TLS1.3 Demonstration

Rev1.01 30-Jun-2023

1 Introduction

Transport Layer Security (TLS) is a cryptographic protocol that provides a secure connection between a client and a server over the network. TLS is widely used in secure web browsing, email, file transferring, voice-over-IP, etc.

TLS is implemented on TCP/IP protocols to provide security of data for the application layer. Hypertext Transfer Protocol (HTTP) is an application layer protocol over TCP/IP that transfers plain data though the network. To protect transferred data, Hypertext Transfer Protocol Secure (HTTPs) is used instead. TLS is implemented to encrypt/decrypt application data when transferring through the TCP/IP layer. Not only data encryption, TLS provides authentication and integrity by verifying server's certificates and authentication tags of each packet.

TLS1.3 demo demonstrates the utilization of DG's security IP-core, including AES256GCMIP, to establish a secure connection using the Transport Layer Security protocol version 1.3 (TLS1.3) as a client that is compatible with general servers such as Node.js. Users can establish a connection with an HTTP/HTTPS server using TLS1Gdemo, similar to using a web browser. User can set network parameters, download and upload data by inputting supported command via serial console.

For further information, including technical details, DG's IP-core and hardware sample, please contact us via https://design-gateway.com.

2 System Overview

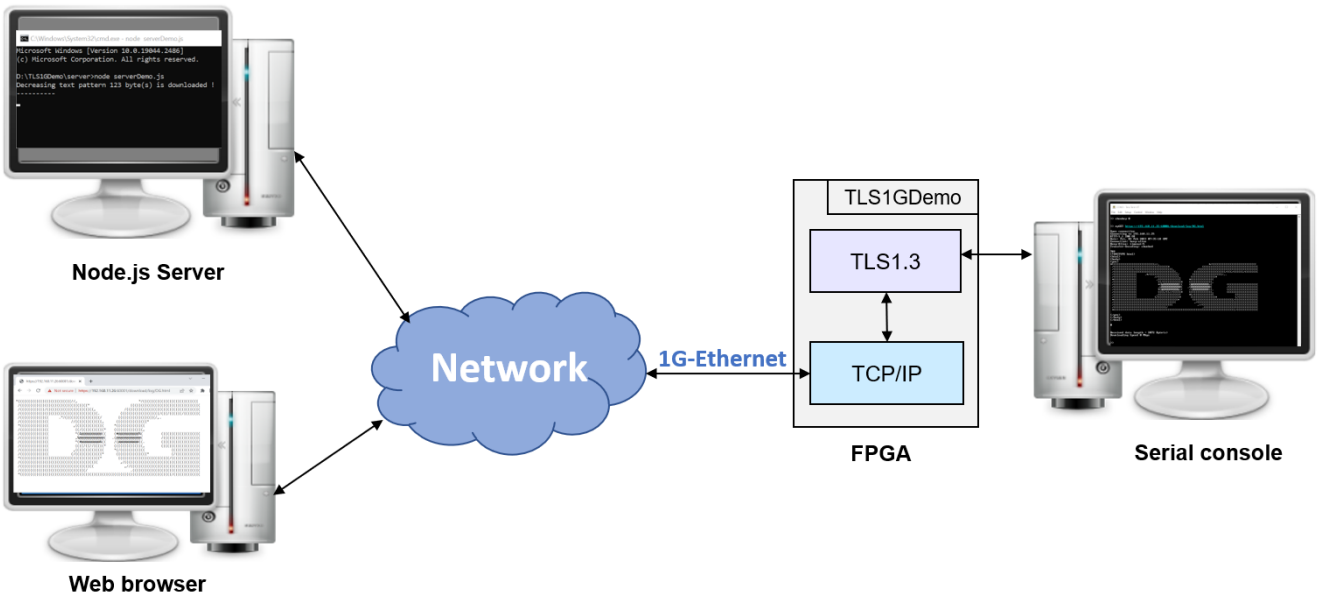

Figure 2-1 System overview

The demonstration system contains server, web browser and TLS1Gdemo on AC701 board connecting together through the network as shown in Figure 2-1. After establishing a connection, the client can upload data to the server via POST method and download data from the server via GET method.

2.1 Environment setup

To operate TLS1G demo, please prepare following test environment.

1) FPGA development board: AC701
2) Test PC.
3) Ethernet cable (Cat5e or Cat6)
4) Micro USB cable for JTAG connection connecting between FPGA board and Test PC.
5) Mini USB cable for UART connection connecting between FPGA 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=115,200, Data=8-bit, Non-parity and Stop=1.
8) Node.js, installed on PC, to run server
9) Demo configuration file (TLS1Gdemo.bit). To download these files, please visit our web site at www.design-gateway.com.

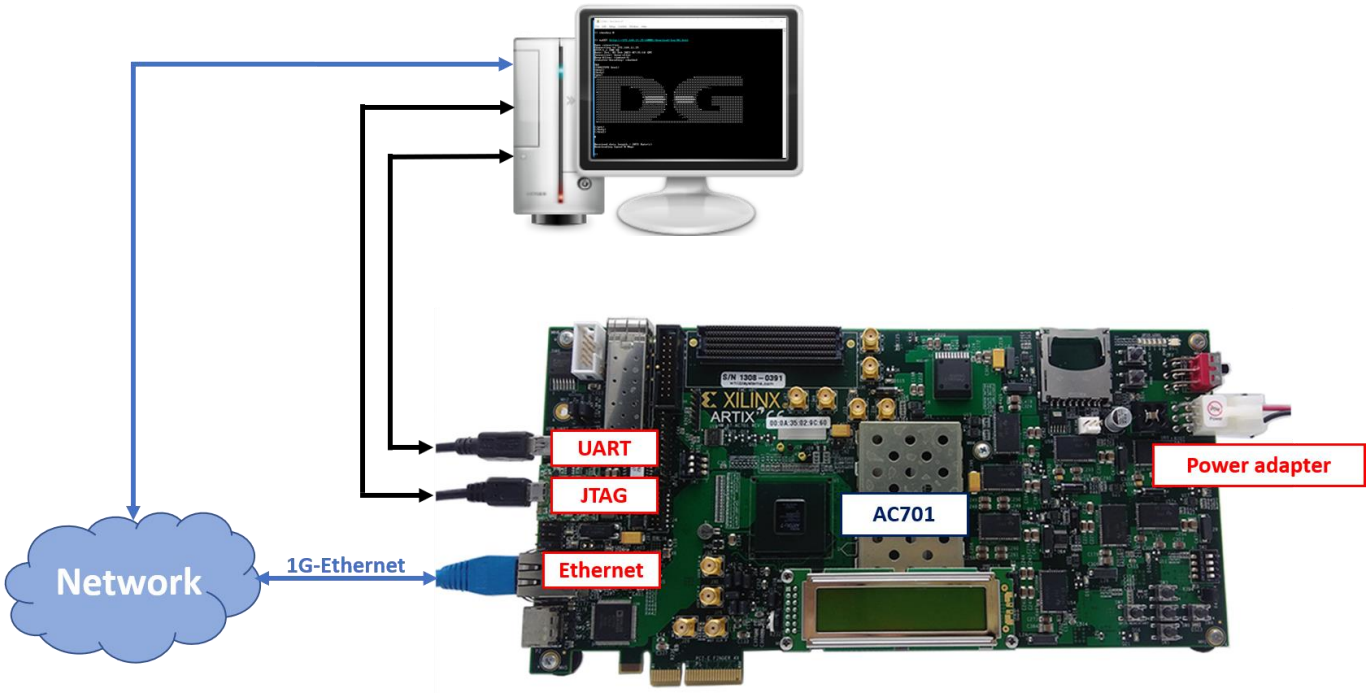

Figure 2-2 TLS1Gdemo environment on AC701 board

2.2 FPGA development board setup

To configure FPGA board, please following steps below,

1) Power off system.
2) Connect micro USB cable and mini USB cable from FPGA board to PC for JTAG programming and USB UART (Serial Console).
3) Connect power supply to FPGA development board.
4) Connect CAT5e or CAT6 cable between RJ45 on FPGA board to network
5) Power on FPGA board.
6) Open Serial console to connect to FPGA board. Serial setting is Baud rate = 115,200, Data=8-bit, Non-parity, and Stop = 1.
7) Open Vivado tool to program FPGA by following steps,
    i) Click open Hardware Manager.
    ii) Open target -> Auto Connect.
    iii) Select FPGA device to program bit file.
    iv) Click Program device. v) Click "…" to select program bit file.
    v) Click Program button to start FPGA Programming.

