Черные ходы в криптографии с открытым ключом

Юрий Решетов

Существует далеко не праздный вопрос о потенциальной уязвимости шифров для посторонних лиц. Ведь, если таковая существует, то значит, ею могут воспользоваться. Например: вездесущие спецслужбы или не менее вездесущие хакеры и крякеры.

В данной статье, мы рассмотрим вопрос о том, как можно слегка модифицировать алгоритмы с открытым ключом, а именно генерации ключей, с целью внедрения в них так называемых «черных ходов», посредством которых посторонние лица легко могут получить доступ к секретной информации.

Общеизвестно, что основным мерилом надежности шифров для большинства обывателей служит длина ключа, посредством которого можно вскрыть криптограмму. Ведь, достаточно взглянуть на показатель криптостойкости, например, 1024 битного ключа RSA, как на душе становится спокойно.

Более умудренные в криптографии специалисты, также обратят внимание еще на ряд факторов, таких, как генерация открытых ключей на изолированных от внешнего мира компьютерах, уничтожении остатков временных файлов после генерации, чистке буфера обмена, а также на порядок хранения открытых ключей.

Но даже все эти факторы не являются достаточными для защиты секретной информации, особенно, если используется криптография с открытым ключом. А причиной тому, является тот факт, что для генерации открытых и закрытых ключей, как правило, используются уже готовое программное обеспечение. И никто не задумывается над тем, что в это самое программное обеспечение можно добавить некий код, который позволит впоследствии свести на нет все предыдущие усилия, направленные на защиту информации. Причем, нет никакой разницы, какой длины ключи, насколько они соответствуют требованиям защиты и каковы были предприняты меры, для защиты закрытых ключей. Для спецслужб внедрение черных ходов является наиболее простой задачей, ведь все производители легального программного обеспечения должны обращаться в компетентные органы, дабы сертифицировать свою продукцию. А, следовательно, при отказе от внедрения в код программ «черных ходов» им может быть также отказано в этой самой сертификации. Самое печальное, что на месте не стоят и злоумышленники, например, крякеры, которые, в отличие от рядовых пользователей не обращают внимания на наличие сертификатов, а занимаются исследованием машинных кодов программ. Следовательно, если в программе генерации открытых ключей будет присутствовать «черный ход», пусть даже и внедренный туда с самыми «благими намерениями» то, значит, существует потенциальная опасность, что этой лазейкой могут воспользоваться и те, кто данную «дыру» изучит. Не следует также забывать и о хакерах. Воспользовавшись доступом к программному обеспечению, они также могут модифицировать модули, отвечающие за безопасность компьютера, например, подменив разделяемые библиотеки, с помощью которых генерируются открытые ключи. Причем сделать они это могут как через сетевой доступ, так и через внедренный вирус или же компьютерную программу, которую пользователь скачает с некоего сайта. Результат в любом случае будет малоприятным.

Для начала рассмотрим алгоритм обмена ключей Диффи – Хелмана (на аналогичном алгоритме основана система шифрования Эль – Гамаля). Так называемая функция дискретного возведения в степень. Принцип ее основан на том, что открытый ключ можно сгенерировать по нижеследующему алгоритму:

1. Генерируем большое простое число p
2. Выбираем произвольное число g (При условии: 1 < g < p).
3. Выбираем произвольное большое число a (при условии: 1 < a < p), которое будет закрытым ключом.
4. Вычисляем y = g^a mod p
5. Публикуем открытый ключ из трех значений. p, g и y.

Если некто захочет обменяться с владельцем открытого ключа, неким ключом для любой симметричной системы шифрования, то ему нужно будет:

1. Выбрать произвольное число k (При условии: 1 < k < p).
2. Вычислить y1 = g^k mod p
3. Передать любым способом y1 владельцу открытого ключа.
4. Вычислить ключ шифрования key = y^k mod p

В этом случае владелец открытого ключа, также может вычислить тот же самый ключ шифрования key = y1^a mod p.

Алгоритм даже не имеет нижней оценки криптостойкости, а сие означает, то, что единственным методом решения обратной задачи для взлома, является полный перебор всех возможных вариантов.

Зато для внедрения «черного хода» данный алгоритм уязвим до безобразия. Если некто вставит в генератор ключей небольшое дополнение, суть которого заключается в вычислении a = f(g), где f() – некая функция, известная злоумышленнику, то результат налицо. В качестве f(g), например, можно взять некую небольшую константу b и добавить в код программы: a = g^b mod p.

Уязвимости открытых ключей RSA

Шифрование RSA основано на том факте, что: x^((p – 1)(q – 1) + 1)*y mod p*q = x. Где x, y – натуральные числа, а p и q – простые. Для шифрования используются два числа: d и e, которые связаны между собой соотношением: d * e = ((p – 1)(q – 1) + 1)*y. Одно из этих чисел – d, является вместе со значением n = p*q открытым ключом, а второе закрытым. Тот, кто захочет отправить некое сообщение x, вычисляет z = x^e mod n. Владелец закрытого ключа может получить отправленное скрытое таким макаром сообщение: x = z ^ e mod n. Для тех, кто не знает секретного числа e, чтобы вскрыть сообщение, проще всего факторизовать (разложить на простые сомножители: p и q) число n. Но процесс сей весьма затруднителен, по причине экспоненциальной сложности.

Есть множество различных методов факторизации целых чисел. Средь которых следует отметить разработанный Джоном Поллардом «p – 1». Суть метода заключается не в факторизации числа n на простые сомножители, а в разложении значений p – 1 и q – 1. Принцип легко объяснить с помощью малой теоремы Ферма: если p – простое, тогда p | x ^ (p – 1) – 1. Также: p | x ^ a*(p – 1) - 1. Следовательно, если p | n, то (n, x ^ a*(p – 1) – 1) = p. Классический метод, разработанный самим Поллардом подразумевает искать значения a простым полным перебором. И подставлять в базу: base = base ^ a mod n. После чего, время от времени проверять базу: (base – 1, n). Если результат проверки не 1, то значит

найден нетривиальный делитель n. (Есть различные, незначительные усовершенствования данного алгоритма).

А что если для метода «p – 1» брать значения не полным перебором всех возможных вариантов, а отсеивать заведомо непригодные. Тогда мы приходим к методу Полларда – Решетова. Суть метода заключается в том факте, что значение d открытого ключа имеет особенность: (d, (p – 1)*(q – 1)) = 1, то бишь полное отсутствие общих сомножителей. Следовательно, можно запросто отсеивать все сомножители d, а остальные класть в базу. Самый простой способ подбора кандидатов a в базу можно получить по формуле: a = x * d – 1. Более изощренный: a = d * (d^(-1) mod b) – 1. Также, не будут иметь общих сомножителей с d все d * x – y, если (y, d) = 1.

Нетрудно доказать, что помимо эффекта отсеивания, любые сомножители p – 1 и q – 1, будут найдены таким макаром гораздо раньше, нежели, при полном переборе. (Вероятно, что некий сомножитель z попадет в базу при использовании метода Полларда - Решетова с такой же скоростью, что и при полном переборе равна 1/z).

Налицо еще и прекрасная возможность использования параллельных вычислений, которая исключена в классическом «p – 1».

Данный метод также можно применять и для факторизации всех чисел, для которых известно отсутствие тех или иных сомножителей функции Эйлера, хотя не известен открытый ключ d. Например, для чисел, являющихся произведением сильнопростых сомножителей, в качестве d, можно брать произведение сомножителей мелких (но обязательно нечетных): d = 3*5*7*11*…

Что касается уязвимости для внедрения «черных ходов», то алгоритм генерации ключей позволяет это сделать.

1. Генерируем два простых числа p и q
2. Вычисляем n = p * q
3. Вычисляем a = f(n). Т.е. также, как и для «черного хода» в генерации ключей Диффи – Хелмана.
4. Вычисляем открытый ключ: d = a^(-1) mod (p – 1)
5. Если d четно, то вернуться на п. 1
6. Вычисляем закрытый ключ e = d^(-1) mod (p – 1) * (q – 1)
7. Если закрытый ключ не удалось вычислить, то вернуться на п. 1
8. Выдаем результат: d, e, n.

Теперь, если владелец закрытого ключа опубликует открытый, то можно выполнить примитивную факторизацию числа n.

$$p = (x^{(f(n)*d - 1)} - 1 \bmod n, n)$$

Но, тот, кто сгенерировал ключи, может поменять местами открытый - d и закрытый - e ключи. Тогда после факторизации получим p = 1.

В этом случае:

$$p = (x^{(e - f(n))} - 1, n)$$