Жуков А.Е.

Криптосистемы со встроенными лазейками

Криптография, реализованная в системах типа «черный ящик», получила в последнее время широкое распространение – например, встроенные криптосистемы в сотовых телефонах или smart-картах. Однако, криптосистема, реализованная как «черный ящик» (т.е. когда пользователь имеет доступ только к информации на входе и выходе криптосистемы), может быть построена таким образом, чтобы предоставлять разработчику уникальные возможности для ее взлома. Основная стратегия состоит в том, чтобы модифицировать криптосистему таким образом, что информация о секретном ключе будет включаться в выходную информацию криптографического устройства. Такие модификации можно осуществить используя скрытый канал передачи информации, т.к. изменения не должны оказывать влияние на нормальную работу устройства. В работе рассмотрены как уже известные, так и новые подходы к построению криптосистем, содержащих скрытые каналы утечки информации.

ВВЕДЕНИЕ. Проблема существования лазеек в криптографических системах

В настоящее время современные общедоступные электронные сети связи используются для передачи самой различной, порой чрезвычайно ценной информации: для выполнения банковских операций, торговли, предоставления услуг, получения информации из баз данных, распределенной обработки и решения практических задач и т.д. Все это требует новых, более совершенных методов защиты информации. Множество коммерческих компаний предлагают разнообразные программно-аппаратные средства для решения задач информационной безопасности. На рынке криптографических услуг в большом количестве представлены программно и аппаратно реализованные криптографические алгоритмы, которые для пользователей зачастую представляются как «черные ящики». Это или аппаратные устройства шифрования, логика которых реализована на низком уровне, или пакет программ, часто без наличия исходных текстов. При отсутствии текстов исходных программ даже в программных продуктах сложно отследить конкретную структуру алгоритма. В свою очередь компании, производящие программное обеспечение для криптографической защиты данных, не заинтересованы, ввиду очевидных причин, в раскрытии и публичном распространении исходных кодов программ. Производители криптографических продуктов, желающие из понятных и, заметим, вполне честных соображений держать свои know-how в секрете, предпринимают всевозможные усилия, затрудняющие декомпиляцию кодов программ. Однако, даже в случаях, когда спецификация становится доступной для пользователя, последний весьма редко проверяет соответствие имеющегося в его распоряжении продукта официальной

документации. Таким образом, пользователи зачастую получают лишь иллюзию защиты. Используя алгоритм, принцип работы которого известен только его разработчику, пользователь располагает единственной гарантией стойкости – утверждением самого разработчика о надежности алгоритма. В то же время разработчики готовых программных продуктов объективно имеют возможность встроить лазейки в алгоритмы шифрования по своей инициативе или заказу конкурентов.

Криптография, реализованная в системах типа «черный ящик», открыто поддерживается и используется правительством США. Также она широко распространена например, в виде встроенных криптосистем в сотовых телефонах или смарт-картах. Однако, в ряде работ было продемонстрировано, что криптосистема, реализованная как «черный ящик» (т.е. когда пользователь имеет доступ только к информации на входе и выходе криптосистемы, реализованной программно или аппаратно) может быть построена таким образом, чтобы предоставлять разработчику уникальные возможности для ее взлома. Как известно, общепризнанные системы, такие как RSA, ElGamal, схемы электронной подписи и целый ряд других криптографических примитивов широко и свободно применяются в самых различных системах обеспечения информационной безопасности. Последнее означает, что они включаются как средства обеспечения конфиденциальности, аутентификации, целостности и др. как в программные продукты, так и в smart-карты и прочее. Вопрос состоит в том, что может сказать пользователь о той конкретной реализации криптографического алгоритма, которая находится в его распоряжении? Насколько в действительности она его защищает? Насколько легко производителям программных продуктов и smart-карт встроить лазейку в свою продукцию так, что она останется незамеченной, но в то же время позволит производителю нарушить конфиденциальность пользователя?

Итак, основными вопросами, возникающими при пользовании криптографическими «черными ящиками» являются следующие:

- Предоставляет ли алгоритм недокументированные возможности и, в частности, содержит ли алгоритм незаявленные включения, позволяющие реализовать недокументированные возможности?
- Допускает ли он утечку секретной информации?
- Возникает ли риск для пользователя, в случае успешного реверс-инженеринга данного алгоритма третьей стороной?

СКРЫТЫЕ КАНАЛЫ ПЕРЕДАЧИ ИНФОРМАЦИИ

Вопрос о существовании лазеек в криптографическом алгоритме тесно связан с понятием скрытого канала передачи информации. Однако, сложившейся, общепринятой русской терминологии в этой области нет. Более того, используемая в настоящее время английская терминология также не может быть признана удачной. Предлагаемая ниже терминология, на наш взгляд, более удачно отражает сущность определяемых понятий.

