УДК 004.056

Анализ быстродействия алгоритмов вычисления секретного ключа в асимметричной криптосистеме RSA

Алексеев А. П., Дикарева К. Н., Макаров М. И.

Постановка задачи: *при расчёте секретного ключа в криптосистеме RSA используется обобщённый алгоритм Евклида. В статье проводится сопоставление быстродействия обобщённого алгоритма Евклида с алгоритмами ALGO 1 и ALGO 2, разработанными авторами.* ***Целью работы*** *является проведение сравнительного анализа быстродействия трёх алгоритмов вычисления секретной экспоненты в криптосистеме RSA. Программная реализация алгоритмов осуществлена с помощью трёх систем программирования: Mathcad, Pascal и C#, что позволяет ослабить влияние среды программирования на полученные результаты.* ***Используемые методы:*** *сравнительный анализ быстродействия трёх алгоритмов выполнен экспериментальным методом путём измерения времени счёта при различных исходных данных. Вывод расчётных соотношений при разработке алгоритма ALGO 1 выполнен аналитически. Обоснование алгоритма ALGO 2 сделано путём доказательства пяти лемм.* ***Новизна:*** *разработано два новых алгоритма вычисления секретных экспонент.* ***Результат:*** *расчётным путём показано, что по убыванию быстродействия рассмотренные алгоритмы расположились так: обобщённый алгоритм Евклида, ALGO 1, ALGO 2. Выявлены причины различия в быстродействии рассмотренных алгоритмов.* ***Практическая значимость:*** *полученные результаты наглядно показали, как эффективность решения одной и той же задачи зависит от способа её решения. Три рассмотренных алгоритма проиллюстрированы программами на языках программирования Mathcad, Pascal и C#, а также примерами ручных расчётов. Это позволяет использовать результаты не только в научных целях, но и в учебном процессе.*

Ключевые слова: *асимметричная криптосистема RSA, алгоритм Евклида, быстродействие, секретный ключ, открытый ключ, экспонента, функция Эйлера, лемма, языки программирования.*

Актуальность

Ассиметричная криптосистема RSA используется для передачи ключей к симметричным криптосистемам, формирования цифровой подписи ответственных документов, создания почтовых клиентов [1, 2, 3]. Выбор значения открытой экспоненты влияет на характеристики криптосистемы. С одной стороны, уменьшение её значения (например, до 3) повышает скорость шифрования. С другой стороны, малые значения открытой экспоненты позволяют произвести успешную атаку на криптосистему [4]. Кроме того, значение открытой экспоненты влияет на время вычисления секретной экспоненты. Представляет интерес сделать попытку усовершенствовать широко используемый обобщённый алгоритм Евклида, который применяется при расчёте секретной экспоненты.

Постановка задачи

При расчёте секретной экспоненты в криптосистеме RSA используется обобщённый алгоритм Евклида. В статье производится сопоставление быстродействия обобщённого алгоритма Евклида с двумя алгоритмами ALGO 1 и ALGO 2, которые разработаны авторами. Оценка быстродействия алгоритмов может быть осуществлена аналитически или экспериментально [5]. В данной работе выбран второй путь.

Теоретическое обоснование

Расчёт секретной экспоненты t в криптосистеме RSA осуществляется с помощью сравнения [6]:

$$s \cdot t \equiv 1(\mathrm{mod}(\varphi(r)), \quad (1)$$

здесь s – число взаимно простое с $\varphi(r)$, так называемая открытая экспонента; r – произведение двух простых чисел p и q (модуль); $\varphi(r)$ – функция Эйлера, которая вычисляется по формуле:

$$\varphi(r) = (p-1)\cdot(q-1). \quad (2)$$

Из сравнения (1) по вычисленному значению функции Эйлера $\varphi(r)$ и выбранному значению s требуется найти такое значение t, при котором целочисленное деление величины $s \cdot t$ на $\varphi(r)$ даст остаток 1.

Вычисление секретной экспоненты t ведётся, как правило, с помощью обобщённого алгоритма Евклида [1, 2], блок-схема которого показана на рис. 1.

Основная идея вычислений состоит в том, что вводятся два числа $\varphi(r)$ и s. Большее число делится на меньшее. Если остаток от деления равен нулю, то вычисления завершаются. В противном случае большее число заменяется остатком от деления, а затем деление циклически продолжается.

На блок-схеме алгоритма операция округления до меньшего целого числа обозначена прямыми скобками, операция нахождения остатка от целочисленного деления – аббревиатурой mod.

Тексты программ для реализации обобщённого алгоритма Евклида на языке математической системы Mathcad 15, языках программирования Pascal и C# приведены в Приложениях 1, 2, 3.

Обобщённый алгоритм Евклида предназначен для решения численным методом уравнения вида:

$$ax + by = gcd(a, b), \quad (3)$$

в котором известны два целых положительных числа a и b, а требуется найти целые числа x, y и наибольший общий делитель чисел $gcd(a,b)$, которые удовлетворяют указанному уравнению. Англоязычная аббревиатура gcd означает: greatest common devisor. В русскоязычной литературе часто используется аббревиатура НОД – наибольший общий делитель.

Пример 1.

Выполним ручной расчёт секретной экспоненты по обобщённому алгоритму Евклида. Результаты расчётов представлены в таблице 1. Согласно обобщённому алгоритму Евклида вычисления нужно завершить при T_1=0 (см. рис. 1). Результаты содержатся в переменных $T_2 \rightarrow x$ и $T_3 \rightarrow y$ на предпоследней итерации. Переменная T_1 на предпоследней итерации содержит значение остатка от целочисленного деления. Переменная P сохраняет значение T_3 на предыдущей итерации.

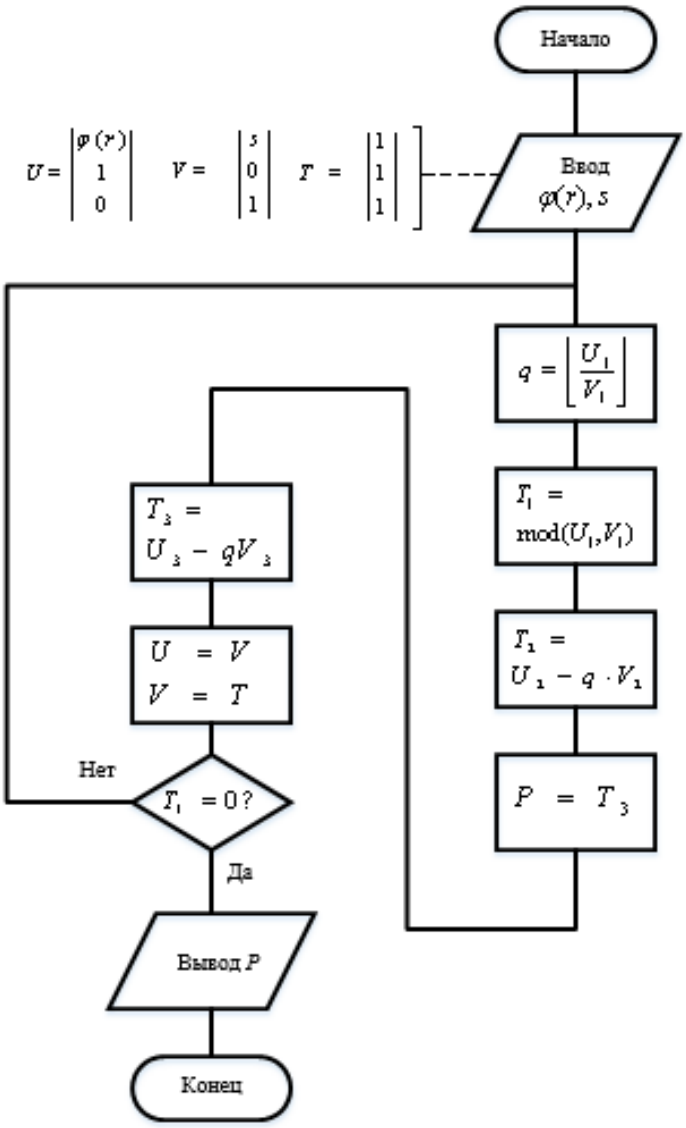

Рис. 1. Обобщённый алгоритм Евклида

Решение.

