DOI: https://doi.org/10.15276/hait.06.2023.22
UDC 004.056.55

Software implementation of the data encryption module on the BeagleBone platform for data transmission systems with increased cryptoresistance

Maksym M. Seniv[1)]
ORCID: https://orcid.org/0000-0003-1044-4628; Maksym.M.Seniv@lpnu.ua. Scopus Author ID: 55816818300
Sviatoslav I. Rovenchak[1)]
ORCID: https://orcid.org/0009-0005-7798-7389; sviatoslav.rovenchak.pz.2019@lpnu.ua
Vitaliy S. Yakovyna[2)]
ORCID: https://orcid.org/0000-0003-0133-8591; yakovyna@matman.uwm.edu.pl. Scopus Author ID: 6602569305
[1)] Lviv Polytechnic National University, 12, Bandera Str. Lviv, 79013, Ukraine
[2)] University of Warmia and Mazury in Olsztyn, 2, Oczapowskiego Str. Olsztyn, 10-719, Poland

ABSTRACT

In today's digital world, where the exchange of information is an integral part of our daily lives, ensuring information security becomes a critical task. **This work aims** to develop an information protection module for data encryption on the BeagleBone platform for data transmission systems with increased crypto resistance. It is a hardware system based on a BeagleBone AI 64 microcomputer with antennas for transmitting/receiving data. Since the information is transmitted over a physically unprotected channel, developing a module that will encrypt the data is necessary. The information protection module ensures the confidentiality of the transmitted data in the system using the AES symmetric encryption algorithm with a variable key length (128/192/256 bits). Regardless of the hardware platform, it is characterized by universality, as it can be run on the Linux kernel adapted for use on embedded systems. Provides options for configuring protocols and encryption algorithms. In the process of developing the security module, modern encryption methods and algorithms were used (AES 128/192/256 – for data encryption, RSA – for the distribution of secret keys), in addition, a lightweight protocol for secure data transmission Scplight was implemented as an alternative to OpenSSL, which improves transfer speed on low-power hardware platforms. The developed information protection module has undergone thorough testing on a real system. The development of the information protection module is based on the need for a universal component that provides high-quality data protection in wireless communication systems. This module will accelerate the development of relatively affordable physical means of secure communication, a critical part of such projects. The module is implemented as a library written in the C language, which implements an API for establishing a secure connection and further forwarding information over an unprotected transmission channel. In addition, a configurator program allows you to change the module settings even in real-time when client applications use it. This ensures continuous, seamless, and secure data exchange and convenient configuration of the module. Many supporting libraries were used in the development process, including Crypton, Libgcrypt, Openssl, Ncurses and Sqlite3.

Keywords: Encryption algorithms; symmetric encryption; data security

For citation: **Seniv M. M., Rovenchak S. I., Yakovyna V. S. "Software implementation of the data encryption module on the BeagleBone platform for data transmission systems with increased cryptoresistance".** ***Herald of Advanced Information Technology.*** **2023; Vol. 6 No. 4: 338–351. DOI: https://doi.org/10.15276/hait.06.2023.22**

INTRODUCTION

Today, data security is a critical aspect of information technology. It plays a crucial role in protecting sensitive and confidential data, including information on government and commercial institutions and users' data. Protection against cybercrime and other threats is becoming increasingly complex due to the constant development of technology and the growth of data volumes, which requires continuous improvement of data security methods and cryptographic algorithms.

Encryption is the primary means of ensuring the confidentiality of data transmitted over communication channels. Information is transferred at the following levels:

– Physical layer – work with the transmission media;

– Data link layer – physical addressing;

– Network layer – route determination and logical addressing;

– The transport layer – a direct connection between the final points;

– Session layer – communication session management;

– The presentation layer – the presentation of data;

– Application layer – access to network services.

Different security protocols are used at each of the above layers.

A form of hardware-implemented encryption is used at the physical layer, for example, frequency spectrum protection.

The data link layer uses tunneling protocols such as PPTP, L2F, and L2TP [1], which are protocols for forming a secure channel.

At the network layer, there is authentication of exchange participants, encryption and tunneling of traffic transmitted between nodes, for example, using the IPSec protocol stack [2].

The session and presentation layers use SSL and TLS protocols [3].

The application layer uses independent data encryption with the help of applications [4].

The development of the information protection module is based on the need for a universal component that provides high-quality data protection in wireless communication systems. This module will accelerate the development of relatively affordable physical means of secure communication, a critical component for such projects.

1. ANALYSIS OF LITERARY DATA

Cryptography has been used for thousands of years to help ensure the privacy of communication [5, 6], [7].

A cryptosystem is a complete specification of keys and how they are used to encrypt and decrypt information. A cryptosystem ensures the security and confidentiality of data by applying cryptographic methods. Encryption algorithms, which are the basis of cryptosystems, ensure the transformation of the original text into encrypted text using mathematical operations. Encryption keys are used to identify and protect access to encrypted data uniquely.

Now, we will consider the types of cryptosystems in more detail. Today, two types are defined: symmetric and asymmetric cryptosystems [4, 5], [6, 7], [8, 9], [10, 11], [12, 13], [14, 15], [16, 17], [18, 19], [20, 21], [22].

In symmetric cryptosystems, the same key is used to encrypt and decrypt a message.

The secret key must be known only to the sender and recipient. A disadvantage of secret-key cryptography is that the two parties must somehow be able to agree on the secret key in advance (before they want to send a message to each other). They can agree on a secret key quite securely when communicating live, but there is a problem with remote locations. One of the possible solutions is the use of asymmetric cryptosystems, or, as they are also called, public key cryptosystems [4-8].

In asymmetric cryptosystems (with a public key), two different keys are used instead of one: a public key, which is known to everyone, and a private key, which is secret and known only to the owner. A message encrypted with a public key can only be decrypted with the corresponding private key. Asymmetric encryption is used for purposes such as key exchange and digital signature.

Public-key cryptography eliminates the need for two parties to agree on a secret key beforehand. However, a way to distribute public keys securely remains to be devised. Public key certificates are one of the common methods of combating this problem [7, 13], [14].

Cryptosystems are also usually divided into the categories of stream ciphers and block ciphers.

Stream encryption performs data operations on a byte-by-byte basis. It uses a key and generates a sequence of pseudo-random bits combined with the plaintext to produce the ciphertext. One of the most famous stream encryption algorithms is RC4. Stream encryption often transmits a rapid data stream, such as network traffic or video streaming.

Block encryption divides the plaintext into blocks of a fixed size (e.g., 64 or 128 bits) and encrypts each block independently. Block encryption algorithms are used to encrypt entire blocks of data. The most common block encryption algorithm is AES (Advanced Encryption Standard). It uses 128-bit blocks and supports keys of different lengths (e.g., 128, 192, or 256 bits) [23, 24], [25, 26], [27, 28], [29].

