Development of a Server Software Module for Protected Data Sharing on the Internet

Osipov D.L.
dmitry.os@gmail.com

Tebueva F.B.
fariza.teb@gmail.com

Ryabtsev S.S.
nalfartorn@yandex.ru

Struchkov I.V.
selentar@bk.ru

North Caucasus Federal University, Stavropol, 355009, Russia Federation

Abstract

This article leads a research on the development of a method and a program module evaluating of developing a server software module for providing secure data exchange based on the open instant messaging protocol Matrix and standardized encryption functions are considered. A comparative analysis of some instant messaging protocols has also been performed.

1 Introduction

Nowadays the information sphere is being actively developed. The quantity and value of the processed information increases. Although the data processing systems are being continuously improved, the information secutrity threats remain relevant. According to InfoWatch reports, in 2017, more than 90% of leaks were related to the compromising of personal data and payment information; small and medium-sized companies accounted for 77% of leaks[Rep17].

Often the existing software does not provide the required security levels in accordance with Russian regulatory standards in the field of cryptography [Fed06]. Therefore, there is a need to develop the software that would meet these requirements.

2 Development of methodology

2.1 Examination of existing data exchange protection technologies

The location of the secure data transmission protocols in the OSI model is shown in the figure 1. The most important among them are:

- SSL, TLS protocols are used to provide a secure connection in HTTPS (however, SSL is already outdated);
- PPTP is used to establish a secure point-to-point tunnel;
- IPsec network layer protocol to protect Internet connections.

In: S. Hölldobler, A. Malikov (eds.): Proceedings of the YSIP-3 Workshop, Stavropol and Arkhyz, Russian Federation, 17-09-2019–20-09-2019, published at http://ceur-ws.org

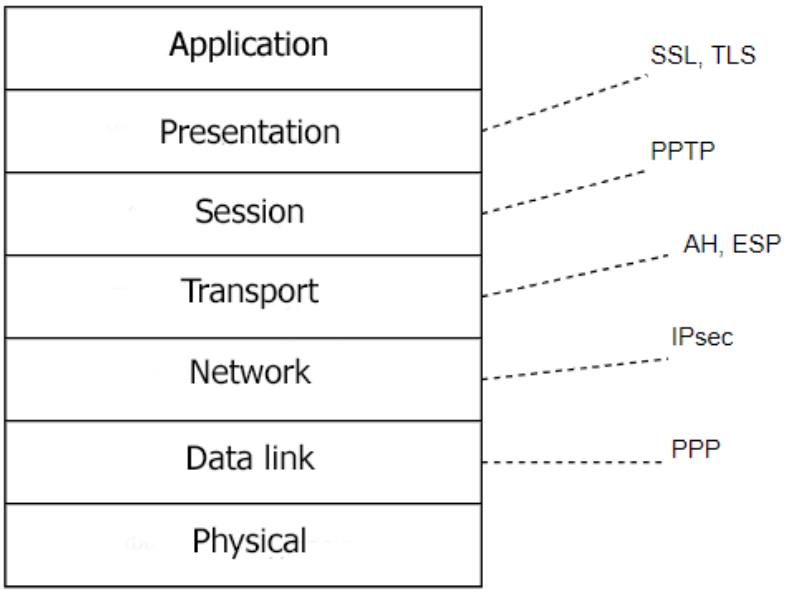

Figure 1: Protection of data exchange at different levels of the OSI model

The lower the protocol in the model, the more transparent and efficient it works, but this complicates its development, and vice versa - the higher the protocol in the model, the easier it is to develop, but the more difficult it is to integrate into the application, as it becomes less transparent and should be created integration for it in every software.

Based on the above, the most rational solution is to use an application-level protocol with an integrated standardized cryptographic system in accordance with the legislation of the Russian Federation.

Consider the comparison of the most popular protocols for messengers, displayed in the table 1 [Mat19, Mtp19, Jam19, Xmp19, Jin19].

The most logical to use is Matrix, since it has broad functionality and has the following features :

- JSON is used as a container for messages, which will simplify the implementation of cryptography and universalize the protocol;
- open standards and detailed specifications;
- fully supports federated communication of server modules of the application.