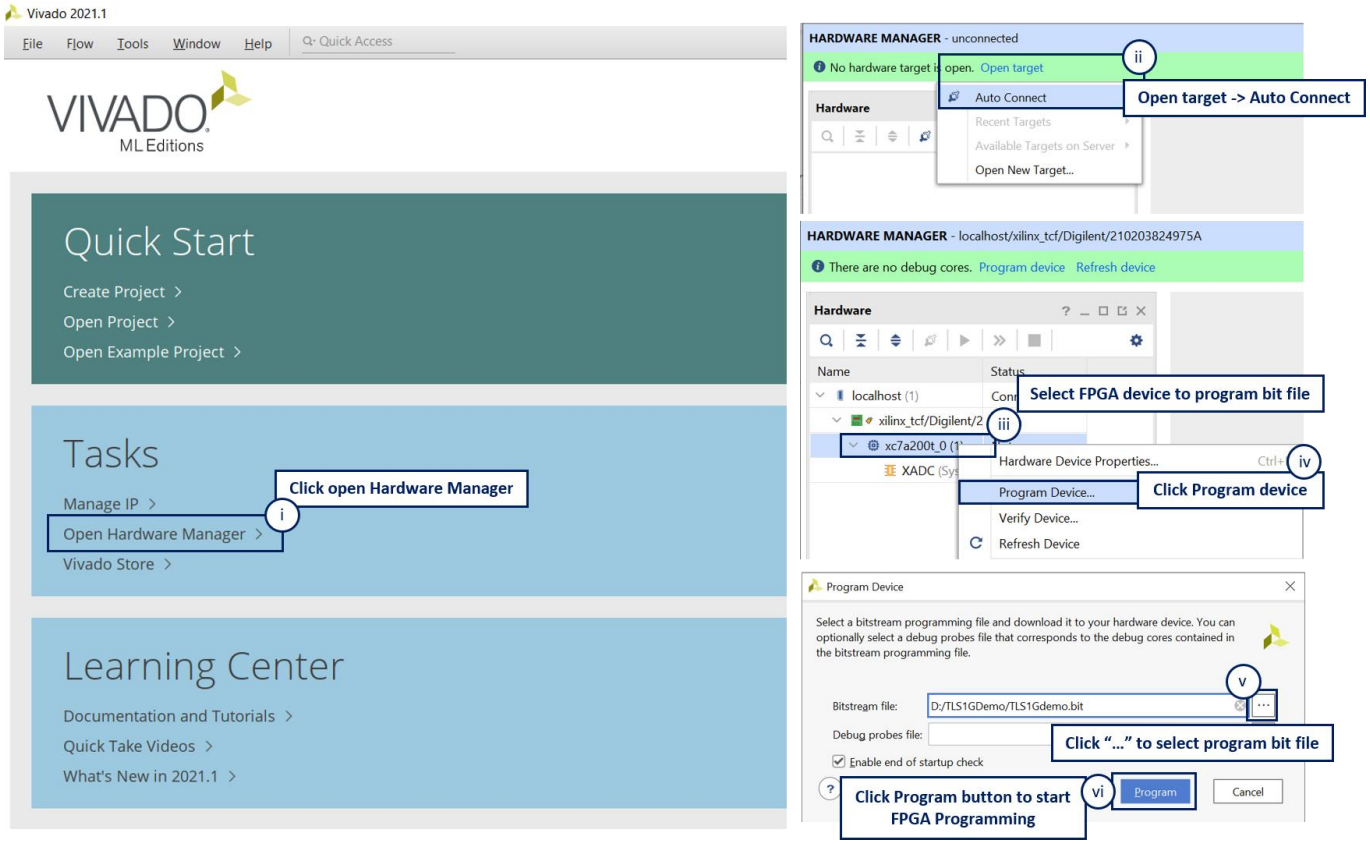

Figure 2-3 Example of programming FPGA board using Vivado tool

3 Node.js server

In this demonstration, a sample server is created using Node.js. The server opens port 60001 for HTTPs connection and port 60002 for HTTP connection. The required files for running the server are provided in ./server which contains the file as follow,

1) serverDemo.js for running server.
2) key.pem and cert.pem as a sample RSA certificate of server.
3) uploadMenu.html for making web browser can upload data to server via POST method.
4) ./log folder for containing resources that are DG.html, bike.html, pinkpanther.html and rex.html. User can add file to ./log folder to be the resource for downloading.

When serverDemo.js is executed, IP address and port number of server are displayed on console as shown in Figure 3-1.

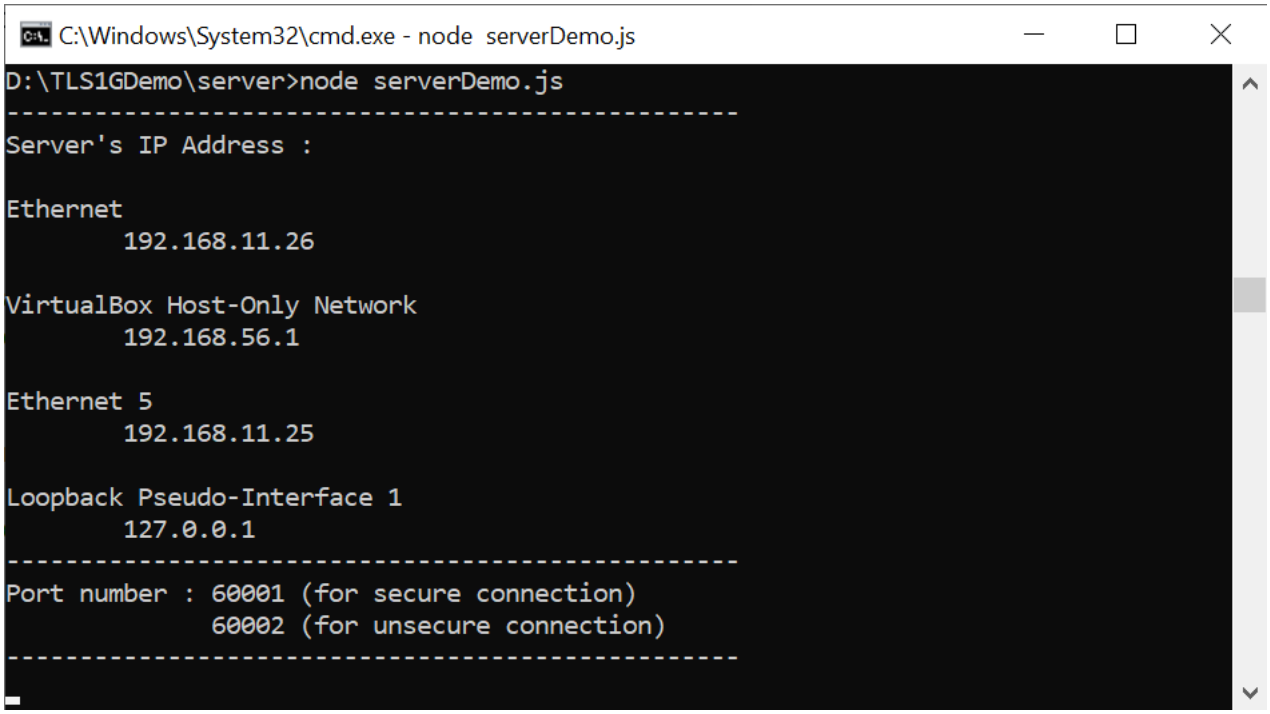

Figure 3-1 Server console when serverDemo.js is executed

Remark

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

1) Allow Node.js port through antivirus firewall setting, if antivirus is installed in the host machine. Figure 3-2 displays an example firewall setting for McAfee.
2) Allow Node.js port through windows firewall as follow,
   i) Go to Windows Defender Firewall
   ii) Click on **Allow an app or feature through windows firewall**
   iii) Search for **Node.js Server Side JavaScript** and mark the boxes both public and private column as shown in Figure 3-3.