2. THE PURPOSE AND OBJECTIVES OF THE RESEARCH

This work aims to develop an information protection module for data encryption on the BeagleBone platform for data transmission systems with increased crypto resistance. It is a hardware system based on the BeagleBone AI 64 microcomputer [30] with antennas for transmitting/receiving data. The security module should provide means for secure information transfer while providing clients with a user-friendly interface that hides the implementation itself and abstracts the entire internal process at the level of available APIs. In addition, the information protection module must store its configuration in the database. These parameters will determine the parameters of its operation. Interaction with this database must be done through a configurator program with a user interface.

The following actual scientific and practical problems are solved in the work:

– Developing a software tool for secure information transmission on the BeagleBone platform for data transmission systems with increased crypto-resistance. The AES 128/192/256 algorithm was used for data encryption, and the RSA algorithm was used for the distribution of secret keys;

– development of a lightweight Scplight secure data transfer protocol as an alternative to OpenSSL;

– Test the developed software and verify the received data.

3. RESEARCH METHODS

The BeagleBone AI-64 microcomputer is a high-performance single-board computer with a 64-bit processor based on the ARM Cortex-A15 architecture. This computer supports a suite of interfaces, including Ethernet, HDMI, and USB, as well as support for network protocols such as Wi-Fi and Bluetooth.

The BeagleBone AI-64 differs from other single-board computers because it has a specialized intelligent processor supporting artificial intelligence, including machine learning and neural networks. This makes it ideal for AI-related projects such as pattern recognition, natural language processing or robot management.

Encryption was performed using the AES (Advanced Encryption Standard) algorithm with an encryption block size (the block with which the algorithm works internally; its size is not variable) of 128 bits. Different key lengths were also used (possible values are 128/192/256 bits).

4. RESEARCH RESULTS

This work aims to develop an information protection module that will be used as a component of a data transmission system consisting of transmitting and receiving devices. Information is shared between these devices using antennas. Since the data will be sent in an open, unprotected environment, it is necessary to implement a module to protect this information. The use of the wireless method of data transmission is determined by the faster and cheaper deployment of such a system. That is why the relevance of implementing such a module is immediately apparent. The wireless environment provides everyone with the right equipment and open access to data, which is worth noting. It is relatively cheap and widespread to purchase and use. The system is based on the BeagleBone AI 64 microcomputer, a single-board computer developed by BeagleBoard.org. This computer is based on the Texas Instruments AM5729 processor, which has two Cortex-A15 and four Cortex-M4F cores. This allows the BeagleBone AI 64 to combine microcontrollers' computing power and flexibility on a single board.

The Scplight lightweight data transfer protocol was designed to implement the information protection module. The lightweight protocol was designed using a standard scheme where the channel configuration parameters are synchronized first, after which direct data transfer begins. The first stage – handshake (synchronization), allows you to inform the opposite side of the communication about the parameters used to enable the other side to configure local structures and work mechanisms, in particular, to choose the desired implementation of the security protocol and to configure the correct encryption algorithms. At this stage, the RSA algorithm is used to deliver the secret key to the other party of the communication. The second stage involves encryption and sending/receiving data. AES symmetric encryption algorithm is used to ensure information confidentiality.

The format of the protocol header files is as follows:

[**Message Type | Message Length | Message Data**]

Let's consider each of these fields separately:

The message type is defined as a 1-byte field that facilitates handling Scplight protocol messages. There are four types available to choose from, presented in Table. 1. The message length is defined as a 4-byte field containing the payload's length.

The message data is defined as a data array with the size of “Message Length”. Depending on the type of Scplight, the message may contain either “bare” data (DATA type) or data in configuration format (HANDSHAKE type).

The format of the configuration parameters is defined as follows:

[Security Protocol | Cryptographic Algorithm]

Table 1. **Scplight security protocol message types**

Numerical Value	Message Type	Purpose
0	HANDSHAKE	Used during the handshake phase
1	ACK	Used as confirmation, helps to synchronize the parties of communication
2	DATA	Indicates a message with user data
3	CLOSE	Used to suspend the use of a secure channel

Source: **compiled by the authors**

Let's consider these fields in more detail.

The security protocol is a 1-byte field. Valid values are presented in Table 2.

Table 2. **Valid values of secure data transmission protocols in the configuration header**

Numerical Value	Protocol
0	Scplight
1	OpenSSL

Source: **compiled by the authors**

The cryptographic algorithm is a 1-byte field. Valid values are presented in Table 3.

Table 3. **Valid values of cryptographic algorithms in the configuration header**

Numerical Value	Cryptographic algorithm
0	AES-128
1	AES-192
2	AES-256

Source: **compiled by the authors**

The real-time synchronization mechanism allows remote reconfiguring of the security module, for example, from the access server (ACS). In implementing this capability, two main approaches were tested: reconfiguration via a separate channel and using a data link for configuration synchronization (single channel). As a result, the latter was chosen.

The first method involves establishing at least two connections. The first of them is service. It is created at the beginning of the system operation and is used only to synchronize the work of the communication parties. Initial messages are transmitted using a predefined secure data transfer protocol. All client data is transmitted over a separate data connection. This approach has several advantages: less traffic must be sent, and there are no minimum delays for processing configuration data in the single-channel method. However, we also have disadvantages, in particular: the logic of the system's operation becomes much more complicated, and end-to-end synchronization of various parts of the module is necessary. For example, during the configuration process, you must prohibit data exchange.

In contrast, the single-channel approach has a much simpler implementation and does not require additional synchronization between the parties. The price is an additional service header to each user data message. The other party analyzes them and determines the configuration change. The mechanism is designed according to a similar principle to the Scplight protocol, which uses empty ACK and CONFIG messages (containing new, updated operating parameters).

It was decided to implement the information protection module as a library. Two methods are available on the user side: writing data (on the transmitter side) and reading data (on the receiver side). The encryption module performs all work on securing this data per the system's configured parameters.

Another variant of its execution was also considered, namely in the form of a separate program. This approach has several advantages, including better isolation of the environment. At the same time, it has disadvantages: to perform each operation, it is necessary to run this program with the required parameters. If data is sent at specific intervals, time will constantly be lost to establish a secure connection.

In case of successful login, we will go to another window, which, depending on the user's role, contains different selection options. In particular, when logging in with the "User" access level, only the "Configure module" option will be available for selection without the possibility of changing any settings – only viewing. The "Manager users" tab will not be available for this role.

The entire menu is designed as lists that expand to the left when you press the "Enter" key or the right arrow. Pressing the left arrow will collapse the current menu. For convenience, selecting a list item at the lowest level automatically collapses it. A complete exit from the configurator is implemented by pressing the "Q" key. Each security protocol has its list of parameters that can be configured. Even though they currently have a typical setup, this approach is much more flexible for further system development.

