Математические структуры и моделирование
2011, вып. 22, с. 131–138

УДК 004.056.55

ПРАКТИЧЕСКИЕ АСПЕКТЫ ГЕНЕРАЦИИ КЛЮЧЕЙ ДЛЯ КРИПТОСИСТЕМЫ ЭЛЬ-ГАМАЛЯ

Е. А. Илюшечкин, А. А. Лаптев

Рассматриваются оптимальные методы выработки ключей шифрования для криптосистемы Эль-Гамаля. Приводится результат практической реализации этих методов.

Введение

В наше время большое практическое значение имеет асимметричная криптография (или криптография с открытым ключом). Одной из популярных криптосистем с открытым ключом является криптосистема Эль-Гамаля, реализованная в таких криптографических пакетах, как PGP и GnuPG.

Весьма важной частью реализации любой криптосистемы является генерация ключей. С одной стороны, неправильно подобранный ключ может серьёзно снизить безопасность системы и, следовательно, практическую применимость данной её реализации. С другой стороны, желательно минимизировать время, затрачиваемое как на генерацию самих ключей, так и на операции, производимые с помощью этих ключей в дальнейшем. К сожалению, в литературе зачастую трудно найти единый комплекс требований, обеспечивающий оптимальную стратегию выбора ключей. Цель данной работы — составить подобный комплекс практических рекомендаций, исследовав различные методы теоретически и сравнив их экспериментально.

1. Теоретический подход к выбору ключей

1.1. Модуль p

Наиболее ресурсоёмким этапом генерации ключей исследуемой криптосистемы (её описание, а также обозначения, принятые в данной работе, можно найти в [1]) является нахождение большого простого числа p. Выбор p ограничен условием: число $q = \frac{p-1}{2}$ должно иметь хотя бы один большой простой делитель. На практике обычно используют более сильное условие: q не должно иметь малых

Омский государственный университет им. Ф. М. Достоевского.
E-mail: ilush2005@yandex.ru

простых делителей вовсе [2, 4]. Было выделено два класса простых чисел, удовлетворяющих второму варианту условия. Первый класс известен в литературе как безопасные простые числа. Числа второго класса для удобства нами были названы «оптимальными», так как на текущий момент выглядят сравнимыми с безопасными по защищённости от известных атак, но время их поиска гораздо меньше.

Простое число p называется безопасным (англ. *safe prime*), если число $\frac{p-1}{2}$ также простое. Безопасные числа достаточно редки. В связи с этим были исследованы методы ускорения поиска таких чисел, в том числе идея об эффективном просеивании кандидатов из [8]. На основе этих методов нами был составлен алгоритм генерации простого числа q, для которого $p = 2q + 1$ заведомо будет простым, т. е. безопасным. Алгоритм описан в табл. 1; через P_B обозначено множество всех простых чисел, меньших B.

Таблица 1

Алгоритм генерации q

1. Выбрать случайное число $q \equiv 5 \ (mod\ 6)$ требуемого размера.
2. Если $\exists\ r \in P_B : q \in \{0, \frac{r-1}{2}\} \ (mod\ r)$, перейти к шагу 5.
3. Если $2^{p-1} \not\equiv 1 \ (mod\ p)$, перейти к шагу 5.
4. Если q проходит тест Рабина–Миллера, вернуть q, алгоритм завершён.
5. Увеличить q на 6 и перейти к шагу 2.

Алгоритм завершает работу за конечное число шагов, поскольку, начиная с некоторого числа, последовательно в порядке возрастания перебирает всех возможных кандидатов и гарантированно распознает первое же число q, для которого $2q+1$ является безопасным простым. Справедливость этого алгоритма базируется на следующих утверждениях.

Теорема 1. *Пусть* $p = 2q + 1$, $r \neq 2$ — *простое. Тогда* $p \equiv 0 \ (mod\ r) \Leftrightarrow q \equiv \frac{r-1}{2} \ (mod\ r)$.

