1. Место дисциплины в структуре образовательной программы

Дисциплина «Информационная безопасность и защита информации» является дисциплиной по выбору вариативной части.

Рабочая программа составлена в соответствии с требованиями Федерального государственного образовательного стандарта высшего образования по направлению подготовки (специальности) 09.03.02 Информационные системы и технологии, утвержденного приказом Министерства образования и науки Российской Федерации от "12" марта 2015 г. № 219.

Целью освоения дисциплины является получение студентами знаний о теоретических аспектах информационной безопасности и защиты информации, об основных подходах к организации аппаратно-технической и программной защиты информации и разграничения доступа к ней, а также об основных угрозах информационной безопасности.

Задачи:

1. Изучение теоретических основ информационной безопасности и организации защиты информации в компьютерных системах, изучение законодательной и нормативной базы в области информационной безопасности;

2. Понимание принципов организации информационной безопасности, шифрования информации, изучение типовых алгоритмов шифрования;

3. Формирование целостного представления о программно-аппаратном обеспечении защиты информации;

4. Овладение приемами использования моделей, методов и средств для разграничения прав доступа к информации и шифрования данных.

№	Компетенция	Код	Уровень освоения, определяемый этапом формирования компетенции*	Название дисциплины (модуля), сформировавшего данную компетенцию
Входные компетенции				
1	Способность находить организационно-управленческие решения в нестандартных ситуациях и готовность нести за них ответственность	ОК-3	Пороговый, 3 этап	Правовые аспекты разработки и использования информационных систем
2	Знание своих прав и обязанностей как гражданина своей страны, способностью использовать действующее законодательство и другие правовые документы в своей деятельности, демонстрировать готовность и стремление к совершенствованию и развитию общества на принципах гуманизма, свободы и демократии	ОК-9	Пороговый, 2 этап	
3	Способность проводить выбор исходных данных для	ПК-4	Пороговый, 2 этап	Операционные системы

№	Компетенция	Код	Уровень освоения, определяемый этапом формирования компетенции*	Название дисциплины (модуля), сформировавшего данную компетенцию
	проектирования			
4	Способность проводить сбор, анализ научно-технической информации, отечественного и зарубежного опыта по тематике исследования	ПК-22	Пороговый, 2 этап	
5	Способность оценивать надежность и качество функционирования объекта проектирования	ПК-6	Пороговый, 1 этап	Администрирование информационных систем
6	Способность участвовать в работах по доводке и освоению информационных технологий в ходе внедрения и эксплуатации информационных систем	ПК-15	Пороговый, 4 этап	
Исходящие компетенции				
1	Понимание сущности и значения информации в развитии современного информационного общества, соблюдение основных требований к информационной безопасности, в том числе защите государственной тайны	ОПК-4	Базовый, 4 этап	Государственная итоговая аттестация

– *пороговый уровень* дает общее представление о виде деятельности, основных закономерностях функционирования объектов профессиональной деятельности, методов и алгоритмов решения практических задач;

– *базовый уровень* позволяет решать типовые задачи, принимать профессиональные и управленческие решения по известным алгоритмам, правилам и методикам;

– *повышенный уровень* предполагает готовность решать практические задачи повышенной сложности, нетиповые задачи, принимать профессиональные и управленческие решения в условиях неполной определенности, при недостаточном документальном, нормативном и методическом обеспечении.

2. Перечень результатов обучения

Процесс изучения дисциплины направлен на формирование элементов следующих компетенций.

Планируемые результаты обучения по дисциплине приведены в таблице ниже.

№	Формируемые компетенции	Код	Знать	Уметь	Владеть
1	Понимание сущности и значения информации в развитии современного информационного общества, соблюдение основных требований к информационной безопасности, в том числе защите государственной тайны.	ОПК-4	– Основные положения информационной безопасности информационных систем; – Программу информационной безопасности России; – Требования к информационной безопасности и защите информации в информационных системах.	– Анализировать состав средств, обеспечивающих информационную безопасность и защиту информации в информационных системах; – Осуществлять выбор вида аппаратного и программного обеспечения для решения задач информационной безопасности информационных систем.	– Компьютерными средствами реализации защиты в информационных системах.
2	Способность использовать современные компьютерные технологии поиска информации для решения поставленной задачи, критического анализа этой информации и обоснования принятых идей и подходов к решению.	ОПК-5	– Основные проблемы и направления развития аппаратных и программных средств защиты информации; – Основные принципы организации и алгоритмы функционирования систем безопасности в современных операционных системах и оболочках.	– Адаптировать известные методы информационной безопасности и защиты информации для конкретных информационных систем.	– Базовыми методами защиты информации в информационных системах.
3	способностью использовать технологии разработки объектов профессиональной деятельности в областях: машиностроение,	ПК-17	- Как использовать технологии разработки		

№	Формируемые компетенции	Код	Знать	Уметь	Владеть
	приборостроение, техника, образование, медицина, административное управление, юриспруденция, бизнес, предпринимательство, коммерция, менеджмент, банковские системы, безопасность информационных систем, управление технологическими процессами, механика, техническая физика, энергетика, ядерная энергетика, силовая электроника, металлургия, строительство, транспорт, железнодорожный транспорт, связь, телекоммуникации, управление инфокоммуникациями, почтовая связь, химическая промышленность, сельское хозяйство, текстильная и легкая промышленность, пищевая промышленность, медицинские и биотехнологии, горное дело, обеспечение безопасности подземных предприятий и производств, геология, нефтегазовая отрасль, геодезия и картография, геоинформационные системы, лесной комплекс, химико-лесной комплекс, экология, сфера сервиса, системы массовой информации, дизайн, медиаиндустрия, а также предприятия различного профиля и все виды деятельности в условиях экономики информационного общества		объектов профессиональной деятельности в различных областях		

3. Содержание и структура дисциплины (модуля)

Общая трудоемкость дисциплины составляет 3 зачетные единицы (108 часов). Дисциплина преподается в 8 семестре.

Трудоемкость дисциплины по видам работ

Вид работы	**Трудоемкость, час.**
Лекции (Л)	14
Практические занятия (ПЗ)	–
Лабораторные работы (ЛР)	24
КСР	3
Курсовая проект работа (КР)	–
Расчетно-графическая работа (РГР)	–
Самостоятельная работа (проработка и повторение лекционного материала и материала учебников и учебных пособий, подготовка к лабораторным и практическим занятиям, коллоквиумам, рубежному контролю и т.д.)	58
Подготовка и сдача экзамена	–
Подготовка и сдача зачета	9
Вид итогового контроля (зачет, экзамен)	Зачет

Содержание разделов и формы текущего контроля

№	Наименование и содержание раздела	Количество часов						Рекомендуемая литература*	Виды интерактивных образовательных технологий**
		Аудиторная работа				СРС	Всего		
		Л	ПЗ	ЛР	КСР				
1	**Основные понятия и положения защиты информации (ЗИ) в информационно-вычислительных системах.** Предмет и объект ЗИ. Угрозы безопасности информации. Понятие угрозы безопасности. Классификация угроз информационной безопасности. Классификация злоумышленников. Основные методы реализации угроз информационной безопасности. Причины, виды и каналы утечки информации.	4		8	1	20	33	Р. 6.1 - №1, 2 Р. 6.2 - №1, 2	лекция классическая, лекция-визуализация
2	**Методы и средства защиты информации в информационно-вычислительных системах.** Правовые и организационные методы защиты информации. Стандарты и спецификации в области информационной безопасности. Административный уровень информационной безопасности. Криптографическая защита информации.	6		8	1	18	33	Р. 6.1 - №1, 2 Р. 6.2 - №1, 2 Р. 6.3 - №1, 2, 3	лекция классическая, лекция-визуализация

№	Наименование и содержание раздела	Количество часов						Рекомендуемая литература*	Виды интерактивных образовательных технологий**
		Аудиторная работа				СРС	Всего		
		Л	ПЗ	ЛР	КСР				
3	**Защита компьютерной информации в информационно-вычислительных сетях.** Модели безопасности. Системы защиты программного обеспечения. Защита информации в корпоративных сетях. Защита от информационных инфекций.	4		8	1	20	33	Р. 6.1 - №1, 2 Р. 6.2 - №1, 2 Р. 6.3 - №1, 2, 3	лекция классическая, лекция-визуализация

Занятия, проводимые в интерактивной форме, составляют 50 % от общего количества аудиторных часов по дисциплине «Информационная безопасность и защита информации».

Лабораторные работы

№	Наименование раздела	Наименование лабораторной работы	Кол-во часов
1	Основные понятия и положения защиты информации в информационно-вычислительных системах	Шифрование и расшифровывание XML-элементов с помощью симметричного ключа	4
		Шифрование и расшифровывание XML-элементов с помощью асимметричного ключа.	4
2	Методы и средства защиты информации в информационно-вычислительных системах	Подписывание XML-документов с помощью цифровых подписей и проверка цифровых подписей XML-документов	4
		Создание и проверка хэша.	4
3	Защита компьютерной информации в информационно-вычислительных сетях.	Ролевая модель разграничения прав доступа в реляционных БД.	4
		Использование сертификатов безопасности.	4

4. Учебно-методическое обеспечение самостоятельной работы студентов

№	Наименование раздела	Вопросы для самостоятельного изучения
1	Основные понятия и положения защиты информации в информационно-вычислительных системах	1) ГОСТ Р 52069.0-2003 Защита информации. Система стандартов. Основные положения. 2) ГОСТ Р 50922-2006 Защита информации. Основные термины и определения. 3) ГОСТ Р ИСО/МЭК 18045-2008 Методология оценки безопасности информационных технологий.
2	Методы и средства защиты информации в информационно-вычислительных системах	1) ГОСТ Р ИСО/МЭК ТО 15446-2008 Руководство по разработке профилей защиты и заданий по безопасности. 2) ГОСТ Р ИСО/МЭК 15408-1-2008 Информационная технология. Методы и средства обеспечения безопасности. Критерии оценки безопасности информационных технологий. Часть 1. Введение и общая модель. 3) ГОСТ Р ИСО/МЭК 15408-2-2008 - Информационная технология. Методы и средства обеспечения безопасности. Критерии оценки безопасности информационных технологий. Часть 2. Функциональные требования безопасности. 4) ГОСТ Р ИСО/МЭК 15408-3-2008 Информационная технология. Методы и средства обеспечения безопасности. Критерии оценки безопасности информационных технологий. Часть 3. Требования доверия к безопасности. 5) ГОСТ Р 51583-2014 Защита информации. Порядок создания автоматизированных систем в защищенном исполнении. Общие положения. 6) ГОСТ Р 56939-2016 Защита информации.