When designing the interface of the configurator program, heuristics of interface design were used, including consistency and standards, stylish and minimalist design, flexibility and efficiency of use.

Also, for the information protection module, a class diagram was developed and presented in Fig. 1, a deployment diagram (Fig. 2), and a sequence diagram to display the algorithm of the Scplight security protocol (Fig. 3).

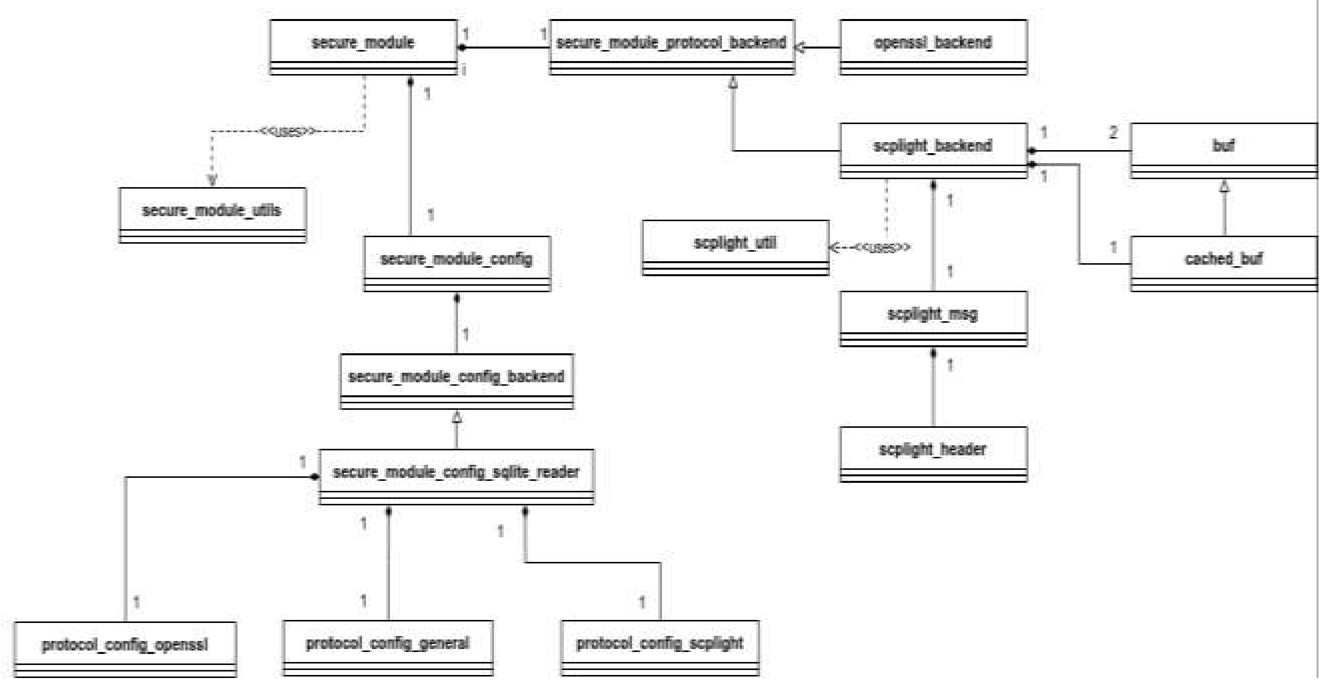

Fig. 1. **Class diagram for the information protection module**
Source: **compiled by the authors**

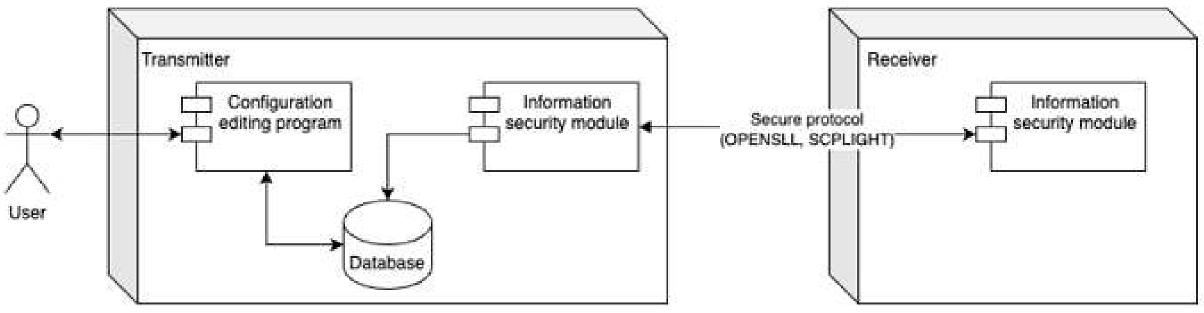
Transmitter
Configuration editing program
Information security module
User
Database
Secure protocol (OPENSLL, SCPLIGHT)
Receiver
Information security module

Fig. 2. **Deployment diagram for the information protection module**
Source: **compiled by the authors**