Доказательство. Двойка обратима по модулю r, поэтому:

Достаточность. $p = 2q + 1 \equiv 2\frac{r-1}{2} + 1 \equiv r \equiv 0 \ (mod\ r)$.

Необходимость. $q = 2^{-1}(p-1) \equiv \frac{r+1}{2}(r-1) \equiv \frac{r-1}{2} \ (mod\ r)$. ■

Теорема 2. *Пусть* $p = 2q+1$, q — *простое. Тогда* p — *простое* $\Leftrightarrow p \not\equiv 0 \ (mod\ 3)$, $2^{p-1} \equiv 1 \ (mod\ p)$.

Доказательство. По теореме Поклингтона: если существует простое $q \mid (p-1)$:

1. $q > \sqrt{p} - 1$

2. $\exists a : (a^{p-1} \equiv 1 \ (mod\ p)) \wedge ((a^{\frac{p-1}{q}} - 1,\ p) = 1)$

$\Rightarrow p$ — простое.

Необходимость. Когда $p = 2q + 1$, первое условие теоремы выполнено для любого положительного q, а $2^{\frac{p-1}{q}} - 1 = 3$. Если $2^{p-1} \equiv 1 \ (mod\ p)$ и $p \not\equiv 0 \ (mod\ 3)$, то второе условие также выполнено для $a = 2 \Rightarrow p$ — простое.

Достаточность. Если же p — простое, то $2^{p-1} \equiv 1 \ (mod\ p)$ в силу малой теоремы Ферма и очевидно, что $p \not\equiv 0 \ (mod\ 3)$. ■

Так как простота q опирается в итоге на результат теста Рабина–Миллера (стандартный подход для современных методов генерации простых чисел), алгоритм является вероятностным и с крайне низкой вероятностью ошибается. Если бы простота p проверялась также с помощью теста Рабина–Миллера, то вероятность ошибки была бы выше. Использование для числа p теста Поклингтона (который является детерминированным и не ошибается) на третьем шаге алгоритма позволяет не только ускорить поиск чисел, но и снизить вероятность ложного результата.

Кроме оптимизации алгоритма, поиск безопасных простых чисел можно улучшить с помощью следующих приёмов.

1. Реализация теста Рабина–Миллера должна учитывать размер числа. Чтобы подобрать минимальное число итераций для заданного размера ключа, можно использовать табл. 4.4 из [5].

2. Желательно подобрать оптимальную границу B. Можно для этого использовать замечание 4.45 об оптимальной границе для пробных делений из [5].

3. Размер таблицы малых простых чисел можно заметно сократить, храня разность между соседними числами, вместо самих чисел.

4. При нахождении остатков от деления на малые простые числа можно использовать остатки, вычисленные для предыдущих кандидатов.

5. Поскольку значение i-го кандидата легко вычислить, последовательность можно разбить на блоки и производить проверку в каждом блоке на отдельном процессоре.

Простое число p называется «оптимальным» (в рамках данной статьи), если $p = 1 + 2 \cdot q_1 \cdot ... \cdot q_n$, где все q_i $(i = 1, ..., n)$ — простые числа, имеющие битовый размер не менее q_{size}. Для оптимальных чисел q_{size} может быть найден по табл. 2. Чтобы ускорить поиск таких чисел, можно использовать идею перебора сочетаний из пула простых чисел, описанную в [4]. Процесс генерации выглядит следующим образом. Сначала вычисляется n — число чисел q_i. Затем генерируется $m > n$ (в нашей программе использовалось $m = 3n$) простых чисел. Теперь последовательно выбираются всевозможные сочетания по n чисел из m. Из каждой такой выборки составляется соответствующее число p, для которого проверяется соответствие заданному размеру и, если размер правильный, p проверяется на простоту.

