Encryption Method for Systems with Limited Computing Resources

Roman Chernenko[1], Andriy Anosov[1], Roman Kyrychok[1], Zoreslava Brzhevska[1], and Svitlana Spasiteleva[1]

[1] *Borys Grinchenko Kyiv University, 18/2 Bulvarno-Kudriavska str., Kyiv, 04053, Ukraine*

Abstract

Due to the active development of the Internet of Things (IoT) technology, more and more systems of interconnected devices and sensors are appearing that collect various data and transmit them through gateways to remote servers. It goes without saying that this data must be protected at all stages. This is especially important for data on the functioning of potentially dangerous objects and devices. Because of the features of devices with limited computing resources, it is impossible to use standard methods of information protection in the gateway-built-in sensor link. The article considers the algorithm of the Internet of Things system using limited devices, which consists of a gateway for receiving data from sensors and transmitting them to servers and limited devices used for data collection and encryption. The proposed algorithm describes the process of data packet generation, key generation, encryption, transmission, and decryption of data received from sensors. The reliability of data encryption transmitted in the gateway-built-in sensor link is ensured by the generation of a truly random sequence - the encryption key, based on the initial measured value on the unconnected and ungrounded analog input of the microcontroller, and a series of arithmetic operations.

Keywords

Internet of Things, IoT, network security, encryption, Vernam cipher, random number generation.

1. Introduction

The rapid development of the Internet of Things has led to the creation of a large number of heterogeneous systems of interconnected computing devices, built-in sensors that collect and measure environmental parameters and transmit them through IoT gateways to a remote server in the cloud [1]. It is clear that all data transmission links of such a system must be reliably protected. This is especially important for systems that collect data on the operation of potentially dangerous objects and devices [2]. Therefore, security is crucial for IoT protocols. Computer systems on restricted devices operate on the basis of standard or proprietary protocols, in which data must be protected from interception, modification and substitution. In the gateway-remote server link, the required level of protection can be provided based on standard protocols [3]. In the gateway-built-in sensor (limited device) link, there is an objective need to use algorithms that employ a minimum of computing resources to ensure the required level of information protection.

2. Formulation of the Problem

The application of encryption methods in computer systems on limited devices creates a limitation in the existing computing resources, which makes it necessary to work out such a method that will employ a minimum of such resources.

After analyzing the algorithms, namely the required number of calculations and device memory for organizing these calculations, it was investigated that for the operation of the RSA

CPITS-2022: Cybersecurity Providing in Information and Telecommunication Systems, October 13, 2022, Kyiv, Ukraine
EMAIL: r.chernenko.asp@kubg.edu.ua (R. Chernenko); a.anosov@kubg.edu.ua (A. Anosov); r.kyrychok@kubg.edu.ua (R. Kyrychok); z.brzhevska@kubg.edu.ua (Z. Brzhevska); s.spasitielieva@kubg.edu.ua (S. Spasiteleva)
ORCID: 0000-0002-1439-961X (R. Chernenko); 0000-0002-2973-6033 (A. Anosov); 0000-0002-9919-9691 (R. Kyrychok); 0000-0002-7029-9525 (Z. Brzhevska); 0000-0003-4993-6355 (S. Spasiteleva)

algorithms and the El-Gamal scheme it is necessary to use an amount of memory that exceeds the amount available in class 0 and 1 devices [4]. Accordingly, they cannot be implemented on limited devices [5].

Thus, a problematic issue arises regarding the development and use of the encryption method in modern objects representing computer systems on limited devices.

The purpose of the article is to increase the level of security of systems in the Internet of Things network by developing a method of data encryption on devices with limited computing resources.

3. Analysis of Recent Research and Publications

In [5], a prototype of the IoT system was developed using limited devices, which provides absolute cryptographic stability due to the use of the Vernam cipher with disposable notebooks [6]. During the development of the prototype, the following security vulnerabilities were eliminated, such as transmission of unencrypted data over an unsecured channel.

During the study of encryption methods, the main operations that are used were highlighted: addition, shuffling, bit shift and binary XOR operation. Considering the concept of using the XOR operation in existing encryption methods, it can be noted that such a task remains a priority [7].