Table 1: Comparison of existing messenger protocols

#	Protocol	Media sync	Audio / Video Transfer	Group chats	Documentation	Federated
1	Matrix	+	+	+	complete	+
2	MTProto	+	Audio only	+	not complete	-
3	Jami	-	+	+	not complete	-
4	XMPP/Jingle	+	+	not completely	complete	+

Having the possibility of audio / video calls, will allow to establish a secure communication channel for meetings with employees located outside the office.

2.2 Matrix protocol analysis

Matrix is an open protocol for real-time communication. It is designed so that users with accounts from one service provider can communicate with users of another service provider through online chat, voice over IP and video telephony. Thus, it aims to ensure uninterrupted real-time operation between different service providers,

just as standard email using a simple mail transfer protocol is now used for email service with saving and forwarding.

From a technical point of view, it is an application-level communications protocol for real-time federated communications. It provides the HTTP API and open source reference implementations for safely distributing and storing messages in JSON format across an open server federation. It can integrate with standard Web services through WebRTC, facilitating browser-browser applications.

Other attempts to define an open protocol for instant messaging or multimedia signaling of this type were hampered, becoming widespread, for example. XMPP and IRCv3 both highlighted technical and political issues [Suk03]. It is unclear whether users have enough demand for services that interact between providers. Since Matrix runs on top of HTTP, it is at the OSI application layer.

The client-server API provides an easy-to-use, lightweight API that allows customers to send messages, monitor rooms, and synchronize their conversation history. It is designed to support both lightweight clients that do not save state and deferred data from the server as needed, as well as heavy clients that maintain a full local permanent copy of the server state.

Before turning to the cryptographic model, consider the structure of the protocol and data containers (JSON).

A mandatory baseline for client-server interaction in Matrix is the exchange of JSON objects through the HTTP API. HTTPS is recommended for communication, although HTTP can be supported as a fallback to support basic HTTP clients [Yus18]. In addition, all API calls use the Content-Type application json. In addition, all strings MUST be encoded as UTF-8. Clients are authenticated using opaque access _ token strings (for details, see Client Authentication), passed as a parameter to the query string in all requests.

API endpoint names for HTTP transport follow an underscore convention to separate words (for example, delete_devices). Key names in JSON objects passed through the API also follow this convention. The analysis of the Matrix specification allows us to distinguish three main entities, systems, figure 2:

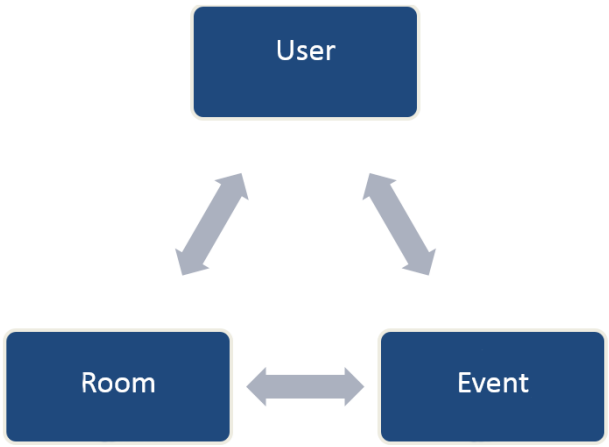

Figure 2: Simplified Data Model in Matrix

Users create events that are recorded in rooms (attached to them in chronological order). Figure 2 shows a simplified diagram of rooms and events in them.

For each event, its author is stored in the "Sender" field. To make requests, each user uses an authorization token, which he receives in the authentication process with the input of a normal login and password.

Using JSON as a container for transferring data and REST as a software interface allows the protocol to be as versatile as possible and independent of the platform or programming language. Thus, virtually all modern programming languages and platforms support the Matrix protocol.

Each Room entity is a parent for Event type entities. Thus, each room is the parent object for all events, and the users themselves are also attached to the rooms (as participants) and can generate events. Text messaging is one of the subtypes of Matrix events.

To separate an object in Matrix, the type parameter is used, with values in the format t1.t2.tn, for example, m.room.message.