Исходные числа:

$\varphi(r) = 460$, $s = 257$.

Таблица 1. Расчёт с помощью обобщённого алгоритма Евклида

Итерация	q	T_1	T_2	T_3	P
1	1	203	1	-1	1
2	1	54	-1	2	-1
…	…	…	…	…	…
6	6	1	-119	213	-34
7	2	0	257	-460	213

В результате расчётов получено: x= -119, y=213.

Для проверки полученных результатов подставим найденные числа в уравнение (3):

$$460 \cdot (-119) + 257 \cdot 213 = -54740 + 54741 = 1.$$

Наибольший общий делитель чисел 460 и 257 должен быть равен 1. Это определяется правилом выбора открытой экспоненты [1]. Проверка показала, что числа -119 и 213 вычислены верно. Секретная экспонента равна 213, а остаток от целочисленного деления равен 1.

Рассмотрим ещё один алгоритм вычисления секретной экспоненты ALGO 1. Из сравнения (1) нужно найти такое целое число t, которое при целочисленном делении числа $m=s\cdot t$ на $\varphi(r)$ даст остаток, равный единице. Таким образом, число m должно быть обязательно больше числа $\varphi(r)$ хотя бы на единицу (так как остаток от целочисленного деления равен 1). Кроме того, число $m=\varphi(r)+1$ должно нацело делиться на число s, поэтому нужно проверять является ли целым число $k=m/s$. Если на первой итерации число k не является целым, то следует взять новое число $m=2\varphi(r)+1$, проверить является ли целым новое число $k=m/s$ и т.д., то есть с каждой новой итерацией число m должно увеличиваться на величину $\varphi(r)$.

Блок-схема алгоритма ALGO 1 показана на рис.2.

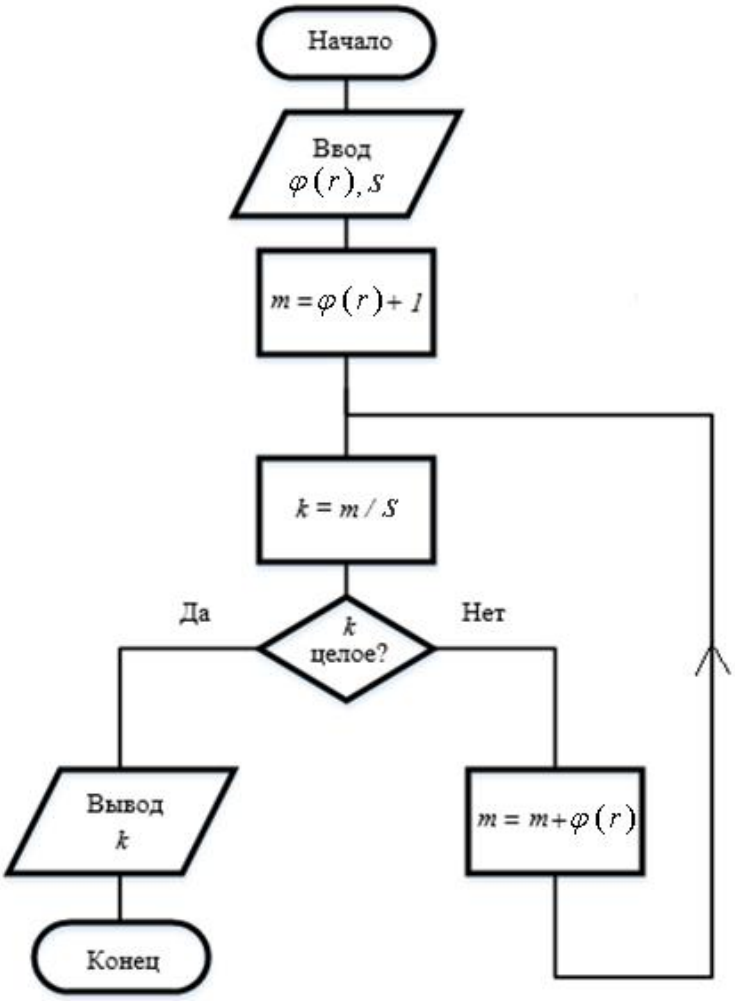

Рис. 2. Блок-схема алгоритма ALGO 1

В соответствии с алгоритмом ALGO 1 секретную экспоненту можно рассчитать по формуле:

$$t = \frac{n \cdot \varphi(r) + 1}{s}.$$

Здесь n определяет число необходимых итераций. Число n следует увеличивать от 1 до такого значения, при котором t станет целым числом. Из формулы видно, что число итераций n будет всегда меньше значения открытой экспоненты s.

Пример 2.

Выполним ручной расчёт с помощью алгоритма ALGO 1.

Решение.

Исходные числа: $s = 27$, $\varphi(r) = 460$.

Результаты расчётов помещены в таблицу 2.

Таблица 2. Расчёт с помощью алгоритма ALGO 1

Итерация	m	k	k целое ?	Экспонента
1	461	17,04	Нет	-
2	921	34,11	Нет	-
…	…	…	…	…
26	11961	443	Да	443

Секретная экспонента равна 443.

В Приложениях 4, 5 и 6 приведены программы для реализации алгоритма ALGO 1 на языке математической системы Mathcad 15, языках Pascal и C#.

Рассмотрим третий алгоритм нахождения секретной экспоненты (ALGO 2). Требование (1) можно записать иначе $s \cdot t-1=0$ $(mod \varphi(r))$, то есть целочисленное деление величины $s \cdot t-1$ на $\varphi(r)$ должно дать остаток 0.

Требование к выбору секретного ключа t можно сформулировать так:

$$\frac{s \cdot t - 1}{\varphi(r)} = n, \qquad (4)$$

где n – целое число.

Поиск числа t, удовлетворяющего соотношению (4), нужно вести среди чисел:

$$t \geq \frac{\varphi(r) + 1}{s}. \qquad (5)$$

Обоснуем разработку алгоритма ALGO 2 с помощью лемм.

Лемма 1

Если $\varphi(r)$ кратно10, а число s заканчивается цифрой 7, то число t должно заканчиваться цифрой 3.

