ИССЛЕДОВАНИЕ СОВРЕМЕННЫХ МЕТОДОВ ГЕНЕРАЦИИ ПРОСТЫХ ЧИСЕЛ, КЛЮЧЕЙ И СХЕМ ЦИФРОВОЙ ПОДПИСИ ДЛЯ МЕТОДА RSA

Качко Е.Г.
Харьковский национальный университет радиоэлектроники
61166, Харьков, пр. Ленина 14, каф. Программного обеспечения ЭВМ,
тел. /факс(057) 70-21-446, E-mail: ekachko@rambler.ru

Research of methods of generation of prime numbers is executed, key information and charts of digital signature in accordance with the standard of FIPS PUB 186-3, adding to him and PKCS #1 v2.1 on calculable complication. Formulated recommendation on application of different algorithms and charts.

Введение. Метод RSA широко используется в современных криптографических системах для цифровой подписи и несимметричного шифрования. О распространенности этого метода можно судить по тому, какой процент сертификатов включают в себя этот алгоритм операционная система Windows XP. Так, при просмотре сертификатов с помощью программы certmgr.msc, сертификаты всех доверенных корневых центров сертификации включают в себя алгоритм RSA. В отличие от других криптографических методов, данный метод не использует общих параметров. Для каждого пользователя должны быть сформированы свой модуль, а также открытый и личный ключи. Строго не рекомендуется использование одних и тех же ключевых данных для цифровой подписи и шифрования, а также в разных системах. И хотя генерация ключей выполняется в асинхронном режиме, в системе с большим числом пользователей, а именно такими являются, как правило, современные системы, требующие защиты, время генерации ключей часто становится критическим параметром. Данная работа посвящена оптимизации временных характеристик функций генерации ключей с учетом современных алгоритмов генерации [1, 2]. Рассмотрены также временные характеристики для операций вычисления и проверки цифровой подписи для современных схем.

Описание требований и алгоритмов. Требования к ключевым данным и алгоритмы генерации ключей, в том числе и для RSA, определены в FIPS PUB 186-3 Digital Signature Standard (DSS) [1]. Черновая версия этого Стандарта опубликована в марте 2006 года. Позже (декабрь 2007 г.) появились дополнения к Стандарту, относящиеся непосредственно к генерации ключей (Дополнения к FIPS 186-3: B.3, C.3, C.6, C.9, C.10 и F) [2]. Согласно этим требованиям, разрешается использовать RSA алгоритм с размерами модулей 1024, 2048 и 3072 бита. Для каждой длины есть несколько альтернативных методов генерации простых чисел. Данные о сравнении этих методов по их вычислительной сложности автору не известны. Проведенное исследование позволило выработать рекомендации по использованию различных методов для всех рекомендуемых длин модулей.

При генерации ключей для этих алгоритмов используется соответствующее значение security strength, которое определяется по длине модуля в соответствии с SP 800-57, часть 1. Значение security strength фактически определяет соответствие между алгоритмом хеширования, который используется для генерации ключей RSA и формирования цифровой подписи и размером модуля для алгоритма RSA.

В табл. 1 приведены значения размера хеша с соответствующим алгоритмом хеширования и длины модуля RSA в зависимости от security strength.

Таблица 1. Рекомендуемые соответствия между security_strength, хеш-функции и размера модуля RSA

Значение ***security_strength***	Размер хеша (бит) и алгоритм	Размер модуля RSA (бит)
80	160 (SHA-1)	1024
112	224 (SHA-224)	2048
128	256 (SHA-256)	3072

Из этой таблицы следует, что алгоритм хеширования определяет размер модуля RSA и наоборот.

Наиболее ресурсоемкими являются алгоритмы формирования простых чисел, произведение которых равно модулю RSA. Рассмотрим методы формирования простых чисел более подробно.

В дополнении к FIPS PUB 186-3 (B.3 IFC Key Pair Generation) определены два класса методов генерации простых чисел, которые используются для вычисления модуля: методы для генерации доказуемо простых чисел и методы для генерации вероятностно простых чисел. В свою очередь алгоритмы второго класса могут использовать для генерации простые числа, сформированные с помощью алгоритмов 1 класса. В данной работе выполняется сравнение этих методов по вычислительной сложности для рекомендуемых длин модуля, оптимизация соответствующих методов и выработка рекомендаций по их использованию.

