УДК 004.738 DOI: 10.17587/it.25.435-440

Р. Э. Асратян, канд. техн. наук, вед. науч. сотр., e-mail: rea@ipu.ru,
Институт проблем управления им. В. А. Трапезникова РАН

Организация обработки защищенных сообщений в распределенных системах на основе Cryptographic Message Syntax

Рассмотрены методы построения сетевой службы защищенных сообщений, предназначенной для реализации безопасной обработки информационных запросов в распределенных информационных системах. Отличительными особенностями службы являются тесная интеграция функций информационной защиты данных с функциями информационного взаимодействия в сети. Описаны особенности реализации службы на основе средств поддержки стандарта Cryptographic Message Syntax (CMS).

Ключевые слова: *распределенные системы, web-технологии, интернет-технологии, информационное взаимодействие, информационная безопасность*

Введение

С момента своего появления сетевая архитектура.Net и технология web-сервисов заняли ведущее положение в теории и практике создания распределенных информационных систем. Решающую роль здесь сыграл целый ряд признанных достоинств этой технологии: высокая гибкость в определении сервисных функций, высокое быстродействие, эффективная поддержка в целом ряде современных систем программирования, удачные сетевые стандарты WSDL, SOAP и т. п. [1, 2]. Тем не менее, разработчики распределенных систем нередко испытывают трудности с обеспечением защиты данных в сети. Эти трудности чаще всего бывают связаны с отсутствием в архитектуре .Net встроенных средств защиты и аутентификации сетевых сообщений и более всего проявляются в разработках систем, предназначенных для работы в сложных, мультисерверных и мультисетевых средах в условиях высоких требований к информационной безопасности [3—5].

В работе [6] описана новая сетевая служба PMS (Protected Message Service), разработанная в целях преодоления вышеуказанного недостатка. Суть подхода заключается в тесной интеграции функций сетевого информационного обмена с функциями защиты и аутентификации данных. Внешне эта интеграция проявляется в том, что отмеченные функции входят в набор методов главного программного класса службы — класса "Защищенное сообщение", отображающего электронный документ (информационный запрос или ответ), снабженный одной или несколькими удостоверяющими электронными цифровыми подписями (ЭЦП). В отличие, например, от технологии web-сервисов описываемая служба опирается не на модель вызова функций объектов на удаленных серверах, а на модель обмена сообщениями. В данном случае это означает, что все сервисные обрабатывающие функции (методы) имеют одинаковую жесткую спецификацию: они получают объект класса "Защищенное сообщение" в качестве параметра и возвращают объект того же класса. Эти обрабатывающие функции группируются в одну или несколько динамических библиотек, которые подключаются к серверу PMS в момент его запуска (каждая библиотека может рассматриваться как отдаленный аналог web-сервиса в .Net), и становятся доступными для клиентских компонентов.

Реализация PMS на основе криптосистемы "КриптоПро" версии 3.6 и проведенные лабораторные эксперименты показали достаточно высокое быстродействие новой службы, не уступающее, а в отдельных случаях превосходящее быстродействие web-сервисов в одинаковых условиях. Однако при данном подходе возникает жесткая "привязанность" PMS к определенной криптосистеме, что может создать неудобства для разработчиков распределенных систем.

В данной работе рассматривается новый подход к архитектурному построению PMS, основанный на применении стандарта Cryptographic Message Syntax (CMS) и его программной поддержки в среде Windows в качестве базисного средства реализации. Главное преимущество этого подхода заключается в том, что он позволяет PMS "унаследовать" способность гибкой настройки на использование любой криптосистемы, поддерживающей стандарт CMS, и тем самым устранить ту жесткую привязку к определенной криптосистеме, о которой говорилось выше.