In [8], a study and comparative analysis of the bandwidth of low-power wireless IoT devices in the role of wireless switches is presented. Such switches can be used as gateways when implementing a prototype of an IoT system using limited devices.

The complexity of the topological structures of wireless sensor networks due to their variability [9] determines the need to create secure data transmission channels and parameters in all links of the functioning of computer systems with limited computing resources.

4. Research Results

The general model of the IoT system using limited devices (Fig. 1) consists of:

- A gateway for receiving data from sensors and transferring them to servers.
- Limited devices used to collect and encrypt data for secure transmission over unsecured channels to the gateway.

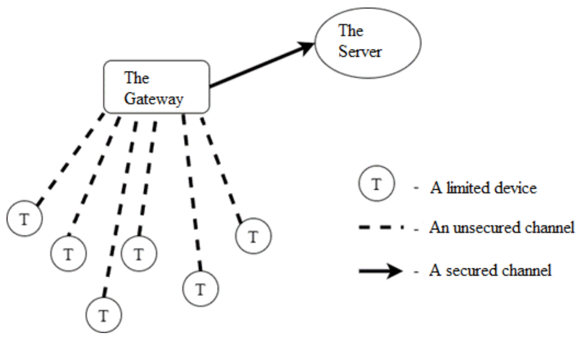


Figure 1: General model of an IoT system using limited devices

The general algorithm of the system functions as follows:

1. Reading data from sensors with a limited device.
2. Generation of a random key, the length of which is equal to the length of the message.
3. Encrypting the message with the Vernam cipher, using the bitwise “exclusive OR” operator.
4. Random selection of one of the predefined keys to encrypt the key itself.
5. Encryption of the key.
6. Sending the message and key to the gateway.
7. Receiving the message by the gateway and sending the encrypted message via secure channels to the company’s servers.
8. Reception of the message by the server and selection of the necessary key to decrypt the key with which important data is encrypted.
9. Data decoding and adding them to special structures for data storage and processing. So, the software implementation of the system consists of three parts, the program running on the limited device is responsible for generating data and encrypting packets for sending. According to the block scheme (Fig. 2), the first step is the generation of the message “M” for further encryption. The prototype uses a temperature and humidity sensor to generate useful values.

After the message generation is completed, the random key sequence generation function is called for data encryption. In general, the function should generate a truly random sequence of characters equal to the length of the message in order to ensure absolute cryptoresistance [10].

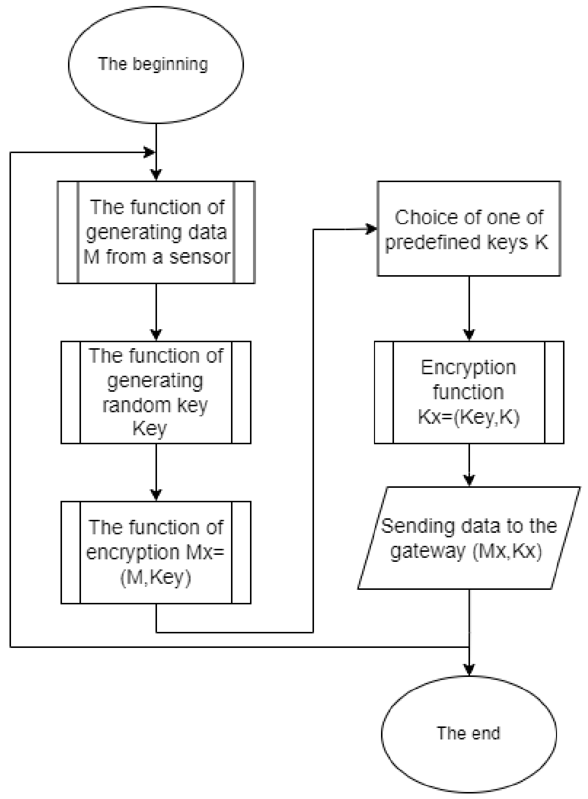

The beginning
The function of generating data M from a sensor
The function of generating random key Key
The function of encryption Mx=(M,Key)
Choice of one of predefined keys K
Encryption function Kx=(Key,K)
Sending data to the gateway (Mx,Kx)
The end