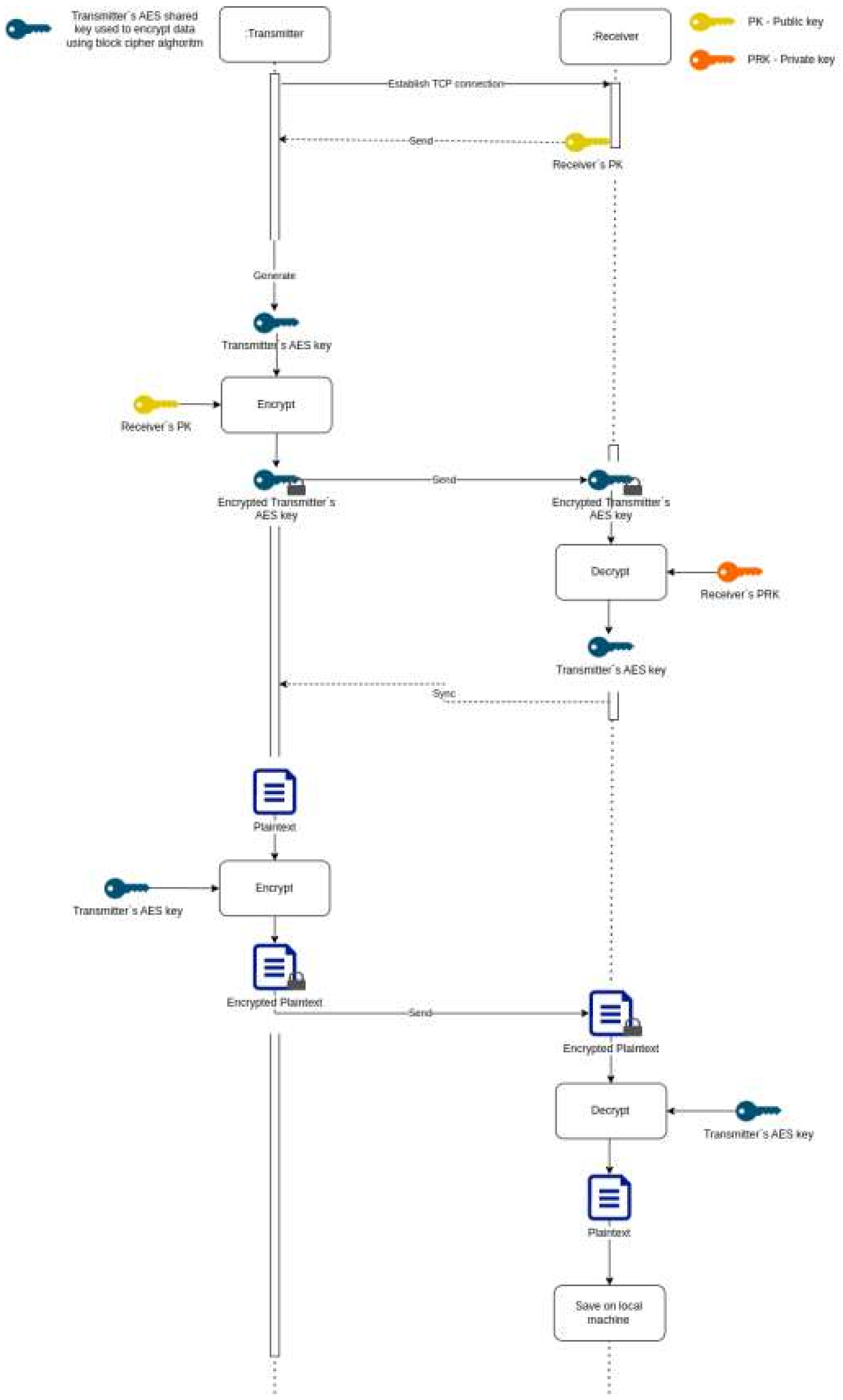

Fig. 3. **Sequence diagram to display the working algorithm of the Scplight security protocol**
***Source:* compiled by the authors**

In the data protection module library, a generalized interface for working with security protocols was implemented using pointers to functions that mimic the behavior of virtual functions. This approach was used to implement support for two different security protocols: TLS and Scplight. The secure_module entity contains the general logic of working with security protocols. When using this object, first, the socket descriptor is set, and the process of connecting/accepting clients is followed by data transfer. These steps are the same regardless of the choice of data transfer protocol, so these steps are described using function pointers that refer to the desired implementation, depending on the configured data protection module configuration. Binding to the required functions occurs when creating an object of the secure_module type.

When initializing a security module object, the configuration is read first. Depending on the selected parameters, these pointers are initialized by the corresponding functions.

Another essential component of the system is implementing a separate protocol for encrypted data transmission, which is easy and fast to send, as it implements only the minimum required for secure communication. The implementation is contained in the logical entity Scplight_backend. The operation of the protocol itself is described by a state machine, which explains its logic in a more precise, defined and structured way to prove the correctness of the operation of such an essential and basic unit. No generative templates were used during its description; all states are described directly using enumerations and the logic is defined in switch-case blocks.

One of the advantages of this information protection module is the ability to define the configuration dynamically. In real-time, without interrupting the work of the client application, changes are applied to a secure channel with a change in key sizes or a complete change in the secure protocol for data transfer. From the outside, users continue to use the write and read data APIs without needing additional calls or validations. The configuration change detection was implemented by monitoring the database file in the file system. This approach is possible because Sqlite3 is implemented as a serverless database, and therefore, we have a guarantee of changes to the configuration file only when we perform similar operations ourselves. This architecture allows you to manage the configuration update process effectively and avoid changes to the database file without updates. From the library's side, this means unquestioning interpretation of all changes to such a file as module configuration updates. When creating a data protection module object, the client side runs the monitor logic in a separate thread, which uses the inotify mechanism implemented by the Linux kernel. Built on events, it allows you to reduce the load on the system compared to constantly checking the status of files using select – an alternative monitoring method. In addition, it provides a broader range of events that can be tracked, including modification, creation, deletion, and movement of files and folders. In general, using inotify provides a more productive and flexible way to monitor file system changes compared to other methods. Accordingly, the companion thread uses a minimum of resources. For single-core systems, it is also possible to implement similar behavior within the main thread (for example, for systems running on Cortex-M4 series processors). Considering that the Beagle Bone Ai has a dual-core Arm Cortex-A72, the decision was made to run the monitor in a separate thread to preserve maximum performance when transferring data in general.

An accompanying sm_conf program was developed for access control (to change module parameters) and configuration of its parameters. The user interface is implemented using the Ncurses library and its derivative menu library, which provides a convenient means of creating control primitives using a symbolic representation. This decision was made due to the greater versatility of this approach: windowed environments usually require more system resources, which could be more efficient in the context of embedded systems. To facilitate the understanding of the multi-level menu system, which has prospects to grow significantly in the development and addition of new features, an auxiliary menu abstraction was implemented on top of the menu. It supports submenus and various hooks that allow you to write the logic for multiple events conveniently and clearly. In particular, the preshow hook, launched every time a specific menu is drawn, will enable you to provide an optimized approach to accessing data in the database.

4. EXPERIMENT AND RESULTS

A series of experiments was carried out to test the performance of the developed module.

Encryption was performed using the AES (Advanced Encryption Standard) algorithm with an encryption block size (the block with which the algorithm works internally; its size is not variable) of 128 bits. By the size of the block (chunk) in Fig. 4 and Fig. 5, we mean the portion of encrypted information (the size of the file). Different lengths of keys were also used (possible values – 128/192/256 bits). For each configuration of the symmetric AES-128/192/256 algorithm, with different sizes of data chunks to be encrypted per call, the timed tests were run five times for 10 seconds. During this time, we measured the amount of data that was managed to be encrypted. After the measurements were obtained, the average value of the speed was calculated. In addition, each of these tests was run six more times. From the results of these tests, new average values were obtained, calculated by the mean square formula to minimize system load problems on the BeagleBone AI-64 (Fig. 4, Fig. 5, Fig. 6 and Fig. 7).

