Mail client with data transfer protected with end-to-end encryption

Gleb Polozhiy[1] and *Nikolay* Boldyrikhin[2*]

[1]Limited Liability Company "Titanium", office 201, building 37-19, Kotlostroitelnaya St., Taganrog, 347910, Rostov region, Russia
[2]Don State Technical University, 1, Gagarin Sq., 344002, Rostov-on-Don, Russia

Abstract. This article is devoted to the study of the issues of secure messaging and file exchange by e-mail at the enterprises of the agro-industrial complex. The purpose of this work is to develop an algorithmic support and software of mail client that supports the functionality of end-to-end data encryption. The following research methods are used in the article: comparison method, observation method, structural analysis method. The comparison method is used to identify differences between key distribution protocols. Methods of observation and analysis make it possible to understand the behavior of various protocols within systems that differ in technical and software properties. The article consists of four parts: Introduction, Materials and Methods, Results, Conclusions. The first part describes the relevance of the topic, discusses the features of various cryptographic protocols. The second part describes the developed algorithms that ensure the operation of the mail client and end-to-end data encryption. The third part discusses the results of software implementation of the developed algorithms. In the fourth part conclusions are made and summed up. The main results of the work are algorithmic implementation of software for local architecture based on the use of symmetric cryptography; a software tool designed according to a local scheme, including modules that allow you to develop a system on a client-server architecture.

1 Introduction

Currently, information security issues are of great importance, including for enterprises of the agro-industrial complex. A modern agricultural enterprise is not only automated product processing lines, but also an extensive IT-infrastructure, process control dispatch centers, automated warehouses, complex information exchange, including by e-mail [1]. Of great interest to agro-industrial enterprises using public networks are aspects related to the confidentiality of the information they transmit [1-16]. However, it is very difficult to identify cases of eavesdropping and traffic analysis. Therefore, to prevent such cases, various types of encryptions of transmitted data are used [6-16], including end-to-end encryption [10, 11]. The main advantage of end-to-end encryption is that only the recipient

* Corresponding author: boldyrikhin@mail.ru

and sender of the message can decrypt the transmitted data. Thus, the confidentiality of information exchange is ensured. Even though end-to-end encryption technologies are not new and have been around for a long time, they have not been widely used for several reasons. Firstly, many service providers are not interested in protecting the privacy of user data, since the distribution of this information to advertising services owners brings more profit. Secondly, almost all software tools where these technologies are implemented require self-management and key exchange, which is quite difficult even for the most technically literate users. In view of this, the topic of this work is relevant.

The object of research is information processes occurring in information systems for the transmission of electronic mail messages. The subject of the study is the encryption of data during their transmission by means of information systems for the transmission of postal electronic messages. The aim of the work is to develop an algorithmic support and software of mail client that supports the functionality of end-to-end data encryption. To achieve this goal, it is necessary to solve several tasks: to analyze cryptographic protocols for key distribution; implement information support for the development of the mail client; develop a software implementation of the mail client.

When implementing an email client with the transfer of messages and attachments protected by end-to-end encryption, the following protocols were selected for secure data exchange for comparison: Diffie-Hellman protocol, MTI protocol (Matsumoto, Takashita, Imai), STS protocol (Station-To-Station) and modifications of these protocols. Next, we will analyze the vulnerabilities of each of them to determine the most appropriate protocol for the software, considering its technical and algorithmic features.

The Diffie-Hellman protocol is a cryptographic protocol that allows two or more subscribers to generate a common secret “word” using a communication channel that is open for listening. The key obtained in this way is used to encrypt the subsequent exchange of messages using symmetric encryption cryptographic algorithms [13, 14]. The Public Key Distribution Order published by Hellman and Diffie was a breakthrough in the world of encryption because it eliminated the problem of key distribution. However, the weakness of the Diffie-Hellman protocol is the lack of authentication of the parties.

MTI are key distribution protocols that were proposed by T. Matsumoto, Y. Takashita, and H. Imai to prevent a man-in-the-middle attack on the Diffie-Hellman scheme [14, 15]. MTIs are public key distribution protocols that rely on the exchange of messages between parties A and B and the calculation of a secret session key. However, these protocols also have disadvantages the MTI/A0 protocol is able to provide key authentication for side A, but cannot provide key authentication for side B. This is a significant drawback of this algorithm.