Figure 2: Block diagram of the general algorithm of a limited device operation

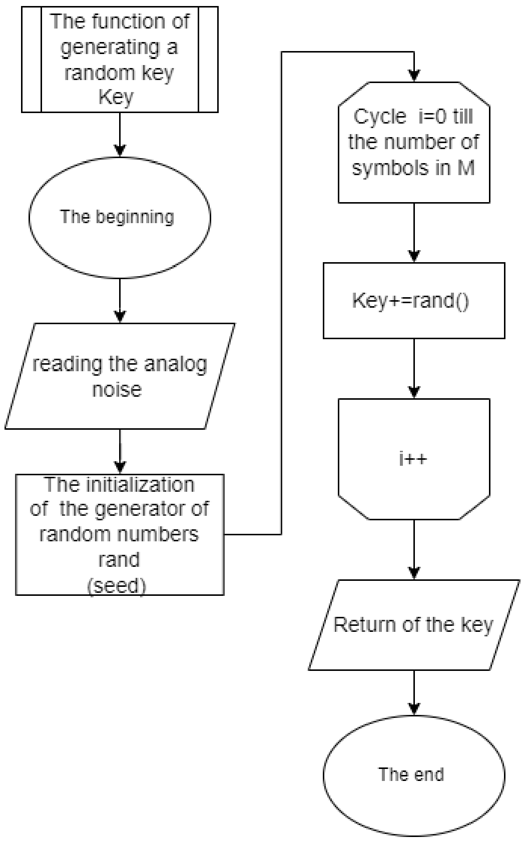

The function of generating a random key Key
The beginning
reading the analog noise
The initialization of the generator of random numbers rand (seed)
Cycle i=0 till the number of symbols in M
Key+=rand()
i++
Return of the key
The end

Figure 3: Block diagram for generating a random key sequence

Any random number generation function performs mathematical operations with some initial value, therefore, to obtain a truly random sequence, the initial value must be random [11]. It was decided to initialize the value measured at the unconnected and ungrounded analog input of the microcontroller, in other words the noise caused by the stray current, and to perform some arithmetic between this value and the sensor readings to make the value even more random. It is worth noting that the use of various “noises” of the environment is a widespread method of forming truly random sequences [12]. So, the random sequence generation function (Fig. 3) performs the following actions:

- Accepts the value of the length of the data to be sent.
- Initializes the initial value for the generation of key symbols.
- In the loop, a key symbol is randomly generated for each symbol of the message.
- The function returns a generated sequence of characters, the length of which is equal to the length of the message.

After running these functions, the program has two variables, the first one stores the value of the message, the second, the value of the key to encrypt this message.

The message is as follows:

```
14.00,23.00,14.00,23.00,
14.00,23.00,14.00,23.00,
14.00,23.00,14.00,23.00,
14.00,23.00,14.00,23.00,
14.00,23.00,14.00,23.00.
```

The generated key looks like this:

```
%ʕ9ʕgʕʕcV(1p'p9ʕCʕʕʕlÛʕgtvgî'
(ʕʕʕnʕ;P=緋=ʕʕʕBʕrʕWʕʕ%ʕNʕʕzʕʕW;
ʕtʕʕ3ʕʕs&Lfʕʕ|ʕCKIW*/
ʕMʕ!Z1ʕ%ʕlʕʕʕ◌҂ʕʕʕʕʕʕUs8ʕʕʕ
```

After generating the message and the key, the encryption function is performed. For encryption, a Vernam cipher is used, which uses a bitwise “exclusive OR” operation to create an encrypted message (Fig. 4). For each symbol of the message in bitwise form, an XOR operation is applied with the corresponding key symbol in bitwise form, for example:

$$\begin{array}{r} \oplus\ \underline{0\,0\,1\,1\,0\,0\,0\,1\ =\ 1} \\ \underline{0\,0\,1\,0\,0\,1\,0\,1 = \ \%} \\ 0\,0\,0\,1\,0\,1\,0\,0 = DC4 \end{array}$$

According to the example, after the bitwise operator XOR was applied to the message character "1," the character DC4 (Device Control 4) was obtained with the corresponding key character "%" at the output. This operation takes place in a loop, for each pair of key and message values. After the end of the loop, the function returns an encrypted message in the form of a text variable, which is ready for transmission to the gateway.