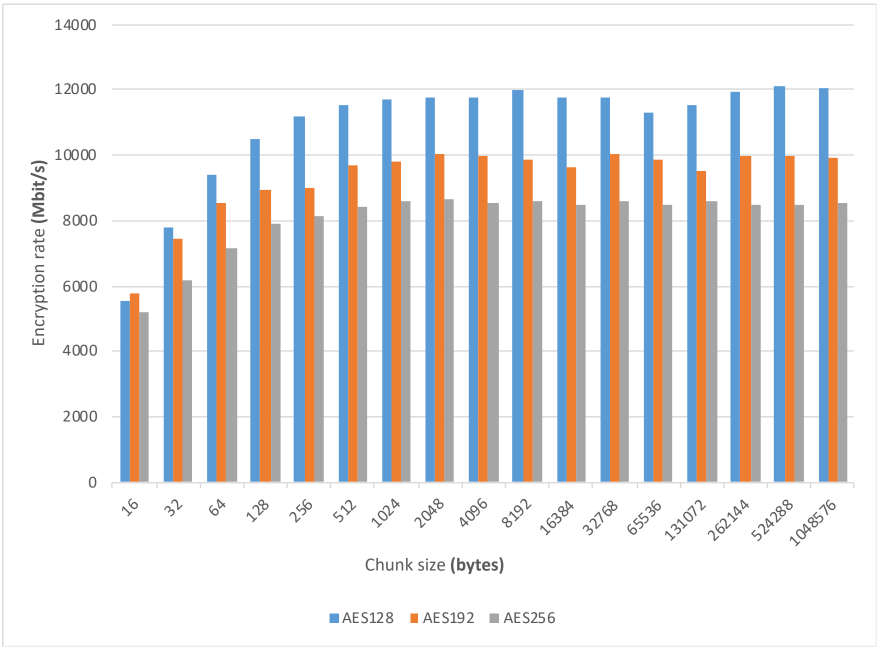

Fig. 4. **Data encryption speed using AES depending on the size of the transmission block**
Source: **compiled by the authors**

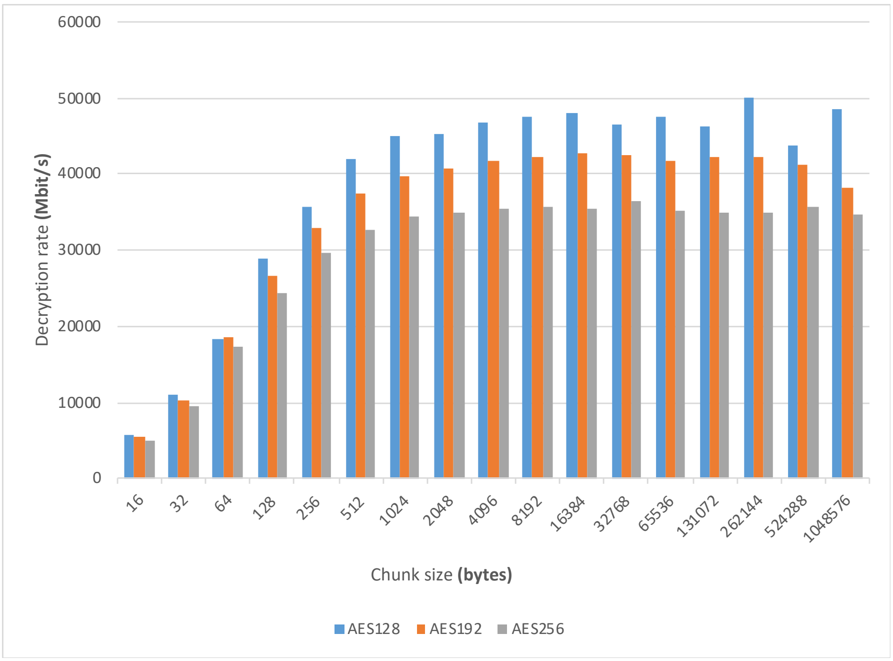

Fig. 5. **AES data decryption speed depending on the size of the transmission block**
Source: **compiled by the authors**

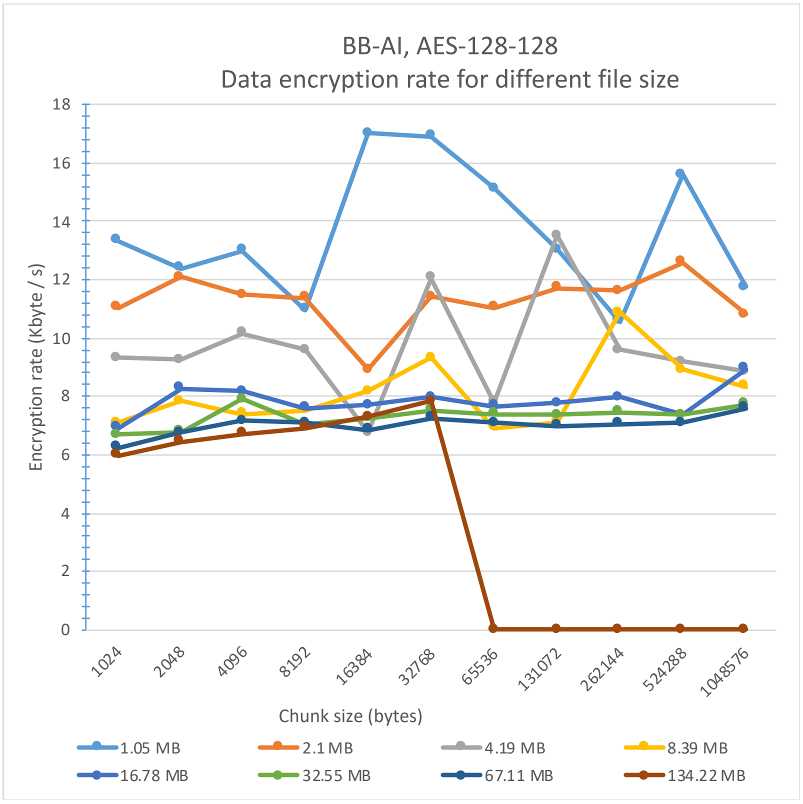

Fig. 6. **The speed of information encryption depending on the size of the encrypted file and the portion of the encrypted information**
Source: **compiled by the authors**

