Indonesian Journal of Electrical Engineering and Computer Science
Vol. 22, No. 1, April 2021, pp. 507~515
ISSN: 2502-4752, DOI:10.11591/ijeecs.v22.i1.pp507-515

Securing sensor data transmission with ethernet elliptic curve cryptography secure socket layer on STM32F103 device

Seniman Seniman[1], Baihaqi Siregar[2], Rani Masyithah Pelle[3], Fahmi Fahmi[4]
[1,2,3]Department of Information Technology, Universitas Sumatera Utara, Indonesia
[4]Department of Electrical Engineering, Universitas Sumatera Utara, Indonesia

Article Info

Article history:

Received Mar 16, 2020
Revised Dec 5, 2020
Accepted Jan 11, 2021

Keywords:

ARM STM32F103
Cryptography
Ethernet communication
Secure socket layer
Sensor

ABSTRACT

Currently there is no method, feature, or ability in securing data transmission in microcontroller systems and applications with client-server scheme communication, while major modern computer systems using secure socket layer (SSL) for establishing secure communication. However, ESP espressif based microcontroller has supported SSL communication to secure data transmission, but only works on the Wi-Fi network. A single-board computer based embedded system has fully supported SSL communication, but it costs a very high price. On the other hand, STM32F103 microcontrollers with a very affordable price even cheaper than the Arduino board has the opportunity to build secure data communication using SSL protocol based on MbedTLS library. In addition to wiznet W5100/W5500 ethernet shield, an STM32F103 SSL client device has been successfully built in this study. The SSL client device supports ECDHE ECDHA AES128 CBC SHA256 SSL cipher suite. The Apache web server must also be configured to support this cipher suite by generating OpenSSL ECC (elliptic curve cryptography) certificate. The system was tested with the LM35 analog temperature sensor, and as a result, the STM32F103 SSL client has successfully secured the data transmission to the Apache SSL web server. The communication time was 3 seconds for the first connection and 42 ms for the next data transmission.

Corresponding Author:

Seniman Seniman
Faculty of Computer Science and Information Technology
University of Sumatera Utara
Jl. Universitas No 9, Fasilkom-TI Kampus USU, Medan, Indonesia
Email: pakniman@usu.ac.id

1. INTRODUCTION

The development of internet of things (IoT) devices has grown widely and rapidly. There are also various communication media used for the IoT system, including wired ethernet [1-3], Wi-Fi [4-8], and cellular communication [9-11]. These communication media are used in the client-server network model. IoT devices usually read sensor data and send it to a remote server. Users can then view the report of IoT devices from desktop, android, or web-based applications [12, 13] or even can make control over the device [14]. It is essential for securing data transmission in the client-server network model since there are many kinds of attacks within the network [15-17]. IoT developers should consider implementing SSL protocol for their system against any possible networking or cyber-attacks. IoT devices are also very vulnerable to be hacked [18, 19]. For IoT devices, there are some existing implementations of SSL protocol nowadays, such as Espressif ESP8266 and ESP32 microcontroller [6-8], but this feature only supports for Wi-Fi network. Single board computer based embedded systems and IoT platforms has a complete support of SSL protocol. This

system was installed with OpenSSL which has mature SSL protocol implementation [20-22]. But, this technology comes at quite a high price, longer start up time and higher power consumption [22].

For the STM32 IoT developer, there is official support of SSL protocol using MbedTLS for an ethernet controller, but this feature only ported to a high-performance device family, such as STM32F407. Even more, TCP/IP stack must also be implemented, and external ethernet PHY (physical layer) must be attached. Actually, there is a more simple solution with extra challenging development by using a cheaper STM32F103 microcontroller device with a wiznet W5100/W5500 ethernet shield. Although with a minimal resource of STM32F103 device, 128 KB flash memory, and 20 KB RAM, but with the help of the W5500 ethernet module, the STM32F103 device does not need to work much hard to handle TCP/IP stack. The W5500 ethernet has been integrated with 8 dedicated sockets with TCP/IP stack. There was also study in our previous research in utilizing this ethernet module [23].

In this research, the MbedTLS SSL protocol library has been studied. And the researchers add support for SSL protocol to STM32F103 and W5500 based MbedTLS library. This results in an STM32F103 SSL client device for wired ethernet communication. Because of the limitation of the STM32F103 microcontroller, not all SSL cipher suite can be integrated into the system. Our research integrated ECDHE ECDHA AES128 CBC SHA256 SSL cipher suite to the system. The working system, STM32F103 SSL client device, was able to secure sensor data transmission to a remote server. This system was expected to be better than AVR or Arduino based IoT devices, which have not any secure communication feature.

2. RESEARCH METHOD

The entire system consists of STM32F103C8T6 microcontroller, Wiznet W5500 ethernet shield, any sensor connected to it, power supply, and web server with SSL enabled. STM32F103 device with very limited resources, 128 KB of flash memory, and 20 KB of RAM. It was configured and programmed to support SSL protocol through W5500 ethernet shield. STM32 device with an ethernet shield act as an SSL client. It sent sensor data with SSL encryption to the webserver, which acts as an SSL server. The overall system architecture is shown in Figure 1.

