

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 through 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

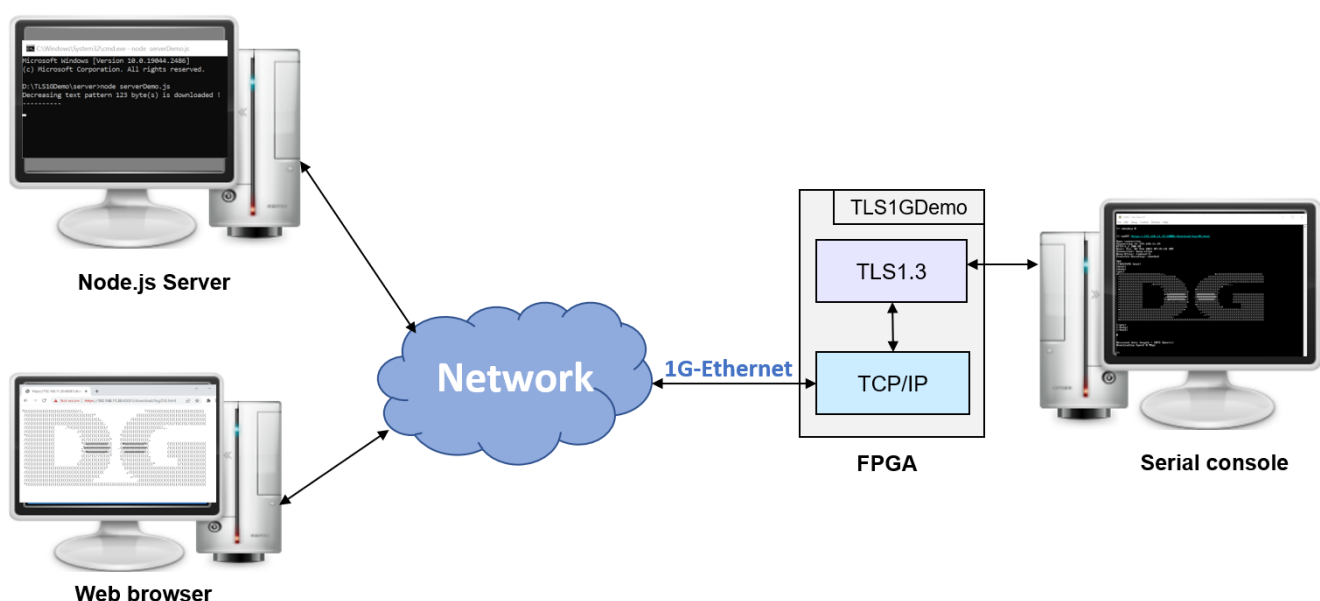


Figure 2-1 System overview

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.

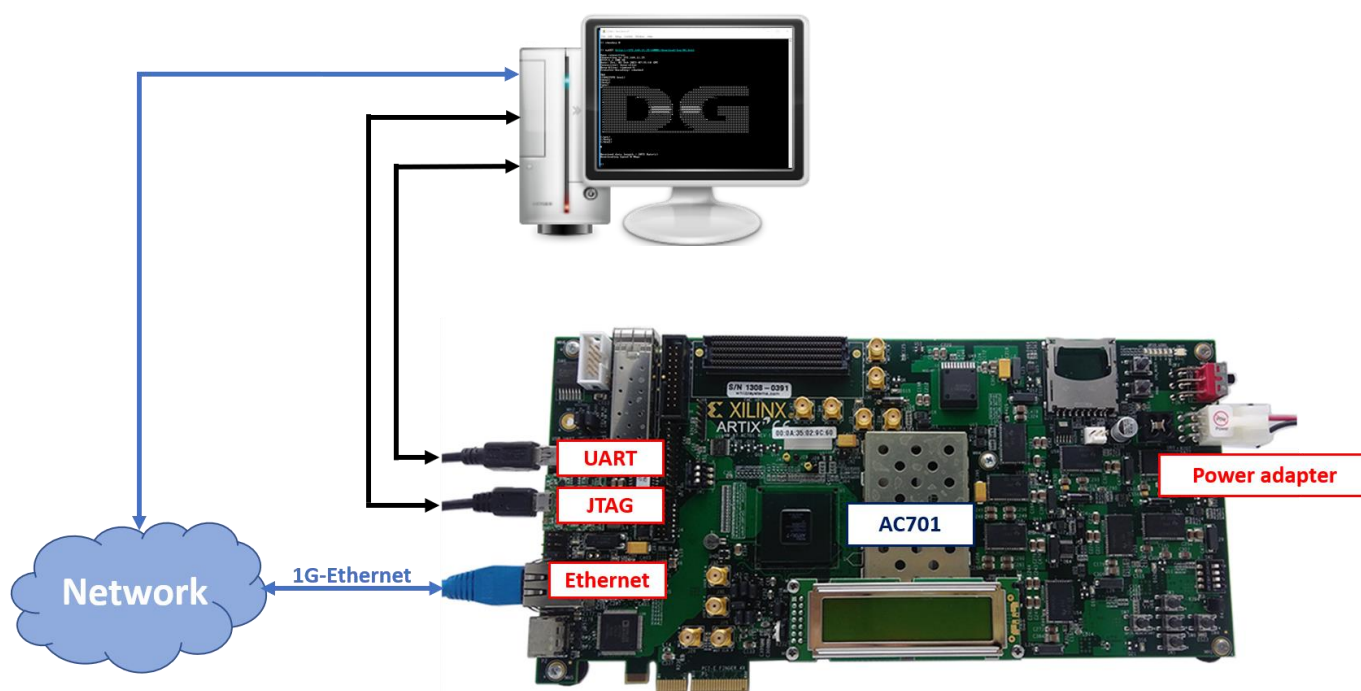


Figure 2-2 TLS1Gdemo environment on AC701 board

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.