Если p прошло проверки, то искомое число найдено, иначе делается следующая выборка, либо генерируется новый пул, если все возможные сочетания были исчерпаны. Чтобы ускорить этот процесс, можно не генерировать весь пул сразу: каждый элемент пула нужно генерировать только тогда, когда он впервые понадобится для некоторой выборки.

Таблица 2

Таблица Винера

p_{size}	512	768	1024	1280	1536	1792	2048	2304	2560	2816	3072
q_{size}	119	145	165	183	198	212	225	237	249	259	269

1.2. Порождающий элемент g

Алгоритмы нахождения порождающего элемента Z_p^* (мультипликативной группы поля вычетов) и её подгруппы можно найти в [5] (номера алгоритмов — 4.80 и 4.81). Для порождающего элемента полной группы необходимо знать разложение числа $p-1$ на простые множители: $p-1 = q_1^{a_1} \cdot q_2^{a_2} \cdot ... \cdot q_n^{a_n}$. Это не является помехой для оптимальных или безопасных чисел, поскольку описанные выше методы их поиска гарантируют, что $p-1$ будет всегда построено из известных простых сомножителей. Для безопасных чисел, кроме того, шаг 2 алгоритма можно заменить на однократную проверку символа Лежандра $(\frac{g}{p})$, который вычисляется на порядок быстрее, чем возведение в степень.

Двойка с точки зрения производительности является лучшим значением для порождающего элемента, поскольку при эффективной реализации возводится в степень по модулю быстрее, чем другие числа [6]. В предложенном алгоритме генерации q для безопасных чисел можно легко исключить из рассмотрения кандидатов, для которых двойка не является порождающим элементом требуемой подгруппы. Для этого нужно увеличивать q не на 6, а на 12 и выбрать на первом шаге случайное число $q \equiv 5 \ (mod\ 12)$ или $q \equiv 11 \ (mod\ 12)$, чтобы 2 имела порядок $2q$ или q соответственно. Такая модификация объясняется следующим утверждением.

Теорема 3. *Пусть $p = 2q + 1 > 5$ — безопасное простое. Тогда 2 имеет в Z_p^* порядок $2q$, если $q \equiv 5 \ (mod\ 12)$, и порядок q, если $q \equiv 11 \ (mod\ 12)$.*

Доказательство. $(\frac{2}{p}) = (-1)^{\frac{p^2-1}{8}} = ((-1)^q)^{\frac{q+1}{2}} = (-1)^{\frac{q+1}{2}}$;

$q \equiv 5 \ (mod\ 12) \Rightarrow \frac{q+1}{2} \equiv 1 \ (mod\ 2) \Rightarrow (\frac{2}{p}) = -1$;

$q \equiv 11 \ (mod\ 12) \Rightarrow \frac{q+1}{2} \equiv 0 \ (mod\ 2) \Rightarrow (\frac{2}{p}) = 1$.

$(\frac{2}{p}) = 2^{\frac{p-1}{2}} = 2^q \ (mod\ p)$. Порядок должен делить $p-1$, поэтому для всех вычетов $2, ..., p-2$ он равен q или $2q$. Соответственно,

$q \equiv 5 \ (mod\ 12) \Rightarrow 2^q \equiv -1 \not\equiv 1 \ (mod\ p) \Rightarrow$ порядок равен $2q$;

$q \equiv 11 \ (mod\ 12) \Rightarrow 2^q \equiv 1 \ (mod\ p) \Rightarrow$ порядок равен q. ■

1.3. Закрытый ключ

Скорость операций в криптосистеме пропорциональна размеру секретной части ключа. Сделав размеры x и k минимально допустимыми для требуемого уровня безопасности, можно получить существенный выигрыш производительности. Но заметим, что секретная часть ключа должна быть достаточно большой, чтобы избежать возможности использования лямбда-метода Полларда. Оптимальный размер секретной части ключа — это значение $q_{size} \cdot C$, где q_{size} определяется из табл. 2 (размер p должен быть задан), а $C \geq 1$ — константа, задающая запас прочности.