№	Наименование раздела	Вопросы для самостоятельного изучения
		Разработка безопасного программного обеспечения. Общие требования.
3	Защита компьютерной информации в информационно-вычислительных сетях.	1) ГОСТ Р 53109-2008 Система обеспечения информационной безопасности сети связи общего пользования. Паспорт организации связи по информационной безопасности. 2) ГОСТ Р 53114-2008 Защита информации. Обеспечение информационной безопасности в организации. Основные термины и определения. 3) ГОСТ Р ИСО/МЭК 27033-1-2011 Информационная технология. Методы и средства обеспечения безопасности. Безопасность сетей. Часть 1. Обзор и концепции. 4) ГОСТ Р ИСО/МЭК 27033-3-2014 «Безопасность сетей. Часть 3. Эталонные сетевые сценарии. Угрозы, методы проектирования и вопросы управления».

Для концентрации внимания студентов именно на этих темах им предлагаются домашние задания в виде самостоятельного поиска информации в открытых источниках (например, в сети Интернет).

5. Фонд оценочных средств

Оценка уровня освоения дисциплины осуществляется в виде текущего и промежуточного контроля успеваемости бакалавров, и на основе критериев оценки уровня освоения дисциплины.

Контроль представляет собой набор заданий и проводится в форме контрольных мероприятий по оцениванию фактических результатов обучения студентов и осуществляется ведущим преподавателем.

Объектами оценивания выступают:

– учебная дисциплина (активность на занятиях, своевременность выполнения различных видов заданий, посещаемость всех видов занятий по аттестуемой дисциплине и пр.);

– степень усвоения теоретических знаний;

– уровень овладения практическими умениями и навыками по всем видам учебной работы;

– результаты самостоятельной работы.

Активность обучающегося на занятиях оценивается на основе выполненных работ и заданий, предусмотренных ФОС дисциплины.

Оценивание проводится преподавателем независимо от наличия или отсутствия обучающегося (по уважительной или неуважительной причине) на занятии. Оценка носит комплексный характер и учитывает достижения обучающегося по основным компонентам учебного процесса за текущий период.

№ п/п	Контролируемые разделы (темы) дисциплины	Код контролируемой компетенции (или ее части)	Уровень освоения, определяемый этапом формирования компетенции	Наименование оценочного средства*
1	Основные понятия и положения защиты информации (ЗИ) в информационно-вычислительных системах.	ОПК-4, ОПК-5	Пороговый, 3 этап Пороговый, 4 этап	Контрольные вопросы, защита ЛР
2	Методы и средства защиты информации в информационно-вычислительных системах.	ОПК-5	Пороговый, 4 этап	Контрольные вопросы, защита ЛР
3	Защита компьютерной информации в информационно-вычислительных сетях.	ОПК-5	Пороговый, 4 этап	Контрольные вопросы, защита ЛР
4	Основные понятия и положения защиты информации (ЗИ) в информационно-вычислительных системах.	ПК-17	Пороговый	Контрольные вопросы

При реализации дисциплины используется балльно-рейтинговая оценка освоения компетенций.

Виды учебной деятельности	Балл за конкретное задание	Число заданий	Баллы	
			Мин.	Макс.
Раздел 1. Основные понятия и положения защиты информации (ЗИ) в информационно-вычислительных системах.				
Текущий контроль			0	4
1. Аудиторная работа	Письменный ответ на вопрос		0	1
	Защита лабораторной работы №1	1	0	5
	Защита лабораторной работы №2	1	0	5
2. Оценка СРС	Конспектирование тем, отведенных для самостоятельной работы студента	3	0	2 (6)
Раздел 2. Методы и средства защиты информации в информационно-вычислительных системах.				
Текущий контроль			0	19
1. Аудиторная работа	Письменный ответ на вопрос		0	1
	Защита лабораторной работы №3	1	0	5
	Защита лабораторной работы №4	1	0	5
2. Оценка СРС	Конспектирование тем, отведенных для самостоятельной работы студента	6	0	2 (12)

Раздел 3. Защита компьютерной информации в информационно-вычислительных сетях.				
Текущий контроль			0	4
1. Аудиторная работа	Письменный ответ на вопрос		0	1
	Защита лабораторной работы №5	1	0	5
	Защита лабораторной работы №6	1	0	5
2. Оценка СРС	Конспектирование тем, отведенных для самостоятельной работы студента	4	0	2 (8)
Поощрительные баллы*				
Посещаемость			**0**	**5**
Итоговый контроль**				
Зачет	«зачтено»		75	100
	«не зачтено»		0	74

Вопросы для контроля

1) Предмет и объект защиты информации. Свойства информации с точки зрения ее защиты.
2) Классификация угроз информационной безопасности.
3) Основные методы реализации угроз информационной безопасности.
4) Правовое регулирование и законодательная база РФ в области безопасности информации.
5) Организационные методы защиты информации.
6) Характеристика основных стандартов и рекомендаций в области инф.безопасности. Оранжевая книга и «Концепция защиты от НСД»
7) Основные понятия и определения криптографии и криптоанализа.
8) Требования к крипотографическим системам.
9) Понятие хеш-функции и ее использование в крипотографии.
10) Функция хеширования SHA.
11) Описание и характеристики хеширования по ГОСТ Р 34.11-2012 «Стрибог».
12) Симметричные криптосистемы. Основные черты и характеристики. Их классификация.
13) Блочные шифры и их генерация.
14) Алгоритм DES и его схема шифрования.
15) Стандарт шифрования ГОСТ Р 34.12-2015 (Кузнечик).
16) Режимы применения блочных шифров.
17) Потоковые шифры.
18) Шифрование с помощью ГПСП и гаммирование.
19) Ассиметричные криптосистемы. Основные черты и характеристики. Их классификация.
20) Алгоритм шифрования в криптосистеме Эль-Гамаля.
21) Крипотосистема RSA. Ее достоинства. Оценка стойкости RSA.
22) Понятие ЭЦП. Алгоритмы выработки и проверки ЭЦП.
23) Стандарт формирования и проверки ЭЦП ГОСТ Р 34.10-2012.
24) Управление крипотографическими ключами. Протоколы обмена и использование сертификатов.
25) Понятие угрозы в ВС. Типы угроз.
26) Понятие политики безопасности. Формальные и неформальные политики безопасности.
27) Формальные методы анализа систем.
28) Понятие модели безопасности. Основные понятия.

29) Функционирование монитора пересылок.
30) Использование логики предикатов при описании моделей безопасности.
31) Понятие субъекта, объекта, доступа.
32) Уровни безопасности. Доверие и секретность.
33) Компоненты моделей безопасности.
34) Модели разграничения доступа. Основные их понятия и черты.
35) Модели дискреционного доступа. Основные их понятия и черты.
36) Вероятностные модели. Их характеристики.
37) Модели мандатного доступа. Основные их понятия и черты.
38) Алгоритмическое описание процесса организации доступа в модели Хартстона.
39) Модель Харрисона-Руззо-Ульмана. Определения.
40) Теоремы модели Харрисона-Руззо-Ульмана.
41) Классическая модель Белла-Лападула.
42) Правила в модели Белла-Лападула.
43) Формулировка и доказательство основной теоремы безопасности для модели Белла-Лападула.
44) Система Z. Ее описание, свойства.
45) Правила спокойствия в модели Белла-Лападула.
46) Формальная модель MMS. Основные определения.
47) Описание состояния системы для формальной модели MMS.
48) Основная теорема безопасности для формальной модели MMS.
49) Предположения безопасности и ограничения безопасности в модели MMS.
50) Модель системы безопасности с полным перекрытием.
51) Модели контроля целостности.
52) Мандатная модель целостности Биба.
53) Правила в модели Биба.
54) Модель Биба с пониженным уровнем субъекта.
55) Модель Биба с пониженным уровнем объекта.
56) Объединенная модель (модель Биба + модель Белла-Лападула).
57) Модель Кларка-Вилсона.
58) Правила модели Кларка-Вилсона
59) Технологии разработки объектов профессиональной деятельности в различных областях

Критерии оценки контрольных вопросов:

- оценка «зачтено» или «1» балл выставляется бакалавру, при ответе на любой вопрос, соответствующей темы изучения. Ответ должен быть корректным.
- оценка «не зачтено» или «0» баллов выставляется бакалавру, при некорректном ответе на вопрос, допущении существенных неточностей.

5.1 Типовые оценочные материалы

Оценочные материалы для лабораторных работ

I. Раздел: Основные понятия и положения защиты информации в информационно-вычислительных системах.

Задание выполняется на лабораторной работе №1. Шифрование и расшифровывание XML-элементов с помощью симметричного ключа.

Пороговый уровень

Практическая часть

Составить программу, шифрующую и расшифровывающую XML-элемент с использованием одного из следующих классов, реализующих шифрование с закрытым ключом:

- *AesManaged;*
- *DESCryptoServiceProvider;*
- *RC2CryptoServiceProvider;*
- *RijndaelManaged;*
- *TripleDESCryptoServiceProvider.*

Программа, должна содержать класс, реализующий метод для шифрования и метод для расшифровывания XML-элемента

Примерные теоретические вопросы для защиты лабораторной работы:

1. В чем состоит особенность шифрования с помощью симметричного ключа?
2. Какие требования предъявляются к шифрованию с помощью симметричного ключа?
3. Каким образом осуществляется обмен ключами при шифровании с помощью симметричного ключа?
4. Перечислите и кратко охарактеризуйте основные алгоритмы симметричного шифрования.
5. Какие классы используются для шифрования в составе пространства имен *System.Security.Cryptography* и *System.Security.Cryptography.Xml*?

Задание выполняется на лабораторной работе №2. Шифрование и расшифровывание XML-элементов с помощью асимметричного ключа.

Пороговый уровень

Практическая часть

1. В чем состоит особенность шифрования с помощью асимметричного ключа?
2. Какие требования предъявляются к шифрованию с помощью асимметричного ключа?
3. Каким образом осуществляеться обмен ключами при шифровании с помощью асимметричного ключа?
4. Перечислите и кратко охарактеризуйте основные алгоритмы асимметричного шифрования.
5. Можно ли внедрять ключи непосредственно в исходный код программы?
6. Каким образом можно удалить криптографический ключ из памяти после завершения его использования?