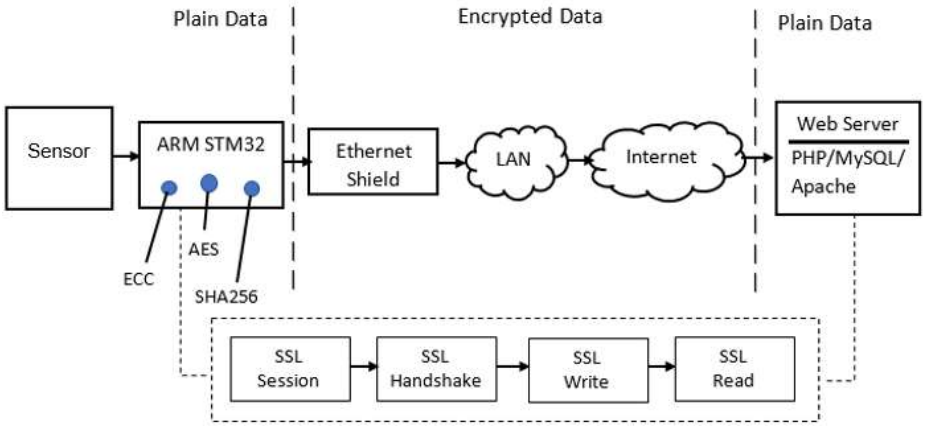

Figure 1. System architecture

2.1. STM32F103 microcontroller configuration

The STM32F103 device was configured to work in its maximum frequency, which is 72 MHz, to achieve the best performance of the entire system. Analog to digital converter (ADC) peripheral needed to be configured for sensor reading and generate a random number. SSL protocol needs a random number to work properly and achieve the best result. STM32F103 device hasn't a random number generator module, but ADC peripheral can be used to produce a random number. Serial peripheral interface (SPI) peripheral was used and configured for communication with the W5500 ethernet shield. The SPI peripheral was configured in the master mode full-duplex. W5500 ethernet shield can be controlled through SPI at up to 80 MHZ maximum speed, but since the STM32F103 device has only 18 MHZ maximum speed, the system used 18 MHZ SPI speed instead. Details of ADC and SPI peripheral configuration is shown in Figure 2.

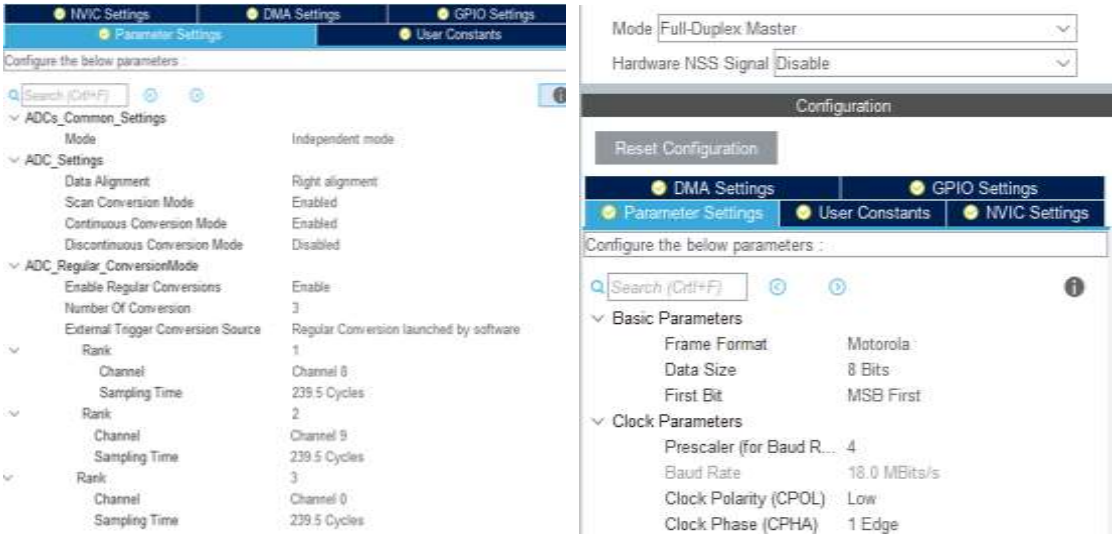

Figure 2. ADC and SPI peripheral configuration

2.2. MbedTLS SSL library and apache web server

In this research, we utilized an open-source MbedTLS SSL library. It is built in C programming language. Actually, the STM32 device family has the official support of MbedTLS. But this feature only works with high-performance device families such as STM32F407. This device has an ethernet controller integrated inside the chip but does not include TCP/IP protocol stack. Unfortunately, there is no available implementation of MbedTLS for the STM32F103 device. There is a various module of MbedTLS to support major SSL cipher suites available nowadays. But, not all MbedTLS modules were implemented in this research because of the limitation of the STM32F103 resources. This research implements ECC SSL in the form of ECDHE ECDSA AES (elliptic curve diffie Hellman ephemeral - elliptic curve diffie-hellman digital signature - advanced encryption standard) based cipher suite.