Определение вычислительной сложности алгоритмов

Рассмотрим вычислительную сложность алгоритмов каждого класса. При реализации алгоритмов, которые требовали вычисление хеша, использовался алгоритм хеширования в соответствии с табл. 1. Для определения вычислительной сложности были реализованы предложенные алгоритмы, сформировано 10 пар простых чисел для рекомендованных и допустимых длин[1], определено максимальное, минимальное время, необходимое для генерации пары простых чисел, а также вычислено среднее время с учетом 10 измерений. При определении требуемого времени выполнялось только одно приложение. В качестве одноядерного процессора использовался процессор AMD Athlon(tm) 64 Processor 3000+, в дальнейшем Процессор 1. В качестве 2-х ядерного – процессор Intel(R) Core(TM)2 Duo CPU E6850 @ 3.00GHz (Процессор 2). Если процессор не указан, то Процессор 1 использовался для однопоточного приложения, Процессор 2 - для параллельных вычислений.

Как показали полученные результаты, вычислительная сложность существенно зависит не только от размера модуля, но и выбранного алгоритма. В табл. 2 представлены значения среднего времени генерации пары простых чисел в зависимости от алгоритма и значения модуля для однопоточного приложения.

Таблица 2. Среднее время генерации пары простых чисел p, q, в зависимости от метода и значения модуля

Метод генерации	Среднее время (с)		
	1024	2048	3072
Доказуемо простые числа (Приложение B.3.2.2)	Не поддерживается	13,2	73,3
Вероятностно простые числа (Приложение B.3.3)	Не поддерживается	26,8	98,3
Доказуемо простые числа (Приложение B.3.4)	1	19,1	145,3
Вероятностно простые числа на основе доказуемо простых чисел (Приложение B.3.5)	1,6	19,8	72,8
Вероятностно простые числа на основе вероятностно простых чисел (Приложение B.3.6)	3,2	44,8	213,7

Как следует из табл. 2, для модуля RSA 1024 бита наиболее эффективным по времени генерации является генерация доказуемо простых чисел в соответствии с алгоритмом приложения B.3.4. Для модуля RSA 2048 бита наиболее эффективным по времени генерации является генерация доказуемо простых чисел в соответствии с алгоритмом приложения B.3.2.2. Для модуля RSA 3072 бита наиболее эффективным по

[1] Не все алгоритмы допускают использование всех допустимых (1024, 2048, 3072) длин модуля. Так, алгоритм генерации доказуемо простых чисел из приложения B.3.2 и алгоритм генерации вероятностно простых из приложения B.3.3 формируют простые только для модулей длиной 2048, 3072.

времени генерации является генерация вероятностно простых чисел на основе доказуемо простых в соответствии с алгоритмом приложения В.3.5.

Так как использование доказуемо простых чисел более надежно, то рекомендуем и для модуля 3072 использовать алгоритм приложения В.3.2.2. как и для модуля 2048.

Оптимизация алгоритмов генерации с учетом архитектуры современных процессоров. С учетом использования многоядерных процессоров наиболее быстрым способом оптимизации является организация параллельных вычислений. Использование параллельных вычислений для определения простых p, q допустимы не для всех методов генерации, так как некоторые из них используют для формирования второго простого числа значение seed, полученное после генерации первого, поэтому будем параллельно вычислять пару простых чисел. Для организации параллельных вычислений используем технологию OpenMP [4].

Для оценки эффективности параллельных вычислений формирование простых выполнялось в одно и двух поточном режиме. Результаты приведены в табл. 3.

Таблица 3. Генерация ключей в последовательном и параллельном режимах (Процессор 2)

Метод генерации	Среднее время (с)					
	1024		2048		3072	
	1	2	1	2	1	2
Доказуемо простые числа (Приложение В.3.2.2)	-	-	17,5	9,3	102,9	57,5
Вероятностно простые числа (Приложение В.3.3)	-	-	20,5	9,1	157,6	62,7
Доказуемо простые числа (Приложение В.3.4)	1,1	1,2	18,8	9,1	99,1	30,7
Вероятностно простые числа на основе доказуемо простых чисел (Приложение В.3.5)	2,4	1,3	26,1	13,2	94,3	104,4
Вероятностно простые числа на основе вероятностно простых чисел (Приложение В.3.6)	1,8	1,6	25,6	11,9	121,4	63,3

1 – OPEN MP выключено;

2 - OPEN MP включено;

Заметим, что для последовательного и параллельного режимов вычислений рекомендации по выбору метода в зависимости от длины модуля остались прежними, хотя алгоритмы выполнены на разных типах компьютеров.

