ЛЕКЦИЯ 6

КРИПТОГРАФИЯ С ОТКРЫТЫМ КЛЮЧОМ

На прошлых лекциях была рассмотрена классическая криптография или **криптография с секретным ключом**. Эта криптография отличается тем, что и для шифрования, и для расшифрования используется один и тот же ключ (либо их можно получить друг из друга тривиальным способом). Например, можно сказать, что в шифре Цезаря для шифрования и расшифрования используется один и тот же ключ ($z = 3$). Также можно сказать, что для шифрования используется $z = 3$, а для расшифрования — $z = -3$. Таким образом, тривиальные преобразования позволяют получить из ключа шифрования ключ для расшифрования.

Этот принцип подходит для всех раундовых ключей в DES, GOST, AES и т. д. — ключи получаются путём перестановки соответствующих раундовых ключей, то есть ключи берутся в обратном порядке. Отличие **криптографии с открытым ключом** от классической криптографии в том, что при использовании ключей получить ключ расшифрования из ключа шифрования сложно. Это приводит к тому, что ключ шифрования нет нужды держать в секрете. Можно рассказать всему миру, какой нужно использовать ключ шифрования, для того, чтобы зашифровать сообщение, при этом ключ расшифрования сохранить в секрете. В результате не потребуется предварительно устанавливать канал связи для того, чтобы передать секретную информацию, а именно ключ, который нужен для шифрования информации. То есть в случае с GOST'ом, AES, DES, для того чтобы зашифровать информацию, нужно предварительно по защищённому каналу связи передать ключ шифрования таким образом, чтобы злоумышленник его не услышал. В случае же шифрования на открытых ключах этого не требуется. Но все же имеются ограничения и у алгоритмов, и у протоколов.

1. Алгоритм Диффи – Хеллмана

Принято считать, что начало криптографии с открытыми ключами было положено работой **Диффи и Хеллмана** 1976-го года. В этой работе был впервые описан протокол, который позволял двум сторонам установить общий секретный ключ. Этот прото-

кол позволял выработать общий секретный ключ, не используя секретный канал связи. Выглядел он следующим образом: собеседники выбирают некие случайные числа p (primary — простое число) и g (generator). Далее первый собеседник выбирает случайное число a и пересылает второму результат вычисления:

$$A = g^{\mathrm{a}} \mod p,$$

а второй выбирае случайное число b, проводит вычисление:

$$B = g^{\mathrm{b}} \mod p$$

и отсылает первому собеседнику результат. Тогда общий ключ, одинаковый для обеих сторон, будет вычисляться следующим образом:

$$K = B^{\mathrm{a}} \mod p = A^{\mathrm{a}} \mod p = g^{\mathrm{ab}} \mod p.$$

Особенность этого протокола в том, что числа A и B можно передавать по открытому каналу. Единственным требованием к этому каналу является то, что злоумышленник не должен иметь возможность что-то менять в канале. Если это условие выполнено, то алгоритм Диффи – Хеллмана позволяет установить защищённый канал связи.

2. Алгоритм RSA

В 1977-м году публикуется краткое описание нового алгоритма, который позже назовут **«алгоритм RSA»** по имени своих создателей: Ривест, Шамир, Адлеман. В этом журнале была предложена задачка по вскрытию алгоритма шифрования. Были предложены некие параметры, ключ, и было сказано, что все желающие могут попробовать получить с помощью известного ключа шифрования исходный текст. Восстановить текст получилось только в 90-х годах, причём в работе участвовало несколько десятков тысяч компьютеров, связанных между собой через Интернет, причём как минимум три из них были факс-машинами.

Полное описание алгоритма вышло в 1978-м году. В 1982-м году была организована компания RSA Security, которой выдали патент в 1983-м году. В 1993-м году алгоритм стандартизировали в качестве стандарта **Private Key Cryptography Standard 1 (PKCS1)** — первый стандарт по криптографии с открытым ключом.

3. Шифрование с открытым ключом

Функция шифрования $E_{\mathrm{K1}}(X)$ и функция расшифрования $D^{\mathrm{K2}}(Y)$ используют разные ключи. Также используются также следующие обозначения: ключ public (K_{pub}) и ключ private (K_{prix}) — открытый ключ и закрытый ключ. Также используются функция генерации подписи ($S_{\mathrm{K2}}(M)$) и функция валидации подписи ($V_{\mathrm{K1}}(M, S)$).

