УДК 519.24

М. И. ГЛАВЧЕВ, канд. экон. наук, НТУ "ХПИ" (г. Харьков),
Ю. Ю. ДРОЗДОВ, НТУ "ХПИ" (г. Харьков)

К ВОПРОСУ ГЕНЕРАЦИИ КЛЮЧЕЙ ДЛЯ АСИММЕТРИЧНЫХ КРИПТОСИСТЕМ

З метою відпрацьовування методів генерації ключей для асимметричних криптосистем було проаналізовано схеми керування ключами. Були розглянуті зменшення простіру імовірних ключей, об'єднанний вибір ключей, розкриття зі словарем, випадкові ключі та стандарт генерації ключей. Було зроблено висновки щодо передачі ключей, перевірки ключей, використання ключей. Уділено увагу також зберіганню ключей, резервним ключам та часу їхнього життя.

With the purpose of improvement of methods of generation of keys for asymmetric cryptosystems the circuits of management of keys were analyzed. The casual keys, use of keys were considered reduction of the standard of probable keys incorporated choice of keys, opening with the dictionary. The attention also to savings of keys, reserve keys and time of their life is given.

Постановка проблемы. Безопасность алгоритма сосредоточена в ключе. Если вы используете криптографически слабый процесс для генерации ключей, то ваша система в целом слаба. Самые надежные алгоритмы, как правило, общедоступны, однако ключи зашифрованных сообщений их можно раскрыть только путем прямого перебора всех возможных значений.

Гораздо труднее сохранить секрет ключей. Ошибки в схемах их распространения, а также небрежное хранение становятся причиной того, что взломщик находит доступ к закрытой информации. Возникает задача управления ключами.

К основным этапам процесса управления ключами можно отнести:

- генерацию ключей,
- передачу ключей,
- использование ключей,
- проверку ключей,
- обновление ключей,
- хранение ключей,
- уничтожение ключей.

Вопросу генерации (создания) ключей для асимметричных криптосистем и посвящена эта статья.

Анализ литературы. С помощью алгоритма *RSA* задача генерации ключей решается путем выбора двух больших, длиной не менее 512 бит, простых чисел p и q. Затем вычисляется их произведение $n = pq$ и выбирается случайное число e, таким образом, что e взаимно просто с $(p-1)(q-1)$.

Также вычисляется число d, такое, что $de = 1 \bmod (p-1)(q-1)$. Эта операция делается с помощью расширенного алгоритма Евклида [1].

Стандарт ANSI X9.17 определяет способ генерации ключей, который не создает запоминающиеся ключи, и больше подходит для генерации сеансовых ключей или псевдослучайных чисел в системе [2]. Для генерации ключей используется криптографический алгоритм DES, но он может быть легко заменен любым другим алгоритмом.

Пусть $E_k(X)$ – это X, зашифрованный DES ключом K, специальным ключом, предусмотренным для генерации секретных ключей; V_0 – секретная 64-битовая стартовая последовательность; T – метка времени. Для генерации случайного ключа R_i вычислим

$$R_i = E_k(E_k(T_i) \oplus V_i). \quad (1)$$

Для генерации V_{i+1} вычислим:

$$V_{i+1} = E_k(E_k(T_i) \oplus R_i). \quad (2)$$

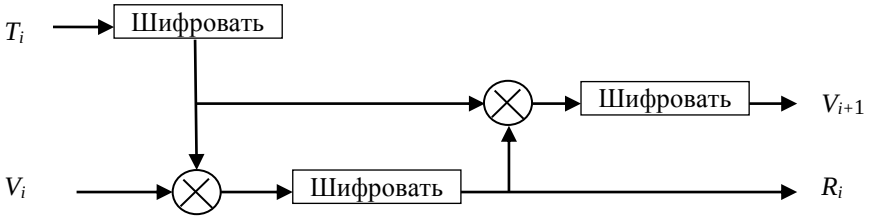

Рис. 1. Генерация ключей ANSI X9.17

Цель статьи. Изложение и оценка методов генерации ключей, а также описание процедуры управления ключами.

При генерации ключей для асимметричных криптосистем, а также для симметричных криптосистем следует избегать уменьшения пространства ключей, которое может возникнуть при использовании только ASCII символов. Подбирая ключи самостоятельно, люди, как правило, останавливают свой выбор на заведомо «ущербных» ключах. Наиболее эффективно против ключей и паролей, выбранных человеком, используется интеллектуальное вскрытие, которое состоит не в переборе всех возможных ключей по порядку, а в том, что сначала взломщик пробует наиболее очевидные ключи. Аналогично работает процедура, которая использует для перебора словарь наиболее вероятных ключей, и называется поэтому «вскрытие со словарем». Вскрытие со словарем намного эффективнее, когда оно направлено не против одного ключа, а против целого файла ключей.

Хорошими ключами считаются строки случайных битов, созданные определенным автоматическим процессом. Наилучшим решением является использование вместо слова целой фразы (ключевой фразы), преобразованной в ключ. Для этого существует методика «перемалывания

ключа». Для преобразования текстовой строки произвольной длины в последовательность псевдослучайных битов используется однонаправленная хэш-функция. Хэш-функции преобразуют строку переменной длины в строку фиксированной, обычно меньшей длины, называемую значением хэш-функции. Значение однонаправленной хэш-функции легко вычислить по прообразу, но трудно создать прообраз, значение хэш-функции которого равно заданной величине.