1.4. Выбор группы

Другой важный этап генерации ключей — выбор рабочей группы. В предположении о неразрешимости проблемы DDH система является семантически бе-

зопасной (по известному шифртексту и открытому ключу невозможно получить значимую информацию об открытом тексте) только при работе в собственной подгруппе Z_p^* простого порядка.

Для безопасных чисел предлагается следующая семантически безопасная схема [7]. Пусть сообщение $M \in \{1, ..., q\}$. Шифруется $M_2 = M^2 \ (mod\ p)$. Восстановление M по M_2: Если $M' = (M_2)^{\frac{q+1}{2}} \ (mod\ p) > q$, то $M = -M' \ (mod\ p)$, иначе $M = M'$.

Отметим, что для расшифрования в данной схеме требуется дополнительное возведение в степень. В результате скорость расшифрования падает примерно вдвое. Кроме того, при уменьшении размера секретного ключа пропорционально повысится лишь скорость шифрования, тогда как скорость расшифрования почти не изменится. Другим очевидным недостатком является уменьшение в два раза пространства разрешённых сообщений, однако этот недостаток является менее существенным, поскольку на практике шифруемые сообщения, как правило, имеют намного меньшую длину, чем q (основная масса сообщений — симметричные ключи длиной 64–256 бит).

Для «оптимальных» чисел представить сообщение как элемент собственной подгруппы Z_p^* простого порядка q значительно сложнее. Манипуляции только лишь с сообщением на порядок увеличивают время шифрования, поэтому необходимо прибегать к модификациям самого процесса шифрования.

Одним из известных подходов, решающих проблему кодирования, является хэшированный вариант криптосистемы, описанный в [3]. Там же авторы предлагают более сложный и оригинальный метод. Необходимо отметить, что оба этих варианта нуждаются в некоторых дополнительных нестандартных предположениях. Такие предположения, в отличие от предположений Диффи–Хеллмана, не являются хорошо изученными и потому выглядят менее убедительными.

2. Практические результаты

Для проверки работоспособности изложенных методов на практике нами была написана программа для генерации ключей и шифрования. Эта программа использовалась в серии тестов, результаты которых можно видеть на представленных ниже графиках.

На рис. 1 сравнивается скорость поиска безопасных простых чисел в криптографическом пакете OpenSSL (функция BN_generate_prime_ex) и в нашей программе.

На рис. 2 сравнивается скорость поиска безопасных и оптимальных чисел. Как видно, оптимальные числа, в отличие от безопасных, могут генерироваться на лету для ключей приемлемого размера.

На рис. 3 сравнивается скорость шифрования и расшифрования для различных вариантов криптосистемы:

1. Классический вариант, работающий в полной группе Z_p^*;
2. Семантически безопасная схема для безопасных чисел;

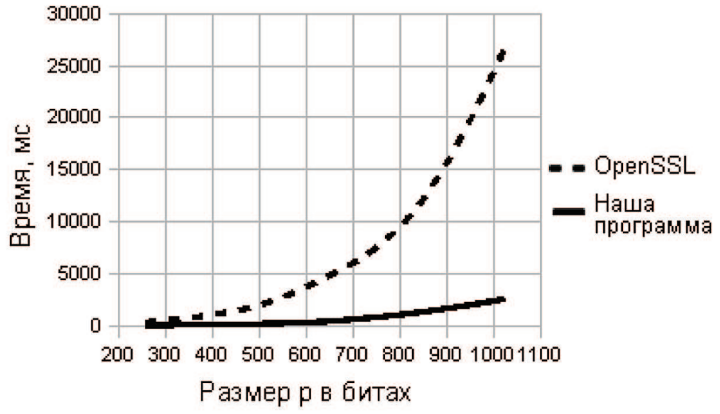

Рис. 1. Сравнение скорости генерации безопасных простых чисел в разных реализациях