Доказательство

Так как произведение указанных чисел s и t будет заканчиваться единицей, то в соответствии с (4) в результате вычитания единицы из произведения $s \cdot t$ будет получено число, кратное 10.

Таким образом, величину t нужно искать среди чисел, у которых последняя цифра 3, например, 3, 13, 23, 33, 43 и т.д.

Пример 3.

Пусть $\varphi(r) = 460$, $s = 27$, тогда в соответствии с (5) ответ нужно искать среди чисел больших 16,33, причём число $27 \cdot t - 1$ должно нацело делиться на число 460. Первое значение числа t, с которого следует начинать поиск числа, удовлетворяющего соотношениям (4) и (5), равно 23.

Результаты расчётов приведены в таблице 3.

Таблица 3. Расчёт с помощью алгоритма ALGO 2

Итерация	t	n	n целое?	Экспонента
1	23	17,04	Нет	-
2	33	34,11	Нет	-
3	43	51,148	Нет	-
…	…	…	…	…
43	443	443	Да	443

Расчёт секретного ключа после сорока трёх итераций дал значение 443. Как видно из таблицы, вычисления ведутся до тех пор, пока число n не станет целым.

Лемма 2

Если число $\varphi(r)$ кратно 10, а число s заканчивается цифрой 3, то число t должно заканчиваться цифрой 7.

Доказательство

Доказательство аналогично доказательству леммы 1.

Таким образом, величину t нужно искать среди чисел, заканчивающихся на цифру 7, например, 7, 17, 27, 37, 47 и т.д.

Пример 4.

Пусть $\varphi(r)=460$, $s=3$, тогда в соответствии с (4) ответ нужно искать среди чисел больших, чем 153,67, причём число $3 \cdot t - 1$ должно нацело делиться на $\varphi(r)$. Первое значение числа t, с которого следует начинать поиск числа, удовлетворяющего соотношениям (4) и (5), равно 157. Поиск секретного ключа в этом случае дал значение 307.

Лемма 3

Если $\varphi(r)$ кратно 10, а s заканчивается цифрой 9, то последняя цифра числа t должна быть 9.

Доказательство

Только произведение двух чисел, оканчивающихся цифрами 9 (при s, заканчивающимся на 9), даёт число, у которого последняя цифра 1. В этих случаях число $s \cdot t-1$ будет кратно 10.

Пример 5.

Пусть $\varphi(r)=120$, $s=19$.

В соответствии с (5) поиск чисел t нужно вести среди целых чисел, которые больше 6,368. Число 9 не удовлетворяет условию (4), а число t=19 является искомым ответом.

Лемма 4

Если $\varphi(r)$ кратно 10, а s заканчивается цифрой 1, то последняя цифра числа t должна быть 1.

Доказательство

Только произведение двух чисел, оканчивающихся цифрами 1(при числе s, заканчивающимся цифрой 1), даёт число, у которого последняя цифра 1. В этих случаях число $s \cdot t-1$ будет кратно 10.

Пример 6.

Пусть $\varphi(r)$=120, s=31.

В соответствии с (5) поиск чисел t нужно вести среди целых чисел, которые больше 3,903.

Числа 11 и 21 не удовлетворяет условию (4), а число t=31 является ответом.

Лемма 5

Если $\varphi(r)$ кратно 10, то число s не может оканчиваться на цифру 5.

Доказательство

Числа $\varphi(r)$ и s должны быть взаимно простыми. Таким образом, при $\varphi(r)$, кратном десяти, число s может заканчиваться только цифрами 1, 3, 7 и 9.

Блок-схема алгоритма ALGO 2 показана на следующем рисунке.

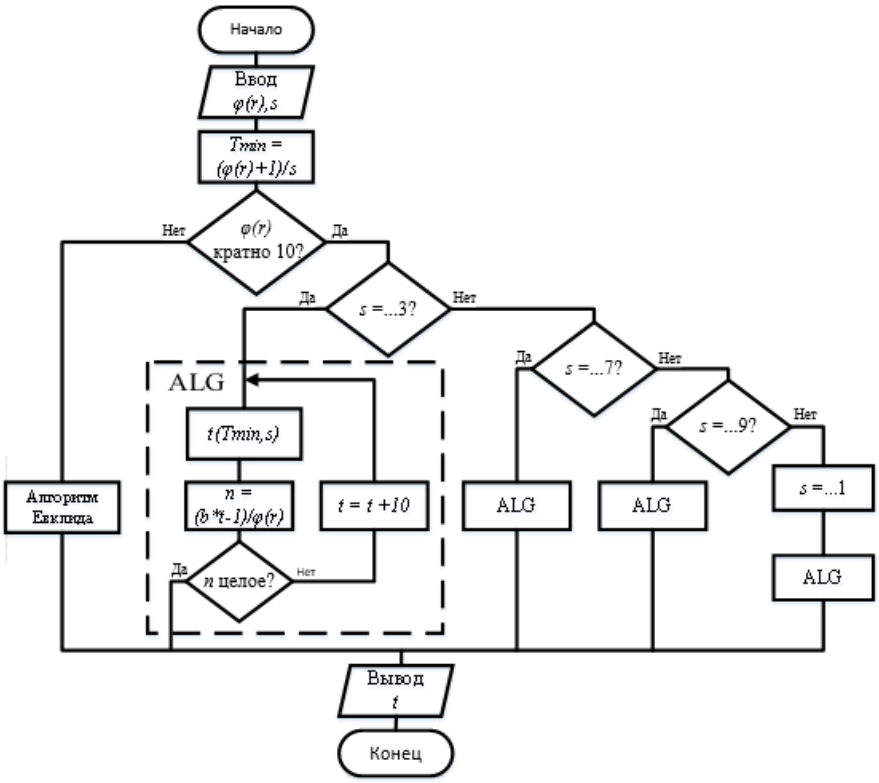

Рис. 3. Блок-схема алгоритма ALGO 2

В Приложении 7 приведён текст программы на языке математической системы Mathcad 15, а в Приложениях 8 и 9 – программы соответственно на языках программирования Pascal и C#.

Величина T_{min} вычисляется по формуле (5) и определяет наименьшее значение секретной экспоненты.

Работу алгоритма ALGO 2 можно описать так. Отличия этого алгоритма от обобщённого алгоритма Евклида существуют только при значениях функции Эйлера, кратных десяти. В этих случаях проводится анализ выбранной открытой экспоненты. Если она оканчивается цифрой 3, то секретный ключ должен оканчиваться цифрой 7. Приближение к искомой секретной экспоненте в этом алгоритме происходит путём многократного прибавления числа 10 к некоторому начальному числу (5). Если открытая экспонента оканчивается числом 7 (или 1, 9), то секретная экспонента должна оканчиваться цифрой 3 (соответственно 1, 9). В любом случае приращения равны 10.

Обсуждение результатов расчёта

Разработанные программы были использованы для оценки времени счёта секретной экспоненты по трём алгоритмам. Предварительные расчёты проводились на трёх ЭВМ, в трёх различных программных средах: Mathcad, Pascal и C#. Таблицы с фрагментами расчётов приведены в Приложениях 10, 11 и 12. Даже беглый анализ таблиц, приведённых в Приложениях 10, 11 и 12. показывает, что алгоритм ALGO 2 не может конкурировать по быстродействию с обобщённым алгоритмом Евклида и алгоритмом ALGO 1.

