# Communication Security on e-Voting using the XTEA Algorithm with ESP32 and MQTT

Gisela Aurora Gitapoetri*, Azril Multazam Pambudi†, Al Khairy Farisy‡
*School of Electrical Engineering and Informatics*
*Institut Teknologi Bandung*
Bandung, Indonesia
{*13221002, †13221004, ‡13221012}@std.stei.itb.ac.id

***Abstract*—E-voting communication security can be encrypted and decrypted properly using MQTT. In its manufacture, the XTEA algorithm is used in the encryption and decryption process. From a voting button it will be encrypted and then sent to another esp32 by decrypting it first. The results will appear on two lights that show the real-time calculation of the voting winners. By using encryption on e-Voting it would give a much safer voting experience.**

***Keywords*—encrypt, decrypt, e-voting, XTEA algorithm**

## I. Introduction

General elections are one of the important aspects of a democratic system. In the current digital era, electronic voting or e-Voting has become an increasingly popular choice. However, with the increasing use of technology in elections, the security of voter data and voter privacy becomes more vulnerable to cyber attacks and data manipulation.

e-Voting is an electronic system that allows voters to vote for candidates in an election using electronic devices such as computers, tablets, or smartphones. This system has been widely used in various countries as an alternative to conventional paper-based voting systems. However, the security of e-Voting systems is a critical issue that needs to be addressed due to vulnerabilities to cyber attacks and vote manipulation.

Therefore, ensuring communication security in e-Voting is crucial. One way to secure communication in e-Voting is by using strong and effective cryptographic algorithms. The XTEA (eXtended Tiny Encryption Algorithm) is one such cryptographic algorithm that can be used to secure communication in e-Voting.

Although there have been some studies on securing e-Voting using the XTEA algorithm, there are still shortcomings in implementation and related research. One way to enhance the security of e-Voting systems is by using a strong encryption algorithm. One encryption algorithm that can be used is XTEA (eXtended Tiny Encryption Algorithm). XTEA is an encryption algorithm designed to encrypt and decrypt data quickly and securely. This algorithm has been widely used in various applications, including computer security systems.

This paper discusses an e-Voting system that uses the XTEA algorithm to enhance system security. Additionally, this paper also discusses how the XTEA algorithm works and how the application of the XTEA algorithm can be implemented in an e-Voting system. By using the XTEA algorithm, it is expected that the e-Voting system will become more secure and reliable for use in general elections. The results of this research are expected to contribute to the development of safer and more trustworthy e-Voting technology.

## II. Related Works

1) E-Voting Concept
e-Voting or Electronic Voting is a general election system that utilizes digital technology to facilitate the voting process. In eVoting, voters use electronic devices such as computers, smartphones, or tablets to cast their desired candidates.
The eVoting system has several advantages compared to traditional manual voting systems. Firstly, e-Voting can enhance efficiency and speed in vote counting and result announcement. Secondly, eVoting can significantly reduce the cost of elections as it does not require a large workforce and printing materials. Thirdly, eVoting can also improve the accuracy of vote counting as the process is automated.
However, e-Voting also comes with security risks that need to be considered. Due to its electronic nature, the e-Voting system is vulnerable to cyberattacks such as hacking and data manipulation. Therefore, ensuring the security of e-Voting is crucial to guarantee the validity and trustworthiness of the election results.

2) XTEA Algorithm
Encryption :

$$v_1 \mathrel{+}= ((v_0 \ll 4) \oplus (v_0 \gg 5)) + v_0 \oplus \text{sum} + k[\text{sum}\&3]$$

$$\text{sum} \mathrel{+}= \text{delta}$$

$$v_0 \mathrel{+}= ((v_1 \ll 4) \oplus (v_1 \gg 5)) + v_1 \oplus \text{sum} + k[\text{sum} \gg 11\&3]$$

Decryption :

$$v_0 \mathrel{-}= ((v_1 \ll 4) \oplus (v_1 \gg 5)) + v_1 \oplus \text{sum} + k[\text{sum} \gg 11\&3]$$

$$\text{sum} -= \text{delta}$$

$$v_1 -= ((v_0 \ll 4) \oplus (v_0 \gg 5)) + v_0 \oplus \text{sum} + k[\text{sum}\&3]$$