Стандарт X9.17 [2] определяет два типа ключей: ключи шифрования ключей и ключи данных. Ключами шифрования ключей при распределении шифруются другие ключи. Ключи данных шифруют сами сообщения. Ключи шифрования ключей должны распределяться вручную (хотя они могут быть в безопасности в защищенном от взлома устройстве, таком как кредитная карточка), но достаточно редко. Ключи данных распределяются гораздо чаще. Другим решением проблемы распределения является разбиение ключа на несколько различных частей и передача их по различным каналам. Одна часть может быть послана телефоном, другая – почтой, третья – службой доставки, четвертая – электронной почтой и т. д. (рис. 2).

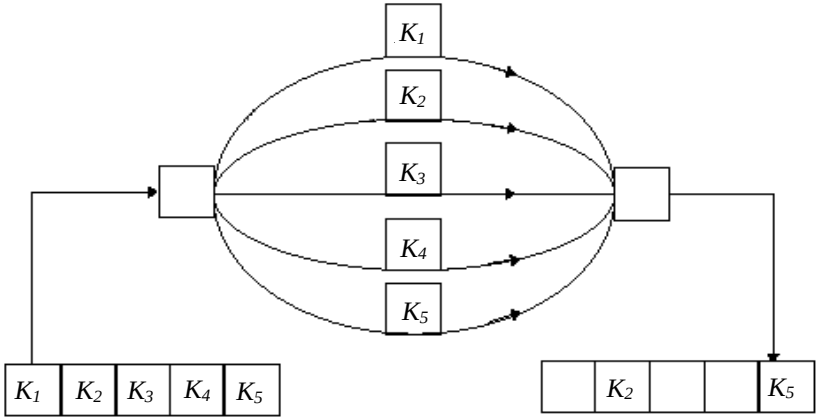

Рис. 2. Распределение ключей по параллельным каналам

Иногда ключи искажаются при передаче. Это является проблемой, так как искаженный ключ может привести к мегабайтам нерасшифрованного шифротекста. Все ключи должны передаваться с обнаружением ошибок и исправлением битов. Таким образом, ошибки при передаче могут быть легко обнаружены и, если потребуется, ключ может быть послан еще раз. Программное шифрование рискованно. Ушли те дни, когда простые микрокомпьютеры работали под управлением единственной программы. Сегодня время Macintosh System 7, Windows NT и UNIX. Невозможно

сказать, когда операционная система остановит работающую программу шифрования, запишет все на диск и разрешит выполняться какой-то другой задаче, и когда операционная система, наконец, опять вернется к шифрованию. Операционная система записывает программу шифрования на диск, и ключ записан зачастую вместе с ней. Ключ, незашифрованный, может лежать на диске, пока компьютер не напишет что-нибудь в эту же область памяти поверх него. Это может случиться на протяжении нескольких минут, а может через несколько месяцев. Этого может и никогда не случиться, но ключ все же может оказаться на диске в тот момент, когда жесткий диск исследуется противником. Аппаратные реализации безопаснее. Многие из устройств шифрования разработаны так, чтобы любое вмешательство приводило бы к уничтожению ключа. Например, в плате шифрования для IBM PS/2 залитый эпоксидной смолой модуль содержит микросхему DES, батарею и память. Конечно, Вы должны верить, что производитель аппаратуры правильно реализовал все необходимые свойства [3].

Для создания и хранения нового ключа предлагается следующий метод. Формирование ключа происходит из фрагмента или фрагментов известного получателю графического файла. Эти фрагменты могут выбираться псевдослучайным образом или задаваться отправителем и получателем в явном виде. Этого вполне достаточно для симметричных криптосистем. Для систем с открытыми ключами выполняется проверка полученных значений на соответствие используемым алгоритмам шифрования. Разрабатываемая система генерации ключей позволит в полной мере реализовать эти функции.

Учитывая размеры графических файлов и многозначность выбора необходимых фрагментов и их длин, предполагается, что хранение ключевой информации допустимо на винчестере, а атака на такой ключ будет приравниваться к атаке методом последовательного перебора, что не приведет к ослаблению алгоритма шифрования.

Выводы. Использование возможности хранения ключа на носителе информации в значительной степени упростит процесс сохранности ключа, а сама процедура его получения хорошо вписывается в программную реализацию.

Список литературы: **1.** *Schneier B.* Applied Cryptography. – N.Y.: John Wiley&Sons, 1994. **2.** *Жельников В.* Криптография от папируса до компьютера. – М.: Мир, 1996. **3.** *Саломаа А.* Криптография с открытым ключем. – М.: Мир, 1995. **4.** *Kahn D.* One-Way Hash Functions // Dr. Dobb's Journal. – 1991. – V. 16.– N 9. **5.** *Koblitz N.* A Course in Number Theory and Cryptography. – N.Y., 1994. **6.** *Adams C.M.* Simple and effective key scheduling for symmetric ciphers. – Ontario, 1994. **7.** *Мираян А.В.* Секреты криптографии. – М.: Мир, 1999. **8.** *Adleman L.* On breaking the iterated Merkle-Hellman public key cryptosystem // Proceedings of the 15th ACM Symposium on the Theory of Computing. – 1983. – P. 402 – 415. **9.** *Brassard G.* A note on the complexity of cryptography // IEEE Transactions on Information Theory IT-25. – 1979. – P. 232 – 233.

Поступила в редакцию 24.09.2004