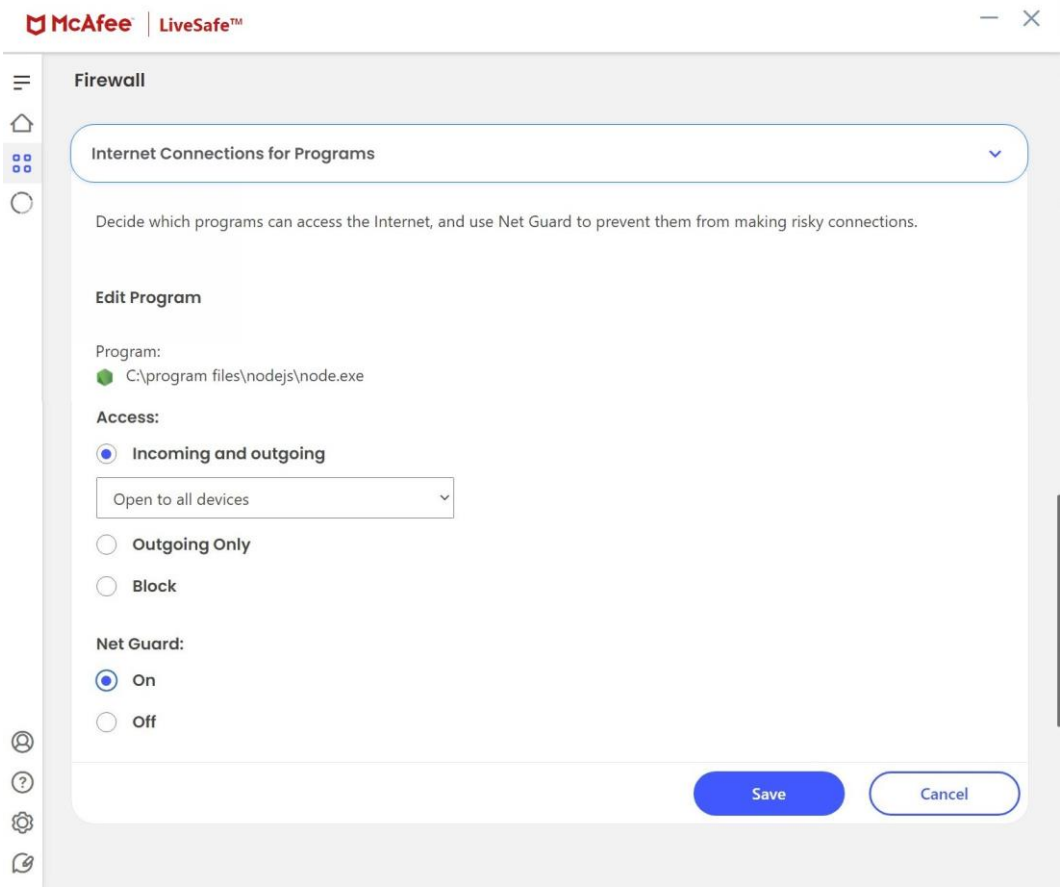

Figure 3-2 McAfee firewall setting

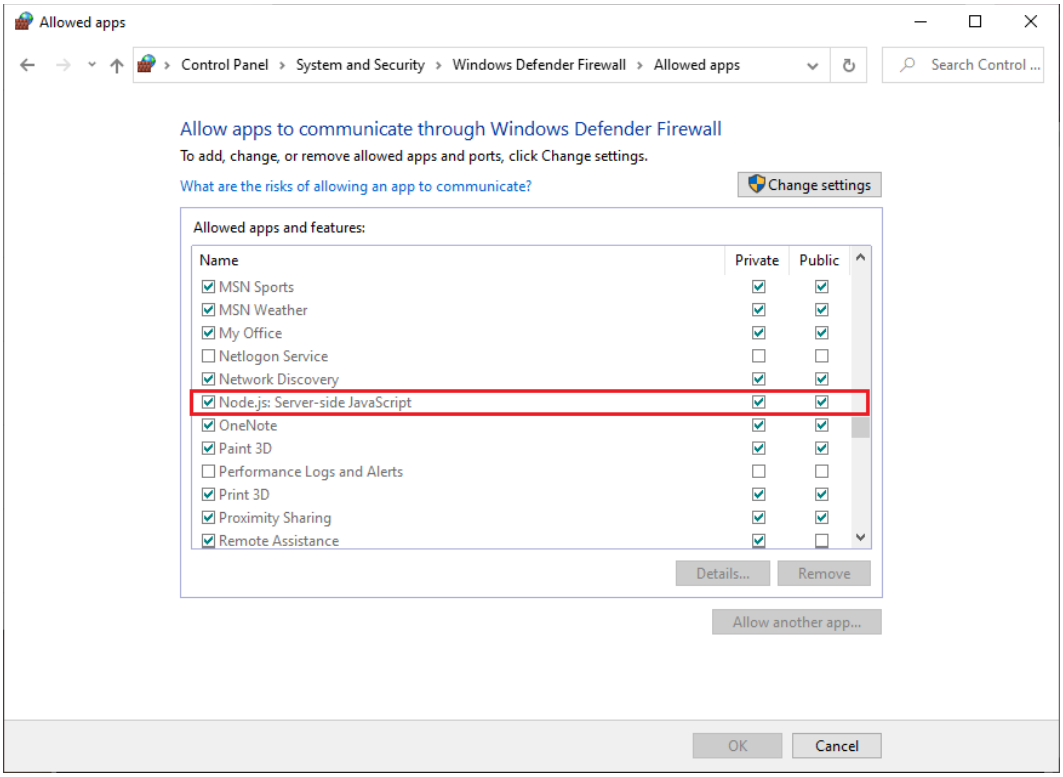

Figure 3-3 Windows firewall setting

Clients can download data patterns or existing files in the ./log folder 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, data pattern and length of requested data are displayed on the server console as shown in Figure 3-4.

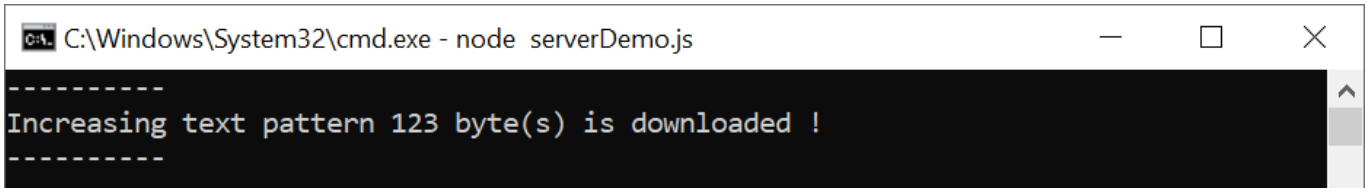

Figure 3-4 Server console when client download data pattern

For downloading html file in ./log folder, when a server receives a GET request, file path of requested data are displayed on the server console as shown in Figure 3-5.

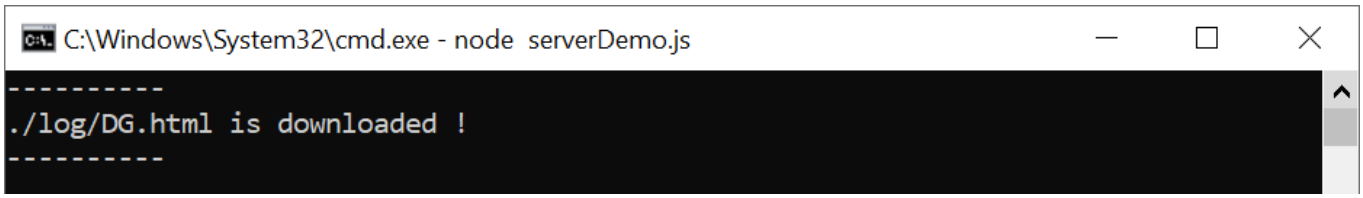

Figure 3-5 Server console when client download ./log/DG.html

Clients can upload data to the server by sending a POST command followed by uploaded data. After completely transferring, received data, length of data and transfer speed are displayed on the server console as shown in Figure 3-6. If data length is more than 16 kB, the server console shows only data length and transfer speed.

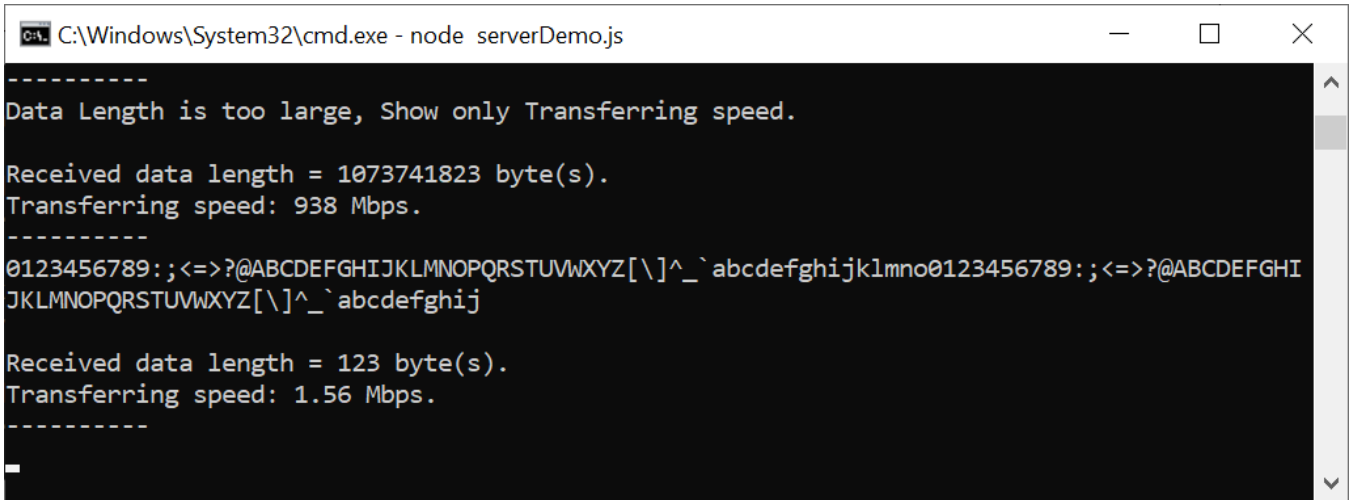

Figure 3-6 Server console when client upload data

4 Web browser

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

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

protocol://ip:port/direction/pattern/length

Where	protocol	represent http for unsecure connection or https for secure connection
	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 that user want to download or upload
	length	represent data length in byte

For example, server's IP address is 192.168.11.26, port number for secure connection is 60001 and the user's URL is https://192.168.11.26:60001/download/t0/123. Secure connection is established, the 123-byte decreasing text pattern is displayed in the web browser as shown in Figure 4-1.

Figure 4-1 Decreasing text pattern shown in web browser

Remark

- Our tested web browser is Google Chrome.
- The RSA certificate used in this demonstration is a self-signed certificate that was not issued by a certification authority (CA). When accessing the server, the web browser may display a "Not Secure" alert.
- The certificate length for this demonstration is limited to a maximum of 2 KB.

For example, server's IP address is 192.168.11.26, port number for secure connection is 60001 and the user's URL is https://192.168.11.26:60001/download/t1/456. Secure connection is established, the 456-byte increasing text pattern is displayed in the web browser as shown in Figure 4-2.