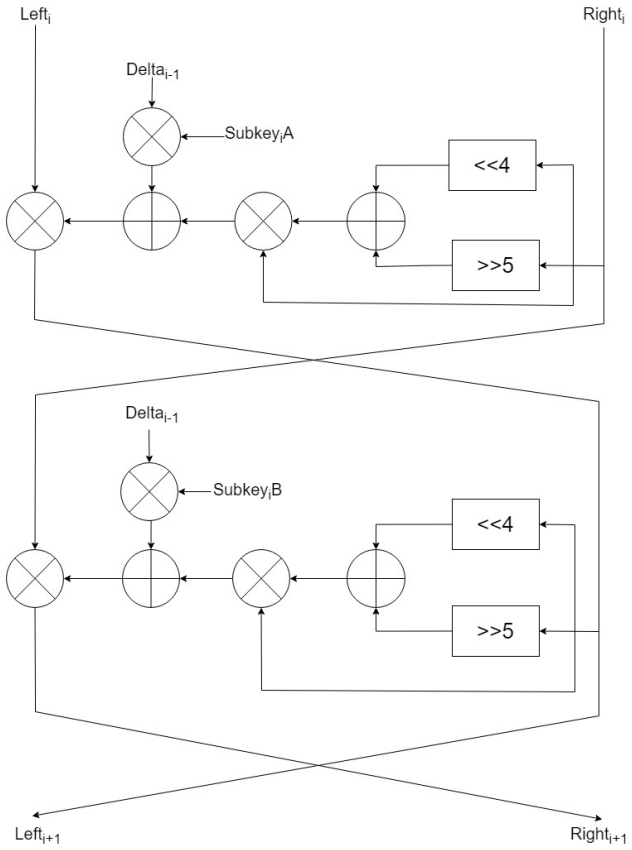


Fig. 1. XTEA Algorithm Diagram

## III. Design and Implementation

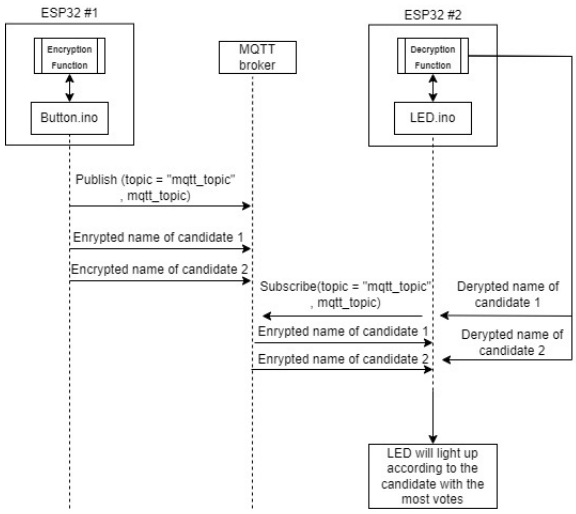


Fig. 2. System Workflow

In this system, two (.ino) files are used to operate two esp32 devices. MQTT Lens broker is used as the message communicator from esp32-1 to esp32-2. The topic used is the same for both subscribe and publish, which is mqtt_topic.

1) button.ino
The button.ino file is used to send a message to an MQTT broker, which will then be transmitted to esp32-2 and encrypted. In this file, variables are declared, including the button pin, str, wifi, and the password used. Then, the mqtt_server is found by opening the command prompt and pinging test.mosquito.org, which will display several numbers that represent the mqtt_server. Within the file, there are several functions: the enchiper function, which serves as the XTEA algorithm; the wifi function, which enables esp32 to connect to Wi-Fi and outputs the Wi-Fi connection status as a return output; the reconnect function, which connects to MQTT and attempts to reconnect to the server if the connection fails until it succeeds; the setup function, which sets up the button to be used. The void loop() function converts str1 and str2 into hexadecimal, detects the pressed button as a HIGH response. If button 1 is pressed, str1 is converted to hexadecimal, encrypted, and published to the mqtt_topic, which is then sent to esp32-2. This process is reversed if button 2 is detected as HIGH.