The STS protocol is a protocol for obtaining a shared secret key over an open channel [16]. A positive feature of the protocol is the two-way confirmation of the key. However, for the considered version of the protocol, a two-way attack with an unknown common key is applicable. As a result, users C and D, who formed a coalition, mislead users A and B, who formed a common key. At the same time, user A is sure that it has formed a shared key with user C, and user B is sure that it has formed a shared key with user D.

Thus, all considered key distribution protocols have vulnerabilities, so the choice of the protocol in favor of Diffie-Hellman is due to the simplicity of implementation, in comparison with MTI/A0 and STS. The Diffie-Hellman vulnerability is offset using the SSL protocol.

The AES cryptographic algorithm is used to encrypt the contents of emails, ensuring secure data transmission from the sender to the recipient without the possibility of viewing this information by third parties.

2 Materials and methods

To develop an email client with data transfer protected by end-to-end encryption, we will define the following practical aspects: the software tool has a client-server architecture; interaction between the server and the client is carried out using the Hyper Text Transfer Protocol Secure (HTTPS) protocol using a certificate certified by a certification authority; Diffie-Hellman protocol is used to provide end-to-end encryption. As additional protection measures, eliminating, in particular, the vulnerability of the Diffie-Hellman protocol to the "man in the middle" attack, the SSL protocol and a certificate of a certification authority are used.

To use the software, the user must create an account. To register, you need to provide data that will then be used for identification and authentication: username and password.

To provide end-to-end encryption concepts, the account password, both during registration and authorization, is transmitted to the server using the Diffie-Hellman protocol. Before being sent, it is converted by the client into a shared key based on the server's public key and the user's initial password. The password should not be known to anyone except the user, since it is used to build a secret key for encrypting and decrypting emails with a symmetric encryption algorithm. Thus, the server does not receive the real password and has no way to "pull" it from the shared key.

If the registration is successful, the server creates a user account in the database. A Diffie-Hellman public key is also automatically generated for the new user. It is later used to create a shared key between the sender and recipient of the email. Since in the event of information leakage from the database, an attacker who has gained access to the account will be able to log in under it, user passwords are stored in a closed form using the PBKDF2 standard with the SHA256 hash and do not allow such a vulnerability. However, it is worth noting that even in the case when an attacker can get a shared key, he will not be able to decrypt messages when he logs into the account, since the secret key of the encryption algorithms is the user's password in its original, which is stored in the local memory of the account owner's device.

Additional protection of user data is provided by Hyper Text Transfer Protocol Secure using a certificate from a trusted server. Thus, all data inside requests to the server is encrypted, and this does not allow an attacker to carry out attacks by listening to a network connection (sniffer attacks, man-in-the-middle attacks, etc.).

To log in, the user needs to go through the authentication procedure: enter and send a name and password. Also, as in the case of registration, the password is converted into a Diffie-Hellman pre-shared key before the form is submitted. Then, in the form, the entered password is replaced with the obtained public key. The form is sent to the server. As a result, the client receives a response about the success or failure of the operation. Thus, the user performs registration and authorization before starting work with e-mail. At this stage, the public (on the server) and private (on the client) keys of the user are created and stored, which are necessary for encrypting and decrypting data when exchanging electronic messages.

To send and view messages from a mail account, it must be linked to a client account. To do this, a form is filled in with the data of the mail account and sent to the server. E-mail and password are specified, the information is stored in the database for further use in the interaction between the server part of the client and mail services. The mail password is stored privately using the AES cryptographic algorithm in CBC mode using a 128-bit key that is not stored in the database. Thus, passwords from mail accounts in the event of a database leak will not be available to an attacker.

The number of linked mail service accounts is not limited. However, there is a ban on linking a mail account that is already linked to another user. This is due to the fact that

when sending a message to users with the same mail, the system will not be able to determine whose public key must be taken for further encryption or decryption. For this reason, the data may not be decrypted correctly at the endpoints.

After binding, the user can enter the mail account management interface. At each login, before responding to the client, the server queries the IMAP or POP3 email access protocols for the contents of the mail. Further, the received information is stored in the server database. Since the messages are encrypted and it is impossible to access them without the user's password, the principles of end-to-end encryption are not violated.

The received messages inside the base views have the architecture of the models. Thus, a tool can universally access all models and their objects using the same set of commands, queries, or functions.