By default, apache webserver installation has been included with SSL enabled. But, default SSL configuration works only for RSA based cipher suite. The researchers have been generated an ECC certificate using OpenSSL and applied to apache configuration to support the ECDHE ECDSA AES cipher suite. Openssl scripts below were used to generate an ECC certificate for the webserver.

openssl ecparam -name prime256v1 -genkey -param_enc named_curve -out privatec.key
openssl req -new -sha256 -newkey ec:./privatec.key -nodes -keyout server.key -x509 -days 1024 -out server.crt
openssl req -new -key server.key -out server.csr -sha256

2.3. W5100/W5500 ethernet shield

Wiznet W5100/W5500 is an embedded ethernet controller module with a hardwired TCP/IP stack on it. W5100 has 16 KB internal buffer memory for 4 dedicated ethernet sockets, whereas W5500 has bigger buffer memory 32 KB for 8 sockets. The implementation of these two kinds of chips has a similar process and driver and only differ in the amount of socket that they can handle. There was also some researches using this ethernet module [24-27]. Those researches were using AVR Arduino microcontroller or FPGA, and there was neither process nor method for securing data transmission.

3. RESULTS AND ANALYSIS

The researchers have been successfully built a prototype device, as shown in Figure 3. The prototype was used to inspect and test the SSL communication between the STM32F103 SSL client device and the apache SSL web server. For the testing process, the client device connects to the apache webserver directly using straight through unshielded twisted pair (UTP) cable. The client device has been configured with 192.168.137.178 IP address, while the web server using 192.168.137.177 IP address. LM35 analog temperature sensor is used to get real sensor data for then to be sent to the webserver. Wireshark was used for inspecting ethernet traffic between client and server.

Figure 3. STM32F103 SSL client prototype device

The random number which is needed by MbedTLS is generated by shifting, and-ing and or-ing bits of two ADC channels. Based on the inspection of researchers, the best result of a random number of two ADC channels is calculated as the formula below.

$$rng = (adc\_ch1 \text{ AND } 0x0F) \text{ OR } (adc\_ch2 << 0x04) \quad (1)$$

The researchers have been successfully ported MbedTLS to work with STM32F103 and its STM32cubeIDE development environment. This results in some minimal configuration scripts of MbedTLS that must be applied to work properly in the STM32F103 environment. The configuration script was saved in config.h of the MbedTLS library package. Table 1 describes a minimal configuration script that must be defined for ECDHE ECDSA AES based cipher suite.

In order to integrate the W5500 ethernet module and STM32F103 microcontroller to support MbedTLS, there were some configuration scripts that must be added to the original W5500 driver. The summary of important configuration scripts from researchers' implementation is listed below.

a) Assign “mbedtls_net_context server_fd” variable to “sock” value of W5500.
b) Using “mbedtls_entropy_add_source” to configure a random number generator with STM32 ADC peripheral.
c) Assign “mbedtls_net_send()” function of MbedTLS to W5500 “EthernetClient.write()” function.
d) Assign “mbedtls_net_recv()” function of MbedTLS to W5500 “EthernetClient.read()” function.

3.1. Sensor data transmission testing without SSL

As a comparison result of this research, plain HTTP communication between client and server must also be inspected. From wireshark inspection window shown in Figure 4, the client’s request format, and the server’s reply data can be obtained easily. The unfortunate consequences of this communication scheme are anyone without any privilege can push/send data to the server and get server replied information. Any sensitive information can be stolen by the untrusted user.

Table 1. Defined configuration script for MbedTLS

MBEDTLS_CIPHER_MODE_CBC	MBEDTLS_SSL_CLI_C
MBEDTLS_ECP_DP_SECP384R1_ENABLED	MBEDTLS_SSL_TLS_C
MBEDTLS_KEY_EXCHANGE_ECDHE_ECDSA_ENABLED	MBEDTLS_ECDH_C
MBEDTLS_SSL_PROTO_TLS1_2	MBEDTLS_ECDSA_C
MBEDTLS_AES_C	MBEDTLS_ECP_C
MBEDTLS_CIPHER_C	MBEDTLS_X509_CRT_PARSE_C
MBEDTLS_BIGNUM_C	MBEDTLS_X509_USE_C
MBEDTLS_CTR_DRBG_C	MBEDTLS_ASN1_PARSE_C
MBEDTLS_NO_DEFAULT_ENTROPY_SOURCES	MBEDTLS_ASN1_WRITE_C
MBEDTLS_NO_PLATFORM_ENTROPY	MBEDTLS_OID_C
MBEDTLS_ENTROPY_C	MBEDTLS_PK_C
MBEDTLS_MD_C	MBEDTLS_PK_PARSE_C
MBEDTLS_SHA256_C	NO_MBEDTLS_SSL_VERIFY
MBEDTLS_SHA256_C	MBEDTLS_AES_ROM_TABLES
MBEDTLS_SHA256_C	MBEDTLS_ECP_NIST_OPTIM
MBEDTLS_ECP_MAX_BITS 256	MBEDTLS_ECP_WINDOW_SIZE 2
MBEDTLS_MPI_MAX_SIZE 48	MBEDTLS_ECP_FIXED_POINT_OPTIM 0
MBEDTLS_SSL_CIPHERSUITES MBEDTLS_TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	
MBEDTLS_SSL_MAX_CONTENT_LEN 1024	