Для определения времени выполнения расчётов в программах на языке программирования C# применялся экземпляр класса Stopwatch, входящий в состав области имён System.Diagnostics. Запуск отсчёта времени выполнения программы осуществлялся методом StartNew, а по завершению счёта использовался метод ElapsedMilliseconds, возвращающий значение затраченного времени, выраженного в миллисекундах.

Чаще всего программы выполнялись за время менее одной миллисекунды. По этой причине в программах использовалось многократное циклическое повторение расчётов от 1000 до 2 000 000 раз. Программы были реализованы в виде консольных приложений. Однако время выполнения операций ввода-вывода через консоль не добавлялось ко времени, затраченному на вычисления.

Перечислим технические характеристики использованных вычислительных средств. При расчётах на Mathcad 15 использовался компьютер Assus R2H, с тактовой частотой 1 ГГц, объёмом ОЗУ 2 Гбайт (операционная система Windows XP). При вычислениях на языке программирования PascalABC.NET версия 2.2, сборка 1013 (17.08.2015) использовался компьютер Assus, с тактовой частотой 3,4 ГГц, объёмом ОЗУ 4 Гбайт. Расчёты на C# 4.0 велись на ЭВМ Lenovo ThinkCentre. с тактовой частотой 1.8 ГГц, объёмом ОЗУ 4 Гбайт (операционная система Windows 7). Программа была реализована в среде разработки Visual Studio 2010.

Более детальный анализ проведём с помощью рисунков, которые получены путём аппроксимации большого массива экспериментальных данных. Аппроксимация осуществлялась с помощью программы TC 3D [7].

На рисунках 4 и 5 показаны зависимости времени t расчёта секретной экспоненты от значений открытой экспоненты s и функции Эйлера $\varphi(r)$ для программ, составленных на языках программирования C# и Pascal (обобщённый алгоритм Евклида). Время на графиках выражено в микросекундах.

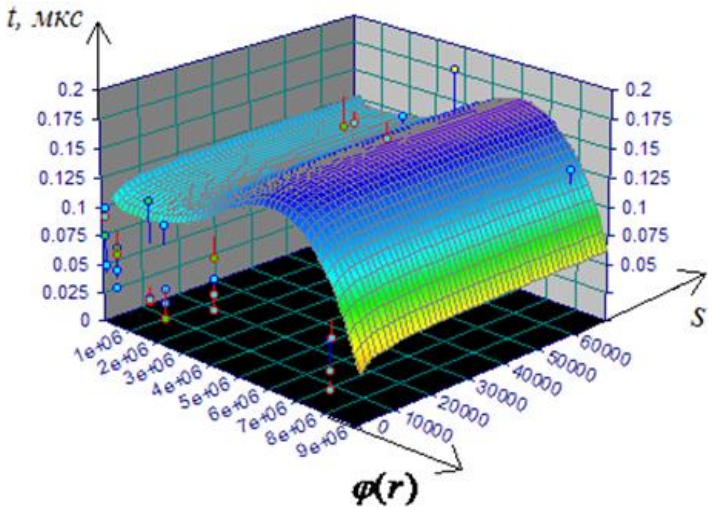

Рис. 4. Зависимость $t=F(s, \varphi(r))$ для обобщённого алгоритма Евклида (C#)

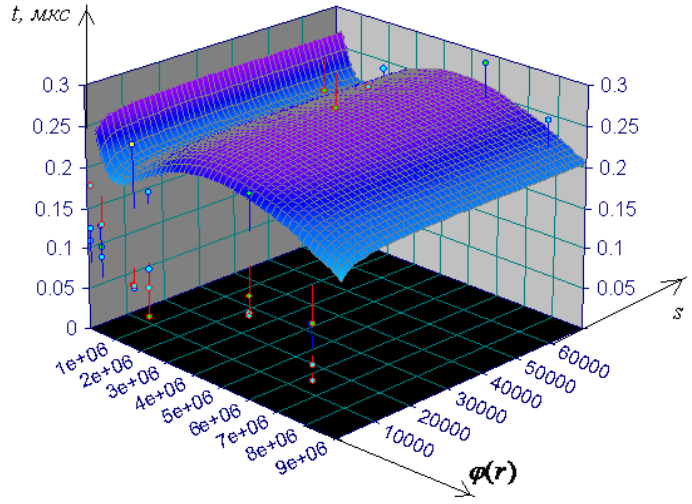

Рис. 5. Зависимость $t=F(s, \varphi(r))$ для обобщённого алгоритма Евклида (Pascal)

Из приведённых зависимостей видно, что для обобщённого алгоритма Евклида время счёта колеблется в небольших пределах (0,2 мкс) и нет явных тенденций изменения (увеличение или уменьшение) с вариацией влияющих переменных.

На рисунках 6 и 7 показаны графики зависимости времени счёта от значений открытой экспоненты и функции Эйлера для алгоритмов ALGO 1 и ALGO 2.

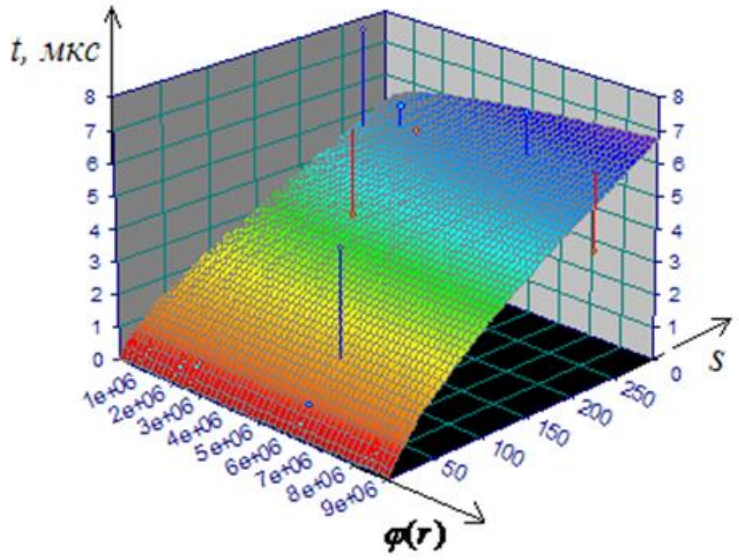

Рис. 6. Зависимость $t = F(s, \varphi(r))$ для алгоритма ALGO 1 (C#)

Из рис. 6 видна сильная зависимость времени счёта от значения открытой экспоненты. С ростом s время счёта программы, составленной по алгоритму ALGO 1, начинает в разы превышать время счёта по обобщённому алгоритму Евклида. В то же время, в области малых значений s алгоритм ALGO 1 конкурирует по быстродействию с обобщённым алгоритмом Евклида.

Анализ рис. 7 показывает, что время счёта по алгоритму ALGO 2 составляет десятки миллисекунд, что на 3…4 порядка хуже, чем у других алгоритмов. Само время счёта растёт с увеличением значений s и $\varphi(r)$.

В алгоритме ALGO 1 приближение к искомому результату идёт шагами (приращениями, квантами), равными значению функции Эйлера. Эти числа имеют большую величину, так как являются произведением двух больших чисел p-1 и q-1.