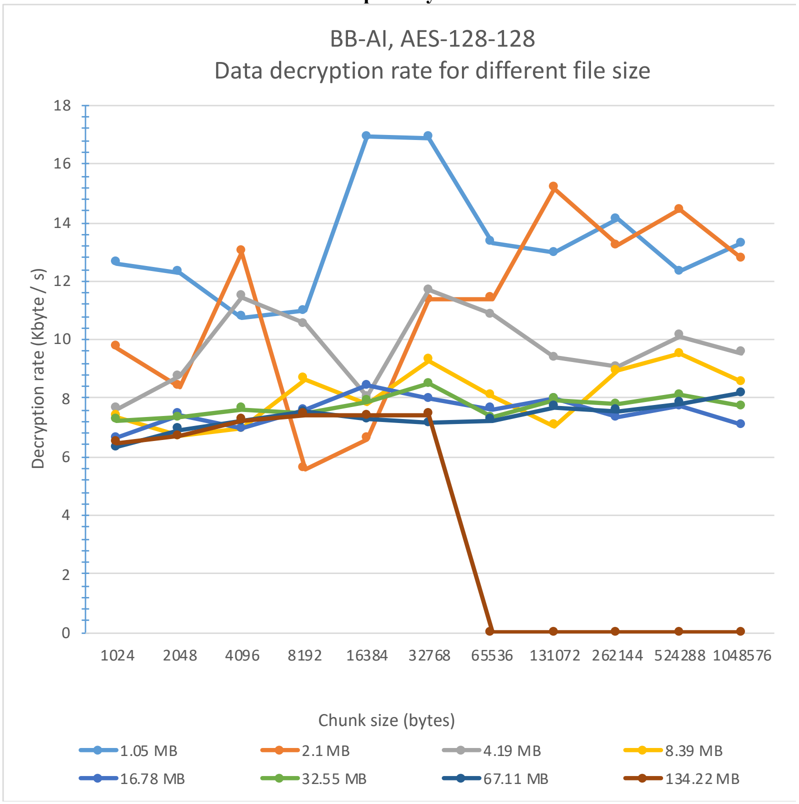

Fig. 7. **The speed of information decryption depending on the size of the encrypted file and the portion of the encrypted information**
Source: **compiled by the authors**

Encryption speed. During experiments on the BeagleBone-AI platform, a significant decrease in the rate of encryption/decryption of information was noticed due to the uneven distribution of resources by the system itself. In particular, the subsidence was 75-86 % of the maximum speeds. For example, one series of test data determined the encryption speed of a 1 MB file with a portion size of 16 Kb as 16.64 Kb/s. In contrast, another series of tests gave a rate of 2.29 Kb/s (Fig. 6). Considering this feature of the platform, all values were calculated using the root mean square (RMS) formula to reduce the weight of test samples with a significantly reduced speed in the entire mass of conducted tests.

Decryption speed. According to the obtained results, it can be concluded that the highest rates can be achieved for small files with a portion size of 16Kb and 32Kb. The best universal data portion sizes are 32 KB and 524 KB. As the file size increases, the speed decreases by half from 16+ Kb/s to ~8 Kb/s (Fig. 7). At the same time, it can be seen that "average" portion sizes work best on smaller files, while for large data – the largest ones.

On the BeagleBone-AI, there is no significant difference between encryption/decryption speeds. However, in some cases, encryption worked slower (maximum by 15 %).

CONCLUSIONS AND PROSPECTS OF FURTHER RESEARCH

This work describes the developed information protection module for data encryption (based on the AES symmetric encryption algorithm) on the BeagleBone platform for data transmission systems with increased crypto resistance. It is a hardware system based on a BeagleBone AI 64 microcomputer with antennas for transmitting/receiving data. The security module provides the means for secure information transfer while providing clients with a user-friendly interface that hides the implementation and abstracts the entire internal process at the level of available APIs. In addition, the information protection module stores its configuration in the database.

A lightweight Scplight security protocol has been developed, an essential component of the entire system. A description of its implementation is provided, the central part of which is a defined state machine, which makes it possible to describe the behavior of the entire protocol deterministically and clearly. An explanation is also provided regarding the operation of the real-time security module configuration change, which at the primary level is implemented through database file monitoring, taking into account the architectural features of the SQLite3 library. In addition, a basic description of the internal structure of the configurator program has been added: the main components of the implementation of the user interface, including a fairly developed abstraction of menu work, written on top of the ncurses library.

During experiments on the BeagleBone-AI platform, a significant decrease in the speed of encryption/decryption of information was noticed due to the uneven distribution of resources by the system itself. In particular, the subsidence was 75-86% of the maximum speeds. According to the obtained results, it can be concluded that the highest rates can be achieved for small files with a portion size of 16Kb and 32Kb. The best universal data portion sizes are 32 KB and 524 KB. As file sizes increase, the speed is halved from 16+ Kb/s to ~8 Kb/s. At the same time, it can be seen that "average" portion sizes work best on smaller files, while for large data – the largest ones.

In the future, the authors plan to work on increasing the data transfer speed and the possibility of cross-platform implementation of this module.

REFERENCES

1. Gaur, K., Kalla, A., Grover, J., Borhani, M., Gurtov, A. & Liyanage, M. "A survey of virtual private LAN services (VPLS): Past, Present and Future". *Computer Networks.* 2021; 196: 108245, https://www.scopus.com/authid/detail.uri?authorId=57211786560.
DOI: https://doi.org/10.1016/j.comnet.2021.108245.

2. Abolade, O., Okandeji, A., Oke, A., Osifeko, M. & Oyedeji, A. "Overhead effects of data encryption on TCP throughput across IPSEC secured network". *Scientific African.* 2021; 13: e00855, https://www.scopus.com/authid/detail.uri?authorId=57222750137.
DOI: https://doi.org/10.1016/j.sciaf.2021.e00855.

3. Elgohary, A., Sobh, T. S. & Zaki, M. "Design of an enhancement for SSL/TLS protocols". *Computers & Security*. 2006; 25 (4): 297306, https://www.scopus.com/authid/detail.uri?authored=14009987800. DOI: https://doi.org/10.1016/j.cose.2006.02.007.

4. Seniv, M. M. & Yakovyna, V. S. "Data and software security" (in Ukrainian).Textbook. Lviv: *Lviv Polytechnic Publishing House*. 2015.

5. Yakovyna, V. S., Fedasyuk, D. V., Saliy, S. I. & Seniv, M. M. "Study of cryptoresistance characteristics of the DES symmetric encryption algorithm" (in Ukrainian). *Bulletin of Lviv Polytechnic National University. Computer systems of design. Theory and practice*. 2008; 626: 55–62.

6. Yakovyna, V. S., Odukha, O. V., Seniv, M. M. & Bilas, O. Ye. "Study the main characteristics of the RC5 symmetric encryption algorithm for constructing the protection module of the distributed thermal design system" (in Ukrainian). *Bulletin of Lviv Polytechnic National University. Computer Science and Information Technologies*. 2008; 616: 143–150.

7. Yakovyna, V. S., Fedasyuk, D. V., Seniv, M. M. & Bilas, O. Ye. "Comparison of the speed of software implementation of symmetric (DES) and asymmetric (RSA) encryption algorithms" (in Ukrainian). *Bulletin of Lviv Polytechnic National University. Computer Science and Information Technologies*. 2007; 598: 181–185.