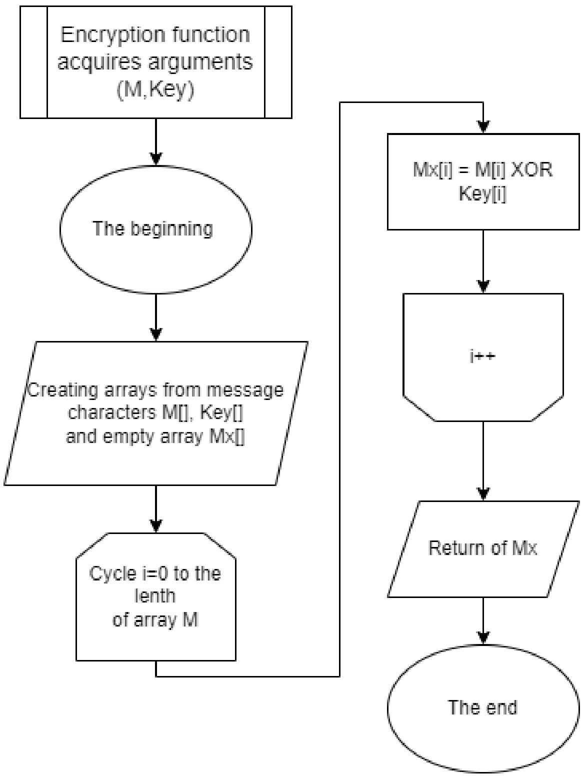

Encryption function acquires arguments (M,Key)
The beginning
Creating arrays from message characters M[], Key[] and empty array Mx[]
Cycle i=0 to the lenth of array M
Mx[i] = M[i] XOR Key[i]
i++
Return of Mx
The end

Figure 4: Block diagram of the operation of the encryption function

Upon completion of the encryption function, one of the predefined keys is selected to encrypt the key with which the message was ciphered. The key is chosen randomly. Since two predefined keys were used in the system prototype, the selection algorithm works as follows. The current value is read on the unconnected analog input:

- With the received value, the operation of the remainder from division by 2 is performed.
- If a 1 is received, then the first key is used, otherwise the second key is used.

Keys for encryption of randomly generated sequences have a fixed length, which is equal to the length of the message and therefore to the length of the key that was generated randomly. Predefined keys must be loaded during flashing of the limited device. The number of such keys may be different depending on the memory of the limited device or the possibility of using additional energy-independent memory in which the keys will be stored. Each limited device must have its own unique keys, so that if one device is compromised, the security of the entire system will not be compromised.

The corresponding keys will be stored on the enterprise server, which will receive and decrypt the received data from the gateway. After choosing a predefined key, the encryption function is called again, but only to encrypt a randomly generated key. At the output, two text variables are obtained, which are the encrypted message and the encrypted key for decrypting the message. After that, these variables can be transferred through any unsecured data transmission channel. As a prototype, transmission through the UART interface is used. In a real system, any standard protocols for Internet of Things networks can be used: ZigBee, Thread, Z-Wave, MQTT, LwM2M [13, 14]. After sending the data, the next data packet is formed. The function of encryption and random sequence generation works very quickly even on limited devices, because it has a linear algorithmic complexity of the algorithm $O(n)$. The data packet sent to the gateway has the following form as in Fig. 5.

