УДК 004.67

Разработка программного обеспечения для создания цифровой подписи на базе алгоритма RSA

Берсенев Александр Иванович
Ярославский государственный технический университет
студент

Никитенко Андрей Владимирович
Ярославский государственный технический университет
к.п.н., доцент кафедры «Информационные системы и технологии»

Аннотация
В статье рассматривается реализация приложения для создания электронной цифровой подписи на базе алгоритма RSA и описываются некоторые временные характеристики ее генерации и проверки в зависимости от размера подписываемого файла.
Ключевые слова: электронная цифровая подпись, RSA.

Development of software for create a digital signature based on the RSA algorithm

Bersenev Aleksandr Ivanovich
Yaroslavl State Technical University
student

Nikitenko Andrey Vladimirovich
Yaroslavl State Technical University
Candidate of pedagogical sciences, associate professor of the department «Information systems and technologies»

Abstract
The article discusses the implementation of an application for creating an electronic digital signature based on the RSA algorithm and describes some time characteristics of its generation and verification, depending on the size of the signed file.
Keywords: electronic digital signature, RSA.

В современном мире очевидно наблюдается тенденция перехода на системы электронного документооборота. Эти системы позволяют значительно сэкономить время на пересылке и поиске нужных документов. Но в тоже время такие документы никак не защищены от модификации и повреждений в процессе пересылки. Может сложиться ситуация, когда получателю придут совсем не те данные, что выслал отправитель. Одним из

способов решения данной проблемы является использование электронной цифровой подписи (далее ЭЦП). В статье описывается один из вариантов реализации приложения для создания ЭЦП на базе алгоритма RSA. Для этого сначала опишем структуру создания цифровой подписи на базе алгоритма RSA [1].

Первым шагом является генерация ключей для RSA: выбираются два простых числа p и q; вычисляется n = p * q; вычисляется функция Эйлера f(n) = (p–1)*(q–1); выбирается открытая экспонента e, таким образом, что e лежит в пределах от 1 до f(n) и взаимно просто с f(n); выбирается d, такое, что d*e=1 (mod f(n)), то есть d – мультипликативно обратное к числу e по модулю f(n). В результате пара (e, n) является открытым ключом, а пара (d, n) – закрытым.

Для следующего этапа необходим алгоритм хеширования. Хеш-функция представляет собой функцию, которая преобразовывает входные данные любой длины в строку фиксированной длины. На сегодняшний день существует множество алгоритмов хеширования: SHA-256, ГОСТ 34.11-94, MD5 и другие. В частности, для шифрования: вычисляется хеш-образ h открытого текста T; вычисляется цифровая подпись s по формуле $s=h^d \bmod n$ (где d – закрытый ключ, n – часть открытого ключа).

Проверка цифровой подписи включает в себя следующую последовательность действий: вычисление хеш-образа полученного сообщения h_1 по полученному сообщению T_1; получение хеш-образа из цифровой подписи по формуле $h=s^e \bmod n$ (e, n – открытый ключ); наконец, если $h_1=h$, можно сделать вывод, что сообщение не было изменено.

Далее перейдем непосредственно к описанию реализации программного обеспечения для создания ЭЦП на базе алгоритма RSA. Поскольку для RSA необходимы числа от 512 бит, а стандартные типы языка C# (и других C-подобных языков) не поддерживают работу с такими большими числами, для их хранения используется структура BigInteger. За генерацию ключей отвечает класс RSAKeyGenerator, в частности, были написаны все необходимые методы для генераций ключей. Для вычисления закрытой экспоненты d используется расширенный алгоритм Евклида.

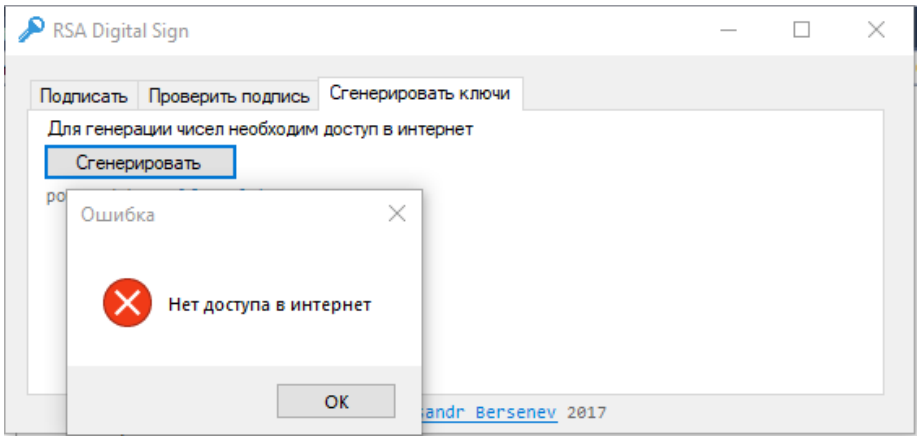

Рисунок 1 – Ответ программы, если генерация ключей происходит без доступа к Интернету