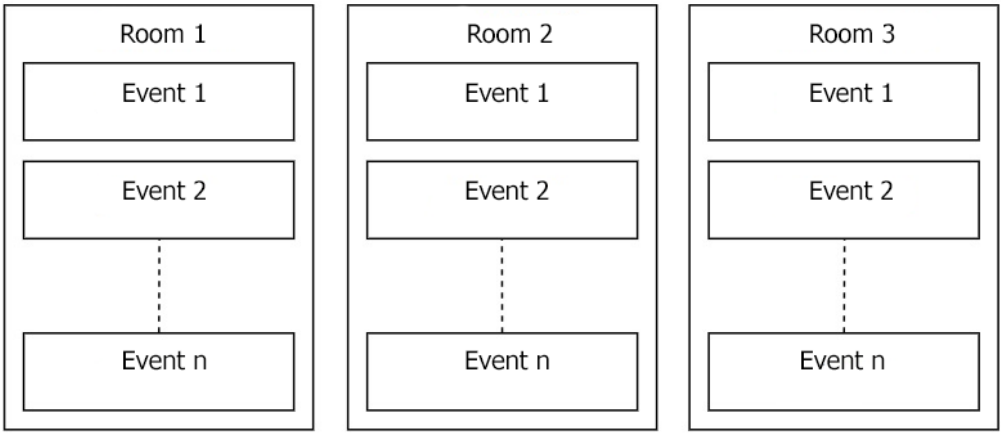

Figure 3: The structure of rooms and events in the Matrix protocol

2.3 Protection of the data exchange process

From the above list of events, the most important is the secure transmission of "Message" events. Events of this type, in turn, are divided into different subtypes (text messages, audio messages, files, etc.), however, within the initial phase of the study, the task of cryptographic protection of text messages is posed. The JSON structure of the text message object is shown in figure 4. This object was received by clients as a result of a request to receive new events to the server (the unnecessary part of the request-response from the server is not shown in the figure).


```
{
   "content": {
      "body":"Здравствуйте! Вас беспокоит СКФУ",
      "format":"org.matrix.custom.html",
      "formatted_body":"Здравствуйте! Вас беспокоит <b>СКФУ</b>",
      "msgtype":"m.text"
   },
   "event_id":"$5640752234dPrZf:domain.com",
   "origin_server_ts":1432735924034,
   "room_id":"!jEsURKDJejlrcePyGS:domain.com",
   "sender":"@example:domain.com",
   "type":"m.room.message",
   "unsigned": {
      "age":1234
   }
}
```


Figure 4: The structure of the event "Text message"

The "body" and "formatted_body" fields contain the text of the received message in the normal and formatted form, respectively. Both fields need cryptographic protection, as well as fields:

- "origin_server_ts" - a label containing the date and time of sending the message with an accuracy of milliseconds;
- sender - unique identifier of the sender;

- event_id - unique identifier of the message;
- room_id - unique identifier of the message;
- age - time from the moment of sending the message.

Figure 5 shows a simplified cryptographic model of event cryptographic protection.

