|  |  |
| --- | --- |
|  | **Министерство образования и науки Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н. Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н. Э. Баумана)** |

ФАКУЛЬТЕТ \_\_\_\_\_\_\_\_\_\_\_ИУ «Информатика и системы управления» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_\_\_\_\_\_\_\_\_\_ИУ8 «Информационная безопасность» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К КУРСОВОЙ РАБОТЕ***

***НА ТЕМУ:***

***Анализ возможностей разработки\_\_\_\_\_\_***

***приложений, использующих крипто-***

***графические функции на основе библиотеки \_\_Java****\_\_\_\_\_\_\_\_\_* ***Cryptography Extension.\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_l***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

Студент \_\_\_\_ИУ8-94\_\_\_ **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** \_\_\_\_\_ А. Г. Филинов\_

(Группа) (Подпись, дата) (И. О. Фамилия)

Руководитель курсового проекта **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** \_\_\_\_ Д. Е. Родионов \_

(Подпись, дата) (И. О. Фамилия)

Консультант **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Подпись, дата) (И. О. Фамилия)

*2017 г.*

**Министерство образования и науки Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технический университет имени Н. Э. Баумана**

**(национальный исследовательский университет)»**

**(МГТУ им. Н. Э. Баумана)**

УТВЕРЖДАЮ

Заведующий кафедрой \_\_ИУ8\_\_

(Индекс)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_ \_М. А. Басараб\_\_\_

(И. О. Фамилия)

« » 2017 г.

**ЗАДАНИЕ**

**на выполнение курсовой работы**

по дисциплине \_\_\_\_\_\_\_\_\_\_\_Безопасность систем баз данных\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент группы \_\_\_\_\_ИУ8-94\_\_\_\_\_\_

Филинов Анатолий Григорьевич 1

(Фамилия, имя, отчество)

Тема курсовой работы Анализ возможностей разработки приложений,\_использующих криптографические функции на основе библиотеки Java Cryptography Extension.\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Направленность КР (учебная, исследовательская, практическая, производственная, др.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_учебная\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Источник тематики (кафедра, предприятие, НИР) \_\_\_\_\_\_\_\_\_\_\_\_\_\_кафедра\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

График выполнения КР: 25% к 4 нед., 50% к 8 нед., 75% к 12 нед., 100% к 16 нед.

***Техническое задание*** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Анализ возможностей Java Cryptography Extension, разработка архитектуры и реализация приложения. 1

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***Оформление курсовой работы:***

Расчетно-пояснительная записка на листах формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата выдачи задания « 17 » сентября 2017 г.

**Руководитель курсовой работы**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_ Д. Е. Родионов\_\_\_

(Подпись, дата) (И.О.Фамилия)

**Студент** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А. Г. Филинов\_\_\_

(Подпись, дата) (И.О.Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

Оглавление

[Введение 4](#_Toc501147908)

[Приложение 7](#_Toc501147909)

Введение

Прежде чем приступить к изучению возможностей фреймворка Java Cryptography Extension необходимо в общих чертах выяснить, во-первых для каких целей данный фреймворк был создан, во-вторых понять из каких частей он состоит.

Java Cryptography Extension - официально выпущенное стандартное расширение для платформы Java и часть Java Cryptography Architecture(JCA). Представляет собой набор пакетов, который является [фреймворк](https://ru.wikipedia.org/wiki/%D0%A4%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA)ом, реализующим такие криптографические задачи, как [шифрование](https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) и дешифрование данных, генерация и проверка на подлинность ключей управления, а также реализацию для Message Authentication Code (MAC) алгоритмов.

Криптографическое расширение Java основано на том же, что и криптографическая архитектура Java (JCA) и рассматривается как часть JCA. Дело в том, что американские законы запрещают экспорт некоторых видов криптографического программного обеспечения (в частности симметричное шифрование и выработку общего ключевого материала) за пределы США и Канады или разрешают экспорт с урезанными ключами. Стандартные классы JCA содержат только [хеш-функции](https://ru.wikipedia.org/wiki/%D0%A5%D0%B5%D1%88-%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F), генераторы ключей и другие функции, которые не попадают под данное ограничение и могут быть спокойно экспортированы в составе платформы Java 2. Однако сильные алгоритмы шифрования, попадающие под экспортные ограничения на криптографию в США, должны быть получены из других источников, поэтому их поставляю в виде отдельного продукта - JCE.

JCA тесно интегрировано в ядро Java API и предоставляет самые базовые криптографические возможности, JCE – надмножество JCA.

Данные особенности имели место до JDK 1.4, В последних версиях библиотеки JDK JCE входит в состав JDK, поэтому различие между JCA и JCE становится менее очевидным. Так как JCE использует ту же архитектуру что и JCA, JCE должно рассматриваться как часть JCA.