Примерные теоретические вопросы для защиты лабораторной работы:

1. В чем состоит особенность шифрования с помощью асимметричного ключа?
2. Какие требования предъявляются к шифрованию с помощью асимметричного ключа?
3. Каким образом осуществляеться обмен ключами при шифровании с помощью асимметричного ключа?
4. Перечислите и кратко охарактеризуйте основные алгоритмы асимметричного шифрования.
5. Можно ли внедрять ключи непосредственно в исходный код программы?
6. Каким образом можно удалить криптографический ключ из памяти после завершения его использования?

II. Раздел: Методы и средства защиты информации в информационно-вычислительных системах.

Задание выполняется на лабораторной работе №3. Подписывание XML-документов с помощью цифровых подписей и проверка цифровых подписей XML-документов.

Пороговый уровень:

Практическая часть

Необходимо разработать программу, в которой создается пара открытый ключ – закрытый ключ, затем документ подписывается закрытым ключом и осуществляется проверка подписи с помощью открытого ключа.

Для создания цифровой подписи используйте один из следующих классов:

– *DSACryptoServiceProvider;*
– *ECDsa;*
– *ECDsaCng.*

Примерные теоретические вопросы для защиты лабораторной работы:

1. Для чего необходима ЭЦП и что удостоверяет цифровая подпись документа?
2. В чем состоит особенность использования асимметричных криптосистем в электронной цифровой подписи?
3. Какие криптоалгоритмы применяются в ЭЦП?
4. Какие классы *Microsoft .NET Framework* наиболее предпочтительны для цифровой подписи? Почему?
5. Укажите наиболее уязвимые места с точки зрения информационной безопасности при использовании ЭЦП.

Задание выполняется на лабораторной работе №4. Создание и проверка хэша.

Пороговый уровень

Практическая часть

Необходимо разработать программу, хэширующую jpeg файл. Расположение файла определяется пользователем на локальном компьютере. Значение хэша должны сохраняться в текстовый файл по указанному пользователем расположению. Программа должна сопоставлять предъявленный файл и значение хэша из текстового файла.

Для создания хэша используйте один из следующих классов:

– *MD5CryptoServiceProvider;*
– *SHA1Managed;*
– *SHA256Managed;*
– *SHA384Managed;*
– *SHA512Managed.*

Примерные теоретические вопросы для защиты лабораторной работы:

1. Для чего используется хэширование?
2. Приведите примеры использования хэширования на практике?
3. Опишите процесс хэширования по алгоритму MD5.
4. Что такое коллизии хэш-функции? С чем они связаны?
5. Оцените затраты времени при хешировании и последующей проверке файлов размером ≈ 1 КБ, ≈ 10 КБ, ≈ 100 КБ, ≈ 1 МБ, ≈ 10 МБ, ≈ 100 МБ.

III. Раздел (тема) дисциплины: Защита компьютерной информации в информационно-вычислительных сетях.

Задание выполняется на лабораторной работе №5. Ролевая модель разграничения прав доступа в реляционных БД.

Пороговый уровень

Практическая часть

Необходимо настроить ролевое разграничение доступа для реляционной базы данных и разработать программу, обращающуюся к этой базе данных с авторизацией пользователя.

Примерные теоретические вопросы для защиты лабораторной работы:

1. В чем отличие Windows-аутентификации от аутентификации на уровне базы данных?
2. Перечислите встроенные роли на уровне сервера СУБД Microsoft SQL Server?
3. Перечислите встроенные роли на уровне базы данных в СУБД Microsoft SQL Server?
4. Какие разрешения можно назначить создаваемой роли на уровне БД?

Задание выполняется на лабораторной работе №6. Использование сертификатов безопасности.

Пороговый уровень

Практическая часть

Необходимо ранее разработанную программу подписать сертификатом разработчика (Code Signing). Для этого следует получить Code Signing-сертификат. Затем, после подписывания, следует проверить как инсталлируется программа в среде Windows.

Примерные теоретические вопросы для защиты лабораторной работы:

1. Для чего используется Code Signing?
2. Как осуществляется валидация сертификата Code Signing?
3. Что гарантирует сертификат Code Signing?

Критерии оценки лабораторных работ:

– 5 баллов выставляется студенту, если он полностью выполнил лабораторную работу, ответил на контрольные вопросы, подготовил отчет, полностью удовлетворяющий требованиям, ответил на дополнительные вопросы преподавателя;

– 4 балла выставляется студенту, если он полностью выполнил лабораторную работу, частично ответил на контрольные вопросы, подготовил отчет, полностью удовлетворяющий требованиям, частично ответил на дополнительные вопросы преподавателя;

– 3 балла выставляется студенту, если он частично выполнил лабораторную работу и/или частично ответил на контрольные вопросы, подготовил отчет, частично удовлетворяющий требованиям и не ответил ни на один из дополнительных вопросов преподавателя;

– 2 балла выставляется студенту, если он частично выполнил лабораторную работу, не ответил ни на один из контрольных вопросов, подготовил отчет, частично удовлетворяющий требованиям, не ответил ни на один из дополнительных вопросов преподавателя;

– 1 балл выставляется студенту, если он приступил к выполнению лабораторной работы, но не завершил ее выполнение, не ответил ни на один из контрольных вопросов, не подготовил отчет, не ответил ни на один из дополнительных вопросов преподавателя;

– 0 баллов выставляется студенту, не приступившему к выполнению лабораторной работы.

Требования, предъявляемые к отчету:

– Четкое формулирование поставленной цели лабораторной работы и задач, решение которых необходимо для достижения поставленной цели.

– Описание в виде пунктов, тех действий, которые требуются для решения поставленных задач. Все рисунки и таблицы последовательно нумеруются и описываются.

5.2 Методические материалы, определяющие процедуры оценивания результатов обучения (знаний, умений, владений), характеризующих этапы формирования компетенций

Приводится методика проведения процедур оценивания конкретных результатов обучения (знаний, умений, владений) формируемого этапа компетенции. То есть для каждого образовательного результата определяются показатели и критерии сформированности компетенций на различных этапах их формирования, приводятся шкалы и процедуры оценивания.

Компетенция, ее этап и уровень формирования	Заявленный образовательный результат	Типовое задание из ФОС, позволяющее проверить сформированность образовательного результата	Процедура оценивания образовательного результата	Критерии оценки
ОПК-4 (пороговый, 3 этап)	**Знать:** – Основные положения информационной безопасности информационных систем; – Программу информационной безопасности России; – Требования к информационной безопасности и защите информации в информационных системах.	Ответы на вопросы для контроля (см. выше)	В конце изучения каждой темы на лекции студенту дается 1 вопрос на выбор. Либо бакалавр отвечает на теоретические вопросы при сдаче зачета.	Критерий оценки (см. выше)
	Уметь – Анализировать состав средств, обеспечивающих информационную безопасность и защиту информации в информационных системах; – Осуществлять выбор вида аппаратного и программного обеспечения для решения задач информационной безопасности информационных систем.	Защита лабораторной работы №1, №2, №3, №4.	Лабораторная работа проводится в соответствии с расписанием проведения занятий. Отчет по лабораторной работе студенты защищают в конце/начале занятия или на специально выделенных консультациях, время защиты – 5 минут. Методические указания по выполнению лабораторной работе приведены в разделе 6.4.	
	Владеть – Компьютерными средствами реализации защиты в информационных системах.			
ОПК-5 (пороговый, 4 этап)	**Знать** – Основные проблемы и направления развития аппаратных и программных средств защиты	Ответы на вопросы для контроля (см. выше)	В конце изучения каждой темы на лекции студенту дается 1 вопрос на выбор. Либо бакалавр отвечает на теоретические вопросы при сдаче зачета.	Критерий оценки (см. выше)

Компетенция, ее этап и уровень формирования	Заявленный образовательный результат	Типовое задание из ФОС, позволяющее проверить сформированность образовательного результата	Процедура оценивания образовательного результата	Критерии оценки
	информации; – Основные принципы организации и алгоритмы функционирования систем безопасности в современных операционных системах и оболочках.			
	Уметь – Адаптировать известные методы информационной безопасности и защиты информации для конкретных информационных систем.	Защита лабораторной работы №5, №6	Лабораторная работа проводится в соответствии с расписанием проведения занятий. Отчет по лабораторной работе студенты защищают в конце/начале занятия или на специально выделенных консультациях, время защиты – 5 минут. Методические указания по выполнению лабораторной работе приведены в разделе 6.4.	
	Владеть – Базовыми методами защиты информации в информационных системах.	Защита лабораторных работ № 5,6		
ПК-17 (пороговый)	**Знать** - Как использовать технологии разработки объектов профессиональной деятельности в различных областях.	Ответы на вопросы для контроля (см. выше)	В конце изучения каждой темы на лекции студенту дается 1 вопрос на выбор. Либо бакалавр отвечает на теоретические вопросы при сдаче зачета.	Критерий оценки (см. выше)

6. Учебно-методическое и информационное обеспечение дисциплины (модуля)

6.1 Основная литература

1. Шаньгин, В.Ф. Информационная безопасность. [Электронный ресурс] – Электрон. дан. – М.: ДМК Пресс, 2014. – 702 с. – Режим доступа: http://e.lanbook.com/book/50578.

2. Мельников, Д.А. Информационная безопасность открытых систем. [Электронный ресурс] – Электрон. дан. – М.: ФЛИНТА, 2014. – 448 с. – Режим доступа: http://e.lanbook.com/book/48368.

6.2 Дополнительная литература

1. Баранова, Е.К. Информационная безопасность и защита информации: учебное пособие / Е.К. Баранова, А.В. Бабаш. – 2-е изд. – М.: РИОР: Инфра-М, 2014. – 256 с.

2. Шаньгин, В.Ф. Защита компьютерной информации. [Электронный ресурс] – Электрон. дан. – М.: ДМК Пресс, 2010. – 544 с. – Режим доступа: http://e.lanbook.com/book/1122.

6.3 Интернет-ресурсы (электронные учебно-методические издания, лицензионное программное обеспечение)

1. Курс лекций «Защита от несанкционированного доступа». Материал из викиучебника. – https://ru.wikibooks.org/wiki/Курс_лекций_Защита_Информации
2. Библиотека материалов по криптографии. – https://www.pgpru.com/biblioteka.
3. Электронные ресурсы на хабе «Информационная безопасность» портала Хабрхабр – https://habrahabr.ru/hub/infosecurity/.

6.4 Методические указания к лабораторным занятиям

Лабораторная работа № 1. Шифрование и расшифровывание XML-элементов с помощью симметричного ключа

1. Цель и задачи работы

Целью работы является изучение студентами стандартных классов фреймворка *Microsoft .NET Framework* для симметричного шифрования.