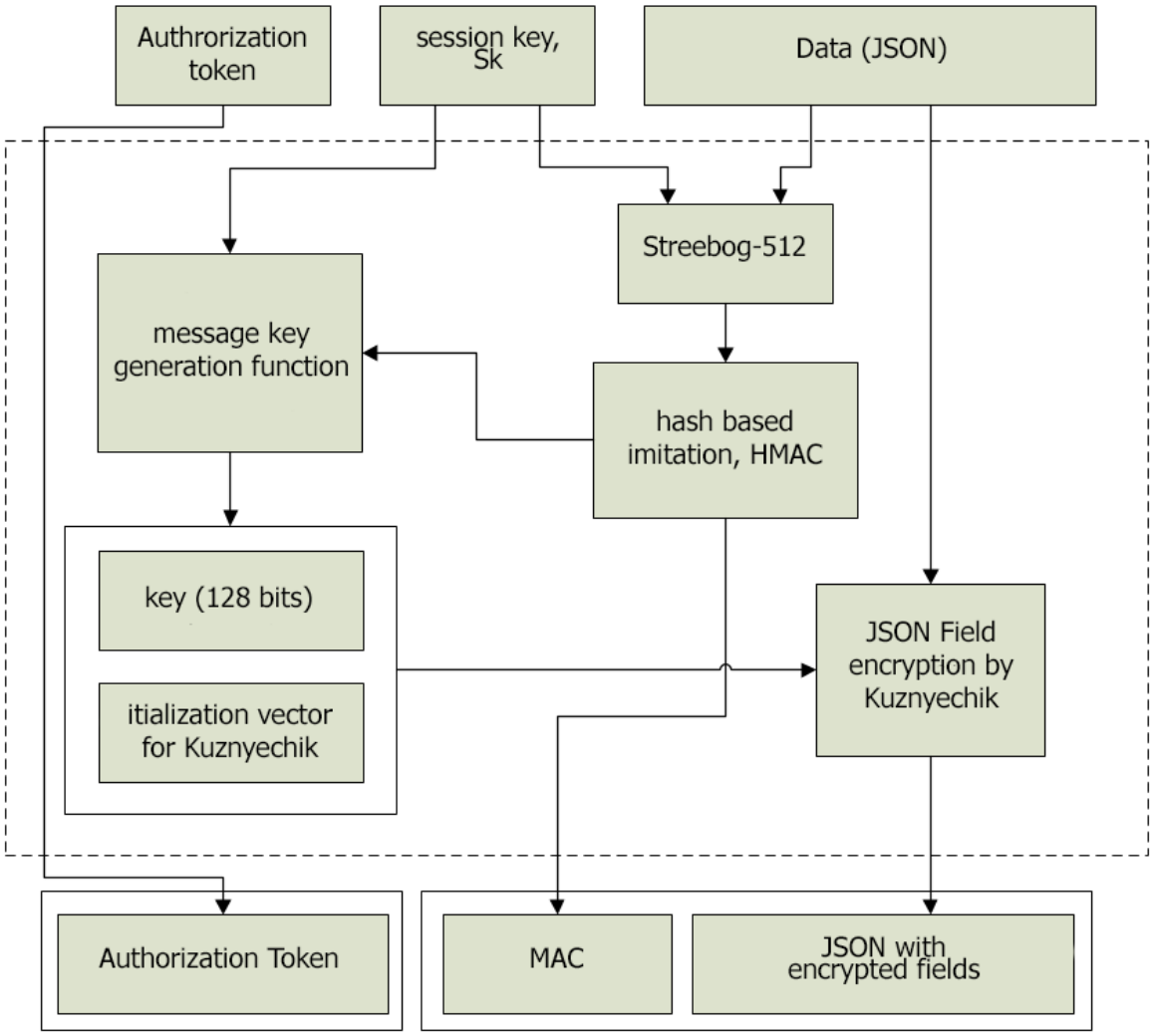

Figure 5: Simplified event cryptographic protection scheme

In the Matrix protocol, the system of access tokens is used for authorization and identification (it is this token that is displayed in the diagram). The scheme also contains the following elements:

- session key Sk, which is unique;
- message authentication code to check on the other side of authenticity;
- Kuznyechik is a symmetric block encryption algorithm;
- Stribog-512 - a hash algorithm with a hash length of 512 bits.

During authentication, the user receives a token and, further, in each request must use it as an identifier and secret for authorization.

```
{
   "content": {
      "body":"U2FsdGVkX18zVl9w/5D6O4lXl02+t3sObz+Q1mBs8E0tM0OaTb4GapPfZVTKk5kC",
      "format":"org.matrix.custom.html",
      "formatted_body":"U2FsdGVkX1DwIu0C73V8GT6dYFEZwCNMe2a2F3htv06qOGAEV19lDNPV1
      4wTGeSYwdSGEWnkSe4NmkKj5NB5g==",
      "msgtype":"m.enctext",
      "macode":"ZS3PcFdBfVTDI2cxospHMwuICSNrh9qJkXQD/qrDTqXb6TjuYP3WBjg7ON+b
      USOdJZaN3xwERcDnq0SChxl9ZA"
   },
   "event_id":"U2FsdGVkX1+H0Un4pzZxfYP4h4vyKyaulqqJTbnotrfrEUZ+9TRtl4iOOfkFVltx",
   "origin_server_ts":"U2FsdGVkX192x+w04AtDxvdV7/dybQjeUq5iH/7RO8w=",
   "room_id":"U2FsdGVkX1/2cqDTttD6/M1qISuFNpOaS0wSQ33KhPFzuzGUUc7MluU+BSkA1pnq",
   "sender":"U2FsdGVkX1/VHZgsavC+O/wAZwnMeDIy+NIjldrLPnOL0gLSwJon0yMOfSmXt5RT",
   "type":"m.room.encmessage",
   "unsigned": {
      "age":"U2FsdGVkX1//t5dUjw/17VHhsiho+BUvngqirJLLDvM="
   }
}
```


Figure 6: JSON structure after cryptographic protection

Since the token is transmitted with each HTTP request, its use as a key source for cryptographic protection of messages may be unsafe.

