На стороне сети этот пароль проверяется с помощью специального серверного ПО.

Второй вариант реализован в продуктах компании RSA Security.

RSA SecurID for Microsoft Windows – это программное решение для проверки подлинности пользователей в вычислительных средах Microsoft Windows.

С точки зрения конечного пользователя разница между обычной процедурой регистрации в системе Windows и аутентификацией в системе RSA SecurID состоит лишь в том, что вместо стандартного пароля требуется ввести составной код доступа, состоящий из личного PIN-кода и комбинации цифр, которая в данный момент отображается на экране жетона-аутентификатора. Затем этот код доступа отсылается серверу RSA Authentication Manager, который и выполняет проверку подлинности пользователя.

RSA SecurID for Microsoft Windows обеспечивает прозрачную интеграцию с контролерами доменов Windows и каталогами Active Directory. База данных пользователей и групп сервера аутентификации RSA Authentication Manager синхронизирована с каталогом Active Directory. Поэтому, когда пользователь успешно проходит аутентификацию, сервер RSA Authentication Manager отправляет его пароль клиентской системе. Затем этот пароль отсылается контролеру домена для завершения аутентификации.

Отличие между этими двумя технологиями заключается в том, что разовый пароль в RSA SecurID изменяется через заранее заданные промежутки времени, а в продукте eToken NG смена разового пароля производится по нажатию кнопки (т. е. по мере надобности).

Вывод

Таким образом, рассмотрев различные технологии аппаратно-программной и парольной аутентификации, можно сделать вывод, что в дальнейшем по мере роста вычислительных мощностей все более востребованным будет именно применение двухфакторной аутентификации, что позволит избежать человеческих ошибок, связанных с применением слабых паролей, и ужесточить требования к парольной аутентификации.

Поступила 29.03.2006

УДК 681.03

Спирягин М.И., Спирягин В.И., Белозеров Е.В., Клюев С.А., Поляков А.С.

ПОВЫШЕНИЕ БЕЗОПАСНОСТИ КОМПЬЮТЕРНЫХ СИСТЕМ ПУТЕМ ИСПОЛЬЗОВАНИЯ JAVA CARD ТЕХНОЛОГИИ

Постановка проблемы

Для обеспечения более эффективной защиты информации необходимо обеспечить аутентификацию пользователя, на уровне одного документа (студенческий, читательский билет) для доступа к электронным ресурсам (библиотека, комплект учебно-методической документации и т.д.), услуг Интернет, а также обеспечение целостности информации в процессе передачи и приема данных через Интернет. Проведенный анализ показывает, что одним из наиболее рациональных средств обеспечения безопасности при аутентификации пользователей является использование смарт-карт.

Основная часть

Смарт-карта является на сегодняшний день одним из самых маленьких вычислительных устройств без источника питания, дисплея и клавиатуры. Тем не менее существует ряд задач, которые решают с помощью смарт-карт. Существуют два основных типа смарт-карт: *интеллектуальные карты*, содержащие микропроцессор, позволяющие читать и записывать информацию, а также производить вычисления, и *карты памяти*, у которых нет микропроцессора и которые могут быть использованы только для хранения информации. Доступ к информации на картах памяти защищен.

Все смарт-карты содержат три типа памяти: постоянную неизменяемую (ROM), постоянную изменяемую (или программируемую) (EEPROM) и оперативную память (RAM).

Для доступа к смарт-карте используется устройство кард-ридер. Оно служит источником питания для карты и устанавливает соединение для передачи данных.

Стандарт ISO-7816 определяет структуру пакетов, которыми обмениваются смарт-карты через установленное кард-ридером соединение. Такие пакеты называются APDU (Application Protocol Data Units) и содержат либо команду, посылаемую смарт-карте, либо ответ, который карта посылает обратно после обработки сообщения.

Обмен сообщениями всегда происходит по схеме запрос-ответ и всегда инициируется кард-ридером.

Важным является подбор рационального типа Java Card для работы со смарт-картами для обеспечения:

- единой среды выполнения;
- использования языка программирования высокого уровня;
- встроенной поддержки шифрования и защиты информации;
- возможности использования общего клиентского программного обеспечения на большинстве современных ОС.