Message files are also stored in encrypted form and are presented as models that store the path along which they are located in the data storage (hard drive, cloud storage, etc.).

Then the server receives a list of letters from the requested user (determined by the session key), arranges them as needed, and issues a response to the client. The client displays the received list to the user. The algorithm for receiving and viewing messages is shown in Figure 1.

The client receives a response from the server. In the list of letters, not all can be sent when using a secure email client. Thus, some of them will not be encrypted and do not need to be decrypted. Such messages are displayed to the user immediately without any additional processing.

Messages sent using this tool are encrypted and require additional decryption procedures before being displayed. Such letters are marked with a special header before being sent, which makes it possible to uniquely identify them among the general set.

The cryptographic algorithm used to encrypt and decrypt data is AES in CBC mode with a 128-bit key. The secret key is generated based on the public key of the Diffie-Hellman protocol. The public key is represented as the sender's or recipient's key when encrypted or decrypted, respectively. The private key is the user's password, which is located on the client and is not transmitted anywhere outside of it.

Thus, having received encrypted messages, the client generates an AES secret key and decrypts them. The subject, content and files are stored in encrypted form in the letter.

Sending an email can take place in two scenarios:

1. To the email address of a person who does not have an email client account. In this case, since the recipient's public key is missing, no encryption will be performed. The letter will be sent in clear text.

2. To the email address of the person who has an email client account. In this case, the client will receive the recipient's public key from the server, form a common key between the sender and the recipient, obtain an AES secret key based on it, and encrypt the message: subject, content, and files.

Let us consider the second case in more detail. The client, having encrypted the message, sends it to the server. The server establishes a connection with the mail service via the SMTP protocol and sends a letter on behalf of the mail account from which the user performed the action. The server then saves a copy of the message to the Sent Items folder via IMAP or POP3 (depending on the mail service).

Thus, using the Diffie-Hellman protocol and the AES cryptographic algorithm, secure data transmission from the sender to the recipient is ensured without the possibility of viewing this information by third parties.