Краткие сведения о PMS

Как и всякая сетевая служба, основанная на базовом сетевом протоколе TCP/IP [7], PMS поддержана клиентским и серверным программным обеспечением. Сервер PMS представляет собой постоянно активную программу, обслуживающую запросы на обработку от клиентов (по умолчанию используется порт 8132). Клиентское программное обеспечение представляет собой библиотеку функций PmsBase.dll, реализующих прикладной программный интерфейс (API) к PMS. Этот интерфейс является "лицом" PMS с точки зрения пользователя.

На рис. 1 представлен фрагмент кода в нотации C#, иллюстрирующий простое обращение к обрабатывающей функции с применением средств защиты данных. Две первые строки кода определяют две переменные типа PmsMessage, представляющего собой главный класс PMS ("Защищенное сообщение"). Первой переменной (Request) присваивается значение: объект класса PmsMessage, инициированный символьной строкой (например, содержащей XML-документ информационного запроса). Вторая переменная

```
PmsMessage Request=  new PmsMessage("<request> … </request>");
PmsMessage Reply;
PmsConnection MyConn = new PmsConnection();
PmsCertList  SenderCerts = PmsCertList( new string [] {"Иванов", "Петров"});
Request.AddSignatures(SenderCerts);

MyConn.Connect("MyServer");
PmsCertList  ReceiverCert = MyConn.GetServerCertificate();

Reply=Request.Process (MyConn, "MyLib.MyFunc  param", ReceiverCert);
if(Reply != null)
{
     string Signer;
     for(int i=0; Reply.Signatures.Length; i++)
        if(Reply.VerifySignature(i, out Signer) >= 0)
             Console.WriteLine ("Ответ подписал: " + Signer);
     Console.WriteLine (Reply.GetString());
}
else
     Console.WriteLine ("Ошибка: " +MyConn.ErrMsg);
MyConn.Disconnect();
```

Рис. 1. Пример использования PMS

(Reply) предназначена для хранения результата обработки. В третьей строке определяется и инициируется переменная класса PmsConnection, предназначенного для создания и прекращения сетевого соединения с сервером.

В четвертой и пятой строках кода выполняется формирование подписей в запросе. Сначала определяется переменная SenderCerts класса PmsCertList. Этот класс предназначен для хранения в памяти списков сертификатов с открытыми ключами в стандарте X509 и содержит несколько конструкторов для загрузки сертификатов из файлов или из системных хранилищ с поиском по имени владельца или по серийному номеру. В переменную SenderCerts загружаются два сертификата, соответствующих именам владельцев "Иванов" и "Петров", для последующего формирования двух ЭЦП в запросе. (Такой способ использования означает, что с этими сертификатами обязательно должны быть связаны парные им закрытые ключи, иначе формирование ЭЦП закончится неудачно.) Вызов метода AddSignatures позволяет сформировать две ЭЦП в сообщении Request.

Собственно вызов обрабатывающей функции начинается с вызова метода Connect, устанавливающего сетевое соединение с сервером с указанным сетевым именем или адресом. Далее выполняется первая сетевая операция — запрос сертификата сервера (метод GetServerCertificate) с занесением результата в переменную ReceiverCert уже знакомого нам класса PmsCertList для последующего шифрования информационного запроса. Сразу же после успешного получения сертификата сервера выполняется вызов удаленной функции с помощью применения метода Process к переменной Request с использованием все того же соединения и полученного сертификата. Предполагается, что к серверу PMS подключена библиотека MyLib.dll, содержащая код функции MyFunc. При запуске функции на сервере ей передаются значение переменной Request и опциональный строковый параметр param в качестве фактических параметров. Результат обработки заносится в переменную Reply. Подчеркнем, что шифрование запроса и дешифрование ответа выполняются автоматически в методе Process.

Последующие строки обеспечивают последовательную проверку всех ЭЦП в полученном ответе сервера (вызов метода VerifySignature) и запись в стандартный вывод сведений о подписантах.