Let us further consider the Stribog hash function.

GOST R 34.11-2012 Information technology. Cryptographic protection of information. Hash function "- the current Russian cryptographic standard that defines the algorithm and the procedure for calculating the hash function. Developed by the Center for the Protection of Information and Special Communications of the Federal Security Service of Russia with the participation of OAO InfoTeKS and put into effect on January 1, 2013 [GOST12].

"Stribog" output gives a hash sum of 256 or 512 bits. Data is processed by 512 bits in 12 rounds. The hash function is based on the iterative construction of Merkl-Damgard using MD-gain. The MD-gain is understood as the addition of an incomplete block when calculating the hash function to the full by adding a vector (0 ... 01) of such length that a complete block is obtained.

Of the additional elements, the following should be noted:

- final transformation, which consists in the fact that the compression function is applied to the checksum of all message blocks modulo 2512;
- when calculating the hash code at each iteration, different compression functions are used. It can be said that the compression function depends on the iteration number [Min14].
- Kuznyechik is a symmetric block encryption algorithm;
- Stribog-512 - a hash algorithm with a hash length of 512 bits.

The Stribog hash function was created taking into account that each transformation inherent in the algorithm opposes a specific cryptanalysis method.

The figure 7 shows a simplified Stribog algorithm. An arbitrary message M to be hashed is input. Next is the initialization of the internal state of the hash function. That is, the starting values of the internal variables are assigned. In the standard, when describing the algorithm, special notation is used.

Some of them are used in the flowchart:

- M - length M (in bits);
- A B - concatenation (connection) of A and B: the sequence of bits B is appended to the right-hand part of the sequence of bits A;

- An - concatenation of the sequence A n times, that is, writing N = 0512 means to write a sequence of 512 zeros into N.

Proceeding from the figure 7, we find that the message of length L will be hashed in K iterations. Where K is calculated by the formula 1:

$$K = \lceil L/512 \rceil \tag{1}$$

The symbols around the fraction in the formula (1) denote rounding up: the smallest integer that is greater than or equal to the number obtained. For each such iteration, modulo 2^{512} addition operations and the compression function g (N, m, h) are performed.