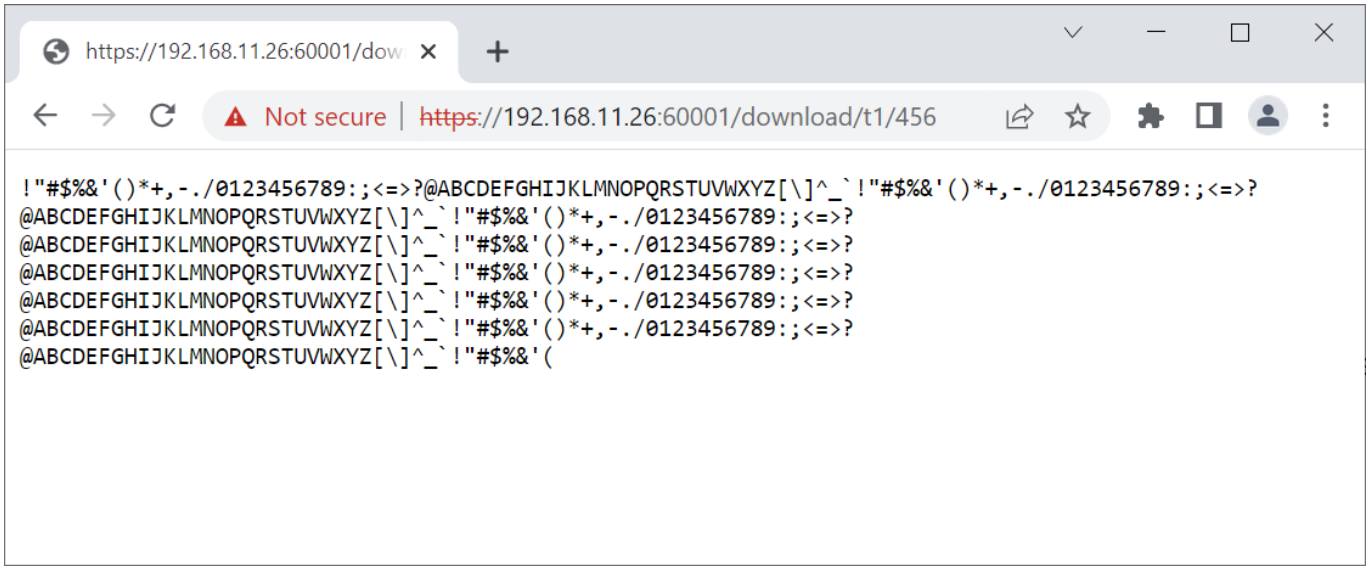

Figure 4-2 Increasing text pattern shown in web browser

In case of downloading binary pattern, **Save as** dialog window appears. User can save file and view the binary data after downloading process is done.

For downloading existing files in ./log folder, user can input URL in the following format,

protocol://ip:port/download/log/filename

For example, server's IP address is 192.168.11.26, port number is 60001 for secure connection and 60002 for unsecure connection.

When user inputs https://192.168.11.26:60001/download/log/DG.html and DG.html exists in log folder. The secure connection is established, the html page is downloaded and displayed on the web browser as shown in Figure 4-3.

When user input's URL is http://192.168.11.26:60002/download/log/bike.html, the unsecure connection is established. The html page is downloaded and displayed on the web browser as shown in Figure 4-4.

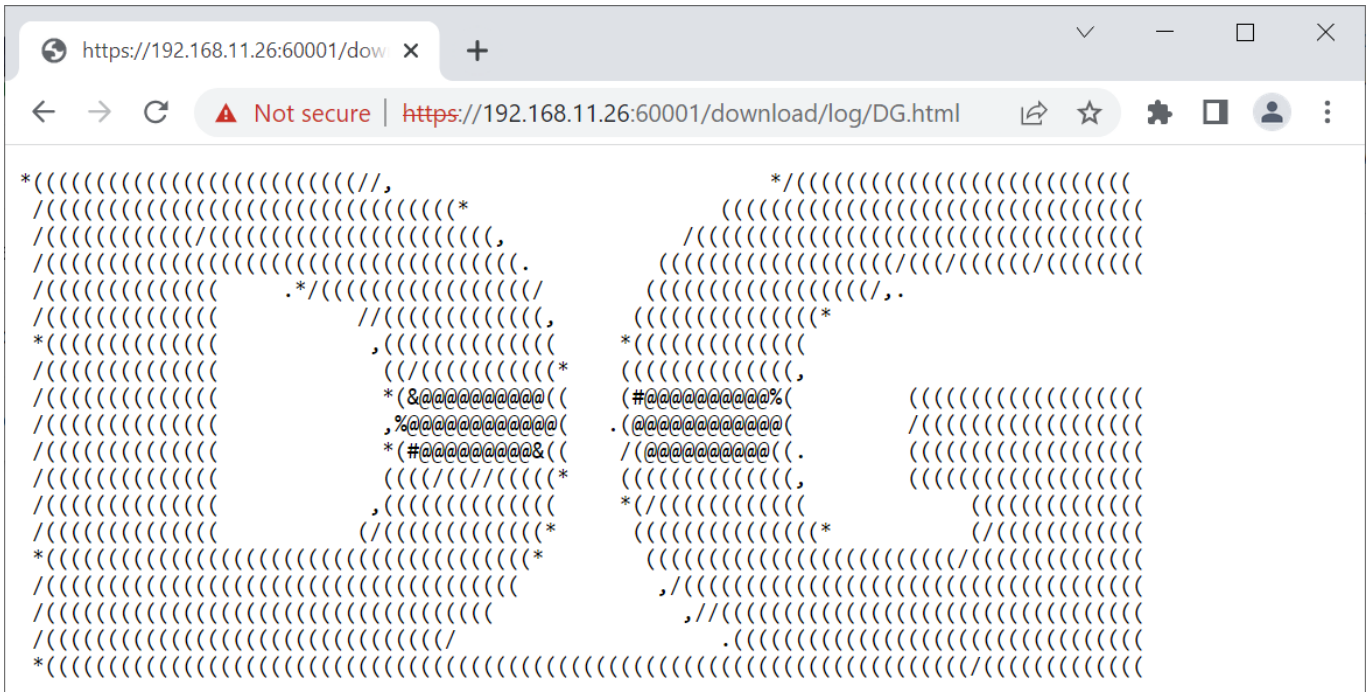

Figure 4-3 DG.html shown in web browser

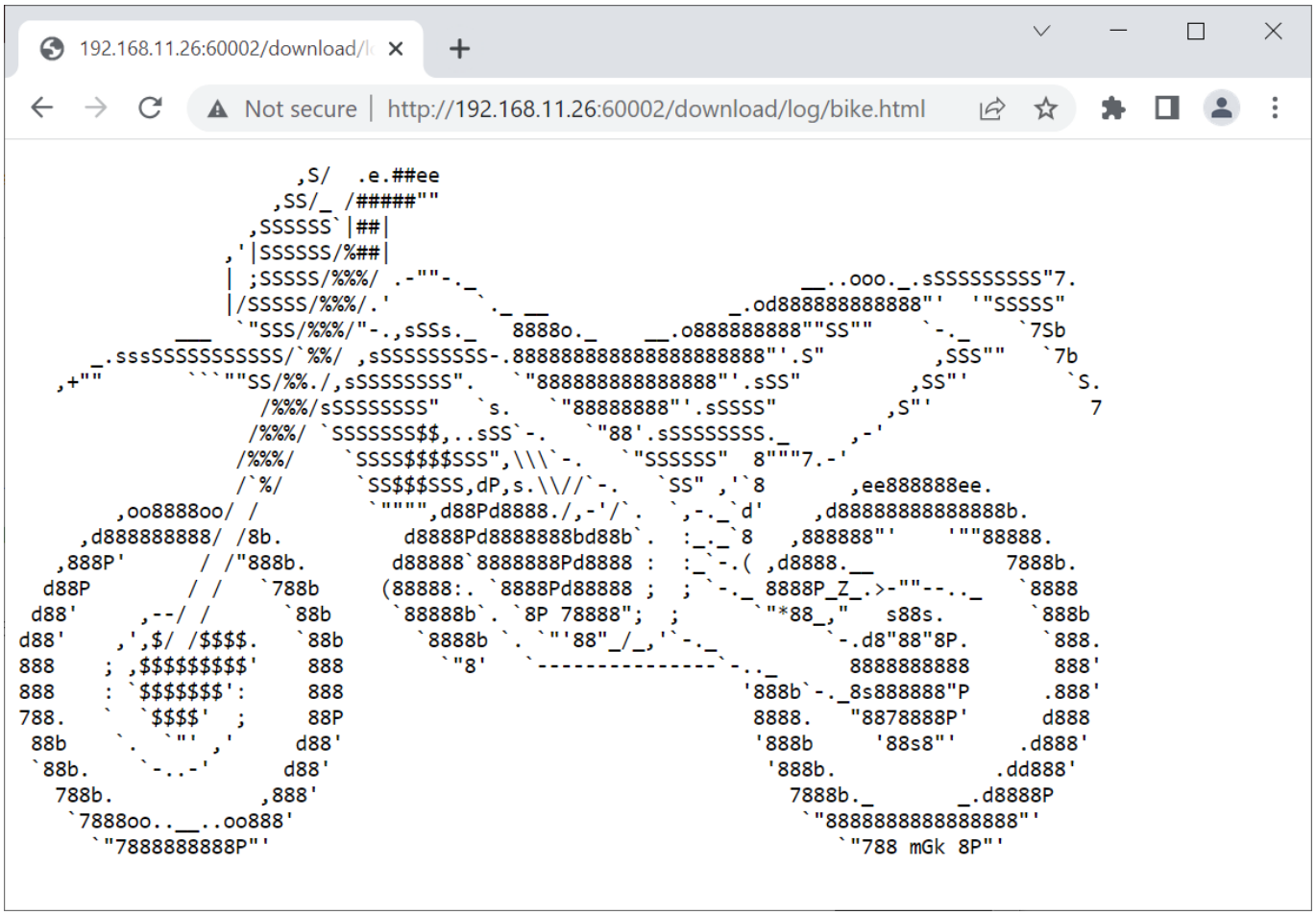

Figure 4-4 bike.html shown in web browser

For unsecure uploading data, a user has to request uploadMenuHTTP.html from http://192.168.11.26:60002/upload/menu to generate data pattern and upload to the server via POST method. Upload menu is displayed in the web browser as shown in Figure 4-5. Users can choose data pattern and data length. Html page will prepare data and send POST command following by data pattern to the server when “POST” button is pressed. When uploading is completed, if the length of data is less than 16 kB, the data, length and transfer speed are displayed on server console as shown in Figure 4-6.