В алгоритме ALGO 2 приращения всегда равны 10, поэтому алгоритм ALGO 2 существенно проигрывает в быстродействии обобщённому алгоритму Евклида и алгоритму ALGO 1, так как для достижения большого значения секретного ключа требуется большое число итераций. В обобщённом алгоритме Евклида движение к цели происходит путём многократного поиска остатков от деления. Операция деления позволяет с большей скоростью продвигаться к цели по сравнению с увеличением величин десятками от начального числа, как это происходит в алгоритме ALGO 2.

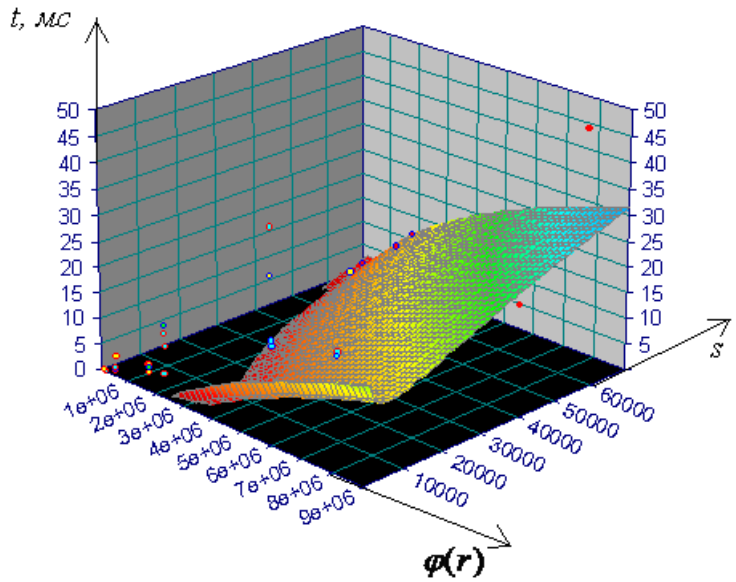

Рис. 7. Зависимость $t=F(s, \varphi(r))$ для алгоритма ALGO 2 (C#)

Для проведения сравнительного анализа алгоритмов введём величину относительного (нормированного) быстродействия R. Эта величина равна отношению времени счёта одного алгоритма ко времени счёта другого алгоритма при одинаковых значениях влияющих параметров (открытой экспоненты и функции Эйлера). Нормирование производилось ко времени счёта, затраченного на вычисления с помощью обобщённого алгоритма Евклида. Измерение времени счёта проводились в одной программной среде C#) и на одном компьютере.

Величину R нужно трактовать следующим образом. Если R=1, то быстродействие алгоритмов одинаковое. Если R>1, то время счёта с помощью обобщённого алгоритма Евклида меньше, чем время счёта сопоставляемого алгоритма в R раз. При R<1 время счёта с помощью обобщённого алгоритма Евклида в $1/R$ раз больше по отношению к альтернативному алгоритму.

На рис. 8 показана зависимость относительного быстродействия от значений открытой экспоненты и функции Эйлера для алгоритма ALGO 1. Из графика видно, что с ростом s алгоритм ALGO 1 начинает в разы проигрывать самому быстрому алгоритму. Лишь при малых значениях открытой экспоненты $R \approx 1$ и алгоритм ALGO 1 порой опережает обобщённый алгоритм Евклида. При значениях открытой экспоненты, которые превышают значения, показанные на графике алгоритм ALGO 1, начинает проигрывать в сотни и тысячи раз.

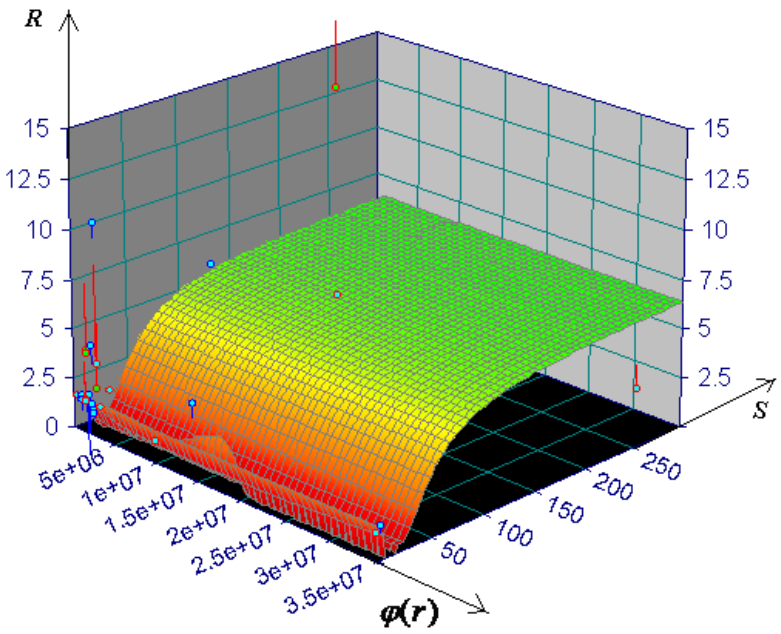

Рис. 8. Зависимость $R=F(s, \varphi(r))$ для алгоритма ALGO 1 (C#)

На следующем рисунке показана зависимость относительного быстродействия R алгоритма ALGO 2 от влияющих параметров.

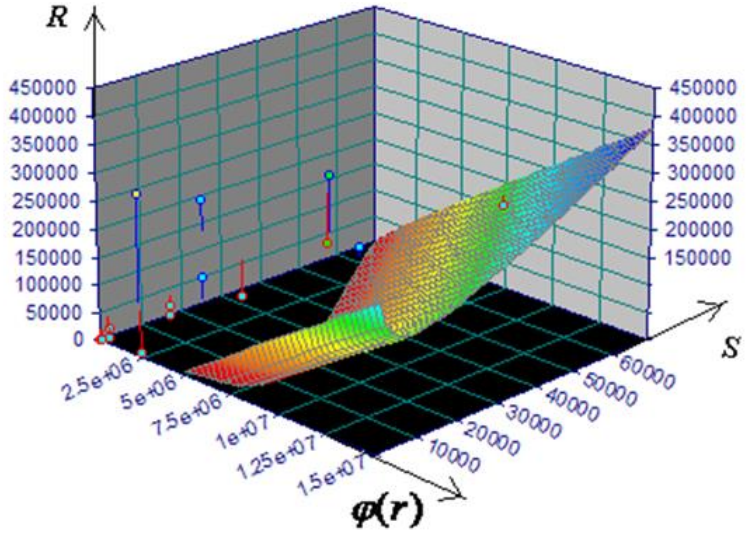

Рис. 9. Зависимость $R=F(s, \varphi(r))$ для алгоритма ALGO 2 (C#)

Время счёта по алгоритму ALGO 2 порой превышало время счёта с использованием обобщённого алгоритма Евклида в несколько сотен тысяч раз. С ростом значений открытой экспоненты и функции Эйлера алгоритм ALGO 2 всё в большей степени «отстаёт» по быстродействию от обобщённого алгоритма Эйлера. При формировании двух последних графиков расчёты проводились в одной программной среде (C#) и на одном компьютере (Lenovo ThinkCentre), поэтому результаты инвариантны по отношению к используемым при моделировании аппаратным средствам.