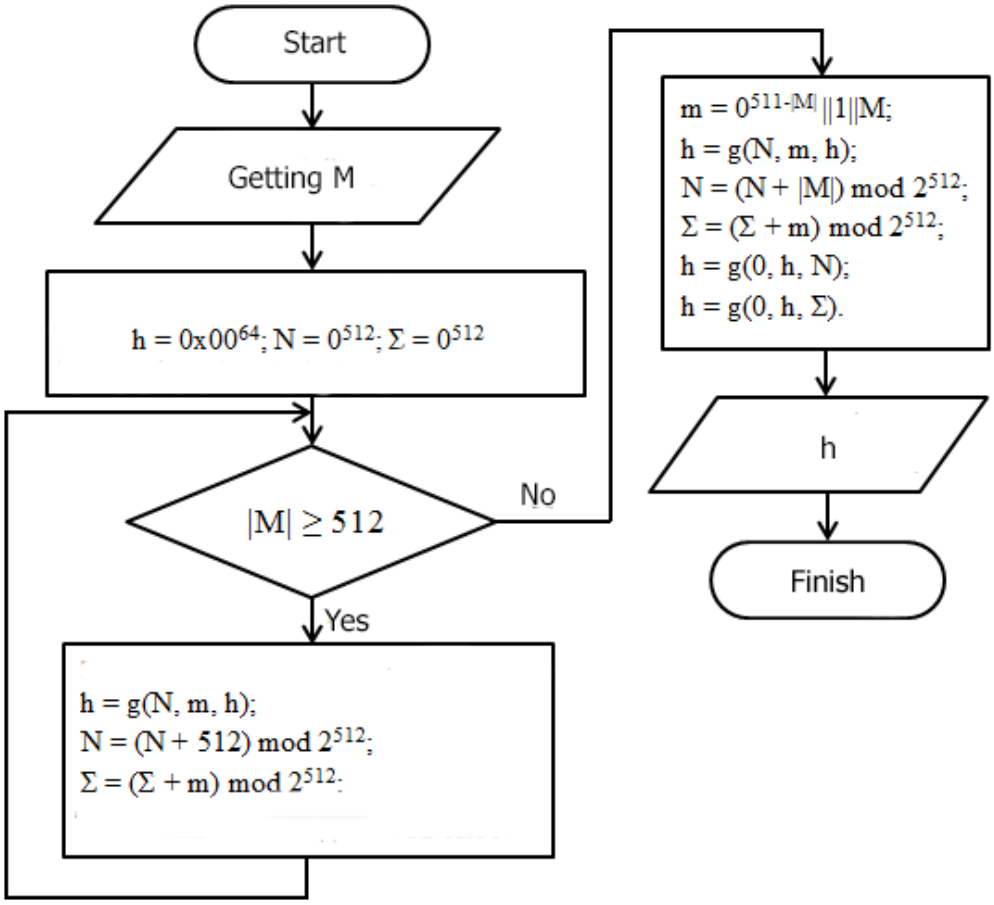

Figure 7: Simplified Stribog Algorithm (512 bits)

This function is the core of the hash function and includes various operations: substitution or substitution transformation, linear transformation and permutation transformation. More information about the compression function can be found in the GOST R 34.11-2012.

2.4 Development

Software development is advisable to produce using a structural style, which involves early thinking about the main issues. This approach to software development is not new - many people have been actively using it since the 1970s[Gag08, Bol15, Bol17]. Unfortunately, this method is not without flaws. One of them is the inability to think over in advance all the issues that will be faced during coding.

In the process of developing programs, quite often one has to make various adjustments to existing algorithms, bases, etc. At best, making such changes stretches the time for preparing a program, and at worst, it can take

the system out of control. Thus, for the subsequent physical implementation of the software, its conceptual development is obvious, which is possible with the use of Constantine structure maps - a powerful modeling tool.

Structural maps of Constantine, designed to describe the relationship between modules, which are the basic building blocks of the software system. All types of modules in any programming language have a number of common properties, the following of which are essential for structural design: a module consists of a set of programming language operators written sequentially, the module has a name by which it can be referred to as a single fragment, the module can accept and / or transmit data as parameters in the calling sequence or link data through fixed cells or common areas.

The Constantin Structural Maps are a model of hierarchy relationships between software modules. The nodes of the structural maps correspond to the modules and data areas, the streams represent intermodule calls. In this case, the cyclic and conditional calls of the modules are modeled by special nodes, so the streams should be depicted as passing through these special nodes. Intermodular communications for data and control are also modeled by special nodes that are tied to flows (that is, to module calls), the arrows indicate the direction of flows and connections[Gag08].

The life cycle of creating a program includes a modeling process. The idea of modeling is a preliminary presentation of the structure of the future software, a description of the interaction of software modules and subsystems. This process should begin with clarifying the structure of the application being developed, during which structural components and their interrelationship are determined. The completion of the stage is the creation of a detailed model of the software being developed, the type of which depends on the approach used. Based on the tasks, a structural model of the server module of the secure data exchange system was designed, figure 8.