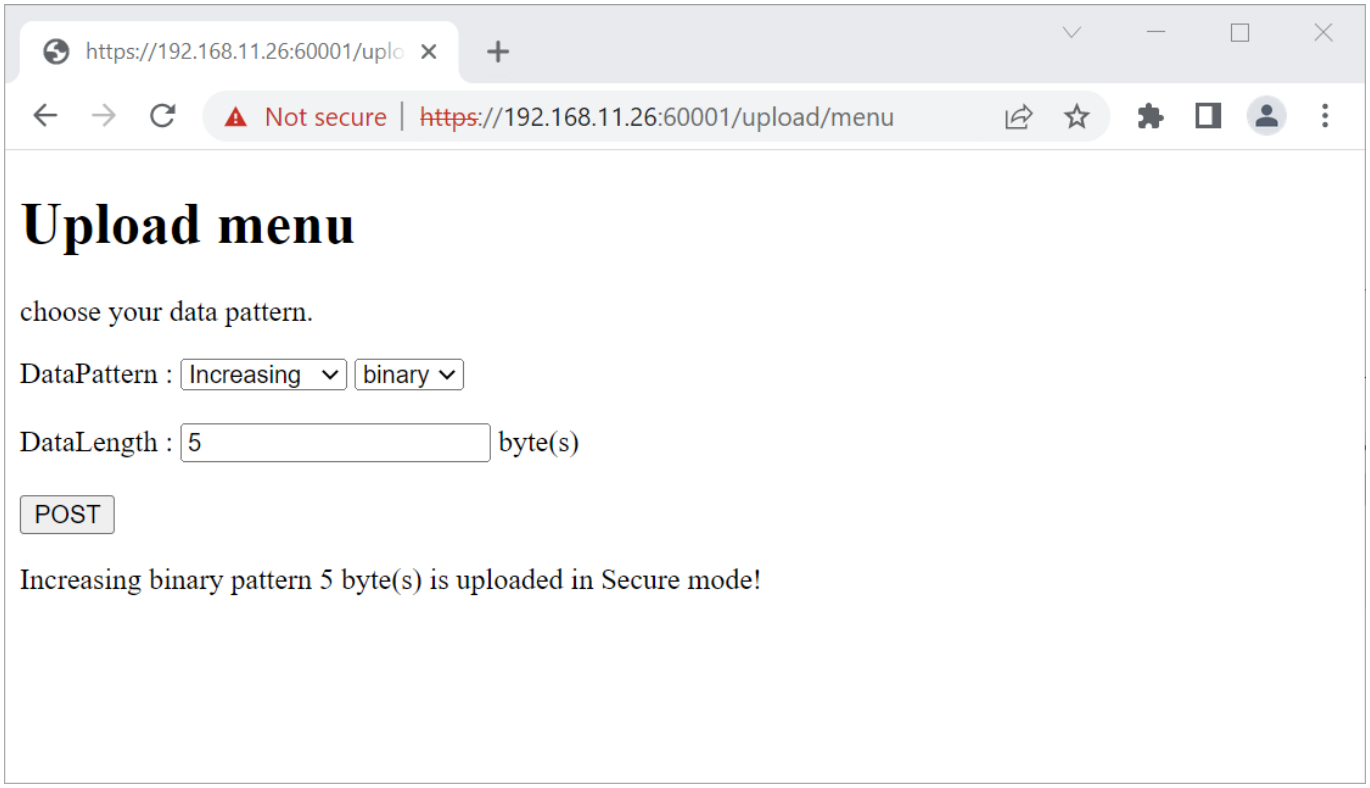

Figure 4-5 Unsecured upload page

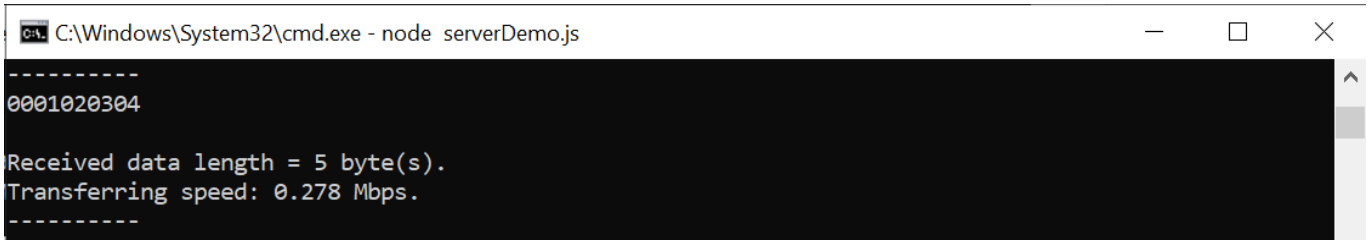

Figure 4-6 Server’s console when client upload data that is less than 16kB.

In the same way, a user can secure upload data by requesting uploadMenuHTTPs.html from https://192.168.11.26:60001/upload/menu. Upload menu is displayed in the web browser as shown in Figure 4-7. Users can choose data pattern and data length. Html page will prepare data and send POST command following by data pattern to the server when “POST” button is pressed. Because the length of data is greater than or equal to 16 kB, when uploading is completed, only data length and transfer speed are displayed on server console as shown in Figure 4-8.

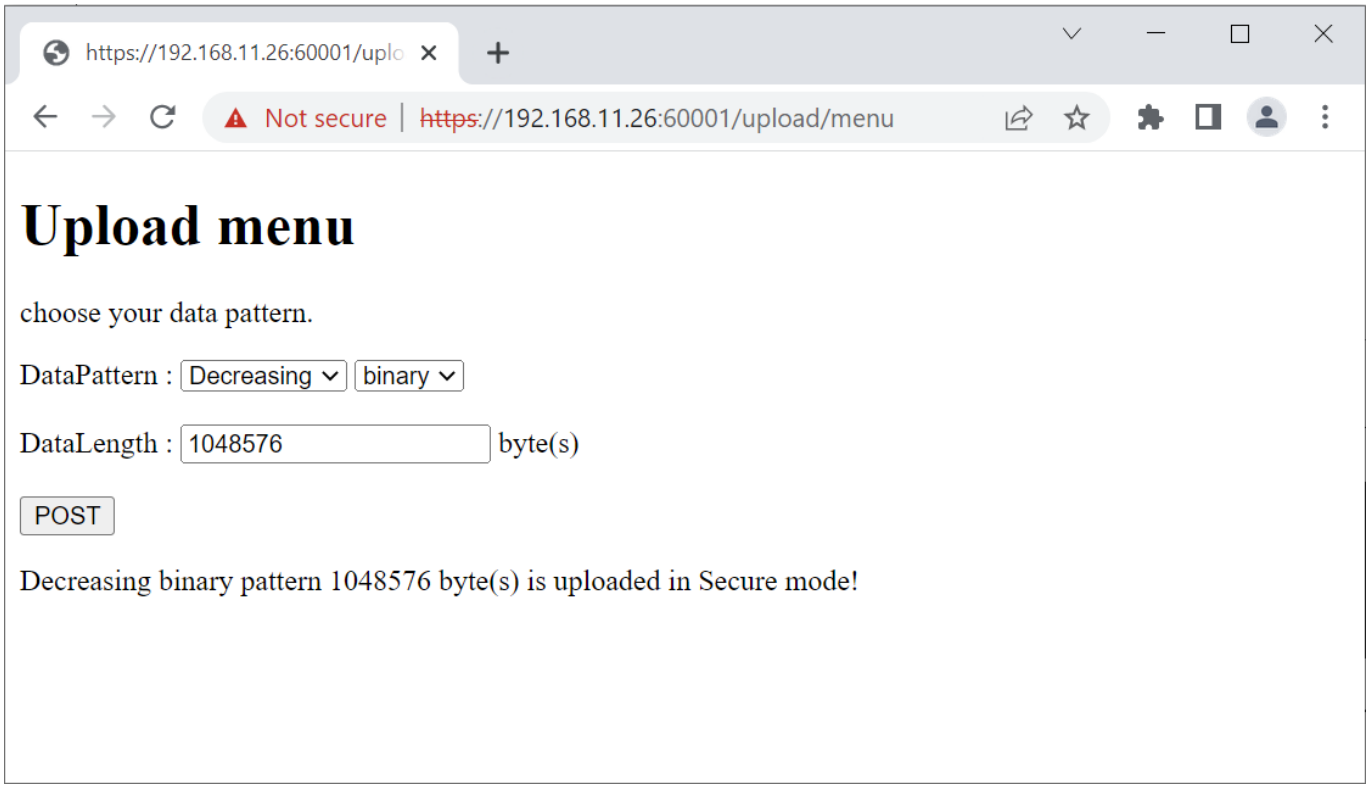

Figure 4-7 Secured upload page

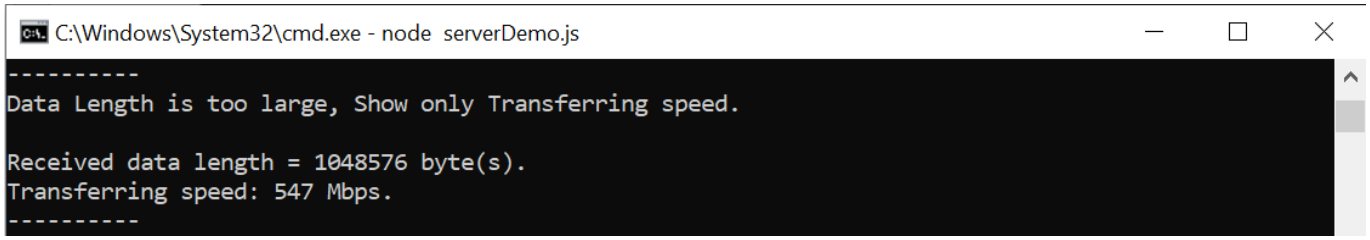

Figure 4-8 Server’s console when client upload data that is greater than or equal to 16kB

5 TLS1GDemo

TLS1Gdemo is designed to establish connection between user and server. The connection can be secure (HTTPs) or unsecure (HTTP).

For secure connection, TLS1Gdemo implements TLS1.3 protocol. Client and server exchange ephemeral key, derive key used to encrypt and decrypt packet data in handshake phase and data transfer phase and verify server's certificate before transferring encrypted data.

TLS1Gdemo supports X25519 for key exchange, Hash-based Key Derivation Function (HKDF) with SHA-384 for deriving keys, AES-256-GCM for encryption/decryption and RSA for certificate verification.

For this demonstration, users can set the IP address, port number and MAC address of FPGA board, enable hardware, enable showkey mode, download and upload data by using the following command as below.

1. 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 192.168.11.42.

2. setport ddddd

This command is used to set the static port number of FPGA in decimal format. By default, FPGA's port number is set to be dynamic. Dynamic ports are in the range 49152 to 65535. User can enable dynamic port again after specifying a port number by using `setport dynamic` command.

3. 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 00-01-02-03-04-05.

4. sethw <1: enable, 0: disable>

This command is used to enable hardware for handling the connection. By default, hardware is enabled. When user disables hardware, all operation of TLS1.3 is handled with firmware only.

5. showkey <1: enable, 0: disable>

This command is used to enable showkey mode. When showkey mode is enabled, the TLS traffic ticket, session keys and IVs for encryption/decryption is displayed on the serial console as shown in Figure 5-1. User can use the TLS traffic ticket as (Pre)-Master-Secret log file for Wireshark* to decrypt transferred data between TLS1Gdemo and server.

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

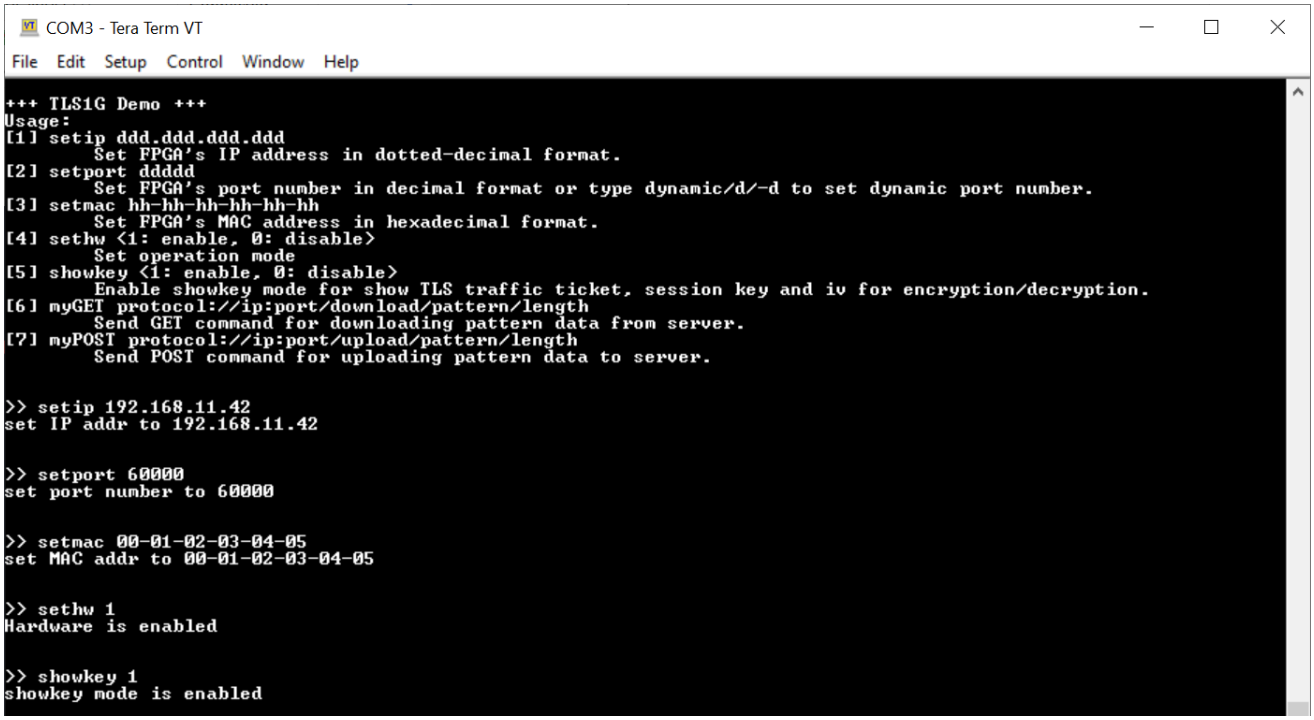

Figure 5-1 Parameter setting

Because client and server encrypt transferred data with different session keys and session keys in the handshake phase are different within the data transfer phase. The session keys and IVs of each sender at each phase is shown in Figure 5-2.

tkchs_key/tkchs_iv and tkshs_key/tkshs_iv represent client's key/iv and server's key/iv for handshake phase, respectively. tkcapp_key/tkcapp_iv and tksapp_key/tksapp_iv represent client's key/iv and server's key/iv for data transfer phase, respectively.

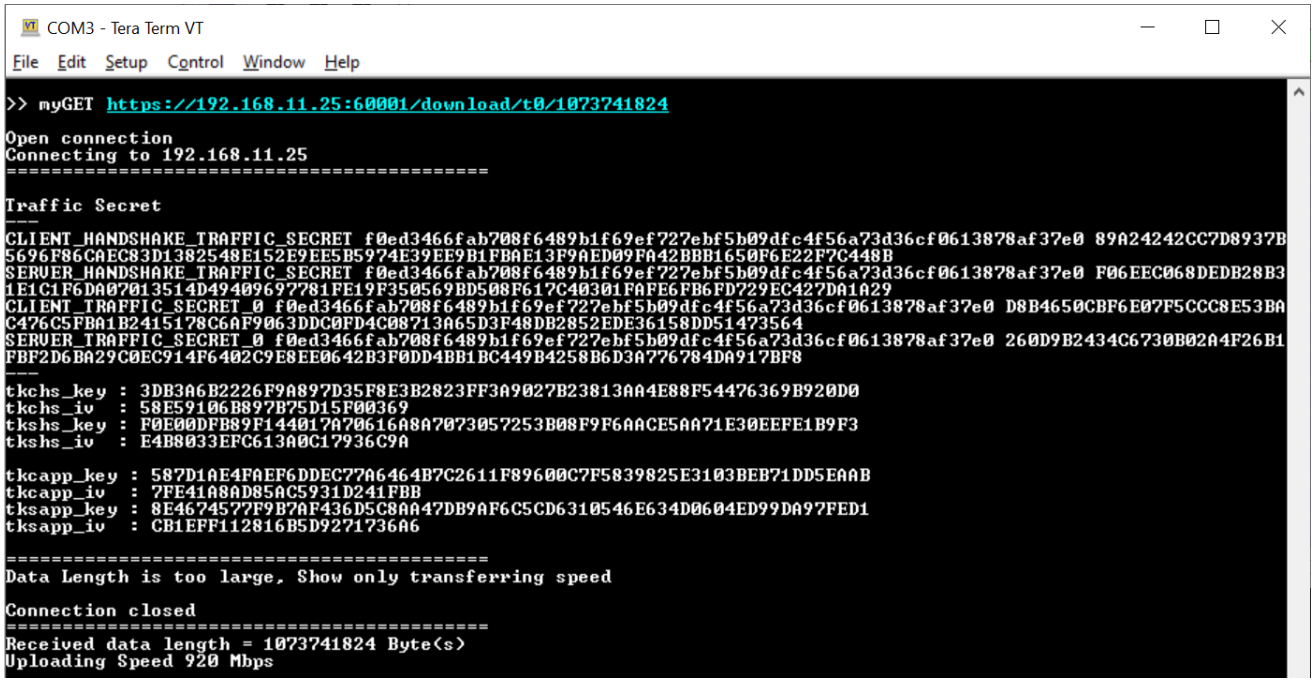

Figure 5-2 Serial console when downloading data more than 16KB

6. myGET protocol://ip:port/download/pattern/length

This command simulates GET method of HTTP to download data from the server. User can input URL and then received data is displayed on the serial console.

If hardware is enabled, user can download data pattern up to 2GB for secure and unsecure connection. If hardware is disabled, CPU takes long time to encrypt/decrypt data in secure connection. So, the maximum data length is limited at 1 MB for secure connection and 2 GB for unsecure connection.

As shown in Figure 5-4, DG.html is downloaded from the server and displayed on the serial console as displayed on the web browser shown in Figure 4-3.

In case of downloaded data length is more than 16 kB, “Data Length is too large, Show only Transferring speed” is shown instead of received data as shown in Figure 5-3.