The gateway, in turn, can work according to two scenarios depending on the needs of the system. In the first option, the gateway acts as a simple intermediary between the server and the limited device, that is, it uses standard communication protocols to transmit encrypted data to the company's servers without changing packets.

```
ʕpʕʕ□HuʕʕrʕʕʕʕʕʕʕʕY[ʕʕʕʕʕyoʕ□v   □
ʕ□pʕmʕ□□ʕt~ʕ|ʕS@ʕʕr□ʕʕʕHPmP ʕʕʕʕʕ□□ʕʕʕʕD□D}bʕ˜ bʕʕʕʕyʕWʕʕWʕ
ʕʕ□□ʕhʕnx□□`ʕyʕʕB ʕʕ
ʕ□ʕʕ□ ʕʕʕʕʕuʕʕʕʕʕʕ%3ʕʕʕ□□ʕy#□f1ʕJ,ʕ2ʕGRʕ□□ʕ□ʕ$□□ʕd]ʕʕʕ□4|&□ʕʕʕʕʕʕʕʕtpʕʕʕʕ5t□k$□ʕ□ʕʕʕʕ
ʕ"ʕʕ□ʕ□ʕʕ~{ʕtʕ.q]□□ʕ□ʕʕ>□ʕʕ
```

Figure 5: A data packet that is sent to the gateway

In the second option, if there is a need to perform calculations with the received data and adjust the operation of the system, the gateway itself decrypts the data and saves them in a format convenient for calculations. It is possible to allow a mixed version of work, in which part of the data will be decrypted at the gateway, and part will be sent to the server without changes. In this case, it is necessary to use different predefined encryption keys to encrypt the randomly generated keys for the server and for the gateway, so that in the event of a breach of the gateway, the data to be transmitted to the server remains protected.

In any case, the software implementation of decryption and data storage in a convenient format will have approximately the same form (Fig. 6). The algorithm will perform the following steps:

- Receiving an encrypted message over an unsecured channel.
- Receiving an encrypted key to decrypt a message over an unsecured channel.
- Selection of one of the predefined keys for decryption.
- Decryption of the key.
- Decoding the message.
- Data storage in a convenient form for calculations option. The CSV format files are used as a prototype to create a data frame from the received data [15].

The algorithm begins its work by receiving data, to which two symbols have been added due to the peculiarity of sending through the UART interface. The data is sent to the server and stored in the form of two arrays of the byte type (Fig. 7).

After receiving the data, the function of selecting a predefined key (Fig. 8) is called to decrypt the key with which the message was encrypted. The function takes three arguments—the first character of the received encrypted message, the first character of the encrypted key and a list of predefined keys.

According to the block diagram, the algorithm iterates all the keys from the array of predefined keys one by one. Two text variables are created to store the first decrypted character. First, the first character of the key is decrypted, but the key was generated on the limited device randomly. Accordingly, it is not possible to verify the validity of the first character of the key, so with the received key character, it is necessary to perform an XOR operation on the first character of the received message.

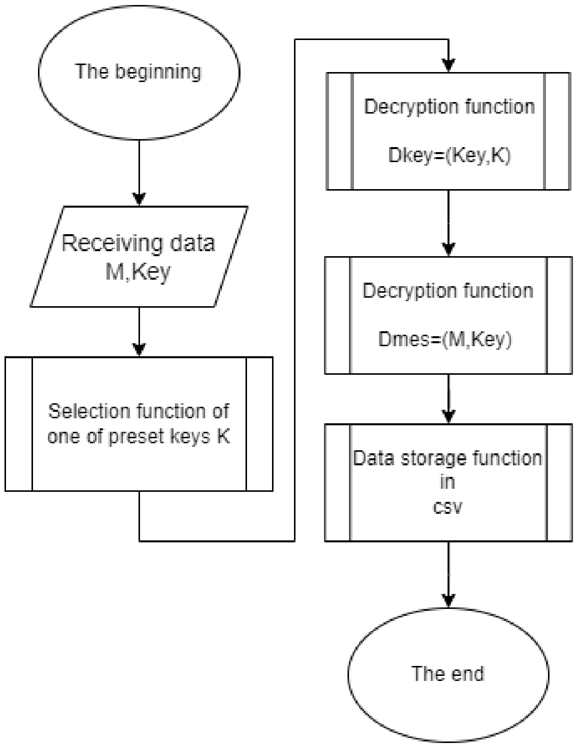
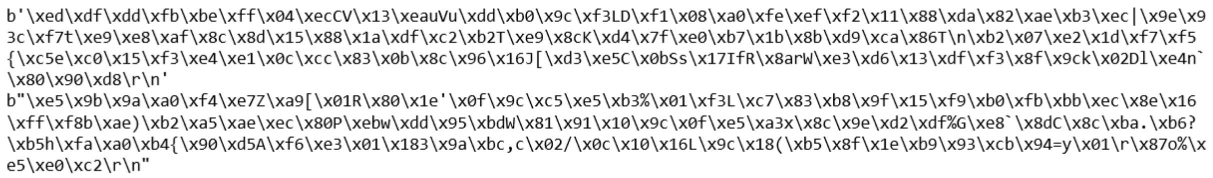