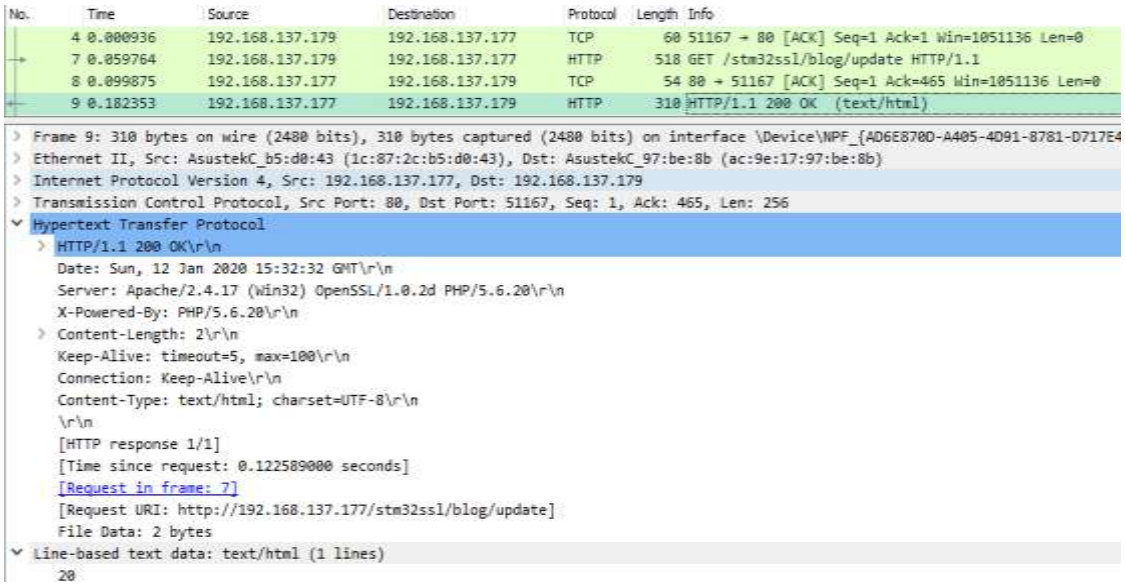

Figure 4. Sensor data transmission without SSL

Based on the figure above, entire HTTP communications which are not encrypted, are easy be revealed and understood. We can gather much information from there which are encapsulated as HTTP header. Those information are date, web server name, destination server path, content type, content length and connection type. And the most important data which is sensor data, it can be viewed directly as plain text number "20". It can be seen by anyone on the network, including threat actors who might be doing the man-in-the-middle (MITM) attack.

3.2. Sensor data transmission testing with SSL

STM32F103 SSL client device has been inspected and tested for data transmission. The initial connection SLL communication, TCP SYNC packet, comes from STM32F103 SSL client device with IP address 192.168.137.178 and port number 1027 trying to connect to the apache SSL web server at 192.168.137.177 IP address and 443 port number as shown in wireshark inspection window in Figure 5 below. The initial connection then successfully connected as indicated by TCP ACK packet.

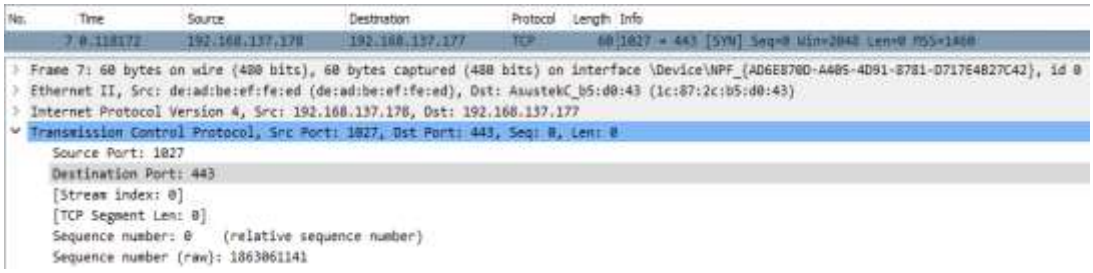

Figure 5. Initial SSL connection

In the next phase of SSL communication, the "*Client Hello*" phase, the SSL client then sends offering of ECDHE ECDHA AES128 CBC SHA256 cipher suite to the SSL server. This communication phase consists of several handshake protocol details. Among of them are SSL TLS version and length, random number which have been generated by STM32 internal ADC peripherals based on (1), session information and cipher suite details. Wireshark inspection window in Figure 6, shows that the STM32F103 SSL client device has been successfully constructed the "*Client Hello*" packet as part of the SSL communication.