Выводы

Два разработанных новых алгоритма, в принципе, могут быть использованы для расчётов, так как при одинаковых исходных данных все три алгоритма дают одинаковые значения секретной экспоненты. Однако быстродействие алгоритмов различно.

Наибольшее быстродействие среди рассмотренных алгоритмов в подавляющем большинстве случаев показал обобщённый алгоритм Евклида. При малых значениях открытой экспоненты лучшие результаты порой даёт алгоритм ALGO 1. Однако, время вычисления секретной экспоненты по алгоритму ALGO 1 резко возрастает с ростом значения открытой экспоненты. Установлено, что число необходимых итераций в алгоритме ALGO 1 всегда меньше значения открытой экспоненты.

Алгоритм ALGO 2 имеет отличительные особенности только для значений функции Эйлера, кратных 10. При всех других значениях функции Эйлера расчёты ведутся с помощью обобщённого алгоритма Евклида. Время счёта по этому алгоритму на три-четыре порядка превышает время счёта по обобщённому алгоритму Евклида и ALGO 1. По этой причине этот алгоритм не может быть использован для практической реализации криптосистем.

Алгоритмы ALGO 1 и ALGO 2 удобно использовать при ручных расчётах, например, в учебном процессе при проведении практических занятий в ВУЗах для объяснения идеи шифра RSA и углублённого изучения алгоритмов формирования секретных ключей [8].

Авторы посвящают статью столетию со дня рождения Алексеева Петра Андреевича.

Приложение 1.

Программа на языке Mathcad 15 для расчёта с помощью обобщённого алгоритма Евклида

Обобщенный алгоритм Евклида
Mathcad 15

ORIGIN := 1

a := 460 b := 257

$$U := \begin{pmatrix} a \\ 1 \\ 0 \end{pmatrix} \qquad V := \begin{pmatrix} b \\ 0 \\ 1 \end{pmatrix} \qquad T := \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

$$k1(a,b) := \left| \begin{array}{l} \text{while } T_1 \neq 0 \\ \quad \left| \begin{array}{l} c \leftarrow \dfrac{U_1}{V_1} \\ q \leftarrow \text{floor}(c) \\ T_1 \leftarrow \text{mod}(U_1, V_1) \\ T_2 \leftarrow U_2 - q \cdot V_2 \\ P \leftarrow T_3 \\ P \leftarrow P + a \text{ if } P < 0 \\ T_3 \leftarrow U_3 - q \cdot V_3 \\ U \leftarrow V \\ V \leftarrow T \end{array} \right. \\ P \end{array} \right.$$

$$f(a,b) := \left| \begin{array}{l} \text{for } i \in 1..2000000 \\ \quad x_i \leftarrow k1(a,b) \\ x_i \end{array} \right.$$

$$f(a,b) = 213$$

Приложение 2.

Программа на языке Pascal для расчёта с помощью обобщённого алгоритма Евклида

```
//Обобщённый алгоритм Евклида
//PascalABC.NET Версия 2.2, сборка 1013(17.08.2015)var
U1,U2,U3,a,b,T1,T2,T3,V1,V2,V3,P,q,i:integer;
begin
Writeln('введите a и b');
readln(a,b);
U1:=a;U2:=1;U3:=0;
V1:=b;V2:=0;V3:=1;
T1:=1;T2:=1;T3:=1;
while T1>0 do
begin
q:=U1 div V1;
T1:=U1 mod V1;
T2:=U2-V2*q;
P:=T3;
If P<0 then P:=P+a;
T3:=U3-q*V3;
U1:=V1;U2:=V2;U3:=V3;
V1:=T1;V2:=T2;V3:=T3;
end;
Writeln(P);
     end.
```

Приложение 3.

Программа на языке C# для расчёта с помощью обобщённого алгоритма Евклида

```
// Обобщённый алгоритм Евклида
// C#
 sw.Restart();
 int q = 0;
 int p = 0;
 for (long i = 0; i < 2000000; i++)
 {
  int u1 = a, u2 = 1, u3 = 0;
  int v1 = b, v2 = 0, v3 = 1;
  int t1 = 1, t2 = 1, t3 = 1;

                while (t1 > 0)
                {
                    q = u1 / v1;
                    t1 = u1 % v1;
                    t2 = u2 - v2 * q;
                    p = t3;
                    if (p < 0) p = p + a;
                    t3 = u3 - q * v3;
                    u1 = v1; u2 = v2; u3 = v3;
                    v1 = t1; v2 = t2; v3 = t3;
                  }
            }
  sw.Stop();
  elapsed = sw.ElapsedMilliseconds;
  Console.WriteLine("Евклид = "+ p);
  Console.WriteLine("Total query time: {0} ms", elapsed);
  Console.ReadKey();
```

Приложение 4.

Программа на языке Mathcad 15 для расчёта с помощью алгоритма ALGO 1

Алгоритм ALGO 1

Mathcad 15

$a := 460 \qquad b := 27$

$k(a,b) :=$ | $m \leftarrow a + 1$
| $d \leftarrow 1$
| while $d \neq 0$
| | $k \leftarrow \frac{m}{b}$
| | $d \leftarrow k - \text{floor}(k)$
| | $m \leftarrow m + a$
| k

$k(a,b) = 443$

Приложение 5.

Программа на языке Pascal для расчёта с помощью алгоритма ALGO 1

```
//Алгоритм ALGO 1
//PascalABC.NET Версия 2.2, сборка 1013(17.08.2015)
var s,f,m,y:integer;
k:real;
begin
writeln('введите s и f');//здесь s – число взаимно простое с f, так называемая
открытая экспонента; f – функция Эйлера; k-секретная экспонента
readln(s,f);

m:=f+1;
k:=m/s;
while frac(k)<>0 do//Пока остаток от деления не равен 0
  begin
k:=m/s;
m:=m+f;
  end;
writeln(k);
      end.
```

Приложение 6.

Программа на языке C# для расчёта с помощью алгоритма ALGO 1

```
//Алгоритм ALGO 1
//C#
int a = int.Parse(Console.ReadLine());
int b = int.Parse(Console.ReadLine());

double m,k = 0;
Stopwatch sw =  Stopwatch.StartNew();for (long i = 0; i < 20000; i++)
{
  m = a + 1;
  k = 0;
  while (m % b != 0)
  {
     k = m / b;
     m += a;

  }
     k = m / b;
 }
 sw.Stop();
 long elapsed = sw.ElapsedMilliseconds;
 Console.WriteLine("Алго-1= "+k);
 Console.WriteLine("Total query time: {0} ms", elapsed);
 Console.ReadKey();
```

Приложение 7.

Программа на языке Mathcad 15 для расчёта с помощью алгоритма ALGO 2

Алгоритм ALGO 2

Mathcad 15

a - функция Эйлера; b - открытая экспонента

$a := 460$ $b := 27$

Секретная экспонента не может быть меньше числа:

$Tmin := \frac{(a + 1)}{b}$ $Tmin = 17.07$

Начальные значения секретной экспоненты и переменной d:

$t := 23$ $d := 1$

$k2(a,b) :=$ while $d \neq 0$
- $n \leftarrow \frac{b \cdot t - 1}{a}$
- $d \leftarrow n - floor(n)$
- $t \leftarrow t + 10$

$t - 10$

$k2(a,b) = 443$

Приложение 8.

Программа на языке Pascal для расчёта с помощью алгоритма ALGO 2

```
//Алгоритм ALGO 2
//PascalABC.NET Версия 2.2, сборка 1013(17.08.2015)
var
  f,s,i: integer;//здесь s – число взаимно простое с f, так называемая открытая
экспонента; f – функция Эйлера, t-секретная экспонента
  t:int64;
  begin
  writeln('введите s и f');//Введём функцию Эйлера f и взаимно простое с ним
число s
  readln(s, f);
 t:=(f+1)div s;//Найдём t минимальное
  i:=0;
  if (f mod 10 = 0) then //Если f кратно 10
  begin
       if (s mod 10 = 7) then //Если s заканчивается на 7
   begin
       If (t mod 10<>3) then //Если t не заканчивается на 3
        repeat t:=t+1;//Увеличиваем t на 1 пока t не будет заканчиваться на 3
        until (t mod 10=3);
        repeat t:=t+10*i;//Увеличиваем t на 10 пока не найдём нужное число для
выражения
        i:=1;
        until (s*t-1)mod f=0;
   end;
       if (s mod 10 = 3) then//Если s заканчивается на 3
    begin
       If (t mod 10<>7) then//Если t не заканчивается на 7
        repeat t:=t+1;//Увеличиваем t на 1 пока t не будет заканчиваться на 7
        until (t mod 10=7);
        repeat t:=t+10*i;//Увеличиваем t на 10 пока не найдём нужное число для
выражения
        i:=1;
        until (s*t-1)mod f=0;
    end;
      if (s mod 10 = 1) then//Если s заканчивается на 1
   begin
      If (t mod 10<>1) then//Если t не заканчивается на 1
        repeat t:=t+1;//Увеличиваем t на 1 пока t не будет заканчиваться на 1
        until (t mod 10=1);
        repeat t:=t+10*i;//Увеличиваем t на 10 пока не найдём нужное число для
выражения
```

```
        i:=1;
        until (s*t-1)mod f=0;
   end;
      if (s mod 10 = 9) then//Если s заканчивается на 9
   begin
      If (t mod 10<>9) then//Если t не заканчивается на 9
        repeat t:=t+1;//Увеличиваем t на 1 пока t не будет заканчиваться на 9
        until (t mod 10=9);
        repeat t:=t+10*i;//Увеличиваем t на 10 пока не найдём нужное число для
выражения
        i:=1;
        until (s*t-1)mod f=0;
   end;
   end
    else
         while (s<>0) and (f<>0) do
   begin
   if s > f then
        s := s mod f
    else
        f:=f mod s;
        t:=s+f;
end;
    writeln(t);//Вывод секретной экспоненты
             end.
```

Приложение 9.

Программа на языке C# для расчёта с помощью алгоритма ALGO 2

```
// Алгоритм ALGO 2
// C#
sw.Restart();
int g = 0;
for (long i = 0; i < 2000; i++)
{
 int ss = b % 10;
 int r = 0;
 switch (ss)
 {
   case 7: r = 3; break;
   case 3: r = 7; break;
   case 1: r = 1; break;
   case 9: r = 9; break;
 }
 int t_min = (a + 1) / b;
 int t = t_min;
 for (; t % 10 != r; t++) ;
int mm = a + 1;
double dd = 1;
double n;
do
{
  n = (b * t - 1) / (double)a;
  dd = n - Math.Floor(n);
  g = t;
  t += 10;
 } while (dd != 0);
}
sw.Stop();
elapsed = sw.ElapsedMilliseconds;
Console.WriteLine("Алго-2= " + g);
Console.WriteLine("Total query time: {0} ms", elapsed);
Console.ReadKey();
```

Приложение 10.

Расчёты на Mathcad

Алгоритм	Время, мкс	Функция Эйлера, $\varphi(r)$	Экспонента, s	Экспонента, t
Евклида	12,5	460	3	307
ALGO 1	3			
ALGO 2	26,5			
Евклида	17,8	460	17	433
ALGO 1	20			
ALGO 2	67			
Евклида	13	460	27	443
ALGO 1	31			
ALGO 2	78			
Евклида	46	460	257	213
ALGO 1	145			
ALGO 2	36,5			
Евклида	16,5	1324800	7	1135543
ALGO 1	9,5			
ALGO 2	220000			
Евклида	100	1324800	257	1211393
ALGO 1	365			
ALGO 2	260000			
Евклида	75	1324800	65537	1295873
ALGO 1	80000			
ALGO 2	235000			
Евклида	13,5	34522660	3	23015107
ALGO 1	4,1			
ALGO 2	1680000			
Евклида	27	34522660	7	24659043
ALGO 1	7,4			
ALGO 2	2800000			
Евклида	31	34522660	17	14215213
ALGO 1	10,8			
ALGO 2	1700000			
Евклида	44	34522660	257	1611953
ALGO 1	16,9			
ALGO 2	215000			

Приложение 11.

Расчёты на C#

Алгоритм	Время, мкс	Функция Эйлера, $\varphi(r)$	Экспонента, s	Экспонента, t
Евклида	0,065	460	3	307
ALGO 1	0,0635			
ALGO 2	0,6655			
Евклида	0,063	460	17	433
ALGO 1	0,4375			
ALGO 2	1,592			
Евклида	0,0625	460	27	443
ALGO 1	0,706			
ALGO 2	1,618			
Евклида	0,199	460	257	213
ALGO 1	3,213			
ALGO 2	1,093			
Евклида	0,171	1324800	7	1135543
ALGO 1	0,063			
ALGO 2	33970			
Евклида	0,176	1324800	257	1211393
ALGO 1	5,88			
ALGO 2	42160			
Евклида	0,279	1324800	65537	1295873
ALGO 1	1587,5			
ALGO 2	93335			
Евклида	0,655	34522660	3	23015107
ALGO 1	0,595			
ALGO 2	3575250			
Евклида	0,115	34522660	7	24659043
ALGO 1	0,138			
ALGO 2	6126650			
Евклида	144	34522660	17	14215213
ALGO 1	189,5			
ALGO 2	3830350			
Евклида	0,156	34522660	257	1611953
ALGO 1	0,393			
ALGO 2	450450			

Приложение 12.

Расчёты на Pascal

Алгоритм	Время, мкс	Функция Эйлера, $\varphi(r)$	Экспонента, s	Экспонента, t
Евклида	0,019	460	3	307
ALGO 1	0,0535			
ALGO 2	0,15			
Евклида	0,019	460	17	433
ALGO 1	0,34			
ALGO 2	0,44			
Евклида	0,021	460	27	443
ALGO 1	0,51			
ALGO 2	0,45			
Евклида	0,081	460	257	213
ALGO 1	2,4			
ALGO 2	0,27			
Евклида	0,019	1324800	7	1135543
ALGO 1	0,15			
ALGO 2	1150			
Евклида	0,07	1324800	257	1211393
ALGO 1	5			
ALGO 2	1500			
Евклида	0,14	1324800	65537	1295873
ALGO 1	2200			
ALGO 2	1750			
Евклида	0,02	34522660	3	23015107
ALGO 1	0,06			
ALGO 2	12222			
Евклида	0,06	34522660	7	24659043
ALGO 1	0,11			
ALGO 2	20000			
Евклида	0,08	34522660	17	14215213
ALGO 1	0,17			
ALGO 2	14000			
Евклида	0,06	34522660	257	1611953
ALGO 1	0,26			
ALGO 2	1600			

Литература

1. Молдовян Н. А., Молдовян А. А. Введение в криптосистемы с открытым ключом. Санкт-Петербург: БХВ-Петербург, 2005. 288 с.

2. Рябко Б. Я., Фионов А. Н. Основы современной криптографии и стеганографии. М.: Горячая линия-Телеком, 2010. 232 с.

3. Алексеев А. П. Информатика 2015. М.: Солон-Пресс, 2015. 400 с.

4. Фергюсон Н., Шнайер Б. Практическая криптография. М.: Издательский дом «Вильям», 2005. 424 с.

5. Sedgewick R. Algorithms in C. 3-е изд. Addison-Wesley Publ., 1998. 1136 p.

6. Rivest R., Shamir A., Adleman L. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems // Communications of the ACM. 1978. Т. 21. № 2. Pp. 120-126.

7. Алексеев А. П., Камышенков Г. Е. Использование ЭВМ для математических расчётов. Самара: Парус, 1998. 190 с.

8. Алексеев А. П. Сборник задач по дисциплине «Информатика» для ВУЗов. М.: Солон-Прессс, 2015. 104 с.

References

1. Moldovian N. A., Moldovian A. A. *Vvedenie v kriptosistemy s otkrytym kliuchom* [Introduction to Public Key Cryptosystems]. Saint-Petersburg: BHV-Petersburg Publ., 2005. 288 p. (in Russian).

2. Rjabko B. Ja., Fionov A. N. *Osnovy sovremennoj kriptografii i steganografii* [The Foundations of Modern Cryptography and Steganography]. Moscow, Gorjachaja linija-Telekom Publ., 2010. 232 p. (in Russian).

3. Alekseev A. P. *Informatika 2015* [Informatics 2015]. Moscow, SOLON-Press Publ., 2015. 400 p. (in Russian).

4. Ferguson N., Schneier B. *Practical Cryptography*. New York, John Wiley & Sons Publ., 2003. 432 p.

5. Sedgewick R. *Algorithms in C*. 3rd Ed. Addison-Wesley Publ., 1136 p. 1998.

6. Rivest R., Shamir A., Adleman L. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 1978, vol. 21, no. 2, pp. 120-126.

7. Alekseev A. P., Kamyshenkov G. E. *Ispol'zovanie EVM dlia matematicheskikh raschetov* [The use of Computers for Mathematical Calculations]. Samara, Parus Publ., 1998. 190 p. (in Russian).

8. Alekseev A. P. *Sbornik zadach po distsipline "Informatika" dlia VUZov*. [Problems in the discipline "Computer science" for High Schools]. Moscow, SOLON-Press Publ., 2015. 104 p. (in Russian).

Статья поступила 14 сентября 2015 г.

Информация об авторах

Алексеев Александр Петрович - кандидат технических наук, доцент, старший научный сотрудник. Профессор кафедры информатики и вычислительной техники. Поволжский государственный университет телекоммуникаций и информатики. Область научных интересов: криптография; стеганография; информатика; контрольно-измерительная техника; дефектоскопия. E–mail: apa_ivt@rambler.ru

Дикарева Ксения Николаевна – студентка. Поволжский государственный университет телекоммуникаций и информатики. Область научных интересов: криптография. E–mail: ksusha199607@yandex.ru

Макаров Максим Игоревич - кандидат технических наук. Доцент кафедры информатики и вычислительной техники. Поволжский государственный университет телекоммуникаций и информатики. Область научных интересов: криптография; стеганография; информатика. E–mail: moox700@gmail.com

Адрес: 443013, г. Самара, ул. Льва Толстого, 23.

Analysis of the Performance of Algorithms for Calculating the Secret Key in Asymmetric Cryptosystem RSA

Alekseev A. P., Dikareva K. N., Makarov M. I.

Purpose. *In many publications indicates that the wrong choice of parameters RSA encryption may reduce the reliability of his. However, no mention of the fact that the public key and private key, in some cases, can completely match the subscriber and then publish the private key. The aim is to prove the possibility of forming twin keys where public and private keys are the same.* ***Methods.*** *The possibility of forming the twin keys theoretically justified with eight lemmas.* ***Novelty****. It is shown that for values of Euler's function, a multiple of ten, there is a possibility of publication of the private key, if the public key numbers ending in 1 or 9.* ***Results****. By calculation confirmed the possibility of the formation of the twin keys.* ***Practical relevance.*** *The results will improve RSA cryptographic cipher by checking for a match generated public and private keys.* ***Purpose.*** *When calculating the secret key used in the RSA cryptosystem generalized Euclidean algorithm. The article compares the performance of the generalized Euclidean algorithm with algorithms ALGO 1 and ALGO 2, developed by the authors. The aim is to conduct a comparative analysis of the performance of three algorithms for computing the secret exponent in the cryptosystem RSA. Software implementation of the algorithms implemented using three systems programming: Mathcad, Pascal and C #, allowing you to reduce the influence of the programming on the results.* ***Methods.*** *Comparative analysis of the performance of three algorithms performed by experimental method of measuring time account for different input data. Output of calculated relations for the development of algorithm ALGO 1 is made analytically. Justification algorithm ALGO 2 done by the evidence of five lemmas.* ***Novelty****. Developed two new algorithm for computing the secret exponents.* ***Results****. By calculation shows that the descending speed of the algorithms are located as follows: generalized Euclidean algorithm, ALGO 1, ALGO 2. The causes of the differences in the speed considered algorithms.* ***Practical relevance.*** ***T****he results obtained clearly demonstrate the efficiency of solving the same problem depends on the method of solving it. The three programs discussed algorithm illustrated in programming languages Mathcad, Pascal and C #, as well as examples of manual calculations. This allows you to use the results, not only for scientific purposes, but also in the educational process*

Keywords: asymmetric cryptosystem RSA, Euclidean algorithm, performance, private key, public key exponent, Euler function, Lemma, programming languages.

Information about Authors

Alekseev Aleksandr Petrovich - Ph.D. of Engineering Sciences, Associate Professor, Senior Research Officer. Professor at the Department of Computer Science and Engineering. Povolzhye State University of Telecommunications and Informatics. Field of research: cryptography; steganography, computer science; testing and measuring equipment; flaw detection. E–mail: apa_ivt@rambler.ru

Dikareva Ksenia Nikolaevna – student at the Department of Computer Science and Engineering. Povolzhye State University of Telecommunications and Informatics. Field of research: cryptography. E-mail: ksusha199607@yandex.ru.

Makarov Maxim Igorevich - Ph.D. of Engineering Sciences, Docent at the Department of Computer Science and Engineering. Povolzhye State University of Telecommunications and Informatics. Field of research: cryptography; steganography, computer science. E-mail: moox700@gmail.com

Address: Russia, 443013, Samara, L. Tolstogo Street, 23.