Java Card – это смарт-карта, на которой может выполняться программа, написанная на языке Java. Виртуальная машина Java Card (JCVM), построенная на основе интегральных схем, скрывает от разработчика приложений для JC реализацию конкретной карты производителем и предоставляет общий интерфейс программирования. Приложения для Java Card называют *апплетами*. На одной карте может быть установлено одновременно несколько апплетов, каждый из которых определен уникальным идентификатором *AID*.

Поскольку ресурсы Java Card серьезно ограничены, JCVM поддерживает не все возможности языка Java.

Виртуальная Java Card-машина состоит из двух частей, одна из которых работает непосредственно на самой карте, а другая – на хост-машине вне карты. Вне Java Card исполняются такие процессы, как загрузка классов, проверка байт-кода, оптимизация и т.п. Критерием такого разделения является наличие или отсутствие ограничений на выполнение процесса во время работы программы.

Java Card API позволяет программам, написанным на языке Java, исполняться на интеллектуальных картах и других устройствах с ограниченными ресурсами.

Апплет Java Card

Апплет Java Card представляет собой класс Java, наследованный от класса javacard.framework.Applet. Для обработки сообщений APDU достаточно переопределить метод process(), а для создания экземпляра объекта – статический метод install().

С точки зрения программиста, разработка апплета Java Card практически ничем не отличается от разработки приложения на Java для ПК или для мобильного телефона. Основное отличие от обычной виртуальной машины Java заключается в том, что JCVM представляет собой набор из двух компонент: то, что находится на самой карте (интерпретатор), и то, что находится вне карты (конвертор). Рабочий цикл может быть проиллюстрирован следующим рис. 1:

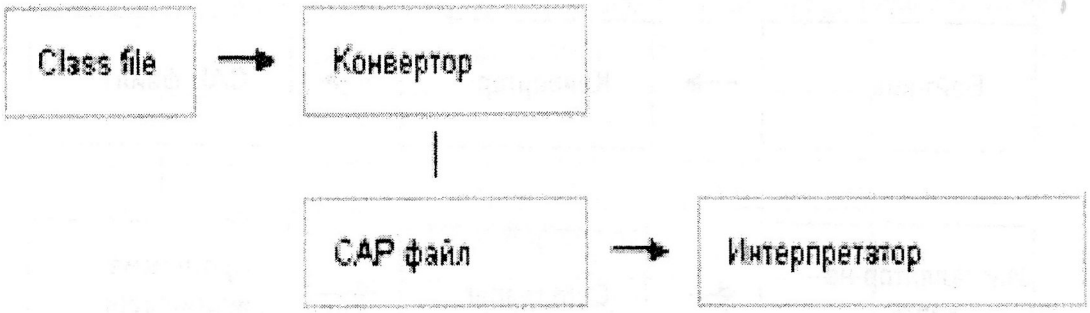

Рис. 1.

Конвертор существует вне смарт-карты. Это Java-приложение, работающее на пользовательском компьютере. Конвертор переводит обычные .class файлы (стандартный байт-код) в специальный формат CAP (converted applet). CAP файл загружается в смарт-карту и далее исполняется интерпретатором.

Все опции языка программирования, которые не поддерживаются спецификацией Java Card, удаляются при конвертировании. Байт-код, определяемый CAP, построен на основе стандартного Java байт-кода и оптимизирован для исполнения в условиях ограниченных ресурсов смарт-карты.

Кроме CAP-файла, конвертор генерирует экспортные файлы. Они не загружаются непосредственно в смарт-карту и, следовательно, не используются интерпретатором. Их назначение – это поддержка процесса верификации. В некотором смысле они подобны файлам заголовков в языке С. Экспортные файлы используются для написания клиентской части, т.е. кода программы, которая будет обращаться к апплету, установленному на Java Card.

Конвертор (как одна из частей JCVM) выполняет свою часть общего процесса исполнения приложения. В частности, конвертор:

- проверяет корректность байт-кода;
- проверяет нарушения стандарта Java Card (подмножества Java);
- выполняет инициализацию статических переменных;
- разрешает символические ссылки для классов, методов и полей класса и переводит их в более компактную форму, лучше приспособленную к ограничениям среды смарт-карты;
- оптимизирует байт-код;
- размещает память и создает структуры JVM для представления классов.

Как и файл заголовков в языке С (или как определение интерфейса в языке Java), экспортный файл не содержит реализации. Это дает возможность свободного распространения экспортных файлов, без раскрытия деталей реализации.