Шифрование с открытым ключом основывается на такой математической проблеме, как существование **односторонних функций с потайной дверцей** (oneway trapdoor function).

Можно привести следующий пример для демонстрации того, что шифрование на открытых ключах возможно, но имеет свои ограничения: есть большой телефонный

справочник, в котором есть фамилии на все буквы алфавита, включая «ь», «ъ», «ы». Каждой фамилии поставлена в соответствие тысяча телефонных номеров (десятки тысяч). Предположим, что никаких компьютеров для работы с этим справочником нет. Будем брать букву открытого текста и шифровать эту букву каким-нибудь телефонным номером, относящимся к фамилии, начинающейся на эту букву. Из открытого текста, состоящего из букв, получится закрытый текст, состоящий из телефонных номеров. В этом состоит принцип шифрования на открытых ключах.

Затратив определённое количество усилий, всё-таки можно восстановить исходный открытый текст. Для этого нужно найти заданный телефонный номер, посмотреть, какой фамилии он соответствует, восстановить буквы и проделать так со всеми буквами. Также можно предварительно составить другой телефонный справочник, в котором отсортированы номера телефонов. Тогда процедура расшифрования займёт гораздо меньше времени. Соответственно, телефонный справочник, в котором данные отсортированы по фамилии — это открытый ключ. Телефонный справочник, который отсортирован по номерам — это закрытый ключ, которым будет пользоваться тот, кто расшифровывает текст.

Также у систем шифрования с открытым ключом существует еще одна особенность: восстановить закрытый ключ по открытому ключу теоретически можно, но на практике не получится, так как не хватит времени. Для всех известных систем есть алгоритмы перевода открытого ключа в закрытый, но все эти алгоритмы работают очень медленно. Не существует эффективных алгоритмов ни перевода закрытого ключа в открытый, ни расшифрования.

4. Криптосистема RSA

Любая криптосистема состоит из трёх (минимум из двух) алгоритмов. Это генерация «открытый-закрытый ключ», алгоритмы шифрования/расшифрования и/или алгоритм генерации и валидации электронно-цифровой подписи.

Алгоритм RSA начинается с того, что выбираются два больших простых числа p и q (от $2^{2047}-1$ до $2^{2048}-1$). Вычисляется число n: $n = p \times q$. Дальше вычисляется функция Эйлера. Теорема Эйлера говорит, что если если число a взаимно просто с неким простым числом p, тогда $(a^{(p-1)})$ даёт $(1 \mod p)$; если числа a и n взаимно просты, то тогда число $a^{\phi(n)}$, где $\phi(n)$ — функция Эйлера, сравнимо с $(1 \mod n)$, причём функция Эйлера показывает число взаимно простых с n чисел, но меньших n. Соответственно, функция Эйлера для любого простого числа будет давать $(p–1)$. Функция Эйлера для произведения двух взаимно простых чисел будет равна $((p–1) \times (q–1))$.

Далее выбирается открытая экспонента e. Она должна быть взаимно простая с функцией Эйлера от n. Обычно выбираются экспоненты следующего вида: $\{100 \ldots 001\}$. Затем вычисляется закрытая (секретная) экспонента: $d = e^{-1} \mod \phi(n)$. Можно указать, что $d \times e$ сравнимо с $1 \mod \phi(n)$. Тогда **открытым ключом** называется пара чисел n и e, а **закрытым**— пара чисел n и d.

n и e будем использовать для шифрования, а n и d — для расшифрования.

5. Процедура шифрования

Пусть m — некое сообщение, причём $0 \leqslant m \leqslant n$–1. Теперь говорим, что m — некое сообщение, длиной меньше, чем 4000 бит. Для функции шифрования имеем формулу:

$$c = E_{(\mathrm{n,e})}(m) = m^e \mod n,$$

а для расшифрования:

$$m = D_{(\mathrm{n,d})}(c) = c^d \mod n.$$

С математической точки зрения, устройство этого шифра намного проще, чем DES и AES. Но с точки зрения реализации, это непростые системы, потому что у них есть большое количество особенностей, вопросов производительности и т. д.

Рассматриваемый шифр имеет следующие **недостатки**:

1. Если m есть нулевое сообщение, то в канал передастся ноль. Вероятность такого события: $P = 2^{-4000}$ (Vanilla RSA).

2. Одинаковые сообщения будут шифроваться одинаково, поэтому перед зашифровкой их дополняют определённым ключом, либо используют режимы сцепления блоков.