Пример заканчивается записью результата обращения в стандартный вывод (предполага-

ется, что результат, как и запрос, имеет форму символьной строки) и закрытием соединения с сервером с помощью метода Disconnect, так как в данном примере оно больше не нужно.

Необходимо отметить, что из фрагмента кода намеренно удалены операторы обработки исключений.

Методы реализации PMS

На рис. 2 проиллюстрированы два архитектурных подхода к реализации PMS:

- на основе прямого подключения определенной криптосистемы к программным модулям PMS использования CMS (архитектура "PMS-Криптосистема"),
- на основе CMS (архитектура "PMS-CMS-Криптосистема") с возможностью использования различных криптосистем.

Первый подход основан на прямом вызове функций определенной криптосистемы из кода программных модулей PMS для выполнения операций формирования ЭЦП, шифрования и т. п. Очевидно, что этот подход обеспечивает наименьше "накладные расходы", но реализация PMS оказывается жестко привязана к выбранной криптосистеме.

Второй подход основан на использовании средств поддержки CMS в качестве промежуточной прослойки между программными модулями PMS и используемой криптосистемой. Главное преимущество — возможность гибкой настройки на применение любой криптосистемы, поддерживающей стандарт CMS без изменения кода программных модулей службы, которые используют универсальный программный интерфейс CMS для выполнения криптофункций.

Как видно из рис. 2, важной составной частью каждого из двух архитектурный решений является используемый сетевой протокол. Независимо от выбранного решения этот протокол организуется по следующим общим принципам:

- PMS в полной мере использует двоичную природу TCP/IP [7,8]. Взаимодействие между клиентом и сервером PMS осуществляется по специальному, достаточно простому протоколу, ориентированному на передачу двоичных сетевых сообщений (PMS-сообщений) в обоих направлениях (никакие преобразования двоичных данных в текстовую форму типа base64 не применяются). Каждое такое сообщение в общем случае содержит два массива байтов: заголовок сообщения и тело сообщения (рис. 3). Первые 4 байта заголовка или тела сообщения содержат целое число — его длину;
- при передаче запроса от клиента к серверу в заголовок сетевого сообщения помещается строка, содержащая полное имя вызываемой функции, а в тело сообщения упаковывается структура PmsMessage в открытой или зашифрованной форме, содержащая информационный запрос. Строка заголовка используется сервером для организации вызова соответствующей обрабатывающей функции;
- при передаче результата обработки от сервера к клиенту в заголовок сетевого сообщения помещается строка диагностического сообщения (значение параметра Msg, сформированное обрабатывающей функцией), а в тело сообщения упаковывается структура PmsMessage, содержащая ответ сервера в открытой или зашифрованной форме, предварительно подписанный собственным закрытым ключом сервера. Никакие двоично-текстовые преобразования (типа base64) не применяются. Полученное от сервера диагностическое сообщение автоматически присваивается члену ErrMsg объекта класса PmsSrvLibraries на стороне клиента (см. рис. 1).