Канал называется ***нестандартным*** (***нелегальным***) – (в английской терминологии – *hidden*, *covert*) , если он специально не проектировался и изначально не предполагался для передачи информации в электронной системе обработки данных. На практике, нестандартность канала утечки информации трактуется шире и рассматриваются не только нестандартные каналы передачи информации, но и нестандартные способы передачи информации по легальным каналам. В качестве примера таких каналов обычно приводятся следующие:

- Передача информации с помощью младших битов в файле с изображением (как известно, значения младших битов в кодировке изображения практически не влияют на его качество, но в то же время могут нести информацию.
- Пусть два абонента, между которыми отсутствует легальный канал связи, имеют возможность обращаться к одному и тому же ресурсу, при этом ресурс одновременно может обслуживать только одного пользователя. То есть, если в данный момент ресурс обслуживает, например, пользователя ***A***, пользователь ***B***, обратившийся с запросом к ресурсу, получит отказ. Тогда между пользователями ***A*** и ***B*** может быть установлен следующий нелегальный канал передачи информации. В фиксированные моменты времени пользователь ***A*** или занимает или не занимает ресурс. Пользователь ***B***, обращаясь в те же моменты времени к ресурсу, получает информацию о том занят ресурс или нет.

Ясно, что называть эти каналы скрытыми, тайными (hidden, covert) можно только при условии, что об их существовании ничего не известно. В противном случае третье лицо, имеющее информацию об этих каналах, будет получать всю информацию, проходящую по ним и даже иметь возможность изменять эту информацию по своему усмотрению. Поэтому, на наш взгляд, такие каналы уместнее называть нестандартными или нелегальными.

Нестандартный канал называется ***скрытым*** (в английской терминологии – *subliminal* – подсознательным, термин из психоанализа), если воспользоваться им может только

обладатель соответствующей информации. Неудачность термина «подсознательный канал», по-моему очевидна; в то же время такие каналы могут быть с полным основанием названы скрытыми.

ПРИМЕРЫ СКРЫТЫХ КАНАЛОВ ПЕРЕДАЧИ ИНФОРМАЦИИ

В открытой печати идея скрытых каналов передачи информации впервые появилась в работах Симмонса (Simmons [5], [6]) на примере скрытого канала в системе электронной подписи с открытым ключом. На рис. 1 изображена принципиальная схема работы системы электронной подписи. Для сообщения M_i с помощью секретного ключа d вырабатывается электронная подпись, которая совместно с исходным сообщением образует подписанное сообщение $c_i = (M_i s_i)$ – корректное подписанное сообщение. Получатель с помощью открытого ключа e проверяет соответствие подписи сообщению и, в случае их согласованности, сообщение M_i рассматривается как аутентичное. Если же получено некорректное подписанное сообщение c_j, то при проверке подпись и сообщение не будут соответствовать друг другу и тогда соответствующее сообщение M_j рассматривается как неаутентичное.