The beginning
Receiving data M,Key
Selection function of one of preset keys K
Decryption function Dkey=(Key,K)
Decryption function Dmes=(M,Key)
Data storage function in csv
The end

Figure 6: The general algorithm of decryption and data storage

Since the data is transmitted from the transmitters, then we can expect a certain symbol of the message, which will already be some kind of information. But if the key was chosen incorrectly, then there will be no useful information in the message. So, you can check the first character of the message: if after decryption it represents the expected data, then the function returns the index of the current operation, accordingly, this is the index of the key in the array that needs to decrypt the randomly generated key. After returning the index, the function completes its work so as not to perform unnecessary operations.



```
b'\xed\xdf\xdd\xfb\xbe\xff\x04\xecCV\x13\xeauVu\xdd\xb0\x9c\xf3LD\xf1\x08\xa0\xfe\xef\xf2\x11\x88\xda\x82\xae\xb3\xec|\x9e\x9
3c\xf7t\xe9\xe8\xaf\x8c\x8d\x15\x88\x1a\xdf\xc2\xb2T\xe9\x8cK\xd4\x7f\xe0\xb7\x1b\x8b\xd9\xca\x86T\n\xb2\x07\xe2\x1d\xf7\xf5
{\xc5e\xc0\x15\xf3\xe4\xe1\x0c\xcc\x83\x0b\x8c\x96\x16J[\xd3\xe5C\x0bSs\x17IfR\x8arW\xe3\xd6\x13\xdf\xf3\x8f\x9ck\x02Dl\xe4n`
\x80\x90\xd8\r\n'
b"\xe5\x9b\x9a\xa0\xf4\xe7Z\xa9[\x01R\x80\x1e'\x0f\x9c\xc5\xe5\xb3%\x01\xf3L\xc7\x83\xb8\x9f\x15\xf9\xb0\xfb\xbb\xec\x8e\x16
\xff\xf8b\xae)\xb2\xa5\xae\xec\x80P\xebw\xdd\x95\xbdW\x81\x91\x10\x9c\x0f\xe5\xa3x\x8c\x9e\xd2\xdf%G\xe8`\x8dC\x8c\xba.\xb6?
\xb5h\xfa\xa0\xb4{\x90\xd5A\xf6\xe3\x01\x183\x9a\xbc,c\x02/\x0c\x10\x16L\x9c\x18(\xb5\x8f\x1e\xb9\x93\xcb\x94=y\x01\r\x87o%\x
e5\xe0\xc2\r\n"
```

Figure 7: Data that is received by the server

If, in the case of packet exchange, the value could not be decrypted, then the function returns a value that is not included in the array index range, and further, the message will not be decrypted, since the key was not matched, and therefore the message did not come from the expected limited device.