Для получения достаточно больших простых чисел используется API сервиса WolframAlpha [2]. Для генерации чисел на сервер Wolfram посылается запрос: «nextPrime[2^1023 + RandomInteger(2^1022)]», Wolfram возвращает случайное простое число 1024 бита. В случае ошибок (например, нет доступа в Интернет) происходит выброс исключения (рис.1).

Для хранения ключей используется класс Key, который для удобства сохраняется в виде файла xml (рис.2).

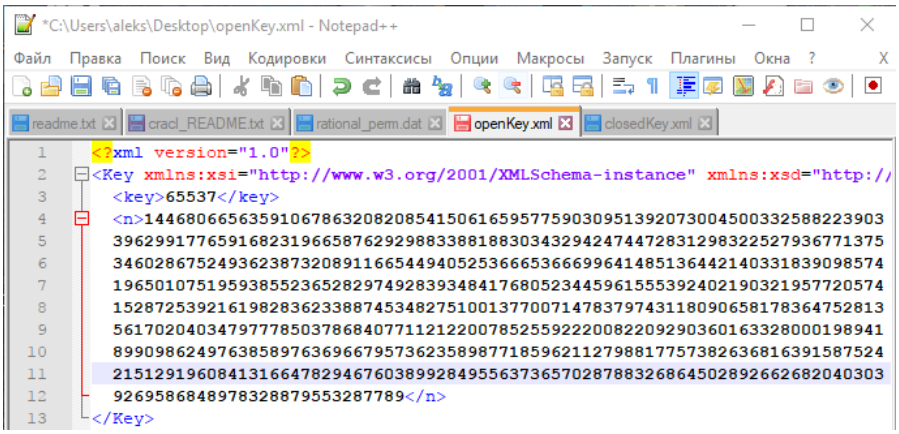

Рисунок 2 – Пример сгенерированного открытого ключа размерности 1024 бита в формате xml

Для хранения ЭЦП в виде строки используется класс DigitalSign. Для удобства считывания файла, в программе используется сохранение класса в виде бинарного файла. Класс DS служит для генерации цифровой подписи. Для этого используется алгоритм хеширования MD5 [3]. Это один из самых быстрых алгоритмов хеширования, но он имеет свои недостатки. Однако для использования программы для генерации цифровой подписи его достаточно.

Для быстрого возведения числа в степень по модулю используется алгоритм бинарного возведения в степень по модулю. Этот алгоритм работает за O log(n), что существенно позволяет ускорить генерацию цифровой подписи.

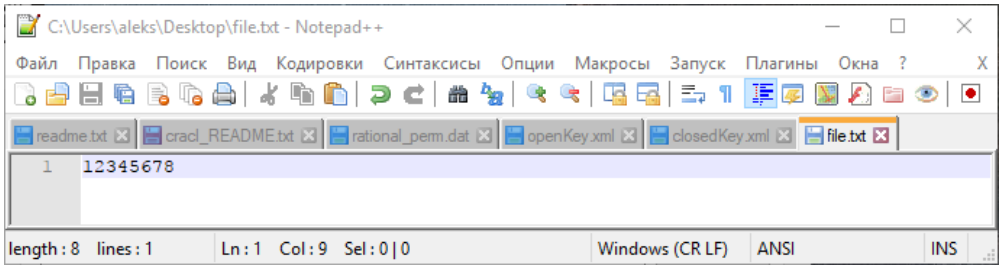

Рисунок 3 – Содержимое файла file.txt

Полная версия исходного кода программы размещена в репозитории [4]. Особенность разработанной программы заключается в возможности генерации ЭЦП для любого типа файлов, включая doc, xls, ppt, pdf, видео и аудио различных форматов. В частности, далее представлен вариант генерации ЭЦП текстового файла. Создадим произвольный файл file.txt (рис.

3). Запустим разработанную программу генерации ЭЦП. В открывшейся форме выберем вкладку «Сгенерировать ключи» и в ней нажмем кнопку «Сгенерировать».

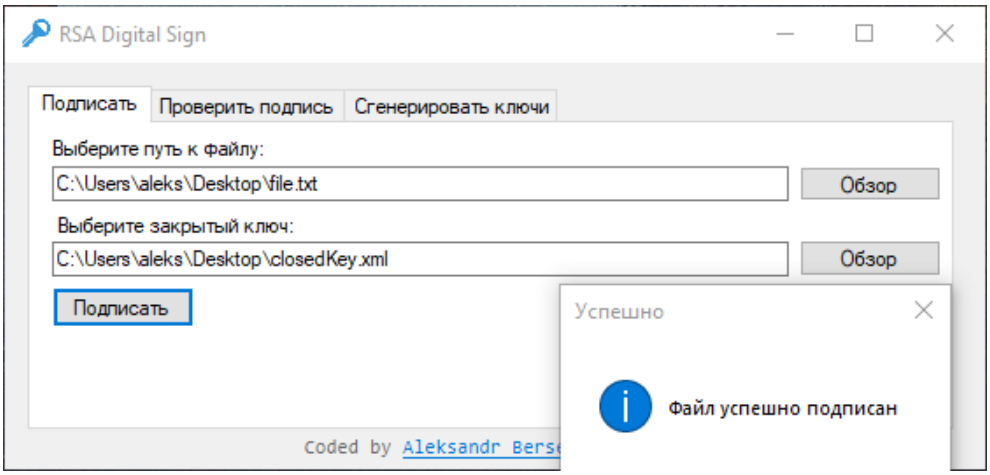

Рисунок 4 – Пример создания ЭЦП

При этом с помощью API сервиса WolframAlpha выполняется генерация открытого ключа размерности 1024 бита, который сохраняется в любом месте по выбору пользователя в файле формата xml. Параллельно с данной процедурой по алгоритму RSA производится и генерация закрытого ключа, который также сохраняется в файле формата xml. После этого открываем вкладку «Подписать», в которой с помощью кнопок «Обзор» выбираются: файл, который необходимо подписать, и сгенерированный закрытый ключ для такой подписи. Далее нажимается кнопка «Подписать» и, в случае успешного подписания, появляется всплывающее окно, которое сигнализирует об окончании данной процедуры (рис. 4). Непосредственно сама цифровая подпись оформляется в виде отдельного файла с расширением sign.

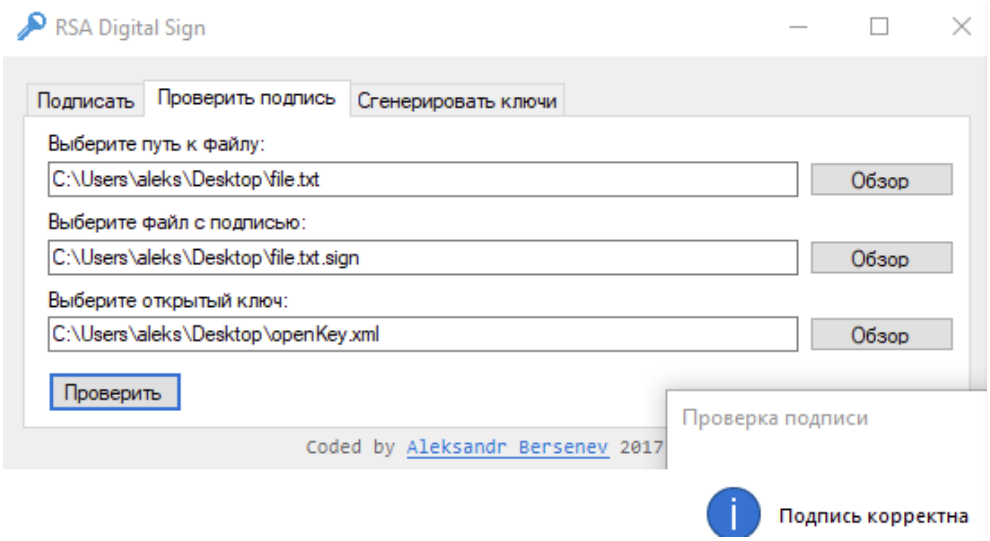

Рисунок 5 – Пример проверки ЭЦП

Проверим корректность подписания файла file.txt. Для этого откроем вкладку «Проверить подпись». В ней выбираются: подписанный файл, файл с цифровой подписью и открытый ключ для проверки подписи. Далее, после нажатия кнопки «Проверка» в случае корректности ЭЦП будет выведено всплывающее окно с соответствующим уведомлением (рис. 5).