Задачами работы являются:

- ознакомление с примером использования класса RijndaelManaged, реализующим алгоритм шифрования с закрытым ключом;
- разработка программы, реализующей симметричное шифрование XML-элемента и его расшифровывание по известному ключу.

2. Теоретические сведения

При шифровании с закрытым ключом для шифровки и дешифровки данных используется один закрытый ключ. Шифрование с закрытым ключом называют такжке симметричным шифрованием, так как для шифрования и расшифровки используется один и тот же ключ. Алгоритмы шифрования с закрытым ключом являются очень быстрыми (по сравнению с алгоритмами шифрования с открытым ключом) и хорошо подходят для осуществления криптографических преобразований больших массивов информации.

Разновидность алгоритмов шифрования с закрытым ключом, называемая блочным шифром, используется для шифрования целого блока данных за один раз. Блочные шифры (такие как *DES*, *TrippleDES* и *AES*) преобразуют входной блок данных длиной в n байтов в выходной блок зашифрованных данных. Если необходимо зашифровать или расшифровать последовательность байтов, следует делать это блок за блоком. Поскольку n достаточно мало (8 байт для *DES* и *TripleDES*; 16 байт [по умолчанию], 24 или 32 байта для *AES*), данные большей длины, чем n, должны шифроваться блоками, один блок за раз. Блоки данных размером менее n байт должны быть увеличены до n байт перед обработкой.

Одна из простейших форм блочного шифра называется режимом электронной кодовой книги (*ECB*). Режим *ECB* не считается безопасным, поскольку в нем не используется вектор инициализации для инициализации первого текстового блока. Для заданного закрытого ключа k простой блочный шифр, не использующий вектор инициализации, зашифрует одинаковые входные блоки текста в одинаковые выходные блоки зашифрованного текста. Поэтому если во входном текстовом потоке присутствуют одинаковые блоки, в зашифрованном потоке также будут присутствовать одинаковые блоки. Такие повторяющиеся выходные блоки сообщают неправомочным пользователям о ненадежных алгоритмах шифрования, которыми можно воспользоваться, о и возможных типах атак. Шифр *ECB* поэтому очень уязвим для анализа и, в конечном итоге, взлому ключа.

Классы блочных шифров, предоставляемые библиотекой базовых классов, используют режим сцепления, называемый сцеплением шифровальных блоков (*CBC*), хотя данную настройку по умолчанию можно изменить. Шифры *CBC* решают проблемы, связанные с использованием шифров *ECB*, благодаря использованию вектора инициализации (*IV*) для шифрования первого текстового блока. Перед шифрованием каждого блока открытого текста он объединяется с зашифрованным текстом предыдущего блока с помощью поразрядной исключающей операции *OR* (*XOR*). Поэтому каждый блок зашифрованного текста зависит от всех предыдущих блоков.

При использовании этой системы шифрования стандартные заголовки сообщений, которые могут быть известны неправомочному пользователю, не могут быть использованы им для восстановления закрытого ключа.

Недостатком шифрования с закрытым ключом является необходимость того, чтобы две стороны согласовали ключ и вектор инициализации, для чего может потребоваться их передача через систему связи. Вектор инициализации не считается секретным и может передаваться вместе с сообщением в текстовом формате. Однако ключ не должен стать известен неправомочным пользователям. Из-за указанных проблем шифрование с закрытым ключом часто используется в сочетании с шифрованием с открытым ключом для безопасной передачи ключа и вектора инициализации.

Предположим, что Алиса и Боб являются двумя сторонами, которые хотят осуществлять связь по незащищенному каналу; они могли бы воспользоваться шифрованием с закрытым ключом следующим образом. Алиса и Боб соглашаются использовать некоторый определенный алгоритм (например, *AES*) с определенным ключом и вектором инициализации. Алиса пишет сообщение и создает сетевой поток, через который можно отправить это сообщение (например, именованный канал или канал электронной почты). Затем она шифрует текст с помощью ключа и вектора инициализации и по интрасети пересылает зашифрованное сообщение и вектор инициализации Бобу. Боб принимает зашифрованный текст и осуществляет расшифровку, используя ранее согласованные ключ и вектор инициализации. Если происходит перехват передаваемых данных, злоумышленник не может восстановить исходное сообщение, так как ему не известны ключ и вектор инициализации. В этой ситуации в секрете должен сохраняться только ключ. В более реалистичном случае либо Алиса, либо Боб создает закрытый ключ и использует шифрование с открытым ключом (асимметричное) для передачи другой стороне закрытого (симметричного) ключа.

Microsoft .NET Framework предоставляет следующие классы, которые реализуют алгоритмы шифрования с закрытым ключом:

- *AesManaged;*
- *DESCryptoServiceProvider;*
- *HMACSHA1;*
- *RC2CryptoServiceProvider;*
- *RijndaelManaged;*

- *TripleDESCryptoServiceProvider.*

3. Практическая работа

Классы пространства имен *System.Security.Cryptography.Xml* могут использоваться для шифрования элементов в XML-документе. Шифрование XML-данных позволяет хранить и передавать важные XML-данные, не беспокоясь о том, что они могут быть прочитаны.

При использовании симметричного алгоритма шифрования, такого как AES, также известного как алгоритм Rijndael, необходимо использовать один и тот же ключ как для шифрования, так и для расшифровки XML-данных. В лабораторной работе предполагается, что зашифрованный XML-документ будет расшифровываться с помощью того же ключа, а между шифрующей и расшифровывающей сторонами существует соглашение по поводу используемых алгоритма и ключа. В этом примере в зашифрованный XML-документ ключ AES не включается (в зашифрованном или незашифрованном виде).

Ниже пошагово демонстрируется пример шифрования и расшифровывания XML-элемента с использованием алгоритма AES (Rijndael).

3.1 Шифрование XML-элементов с помощью симметричного ключа

1) Создайте новый проект в среде *Microsoft Visual Studio 2010* и выберите язык программирования *C#.*

2) Включите в проект следующие пространства имен:

```
System.Xml, System.Security.Cryptography и
System.Security.Cryptography.Xml.
```

3) Создайте в том же каталоге, где будет располагаться скомпилированная программа, XML-файл с именем "test.xml" и элементом "creditcard":

```
<root>
<creditcard>
        <number>19834209</number>
        <expiry>02/02/2002</expiry>
</creditcard>
</root>
```

4) Создайте симметричный ключ, используя класс *RijndaelManaged*. Этот ключ будет использоваться для шифрования XML-элемента:

```
RijndaelManaged key = null;
key = new RijndaelManaged();
```

5) Создайте объект *XmlDocument* путем загрузки XML-файла с диска. Объект *XmlDocument* содержит XML-элемент, подлежащий шифрованию:

```
XmlDocument xmlDoc = new XmlDocument();
xmlDoc.PreserveWhitespace = true;
xmlDoc.Load("test.xml");
```

6) Найдите указанный элемент в объекте *XmlDocument* и создайте новый объект *XmlElement*, который будет представлять элемент, подлежащий шифрованию ("creditcard"):

```
XmlElement elementToEncrypt =
Doc.GetElementsByTagName(ElementName)[0] as XmlElement;
```

7) Создайте новый экземпляр класса *EncryptedXml* и используйте его для шифрования *XmlElement* с помощью симметричного ключа. Метод *EncryptData* возвращает зашифрованный элемент в виде массива шифрованных байтов:

```
EncryptedXml eXml = new EncryptedXml();
byte[] encryptedElement = eXml.EncryptData(elementToEncrypt, Key, false);
```

8) Создайте объект *EncryptedData* и разместите в нем URL-идентификатор зашифрованного XML-элемента. URL-идентификатор сообщает расшифровывающей стороне о том, что XML-документ содержит зашифрованный элемент. Для указания URL-идентификатора может использоваться поле *XmlEncElementUrl*:

```
EncryptedData edElement = new EncryptedData();
edElement.Type = EncryptedXml.XmlEncElementUrl;
```

9) Создайте объект *EncryptionMethod*, инициализируемый с помощью URL-идентификатора криптографического алгоритма, использовавшегося для создания ключа. Передайте объект *EncryptionMethod* свойству *EncryptionMethod*:

```
string encryptionMethod = null;
if (Key is TripleDES)
{
encryptionMethod = EncryptedXml.XmlEncTripleDESUrl;
}
else if (Key is DES)
{
encryptionMethod = EncryptedXml.XmlEncDESUrl;
}
if (Key is Rijndael)
{
switch (Key.KeySize)
{
case 128:
        encryptionMethod = EncryptedXml.XmlEncAES128Url;
        break;
case 192:
        encryptionMethod = EncryptedXml.XmlEncAES192Url;
        break;
case 256:
        encryptionMethod = EncryptedXml.XmlEncAES256Url;
        break;
}
}
else
{
throw new CryptographicException("Алгоритм не поддерживает XML
Encryption");
}
edElement.EncryptionMethod=
new EncryptionMethod(encryptionMethod);
```

10) Добавьте зашифрованные данные элемента в объект *EncryptedData*:

```
edElement.CipherData.CipherValue = encryptedElement;
```

11) Замените элемент из исходного объекта *XmlDocument* элементом *EncryptedData*:

```
EncryptedXml.ReplaceElement(elementToEncrypt, edElement, false);
```

3.2 Расшифровывание XML-элементов с помощью симметричного ключа

1) Выполните шифрование XML-элемента с помощью предварительно созданного ключа в п.3.1.

2) Найдите элемент *EncryptedData* в объекте *XmlDocument*, содержащий зашифрованные XML-данные, и создайте новый объект *XmlElement* для представления этого элемента:

```
XmlElement encryptedElement = Doc.GetElementsByTagName("EncryptedData")[0] as XmlElement;
```

3) Создайте объект *EncryptedData*, загрузив необработанные XML-данные из ранее созданного объекта *XmlElement*:

```
EncryptedData edElement = new EncryptedData();
edElement.LoadXml(encryptedElement);
```

4) Создайте новый объект *EncryptedXml* и используйте его для расшифровки XML-данных с помощью ключа, который использовался при шифровании:

```
EncryptedXml exml = new EncryptedXml();
byte[] rgbOutput = exml.DecryptData(edElement, Alg);
```

5) В XML-документе замените зашифрованный элемент созданным элементом, имеющим формат обычного текста:

```
exml.ReplaceData(encryptedElement, rgbOutput);
```

3.3 Задание к лабораторной работе

Составить программу, шифрующую и расшифровывающую XML-элемент с использованием одного из следующих классов, реализующих шифрование с закрытым ключом:

- *AesManaged;*
- *DESCryptoServiceProvider;*
- *RC2CryptoServiceProvider;*
- *RijndaelManaged;*
- *TripleDESCryptoServiceProvider.*

Программа, должна содержать класс, реализующий метод для шифрования и метод для расшифровывания XML-элемента

Контрольные вопросы

1. В чем состоит особенность шифрования с помощью симметричного ключа?
2. Какие требования предъявляются к шифрованию с помощью симметричного ключа?
3. Каким образом осуществляеться обмен ключами при шифровании с помощью симметричного ключа?
4. Перечислите и кратко охарактеризуйте основные алгоритмы симметричного шифрования.
5. Какие классы используются для шифрования в составе пространства имен *System.Security.Cryptography* и *System.Security.Cryptography.Xml*?

Лабораторная работа № 2. Шифрование и расшифрование XML-элементов с помощью асимметричного ключа

1. ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Целью работы является изучение студентами стандартных классов фреймворка *Microsoft .NET Framework* для асимметричного шифрования.

Задачами работы являются:

- ознакомление с примером использования класса *RSACryptoServiceProvider*, реализующим алгоритм шифрования с открытым;
- разработка программы, реализующей шифрование XML-элемента открытым ключом и его расшифровывание закрытым ключом.

2. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

При шифровании с открытым ключом используются закрытый ключ, который должен храниться в секрете от неправомочных пользователей, а также открытый ключ, который может предоставляться кому угодно. Открытый и закрытый ключи математически взаимосвязаны; данные, зашифрованные с помощью открытого ключа,

можно расшифровать исключительно с помощью соответствующего закрытого ключа. Открытый ключ может быть предоставлен любому лицу; он используется для шифрования данных, которые должны быть отправлены хранителю закрытого ключа. Алгоритмы шифрования с открытым ключом также известны как асимметричные алгоритмы, потому что для шифрования данных требуется один ключ, а для расшифровки – другой ключ. Основное правило шифрования запрещает повторное использование ключа; оба ключа в каждом сеансе шифрования должны быть уникальными. Однако на практике ассиметричные ключи обычно используются подолгу.

Использование алгоритмов шифрования с открытым ключом имеет свои особенности:

- Алгоритмы шифрования с открытым ключом используют буфер фиксированного размера, в то время как алгоритмы шифрования с закрытым ключом могут использовать буфер переменного размера.
- Алгоритмы шифрования с открытым ключом не могут быть использованы для сцепления блоков данных в потоки тем же образом, как в алгоритмах шифрования с закрытым ключом, потому что можно шифровать только маленькие порции данных. Соответственно, асимметричные операции не используют такую же потоковую модель, как симметричные операции.
- Шифрование с открытым ключом имеет намного большее пространство ключей (диапазон возможных значений ключа), чем шифрование с закрытым ключом. Поэтому такое шифрование менее уязвимо к атакам полного перебора, когда проверяются все возможные значения ключа.
- Открытые ключи можно легко распространять, поскольку они не требуют особой защиты при условии, что существует некий способ установления подлинности источника.
- Некоторые алгоритмы шифрования с открытым ключом (такие как *RSA* и *DSA*, но не *Diffie-Hellman*) могут быть использованы для создания цифровых подписей, служащих для подтверждения идентичности лица, от которого исходят данные.
- *RSA* допускает как шифрование, так и подписывание, в то время как *DSA* может использоваться только для подписывания, а *Diffie-Hellman* – только для генерации ключей. В целом, алгоритмы с открытым ключом имеют более ограниченную сферу применения, чем алгоритмы с закрытым ключом.
- Алгоритмы шифрования с открытым ключом являются весьма медленными (по сравнению с алгоритмами шифрования с закрытым ключом) и не предназначены для шифрования больших объемов данных. Использование шифрования с открытым ключом имеет смысл только при передаче очень малых массивов данных. Обычно шифрование с открытым ключом применяется для того, чтобы зашифровать ключ и вектор инициализации, которые будут использоваться при шифровании с закрытым ключом. После передачи ключа и вектора инициализации используется уже шифрование с закрытым ключом.

Рассмотрим ситуацию с использованием шифрования с открытым ключом.

Две стороны (Алиса и Боб) могут использовать шифрование с открытым ключом следующим образом. Сначала Алиса создает набор, состоящий из открытого и закрытого ключей. Если Боб хочет послать Алисе зашифрованное сообщение, он запрашивает у Алисы ее открытый ключ. Алиса высылает Бобу свой открытый ключ через незащищенную сеть, и Боб использует полученный ключ для шифрования своего сообщения. Боб пересылает зашифрованное сообщение Алисе, а она расшифровывает полученные данные с помощью своего закрытого ключа. Если Боб получил ключ Алисы по незащищенному каналу, например через открытую сеть, он оказывается уязвимым для атак типа "злоумышленник в середине". Поэтому Бобу необходимо убедиться, что Алиса имеет правильную копию открытого ключа.

Во время передачи Бобу открытого ключа Алисы может произойти несанкционированный перехват ключа третьим лицом. Более того, возможна ситуация, что то же самое лицо перехватит зашифрованное сообщение Боба. Тем не менее неправомочная сторона не может осуществить расшифровку сообщения с помощью открытого ключа. Сообщение можно расшифровать только с помощью закрытого ключа Алисы, который не передавался через какие-либо системы связи. Алиса не использует свой закрытый ключ для шифрования ответного сообщения Бобу, потому что любое лицо может расшифровать это сообщение с помощью открытого ключа. Если Алиса желает послать Бобу ответное сообщение, она запрашивает у Боба его открытый ключ и шифрует свое сообщение, используя полученный открытый ключ. Затем Боб осуществляет расшифровку сообщения с помощью своего соответствующего закрытого ключа.

В этом случае Алиса и Боб используют шифрование с открытым ключом (асимметричное) для передачи закрытого (симметричного) ключа, а затем пользуются шифрованием с этим закрытым ключом в течение сеанса связи.

Microsoft .NET Framework предоставляет следующие классы, которые реализуют алгоритмы шифрования с открытым ключом:

- *DSACryptoServiceProvider;*
- *RSACryptoServiceProvider;*
- *ECDiffieHellman;*
- *ECDiffieHellmanCng;*
- *CDiffieHellmanCngPublicKey;*
- *ECDiffieHellmanKeyDerivationFunction;*
- *ECDsaCng.*

3. ПРАКТИЧЕСКАЯ РАБОТА

В лабораторной работе необходимо реализовать шифрование XML-элемента с использованием двух ключей.

В демонстрируемом ниже примере создается пара ключей, состоящая из открытого и закрытого ключа *RSA*, которая сохраняется в безопасный контейнер ключа. Далее создается отдельный ключ сеанса с использованием алгоритма *AES (Rijndael*) для шифрования XML-документа с последующим использованием открытого ключа *RSA* для шифрования ключа сеанса *AES*. Наконец зашифрованный ключ сеанса *AES* и зашифрованные XML-данные сохраняются в XML-документе в новом элементе *EncryptedData*.

Для расшифровки XML-элемента из контейнера ключа извлекается закрытый ключ *RSA*, который затем используется для шифрования ключа сеанса. Ключ сеанса далее используется для расшифровки документа.

Этот пример применим в том случае, если необходимо обеспечить общий доступ к зашифрованным данным со стороны нескольких приложений, или если приложение должно сохранять зашифрованные данные между запусками.

3.1 Шифрование XML-элементов с помощью асимметричного ключа

1) Создайте новый проект в среде *Microsoft Visual Studio 2010* и выберите язык программирования *C#.*

2) Включите в проект следующие пространства имен:

```
System.Xml, System.Security.Cryptography
и System.Security.Cryptography.Xml.
```

3) Создайте в том же каталоге, где будет располагаться скомпилированная программа, XML-файл с именем "test.xml" и элементом "creditcard":

```
<root>
        <creditcard>
                <number>19834209</number>
                <expiry>02/02/2002</expiry>
```

```
        </creditcard>
    </root>
```

4) Создайте объект *CspParameters* и укажите имя контейнера ключа:

```
CspParameters cspParams = new CspParameters();
cspParams.KeyContainerName = "XML_ENC_RSA_KEY";
```

5) Создайте симметричный ключ, используя класс *RSACryptoServiceProvider*. Ключ автоматически сохраняется в контейнере ключа при передаче объекта *CspParameters* конструктору класса *RSACryptoServiceProvider*. Этот ключ будет использоваться для шифрования ключа сеанса *AES* и может быть извлечен в дальнейшем для его расшифровки:

```
RSACryptoServiceProvider rsaKey =
new RSACryptoServiceProvider(cspParams);
```

6) Создайте объект *XmlDocument* путем загрузки XML-файла с диска. Объект *XmlDocument* содержит XML-элемент, подлежащий шифрованию:

```
XmlDocument xmlDoc = new XmlDocument();
try
{
        xmlDoc.PreserveWhitespace = true;
        xmlDoc.Load("test.xml");
}
catch (Exception e)
{
        Console.WriteLine(e.Message);
}
```

7) Найдите указанный элемент ("creditcard") в объекте *XmlDocument* и создайте новый объект *XmlElement*, который будет представлять элемент, подлежащий шифрованию:

```
XmlElement elementToEncrypt =
Doc.GetElementsByTagName(ElementToEncrypt)[0] as XmlElement;

if (elementToEncrypt == null)
{
        throw new XmlException("Указанный элемент не найден");
}
```

8) Создайте новый ключ сеанса с помощью класса *RijndaelManaged*. Этот ключ будет использоваться для шифрования XML-элемента, а затем будет сам зашифрован и помещен в XML-документ:

```
sessionKey = new RijndaelManaged();
sessionKey.KeySize = 256;
```

9) Создайте новый экземпляр класса *EncryptedXml* и используйте его для шифрования указанного элемента с помощью ключа сеанса. Метод *EncryptData* возвращает зашифрованный элемент в виде массива шифрованных байтов:

```
EncryptedXml eXml = new EncryptedXml();
byte[] encryptedElement = eXml.EncryptData(elementToEncrypt, sessionKey, false);
```

10) Создайте объект *EncryptedData* и разместите в нем URL-идентификатор зашифрованного XML-элемента. URL-идентификатор сообщает расшифровывающей стороне о том, что XML-документ содержит зашифрованный элемент. Для указания URL-идентификатора может использоваться поле *XmlEncElementUrl*. XML-элемент в формате простого текста будет замещен элементом *EncryptedData*, инкапсулированным в объекте *EncryptedData*:

```
EncryptedData edElement = new EncryptedData();
edElement.Type = EncryptedXml.XmlEncElementUrl;
edElement.Id = EncryptionElementID;
```

11) Создайте объект *EncryptionMethod*, инициализируемый с помощью URL-идентификатора криптографического алгоритма, использовавшегося для создания ключа сеанса. Передайте объект *EncryptionMethod* свойству *EncryptionMethod*:

```
edElement.EncryptionMethod = new
EncryptionMethod(EncryptedXml.XmlEncAES256Url);
```

12) Создайте объект *EncryptedKey*, который будет содержать зашифрованный ключ сеанса. Зашифруйте ключ сеанса, добавьте его в объект *EncryptedKey* и введите имя ключа сеанса и URL-идентификатор ключа:

```
EncryptedKey ek = new EncryptedKey();
byte[] encryptedKey = EncryptedXml.EncryptKey(sessionKey.Key, Alg, false);
ek.CipherData = new CipherData(encryptedKey);
ek.EncryptionMethod =
new EncryptionMethod(EncryptedXml.XmlEncRSA15Url);
```

13) Создайте новый объект *DataReference*, с помощью которого зашифрованные данные будут сопоставляться с определенным ключом сеанса. Эта необязательная операция позволяет легко указать различные части XML-документа, зашифрованные с помощью одного ключа:

```
DataReference dRef = new DataReference();
dRef.Uri = "#" + EncryptionElementID;
ek.AddReference(dRef);
```

14) Добавьте зашифрованный ключ в объект *EncryptedData*:

```
edElement.KeyInfo.AddClause(new KeyInfoEncryptedKey(ek));
```

15) Создайте новый объект *KeyInfo*, чтобы указать имя ключа *RSA*. Добавьте его в объект *EncryptedData*. Это позволяет расшифровывающей стороне правильно определить асимметричный ключ, который необходимо использовать для расшифровки ключа сеанса:

```
KeyInfoName kin = new KeyInfoName();
kin.Value = KeyName;
ek.KeyInfo.AddClause(kin);
```

16) Добавьте зашифрованные данные элемента в объект *EncryptedData*:

```
edElement.CipherData.CipherValue = encryptedElement;
```

17) Замените элемент из исходного объекта *XmlDocument* элементом *EncryptedData*:

```
EncryptedXml.ReplaceElement(elementToEncrypt, edElement, false);
```

18) Сохраните объект *XmlDocument*:

```
xmlDoc.Save("test.xml");
```

3.2 Расшифровка XML-элементов с помощью асимметричного ключа

1) Выполните п.3.1.

2) Создайте объект *CspParameters* и укажите имя контейнера ключа:

```
CspParameters cspParams = new CspParameters();
cspParams.KeyContainerName = "XML_ENC_RSA_KEY";
```

3) Извлеките ранее созданный асимметричный ключ из контейнера с помощью объекта *RSACryptoServiceProvider*. Ключ автоматически извлекается из контейнера ключа при передаче объекта *CspParameters* конструктору *RSACryptoServiceProvider*:

```
RSACryptoServiceProvider rsaKey =
new RSACryptoServiceProvider(cspParams);
```

4) Создайте новый объект *EncryptedXml* для расшифровки документа:

```
EncryptedXml exml = new EncryptedXml(Doc);
```

5) Добавьте сопоставление ключа и имени, чтобы привязать ключ *RSA* к элементу, подлежащему расшифровке, в документе. Для ключа следует использовать то же имя, что и при шифровании документа. Обратите внимание, что это имя независимо от имени, использовавшегося для определения ключа в контейнере ключа, которое было указано в п. 3.1:

```
exml.AddKeyNameMapping(KeyName, Alg);
```

6) Вызовите метод *DecryptDocument* для расшифровки элемента *EncryptedData*. Этот метод использует ключ *RSA* для расшифровки ключа сеанса и автоматически использует ключ сеанса для расшифровки XML-элемента. Он также автоматически замещает элемент *EncryptedData* исходным элементом в формате простого текста:

```
exml.DecryptDocument();
```

7) Сохраните XML-документ:

```
xmlDoc.Save("test.xml");
```

3.3 Задание к лабораторной работе

Необходимо разработать программу, в которой создается пара ключей, состоящая из открытого и закрытого ключа, которая сохраняется в безопасный контейнер ключа. Далее создается отдельный ключ сеанса с использованием алгоритма *AES (Rijndael*) для шифрования XML-документа с последующим использованием открытого ключа для шифрования ключа сеанса *AES*.

Далее программа должна расшифровывать зашифрованное сообщение.

Для создания открытого ключа используйте класс *RSACryptoServiceProvider*.

Контрольные вопросы

1. В чем состоит особенность шифрования с помощью асимметричного ключа?
2. Какие требования предъявляются к шифрованию с помощью асимметричного ключа?
3. Каким образом осуществляться обмен ключами при шифровании с помощью асимметричного ключа?
4. Перечислите и кратко охарактеризуйте основные алгоритмы асимметричного шифрования.
5. Можно ли внедрять ключи непосредственно в исходный код программы?
6. Каким образом можно удалить криптографический ключ из памяти после завершения его использования?

ЛАБОРАТОРНАЯ РАБОТА № 3. ПОДПИСЫВАНИЕ XML-ДОКУМЕНТОВ С ПОМОЩЬЮ ЦИФРОВЫХ ПОДПИСЕЙ И ПРОВЕРКА ЦИФРОВЫХ ПОДПИСЕЙ XML-ДОКУМЕНТОВ

1. ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Целью работы является изучение студентами стандартных классов фреймворка *Microsoft .NET Framework* для подписывания электронных документов.

Задачами работы являются:

- ознакомление с примером использования класса *RSACryptoServiceProvider* для создания электронной подписи;
- разработка программы, реализующей создание и прикрепление цифровой подписи к документу с последующей проверкой цифровой подписи.

2. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Алгоритмы шифрования с открытым ключом могут также использоваться для создания цифровых подписей. Цифровые подписи удостоверяют подлинность источника данных (если вы доверяете открытому ключу источника) и защищают целостность данных. С помощью открытого ключа, созданного Алисой, получатель данных,

отправленных Алисой, может проверить, что их источником действительно является Алиса, сравнив цифровую подпись полученных данных с открытым ключом Алисы.

Чтобы использовать шифрование с открытым ключом для создания цифровой подписи сообщения, Алиса сначала применяет к этому сообщению алгоритм хэширования, который создает хэш-функцию сообщения. Хэш-функция сообщения является компактным и уникальным отображением данных. Для создания своей личной цифровой подписи Алиса шифрует хэш-функцию сообщения с помощью своего закрытого ключа. При получении сообщения и цифровой подписи Боб расшифровывает подпись с помощью открытого ключа Алисы, восстанавливая зашифрованную хэш-функцию сообщения, а затем осуществляет хэширование сообщения с помощью того же алгоритма, который использовала Алиса. Если вычисленная Бобом хэш-функция в точности совпадает с хэш-функцией, полученной от Алисы, Боб может быть уверен, что сообщение пришло действительно от владельца закрытого ключа, а также в том, что данные в сообщении не подвергались модификациям. Если Боб уверен, что именно Алиса владеет закрытым ключом, он удостоверяется, что сообщение пришло от Алисы.

Цифровая подпись может быть проверена любым лицом, потому что открытый ключ источника данных является общедоступным и обычно включается в формат цифровой подписи. Следует помнить о том, что цифровая подпись не обеспечивает секретности сообщения; для обеспечения секретности его следует еще и зашифровать.

Microsoft .NET Framework предоставляет следующие классы, которые реализуют алгоритмы создания цифровой подписи:

- *DSACryptoServiceProvider;*
- *RSACryptoServiceProvider;*
- *ECDsa;*
- *ECDsaCng.*

3. ПРАКТИЧЕСКАЯ РАБОТА

В лабораторной работе необходимо прикрепить цифровую подпись к XML-документу, а затем проверить указанную цифровую подпись.

В демонстрируемом ниже примере создается цифровая подпись для XML-документа, которая затем прикрепляется к нему с помощью элемента *Signature*.

Цифровая подпись представляет собой подписывающий ключ *RSA*, который затем добавляется в безопасный контейнер ключа и используется для подписывания XML-документа. Этот ключ может быть извлечен для проверки цифровой подписи XML, либо может использоваться для подписывания другого XML-документа.

3.1 Подписывание XML-документа с помощью цифровой подписи

1) Создайте новый проект в среде *Microsoft Visual Studio 2010* и выберите язык программирования *C#.*

2) Включите в проект следующие пространства имен:

```
System.Xml, System.Security.Cryptography
и System.Security.Cryptography.Xml.
```

3) Создайте в том же каталоге, где будет располагаться скомпилированная программа, XML-файл с именем "test.xml" и элементом "creditcard":

```
<root>
        <creditcard>
                <number>19834209</number>
                <expiry>02/02/2002</expiry>
        </creditcard>
</root>
```

4) Создайте объект *CspParameters* и укажите имя контейнера ключа:

```
CspParameters cspParams = new CspParameters();
cspParams.KeyContainerName = "XML_DSIG_RSA_KEY";
```

5) Создайте симметричный ключ, используя класс *RSACryptoServiceProvider*. Ключ автоматически сохраняется в контейнере ключа при передаче объекта *CspParameters* конструктору класса *RSACryptoServiceProvider*. Этот ключ будет использоваться для подписывания XML-документа:

```
RSACryptoServiceProvider rsaKey =
new RSACryptoServiceProvider(cspParams);
```

6) Создайте объект *XmlDocument* путем загрузки XML-файла с диска. Объект *XmlDocument* содержит XML-элемент, подлежащий шифрованию:

```
XmlDocument xmlDoc = new XmlDocument();
xmlDoc.PreserveWhitespace = true;
xmlDoc.Load("test.xml");
```

7) Создайте новый объект *SignedXml* и передайте ему объект *XmlDocument*:

```
SignedXml signedXml = new SignedXml(xmlDoc);
```

8) Добавьте подписывающий ключ *RSA* в объект *SignedXml*:

```
signedXml.SigningKey = Key;
```

9) Создайте объект *Reference*, определяющий подписываемый документ. Чтобы подписать весь документ, установите для свойства *Uri* значение " ":

```
Reference reference = new Reference();
reference.Uri = "";
```

10) Добавьте объект *XmlDsigEnvelopedSignatureTransform* в объект *Reference*. Преобразование позволяет проверяющему представлять XML-данные в той же форме, которая использовалась подписывающей стороной. XML-данные могут представляться по-разному, поэтому этот шаг крайне важен для проведения проверки:

```
XmlDsigEnvelopedSignatureTransform env = new
XmlDsigEnvelopedSignatureTransform();
reference.AddTransform(env);
```

11) Добавьте объект *Reference* в объект *SignedXml* и произведите вычисление подписи, вызвав метод *ComputeSignature*:

```
signedXml.AddReference(reference);
signedXml.ComputeSignature();
```

12) Извлеките XML-представление подписи (элемент *Signature*) и сохраните его в новый объект *XmlElement*, а затем присоедините элемент к объекту *XmlDocument*:

```
XmlElement xmlDigitalSignature = signedXml.GetXml();
xmlDoc.DocumentElement.AppendChild (xmlDoc.ImportNode(xmlDigitalSignature,
true));
```

13) И, наконец, сохраните документ:

```
xmlDoc.Save("test.xml");
```

3.2 Проверка цифровой подписи XML-документа

1) Выполните п.3.1.

2) Для проверки документа, необходимо использовать тот же асимметричный ключ, который использовался при подписывании. Создайте объект *CspParameters* и укажите имя контейнера ключа, который использовался при подписывании:

```
CspParameters cspParams = new CspParameters();
cspParams.KeyContainerName = "XML_ENC_RSA_KEY";
```

3) Извлеките открытый ключ, используя класс *RSACryptoServiceProvider*. Ключ автоматически извлекается из контейнера ключа при передаче объекта *CspParameters* конструктору *RSACryptoServiceProvider*:

```
RSACryptoServiceProvider rsaKey =
new RSACryptoServiceProvider(cspParams);
```

4) Создайте объект *XmlDocument* путем загрузки XML-файла с диска. Объект *XmlDocument* содержит подписанный XML-документ, подлежащий проверке:

```
XmlDocument xmlDoc = new XmlDocument();
xmlDoc.PreserveWhitespace = true;
xmlDoc.Load("test.xml");
```

5) Создайте новый объект *SignedXml* и передайте ему объект *XmlDocument*:

```
SignedXml signedXml = new SignedXml(Doc);
```

6) Найдите элемент *signature* и создайте новый объект *XmlNodeList*:

```
XmlNodeList nodeList = Doc.GetElementsByTagName("Signature");
```

7) Загрузите XML-данные первого элемента *signature* в объект *SignedXml*:

```
signedXml.LoadXml((XmlElement)nodeList[0]);
```

8) Проверьте подпись с помощью метода *CheckSignature* и открытого ключа *RSA*. Метод возвращает логическое значение, указывающее на успешность выполнения операции:

```
return signedXml.CheckSignature(Key);
```

3.3 Задание к лабораторной работе

Необходимо разработать программу, в которой создается пара открытый ключ – закрытый ключ, затем документ подписывается закрытым ключом и осуществляется проверка подписи с помощью открытого ключа.

Для создания цифровой подписи используйте один из следующих классов:

- *DSACryptoServiceProvider;*
- *ECDsa;*
- *ECDsaCng.*

Контрольные вопросы

6. Для чего необходима ЭЦП и что удостоверяет цифровая подпись документа?
7. В чем состоит особенность использования асимметричных криптосистем в электронной цифровой подписи?
8. Какие криптоалгоритмы применяются в ЭЦП?
9. Какие классы *Microsoft .NET Framework* наиболее предпочтительны для цифровой подписи? Почему?
10. Укажите наиболее уязвимые места с точки зрения информационной безопасности при использовании ЭЦП.

ЛАБОРАТОРНАЯ РАБОТА № 4. СОЗДАНИЕ И ПРОВЕРКА ХЭША

1. ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Целью работы является изучение студентами стандартных классов фреймворка *Microsoft .NET Framework* для хэширования данных.

Задачами работы являются:

- ознакомление с примером использования класса *SHA1Managed* для создания хэша;
- разработка программы, реализующей создание и проверку хэша произвольного файла.

2. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Алгоритмы хэширования преобразуют двоичные последовательности произвольной длины в двоичные последовательности фиксированного меньшего размера, известные как хэш-коды. Хэш-код является числовым представлением порции данных. Если осуществляется хэширование абзаца текста и в нем изменяется хотя бы одна буква, результат хэширования изменится. Если хэш является криптостойким, его код значительно изменится. Например, если изменяется один бит сообщения, результат выполнения криптостойкой хэш-функции может отличаться на 50%. Несколько входных

значений могут преобразовываться в один хэш-код. Однако в вычислительном плане немыслимо найти два разных входных набора данных, результаты хэширования которых полностью совпадают.

Две стороны (Алиса и Боб) могут использовать хэш-функцию для проверки целостности сообщений. Им нужно выбрать хэш-алгоритм для подписывания сообщений. Алиса будет писать сообщение, а затем создавать хэш этого сообщения с помощью выбранного алгоритма. Затем стороны могут применять один из следующих вариантов дальнейших действий.

Алиса отправляет Бобу сообщение с открытым текстом и хэш сообщения (цифровую подпись). Боб получает сообщение, применяет к нему хэш-алгоритм и сравнивает свое значение хэша со значением, полученным от Алисы. Если хэш-коды совпадают, значит, данные не изменялись. Если хэш-коды не совпадают, значит, данные изменялись после создания.

Однако этот способ не позволяет установить подлинность отправителя. Любой человек может выдавать себя за Алису и отправлять сообщения Бобу. Для этого достаточно подписывать сообщения с помощью такого же хэш-алгоритма, и Боб сможет лишь проверить, что сообщение соответствует подписи. Это одна из форм атаки "злоумышленник в середине".

Алиса отправляет Бобу сообщение с открытым текстом по незащищенному открытому каналу. По защищенному закрытому каналу она оправляет Бобу хэш сообщения. Боб получает сообщение с открытым текстом, хэширует его и сравнивает хэш со значением, полученным по закрытому каналу. Если хэши совпадают, Боб может сделать два вывода:

1) сообщение не было изменено;
2) отправитель сообщения подлинный (Алиса).

Для того чтобы такая система работала, Алиса должна скрывать оригинальный хэш-код от всех, кроме Боба.

- Алиса по незащищенному открытому каналу отправляет Бобу сообщение в виде открытого текста, а хэш сообщения помещает на свой общедоступный веб-узел.

Такой подход позволяет защититься от изменения хэша третьей стороной. Хотя сообщение и его хэш могут видеть все, изменить хэш может только Алиса. Злоумышленнику, который хочет выдать себя за Алису, потребуется доступ к веб-узлу Алисы.

Ни один из описанных выше способов не защищает Алису от чтения ее сообщений третьими лицами, поскольку сообщения передаются в виде открытого текста. Для обеспечения полной защиты обычно требуется использовать цифровые подписи и шифрование.

Microsoft .NET Framework предоставляет следующие классы, которые реализуют алгоритмы хэширования:

- *HMACSHA1;*
- *MACTripleDES;*
- *MD5CryptoServiceProvider;*
- *RIPEMD160;*
- *SHA1Managed;*
- *SHA256Managed;*
- *SHA384Managed;*
- *SHA512Managed;*
- *Разновидности HMAC всех алгоритмов SHA, MD5 и RIPEMD-160;*
- *Реализации CryptoServiceProvider (оболочки управляемого кода) всех алгоритмов SHA;*

- *Реализации криптографии следующего поколения (CNG) всех алгоритмов MD5 и SHA.*

В настоящее время алгоритмы *MD5* и *SHA-1* признаны недостаточно стойкими и вместо них рекомендуется использовать алгоритм *SHA-2*.

Проверку целостности данных можно производить на основании сравнения их с хэш-кодом. Обычно данные хэшируются в некоторый момент времени, а затем их хэш-код защищается каким-либо образом. Позже можно снова хэшировать эти данные и результат сравнивать с полученным ранее защищенным хэш-кодом. Если хэш-коды совпадают, значит, данные не изменялись. Несовпадение хэш-кодов свидетельствует о том, что данные были повреждены. Чтобы такой механизм был работоспособен, защищенный хэш-код должен быть зашифрован или являться недоступным для всех лиц, не имеющих достаточного доверия.

3. ПРАКТИЧЕСКАЯ РАБОТА

В лабораторной работе необходимо создать и проверить хэш указанной строки. Управляемые классы, реализующие хэширование, могут быть использованы для хэширования либо байтового массива, либо управляемого объекта потока. В демонстрируемом ниже примере хэш-алгоритм *SHA1* используется для создания хэш-кода строки. Здесь класс *UnicodeEncoding* используется для преобразования строки в массив байтов, которые хэшируются с помощью класса *SHA1Managed*. После этого хэш-код выводится на консоль.

При проверке хэша осуществляется сравнение ранее полученного хэш-кода строки с ее новым хэш-кодом. В приведенном примере реализован цикл, производящий побайтовое сравнение хэш-кодов.

3.1 Создание хэша

1) Создайте новый проект в среде *Microsoft Visual Studio 2010* и выберите язык программирования *C#*.

2) Включите в проект следующие пространства имен:

```
System.IO, System.Security.Cryptography и System.Text.
```

3) Создайте строку с сообщением:

```
string MessageString = "Исходное сообщение";
```

4) Создайте объект *UnicodeEncoding* и преобразуйте исходную строку в байтовый массив:

```
UnicodeEncoding UE = new UnicodeEncoding();
byte[] MessageBytes = UE.GetBytes(MessageString);
```

5) Создайте объект SHA1Managed для преобразования входного байтового массива в байтовый массив с хэшем:

```
SHA1Managed SHhash = new SHA1Managed();
byte[] HashValue;
HashValue = SHhash.ComputeHash(MessageBytes);
```

6) Выведите на экран значение хэша:

```
foreach (byte b in HashValue)
{
        Console.Write("{0} ", b);
};
```

Для приведенной строки значение хэша будет:

```
83 59 33 252 46 185 17 150 15 74 74 53 234 146 96 88 226 72 16 142
```

3.2 Проверка хэша

1) Выполните п.3.1.

2) Создайте строку с известным сообщением:

```
string MessageString = "Исходное сообщение";
```

3) Создайте объект *UnicodeEncoding* и преобразуйте исходную строку в байтовый массив:

```
UnicodeEncoding UE = new UnicodeEncoding();
byte[] MessageBytes = UE.GetBytes(MessageString);
```

4) Создайте объект SHA1Managed для преобразования входного байтового массива в байтовый массив с хэшем:

```
SHA1Managed SHhash = new SHA1Managed();
byte[] HashValue;
HashValue = SHhash.ComputeHash(MessageBytes);
```

5) Создайте битовый массив из известного хэша:

```
byte[] SentHashValue =
{83,59,33,252,46,185,17,150,15,74,74,53,234,146,96,88,226,72,16,142};
```

7) Проведите побайтовое сравнение между известным хэшем и новым хэшем:

```
bool Same = true;
for (int x = 0; x < SentHashValue.Length; x++)
{
        if (SentHashValue[x] != CompareHashValue[x])
        {
                Same = false;
        }
}
```

3.3 Задание к лабораторной работе

Необходимо разработать программу, хэширующую jpeg файл. Расположение файла определяется пользователем на локальном компьютере. Значение хэша должны сохраняться в текстовый файл по указанному пользователем расположению. Программа должна сопоставлять предъявленный файл и значение хэша из текстового файла.

Для создания хэша используйте один из следующих классов:

- *MD5CryptoServiceProvider;*
- *SHA1Managed;*
- *SHA256Managed;*
- *SHA384Managed;*
- *SHA512Managed.*

Контрольные вопросы

4. Для чего используется хэширование?
5. Приведите примеры использования хэширования на практике?
6. Опишите процесс хэширования по алгоритму MD5.
7. Что такое коллизии хэш-функции? С чем они связаны?
8. Оцените затраты времени при хешировании и последующей проверке файлов размером ≈ 1 КБ, ≈ 10 КБ, ≈ 100 КБ, ≈ 1 МБ, ≈ 10 МБ, ≈ 100 МБ.

7. Образовательные технологии

При изучении дисциплины «Информационная безопасность и защита информации» предусматривается лекционное изложение курса, работа с учебниками и учебными пособиями, лабораторные работы и консультации по курсу.

Перед посещением лекции студенту рекомендуется прочесть основную и дополнительную литературу к данной лекции. Во время лекции рекомендуется делать запись и дополнительные пометки к тексту лекции.

Более глубокому усвоению знаний и умений способствует выполнение практических и лабораторных занятий. Перед выполнением лабораторных занятий следует повторить материал соответствующих лекций. Во время лабораторных занятий

выполнять учебные задания с максимальной активностью. По окончании лабораторного занятия оформить отчет по выполненным заданиям.

При проведении лекционных занятий используются следующие образовательные технологии:

1. Лекция классическая – систематическое, последовательное, монологическое изложение учебного материала.

2. Лекция-визуализация – передача информации посредством схем, таблиц, рисунков, видеоматериалов, проводится по ключевым темам с комментариями.

При проведении практических и лабораторных занятий используются следующие образовательные технологии:

1. Работа в команде – совместная деятельность студентов в группе под руководством лидера, направленная на решение общей задачи путем творческого сложения результатов индивидуальной работы членов команды с делением полномочий и ответственности.

2. Проблемное обучение – стимулирование студентов к самостоятельному приобретению знаний, необходимых для решения конкретной проблемы.

3. Контекстное обучение – мотивация студентов к усвоению знаний путем выявления связей между конкретным знанием и его применением.

4. Обучение на основе опыта – активизация познавательной деятельности студента за счет ассоциации и собственного опыта с предметом изучения.

5. Опережающая самостоятельная работа – изучение студентами нового материала до его изучения в ходе аудиторных занятий.

8. Методические указания по освоению дисциплины

Содержание учебной программы дисциплины «Информационная безопасность и защита информации» реализуется посредством лекций и лабораторных занятий, а также самостоятельной работы студентов.

Познавательная активность студентов на лабораторных занятиях обеспечивается рациональным сочетанием словесных, наглядных и практических методов с элементами обучения на основе опыта, работой с различными информационными источниками, решением познавательных и практикоориентированных задач.

Дисциплина состоит из 3 разделов.

Раздел 1. Основные понятия и положения защиты информации (ЗИ) в информационно-вычислительных системах.

Лекции – 4ч., лабораторные занятия – 8ч., КСР – 1 ч., СРС – 20 ч.

Предмет и объект ЗИ. Угрозы безопасности информации. Понятие угрозы безопасности. Классификация угроз информационной безопасности. Классификация злоумышленников. Основные методы реализации угроз информационной безопасности. Причины, виды и каналы утечки информации.

В ходе лекционного курса проводится систематическое изложение современных научных и технических материалов, освещение основных проблем, связанных с информационной безопасностью и защитой информации. Освещается законодательная база РФ в области информационной безопасности.

В тетради для конспектирования лекций необходимо иметь поля, где по ходу конспектирования студент делает необходимые пометки. Записи должны быть избирательными, полностью следует записывать только определения. В конспектах рекомендуется применять сокращения слов, что ускоряет запись.

Для проведения лабораторных работ №1 и №2 используется среда разработки Microsoft Visual Studio 2010 Express Edition и язык программирования C#. Рекомендуется изучить отечественные стандарты симметричного и ассиметричного шифрования.

Раздел 2. Методы и средства защиты информации в информационно-вычислительных системах.

Лекции – 6ч., лабораторные занятия – 8ч., КСР – 1 ч., СРС – 18 ч.

Правовые и организационные методы защиты информации. Стандарты и спецификации в области информационной безопасности. Административный уровень информационной безопасности. Криптографическая защита информации.

Для максимального усвоения дисциплины рекомендуется изложение лекционного материала с элементами обсуждения. Рекомендуется изучить отечественные стандарты хеширования и ЭЦП.

Для проведения лабораторных работ №3 и №4 используется среда разработки Microsoft Visual Studio 2010 Express Edition и язык программирования C#.

Раздел 3. Защита компьютерной информации в информационно-вычислительных сетях.

Лекции – 4ч., лабораторные занятия – 8ч., КСР – 1 ч., СРС – 20 ч.

Модели безопасности. Системы защиты программного обеспечения. Защита информации в корпоративных сетях. Защита от информационных инфекций.

Рекомендуемые виды самостоятельных работ: конспектирование, реферирование, составление опорных схем, составление аннотированных каталогов и аналитических обзоров информационных ресурсов, творческая работа по созданию графических изображений.

Для проведения лабораторной работы №5 используется СУБД Microsoft SQL Server 2012 Express и среда Microsoft SQL Management Studio 2012.

Выполнение всего цикла лабораторных занятий является обязательным для получения допуска студента к зачету. В случае пропуска занятий по уважительной причине пропущенное занятие подлежит отработке. В ходе лабораторных занятий студент под руководством преподавателя выполняет комплекс лабораторных заданий, позволяющих закрепить лекционный материал по изучаемой теме.

9. Материально-техническое обеспечение дисциплины

При изучении дисциплины используются специальные помещения представляющие собой учебные аудитории университета для проведения занятий лекционного типа, занятий семинарского типа, курсового проектирования (выполнения курсовых работ), групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации, а также помещения для самостоятельной работы и помещения для хранения и профилактического обслуживания учебного оборудования. Специальные помещения укомплектованы специализированной мебелью и техническими средствами обучения, служащими для представления учебной информации большой аудитории. Для проведения занятий лекционного типа используются наборы стационарного и/или переносного демонстрационного оборудования.

Распределение аудиторного фонда соответствует утверждённому расписанию занятий, при указании шифра учебного помещения префикс соответствует номеру учебных корпусов, расположенных по адресам: префикс 1-9 – 450008, г. Уфа, ул. Карла Маркса, 12; префикс 10 – 450005, г. Уфа, ул. Мингажева, 158/2; префикс 11 – 450076, г. Уфа, ул. Аксакова, 94.

Для проведения лабораторного практикума, курсового проектирования (выполнения курсовых работ), групповых и индивидуальных консультаций, текущего контроля предназначены учебные лаборатории кафедры с наличием вычислительного и телекоммуникационного оборудования и программных средств, необходимых для реализации ОПОП ВО и обеспечения физического доступа к информационным сетям, используемым в учебном процессе (аудитории 6-301, 6-303, 6-309).

Для проведения занятий лекционного типа (консультаций, текущего контроля и промежуточной аттестации) с набором демонстрационного оборудования и учебно-наглядных пособий (в том числе мобильных) предназначены ауд. 3-305.

Для лабораторных работ студентов (укомплектованная специализированной мебелью, техническими средствами обучения и лабораторным оборудованием) предназначены ауд. 6-301, ауд. 6-303.

Для самостоятельной работы обучающихся предназначены ауд. 6-301, 6-303, 6-309 (в свободное от занятий время), оснащенные компьютерной техникой с возможностью подключения к сети «Интернет» и обеспечением доступа в электронную информационно-образовательную среду организации.

Для профилактического обслуживания и хранения учебного оборудования используются аудитория 6-302а.

Используется лицензионное программное обеспечение:

Microsoft Office, Microsoft Windows, Dr.Web Desktop Security Suite, Microsoft Visual Studio 2008, Microsoft SQL Server.

10. Адаптация рабочей программы для лиц с ОВЗ

Адаптированная программа разрабатывается при наличии заявления со стороны обучающегося (родителей, законных представителей) и медицинских показаний (рекомендациями психолого-медико-педагогической комиссии). Для инвалидов адаптированная образовательная программа разрабатывается в соответствии с индивидуальной программой реабилитации.

Лист согласования рабочей программы дисциплины

ЛИСТ
согласования рабочей программы

Направление подготовки: 09.03.02 Информационные системы и технологии
код и наименование

Направленность подготовки (профиль) Геоинформационные системы

Дисциплина: Информационная безопасность и защита информации

Учебный год 2015/2016

РЕКОМЕНДОВАНА заседанием кафедры геоинформационных систем
наименование кафедры

протокол № 10 от "28" 06 2015 г.

Заведующий кафедрой _____ Христодуло О. И. 26.08.16
подпись расшифровка подписи

Исполнители:
доцент каф.ГИС _____ Абдуллин А.Х. 26.08.16
должность подпись расшифровка подписи

СОГЛАСОВАНО:

Заведующий кафедрой[1]
ГИС _____ Христодуло О. И.
наименование кафедры личная подпись расшифровка подписи дата

Председатель НМС по УГСН 09.00.00 Информатика и вычислительная техника
протокол № 3 от "28" 08 2015 г.

_____ Фрид А. И.
личная подпись расшифровка подписи

Библиотека _____ Евсеевская Л.В. 28.08.16
личная подпись расшифровка подписи дата

Декан факультета ИРТ _____ Юсупова Н.И. 28.08.16
личная подпись расшифровка подписи дата

Рабочая программа зарегистрирована в ООПМА и внесена в электронную базу данных

Начальник _____ Шерышева 28.08.16
личная подпись расшифровка подписи дата

[1] Согласование осуществляется с выпускающими кафедрами (для рабочих программ, подготовленных на кафедрах, обеспечивающих подготовку для других направлений и специальностей)