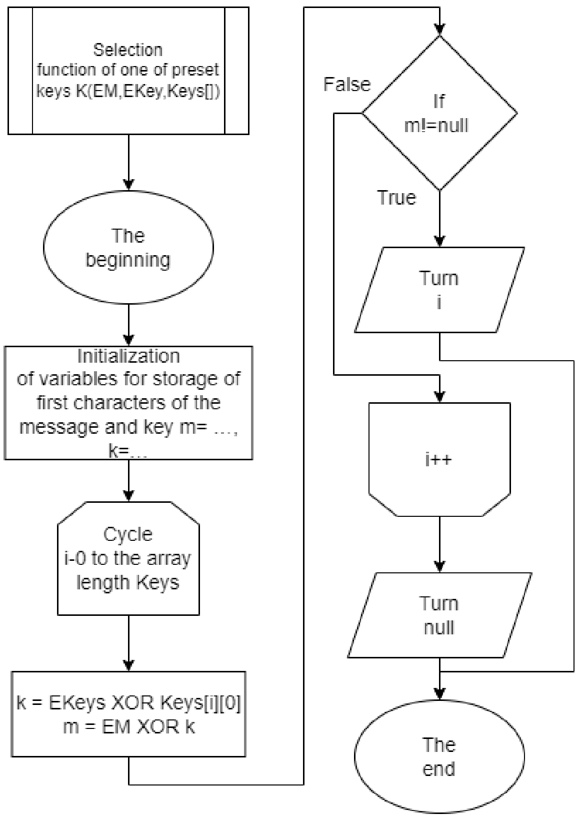

Selection function of one of preset keys K(EM,EKey,Keys[])
The beginning
Initialization of variables for storage of first characters of the message and key m= ..., k=...
Cycle i-0 to the array length Keys
k = EKeys XOR Keys[i][0]
m = EM XOR k
If m!=null
False
True
Turn i
i++
Turn null
The end

Figure 8: Block diagram of the predefined key selection function

After choosing a predefined key, the function for decryption is called (Fig. 9), which accepts two arguments, the text to be decrypted and the key for decrypting the text. The received encrypted key as text and one key from the array of predefined keys whose index was found in the previous step are passed as arguments to the function.

The first step initializes the variable in which the decrypted text will be stored. In this case of the call, the decrypted randomly generated key will be stored in the variable to decrypt the message. Next, in a loop that works for each element in the array of text bytes, except for the last two characters that do not carry information and are the end characters of the string added when sending, a bitwise exclusive OR operation is applied to the corresponding element of the selected key.

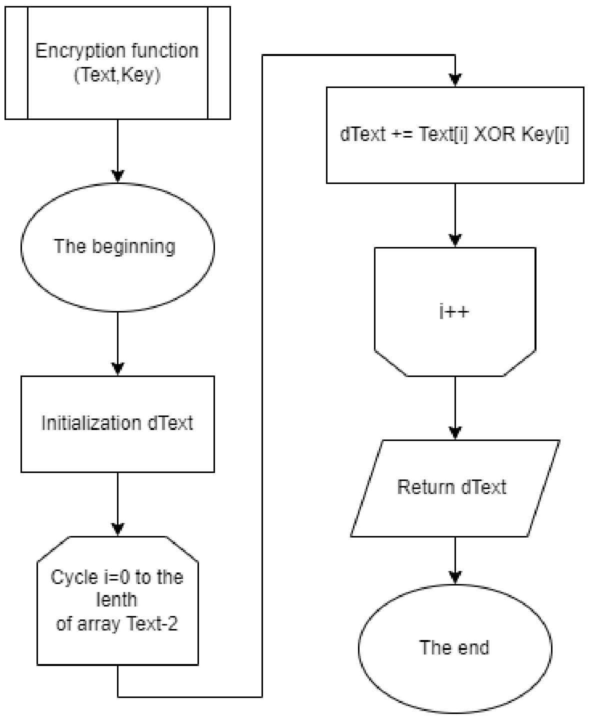

Encryption function (Text,Key)
The beginning
Initialization dText
Cycle i=0 to the lenth of array Text-2
dText += Text[i] XOR Key[i]
i++
Return dText
The end

Figure 9: Block diagram of the decryption function

After the loop is finished, the function returns the value of the decrypted text, which in this case is the key for the next call to the decryption function. After finding the required key and decrypting the encrypted key, the decryption function is called again, but now as arguments, the encrypted message and the key that was decrypted in the previous step are passed. At the output, the function returns a decrypted message that looks like this:

```
91.00,24.00,91.00,24.00,
91.00,24.00,91.00,24.00,
91.00,24.00,91.00,24.00,
91.00,24.00,91.00,24.00,
91.00,24.00,91.00,24.00.
```

Thus, the initial values transmitted from the limited device were obtained.