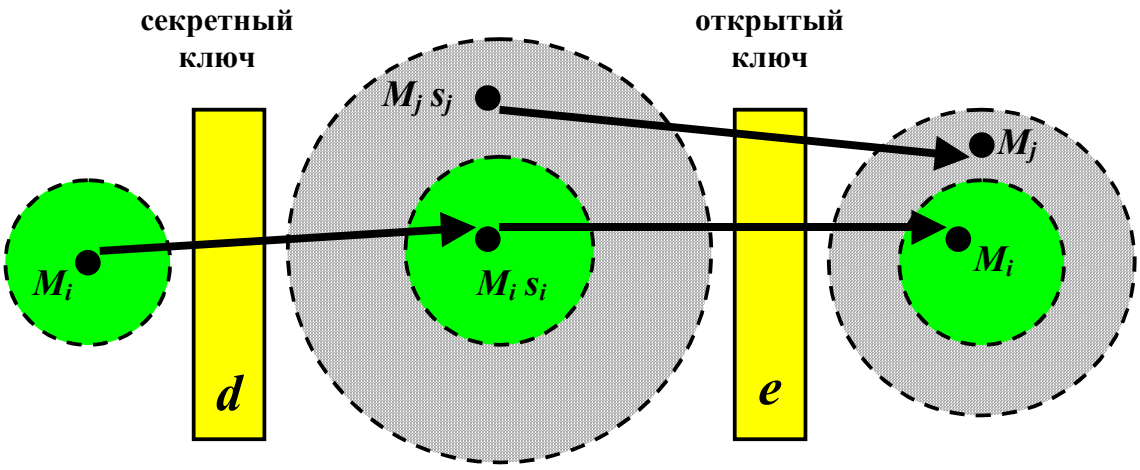
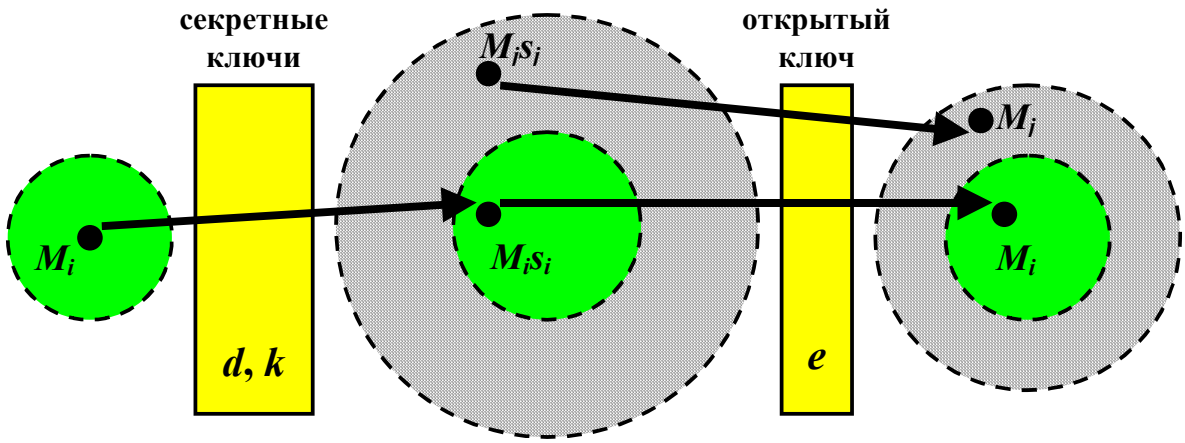


Рис.1

Однако, если внимательно рассмотреть алгоритмы выработки электронной подписи, можно сразу заметить, что в подавляющем большинстве из них помимо ключей e и d фигурирует также некоторое случайное число, выбираемое отправителем каждый раз для выработки одной конкретной подписи. Число это должно оставаться в секрете. Для

проверки корректности подписи знания этого числа не требуется. В то же время знание этого числа позволяет подделывать подпись. Таким образом, более точным будет утверждение, что в системе электронной подписи фигурируют *три* ключа: открытый ключ e и секретный ключ d, которые можно рассматривать как долговременные ключи и секретный разовый ключ k, участвующий в выработки данной подписи (см. рис. 2).



e – открытый ключ, d – секретный ключ } – долговременные ключи

k – *секретный разовый ключ*

Рис. 2

Тогда передача 1 бита информации может быть организована следующим образом. Отправитель и получатель заранее выбирают два числа k_0 и k_1. В зависимости от значения бита, который надо передать, отправитель формирует подпись с помощью разового ключа k_0 или k_1.