2) LED.ino
The LED.ino file is used to receive messages from an MQTT broker that are sent from esp32-1 and decrypt the messages. In this file, variables are declared, including wifi, and the same mqtt_server as in the button.ino file is used. The file contains several functions: the hextoString function, which converts a hexadecimal input into a string that is used to assist in deciphering; the decipher function, which decrypts the received message from esp32 into a string to check its voting choice, and then compares the decrypted result with the choices. If choice 1 is selected, it will return 1, but if choice 2 is selected, it will return 2. The wifi function connects esp32 to Wi-Fi and outputs the Wi-Fi connection status as a return output. The reconnect function connects to MQTT and attempts to reconnect to the server if the connection fails until it succeeds. The callback function receives messages from esp32 through MQTT, decrypts the message, and checks its choice to determine which LED should be turned on based on the majority choice. The setup function is used to declare the pins that will be used for the LEDs. The loop function is used to reconnect if there is no connection yet.

*https://bit.ly/githubeVotingXTEA*

## IV. Result and Discussion

From the project we will 3 main parts which is encryption and publising data to MQTT Broker, Subscribe to MQTT Broker and decryption, and Output received.

### A. Encryption and Publising Data to MQTT Broker

When there are button pressed the information from that button, which is the name of the candidate, will be encrypted by esp32. Encryption text itself is the result from the name of the candidate encrypted with XTEA algorithm. The XTEA algorithm itself is available at reference [9]. Variable key used in this program is fixed but we can change it as we like to make encryption text more random. In this project, encryption worked perfectly with each button already assigned to a specific candidate, On serial monitor we could see the encryption text as well as execution time anad resource usage. After ESP32 encrypt the candidate's name, it will then send the text to MQTT Broker. Make sure MQTT broker is already subscribed with the assigned topic. In this project we use mqtt_topic for our topic. The result will be just like the fogure below

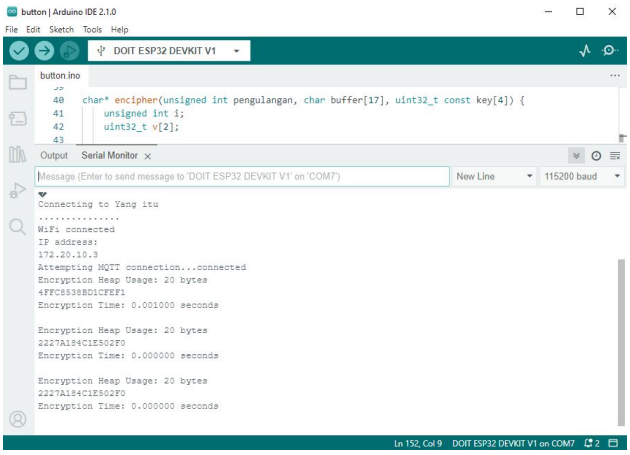


Fig. 3. Enrypted Message on Arduino IDE's serial monitor

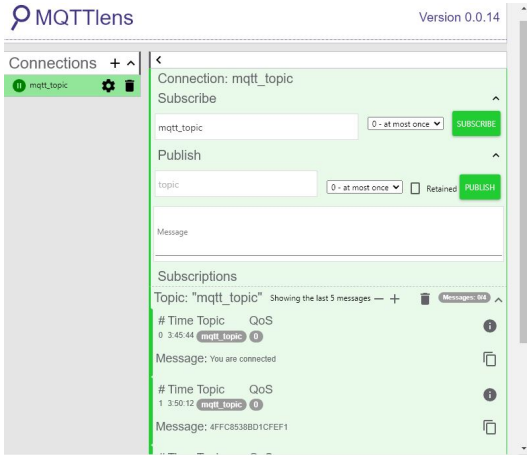


Fig. 4. Encrypted Message on MQTT Broker

### B. Subscribe to MQTT Broker and Decryption

After subscribing to mqtt_topic on MQTT Broker the other esp32 will receive data which is the encrypted text. Then the encryption text will be decrypted to result the name of the candidate. The XTEA algorithm for decryption text could also be seen from reference [2]. At first, our team have a bit of a struggle because the encrypted text didn't get to the second ESP32. To solve this problem, we try to fix our Arduino-IDE code by making sure each code use the same name of mqtt topic and making sure decryption function could function perfectly. After fixing those parts we could receive data, decrypt, also receive the name of the candidate. On the serial monitor we could see on real-time how many votes each candidate have as well as the execution time and resource usage. The result will be just like the fogure below