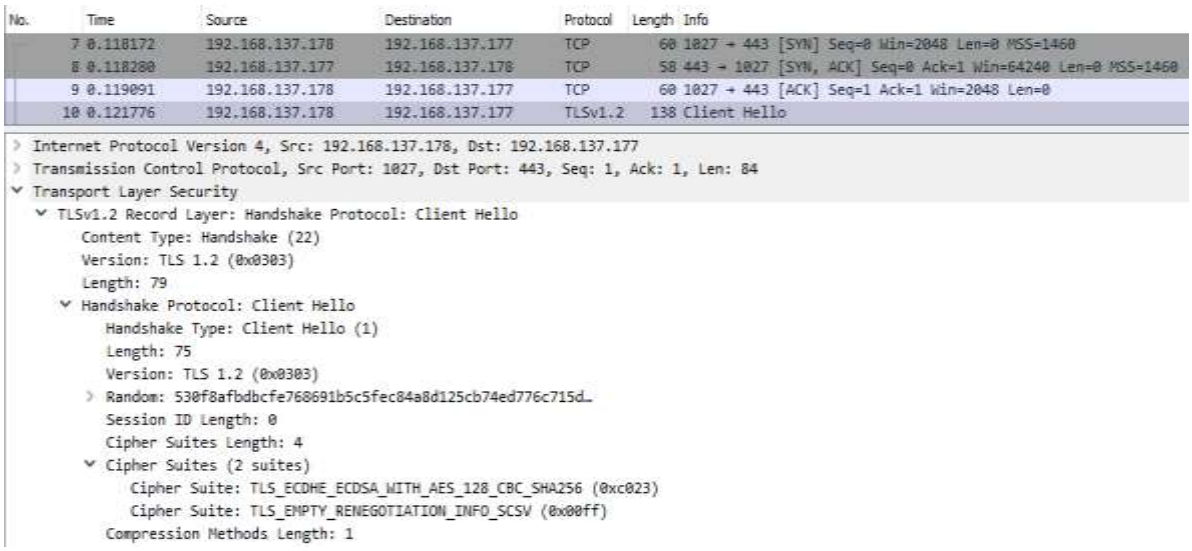

Figure 6. ECDHE ECDHA AES128 CBC SHA256 cipher suite offered by the client

The SSL server then replies with "*Server Hello, Certificate, Server Key Exchange, Server Hello Done*" packet at once as much as 932 bytes of data. This means that the SSL server exactly understands about "*Client Hello*" data packet which has previously constructed and sent by the STM32F103 SSL client device. The overview inspection of this phase is shown in Figures 7(a), (b).

This research only covers self-signed certificate for SSL communication, thus the "*Certificate*" part in the "*Handshake Protocol*" was skipped in SSL communication above. This method significantly reduces space usage of STM32 flash memory and achieve faster SSL communication speed.

The next stage, SSL communication process continues with "*Client Key Exchange*", "*Session Ticket*", "*Change Cipher Spec*", "*Encrypted Handshake*" and the last is the "*Application Data*" phase, which actually contains sensor data embedded to it. As a result of this SSL data communication, anyone can't understand anything inside the encrypted data. The sensor data is successfully secured sent to the server, as shown in Figure 8.

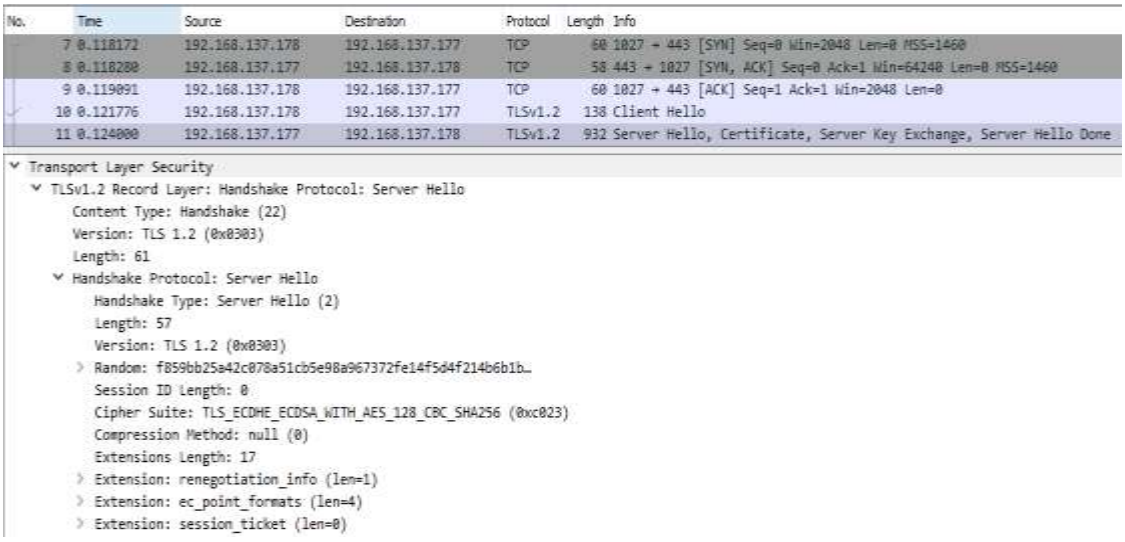

Figure 7(a). "*Server Hello, Certificate, Server Key Exchange, Server Hello Done"* phase