Параллельные вычисления позволяют уменьшить вычислительную сложность в среднем в 2 раза при числе ядер, равном 2. Увеличение количества ядер приведет к уменьшению вычислительной сложности.

Определение вычислительной сложности операций вычисления и проверки цифровой подписи. Вычисление и проверка цифровой подписи должны быть выполнены согласно PKCS #1 v2.1 [1]. По этому стандарту возможно использование двух схем формирования и проверки цифровой подписи RSASSA-PSS и RSASSA-PKCS1-v1_5. Различие в этих схемах состоит в способе дополнения результата хеширования до длины модуля для вычисления цифровой подписи. Первая схема рекомендована к использованию в новых приложениях, существующие приложения используют вторую схему. Поэтому современные системы цифровой подписи должны поддерживать работу для обеих схем. В данной работе исследуются эти схемы.

Для вычисления цифровой подписи используется экспонента личного ключа d. Для увеличения скорости выполнения базовой операции метода RSA возведения в

степень по модулю используется расширенное представление ключа d согласно PKCS #1 v2.1

В таблице 4 приведены усредненные значения времени для операций вычисления и проверки цифровой подписи для обеих схем для 10 измерений для открытого ключа, равного 65537.

Таблица 4. Усредненные значения времени для операций вычисления и проверки цифровой подписи для RSASSA-PSS и RSASSA-PKCS1-v1_5 схем (Процессор 2)

Модуль (Бит)	Время (с)			
	RSASSA-PSS		RSASSA-PKCS1-v1_5	
	Вычисление цифровой подписи	Проверка цифровой подписи	Вычисление цифровой подписи	Проверка цифровой подписи
1024	0,0088	0,0012	0,0088	0,0011
2048	0,0629	0,0041	0,0631	0,0041
3072	0,2045	0,0092	0,2041	0,0091

Как видно из таблицы, первая схема требует практически столько же времени для вычисления цифровой подписи, но она не является детерминированной в отличие от второй схемы. Для вычисления цифровой подписи по первой схеме для дополнения хеша сообщения используется функция генерации маски, на вход которой поступает случайный компонент. В случае второй схемы для дополнения хеша используются постоянные символы. Результаты для проверки схем для обеих схем также совпадают. Столь существенные различия во времени вычисления и проверки цифровой подписи (от 8 до 22 раз в зависимости от размера модуля) связаны с тем, что открытый ключ имеет специальную структуру, содержащую только два единичных бита в своем представлении. В этом случае требуется только 16 операций возведения в квадрат и одно умножение по модулю. Длина личного ключа совпадает с длиной модуля RSA и содержит приблизительно половину единичных битов. Использование открытых ключей, отличных от 65537, не поддерживается некоторыми RSA системами. Это различие в скорости операций следует учитывать при разработке архитектуры RSA системы, в которой желательно операции вычисления и проверки цифровой подписи выполнять в разных потоках. Одним из способов выравнивания скоростей является варьирование приоритетами потоков, использование процессоров разной производительности для выполнения потоков в многопроцессорной системе и т.д.

Выводы по работе.

1. Для генерации ключей лучше использовать методы с доказуемо простыми числами (для модуля 1024 бита – алгоритм из приложения B.3.4, для модулей 2048 и 3072 - алгоритм из приложения B.3.2.2).
2. После генерации личного ключа его записывать в расширенном формате.
3. Для обеспечения совместимости со старыми приложениями необходимо поддерживать схему 2 вычисления цифровой подписи.
4. Для полного выполнения требований стандарта PKCS #1 v2.1 необходимо поддерживать схему 1 вычисления цифровой подписи.
5. Для обеспечения масштабируемости приложений использовать технологию OPEN MP.

Литература. 1. RSA Cryptography Standard. FIPS PUB 186-3. FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION. Digital Signature Standard (DSS) Issued March, 2006. 2. Fips186-3_Strong-Prime-Sections_Dec2007.pdf. Appendices from FIPS 186-3: B.3, C.3, C.6, C.9, C.10 and F. 3. OpenMP и C++. MSDN Magazine / Русская Редакция, Октябрь 2005, 4. PKCS #1 v2.1: