11TH INTERNATIONAL SCIENTIFIC CONFERENCE
ON DEFENSIVE TECHNOLOGIES
OTEH 2024
Serbia, Tara, 9-11 October 2024

CASE STUDY: IMPLEMENTATION PERSPECTIVES OF END-TO-END ENCRYPTION IN MILITARY IOT

KRISTINA ŽIVANOVIĆ
Center for Applied Mathematics and Electronics, Belgrade, kristina.zivanovic@vs.rs

DIMITRIJE KOLAŠINAC
Center for Applied Mathematics and Electronics, Belgrade, dimitrije.kolasinac@vs.rs

STEFAN IVANOVIĆ
Center for Applied Mathematics and Electronics, Belgrade, stefan.ivanovic@vs.rs

JOVANA MIHAILOV
Center for Applied Mathematics and Electronics, Belgrade, jovana.mihailov@vs.rs

MARIJA ŠEKLER
Ministry of Defence, Belgrade, marija.sekler@mod.gov.rs

Abstract: *In real-life IoT applications, data protection is a significant challenge, especially in military contexts where data can be hijacked or monitored. This paper focuses on implementing end-to-end security to safeguard data. System includes ArduinoUNO board with a GPS sensor, gathering coordinates of the monitored object, transfering them directly to a client Raspberry PI computer, which is in charge of encrypting the acquired data with a lightweight encryption algorithm, thus preparing it for transport through an unprotected network to the server. Server application receives data from the network, deciphers it and displays it on the map. The main idea is to provide the commander with a complete overview of the area of operation. The server application was developed using Qt Framework, while the client application was developed as a standard C++ console application. Another part of this research was also an Qt compatible C++ implementation of the elliptic curve algorithm suite in a separate library which was used both client and server-side. The system was tested and the coordinates data was successfully displayed on the server side. This system was correctly responding and deciphering data in real time causing negligible delay which does not impact practical usability of the system.*

Keywords: *Elliptic curves, Qt Framework, IoT systems, GIFT-COFB, Raspberry PI.*

1. INTRODUCTION

In 2013. Global Standards Initiative on Internet of Things (IoT-GSI) has defined IoT (Internet Of Things) as global infrastructure of informatic society which provides advances services by physical and virtual cross-linking of things. [1]

Development of network infrastructure and mass incorporation of sensors in many portable devices, cars and vehicles, was the great base for developing this systems. Due to huge number of devices connected in network and large amount of data that are stored in them, there is a big risk of unauthorized access. [2] Security of data in IoT system must fulfill two demands: providing physical protection of devices, and providing integrity, secrecy, authenticity of data itself. [2]

Generally speaking, data protection comprises access control, user authentication, configuration of protective barriers, Intrusion prevention and intrusion detection systems, secure boot etc. [3] All these methods leave risk of compromising end devices. The only way that guarantees data protection at the lowest level is encrypting those at the edge devices.

However, usually those devices have small processing power, limited memory and it is necessary to optimize existing cryptography algorithms which are usually very demanding in case of memory usage, and processing unit occupation. [4]

In this paper, the symmetric encryption algorithm that is used is the GIFT-COFB algorithm, which was one of the top 10 NIST (National Institute of Standards and Technology) finalists for the Lightweight Cryptography Standardization process, demonstrating its efficiency and security for constrained environments such as IoT devices and embedded systems. [4]

For digital signature and key exchange purposes, a dedicated library was developed encompassing all functionalities needed for implementing ECDSA (Elliptic curve Digital Signature Algorithm) and basic operations with elliptic curves. This library supports secure digital

Corresponding author: K.Živanović;
E-mail: kristina.zivanovic@vs.rs
DOI: 10.5937/OTEH24064Z

signing using the ECDSA algorithm and facilitates key exchange operations. It ensures cryptographic operations are efficiently handled, making it suitable for applications requiring robust security measures in handling sensitive information.

2. SYSTEM ARCHITECTURE

The system designed and implemented for this research includes an Arduino UNO microcontroller, which, in combination with a GPS sensor, collects coordinate data of the tracked object. A Raspberry Pi 4 model B V1.2 acts as the gateway device, receiving data from the Arduino UNO microcontroller, encrypting it, and transmitting it over the network to a server application. The server application, hosted on a Linux operating system, decrypts the data and displays the locations on a map.

The infrastructure of system is shown on figure 1.

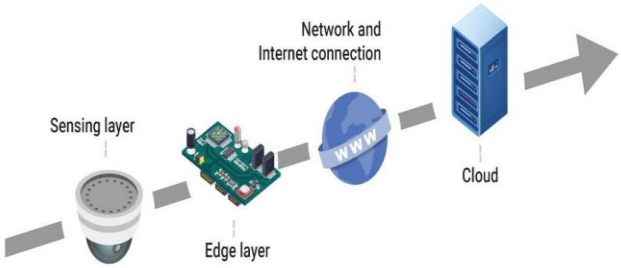

Figure 1. System infrastructure

The solution proposed in this paper precisely aligns with the structure depicted in Figure 1. In the future, the server application could be hosted on a remote server in the cloud. In that case, it would only be necessary to connect all the gateway devices (Raspberry Pi computers in this instance) to the network.

3. IMPLEMENTATION

The system designed and implemented in this work consists of a client and server application. The client application, hosted on a Raspberry Pi computer, is developed in C++. At this level, data from the Arduino microcontroller is collected via the port on the Raspberry Pi, continuously loaded into the program, encrypted, and sent over the TCP protocol to the "remote" server application. The server application is developed in C++ using the Qt framework.

Figure 2. shows the graphical interface of the server application. To access the database containing coordinate data, the QtQuick, QtLocation, and QtPositioning modules were required.

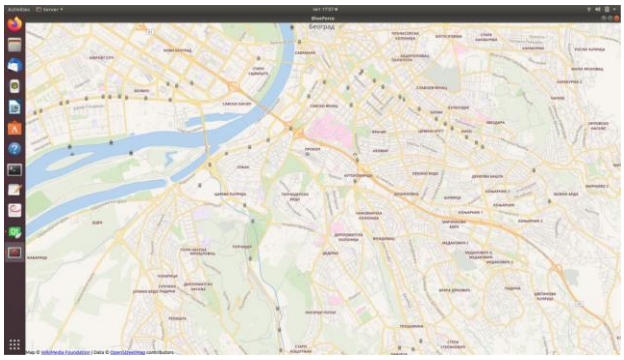

Figure 2. Server side application

QtQuick is a module in Qt that provides the language infrastructure and libraries for developing graphical applications. Instead of using QtWidgets, which is the foundation for creating graphical applications in the Qt environment, QtQuick objects and controls are used. Manipulation of these objects is achieved by writing code in QML (Qt Modeling Language). QML is a declarative language for designing graphical interfaces in Qt Quick applications. Additionally, this application can be extended with standard C++ code, facilitated by the C++ API, which allows calling QML functions from a C++ application and vice versa. QML is a declarative language, meaning it defines the application logic without specifying how it is achieved. QtQuick applications contain a main file, main.cpp (like all C++ applications), which loads the .qml file. [15]

The QtLocation module provides access to geolocation information of objects on the earth. It also enables searching for places and gathering information about them. Displaying the map itself cannot be done using C++ instead, QML must be used.

The system is designed to receive data from the network in a constantly active thread, verify the authenticity of the digital signature using the ECDSA algorithm, and decrypt the data using a symmetric algorithm GIFT-COFB. Verification of the digital signature also utilizes the SHA512 algorithm. Before receiving coordinates, the parties perform key exchange and session key generation. Once decrypted, the coordinates are signaled to the main user interface thread, which parses the received string, extracts the coordinates, and updates the map accordingly. Upon application startup, the map is loaded, an instance of the data-receiving class is created, the thread is started, and data reception begins.

The client application is a console-based program that involves receiving data from a GPS sensor, signing it, encrypting it, and sending it over the network to the server side. This application acts as the client in a TCP communication session. Messages are encrypted and sent in 64-bit sequences to the server.

4. RESULTS

The client application, operating in a console environment, gathers GPS data from a sensor via a USB

port (Arduino board), signs with ECDSA and encrypts it using the GIFT-COFB algorithm, and sends the encrypted messages over a TCP session to the server. On the server side, the application receives these encrypted data packets, decrypts them using a symmetric decryption, verify signature and subsequently processes and displays the extracted coordinates. The coordinates received on the server side, after decryption, are showcased in Figures 3 and 4. These results affirm the system's seamless operation in real-time, encompassing encryption, transmission, decryption upon reception, and the accurate display of sensor holder coordinates. This successful demonstration validates the system's capability to securely handle and present sensor data in a live environment.

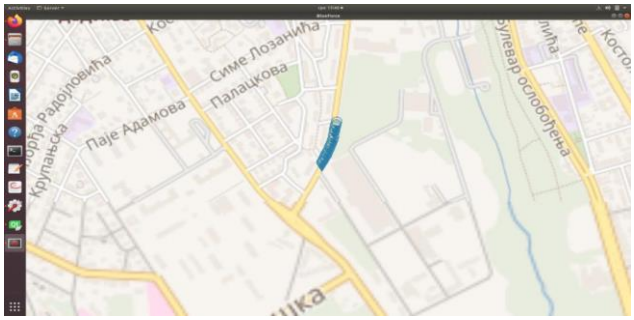

Figure 3. Received coordinates to server side application

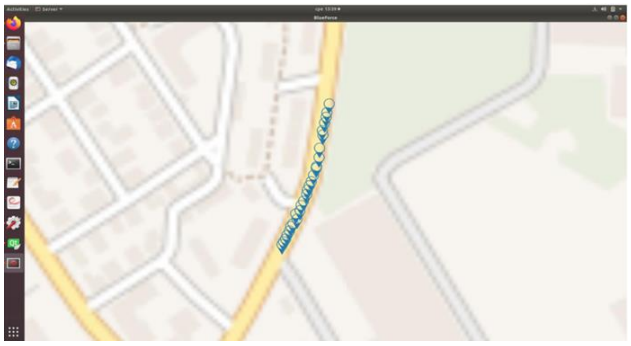

Figure 4. Received coordinates to server side application

Regarding time of execution of the encryption with GIFT-COFB, Table 1. presents measured time on Raspberry Pi 4B. With 4GB of RAM and Quad core Cortex-A72 (ARM v8) 64-bit 1.8GHz processor its been achieved 0,1178 ms per encryption. Encryption algorithm (GIFT-COFB) performed with 10 rounds.

Table 1. Time of execution

Algorithm	Time of execution [ms]
GIFT-COFB	0.1178

5. FUTURE APPLIENCES

This system could be applied effectively in scenarios such as vehicle tracking within a secure environment like a computer network supporting command operations.

To extend the system for tracking multiple objects simultaneously (such as people, vehicles, etc.) and displaying their consolidated positions, certain modifications would be necessary on both the client and server applications. Each client would need to be assigned a unique identifier so that upon reception on the server side, it can accurately associate incoming coordinates with the respective object.

Regarding the network infrastructure, no significant modifications would be required due to the use of the TCP protocol, which inherently supports multiple sessions with the server side. This capability allows for efficient handling of data from multiple clients simultaneously without the need for extensive network changes.

In summary, by implementing unique identifiers for clients and leveraging the existing TCP protocol, the system can effectively scale to track and display the positions of multiple objects in real-time within a secure network environment.

6. CONCLUSION

Thanks to their shorter keys, which require less memory and enable faster arithmetic operations compared to other cryptosystems, elliptic curves are the good choice for implementing public key cryptography in small devices with limited processing power and memory. [12]

Due to the specific requirements of an IoT system and the need for data encryption, the use of lightweight algorithms such as the GIFT COFB algorithm as symmetric algorithm has contributed significantly to the efficiency of the system.

Moving forward, it would be necessary to assess performance and potentially make modifications to meet the required efficiency in specific applications. The selection of algorithms, devices, and application architectures provides a solid foundation for further development in terms of performance optimization.

References

[1] K. Rose, S. Eldridge, L. Chapin, The Internet of Things: An Overview, The internet society (ISOC), 2015, 80.15: 1-53.

[2] M. Nawir, A. Amir, N. Yaakob: Internet of Things (IOT): Taxonomy of security attacks, 3rd International Conference on Electronic Design (ICED), 2016, 11-12.

[3] N. Abosata, S, Al. Rubaye: Internet of thing system of integrity: A comprehensive survey on security attacks countermeasures, Attacks and Countermeasures for Industrial Applications. Sensors 2021, 21, 3654.

[4] M. El-hajj, H. Mousawi, A. Fadlallah: Analysis of Cryptographic Algorithms on IoT Hardware platforms, Future Internet, 2023.

[5] S. Burton, Jr. Kaliski, Elliptic curves and Cryptography: A Pseudorandom Bit Generator and Other Tools, the Department of Electrical Engineering and Computer Science MIT, 1988.

[6] S. Sridhar, S. Smys: Intelligent Security Framework for IoT Devices, Cryptography based End -To- End security Architecture, International Conference on Inventive Systems and Control, 2017.

[7] W. Stallings, Cryptography and network security, Pearson, 2017.

[8] Z. Musulin, Elipticke krivulje i kriptiranje, Prirodno-matematički fakultet, Zagreb, 2016.

[9] B. Schneier: Primenjena kriptografija, Микрокњига, Београд, 2007.

[10] B. S. Verkohovsky, Enhanced Euclid Algorithm for Modular Multiplicative Inverse and Its Application in Cryptographic Protocols, Scientific Research, 2010.

[11] G. C. Kessler: An Overview of cryptography, Academia, 2015.

[12] C. A. Lara-Nino, A. Diaz-Perez and M. Morales-Sandoval, "Elliptic Curve Lightweight Cryptography: A Survey," IEEE Access, vol. 6, pp. 72514-72550, 2018

[13] A. Palve, H. Patel, Towards Securing Real time data in IoMT Environment, International Conference on Communication Systems and Network Technologies, 2018.

[14] NMEA Generator, 03.08.2022.

[15] L. Z. Eng, Hands-On GUI Programming with C++ and Qt5, Packt Publishing, Birmingham, 2018.