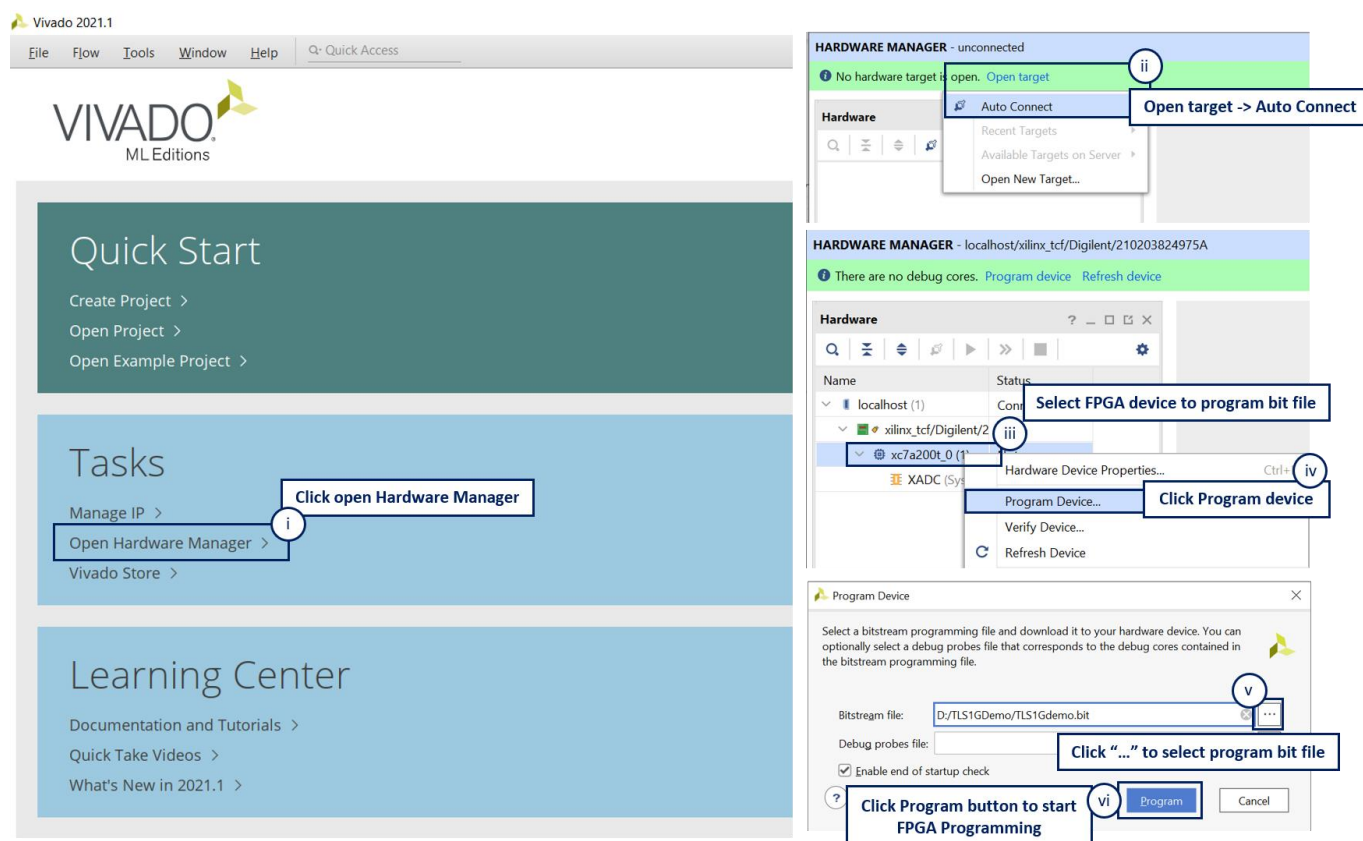


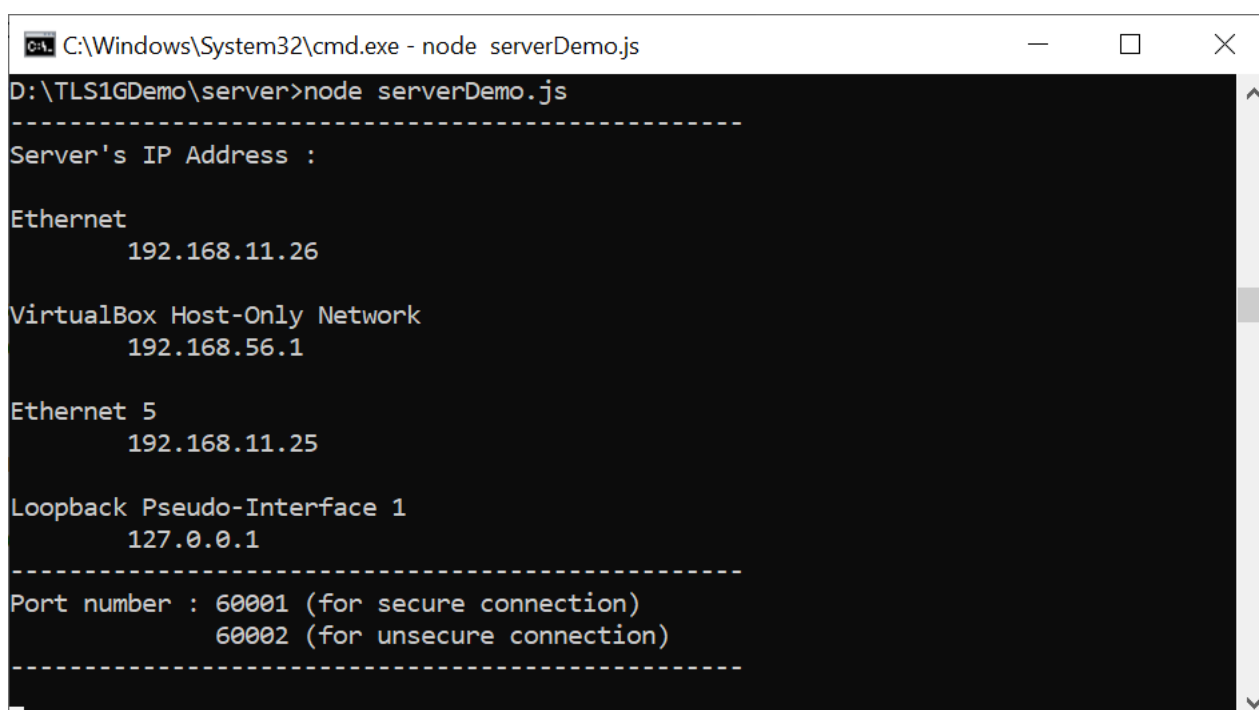
Figure 2-3 Example of programming FPGA board using Vivado tool

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.



```

C:\Windows\System32\cmd.exe - node serverDemo.js
D:\TLS1GDemo\server>node serverDemo.js
-----
Server's IP Address :

Ethernet
    192.168.11.26

VirtualBox Host-Only Network
    192.168.56.1

Ethernet 5
    192.168.11.25

Loopback Pseudo-Interface 1
    127.0.0.1
-----
Port number : 60001 (for secure connection)
              60002 (for unsecure connection)
-----
  
```

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.

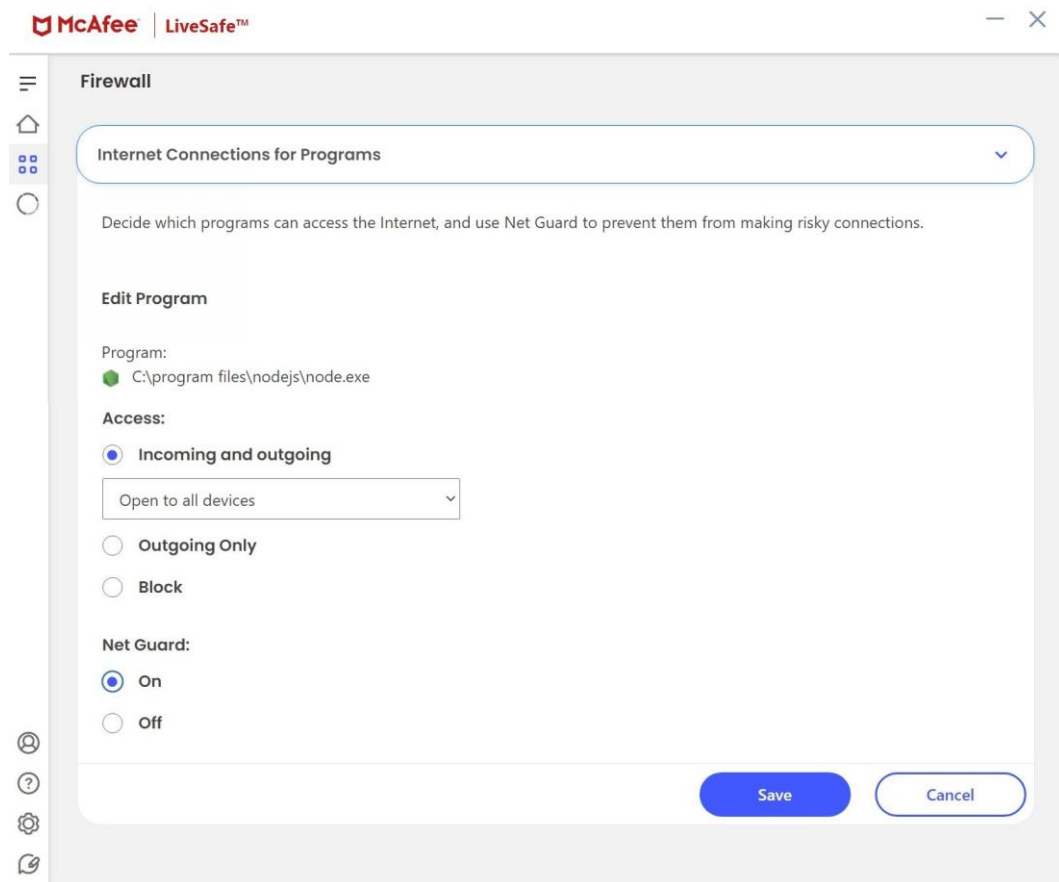


Figure 3-2 McAfee firewall setting

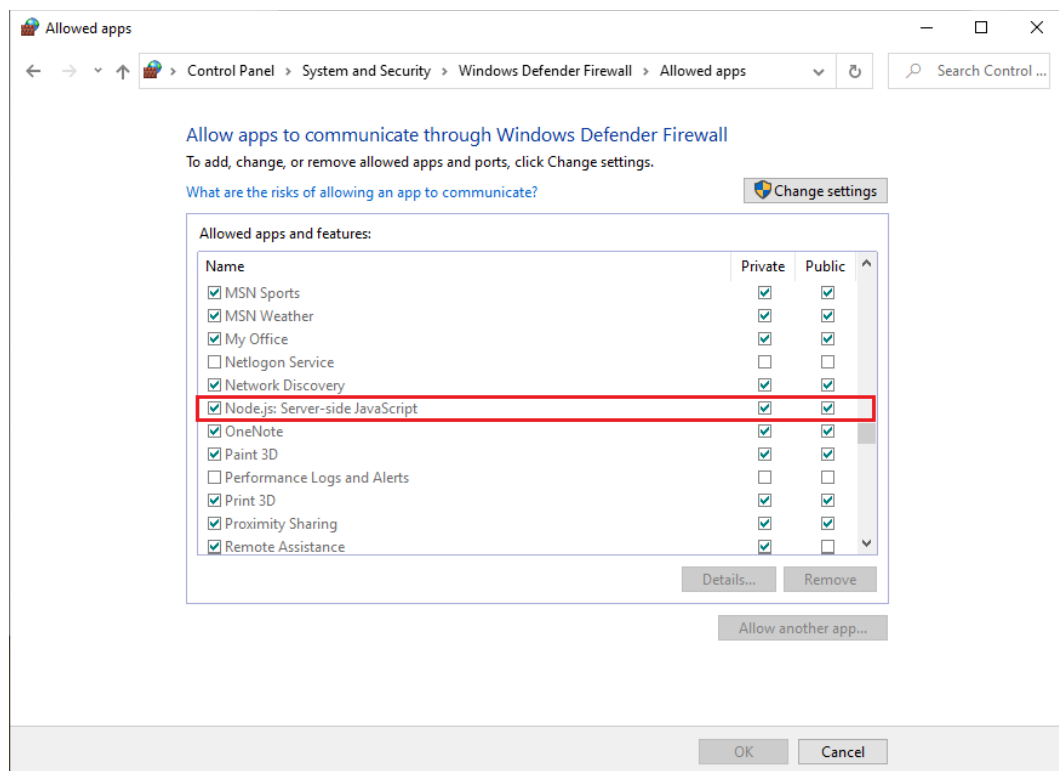
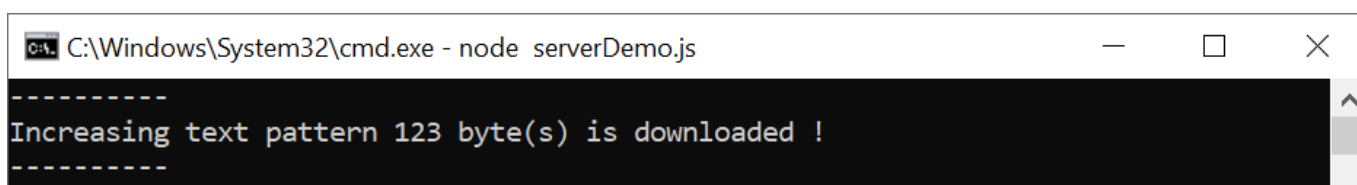


Figure 3-3 Windows firewall setting

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.



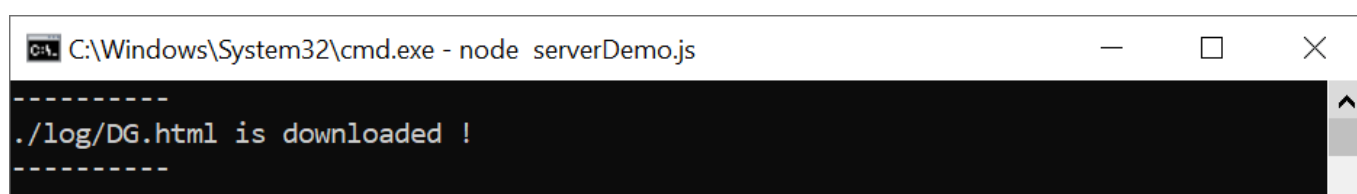
```

C:\Windows\System32\cmd.exe - node serverDemo.js
-----
Increasing text pattern 123 byte(s) is downloaded !
-----

```

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.



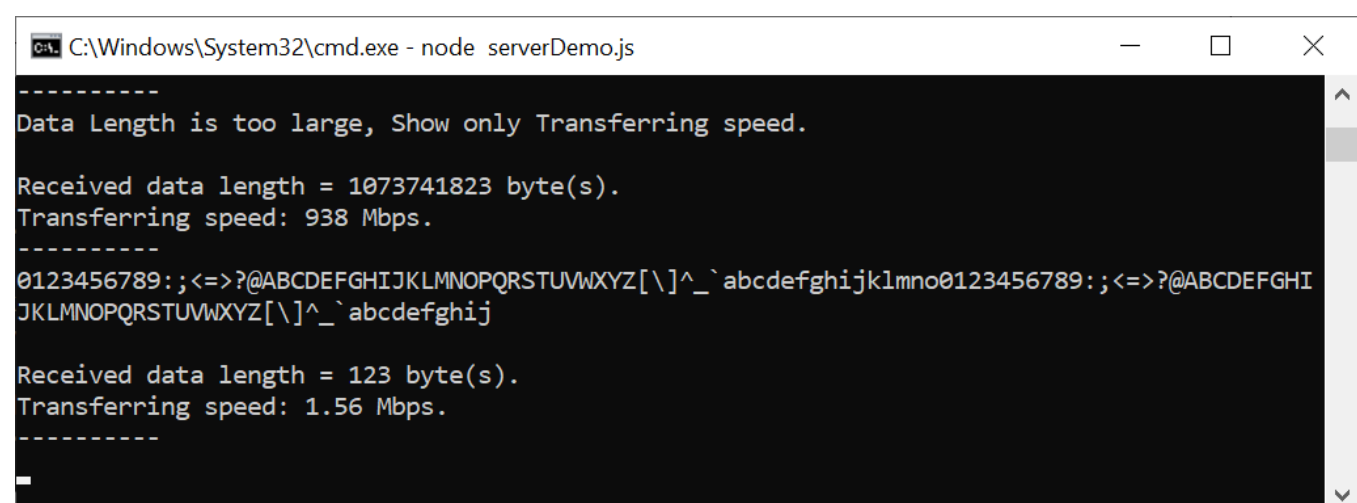
```

C:\Windows\System32\cmd.exe - node serverDemo.js
-----
./log/DG.html is downloaded !
-----

```

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.



```

C:\Windows\System32\cmd.exe - node serverDemo.js
-----
Data Length is too large, Show only Transferring speed.
Received data length = 1073741823 byte(s).
Transferring speed: 938 Mbps.
-----
0123456789;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz0123456789;<=>?@ABCDEFGHI
JKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz
Received data length = 123 byte(s).
Transferring speed: 1.56 Mbps.
-----

```

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.

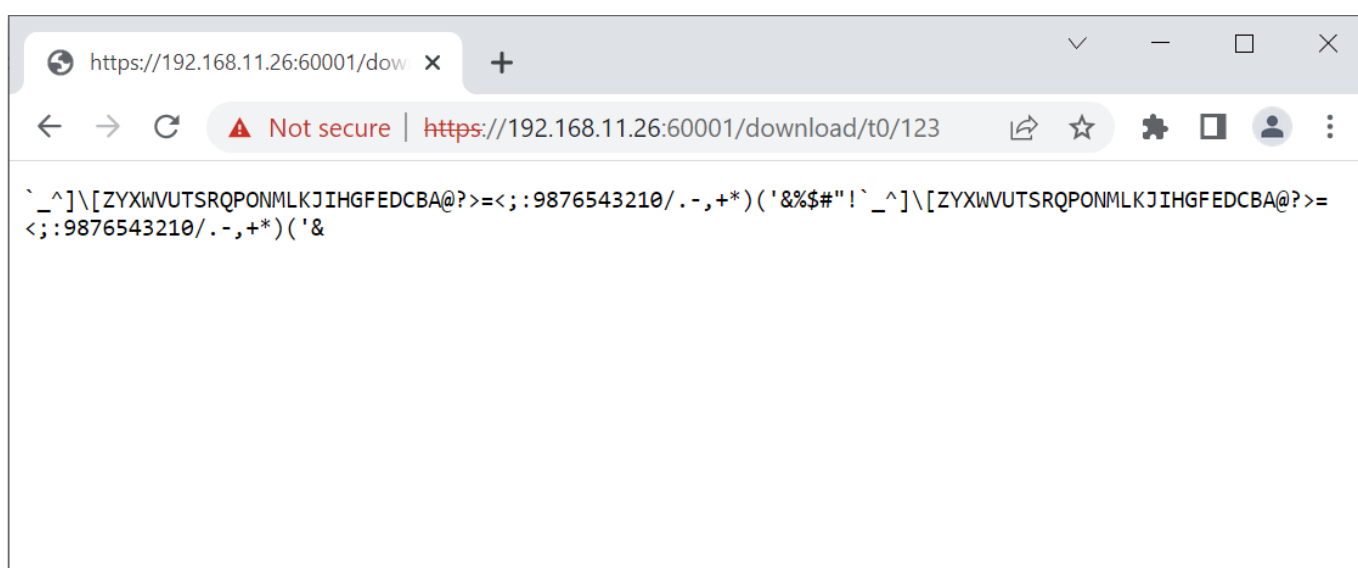


Figure 4-1 Decreasing text pattern shown in web browser

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.

A screenshot of a web browser window. The address bar shows a URL starting with https://192.168.11.26:60001/download/t1/456. A red warning icon and the text "Not secure" are visible next to the URL. The page content displays a series of repeated strings, each consisting of a full ASCII character set followed by a backslash and an underscore, such as "!\"#\$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_!\"#\$%&'()*+,-./0123456789:;<=>?".

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.

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

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.

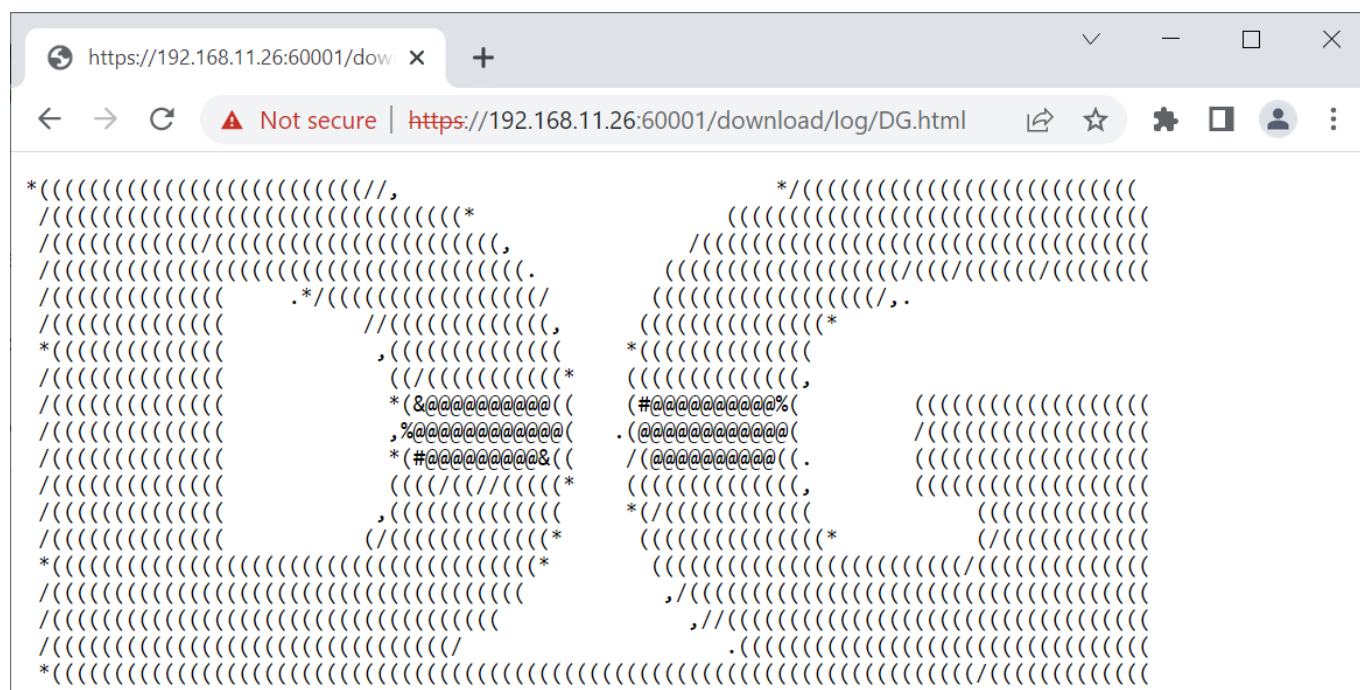


Figure 4-3 DG.html shown in web browser

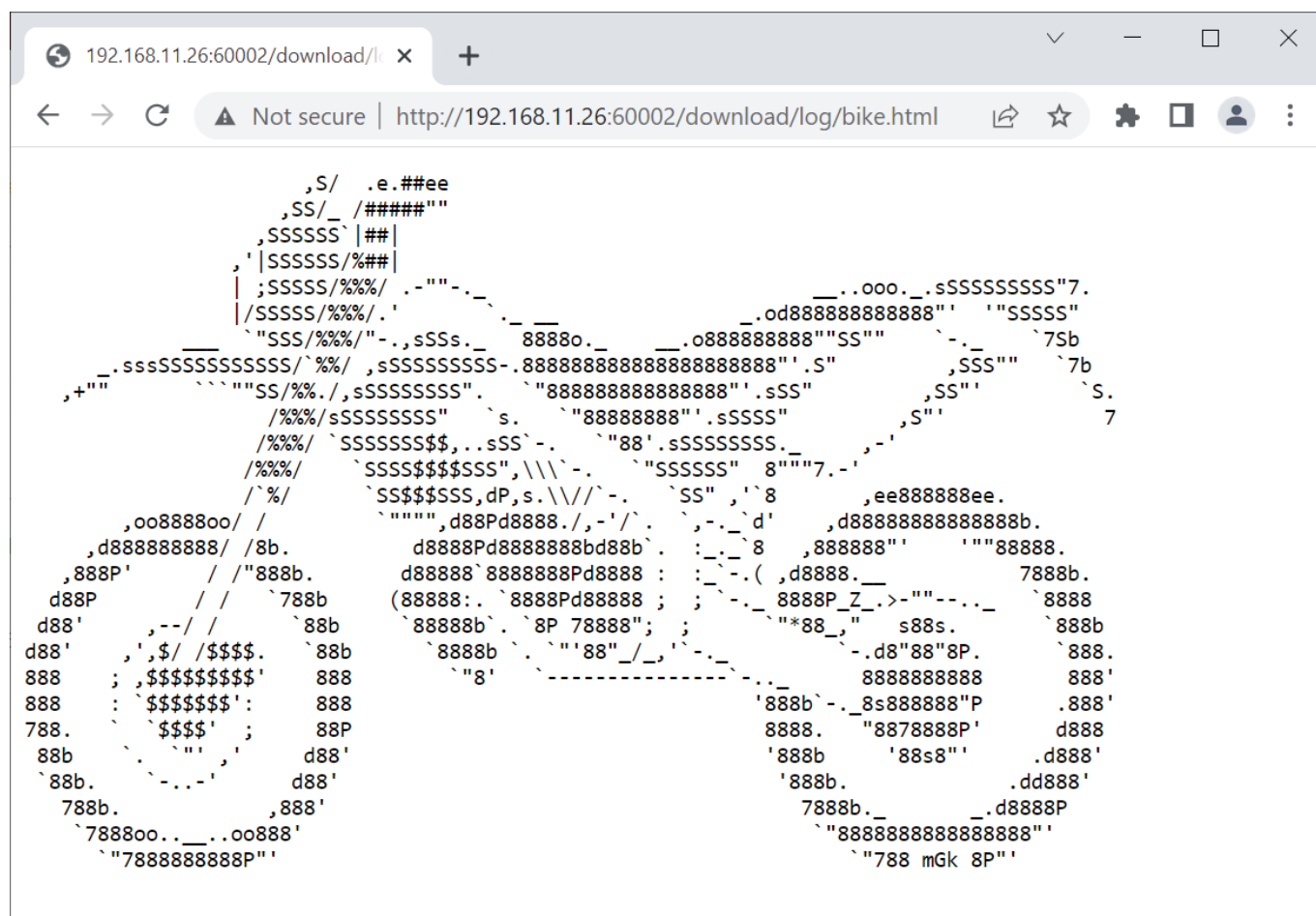


Figure 4-4 bike.html shown in web browser

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.

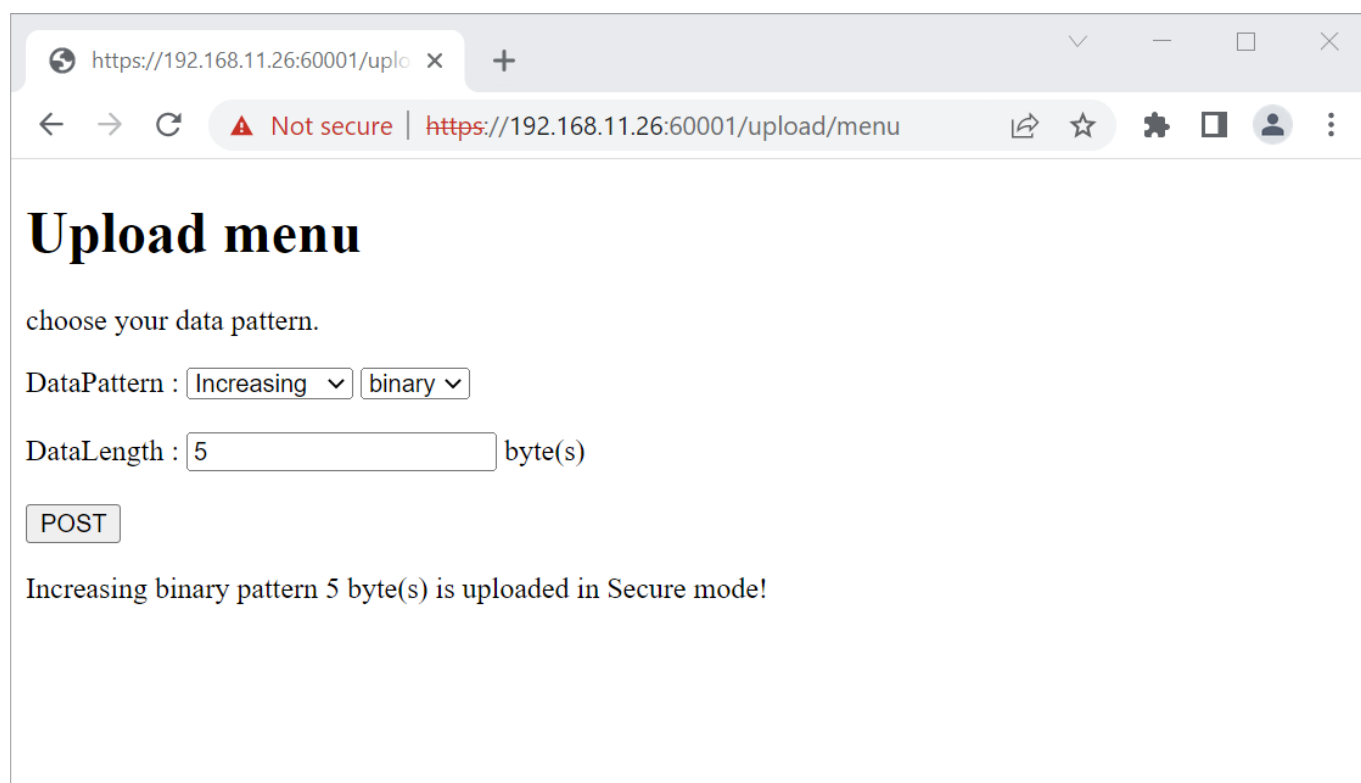


Figure 4-5 Unsecured upload page

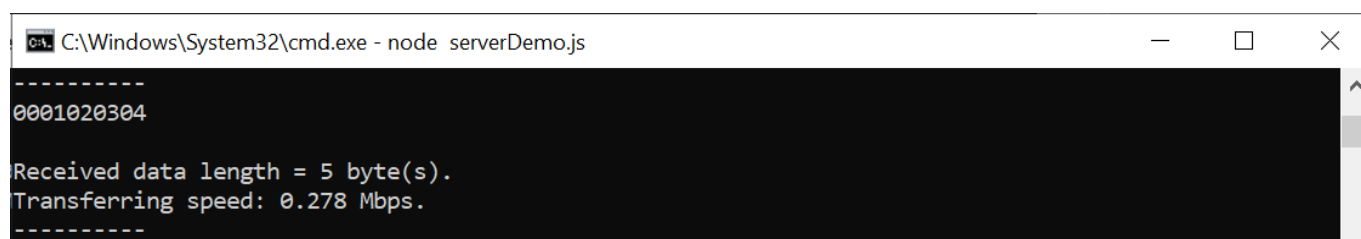


Figure 4-6 Server's console when client upload data that is less than 16kB.

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.

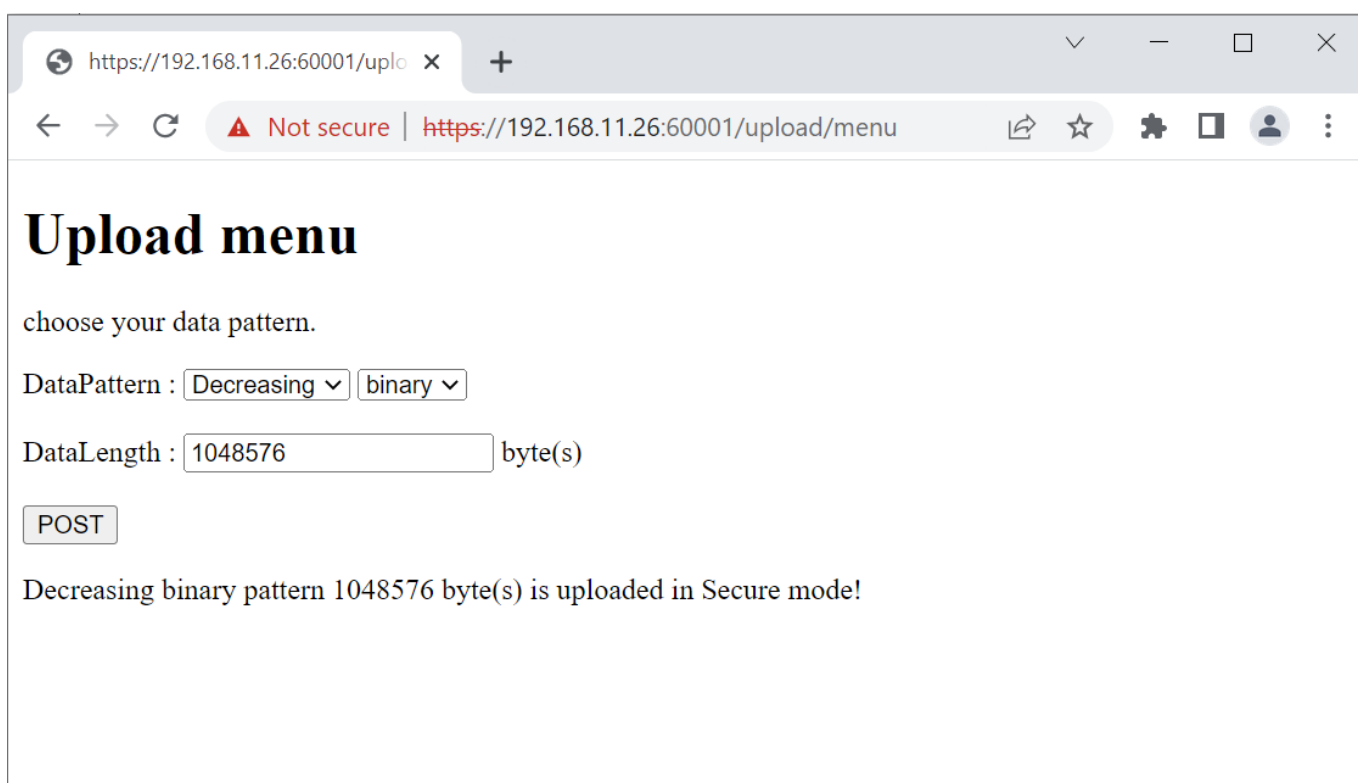


Figure 4-7 Secured upload page

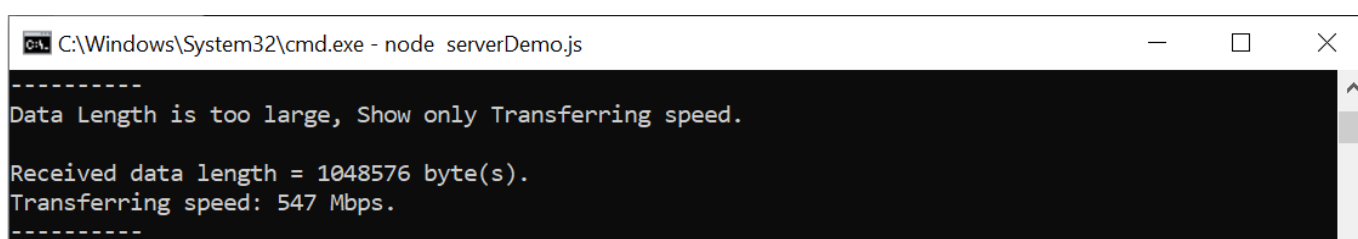


Figure 4-8 Server's console when client upload data that is greater than or equal to 16kB

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.

```
COM3 - Tera Term VT
File Edit Setup Control Window Help

+++ TLS1G Demo +++
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] sethw <1: enable, 0: disable>
    Set operation mode
[5] showkey <1: enable, 0: disable>
    Enable showkey mode for show TLS traffic ticket, session key and iv for encryption/decryption.
[6] myGET protocol://ip:port/download/pattern/length
    Send GET command for downloading pattern data from server.
[7] myPOST protocol://ip:port/upload/pattern/length
    Send POST command for uploading pattern data to server.

>> setip 192.168.11.42
set IP addr to 192.168.11.42

>> setport 60000
set port number to 60000

>> setmac 00-01-02-03-04-05
set MAC addr to 00-01-02-03-04-05

>> sethw 1
Hardware is enabled

>> showkey 1
showkey mode is enabled
```

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.

```
COM3 - Tera Term VT
File Edit Setup Control Window Help

>> myGET https://192.168.11.25:60001/download/t0/1073741824
Open connection
Connecting to 192.168.11.25
=====
Traffic Secret
-----
CLIENT_HANDSHAKE_TRAFFIC_SECRET f0ed3466fab708f6489b1f69ef727ebf5b09dfc4f56a73d36cf0613878af37e0 89A24242CC7D8937B
5696F86CAEC83D1382548E152E9EE5B5974E39EE9B1FBAE13F9AED09FA42BBB1650F6E22F7C448B
SERVER_HANDSHAKE_TRAFFIC_SECRET f0ed3466fab708f6489b1f69ef727ebf5b09dfc4f56a73d36cf0613878af37e0 F06EEC068DED8B2B3
1E1C1F6DA07013514D49409697781FE19F350569BD508F617C40301FAFE6FB6FD729EC427DA1A29
CLIENT_TRAFFIC_SECRET_0 f0ed3466fab708f6489b1f69ef727ebf5b09dfc4f56a73d36cf0613878af37e0 D8B4650CBF6E07F5CCC8E53BA
C476C5FBA1B2415178C6AF9063DDC0FD4C08713A65D3F48DB2852EDE36158DD51473564
SERVER_TRAFFIC_SECRET_0 f0ed3466fab708f6489b1f69ef727ebf5b09dfc4f56a73d36cf0613878af37e0 260D9B2434C6730B02A4F26B1
FBF2D6BA29C0EC914F6402C9E8EE0642B3F0DD4BB1BC449B4258B6D3A776784DA917BF8
-----
tkchs_key : 3DB3A6B2226F9A897D35F8E3B2823FF3A9027B23813AA4E88F54476369B920D0
tkchs_iv : 58E59106B897B75D15F00369
tkshs_key : F0E00DFB89F144017A70616A8A7073057253B08F9F6AAACE5AA71E30EEFE1B9F3
tkshs_iv : E4B8033EFC613A0C17936C9A

tkcapp_key : 587D1AE4FAEF6DDEC77A6464B7C2611F89600C7F5839825E3103BEB71DD5EAA6
tkcapp_iv : 7FE41A8AD85AC5931D241FBB
tksapp_key : 8E4674577F9B7AF436D5C8AA47DB9AF6C5CD6310546E634D0604ED99DA97FED1
tksapp_iv : CB1EFF112816B5D9271736A6
=====
Data Length is too large, Show only transferring speed

Connection closed
=====
Received data length = 1073741824 Byte(s)
Uploading Speed 920 Mbps
```

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

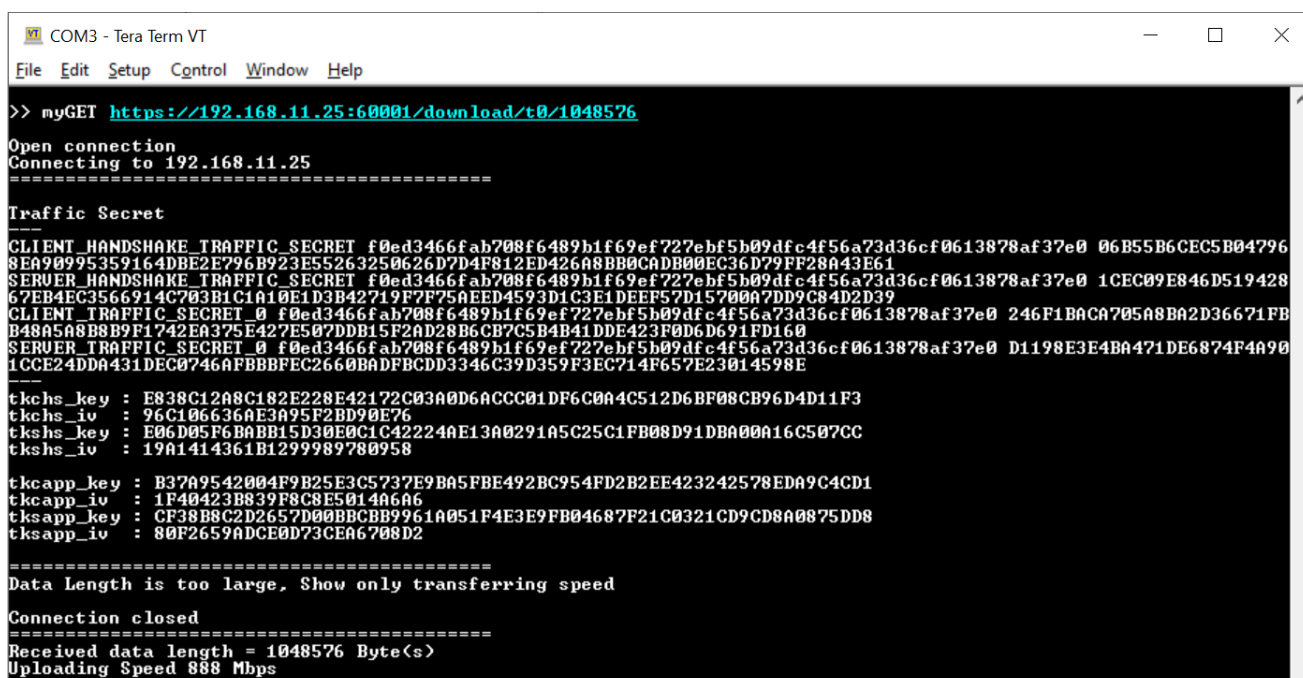
6. myGET protocol://ip:port/download/pattern/length