Интерпретатор обеспечивает поддержку языка Java (байт-кода) во время исполнения. Собственно интерпретация (в отличие от компиляции) обеспечивает независимость кода апплета от аппаратуры. Интерпретатор выполняет, например, следующие задачи:

- исполнение байт-кода;
- управление памятью и созданием объектов;
- обеспечение безопасности.

Интерпретатор не загружает сам CAP-файлы. Интерпретатор ответственен только за исполнение кода из компрессированных файлов. Вся загрузка и инсталяция вынесена в отдельную компоненту – инсталятор. Общая схема следующая (рис. 2):

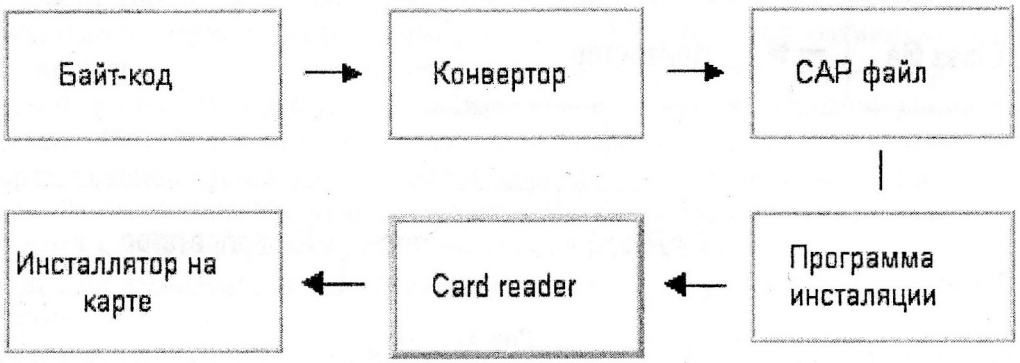

Рис.2

Здесь card reader – это устройство чтения карт. В английской литературе используется аббревиатура CAD (card acceptance device).

Таким образом, инсталятор размещается непосредственно на смарт-карте. Он взаимодействует с программой инсталяции на пользовательской рабочей станции.

Программа инсталяции передает файл в CAP-формате инсталятору на карте. Последний записывает двоичный файл в память смарт-карты, связывает его с другими классами, возможно размещенными на карте, создает и инициализирует структуры данных, используемые при выполнении апплета. Подобное разделение позволяет уменьшить размер собственно интепретатора, что, естественно, очень важно в условиях ограничения ресурсов.

Рассмотрим среду выполнения JCRE (Java Card Runtime Environment). JCRE представляет собой набор системных компонент, исполняемых на самой карте. JCRE управляет ресурсами, поддерживает передачу данных и исполнение апплета, а также обеспечивает систему безопасности как карты, так и апплета. Другими словами, это есть операционная система смарт-карты.

Внутри карты, после загрузки на нее апплетов, управление передается среде выполнения, которая устанавливает апплет вызовом статического метода install(). Этот метод должен быть определен, и он должен создавать экземпляр объекта нашего класса апплета.

Среда выполнения предоставляет разработчику ряд классов, необходимых для работы Java Card. Среди них классы Applet, PIN, APDU, System, Util, а также классы для поддержки криптографии и безопасности данных. Последние представляют собой наиболее важную часть JCRE, т.к. смарт-карты в первую очередь предназначены именно для хранения личных данных пользователя, необходимых для безопасного доступа к защищенным ресурсам.

Клиентская часть

Под клиентской частью обычно понимают приложение, выполняемое на пользовательском компьютере, которое получает доступ к смарт-карте и обменивается с ней данными. Не имеет значения, на каком языке программирования написано такое приложение и с каким типом смарт-карт оно работает. Такое приложение должно использовать протокол для работы со смарт-картой, определенный в ISO-7816.

К счастью, существуют библиотеки, скрывающие от разработчиков реализацию таких рутинных процедур, как поиск и выбор кард-ридера, инициализация и завершение работы с картой и даже обмен APDU-сообщениями. Вместо чего предоставляют простой и удобный интерфейс для работы со смарт-картами.

Для Java приложений такой библиотекой является *Open Card Framework*. При использовании этой библиотеки схему работы с Java-картой можно представить в виде, представленном на рис. 3.