8. Shariatzadeh, M., Rostami, M. J. & Eftekhari, M. "Proposing a novel Dynamic AES for image encryption using a chaotic map key management approach". *Optik*. 2021; 246: 167779, https://www.scopus.com/authid/detail.uri?authorId=57226896674.
DOI: https://doi.org/10.1016/j.ijleo.2021.167779.

9. AbdelWahab, O. F., Hussein, A. I., Hamed, H. F. A., Kelash, H. M. & Khalaf, A. A. M. "Efficient combination of RSA cryptography, lossy and lossless compression steganography techniques to hide data". *Procedia Computer Science*. 2021; 182: 5–12, https://www.scopus.com/authid/detail.uri?authorId=57210557983. DOI: https://doi.org/10.1016/j.procs.2021.02.002.

10. Adhikari, S., Ray, S., Obaidat, M. S. & Biswas, G. "Efficient and secure content dissemination architecture for content centric network using ECC-based public key infrastructure". *Computer Communications*. 2020; 157: 187– 203, https://www.scopus.com/authid/detail.uri?authorId=57204426493. DOI: https://doi.org/10.1016/j.comcom.2020.04.024.

11. Ahmad, N. & Hasan, S. M. R. "A new ASIC implementation of an advanced encryption standard (AES) crypto-hardware accelerator". *Microelectronics Journal*. 2021; 117: 105255, https://www.scopus.com/authid/detail.uri?authorId=36665886500.
DOI: https://doi.org/10.1016/j.mejo.2021.105255.

12. Huo, X. & Wang, X. "Internet of things for smart manufacturing based on advanced encryption standard (AES) algorithm with chaotic system". *Results in Engineering*. 2023; 20: 101589, https://www.scopus.com/authid/detail.uri?authorId=58158665900.
DOI: https://doi.org/10.1016/j.rineng.2023.101589.

13. Stinson, D. R. & Paterson, M. "Cryptography: theory and practice". 4th ed. *New York: Chapman and Hall/CRC*. 2018, https://www.scopus.com/authid/detail.uri?authorId=23168455900.

14. Liu, B. "Research and implementation of RSA IP Core Based on FPGA". *Advances in Intelligent Systems and Computing*. 2020; 1088: 1311–1319, https://www.scopus.com/authid/detail.uri?authorId=57208684489. DOI: https://doi.org/ 10.1007/978-981-15-1468-5_154.

15. Inam, S., Kanwal, S., Firdous, R., Zakria, K. & Hajjej, F. "A new method of image encryption using advanced encryption Standard (AES) for network security". *Physica Scripta*, 2023; 98 (12): 126005, https://www.scopus.com/authid/detail.uri?authorId=56997275200. DOI: https://doi.org/10.1088/1402-4896/ad0944.

16. Shivaramakrishna, D. & Nagaratna, M. "A novel hybrid cryptographic framework for secure data storage in cloud computing: Integrating AES-OTP and RSA with adaptive key management and Time-Limited access control". *Alexandria Engineering Journal*. 2023; 84: 275–284, https://www.scopus.com/authid/detail.uri?authorId=58688897100. DOI: https://doi.org/10.1016/j.aej.2023.10.054.

17. Nagaraju, S., Nagendra, R., Balasundaram, S. & Kiran Kumar, R. "Biometric key generation and multi round AES crypto system for improved security". *Measurement: Sensors.* 2023; 30: 100931, https://www.scopus.com/authid/detail.uri?authorId=57211773047.
DOI: https://doi.org/ 10.1016/j.measen.2023.100931.

18. Karthika, S. K. & Prasad Jones Christydass, S. "A novel AES and Chacha based block level processing oriented image encryption scheme". *International Journal of Information Technology*. 2023: 15 (8): 4085–4095, https://www.scopus.com/authid/detail.uri?authorId=57471820400.
DOI: https://doi.org/10.1007/s41870-023-01503-4.

19. Fernando, E., Agustin, D., Irsan, M., Murad, D. F., Rohayani, H. & Sujana, D. "Performance comparison of symmetries encryption algorithm AES and DES with Raspberry Pi". *International Conference on Sustainable Information Engineering and Technology (SIET).* Lombok: Indonesia. 2019. p. 353–357, https://www.scopus.com/authid/detail.uri?authorId=57189355900.
DOI: https://doi.org/10.1109/SIET48054.2019.8986122.

20. Nitaj, A., Susilo, W. & Tonien, J. "Enhanced S-boxes for the advanced encryption standard with maximal periodicity and better avalanche property". *Computer Standards and Interfaces*, 2023; 87: 103769, https://www.scopus.com/authid/detail.uri?authorId=6506961835.
DOI: https://doi.org/ 10.1016/j.csi.2023.103769.

21. Dai, X., Wang, X., Qu, X., Mao, B. & Hu, W. "Fault Analysis on AES: A Property-Based Verification Perspective". *Tsinghua Science and Technology.* 2023; 29 (2): 576–588, https://www.scopus.com/authid/detail.uri?authorId=58652127900.
DOI: https://doi.org/10.26599/TST.2023.9010035.

22. Santoso, P. P., Rilvani, E., Trisnawan, A. B., Adiyarta, K., Napitupulu, D., Sutabri, T. & Rahim, R. "Systematic literature review: Comparison study of symmetric key and asymmetric key algorithm". *IOP Conference Series: Materials Science and Engineering.* 2018; 420 (11): 012111, https://www.scopus.com/authid/detail.uri?authorId=57204184720. DOI: https://doi.org/10.1088/1757-899X/420/1/012111.

23. Sultana, Sk. A., Ch, R. & Malleswari, R. P. "Keyless lightweight encipher using homomorphic and binomial coefficients for smart computing applications". *ViTECoN 2023 – 2nd IEEE International Conference on Vision towards Emerging Trends in Communication and Networking Technologies.* 2023, https://www.scopus.com/authid/detail.uri?authorId=58107553000.
DOI: https://doi.org/10.1109/ViTECoN58111.2023.10157660.

24. Banoth, R. & Regar, R. "Classical and modern cryptography for beginners". *Springer Nature.* 2023, https://www.scopus.com/authid/detail.uri?authorId=58400513300. DOI: https://doi.org/10.1007/978-3-031-32959-3.

25. Vidhya, K., Nagarajan, B., Aisvarya, S., Anuprabha, K. & Ashley, R. "Enhanced cloud storage security using elliptic curve cryptography and entity-based access control". *AIP Conference Proceedings.* 2023; 2764 (113): 060011, https://www.scopus.com/authid/detail.uri?authorId=57213451336. DOI: https://doi.org/10.1063/5.0145589.

26. Wright, C. P., Dave, J. & Zadok, E. "Cryptographic file systems performance: What you don't know can hurt you". In *Security in Storage Workshop, SISW'03. Proceedings of the Second IEEE International.* 2003. p. 47–47, https://www.scopus.com/authid/detail.uri?authorId=12646346100. DOI: https://doi.org/10.1109/SISW.2003.10005.