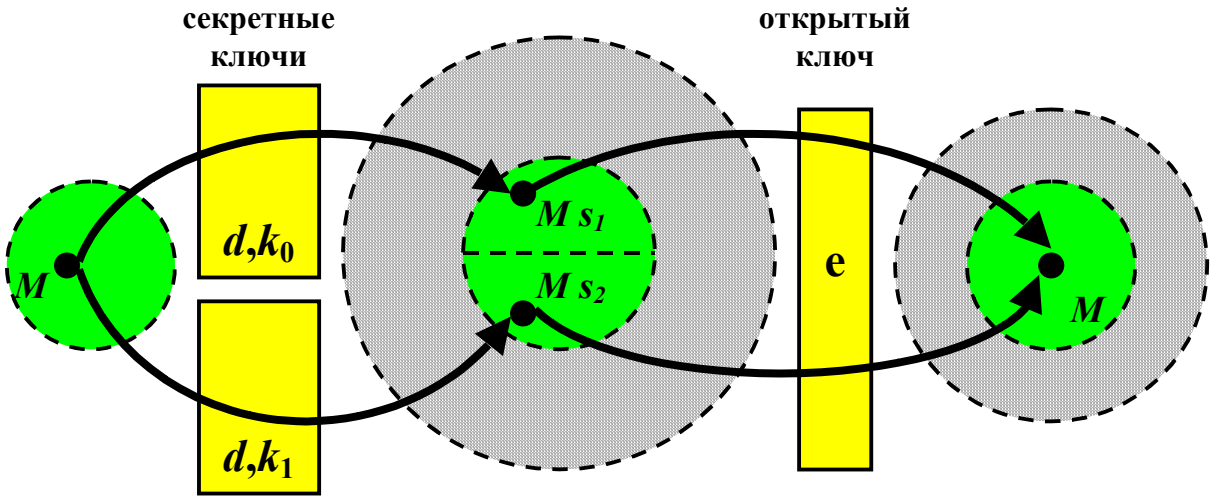


Рис. 3

И в том и в другом случае получается корректно выработанная подпись (см. рис. 3). Корректность этой подписи может проверить любой, знающий открытый ключ e. Никакой другой дополнительной информации он получить не сможет. В то же время получатель, знающий секретный ключ d и секретные разовые ключи k_0 и k_1 может напрямую проверить, с помощью какого из двух разовых ключей была получена данная подпись и, тем самым, получить дополнительную информацию в 1 бит (рис. 4).

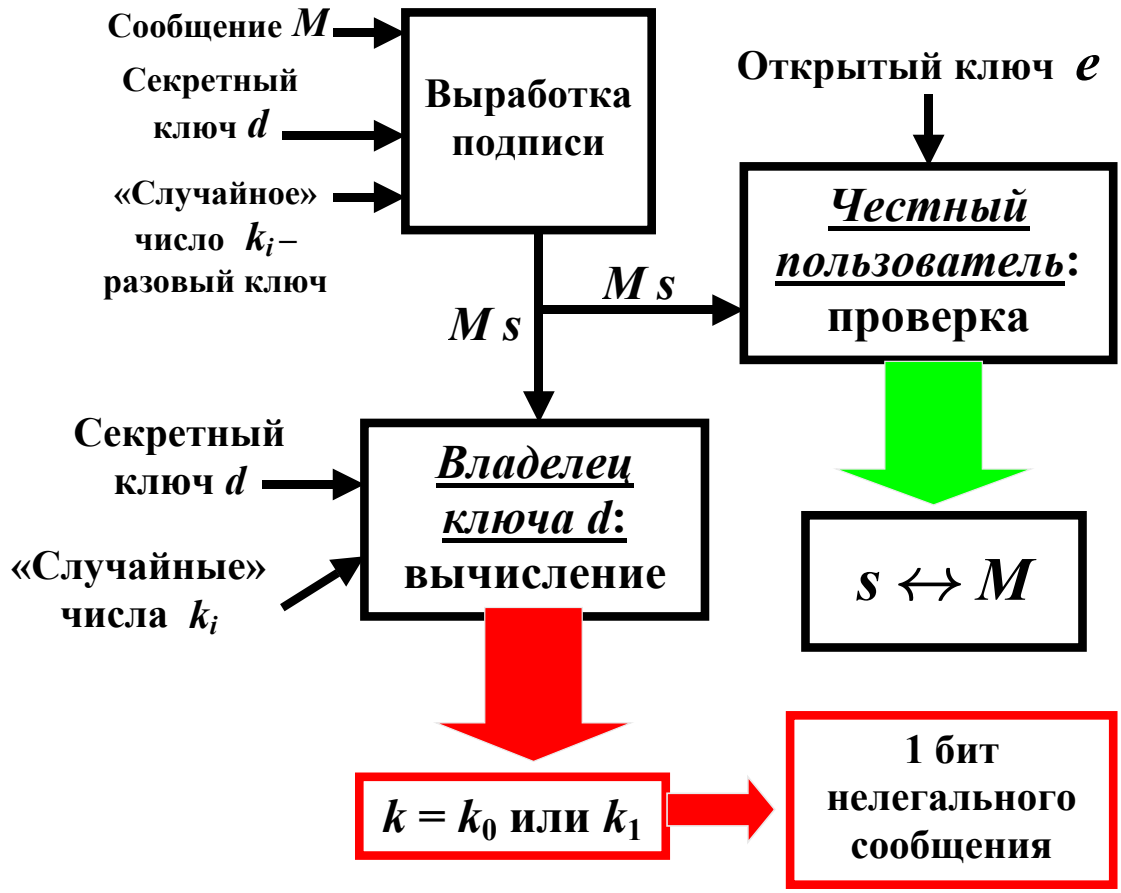


Рис. 4

Разумеется, приведенная схема непрактична. Она приведена исключительно для иллюстрации принципа работы скрытого канала в системе электронной подписи. Отметим, что данный канал существует *объективно*, для его существования не требуется никакого вмешательства в структуру алгоритма выработки электронной подписи.