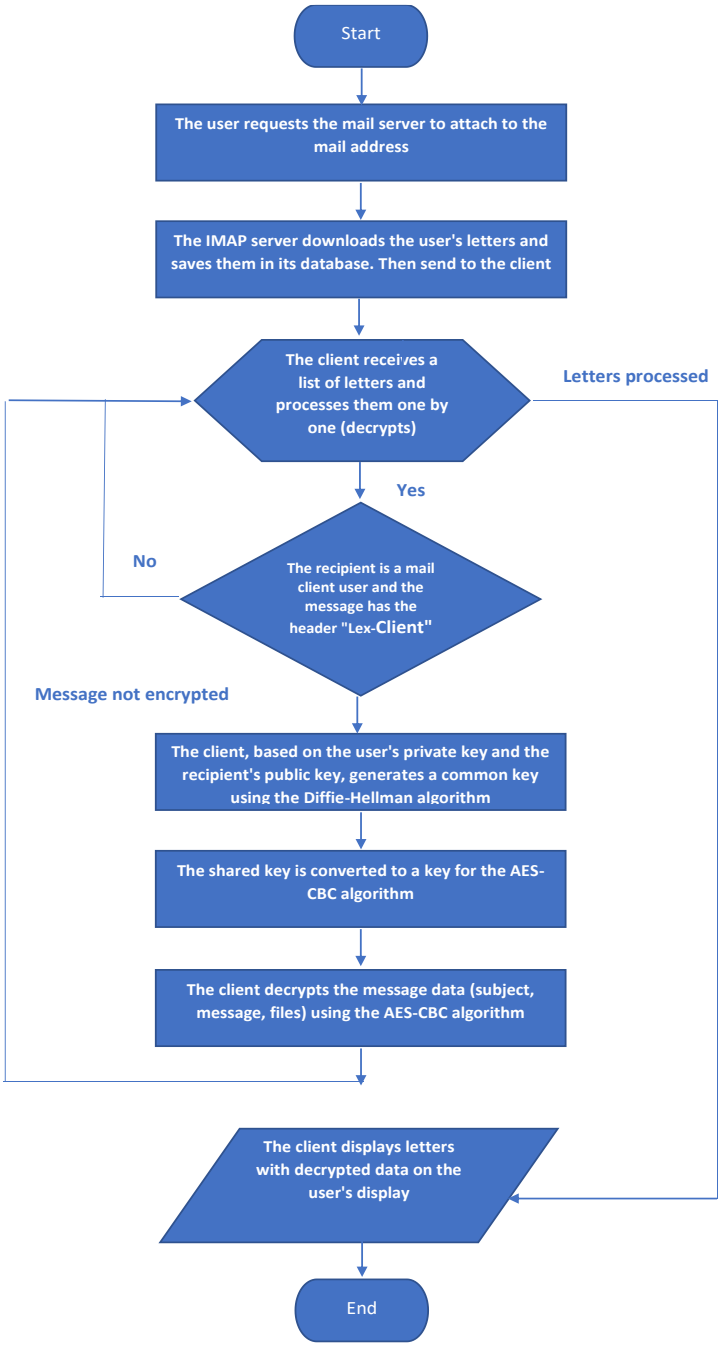

Fig. 1. Algorithm for receiving messages.

3 Results

Consider the software implementation of the developed algorithms.

The client part of the software is an interface for communication between a user and an email client and implements encryption and decryption of emails.

The user begins his interaction with the software through the registration window. The user needs to fill out a form: enter a login, name, and password, which will then be used by him to log into his account in the mail client.

After filling, the user clicks the "Register" button. Next, the "onclick" browser event fires, which fires the trigger function. This function converts the user's password into a shared Diffie-Hellman key between the server and the client. Thus, the server does not receive the password in the clear and has no way of knowing it in any way. The public key of the server is always present inside the HTML code of the page and is substituted there every time the pages are rendered by the server. Thus, the server "shares" the key with the clients.

After the form is processed by the script, it is sent to the server. Then it is received by the SignUpView handler function and creates a new user within the system.

After registration, the user must go through the authentication and authorization procedures. To do this, he must enter the login and password specified during registration. On login, the trigger function converts the password into a shared key between the server and the client and submits the form to the server. There, the handler function performs authentication and authorization. If successful, a session is established and a session key is returned, which is then used in all requests that require read or write access from the user. If unsuccessful, a list of errors will be returned to the user.

After a successful login, the user is redirected to a web page for viewing and editing personal account information. Login and password cannot be changed. Also, this data is not provided for viewing by the user.

Next, the user is taken to the email connection page. On this page, the user can connect a new e-mail or change the data for authorization in an existing one.

To work with the mail account, a separate web page is provided with the interface shown in Figure 2.

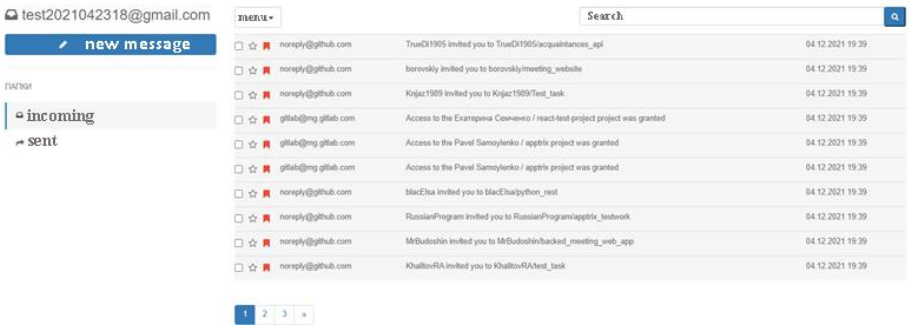

Fig. 2. Mail account management interface.

To get a web page, a GET request is made to a special handler named SMTPAccountDetailView. Before rendering the page, it calls the mail service and requests a list of emails that have not yet been uploaded to the tool's database in order to retrieve and store them. After that, the page is rendered: a list of letters is built in an HTML document. The listing only shows the subject, sender, and date the email was received. The user can choose which folder to view messages from.

Having received a response from the server with a ready web page, the client decrypts the encrypted part of the letter.

The user fills out the form: specifies the recipient's address, subject and content. Attaches files as needed. Separately, it is worth noting that when entering a recipient, GET requests are made to the server using AJAX, which return information on the presence of such mail in the mail client. Thus, if such mail is found, then the transmission of the letter will be protected by end-to-end encryption using the Diffie-Hellman protocol. Moreover, the public key of the user with such mail will be returned in the same GET request.