Также к недостаткам можно отнети проблему обмена открытым ключом между собеседниками. Один из собеседников должен предварительно рассказать, какой у него открытый ключ, но сделать это он может только по открытому каналу связи. То есть открытый ключ не является секретным.

6. Китайская теорема об остатках

Если натуральные числа a_1, a_2, ... , a_n попарно взаимно просты, то для любых целых r_1, r_2, ... , r_n таких, что $0 \leqslant r_i < a_i$ при всех $i \in \{1, 2, \dots, n\}$, найдётся число N, которое при делении на a_i даёт остаток r_i при всех $i \in \{1, 2, \dots, n\}$.

Более того, если найдутся два таких числа N_1 и N_2, то $N_1 \equiv N_2 \mod a_1 \times a_2 \times \dots \times a_n$.

Основная идея этой теоремы состоит в том, что можно ввести некий базис разложения чисел по взаимно простым числам. Координатами в этом базисе будут остатки от деления на $a_1 \dots a_n$. Любые простые числа от нуля до произведения a_n они будут задаваться однозначно; если же какие-то два числа раскладываются по этому базису с одними и теми же координатами, то эти два числа равны с точностью до модуля произведений чисел a_i. Эта теорема широко используется в теории чисел; в RSA она задействована для доказательства корректности схем.

7. Доказательство корректности

Доказать корректность алгоритма шифрования — это значит доказать, что он зашифровывает и расшифровывает верно (если зашифровать сообщение с одним ключом, то после расшифрования с соответствующем ключом на выходе должно получиться исходное сообщение; если текст не поменялся, то его электронная подпись валидна, а если поменялся, то с большой вероятностью не валидна, и т. д.).

Докажем корректность алгоритма RSA. Сначала рассмотрим случай $m \neq 0 \mod p$ (m не кратно p)

$$ed = 1 \mod \phi(n) = 1 + k(p–1)(q–1),$$

$$m^{ed} = m^{1+k(p–1)(q–1)} \mod p.$$

Согласно теореме Ферма, $m^{(p–1)} = m^1 \mod p$. m и p взаимно просты, так что такая замена обоснованна.

$$m^{1+k(p–1)(q–1)} \mod p = m(1)^{k(q–1)} \mod p = m \mod p.$$

Если $m = 0 \mod p$ (m кратно p), то $m^{ed} = 0 \mod p$. То есть криптосистема RSA корректна для шифрования и для расшифрования при заданных формулах вычислений.

8. Сложность криптосистемы RSA

В алгоритме RSA сложно произвести операцию, обратную шифрованию. Операция шифрования описывается следующей формулой:

$$m^e \mod n,$$

а обратная операция:

$$m = \sqrt[e]{c} \mod n.$$

Никакого общего алгоритма для вычисления подобного рода нет. Эту проблему можно решить, если сначала вычислить закрытый ключ из открытого ключа ($(n, e) \to (n, d)$), а это можно сделать только одним способом: если разложить число n на числа p и q.

Задача вычисления корня и задача факторизации являются сложными вычислительными задачами. В настоящий момент никто не придумал эффективных алгоритмов, выполняющихся на машине Тьюринга, которые позволяли бы решить такую задачу.

Функция шифрования является односторонней функцией с потайной дверцей.

$$c = m^e \mod n.$$

Здесь нельзя произвести обратную операцию, если не знать, как n раскладывается на p и q. Если это известно, то можно вычислить эту однонаправленную функцию в обратном направлении. Для этого нужно m^e возвести в степень d, где d задаётся следующей формулой:

$$d = e^{-1} \mod \phi(n) = e^{-1} \mod (p–1)(q–1).$$

Но для того, чтобы это сделать, нужно вычислить функцию Эйлера от числа n, а для этого необходимо знать, как n раскладывается на простые множители. Перебор всех чисел от 0 до $(n–1)$ и проверка, являются они взаимно простыми или нет, не будут эффективными: n занимает 4000 бит.

9. Цифровая подпись

Новая возможность, которую предоставляет криптография с открытым ключом — это возможность генерации цифровой подписи. Для произвольного сообщения алгоритм RSA генерирует дополнительное число, причём это число может сгенерировать только владелец закрытого ключа, а любой владелец открытого числа может проверить, правильно это число было сгенерировано или нет.

В схеме собеседники обозначены как Алиса и Боб.

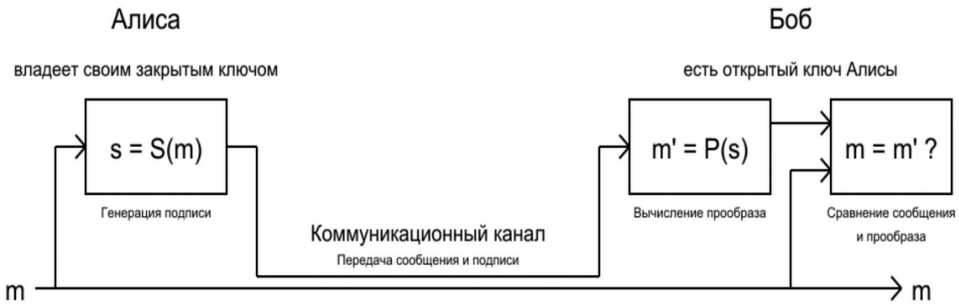

Рис. 6.1

Генерация выполняется следующим образом:

$$s = S_{(\mathrm{n,d})}(m) = m^d \mod n$$

Во время проверки полученная подпись возводится в степень e, и получается m':

$$m' = s^e \mod n.$$

Если $m' = m$, то подпись валидна.

Доказательство корректности аналогично предыдущему.

Зашифровать сообщение может только владелец пары (d, n), потому что число d известно только ему.

Теперь предположим, что злоумышленнику известно сообщение и подпись. Он не сможет узнать, в какую степень надо возвести сообщение, чтобы получить подпись. Чтобы найти d, ему нужно вычислить $d = \log_m S \mod n$.

Задача вычисления дискретного логарифма, точно также как задача факторизации, в настоящий момент решается только на квантовых компьютерах. Причём если задачи факторизации уже были выполнены на практике, то дискретный алгоритм пока не реализован. Таким образом, схема одновременной передачи (n, s) довольно надёжна.

Получатель вычисляет прообраз сообщения и сравнивает его с тем, что было передано. Таким образом можно проверить, что сообщение было передано конкретным отправителем.

10. Надёжность системы RSA

Это очень надёжная система при соответствующей реализации: нужно оптимально выбрать открытую экспоненту, вычислить секретную, убедится, что в реализации нет недостатков, связанных там с атаками по таймингу, с атаками по сторонним каналам, по кэшам и т. д.

Система RSA обладает высокой производительностью в отличие от шифрования на открытых ключах. Возведение числа в степень размером 4000 бит — очень сложная операция. В DES, AES и GOST используется намного меньше операций.

Для повышения производительности был предложен другой подход. Когда нужно зашифровать сообщение, вначале с помощью криптографически стойкого генератора случайных чисел генерируется новый ключ того размера, который нужен для шифрования блочным шифром. Дальше этот ключ («сессионный ключ») возводится в секретную экспоненту по модулю n и вручается отправителю. Далее, все данные в этой сессии шифруются этим ключом, который в свою очередь зашифрован надёжным алгоритмом с открытым ключом.

Подобную систему можно использовать и при передаче данных по открытым каналам, потому что по открытому каналу сначала передаётся зашифрованный сессионный ключ; отправитель сессионный ключ расшифровывает, и злоумышленник ничего не узнает. Этим ключом осуществляется шифрование между собеседниками. При этом шифрование на открытых ключах выполняется один раз в начале сессии (или несколько раз в течение сессии, если требуется так называемая **ревалидация ключей**). Это делается относительно быстро, так как работа идет с небольшими объёмами данных.

Такая схема является безопасной и быстрой, потому что ключ шифруется сначала медленным алгоритмом (но там данных мало), а потом основные данные шифруются блочным шифром (быстрый шифр).

Аналогично и в случае электронно-цифровой подписи: Вычисляется хеш-функция от заданного сообщения, причём хеш-функция должна быть криптографически стойкая, чтобы её нельзя было подделать. Далее эта хеш-функция возводится в степень секретной экспоненты по модулю n, и получившаяся конструкция уже называется цифровой подписью. Далее отправитель передаёт получателю сообщение и цифровую подпись. Тот вычисляет хеш-функцию, возводит полученную подпись в открытую степень (s^e) и проверяет, совпали числа или нет. Если совпали, то сообщение сконструировал тот, кто владеет открытым ключом отправителя.

11. Недостаток системы RSA

Криптосистема на открытых ключах позволяет создать защищённый канал связи (связанный общим секретным ключом) в предположении, что собеседники общаются по открытому каналу, а злоумышленник не может в этом канале менять данные. Если у злоумышленника имеется такая возможность, то происходит следующее: предположим, один из собеседников хочет отправить сообщение другому и запрашивает у него открытый ключ. Его перехватывается злоумышленником, после чего тот отправляет первому собеседнику свой открытый ключ, выдавая его за ключ второго. Затем первый собеседник шифрует всю информацию открытым ключом, полученным от злоумышленника.

Понятно, что осуществить такую атаку сложнее: подменить сообщение от обладателя ключа труднее, чем просто перехватить сообщение от первого собеседника. Тем не менее, именно такие атаки являются основной угрозой криптосистемы на открытых ключах.

Существует **способ борьбы с перехватом сообщений**: выделяется некий промежуточный центр (**центр сертификации**), и он подписывает своей электронно-цифровой подписью ключи собеседников, причём открытый ключ этого центра есть у всех возмож-

ных участников.

Например, в качестве такого центра может выступать государство. В центре сертификации каждый может получить открытый и закрытый ключ. Открытый ключ обладателя подписан открытым ключом государства, и у обладателя этот открытый ключ государства тоже есть. Когда тот с кем-нибудь общаетесь, он можете проверить, что открытый ключ собеседника подписан государством, и он может это проверить, используя открытый ключ государства.

В общем случае есть две схемы сертификации: **сеть с корневыми центрами сертификации** и **распределённая сеть (PGP)**

12. Генерация больших простых чисел

В настоящий момент нет эффективных алгоритмов генерации больших простых чисел, то есть тех, которые можно было бы использовать в RSA. Поэтому используются другие алгоритмы — **алгоритмы генерации псевдопростых чисел** (это числа, которые простые с большой степенью вероятности).

Эти алгоритмы выбирают число (нечётное) и проверяют, является ли оно простым или нет с некоторой вероятностью. Если является, и эта вероятность удовлетворительна, то его используют. Если не является, или полученная вероятность неудовлетворительна, то его не используют.

13. Примеры алгоритмов генерации псевдопростых чисел

1. **Решето Эратосфена:** этот алгоритм для заранее заданного массива чисел строит так называемое решето и отфильтровывает те числа, которые не являются простыми, и в результате оставляет только простые.

   Составляется таблица чисел. Число 2 помечается (обводится кругом), зачёркиваются все числа, которые делятся на 2. Дальше помечается число 3, вычёркиваются все числа, которые делятся на 3. Следующее незачёркнутое число: 5. Оно помечается, а все числа, которые делятся на 5, вычёркиваются, и так далее.

   Несложно посчитать, что на одно число придётся $\frac{1}{\sqrt{n}}$ операций. Но при этом придётся проходить весь массив чисел от 1 до n. То есть это не алгоритм проверки на простоту, а алгоритм поиска простых чисел от 1 до n, и для конкретного числа n он совершенно не эффективен. Гораздо эффективнее, напрмер, проверить, делится ли число n на числа от 1 до $\sqrt{n}$, причём нужно проверять только простые числа, которые можно найти с помощью решета Эратосфена.

2. **Алгоритм Ферма.** Назовём число a свидетелем простоты числа n по Ферма (при условии, что a и n — взаимно простые, а $(a^{(n-1)})$ сравнимо с $(1 \mod n)$ (проверяем, что для всех чисел a, взаимно простых с n, выполняется малая теорема Ферма).

   Здесь возникают проблемы: нужно перебрать все числа от 2 до $(n–1)$. Также если не обращать внимание на необходимость проверки на простоту, то упомянутое условие выполняется ещё и для составных чисел (также есть так называемые **числа Кармайкла**, для которых условие малой теоремы Ферма выполняется всегда.

   Алгоритм Ферма применяется нечасто: он долго работает, а также на практике часто оказывается неэффективным из-за чисел Кармайкла. Поэтому используется другой алгоритм: берётся число n, затем от него отнимается единица (чтобы получить четное число). Получившееся число делится на 2, потом снова, и т. д. Получается набор чисел: $(n–1)$, $\frac{n-1}{2}$ ... $\frac{n-1}{2^t}$, где t — максимально возможная степень. Далее некое число a и возводится в эти степени: $a^{\frac{n-1}{2^t}} \mod n$ и так далее до $(a^{n-1} \mod n$.