Рассмотрим теперь некоторые примеры скрытых каналов, существующих в реальных системах электронной подписи.[*)]

Алгоритм DSA

Кратко работа алгоритма DSA может быть описана следующим образом [3]. Выбирается простое число p в диапазоне 512-1024 бит и простое число q длины 160 бит, делящее число $(p-1)$. Выбираются числа $d < q$ и $h < p-1$; после чего вычисляются

$$g = h^{\left(\frac{p-1}{q}\right)} \bmod p; \qquad e = g^d \bmod p.$$

Тогда открытый ключ – (p, q, g, e); секретный ключ – d.

Выработка подписи осуществляется следующим образом: пусть M – сообщение, $k < q$ – *случайное число* – секретный разовый ключ. Вычисляются величины

$$r = \left(g^k \bmod p\right) \bmod q; \quad s = \left(k^{-1}\left(H(M) + dr\right)\right) \bmod q,$$

где H – некоторая функция хэширования. Подписанным сообщением является набор (M, r, s).

Для проверки подписи вычисляются величины $w = s^{-1} \bmod q$, $u_1 = \left(H(M) \cdot w\right) \bmod q$, $u_2 = rw \bmod q$, $v = \left(g^{u_1} \cdot e^{u_2} \bmod p\right) \bmod q$. Подпись признается корректной, а сообщение аутентичным если выполняется равенство $v = r$.

Скрытый канал работает следующим образом [7]. В качестве случайного числа k берем само сообщение m, которое требуется передать по скрытому каналу: $k=m$.[**)] При известном секретном ключе d скрытое сообщение m легко может быть получено из легального подписанного сообщения (M, r, s) по формуле:

$$m = k = \left(s^{-1}\left(H(M) + dr\right)\right) \bmod q.$$

Заметим, что не зная d эти вычисления проделать невозможно, т.е. указанный канал действительно является скрытым.

[*)] Приводимые ниже примеры в некоторых деталях будут отличаться от приводимых в первоисточниках: нашей целью является демонстрация *механизма работы* скрытого канала. Опущенные же детали в основном обеспечивали полиномиальную неразличимость выходов (см. определение SETUP-механизма)

[**)] На практике, разумеется, в качестве k следует брать не сообщение m в чистом виде, а какое-то его взаимно однозначное преобразование, приближающее его статистические свойства к случайным (например, заархивированное сообщение или что-то в этом роде).