By clicking on the "Send message" button, the lexEncrypt encryption function is launched. First, it checks to see if the recipient is on the mail client's system. If it is not present, then the function does nothing and immediately sends a POST request to the server. Otherwise, based on the recipient's public key and the user's password, which is stored in the browser's local memory, a pre-shared Diffie-Hellman key is generated. This key is then converted into the secret key of the AES algorithm. Text and file information is encrypted. The form overrides the values with the new encrypted ones and sends it to the server as a POST request to the SMTPAccountDetailView handler. The handler then sends the message to the mail service server using the SMTP protocol and makes a copy and saves it through the IMAP, POP3 or other protocol (depending on which mail service is used).

4 Conclusions

As part of these studies, the following was done:

- cryptographic protocols of key distribution were analyzed;
- implemented algorithmic support of the mail client with the transfer of data protected by end-to-end encryption;
- the architecture of the final software tool was formulated, consisting of client and server parts, each of which performs its own functionality to ensure the transfer of data protected by end-to-end encryption.

Thus, the tasks set are solved, the purpose of the study is achieved.

References

1. V. Bolek, A. Látečková, A. Romanová, F. Korček, AGRIS on-line Papers in Economics and Informatics **8(4)**, 1-14 (2016) https://doi.org10.7160/aol.2016.080404
2. A. Vedadi, M. Warkentin, A. Dennis, Information & Management **58(8)**, 103526 (2021) https://doi.org/10.1016/j.im.2021.103526
3. W. Stallings, Computer security: principles and practice (Pearson, Boston, 2012)
4. N. Lyashenko, K. Rysyatova, L. Chemerigina, et al., Lecture Notes in Networks and Systems **246**, 164–172 (2022) https://doi.org/10.1007/978-3-030-81619-3_18
5. Y. Kovtun, L. Cherckesova, E. Revyakina, O. Safaryan, E. Roshchina, Smart Innovation, Systems and Technologies **247**, 65–71 (2022) https://doi.org/ 10.1007/978-981-16-3844-2_8
6. M. Babayan, A. Buglak, N. Gordov, I. Pilipenko, L. Cherckesova, O. Safaryan, Lecture Notes in Networks and Systems **246**, 154–163 (2022) https://doi.org/ 10.1007/978-3-030-81619-3_17
7. H. Pourbabak, T. Chen, W. Su, The Energy Internet. Woodhead Publishing, 181-199 (2019) https://doi.org/10.1016/B978-0-08-102207-8.00008-4

8. L. Cherckesova, O. Safaryan, E. Pinevich, et al., E3S Web of Conferences (2021) https://doi.org/10.1051/e3sconf/202127308045
9. O. Safaryan, L. Cherckesova, N. Boldyrikhin, et al., Journal of Physics: Conference Series **2131(3)**, 032112 (2021) https://doi.org/10.1088/1742-6596/2131/3/032112
10. M. A. Jan, W. Zhang, M. Usman, Z. Tan, F. Khan, E. Luo, Journal of Network and Computer Applications **137**, 1-10 (2019) https://doi.org/10.1016/j.jnca.2019.02.023.
11. C. Borcea, A. D. Gupta, Y. Polyakov, K. Rohloff, G. Ryan Future Generation Computer Systems **71**, 177-191 (2017) https://doi.org/10.1016/j.future.2016.10.013.
12. L. Harn, C. Lin, Computers & Electrical Engineering 40(6), 1972-1980 (2014) https://doi.org/10.1016/j.compeleceng.2013.12.018.
13. D. Roh, S. G. Hahn Information Processing Letters **110(18–19)**, 799-802 (2010) https://doi.org/10.1016/j.ipl.2010.07.001.
14. T. Matsumoto, Y. Takashima, H. Imai, Transactions of the Institute of Electronics and Communication Engineers of Japan. Section E—Elsevier BV 69(2), 99—106
15. C. Boyd, A. Mathuria A (2003) Springer Science+Business Media **321**, 147—155 (2003) https://doi.org /10.1007/978-3-662-09527-0
16. D. Liu, W. Liu, J. Xu, Applied Mechanics and Materials **644-650**, 2202-2205 (2014) https://doi.org/10.4028/www.scientific.net/AMM.644-650.2202.