Java Card API обеспечивает набор пользовательских классов для программирования смарт-карт в соответствии с моделью ISO 7816. API содержит три базовых пакета и один

дополнительный пакет для криптографии. Базовые пакеты: java.lang, javacard.framework и

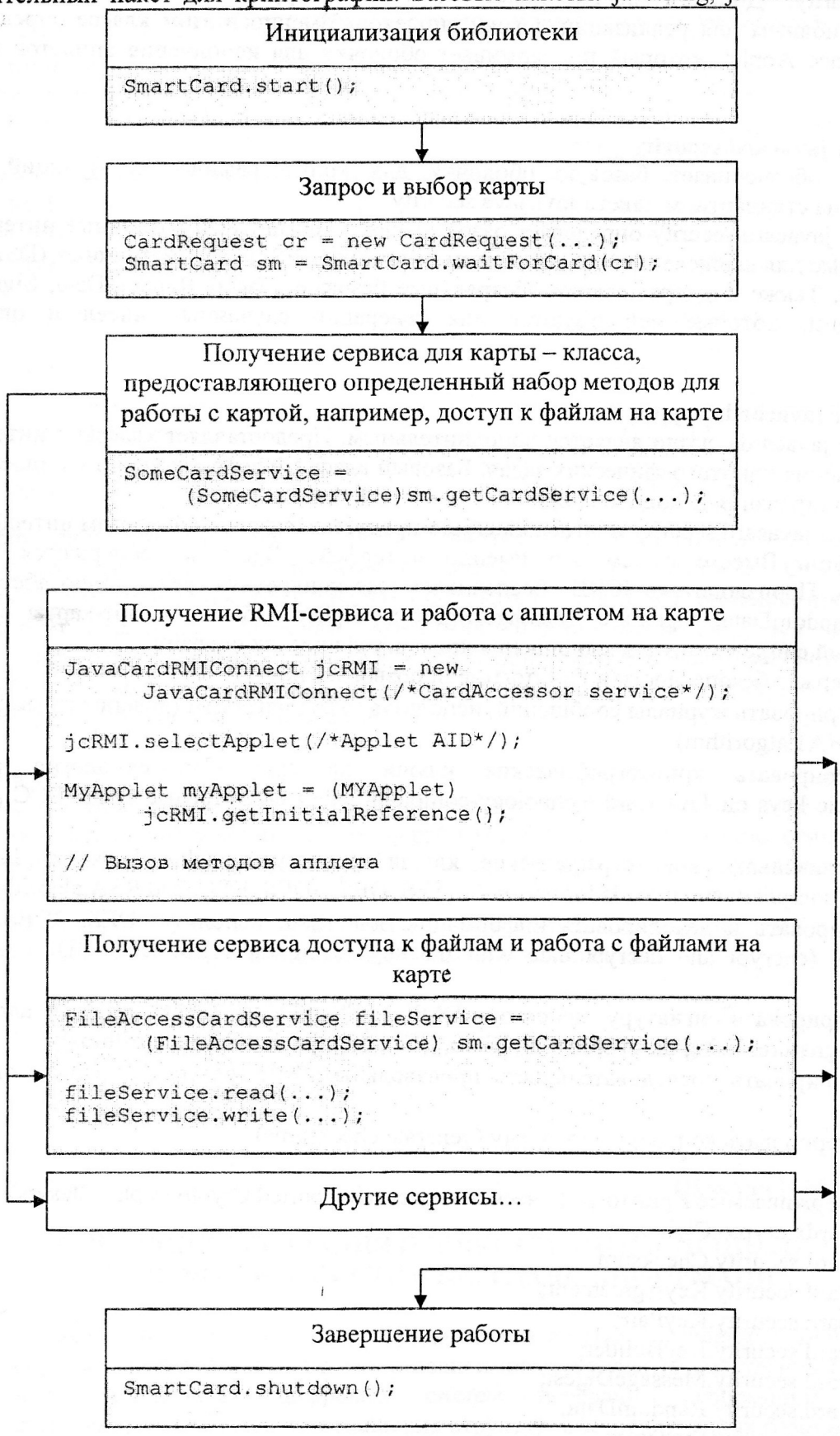

Рис.3. Схема работы с Java-картой

javacard.security. Дополнительный пакет – javacardx.crypto. Пакет javacard.framework является основным для реализации логики апплетов. Именно в этом классе определяется базовый класс Applet, который поддерживает оболочку для исполнения апплетов в среде JCRE.

Пакет javacard.security

Пакет обеспечивает базовую оболочку для криптографических функций. Пакет базируется на стандартном пакете Java java.security.

Пакет javacard.security определяет базовый класс keyBuilder и различные интерфейсы, используемые для вычисления ключей в симметричных (DES) и асимметричных (DSA, RSA) алгоритмах. Также поддерживаются абстрактные базовые классы RandomData, Signature и MessageDigest, которые используются для генерации случайных чисел и цифровых подписей.

Пакет javacardx.crypto

Пакет javacardx.crypto является дополнительным. Предоставляет классы и интерфейсы для выполнения криптографических задач. Базовый класс Cipher собственно и поддерживает функции кодирования/декодирования.

Пакеты javacard.security и javacardx.crypto предоставляют всем апплетам интерфейс для криптографии. Вместе с тем это именно интерфейс. Здесь не содержится никакой реализации. Производитель JCRE обеспечивает уже конкретную реализацию абстрактных классов RandomData, Signature, MessageDigest и Cipher. Обычно смарт-карты содержат специальный сопроцессор для выполнения криптографических операций.

Поддержка безопасности и криптографии позволяет нам:

- генерировать журналы сообщений, используя SHA1 алгоритм (generate message digests using the SHA1 algorithm);
- генерировать криптографические ключи на Java Card-технологии (generate cryptographic keys on Java Card technology-compliant smart cards for use in the ECC and RSA algorithms);
- устанавливать криптографические ключи (set cryptographic keys on Java Card technology-compliant smart cards for use in the AES, DES, 3DES, ECC, and RSA algorithms);
- кодировать и декодировать информацию ключами, используя DES, 3DES, и RSA алгоритмов (encrypt and decrypt data with the keys using the AES, DES, 3DES, and RSA algorithms);
- генерировать сигнатуру, используя DES, 3DES, ECC, or SHA and RSA алгоритмов (generate signatures using the AES, DES, 3DES, ECC, or SHA and RSA algorithms);
- генерировать последовательность произвольных байт (generate sequences of random bytes);
- генерировать контрольную сумму (generate checksums).

Поддерживаемые Криптографические классы (Supported Cryptography Classes):
javacardx.crypto.Cipher;
javacard.security.Checksum;
javacard.security.KeyAgreement;
javacard.security.KeyPair;
javacard.security.KeyBuilder;
javacard.security.MessageDigest;
javacard.security.RandomData;
javacard.security.Signature.

Криптографические алгоритмы, используемые для Java Secure Socket Extension

Криптографический алгоритм	Длина ключа(Bits)
RSA	2048 (authentication), 2048 (key exchange), 512(key exchange)
RC4	128, 128 (40 effective)
DES	64 (56 effective), 64 (40 effective)
Triple DES	192, (112 effective)
AES	256, 128
Diffie-Hellman	1024, 512
DSA	1024

Вышеприведенный алгоритм позволяет построить систему, лишенную подавляющего большинства недостатков, свойственных современным технологиям. Она исключает подделку и изменение как со стороны пользователя, так и со стороны проверяющего технического персонала. Подобная технология отлично масштабируется и позволяет построить централизованную систему в рамках предприятия. В то же время система открыта для доработки и может быть дополнена другими возможностями, например, статистическим анализом, распечаткой отчетов, сведений и др.

Выводы

Выполненное исследование позволяет сделать следующие выводы:

1.Проведен анализ основных аспектов работы с использованием технологии SmartCard.

2.Изучено использование аппаратного и программного обеспечения для работы с описанной технологией.

3.Предложен алгоритм, реализующий описанную технологию и позволяющий избавиться от недостатков, имеющихся в современных системах.

Поступила 07.03.2006

УДК 621.396.6

Шокало В.М., Цопа А.И.

КОНЦЕПЦИЯ СОЗДАНИЯ ОТЕЧЕСТВЕННЫХ СПЕЦИАЛЬНЫХ ЦИФРОВЫХ СИСТЕМ ПЕРЕДАЧИ ИНФОРМАЦИИ

Рост числа техногенных и природных катастроф, усиление терроризма и кибертерроризма предполагает наличие у независимого государства собственных разработок защищенных специальных цифровых систем передачи информации (СЦСПИ), предназначенных для технического оснащения структур, обеспечивающих безопасность всех сторон жизнедеятельности государства.