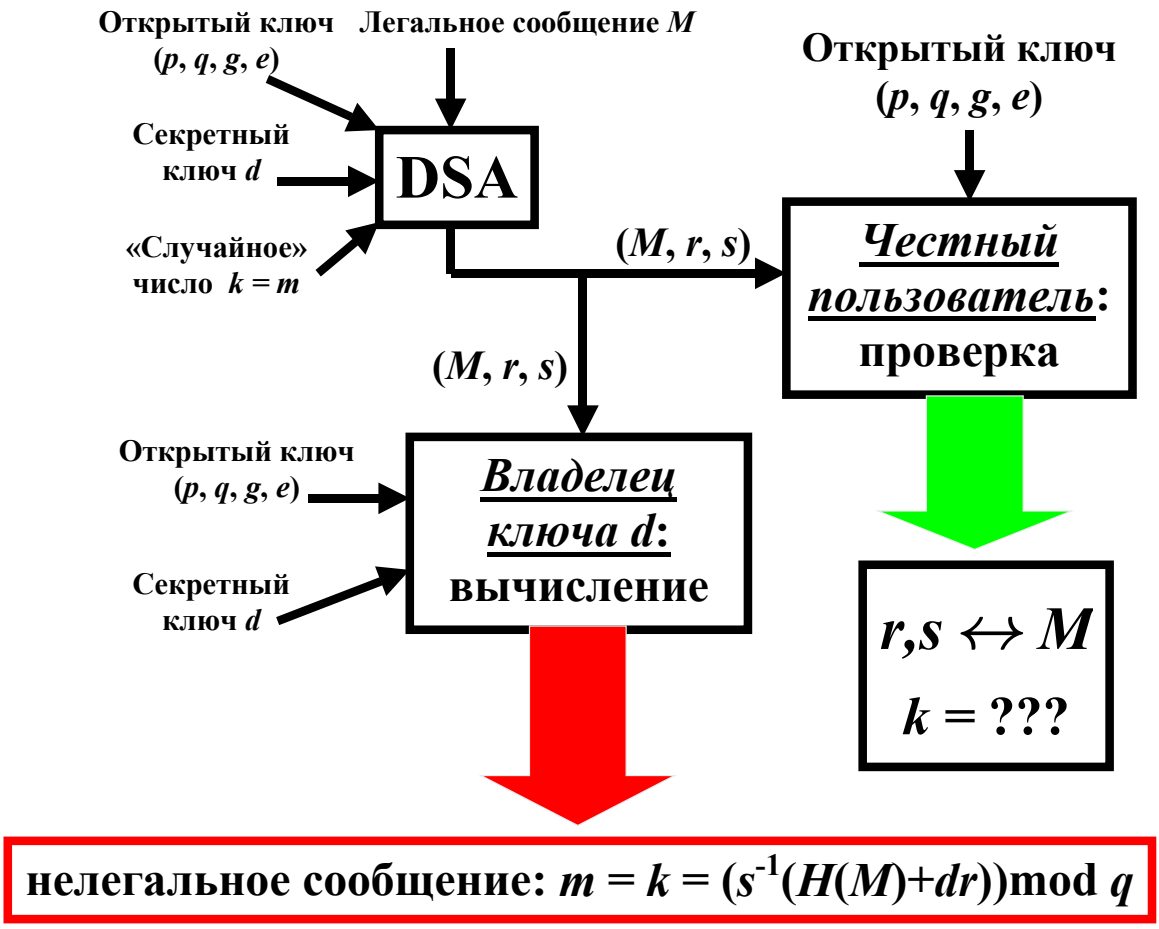


Рис. 5

Алгоритм подписи Ong-Schnorr-Shamir

Для этого выбирается большое n и случайное d, НОД(n,d) = 1 и вычисляется величина $e = -d^{-2} (\mathrm{mod}\, n)$. Открытый ключ: ($n$, e), секретный ключ: d.

Выработка подписи происходит по следующей схеме: M – сообщение, k – *случайное число* – секретный разовый ключ. Тогда подписанным сообщением является набор: (M, s_1, s_2), где $s_1 = \frac{1}{2}\left(\frac{M}{k} + k\right)(\mathrm{mod}\, n)$, $s_2 = \frac{d}{2}\left(\frac{M}{k} - k\right)(\mathrm{mod}\, n)$.

Проверка подписи: вычисляется величина $a = s_1^2 + e \cdot s_2^2 (\mathrm{mod}\, n)$. Подпись признается корректной, а сообщение аутентичным если выполняется равенство $M = a$ [1].

Для работы скрытого канала [5], [6], аналогично алгоритму DSA, в качестве случайного числа k берем само сообщение m, которое требуется передать по скрытому каналу: k=m. При известном секретном ключе d сообщение m может быть получено непосредственно из подписанного сообщения (M, s_1, s_2) по формуле:

$$m = k = \frac{M}{s_1 + s_2 \cdot d^{-1}} (\mathrm{mod}\, n).$$

Заметим, что и в этом случае указанный канал передачи информации является скрытым.

ВСТРОЕННЫЕ КАНАЛЫ УТЕЧКИ ИНФОРМАЦИИ (ЛАЗЕЙКИ)

Определим понятие ***клептография***, как теорию построения информационных систем, содержащих скрытые[*)] каналы утечки секретной информации.

Определим понятие шифра с лазейкой. ***Шифр с лазейкой*** (trapdoor) – это шифр, алгоритм которого содержит некоторую скрытую структуру (лазейку), обеспечивающую существование скрытого канала передачи информации; знание этой структуры позволяет получать секретную информацию (например, о секретном ключе). Без знания лазейки шифр кажется надежным.

При анализе работы клептографических систем следует выделить следующих трех участников:

- ***Разработчик***: обладает информацией о лазейке, владеет секретным ключом к лазейке, не владеет секретным ключом пользователя.
- ***Пользователь***: владеет секретным ключом пользователя, в случае успешного реверс-инженеринга обладает информацией о лазейке, но не владеет ее секретным ключом.
- ***Злоумышленник***: в случае успешного реверс-инженеринга обладает информацией о лазейке, но не владеет ее секретным ключом, а также секретным ключом пользователя.

Одним из наиболее важных типов лазеек, встраиваемых в криптографические алгоритмы является так называемый ***SETUP-механизм*** (SETUP – Secretly Embedded Trapdoor with Universal Protection – секретно встроенная лазейка с универсальной защитой) [11], [12]. SETUP-механизм видоизменяет заданный криптографический алгоритм таким образом, что позволяет производителю криптосистемы получать секретную информацию пользователя (чаще всего информацию о его секретных ключах).