   Если a и n — взаимно простые то результатом будет 1. Теперь рассмотрим

   $$x = a^{\frac{n-1}{2}}, y = (n-1).$$

   Тогда

   $$x^2 = y \mod n.$$

   Если $y = 1$, то x равен либо 1, либо $(n–1)$ при условии, что n является простым (группа по умножению, образованная n, не содержит делителей ноля). Если содержит, то возможны нетривиальные корни, но если n — простое, тогда x принимает значения 1 или $(–1)$. Поэтому a называют **свидетелем простоты**, если этот ряд содержит все единицы (то есть ряд начинается с 1 и, очевидно, оканчивается на 1, или ряд заканчивается на 1, но при этом содержит $(–1)$. Очевидно, что если встречается $(–1)$, то дальейшими членами будут единицы. До $(–1)$ может быть любое

число. Во всех остальных случаях число a не является свидетелем простоты числа n по Миллеру.

Если хотя бы один несвидетель простоты найден, то n не является простым. Если a является свидетелем простоты, то n является простым с вероятностью $\frac{3}{4}$ (по статистике). То есть, если число n — не простое, то существует только четверть таких чисел a, которые будут являться свидетелями, причём первое из чисел, которое не является свидетелем, если верить **расширенной гипотезе Римана о нулях дзета-функции на комплексной плоскости**, встретится до числа $70\,(\ln n)^2$. Перебрав все числа от 0 до $70\,(\ln n)^2$, можно найти хотя бы одно такое a, которое не будет являться свидетелем простоты числа n, если n — не простое.

3. В **алгоритме Миллера** перебираются все числа от нуля до $70\,(\ln n)^2$. Если хотя бы одно из них не будет являться свидетелем простоты, то n — не простое, а если все будут являться, значит n — простое. Чуть позже этот результат в 1989 году сократил **Бах** до $2\,(\ln n)^2$. Позже **Владимиров** показал, что проверять надо только простые числа.

4. **Алгоритм Миллера – Рабина** — модификация алгоритма Миллера, состоящая в следующем: берутся k чисел, после чего проверяется, являются ли они свидетелями. Если все они являются свидетелями, то n — простое с большой вероятностью (вероятность ошибки будет составлять $\frac{1}{4^k}$, где k — количество проверенных случайных чисел).

   В отличие от алгоритма Миллера этот алгоритм требует проверки очень малого количества чисел: в современных системах это порядка 10-ти или 20-ти разных a.

5. **Тест AKS** тоже основан на модификации малой теоремы Ферма, но в которой проверяют, что если взять многочлен $(x + a)^n$, то после взятия его по модулю n должен получиться многочлен $(x^n + a)$, причём это будет справедливо только в том случае, если n — простое.

   Дальше было показано, что можно выбрать достаточно малый многочлен в качестве модуля, используя который и перебрав различные a, можно с уверенностью утверждать, что n будет являться простым.

Алгоритмы проверки чисел на простоту резделяются на:

1. **Детерминированные** или нет. **Детерминированными** называются те алгоритмы, которые всегда дают один и тот же результат без использования случайных чисел. Алгоритм Миллера алгоритм Ферма и AKS являются детерминированными, а алгоритм Миллера – Рабина — нет.

2. **Полиномиальные** или нет, то есть выполняющиеся или нет за полиномиальное время относительно длины проверяемого числа. Среди всех рассмотренных, полиномиальными являются два: алгоритм Миллера – Рабина (при условии, что число выбираемых кандидатов k постоянно) и алгоритм Агравал – Каял – Саксены.

3. **Корректные** или нет, то есть доказана корректность алгоритма или нет.

Алгоритм Ферма некорректен (так как есть числа Кармайкла). Алгоритм Миллера: корректность не доказана, потому что не доказана корректность расширенной гипотезы Римана. Алгоритм Миллера – Рабина — корректный, полиномиальный, но не детерминированный. Алгоритм Агравал – Каял – Саксены является полиномиальным, детерминированным, корректным, но он работает намного медленнее, чем все остальные алгоритмы , поэтому в настоящий момент на практике используется дополнительно алгоритм на эллиптических кривых, использующий краткую проверку, а потом — основную проверку с помощью алгоритма Миллера – Рабина: выбирается примерно 10 кандидатов и проверяется, являются они свидетелями простоты числа n или нет. Если все являются, тогда число n называется простым, и его можно использовать в качестве числа p или в качестве числа q для схемы RSA. Вообще, все существующие, используемые на практике криптосистемы на открытых ключах так или иначе используют простые числа. Это является непременным атрибутом существующих схем.