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.



```
COM3 - Tera Term VT
File Edit Setup Control Window Help

>> myGET https://192.168.11.25:60001/download/t0/1048576

Open connection
Connecting to 192.168.11.25
=====
Traffic Secret
-----
CLIENT_HANDSHAKE_TRAFFIC_SECRET f0ed3466fab708f6489b1f69ef727ebf5b09dfc4f56a73d36cf0613878af37e0 06B55B6CEC5B04796
8EA90995359164DBE2E796B923E55263250626D7D4F812ED426A8BB0CADB00EC36D79FF28A43E61
SERVER_HANDSHAKE_TRAFFIC_SECRET f0ed3466fab708f6489b1f69ef727ebf5b09dfc4f56a73d36cf0613878af37e0 1CEC09E846D519428
67EB4EC3566914C703B1C1A10E1D3B42719F7F75AEE4593D1C3E1DEEF57D15700A7DD9C84D2D39
CLIENT_TRAFFIC_SECRET_0 f0ed3466fab708f6489b1f69ef727ebf5b09dfc4f56a73d36cf0613878af37e0 246F1BACA705A8BA2D36671FB
B48A5A8B8B9F1742EA375E427E507DDB15F2AD28B6CB7C5B4B41DDE423F0D6D691FD160
SERVER_TRAFFIC_SECRET_0 f0ed3466fab708f6489b1f69ef727ebf5b09dfc4f56a73d36cf0613878af37e0 D1198E3E4BA471DE6874F4A90
1CCE24DDA431DEC0746AFBBBFEC2660BADFBCCDD3346C39D359F3EC714F657E23014598E
-----
tkchs_key : E838C12A8C182E228E42172C03A0D6ACCC01DF6C0A4C512D6BF08CB96D4D11F3
tkchs_iv : 96C106636AE3A95F2BD90E76
tkchs_key : E06D05F6BABB15D30E0C1C42224AE13A0291A5C25C1FB08D91DBA00A16C507CC
tkchs_iv : 19A1414361B1299989780958

tkcapp_key : B37A9542004F9B25E3C5737E9BA5FBE492BC954FD2B2EE423242578EDA9C4CD1
tkcapp_iv : 1F40423B839F8C8E5014A6A6
tkcapp_key : CF38B8C2D2657D00BBCBB9961A051F4E3E9FB04687F21C0321CD9CD8A0875DD8
tkcapp_iv : 80F2659ADCE0D73CEA6708D2
-----
Data Length is too large, Show only transferring speed

Connection closed
=====
Received data length = 1048576 Byte(s)
Uploading Speed 888 Mbps
```

Figure 5-3 Serial console when downloading 1MB data

[illegible]

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.

```
COM3 - Tera Term VT
File Edit Setup Control Window Help

>> myPOST https://192.168.11.25:60001/upload/b1/1073741823

Open connection
Connecting to 192.168.11.25
=====
Traffic Secret
-----
CLIENT_HANDSHAKE_TRAFFIC_SECRET f0ed3466fab708f6489b1f69ef727ebf5b09dfc4f56a73d36cf0613878af37e0 3154DD48C412ED75B
878812382D08AA0146C26B8D47557DA90350176E6215D016EB5C109ED38DC0A14F40B1221438F1B
SERUER_HANDSHAKE_TRAFFIC_SECRET f0ed3466fab708f6489b1f69ef727ebf5b09dfc4f56a73d36cf0613878af37e0 F005DA1FDD1DDF4EA
25F25C7C370E77CA7B0524AB6A904DB201637E9128CEFE22C3140E8024F60CD54E1CAC0D466BF3F6
CLIENT_HANDSHAKE_TRAFFIC_SECRET_0 f0ed3466fab708f6489b1f69ef727ebf5b09dfc4f56a73d36cf0613878af37e0 28727E0729490E2FEB07845C5
9B90BF2FEF94EAF703025D55206B6E89D97AB66D615173F683045C9D405BC7E01F8EB1
SERUER_HANDSHAKE_TRAFFIC_SECRET_0 f0ed3466fab708f6489b1f69ef727ebf5b09dfc4f56a73d36cf0613878af37e0 C4F3E175643824975613AC401
E02B6CE8BFDCAE3446F13A4E229F2DC1740D255CDF6C8001E60F9E310AFABF38F65E7C7

tkchs_key : 1A7ECD4364E61212F7C3B05A35ABFF7943947F8A2F983A9DD3796CAD8826B1DC
tkchs_iv : 2BCEA312A4B2EA5F92EE2521
tkshs_key : 6877245FCA8C4150562E811BD0228CB25A03D311A369E2AADE78CC1DB0B2D210
tkshs_iv : FE2ADCCA3B106A6A0BC51622

tkcapp_key : DF9FFFAA6E920D769DF3E8FE063792299CEE9A73BF91A3A3C8CD2D4A078351DC
tkcapp_iv : 0C7390613CA63E703D638801
tkscapp_key : CA00BD92604162176107C626A5F58A3AD10E86B1A11BBD7BE640C0787F7458CD
tkscapp_iv : A1FBC5C4CDAC27D858C4CA8

=====
Uploading...

Close connection
=====
Sending data length = 1073741823 Byte(s)
Uploading Speed 944 Mbps
```

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

```
COM3 - Tera Term VT
File Edit Setup Control Window Help

>> myPOST https://192.168.11.25:60001/upload/b1/123

Open connection
Connecting to 192.168.11.25
=====
Traffic Secret
-----
CLIENT_HANDSHAKE_TRAFFIC_SECRET f0ed3466fab708f6489b1f69ef727ebf5b09dfc4f56a73d36cf0613878af37e0 6FA07EA45F36EDAD0
2697C0265C1151F8D0C94BFA8B69AA208571B09250D37DF5ED107EAC7F1528A6625642A87482A49
SERUER_HANDSHAKE_TRAFFIC_SECRET f0ed3466fab708f6489b1f69ef727ebf5b09dfc4f56a73d36cf0613878af37e0 B8A684590EA0B61A9
08AAEC7B13F4491C2FADC3AA4D1343579AB30292B6B84F265EAA27E8B3A1E4C7DBE96282C7A93C6
CLIENT_HANDSHAKE_TRAFFIC_SECRET_0 f0ed3466fab708f6489b1f69ef727ebf5b09dfc4f56a73d36cf0613878af37e0 187860752075B84429796D199
30C3D6863797354109C476BCCFEC115062EFC88DD4BF5382D43893EA290096E51E41D44
SERUER_HANDSHAKE_TRAFFIC_SECRET_0 f0ed3466fab708f6489b1f69ef727ebf5b09dfc4f56a73d36cf0613878af37e0 7452D74E048FD198D801E8F3C
EFA0C99FC5093182A15B0CC2544B1908F0582B443A0CFA02E0288DF9DF505788E1B72926

tkchs_key : 7AC87B5A8F563154172359684A1DF921FBBE39C2DBA9767C38A905D2009B2F7B
tkchs_iv : 89FF21EBB5A759A902E92B1A
tkshs_key : F41196EC9589D3F1FA6B0272E6B87F1965C99B814C983A4A407213062B6CA94F
tkshs_iv : BB62486DDFD7D70C93A16B10

tkcapp_key : CC0F0AD634A0797535ECFC9D2F206B74CF31A826745BDE9E0E2FF919A7DAE03
tkcapp_iv : 20942D0486CB3153C7AE374B
tkscapp_key : A6177BBEC8B269E1814AB01DB27F09DC6388B3A64C1B03AE272A2614706F8345
tkscapp_iv : D87F0063C8E4ECFF0E75891A

=====
Uploading...

Close connection
=====
Sending data length = 123 Byte(s)
Uploading Speed 4.53 Mbps
```

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

```

C:\Windows\System32\cmd.exe - node serverDemo.js
-----
Data Length is too large, Show only Transferring speed.

Received data length = 1073741823 byte(s).
Transferring speed: 944 Mbps.
-----
000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f303
132333435363738393a3b3c3d3e3f404142434445464748494a4b4c4d4e4f505152535455565758595a5b5c5d5e5f606162
636465666768696a6b6c6d6e6f707172737475767778797a

Received data length = 123 byte(s).
Transferring speed: 3.80 Mbps.
-----

```

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 : ModularMultiplier	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