В данном случае подпись корректна. Произвольным образом изменим file.txt, и снова проверим корректность подписи этого файла (рис. 6).

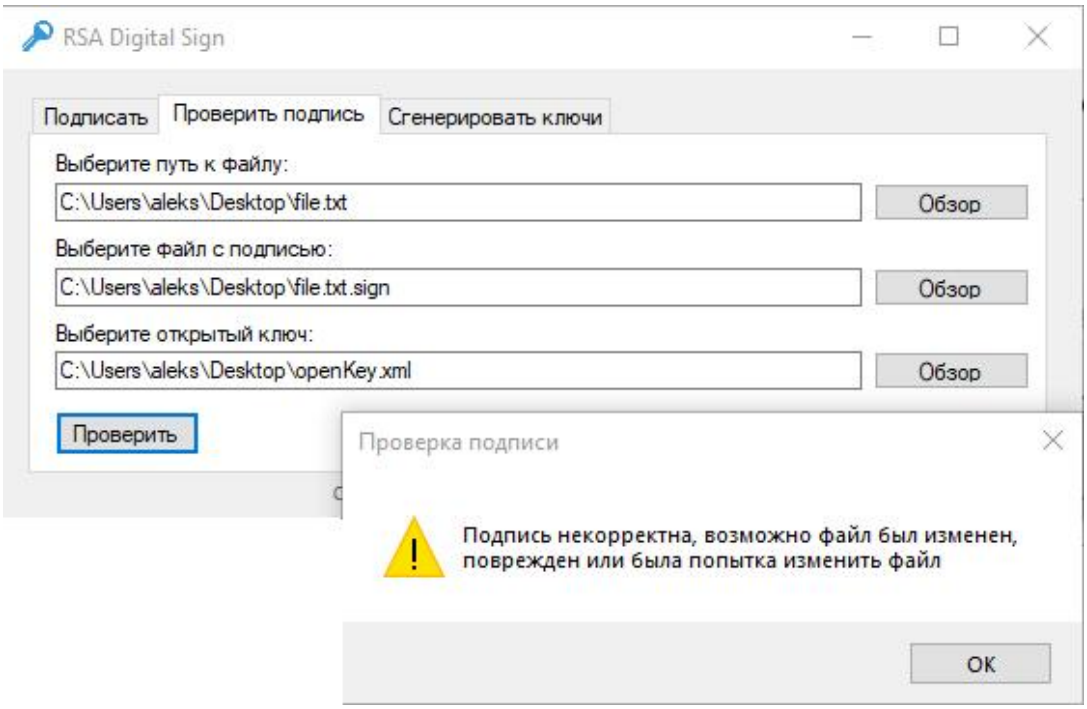

Рисунок 6 – Проверка корректности ЭЦП измененного файла

Как видно по рисунку 6, подпись некорректна, что может быть следствием того, что файл был изменен, как в нашем случае, или поврежден. Более того, разработанная программа для некоторых типов файлов, таких как файлы Microsoft Word, позволяет определить также попытку изменения файла. Дело в том, что в подобных файлах хранится не только информация, которая была введена пользователем, а также служебные данные, записанные самой MS Word. Поэтому даже откат изменений не сможет восстановить файл до первоначального состояния и подпись не будет соответствовать исходному файлу.

Таблица 1 – Время генерации ЭЦП в зависимости от размера подписываемого файла

Генерация ключа 1024 бита, мс	5700		
Размер подписываемого файла	5 Кбайт	55 байт	4,3 Гбайта
Подписывание файла, мс	367	622	24222
Проверка подписи файла, мс	94	623	39969

В заключении приведем некоторые временные характеристики генерации ЭЦП и ее проверки в зависимости от размера подписываемого файла. В частности, программа была апробирована на компьютере с процессором AMD FX6100 3,3GHz, оперативной памятью 12 ГБ, операционной системой Windows 10 x64. Результаты работы представлены в таблице 1.

Библиографический список

1. Rivest R., Shamir A., Adleman L. A method for obtaining digital signatures and public-key cryptosystems // Commun. ACM. New York City: ACM, 1978. Т. 21, №. 2. С. 120–126.
2. Wolframalpha [Электронный ресурс]. URL: https:// products.wolframalpha.com/api/, свободный (15.02.2018).
3. Rivest R. RFC 1321, The MD5 Message-Digest Algorithm: The MD5 Message-Digest Algorithm // Request for Comments — Internet Engineering Task Force, 1992. 21 с.
4. Github.com [Электронный ресурс]. URL: https://github.com/Disshell/Digital-sign-RSA, свободный (15.02.2018).