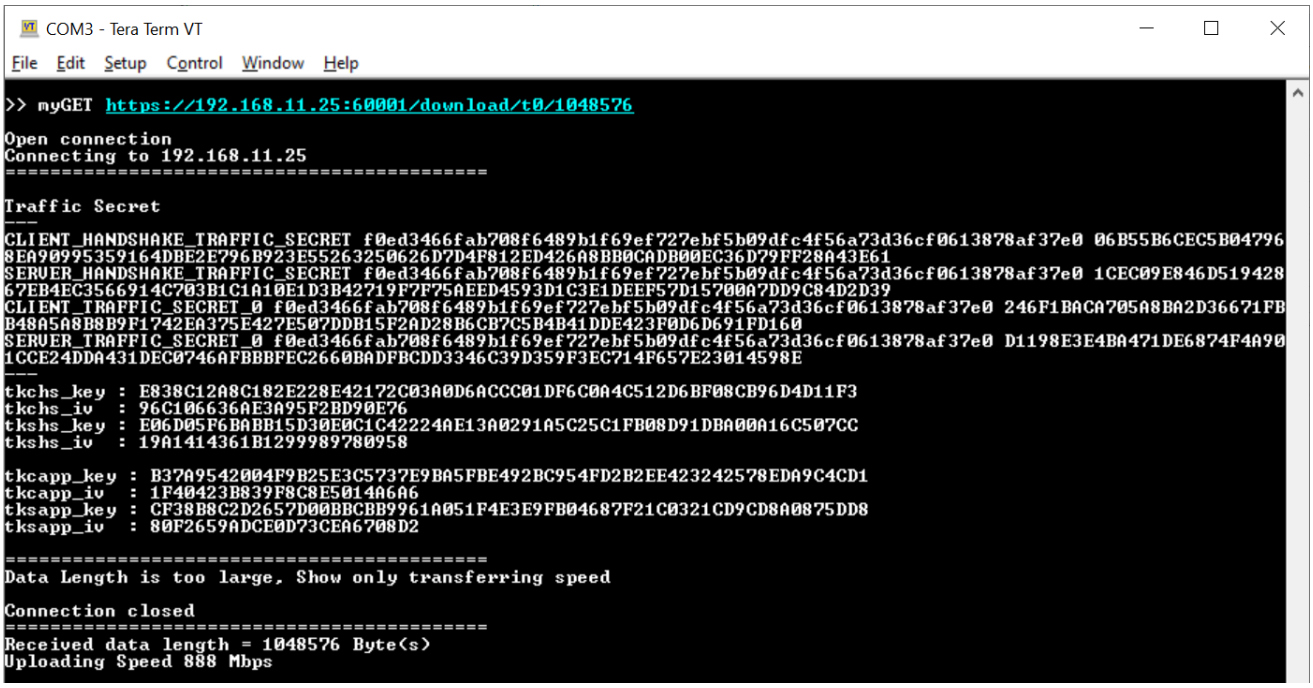

Figure 5-3 Serial console when downloading 1MB data

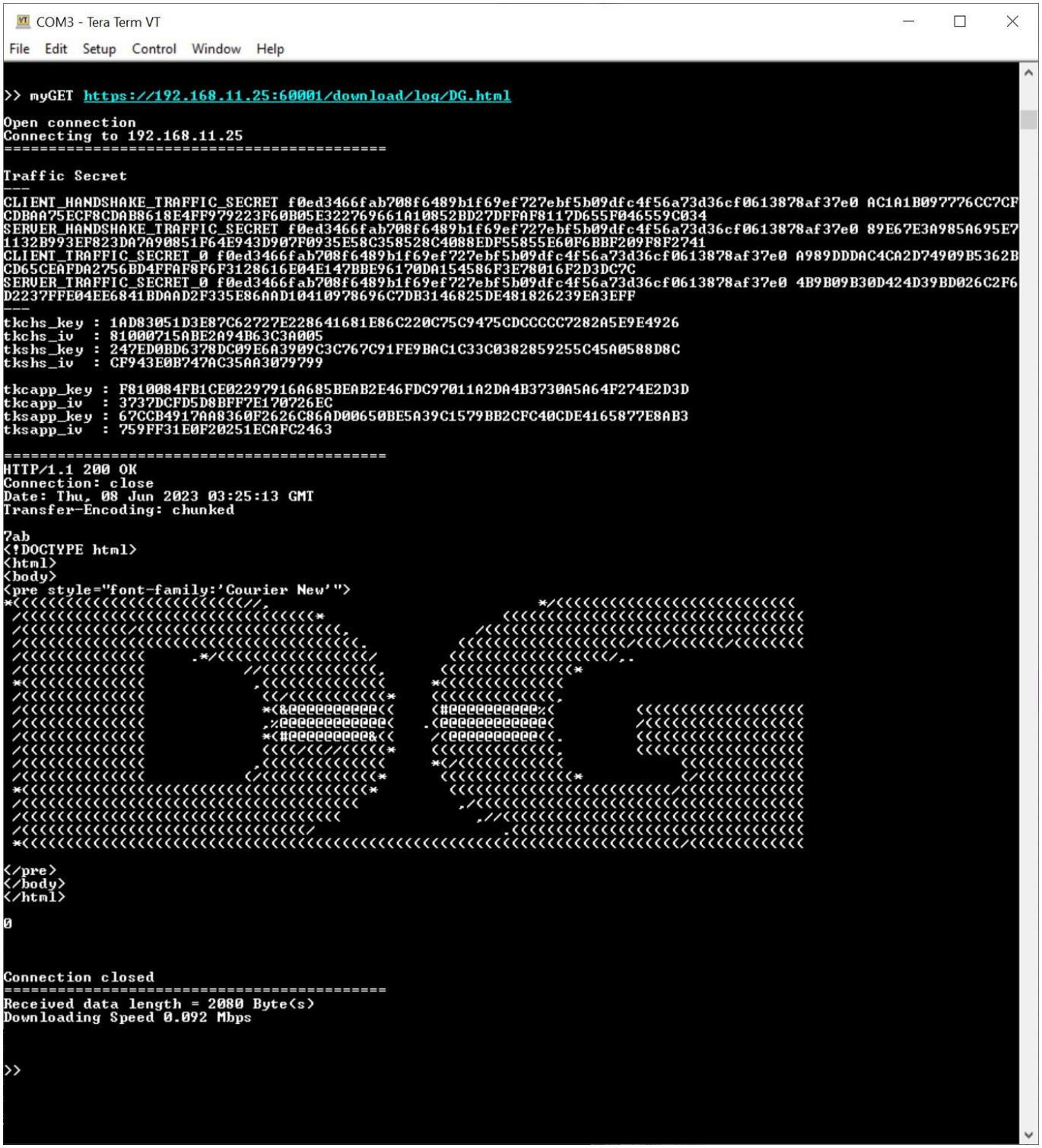

Figure 5-4 Serial console when downloading DG.html

7. myPOST protocol://ip:port/upload/pattern/length

This command simulates POST method of HTTP to upload data to the server. User can indicate data pattern and data length in URL. After uploading is done, data length and uploading speed is displayed as shown in Figure 5-5 and Figure 5-6. On server's console, the number of received data from TLS1Gdemo and transfer speed is displayed. In case of the data length is less than 16 kB, the received data is also displayed as shown in Figure 5-7.