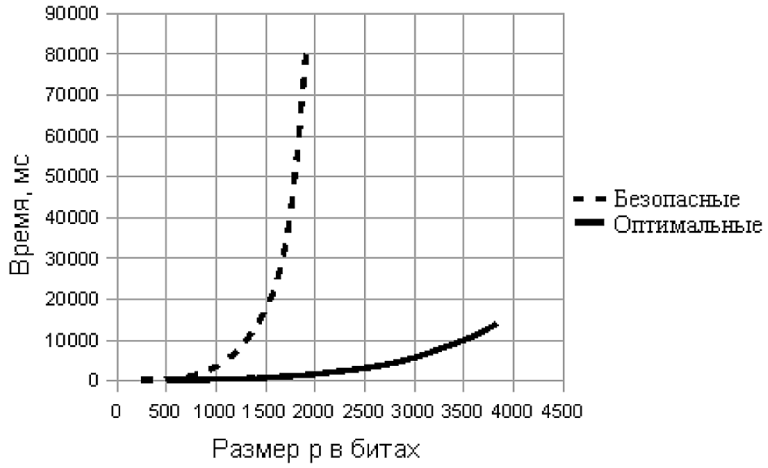

Рис. 2. Сравнение скорости генерации разных типов простых чисел

3. Хэшированный вариант криптосистемы для оптимальных чисел, использующий в качестве хэш-функции SHA.

Представленные на рис. 3 графики показывают зависимость времени шифрования и расшифрования одного сообщения от размера числа p. Поскольку асимметричные криптосистемы, как правило, применяются для шифрования довольно небольших симметричных сеансовых ключей, на практике время шифрования одного сообщения важнее времени шифрования фиксированного объёма информации. Размер подгруппы для оптимальных чисел выбирался по табл. 2. Секретная часть ключа везде выбиралась из всех возможных значений.

Как видно из графиков, скорости шифрования первых двух вариантов практически равны, а скорости расшифрования отличаются примерно вдвое, что сходится с ожидаемым результатом. Скорость шифрования и расшифрования для третьего варианта меньше примерно во столько же раз, что и размер используемой рабочей группы. Таким образом, вычисление хэш-функции по вре-

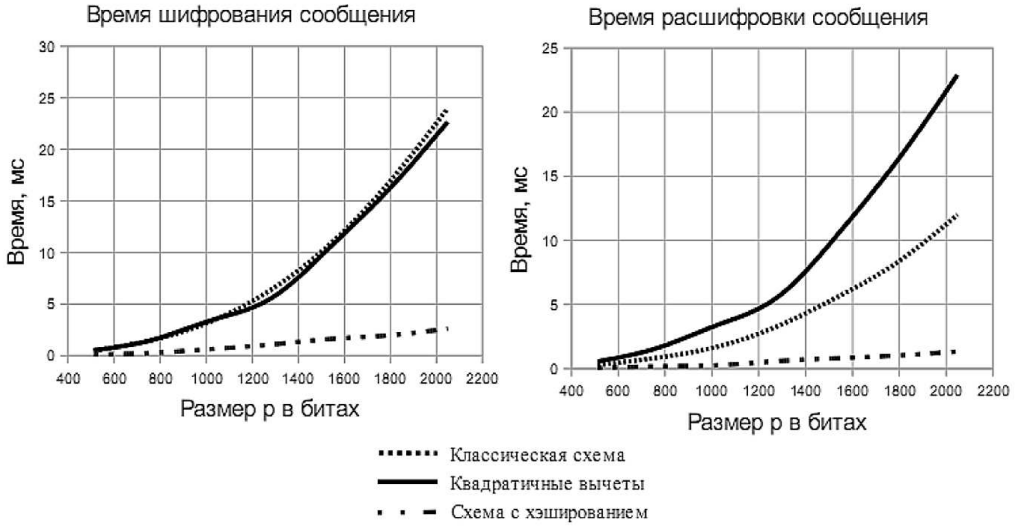

Рис. 3. Сравнение скорости генерации безопасных простых чисел в разных реализациях

мени сравнимо с умножением и заметно не влияет на скорость. Стоит отметить, что заметный отрыв графика для хэшированной схемы связан с тем, что возможный диапазон размера секретного ключа изначально гораздо меньше. Как было сказано ранее, для двух других схем секретный ключ также может выбираться из меньшего диапазона. В этом случае хэшированная схема не будет иметь преимущества по скорости.