27. Yenuguvanilanka, J. & Elkeelany, O. "Performance evaluation of hardware models of Advanced Encryption Standard (AES) algorithm". In *Southeastcon, IEEE.* 2008. p. 222–225, https://www.scopus.com/authid/detail.uri?authorId=247220926002008.
DOI: https://doi.org/10.1109/SECON.2008.4494289.

28. Farooq, U. & Aslam, M. F. "Comparative analysis of different AES implementation techniques for efficient resource usage and better performance of an FPGA". *Journal of King Saud University – Computer and Information Sciences.* 2016, https://www.scopus.com/authid/detail.uri?authorId=8305505000. DOI: https://doi.org/10.1016/j.jksuci.2016.01.004.

29. Brown, J. A., Houghten, S. & Ombuki-Berman, B.”Genetic algorithm cryptanalysis of a substitution permutation network”. *IEEE Symposium on Computational Intelligence in Cyber Security, CICS.* 2009; 4925098, https://www.scopus.com/authid/detail.uri?authorId=55724408300. DOI: https://doi.org/10.1109/ CICYBS.2009.4925098.

30. “BeagleBone AI-64” – Available from: https://beagleboard.org/ai-64. – [Accessed: 01 June 2023].

Conflicts of Interest: the authors declare no conflict of interest

Received 14.09.2023
Received after revision 07.12.2023
Accepted 15.12.2023

DOI: https://doi.org/10.15276/hait.06.2023.22
УДК 004.056.55

Програмна реалізація модуля шифрування даних на платфоромі BeagleBone для систем передавання даних з підвищеною криптостійкістю

Сенів Максим Михайлович[1)]
ORCID: https://orcid.org/ 0000-0003-1044-4628; Maksym.M.Seniv@lpnu.ua. Scopus Author ID: 55816818300
Ровенчак Святослав Ігорович[1)]
ORCID: https://orcid.org/0009-0005-7798-7389; sviatoslav.rovenchak.pz.2019@lpnu.ua
Яковина Віталій Степанович[2)]
ORCID: https://orcid.org/0000-0003-0133-8591; yakovyna@matman.uwm.edu.pl. Scopus Author ID: 6602569305
[1)] Національний університет «Львівська політехніка», вул. Степана Бандери, 12. Львів, 79013, Україна
[2)] Варміньсько-Мазурський університет в Ольштині, вул. Очаповського 2. Ольштин, 10-719, Польща

АНОТАЦІЯ

У сучасному цифровому світі, де обмін інформацією є невід'ємною частиною нашого повсякденного життя, забезпечення безпеки цієї інформації стає надзвичайно важливим завданням. **Метою даної роботи** є розробка модуля захисту інформації для шифрування даних на платформі BeagleBone для систем передавання даних з підвищеною криптостійкістю. В загальному це апаратна система на основі мікрокомп’ютера BeagleBone AI 64 з набором антен для передачі/приймання даних. Оскільки інформація передається по фізично незахищеному каналі, потрібно розробити модуль, який шифруватиме дані. Модуль захисту інформації забезпечує конфіденційність переданих даних в системі та характеризується універсальністю незалежно від апаратної платформи, оскільки може запускатися на ядрі Linux, пристосований для використання на вбудованих системах, надає можливості для конфігурації протоколів та алгоритмів шифрування. Шифрування – це оборотне перетворення даних, з метою приховування інформації. Шифрування відбувається із застосуванням криптографічного ключа. Ключ – це певна кількість символів, сформованих вільним чином з символів, що доступні у системі шифрування. Методи шифрування в основному поділяють на симетричні та асиметричні. У процесі розробки модуля безпеки були використані сучасні методи і алгоритми шифрування, крім того, був реалізований спрощений алгоритм для безпечної передачі даних, що покращує швидкість передачі на малопотужних апаратних платформах. Розроблений модуль захисту інформації пройшов ретельне тестування на реальній системі. Розробка модуля захисту інформації обґрунтована необхідністю універсальної компоненти, яка забезпечує високу якість захисту даних у безпровідних системах комунікації. Цей модуль дозволить прискорити розробку порівняно доступних фізичних засобів безпечного зв'язку, який є критично важливою складовою для таких проектів. Модуль реалізований у вигляді бібліотеки, написаній на мові C, яка реалізує API для встановлення безпечного з'єднання та подальшого безпечного пересилання інформації по не захищеному каналу передачі. Додатково, існує програма-конфігуратор, яка дозволяє змінювати налаштування модуля навіть в режимі реального часу, коли той використовується клієнтськими застосунками. Це забезпечує безперервний, безшовний та безпечний обмін даними, а також зручне налаштування модуля. В процесі розробки було використано багато допоміжних бібліотек, зокрема crypton, Libgcrypt, Openssl, Ncurses та Sqlite3.

Ключові слова: Алгоритми шифрування; симетричне шифрування; безпека даних

ABOUT THE AUTHORS

Maksym M. Seniv - PhD, Associate Professor of Software Department. Lviv Polytechnic National University, 12, Bandera Str. Lviv, 79013, Ukraine
ORCID: https://orcid.org/ 0000-0003-1044-4628; Maksym.M.Seniv@lpnu.ua. Scopus Author ID: 55816818300
Research field: Reliability of the software; software development methodologies; data security

Сенів Максим Михайлович - кандидат технічних наук, доцент. Доцент кафедри Програмного забезпечення. Національний університет «Львівська політехніка», вул. Степана Бандери, 12. Львів, 79013, Україна

Sviatoslav I. Rovenchak - Student of Software Department. Lviv Polytechnic National University, 12, Bandera Str. Lviv, 79013, Ukraine
ORCID: https://orcid.org/0009-0005-7798-7389; sviatoslav.rovenchak.pz.2019@lpnu.ua
Research field: Data security

Ровенчак Святослав Ігорович - студент кафедри Програмного забезпечення. Національний університет «Львівська політехніка», вул. Степана Бандери, 12. Львів, 79013, Україна

Vitaliy S. Yakovyna - Dr hab., Associate Professor. Faculty of Mathematics and Computer Science. University of Warmia and Mazury in Olsztyn, 2, Oczapowskiego St. Olsztyn, 10-719, Poland
ORCID: https://orcid.org/0000-0003-0133-8591; yakovyna@matman.uwm.edu.pl. Scopus Author ID: 6602569305
Research field: Software reliability and safety; machine learning; computational intelligence

Яковина Віталій Степанович - доктор технічних наук, професор. Професор факультету Математики та комп'ютерних наук. Вармінсько-Мазурський університет в Ольштині, вул. Очаповського, 2. Ольштин, 10-719, Польща