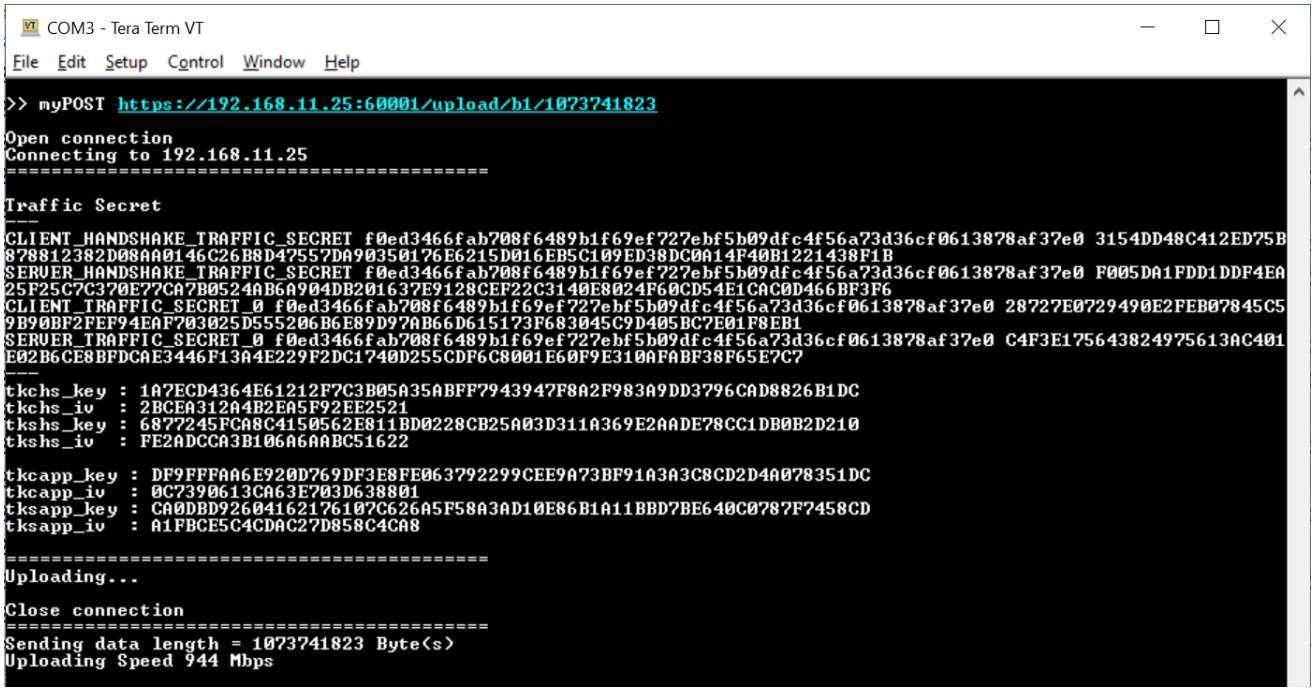

Figure 5-5 Serial console when uploading 1M-byte data

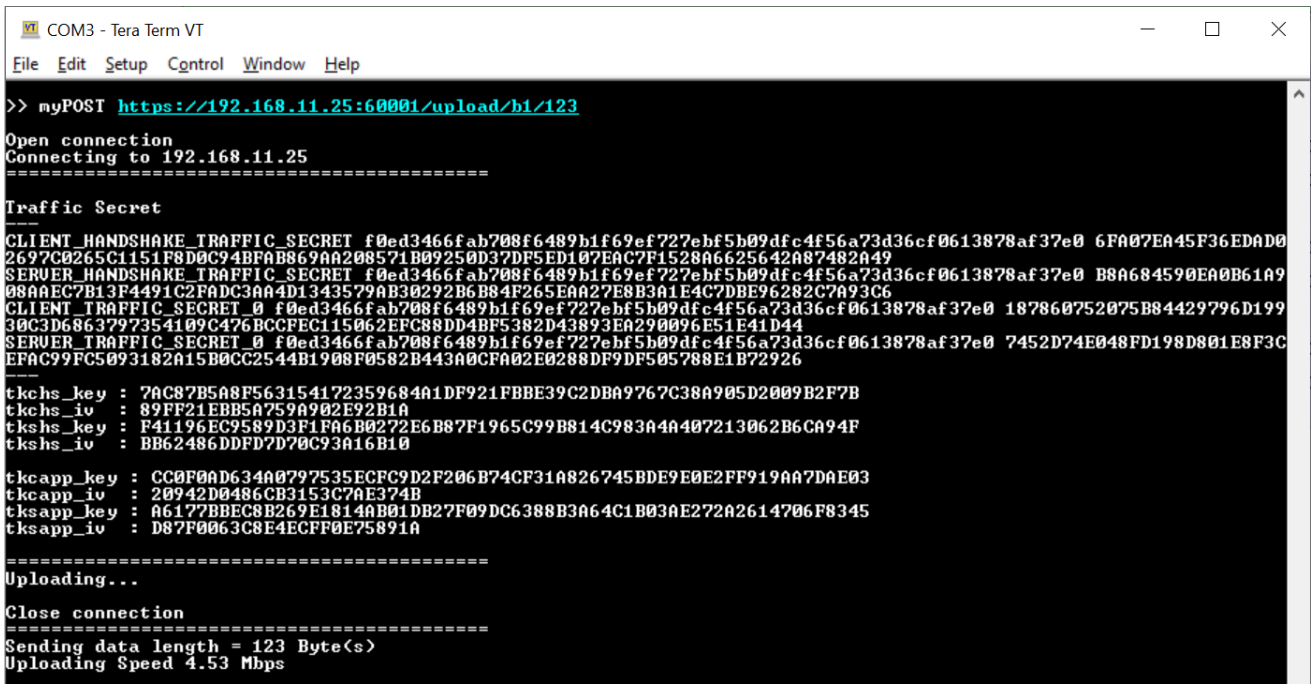

Figure 5-6 Serial console when uploading 123-byte data

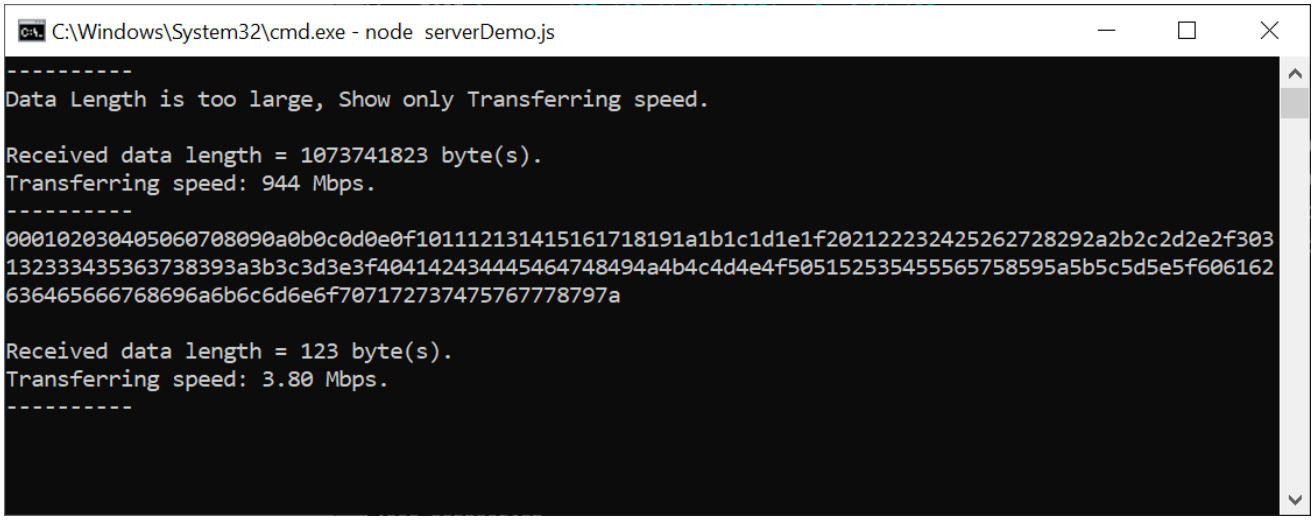

Figure 5-7 Server console when uploading data

6 FPGA resource and Performance

TLS1Gdemo implements TLS1.3 over TCP/IP offload engine on AC701 at 125 MHZ working with a Microblaze processor operating at 100 MHz. Table 6-1 shows resource usage of TLS1Gdemo on AC701. By hardware-accelerated, the throughput of secure communication is unaffected by handling TLS protocol.

Table 6-1 Resource usage of TLS1Gdemo on AC701

Name	Slice LUTs	Slice Registers	Slice	Block RAM
TLS1GCPUtest (Total)	19159	12681	6108	188
• System	2076	1776	825	128
• TCP/IP	3257	3654	1311	37.5
• TLS1.3 : AES256GCM	6305	1943	2044	0
• TLS1.3 : ModularMultipler	1290	280	361	0
• TLS1.3 : SHA256	1251	792	385	1
• TLS1.3 : SHA384	2212	1542	624	2

Table 6-2 displays the transfer speed between the web browser and AC701 when transferring data with the sample server over unsecure connection. Table 6-3 displays the transfer speed between the web browser AC701 without hardware-accelerated (handle TLS1.3 by firmware) and AC701 with hardware-accelerated (handle TLS1.3 by hardware) when transferring data with the sample server over secure connection. Monitoring the transfer speed between the web browser and server using the task manager is not precise, especially when transferring small amounts of data. So, the transfer speed for transferring 1 MB data is considerably slow, while the transfer speed for transferring 1 GB data is almost 1 Gbps.

According to the overhead time in network protocol, the throughput for transferring small data is falling off. To achieve the maximum throughput, the size of transferring data must be large. As shown in Table 6-2, the transfer speed between the sample server and AC701 with TCP/IP offload engine for transferring 1 GB data is almost 1Gbps and is dropped for transferring 1 MB-data.

For secure connection, client has to handle cryptographic algorithm for handshaking and transferring data. In case of high-performance controller, the web browser is able to handle the connection with throughput nearly by 1Gbps and the utilization of the Intel i7 CPU is approximately 10%, as monitored by the PC's task manager. In case of low-performance controller such as Microblaze in AC701, Microblaze is not able to handle TLS1.3 protocol to achieve 1 Gbps throughput. As shown in Table 6-3, transfer speed between server and AC701 while handling TLS1.3 with firmware is dramatically decreased. Enabling hardware in TLS1Gdemo not only recovers transfer speed to achieve nearly 1 Gbps but also is an offload engine to allow CPU handle another task.

Table 6-2 Transfer speed in unsecure connection between the sample server and clients

Client	Data size	Downloading speed	Uploading speed
Web browser	1 MB	17.5 Mbps*	17.5 Mbps*
	1 GB	972 Mbps*	981 Mbps*
AC701	1 MB	896 Mbps	776 Mbps
	1GB	938 Mbps	940 Mbps

Table 6-3 Transfer speed in secure connection between the sample server and clients

Client	Data size	Downloading speed	Uploading speed
Web browser	1 MB	17.5 Mbps*	17.5 Mbps*
	1 GB	970 Mbps*	977 Mbps*
AC701 with firmware	1 MB	0.161 Mbps	0.185 Mbps
	1 GB	0.160 Mbps	0.160 Mbps
AC701 with hardware	1 MB	888 Mbps	728 Mbps
	1 GB	920 Mbps	928 Mbps

* Approximately transfer speed monitoring by task manager on PC (intel i7-11700K@3.6GHz)

7 Revision History

Revision	Date	Description
1.01	21-Jun-2023	Update remark for firewall setting and serial console capture
1.00	21-Mar-2023	Initial version release