[*)] В том смысле, что пользоваться этими каналами может только разработчик криптосистемы

В то же время для любого наблюдателя, отличного от разработчика, работа модифицированного алгоритма неотличима от работы исходного[*)].

Модифицированные таким образом криптосистемы иногда называют ***зараженными*** (или ***инфицированными***) криптосистемами.

Определение ([11], [12]). Пусть C – это криптосистема с объявленной спецификацией. Обычный SETUP-механизм – это такая модификация криптоалгоритма C в алгоритм C_1, что:

1. Параметры входа C_1 согласуются с объявленной спецификацией входных параметров C.
2. Параметры выхода C_1 соответствуют объявленной спецификации параметров выхода C. В тоже время, выход C_1 содержит секретные биты (например, биты секретного ключа пользователя), которые легко извлекаются разработчиком.
3. По выходам алгоритмы C_1 и C полиномиально неразличимы для всех, кроме разработчика.
4. Выход C_1 эффективно вычисляется с использованием встроенной в C_1 функции шифрования с открытым ключом E, а также, возможно, других функции, содержащихся в C_1.
5. Секретная функция расшифрования D, обратная к E, не содержится в C_1 и известна только разработчику.
6. После обнаружения в реализации алгоритма SETUP-механизма и выяснения его особенностей, например, при помощи реверс-инженеринга программы или аппаратного защищенного устройства, и пользователи и злоумышленники (все за исключением разработчика) не могут определить использованные (или будущие) секретные ключи пользователя. В этом смысле SETUP-механизм обеспечивает «криптостойкость» системы по отношению ко всем злоумышленникам, кроме ее разработчика.

В работах Adam Young и Moti Yung [11], [12], [13] показано, что целый ряд общепризнанных криптографических примитивов может быть модифицирован путем включения в тело соответствующей программы SETUP-механизма. При этом,

---

[*)] В том смысле, что без проведения реверс-инженеринга даже эксперты не могут дать ответ о наличии лазейки

модифицированные криптосистемы имеют спецификации, совпадающие со спецификациями исходных криптосистем.

ЛАЗЕЙКИ В КРИПТОСИСТЕМАХ С ОТКРЫТЫМ КЛЮЧОМ

Лазейки, которые мы рассмотрим ниже, используют криптографические системы с открытым ключом, встроенные в стандартные криптографические примитивы. Основное достижение такого подхода состоит в том, что он реализует SETUP-механизм без использования явных каналов утечки информации.

Как и раньше, примеры в некоторых деталях будут отличаться от приводимых в первоисточниках, но сохранят основную идею работы механизма лазеек [11] – [13] и др.

SETUP-механизм в протоколе Диффи-Хэллмана

Кратко напомним протокол Диффи-Хэллмана (DH) для выработки секретного ключа, использующий для обмена информацией открытый канал связи. Протокол DH использует для этого параметры p – большое простое число и g – примитивный элемент по модулю p. Эти параметры известны. Для выработки секретного ключа k пользователи ***A*** и ***B*** поступают следующим образом. ***A*** выбирает случайное число a такое, что $a < p-1$ – разовый ключ пользователя ***A***. ***B*** аналогично выбирает случайное число b – разовый ключ пользователя ***B***. ***A*** отправляет ***B*** сообщение вида $A = g^a \bmod p$, а ***B***, в свою очередь, отправляет ***A*** сообщение $B = g^b \bmod p$. После этого каждый из них может вычислить величину $k = A^b = B^a = g^{ab} (\bmod p)$, которая в дальнейшем может использоваться как секретный ключ в симметричной криптосистеме.

Рассмотрим следующую клептосистему, встроенную пользователю ***A***. Пусть $Y = g^{\delta}$, где δ – секретно и известно только разработчику. При выработке k_1 – первого ключа система использует случайное число a_1 и сообщение, отправляемое пользователем ***A***, имеет вид $A_1 = g^{a_1}$. Однако, уже при выработке k_2 – второго ключа в качестве параметра a_2 система выбирает величину $a_2 = Y^{a_1} + H(A_1) = g^{\delta a_1} + H(A_1)$, где H – некоторая «хорошая» хэш-функция. Соответственно сообщение, отправляемое

пользователем ***A***, имеет вид $A_2 = g^{a_2}$. Аналогично, при выработке k_3 – третьего ключа в качестве параметра a_3 выбирается величина $a_3 = Y^{a_2} + H(A_2) = g^{\delta a_2} + H(A_2)$, и т.д. Заметим, что величины $a_i = g^{a_{i-1}\delta} + H(A_{i-1}) = (A_{i-1})^{\delta} + H(A_{i-1})$ могут быть вычислены для всех $i > 1$ если известны $A_1, A_2, \ldots$ и δ. Таким образом, в случае многократного использования этого протокола разработчик, перехвативший информацию, которой обменивались пользователи ***A*** и ***B***, имеет возможность восстановить величины $a_2, a_3, \ldots$, а значит и все ключи k_i, начиная со второго.