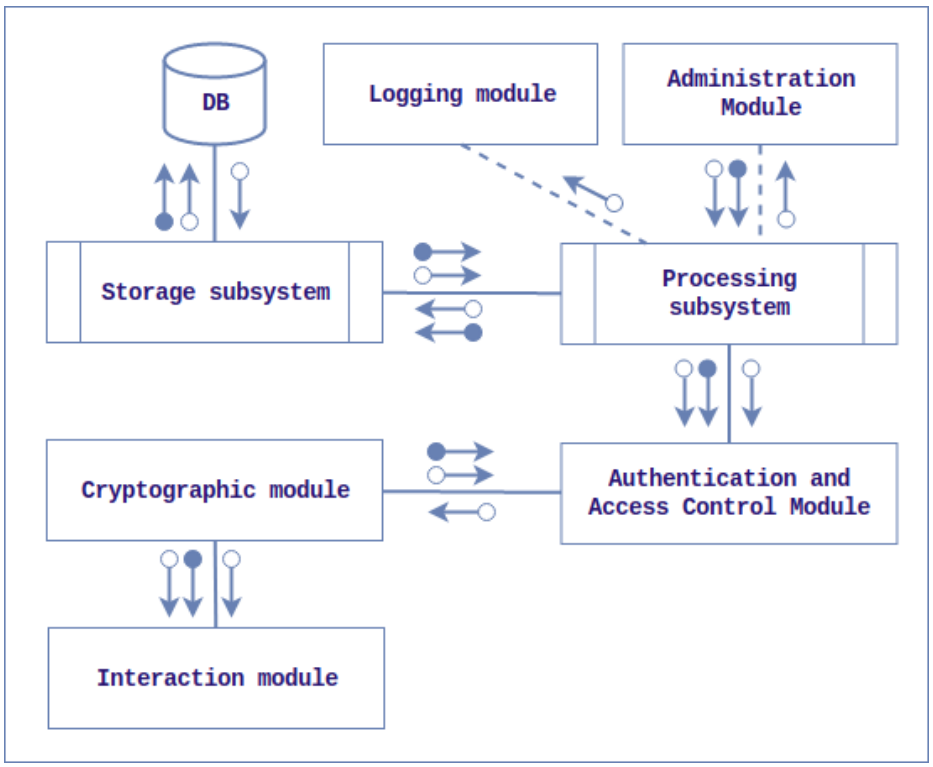

Figure 8: Block diagram of the server module based on Constantine structured maps

The depicted block diagram of the server module reflects the composition and interaction of the subsystems of the software being developed. The information system consists of two main parts; the server and client modules; communication of the system parts is carried out through the subsystem interaction module.

As mentioned above, the PostgreSQL DBMS was selected for storing user data. This DBMS allows you to implement standard relational DBMS functionality, in particular, allows you to create two-dimensional tables for storing data, as well as to declare relationships for these tables [Mir16]. The ER model of the table structure

for storing user data is presented in figure 9.

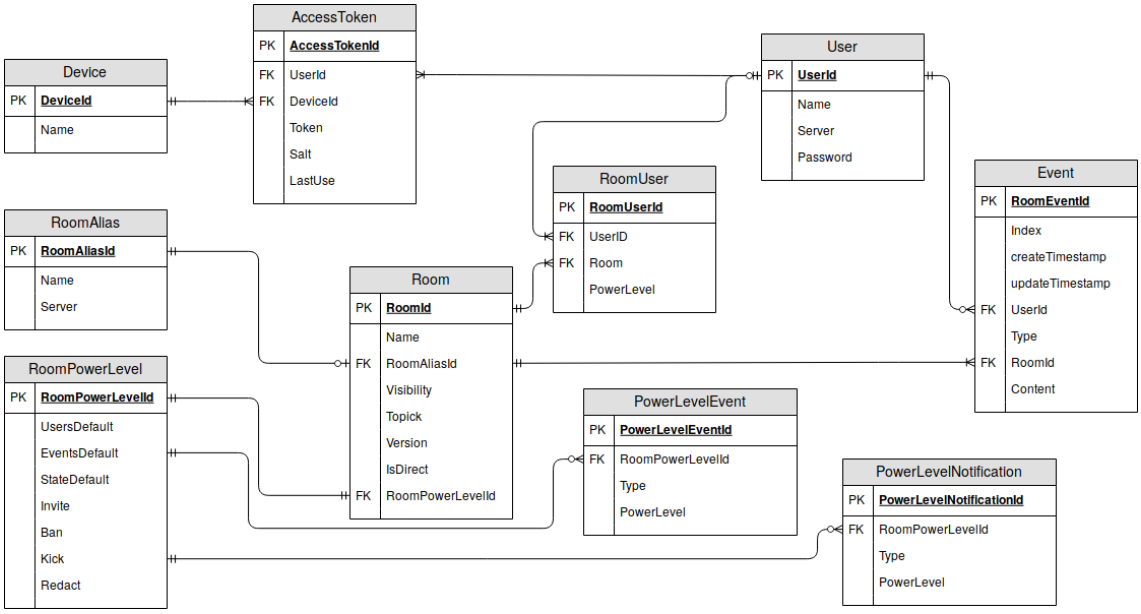

Figure 9: ER-model server database

The central role in the database model is played by three tables that correspond to the three key entities of the Matrix protocol: room, user, event. Consider the structure and purpose of the main tables of this model:

- the Room table corresponds to the Room entity and stores information inherent in each such entity, in particular, the full and short name, visibility for all users, as well as a link to the table containing the rights of users in this room;
- The User table corresponds to the User entity and stores such information as the name, server, and hash value of the user password.
- the Event table corresponds to the Event entity of the Matrix protocol and stores the following information: the sequence number of the event in the given room, the time it was created and updated, the link to the user who sent the event, and the room to which it is attached, as well as and the contents of this event;
- The RoomUser table implements many-to-many links between the Room and User tables, thus representing the user's display in a given room, and stores user-specific information for each room, such as its access level in a given room;
- the AccessToken table stores information about the access token for each user, as well as information related to this token, such as the device to which the token is associated, as well as the time when this token was last used;
- the RoomPowerLevel table stores information about the required power level level for users to perform certain actions.