Таким образом, при выборе между классическим вариантом криптосистемы и вариантом, основанным на подгруппах, следует решить, насколько системе нужна семантическая безопасность. Скорее всего, она будет нужна, если шифруемые сообщения очень специфичны. Если семантическая безопасность не важна, то лучше использовать классическую схему. Для достижения скорости, сравнимой при работе в подгруппах, нужно будет, разумеется, выбирать секретный ключ из минимального диапазона. Если же семантическая безопасность важна, то необходимо работать в подгруппах. В этом случае использовать описанную выше семантически безопасную схему для безопасных чисел следует в том случае, если приоритетом является долгосрочная безопасность, но не скорость расшифрования, которая будет заметно ниже. При отсутствии требования долгосрочной безопасности стоит работать с «оптимальными» числами, но в этом случае нужно хорошо продумать способ кодирования сообщений.

3. Заключение

В данной работе нами были проанализированы подходы к генерации ключей для базовой криптосистемы Эль-Гамаля и её варианта, работающего в подгруппе простого порядка группы Z_p^*. Основываясь на полученных результатах, мы предлагаем следующие правила для выработки оптимальных ключей с точки зрения безопасности и производительности:

1. В качестве модуля p лучше всего брать безопасные или «оптимальные» числа. Для эффективного поиска безопасных чисел выше предложен соответствующий алгоритм.
2. Генерация безопасных чисел занимает гораздо больше времени, чем генерация «оптимальных», поэтому первые следует генерировать заранее.
3. Для безопасных чисел есть возможность ускорения операций за счёт использования двойки в качестве порождающего элемента.
4. Поскольку размер секретного ключа сильно влияет на скорость операций в системе, мы рекомендуем выбирать секретный ключ минимально необходимого размера.
5. Выбрать рабочую группу в зависимости от требований семантической и долгосрочной безопасности.

Для дальнейшего изучения хотелось бы выделить следующие направления:

1. Вопросы кодирования для варианта, использующего подгруппы;
2. Исследование генерации ключей для варианта криптосистемы на эллиптических кривых.

Литература

1. Allen B. Implementing several attacks on plain ElGamal encryption. Iowa State University, 2008.
2. Boneh D., Joux A., Nguyen P. Q. Why textbook elgamal and rsa encryption are insecure // In ASIACRYPT '00: Proceedings of the 6th International Conference on the Theory and Application of Cryptology and Information Security. London: Springer, 2000. P. 30–43.
3. Chevallier-Mames B., Paillier P., Pointcheval D. Encoding-Free ElGamal Encryption Without Random Oracles // Public Key Cryptography – 2006. Berlin: Springer, 2006.
4. Lee C. H., Lee P. J. A Key Recovery Attack on Discrete Log-based Schemes Using a Prime Order Subgroup // Advances in Cryptology. CRYPTO, 1997.
5. Menezes A. J., Oorschot P. C., Vanstone S. A. Handbook of Applied Cryptography. CRC Press, 1997.
6. Oorschot P. C., Wiener M. J. On Diffie-Hellman Key Agreement with Short Exponents // Proc. Eurocrypt'96. 1996.
7. Reyzin L. Diffie-Hellman, ElGamal, and a Bit of History // Lecture Notes for Boston University CAS CS 538. 2004.
8. Wiener M. J. Safe Prime Generation with a Combined Sieve. 2003.