JCA состоит из двух компонентов:

* фреймворка, который определяет и поддерживает криптографические сервисы для которых провайдеры поддержимают имплементации. Этот фреймворк включает в себя такие пакеты как java.security, javax.crypto, javax.crypto.spec, andjavax.crypto.interfaces.
* Сами провайдеры, такие как Sun, SunRsaSign, SunJCE. Они содержат фактические криптографические реализации.

Платформа Java придает особое значение безопасности, включая языковую безопасность, криптографии, инфраструктуре на основе открытого ключа, аутентификации, безопасной коммуникации и контролю доступа.

JCA – важная часть платформы, которая содержит в себе архитектуру «провайдера» и множество API для цифровых подписей, сертификатов и подтверждения сертификатов, шифрования (на основе симметричных/ассиметричных, блочных/потоковых алгоритмов), генерации ключей и их управления, а также безопасных генераторов случайных чисел. Данные API позволяют разработчикам легко интегрировать безопасность в код своих приложений. Архитектура была разработана на основе следующих принципов:

Независимость реализации: приложения не должны реализовывать алгоритмы шифрования. Они должны запросить сервисы безопасности у платформы Java. Сервисы, отвечающие за безопасность, реализованы в провайдерах, которые встроены в платформу Java через стандартный интерфейс. Приложение может использовать несколько независимых провайдеров для безопасного функционирования.

Интегрируемость реализации: провайдеры интегрируемы между приложениями. В частности, приложение не привязано к определенному провайдеру и провайдер не привязан к определенному приложению.

Расширяемость алгоритма: платформа Java включает в себя много встроенных провайдеров, которые реализуют базовое множество сервисов безопасности, которые широко используются в наши дни. Несмотря на это, некоторые приложение могут опираться на стандарты, которые ещё не реализованы или на проприетарные сервисы. Платформа Java поддерживает установку самодельных провайдеров, которые реализуют такие сервисы.

Алгоритмическая независимость достигается заданием типов криптографических “движков”(сервисов) и определением классов, которые предоставляют функциональность этих движков. Под это определение, например, подходят следующие классы: the [MessageDigest](https://docs.oracle.com/javase/8/docs/technotes/guides/security/crypto/CryptoSpec.html#MessageDigest), [Signature](https://docs.oracle.com/javase/8/docs/technotes/guides/security/crypto/CryptoSpec.html#Signature), [KeyFactory](https://docs.oracle.com/javase/8/docs/technotes/guides/security/crypto/CryptoSpec.html#KeyFactory), [KeyPairGenerator](https://docs.oracle.com/javase/8/docs/technotes/guides/security/crypto/CryptoSpec.html#KeyPairGenerator), and [Cipher](https://docs.oracle.com/javase/8/docs/technotes/guides/security/crypto/CryptoSpec.html#Cipher) .

Независимость реализации достигается за счет использования так называемой архитектуры “провайдеров”. Термин Cryptographic Service Provider (CSP) относится к пакету или множеству пакетов, которые реализуют один или несколько криптографических сервисов, такие как алгоритмы цифровой подписи, сервисы обмена ключами. Программа может просто потребовать определенный тип объекта(такой как Signature) реализующий определенный сервис(например DSA) и получить имплементацию из установленных провайдеров. При желании программа может использовать определенный тип провайдера.

Совместимость позволяет различным реализациям работать друг с другом, используя общие ключи или проверку подписей друг друга. Это значит, например, что для одинаковых алгоритмов, ключ, сгенерированный одним провайдером, может быть использован другим, а также подпись сгенерированная одним провайдером, сможет быть проверена другим.

Расширяемость алгоритма означает, что новые алгоритмы могут добавляться в библиотеку с легкостью.

Среди различных реализаций JCE можно выделить пакеты [Bouncy Castle,](https://ru.wikipedia.org/wiki/Bouncy_Castle) Cryptix JCE, и пакет IAIK JCE.

Будем рассматривать пакет [Bouncy Castle](https://ru.wikipedia.org/wiki/Bouncy_Castle). В основе архитектуры лежит набор низкоуровневых API, которые реализуют все криптографические алгоритмы. Причина, по которой используется именно низкоуровневое API, заключается в том, что в некоторых устройствах, работающих на платформе JavaME, очень ограничены ресурсы памяти, либо когда доступ к библиотеке [JCE](https://ru.wikipedia.org/wiki/Java_Cryptography_Extension) невозможен (такая ситуация может возникнуть, например, при использовании [апплетов](https://ru.wikipedia.org/wiki/%D0%90%D0%BF%D0%BF%D0%BB%D0%B5%D1%82)).

[Криптопровайдер](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%BF%D1%80%D0%BE%D0%B2%D0%B0%D0%B9%D0%B4%D0%B5%D1%80), совместимый с JCE, построен на низкоуровневом API. Таким образом, исходный код криптопровайдера JCE может служить примером того, как решить многие «насущные» проблемы криптографии, используя низкоуровневый API.

JCE\_Application