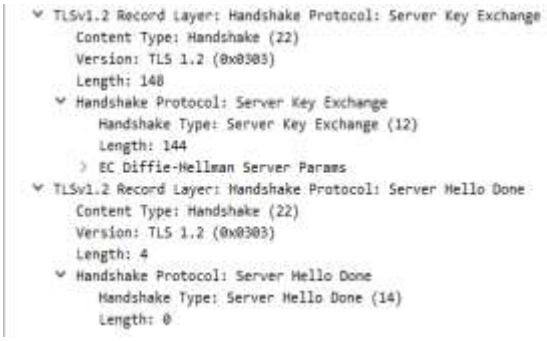

Figure 7(b). Continuation of "*Server Hello, Certificate, Server Key Exchange, Server Hello Done"* phase

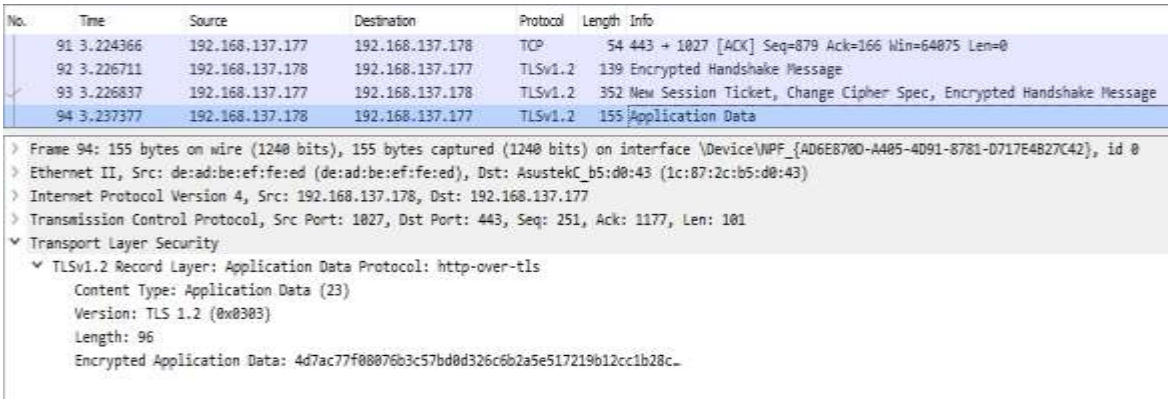

Figure 8. Encrypted application data

3.3. System performance

System performance in this research was measured by observing the execution and communication time of each phase of the SSL communication process in wireshark inspection. For example, a measuring between "*Server Hello, Certificate, Server Key Exchange, Server Hello Done"* phase and "*Client Key Exchange*" phase at row numbered 11 and 89 is shown in Figure 9. They were successfully executed at 0.124000 and 3.215127 wireshark second time.

The most consuming time is the "*Server Hello, Certificate, Server Key Exchange, Server Hello Done*" phase, which takes about three seconds. But, for every next data transmission, which is the "*Application Data*" phase takes only about 42ms. This should be very fast enough for sensor data transmission. Detail of execution time for each SSL communication phase described in Table 2.

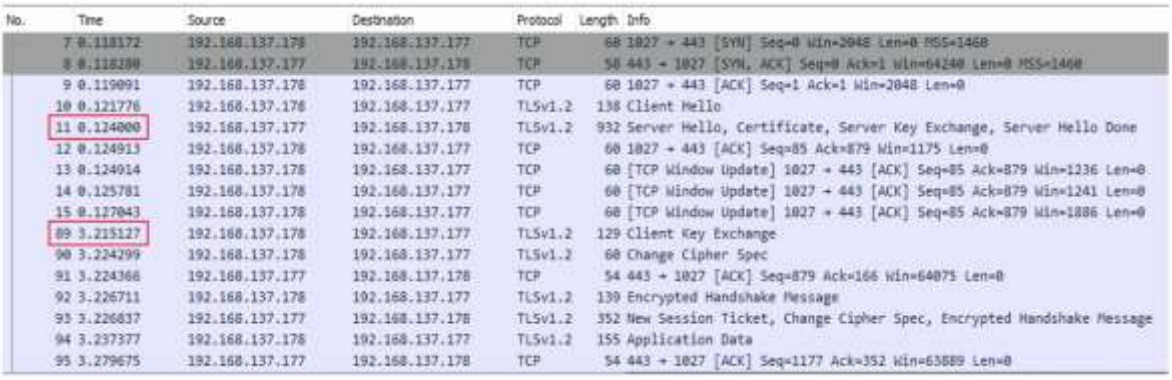

Figure 9. Measuring SSL execution and communication time in wireshark

Table 2. Communication time of each SSL communication phase

SSL Handshake Process	Time (s)
Client Hello	0.002224
Server Hello, Certificate, Server Key Exchange, Server Hello Done	3.091127
Client Key Exchange	0.009172
Change Cipher Spec	0.002412
Encrypted Handshake Message	0.000126
New Session Ticket, Change Cipher Spec, Encrypted Handshake	0.010540
Application Data	0.042298

In addition to the compilation process of STM32F103 microcontroller using system workbench for STM32 version 2.5, the entire system consumes (text section) about 82204 bytes flash memory of 128 Kbytes total available. It means only 62.7% of flash memory is utilized. The rest space of flash memory can be used for other programming purposes, tasks or processes. The use of memory (RAM) for initialized variables (data section) is 220 bytes and uninitialized data (bss section) is 3208 bytes. The compiled result and resource allocation of the entire system shown in Figure 10.