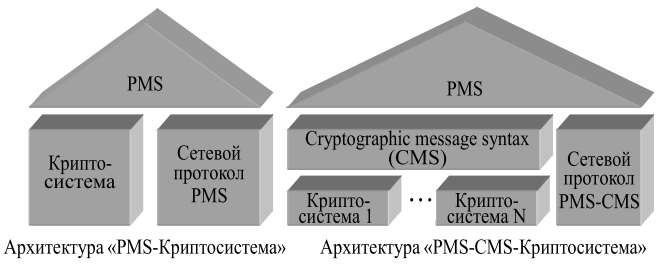

Рис. 2. Архитектурные решения для реализации PMS

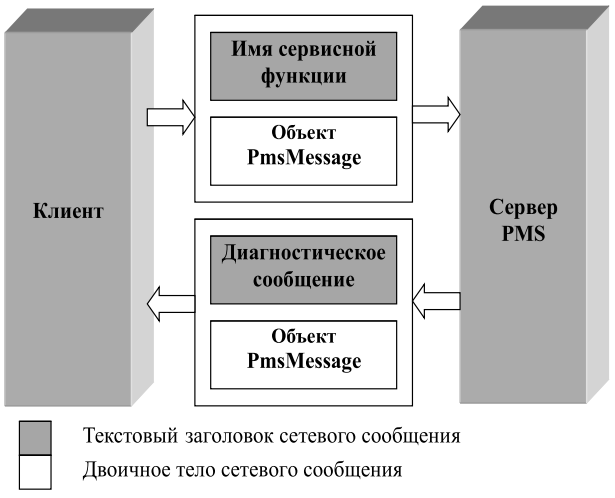

Рис. 3. Сетевой протокол Службы защищенных сообщений

Тем не менее, в архитектуре "PMS-Криптосистема" детали реализации сетевого протокола PMS могут различаться в зависимости от выбранной криптосистемы, так как от последней зависит состав и структура криптоданных, включенных в подписанный и/или зашифрованный объект PmsMessage в теле сетевого сообщения. Важнейшее преимущество архитектуры "PMS-CMS-Криптосистема" заключается в том, что в этом случае реализация сетевого протокола (протокол "PMS-CMS") не зависит от применяемых криптосистем и целиком основывается на универсальном стандарте представления защищенных данных в CMS (см. RFC 5652 в https://tools.ietf.org/html/rfc5652).

Реализация PMS на базе CMS

Данный подход к реализации PMS основан на функциональном сходстве ее главного класса (PmsMessage) с главным классом CMS (SignedCms): оба класса представляют контейнер для хранения произвольных данных, оснащенный необходимыми методами для формирования и проверки электронных подписей. Вместе с тем CMS не содержит классов и методов для удаленной обработки данных в сети (аналогов класса PmsConnection или метода PmsMessage.Process). Фактически описываемый подход можно рассматривать как создание своего рода "надстройки" над CMS, направленной на сетевую обработку данных.

На рис. 4 проиллюстрировано соотношение основных классов и методов PMS и CMS, представляющее собой основу описываемого подхода. Стрелки обозначают прямой вызов одного метода другим. В частности, метод AddSignatures класса PmsMessage (см. рис. 1) выполняет вызов метода ComputeSignature класса SignedCms для формирования каждой ЭЦП в защищенном сообщении, а метод Process опирается в своей работе на методы класса EnvelopedCms для выполнения шифрования отправляемых в сеть данных и дешифрования данных, принятых из сети (методы Encrypt и Decrypt соответственно).

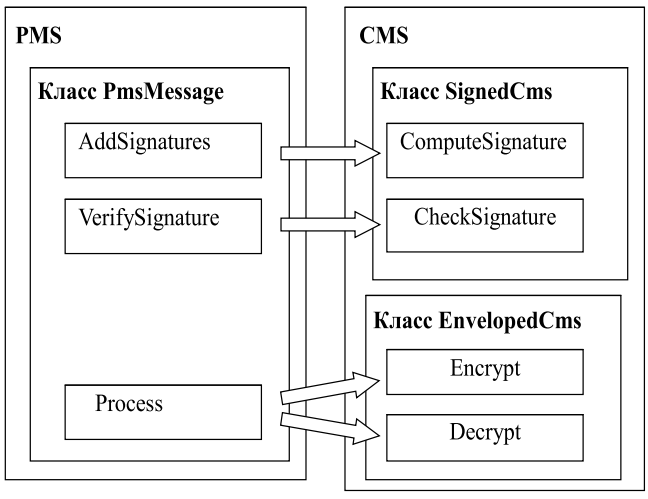

Рис. 4. Основные классы и методы PMS и CMS

Временные оценки

Основная цель экспериментов с PMS заключалась в сравнении быстродействия двух методов ее реализации (на основе архитектур "PMS-Криптосистема" и "PMS-CMS-Криптосистема") между собой, а также с быстродействием web-сервисов в одинаковых условиях. Главное внимание уделялось вызовам сервисных функций с относительно малым (от нескольких миллисекунд до нескольких сотен миллисекунд) временем выполнения (при более длительной обработке разница между двумя технологиями практически нивелируется) с применением средств ЭЦП и шифрования сообщений на основе криптосистемы "КриптоПро" версии 3.6, соответствующей требованиям действующих в России ГОСТ в области криптографической защиты информации. В экспериментах с PMS использовались средства криптозащиты "КриптоПро", интегрированные в клиентскую библиотеку PmsBase.dll и сервер PMS или напрямую или посредством средств CMS. В экспериментах с web-сервисами средства криптосистемы "КриптоПро" подключались непосредственно к программе клиента и программе web-сервиса. И серверы PMS с модельными библиотечными функциями, и Internet Information Server с модельными Web-сервисами были установлены на одном и том же четырехъядерном сервере приложений с тактовой частотой 2,4 ГГц в операционной среде Window 2003 Server, а в качестве клиентской рабочей станции использовался одноядерный компьютер с тактовой частотой 2,8 ГГц.

На рис. 5 показаны характерные результаты экспериментов с очень быстрой сервисной функцией, выполняющей простое перекодирование полученного строчного сообщения в верхний регистр и возврат результата клиенту, при длине сообщения в 2 Кбайт, 50 Кбайт и 100 Кбайт соответственно. На рис. 5 приведена диаграммы времен выполнения операции

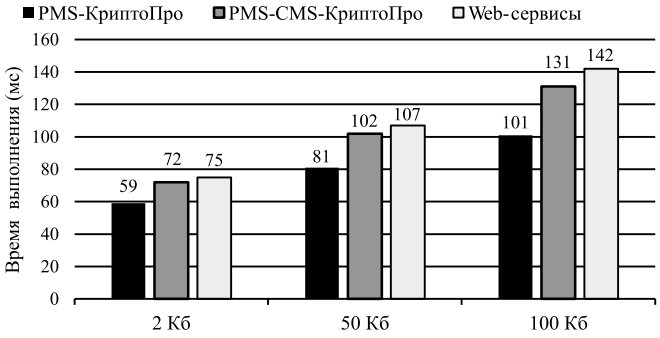

Рис. 5. Оценки быстродействия двух реализаций PMS и Web-сервиса

на сервере в реализации "PMS-КриптоПро" (черный столбик), в реализации "PMS-CMS-КриптоПро" (серый столбик) и с помощью web-сервиса (белый столбик). В каждом режиме время выполнения вычислялось как среднее значение для ста последовательных вызовов сервисной функции.

На рис. 6 и 7 (см. четвертую сторону обложки) приведены результаты экспериментов с относительно медленными сервисными функциями со временем выполнения в несколько сотен миллисекунд. Главное внимание в этой серии экспериментов уделялось сравнению быстродействия двух реализаций PMS и web-сервисов в условиях высокой нагрузки: при одновременной обработке пакетов информационных запросов и при применении криптозащиты. Скорость обработки вычислялась как частное от деления числа запросов в пакете на полное время его выполнения. Характерные кривые, приведенные на рис. 6 и 7, отражают зависимость скорости обработки от числа запросов в пакете для двух реализаций PMS и web-сервисов при обращении к модельным сервисным функциям со временами выполнения 0,5 и 1 с соответственно с применением средств криптозащиты и длине входного и выходного сообщений 50 Кбайт.

Как видно из рис. 6, 7 (см. четвертую сторону обложки), все кривые ведут себя в целом одинаково. При увеличении числа запросов в пакете растет и скорость обработки, что объясняется положительным эффектом от многопоточной обработки запросов и в IIS и в сервере PMS. Например, при двадцати запросах в пакете и времени выполнения сервисной функции 500 мс скорость обработки достигает 15 запросов в секунду у PMS и 14 — у web-сервиса (при последовательной обработке скорость не могла бы превысить значения 2). Однако при дальнейшем увеличении размеров пакета рост скорости обработки замедляется, а потом и вовсе останавливается вследствие достижения предельной производительности. Как видно из графиков, в этой серии экспериментов обе реализации PMS несколько превышают web-сервисы по скорости обработки, но это превышение не является значительным.

В целом результаты экспериментов позволяют сформулировать следующие выводы:

- реализация PMS на основе архитектуры "PMS-CMS-КриптоПро" в целом несколько уступает в быстродействии "прямой" реализации на основе архитектуры "PMS-КриптоПро", но в случае использования относительно медленных сервисных функций (с временем выполнения более 0,5 с) разница становится пренебрежимо малой (рис. 7, см. четвертую сторону обложки);
- обе реализации PMS не уступают в быстродействии web-сервисам;
- применение средств криптозащиты в обеих реализациях PMS не разрушает положительного эффекта от многопоточной обработки запросов;
- как и web-сервисы, обе реализации PMS вполне позволяют поддерживать скорость обработки до нескольких и даже нескольких десятков запросов в секунду даже при использовании средств криптозащиты, что обычно бывает достаточным для большинства информационных систем.

Заключение

Главное преимущество реализации PMS на основе CMS заключается в возможности гибко настраиваться на работу с различными криптосистемами, поддерживающими этот стандарт. Отметим, что настройка осуществляется непосредственно во время выполнения в зависимости от используемых сертификатов. Например, в одном из экспериментов автор использовал на рабочих станциях самоподписанные сертификаты с криптоалгоритмами RSA, а на сервере — сертификат, сформированный удостоверяющим центром "КриптоПро" с криптоалгоритмами, соответствующими российским ГОСТ, и все взаимодействующие стороны прекрасно "понимали друг друга" (разумеется, "КриптоПро" была предустановлена на рабочих станциях и на сервере, а сертификаты обоих типов были снабжены закрытыми ключами для формирования ЭЦП). При этом смена типа используемого сертификата на сетевом узле в процессе эксперимента не создавала никаких проблем.

Вместе с тем следует отметить, что если правовые или корпоративные нормы жестко ориентируют разработчиков на использование определенной криптосистемы, то упомянутая "гибкость" оказывается невостребованной и даже нежелательной, в особенности если речь идет о проекте с высокими требованиями к информационной безопасности. Более того, даже если с технической точки зрения криптосистема удовлетворяет требования проекта (и это подтверждено надлежащими сертификатами соответствия), правомочность ее использования через посредничество средств поддержки CMS остается не вполне ясной. Поэтому в этих условиях простая архитектура "PMS-Криптосистема", основанная на прямых вызовах криптофункций, может все-таки оказаться более предпочтительной.

Список литературы

1. **Шапошников И. В.** Web-сервисы Microsoft.NET. СПб.: БХВ-Петербург, 2002. 336 с.
2. **Мак-Дональд М., Шпушта М.** Microsoft ASP.NET 3.5 с примерами на C# 2008 и Silverlight 2 для профессионалов. М.: Вильямс, 2009. 1408 с.
3. **Згоба А. И., Маркелов Д. В., Смирнов П. И.** Кибербезопасность: угрозы, вызовы, решения // Вопросы кибербезопасности. 2014. № 5. С. 30—38.
4. **Щеглов А. В.** Защита компьютерной информации от несанкционированного доступа. СПб.: Наука и техника, 2004. 384 с.
5. **Козлов А. Д., Орлов В. Л.** Методы и средства обеспечения информационной безопасности распределенных корпоративных систем. М.: ИПУ РАН, 2017. 156 с.
6. **Асратян Р. Э.** Интернет-служба защищенной обработки информационных запросов в распределенных системах // Программная инженерия. 2016. № 11. С. 490—497.
7. **Снейдер Й.** Эффективное программирование TCP/IP. Библиотека программиста. СПб.: Символ-Плюс, 2002. 320 с.
8. **Хант К.** TCP/IP. Сетевое администрирование. СПб.: Питер, 2007. 816 с.

R. E. Asratian, Leading Researcher, rea@ipu.ru,
V. A. Trapeznikov Institute of Control Sciences of Russian Academy of Sciences,
Moscow, 117997, Russian Federation

Protected Message Processing in Distributed Systems on the Basis of Cryptographic Message Syntax

The principles of the implementation of the new network service — Protected Message Service (PMS) — intended for protected queries processing in the distributed information systems are considered. Distinctive feature of PMS is the close integration of authentication and data protection functions with functions of network information exchange and data processing. From the client point of view, the service architecture is based on two main program classes: "Protected message" (PmsMessage) and "Network Connection" (PmsConnection). This classes offer necessary functionality not only for creating and protecting messages, but also for transferring them to remote server via established network connections for processing. The essence of the approach consists in using Cryptographic Message Syntax (CMS) standard as a basis of protected data representation in the network. This approach to the implementation of PMS is based on the functional similarity of its main class (PmsMessage) with the main class CMS (SignedCms): both classes represent a container for storing arbitrary data, equipped with the necessary methods for the formation and verification of electronic signatures. However, CMS does not contain classes and methods for remote data processing in the network (analogs of PmsConnection class or PmsMessag.Process method). Actually, the described approach can be considered as creation of some kind of "superstructure" over CMS directed to network data processing. The experimental implementation of PMS over CMS in C# for Microsoft Framework 4.0 and study of performance of new service were carried out. The results of this study (in comparison with web services in.NET architecture and with "direct" PMS implementation without CMS) are presented.

Keywords: *distributed systems, Web-technologies, Internet-technologies, network interactions, data security*

DOI: 10.17587/it.25.435-440

References

1. **Shaposhnikov I. V.** *Web-servisy Microsoft.NET* (Web-services of Microsoft.NET), SPb, BHV-Peterburg, 2002, 336 p. (in Russian).
2. **MacDonald M., Szpuszta M.** Microsoft *ASP.NET 3.5 s primerami na C# 2008 i Silverlight 2 dlja professionalov* (Pro Microsoft ASP.NET 3.5 in C# 2008 includes Silverlight 2), Moscow, Viljams, 2009, 1408 p. (in Russian).
3. **Zgoba A. I., Markelov D. V., Smirnov P. I.** *Kiberbezopasnost: ugrozy, vyzovy, reshenija* (Cybersafety: threats challenges, decisions), *Voprosy Kiberbezopasnosti*, 2014, no. 5, pp. 30—38 (in Russian).
4. **Shheglov A. V.** *Zashhita komp'juternoj informacii ot nesankcionirovannogo dostupa* (Protection of computer information against illegal access), SPb.: *Nauka i Tehnika*, 2004, 384 p. (in Russian).
5. **Kozlov A. D., Orlov V. L.** *Metody i sredstva obespechenija informacionnoj bezopasnosti raspredelennyh korporativnyh sistem* (Methods and means of ensuring of information security in distributed enterprise systems), Moscow, IPU RAN, 2017, 156 p. (in Russian).
6. **Asratian R. E.**, *Internet-sluzhba zashhishhennoj obrabotki informacionnyh zaprosov v raspredelennyh sistemah* (Internet service for protected information queries processing in distributed systems), *Programmnaja Inzhenerija,* 2016, no. 11, pp. 490—497 (in Russian).
7. Snader J. *Effektivnoe programmirovanie TCP/IP* (Effective TCP/IP programming), SPb, Simvol-Pljus, 2002, 320 p. (in Russian).
8. **Hunt C.** *TCP/IP. Setevoe administrirovanie* (TCP/IP Network administration), SPb, Piter, 2007, 816 p. (in Russian).