5. Analysis Results

Using the initial measured value for the initialization on the unconnected and ungrounded analog input of the microcontroller and performing several arithmetic operations according to the proposed algorithm, it is possible to generate a truly random sequence of characters, as long as the length of the message, to ensure absolute cryptoresistance.

If there is a need to perform calculations with the received data and adjust the system operation on the gateway of the Internet of Things system model, it is necessary to use different predefined encryption keys to encrypt the randomly generated keys, for the server and for the gateway, so that in the event of a breach of the gateway, the data to be transmitted to server, remained protected.

6. Conclusions

The developed method makes it possible to eliminate the threat of unauthorized access to data in the gateway-built-in sensor link by encrypting data packets.

Since these algorithms can be used on devices with limited computing resources due to the minimization of calculations, since elementary operations are used for encryption. Encryption reliability in this case is ensured by a unique encryption key for each data packet. To generate random key values, analog noises are used, read from the unconnected input of the microcontroller, so the resulting value is truly random. Preset keys are used to encrypt the keys with which the encrypted message is ciphered. Since message encryption keys are random and unique, encrypting them with preset keys makes it impossible for an attacker to learn the preset key.

In further research, it is necessary to evaluate the reliability of the algorithm for generating random numbers for key generation, in particular, the ability to influence analog noise using electromagnetic radiation and to analyze the developed method of information encryption using the criteria of various performance indicators such as execution time, power consumption, memory requirement for performing calculations.

7. References

[1] N. Srivastava, P. Pandey, Internet of Things (IoT): Applications, Trends, Issues and Challenges, Materials Today, 2022.

[2] F. Yuan, et al., Internet of People Enabled Framework for Evaluating Performance Loss and Resilience of Urban Critical Infrastructures, Safety Science, vol. 134, 2021, 105079.

[3] S. Zeadally, A. K. Das, N. Sklavos, Cryptographic Technologies and Protocol Standards for Internet of Things, Internet of Things, vol. 14, 2021, 100075.

[4] C. Bormann, M. Ersue, A. Keranen. Terminology for Constrained-Node Networks, Internet Engineering Task Force, 2014.

[5] R. Chernenko, et al., Increasing the Security Level of Internet of Things Network Systems Due to Data Encryption on Devices with Limited Computing Resources, Cybersecurity: Education, Science, Technology, vol. 3, no. 11, pp. 124–135.

[6] C. Shannon, Communication Theory of Secrecy Systems, Bell System Technical Journal, vol. 28, no. 4, 1949, pp. 656–715. doi: 10.1002/j.1538-7305.1949.tb00928.x.

[7] K. Rosen, Discrete Mathematics and Its Applications, 6th Ed., McGraw-Hill Edu., 2006.

[8] V. Sokolov, B. Vovkotrub, E. Zotkin, Comparative Analysis of Throughput of Low-Power Wireless IoT Switches, Cybersecurity: Education, Science, Technology, vol. 5, 2019, pp. 16–30.

[9] O. Semko, et al., Methodology of Intelligent Routing Management in Conflicting Sensor Networks of Variable Topology, Modern Special Equipment, vol. 55, no. 4, 2019, pp. 64–76.

[10] C. Henk, (2005). Encyclopedia of Cryptography and Security, Springer Science and Business Media.

[11] C. S. Petrie, J. A. Connelly, A Noise-based IC Random Number Generator for Applications in Cryptography, IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, vol. 47, 2000.

[12] N.G. Bardis, et al., True Random Number Generation Based on Environmental Noise Measurements for Military Applications, in 8th WSEAS International Conference on Signal Processing, Robotics and Automation, 2009.

[13] A. Karpenko, et al., Ensuring Information Security in Wireless Sensor Networks, Cybersecurity: Education, Science, Technology, vol. 2, no. 10, 2020, pp. 54–66. doi: 10.28925/2663-4023.2020.10.5466.

[14] I. Opirskyy, et al., Problems and Security Threats of IoT Devices, Cybersecurity: Education, Science, Technology, vol. 3, no. 11, 2021, pp. 31–42. doi: 10.28925/2663-4023.2021.11.3142.

[15] CSV-1203, CSV File Format Specification, 2012.