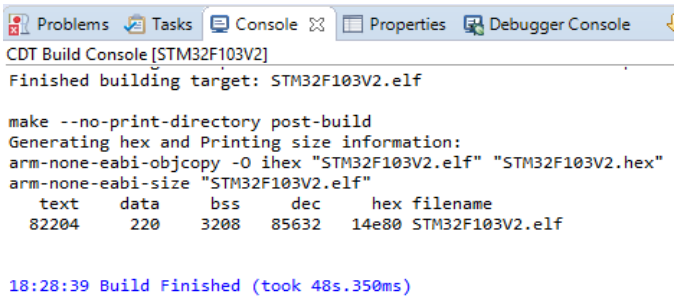

Figure 10. Compile result for entire system

4. CONCLUSION

The functional system of STM32F103 microcontroller and W5500 ethernet shield has been successfully built into an SSL client device. The STM32F103 SSL client device supported ECDHE ECDHA AES128 CBC SHA256 SSL cipher suite. The communication time of the first connection took about 3 seconds, and for the next data transmission, it only took about 42 ms. The main system has successfully secured sensor data transmission to the web server using SSL communication protocol as part of the HTTPS (hypertext transfer protocol secure) protocol. While ESP8266 and ESP32 Wi-Fi-based SSL platforms do not offer any ethernet version of SSL solution, and single-board computer SSL platform comes with a higher cost, longer start up time and higher power consumption. Otherwise, this research results a novelty and contribution of porting complex SSL protocol in ethernet communication media by using a low-cost microcontroller platform. This system should be a promising solution for low-cost IoT platform for better security concern.

REFERENCES

[1] V. Nivedan and R. Kannusamy, "Weather Monitoring System using IoT with Arduino Ethernet Shield," in *International Journal for Research in Applied Science and Engineering Technology*, vol. 7, no. 1, pp. 218-221, 2019, doi:10.22214/ijraset.2019.1038.

[2] L. N. Pintilie, T. Pop, I. C. Gros and A. Mihai Iuoras, "An I2C and Ethernet based open-source solution for home automation in the IoT context," in *2019 54th International Universities Power Engineering Conference (UPEC)*, Bucharest, Romania, 2019, pp. 1-4, doi:10.1109/UPEC.2019.8893583.

[3] B. Siregar, A. Hizriadi, M. Faizal, and Sulindawaty, "Monitoring System Volume of Crude Palm Oil on Vertical Tank Using Ultrasonic Sensor and Solenoid Valve," in *Journal of Physics: Conference Series*, vol. 1255, 2019, p. 012041, doi:10.1088/1742-6596/1255/1/012041.

[4] O. O. Flores-Cortez, R. A. Cortez, and V. I. Rosa, "A Low-cost IoT System for Environmental Pollution Monitoring in Developing Countries," in *2019 MIXDES - 26th International Conference Mixed Design of Integrated Circuits and Systems*, 2019, pp. 386-389, doi:10.23919/MIXDES.2019.8787056.

[5] Andreas, C. R. Aldawira, H. W. Putra, N. Hanafiah, S. Surjarwo, and A. Wibisurya, "Door Security System for Home Monitoring Based on ESP32," in *Procedia Computer Science*, vol. 157, 2019, pp. 673-682, doi:10.1016/j.procs.2019.08.218.