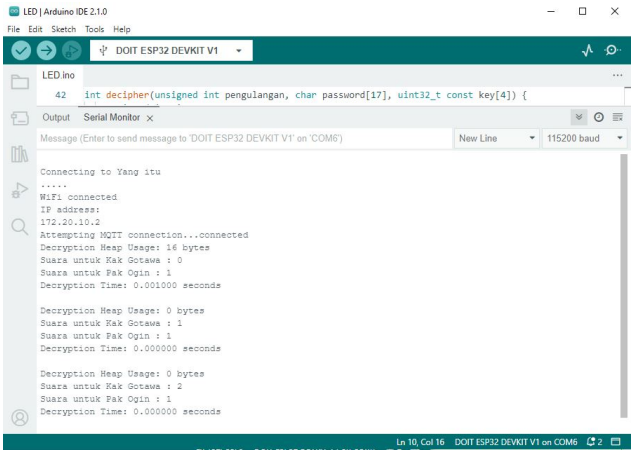


Fig. 5. Decrypted Message on Arduino IDE's serial monitor

### C. Output received

With data successfully decrypted, program could count number of votes for each candidate. Each candidate has already been assigned with their own LED lights. LED will light up according to the candidate with the most votes.

TABLE I
AVERAGE OF EXECUTION TIME AND ESP32 MEMORY USAGE

| No | Program | Execution time | Memory usage |
|---|---|---|---|
| 1 | Publish With Encryption | 0.001 (ms) | 20 bytes |
| 2 | Encryption | 0.00 | 20 bytes |
| 3 | Subscribe With Decryption | 0.001(ms) | 0 |
| 4 | Decription | 0.00 | 0 |

From table above we could grasp the execution time and resource usage. This data explains the process of encryption and decryption, both using ESP32 and not, is extreamly fast with only just 0.001 ms. Encryption code uses malloc thus it will need 20 bytes. Unlike decryption code have 0 bytes meaning it has a constant need of memory usage.

A more clear result on how this projects work could be seen from the video on this link *https://bit.ly/EvotingwithXTEAusingMQTTandESP32*

## V. COUCLUSION

A modernized voting style could be a way to have a much efficient and safe voting experience. By using encryption, people couldn't tamper the choices made by voters as it has keys for it to be decrypted. With LED, it enables people to know the real-time result. Also, by using ESP32 and MQTT broker it gives an efective and secure way on communicating information. Though it is only a small scale project but is applicable to be used on a bigger scale ensure a much more safe voting experience.

## REFERENCES


[1] T. Ahmad, J. Hu and S. Han, "An Efficient Mobile Voting System Security Scheme Based on Elliptic Curve Cryptography," 2009 Third International Conference on Network and System Security, Gold Coast, QLD, Australia, 2009, pp. 474-479.

[2] L. P. K., M. N. K. Reddy and L. M. Manohar Reddy, "An Integrated and Robust Evoting Application Using Private Blockchain," 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184), Tirunelveli, India, 2020, pp. 842-846.

[3] A. K. Goel, A. Rai, A. Narain, A. Richard and K. Kumar, "Trusted Vote: Reorienting eVoting using Blockchain," 2022 Sixth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), Dharan, Nepal, 2022, pp. 129-138.

[4] A. L. Abba, M. Awad, Z. Al-Qudah and A. H. Jallad, "Security analysis of current voting systems," 2017 International Conference on Electrical and Computing Technologies and Applications (ICECTA), Ras Al Khaimah, United Arab Emirates, 2017, pp. 1-6.

[5] S. R. Misal, S. R. Prajwal, H. M. Niveditha, H. M. Vinayaka and S. Veena, "Indoor Positioning System (IPS) Using ESP32, MQTT and Bluetooth," 2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 2020, pp. 79-82.

[6] L. Hardjaloka and V. M. Simarmata, "E-Voting: Need vs. Readiness (Welcome) E-Democracy," Faculty of Law University of Indonesia, Depok, Indonesia, 2011

[7] Zare. Atefeh and Iqbal. M. Tariq, "Low-Cost ESP32, Raspberry Pi, Node-Red, and MQTT Protocol Based SCADA System," 2020 IEEE International IOT. Electronics and Mechatronics Conference (IEMTRONICS). Canada. 2020

[8] Foltýnek. Petr, Marek Babiuch, and Pavel Šuránek. "Measurement and data processing from Internet of Things modules by dual-core application using ESP32 board", Institute of Measurement and Control. London. 2019

[9] San. Ismail and Nuray At. "Lightweight Hardware Architecture for XTEA Cryptographic Algorithm". Conference: The 2012 International Conference on Embedded Systems and Intelligent Technology (ICESIT 2012). Eskisehir Technical University. Turkey. 2012