3 Results

A comparative analysis of the application layer protocols for instant messaging has been carried out and the application layer protocol Matrix has been chosen, since it has the greatest functionality and capabilities to extend the current functionality of the protocol in the final application, and also allows the implementation of a federated server structure.

A cryptographic model has been developed to protect the transmitted JSON objects. The block cipher Kuznyechik (GOST R 34.12-2015) and the hashing function Stribog (GOST R. 34-12) are used for this. A comparison was made between the Kuznyechik algorithm and AES, and the comparative analysis of SHA-512

and Stribog. The analysis has shown that domestic encryption algorithms can be embedded in the software being developed.

A block diagram of the server module has been developed, figure 8. According to the developed structure, the following software modules should include the following modules and subsystems: data storage subsystem, data processing subsystem, authentication and access control module, crypto-protection module, interaction module, logging module, administration module. Considered the most important steps in the implementation of these modules. The result of the task is a software server module that implements client-server requests according to the Matrix protocol, with support for their encryption using the national GOST algorithm.

4 Conclusion

This article proposes the architecture of a system for securely exchanging information over the Internet, and discusses the algorithms and ways of interaction between individual program modules.

The proposed architecture is modular, which simplifies the development, modification and maintenance of software. The parallel execution of the modules allows you to take advantage of the multi-threaded features of modern processors.

As a result of the development, a server module was obtained that allows to receive and send text messages based on the algorithms described above. The functionality of the application can be further expanded through the introduction of support for secure transmission of media information.

References

[Rep17] Data leakage. Russia. 2016 - URL: https://www.infowatch.ru/report_ru2016

[Fed06] Federal Law of 27.07.2006 N 152-FL “On Personal Data”.

[GOST12] GOST R 34.11-2012. ”Information technology. Cryptographic protection of information. Hash function.

[Suk03] Sukharev E.M. System-wide issues of information security. - M .: Radio Engineering, 2003. - 296 p.

[Min14] Minkov S.L. Feasibility study of the project: Methodical manual. - Tomsk: TUSUR, 2014. - 30 p.

[Gag08] Gagarina L.G., Kokorev E.V., Vysnadul B.D. Technology software development. Moscow: The Formum Publishing House - INFRA-M, 2008. - 400 p.

[Bol15] Approach to the effective controlling cloud computing resources in data centers for providing multimedia services Bolodurina, I., Parfenov, D., Shukhman, A. 2015 2015 International Siberian Conference on Control and Communications, SIBCON 2015 Proceedings 7147170

[Bol17] A model of cloud application assignments in software-defined storages. Bolodurina, I.P., Parfenov, D.I., Polezhaev, P.N., Shukhman, A.E. 2017 Journal of Physics: Conference Series 803(1),012024

[Mir16] Situation-oriented databases: Document management on the base of embedded dynamic model Mironov, V., Gusarenko, A., Yusupova, N. 2016 CEUR Workshop Proceedings 1761, p. 238-247

[Yus18] Yusupova, N., Mironov, K. Key information technologies for digital economy // Proceedings of REMS 2018 Russian Federation & Europe Multidisciplinary Symposium on Computer Science and ICT. 2018. Vol-2254. P. 330-334.

[Mat19] Matrix: An open network for secure, decentralized communication. Available at: https://matrix.org/ (accessed 1 August 2019).

[Mtp19] MTProto Mobile Protocol. Available at: https://core.telegram.org/mtproto (accessed 1 August 2019).

[Jam19] Jami official site. Available at: https://ring.cx (accessed 1 August 2019).

[Xmp19] XMPP - The universal messaging standard. Available at: https://xmpp.org/ (accessed 1 August 2019).

[Jin19] An Overview of XMPP. Available at: https://xmpp.org/about/technology-overview.htmljingle (accessed 1 August 2019).