFull duplex https://192.168.7.25:60001/fullduplex/b1/2147483640 ----- connecting to 192.168.7.25 Expected transmitted data length : 2147483640 Byte Generate Data: 2147483640 Byte Success! Expected received data length : 2147483640 Byte Received data: 2147483640 Byte success. Sending data is done. ----- Send total data: 2159018069 Byte Success! Thoughtput 5742.53 Mbps Recv total data: 2147483760 Byte Success! Thoughtput 6488.82 Mbps ----- >> </pre> |

Figure 10-7 Full duplex test results between TLS10GSdemo and “client” application

| TLS10GS | TLS10GC |
|---|---|
| <pre> Wait Open connection ... Connected from 192.168.7.42 ===== Transferring... Close connection Connection is closed ===== Transmitted data length = 999999999 Byte(s) Transmit Speed 9360 Mbps ----- Received data length = 999999999 Byte(s) Receive Speed 9360 Mbps Wait Open connection ... </pre> | <pre> >> myFull duplex https://192.168.7.25:60001/fullduplex/b1/999999999 ----- Open connection Connecting to 192.168.7.25 ===== Waiting... Close connection Connection closed ===== Transmitted data length = 999999999 Byte(s) Uploading Speed 9360 Mbps ----- Received data length = 999999999 Byte(s) downloading Speed 9360 Mbps </pre> |

Figure 10-8 Full duplex test results between TLS10GSdemo and TLS10GCdemo

11 Revision History

| Revision | Date | Description |
|----------|------------|-------------------------|
| 1.00 | 5-Mar-2024 | Initial version release |