[6] O. Barybin, E. Zaitseva, and V. Brazhnyi, "Testing The Security Esp32 Internet Of Things Devices," in *Cybersecurity: Education, Science, Technique*, vol. 2, no. 6, pp. 71-81, 2019, doi:10.28925/2663-4023.2019.6.7181.
[7] N. Nikolov and O. Nakov, "Research of Secure Communication of Esp32 IoT Embedded System to.NET Core Cloud Structure using MQTTS SSL/TLS," in *2019 IEEE XXVIII International Scientific Conference Electronics (ET)*, 2019, doi:10.1109/ET.2019.8878636.
[8] S. S. Alam, A. J. Islam, M. M. Hasan, M. N. M. Rafid, N. Chakma, and M. N. Imtiaz, "Design and Development of a Low-Cost IoT based Environmental Pollution Monitoring System," in *2018 4th International Conference on Electrical Engineering and Information & Communication Technology (iCEEiCT)*, 2018, doi:10.1109/CEEICT.2018.8628053.
[9] F. Fahmi, F. Nurmayadi, B. Siregar, M. Yazid, and E. Susanto, "Design of Hardware Module for the Vehicle Condition Monitoring System Based on the Internet of Things," in *IOP Conference Series: Materials Science and Engineering*, vol. 648, 2019, p. 012039, doi:10.1088/1757-899X/648/1/012039.
[10] B. Siregar, F. Rachman, S. Efendi, and Sulindawaty, "Monitoring the Value of Water Quality and Condition Parameters Using the Open Sensor Aquarium," in *Journal of Physics: Conference Series*, vol. 1255, p. 012036, 2019, doi:10.1088/1742-6596/1255/1/012036.
[11] Soeharwinto, M. F. Ariska, B. Siregar, U. Andyani and F. Fahmi, "Power meter monitoring for home appliances based on mobile data communication," in *2017 International Conference on Smart Cities, Automation & Intelligent Computing Systems (ICON-SONICS)*, Yogyakarta, 2017, pp. 116-120, doi:10.1109/ICON-SONICS.2017.8267832.
[12] R. F. Rahmat, A. Yusuf and T. Z. Lini, "Real-Time Sensor Application for the Measurement of the Vertical Profiles of Atmosphere," in *2019 3rd International Conference on Electrical, Telecommunication and Computer Engineering (ELTICOM)*, Medan, Indonesia, 2019, pp. 182-188, doi:10.1109/ELTICOM47379.2019.8943853.
[13] P. Sihombing, M. Zarlis and Herriyance, "Automatic Nutrition Detection System (ANDES) for Hydroponic Monitoring by using Micro controller and Smartphone Android," in *2019 Fourth International Conference on Informatics and Computing (ICIC)*, Semarang, Indonesia, 2019, pp. 1-6, doi:10.1109/ICIC47613.2019.8985851.
[14] T. H. Nasution, M. Yasir, Fahmi and S. Soeharwinto, "Designing an IoT system for monitoring and controlling temperature and humidity in mushroom cultivation fields," in *2019 International Conference on Electrical Engineering and Computer Science (ICECOS)*, Batam Island, Indonesia, pp. 326-331, 2019, doi:10.1109/ICECOS47637.2019.8984446.
[15] J. Liu and F. Labeau, "From Wired to Wireless: Challenges of False Data Injection Attacks Against Smart Grid Sensor Networks," in *2018 IEEE Canadian Conference on Electrical & Computer Engineering (CCECE)*, Quebec City, QC, pp. 1-6, 2018, doi:10.1109/CCECE.2018.8447662.
[16] M. Dulik, "Network attack using TCP protocol for performing DoS and DDoS attacks," in *2019 Communication and Information Technologies (KIT)*, Vysoke Tatry, Slovakia, pp. 1-6, 2019, doi:10.23919/KIT.2019.8883481.
[17] P. P. Lokulwar and H. R. Deshmukh, "Threat analysis and attacks modelling in routing towards IoT," in *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, Palladam, pp. 721-726, 2017, doi:10.1109/I-SMAC.2017.8058273.
[18] K. Kaushik and S. Dahiya, "Security and Privacy in IoT based E-Business and Retail," in *2018 International Conference on System Modeling & Advancement in Research Trends (SMART)*, Moradabad, India, 2018, pp. 78-81, doi:10.1109/SYSMART.2018.8746961.
[19] I. K. Poyner and R. S. Sherratt, "Privacy and security of consumer IoT devices for the pervasive monitoring of vulnerable people," in *Living in the Internet of Things: Cybersecurity of the IoT - 2018*, London, pp. 1-5, 2018, doi:10.1049/cp.2018.0043.
[20] S. Iyer, G. V. Bansod, P. N. V and S. Garg, "Implementation and Evaluation of Lightweight Ciphers in MQTT Environment," in *2018 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT)*, 2018, pp. 276-281,doi:10.1109/ICEECCOT43722.2018.9001599.
[21] R. Munoli and S. Dasiga, "Secure Data Transmission for Iot Applications," in *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 5, no. 7, pp. 548-552, 2016.
[22] M. El-Hajj, M. Chamoun, A. Fadlallah and A. Serhrouchni, "Analysis of Cryptographic Algorithms on IoT Hardware platforms," in *2018 2nd Cyber Security in Networking Conference (CSNet)*, Paris, 2018, pp. 1-5, doi:10.1109/CSNET.2018.8602942.
[23] M. A. Muchtar, Seniman, D. Arisandi, and S. Hasanah, "Attendance fingerprint identification system using arduino and single board computer," in *Journal of Physics: Conference Series*, vol. 978, p. 012060, 2018, doi:10.1088/1742-6596/978/1/012060.
[24] A. Myasischev, L. Komarova, R. Gritsky, and K. Kulik, "Web Server On Arduino With Authorization And Graphic Represen-Tation Of Information From Sensors," in *Collection of scientific works of the Military Institute of Kyiv National Taras Shevchenko University*, no. 64, pp. 99-112, 2019, doi:10.17721/2519-481X/2019/64-10
[25] S. S. Dewi, D. Satria, E. Yusibani, and D. Sugiyanto, "Design of Web Based Fire Warning System Using Ethernet Wiznet W5500, " in *Proceedings of MICoMS 2017 Emerald Reach Proceedings Series*, 2018, pp. 437-442.
[26] H. Kurosaki, K.-I. Yasuba, T. Okayasu, and T. Hoshi, "UDP Packet-Processing Capacity on an Arduino Node for Ubiquitous Environment Control Systems, " in *Agricultural Information Research*, vol. 25, no. 1, pp. 19-28, 2016, doi:10.3173/air.25.19.
[27] X. Liu, F. Wang, and B. Wu, "A Remote Monitoring Method of Radio Based on W5500, " in *2015 Fifth International Conference on Instrumentation and Measurement, Computer, Communication and Control (IMCCC)*, 2015, doi:10.1109/IMCCC.2015.81.