Такой протокол имеет скрытый канал утечки информации. Протокол предоставляет исключительные права разработчику. Без знания величины δ, которая известна только разработчику, задача определения секретных разовых ключей a_n сведется к решению задачи дискретного логарифмирования, которая в настоящее время является очень сложной задачей!

Встраивание лазейки в алгоритм генерации ключей для схемы ElGamal

Напомним основные идеи на которых основана схема шифрования с открытым ключом Эль-Гамаля. Пусть p – большое простое число и g – примитивный элемент по модулю p. Секретный ключ пользователя d, где $d < p-1$. Открытый ключ пользователя – (e, g, p), где $e = g^d \bmod p$. Для того, чтобы зашифровать сообщение M $(M < p)$, выбирается секретное случайное число k – секретный разовый ключ, такое, что $k < p-1$. Затем вычисляются $r = g^k \bmod p$, и $s = e^k M \bmod p$. Шифртекстом является пара чисел (r, s). Для того, чтобы получить M необходимо вычислить $M = s / r^d \bmod p$.

Встроенный SETUP-механизм работает на этапе выработки ключей алгоритма ElGamal. Пусть $E(.)$ – некоторый алгоритм шифрования с открытым ключом, $D(.)$ – соответствующее обратное преобразование, известное только разработчику. Тогда в процессе генерации ключей алгоритма Эль-Гамаля первоначально выбираются p – большое простое число и d – секретный ключ пользователя. Затем вычисляется $g = E(d)$. Если g – примитивный элемент по модулю p, то $e = g^d (\bmod p)$, в противном случае выбирается новое d и процедура повторяется. Имея открытый ключ пользователя – (e, g, p), разработчик легко получает его секретный ключ: $d = D(g)$.

Встраивание лазейки в алгоритм генерации ключей RSA

SETUP-механизм может быть встроен и в алгоритм генерации ключей схемы RSA. Для выработки ключей схемы RSA необходимо выбрать p, q – большие простые числа и случайное число $1 < e < n = pq$, такое, что $\text{НОД}\left(e, \varphi(n)\right) = 1$. Тогда открытым ключом является пара (n, e), а секретным ключом – величина $d = e^{-1} (\text{mod}\, \varphi(n))$.

В работе SETUP-механизма, как и в предыдущем случае, будут использоваться некоторый алгоритм шифрования с открытым ключом $E(.)$ и соответствующее обратное преобразование $D(.)$, известное только разработчику. В процессе генерации ключей SETUP-механизмом первоначально выбираются простые числа p и q. Затем вычисляется $e = E(p)$. Если $\text{НОД}\left(e, \varphi(n)\right) = 1$, то e объявляется открытым ключом и вычисляется $d = e^{-1} (\text{mod}\, \varphi(n))$, в противном случае выбирается другое p. По открытому ключу пользователя – (n, e), разработчик получает секретный параметр $p = D(e)$.

ВСТРАИВАНИЕ ЛАЗЕЕК В СИММЕТРИЧНЫЕ АЛГОРИТМЫ ШИФРОВАНИЯ

Пример блочного шифра со встроенной лазейкой

Излагаемая ниже схема построения нелегального канала утечки информации в блочном шифре базируется на идеях работы [13]. Определим алгоритм блочного шифрования следующим образом. Размер информационного блока ([14]) равен 64 битам. В информационного блока открытого текста m через m_h обозначим старший бит, а через m_l – младшие 63 бита: $m = (m_h, m_l); m_h \in \mathbb{Z}_2, m_l \in \mathbb{Z}_2^{63}$. Аналогично c_h и c_l обозначают старший и 63 младших бита информационного блока шифртекста c, соответствующего открытому тексту m: $c = (c_h, c_l); c_h \in \mathbb{Z}_2, c_l \in \mathbb{Z}_2^{63}$. Пусть $E: \mathbb{Z}_2^{63} \times K \to \mathbb{Z}_2^{63}$ – некоторый алгоритм блочного шифрования (без лазейки) с 63-битным информационным блоком, а $D: \mathbb{Z}_2^{63} \times K \to \mathbb{Z}_2^{63}$ – соответствующий алгоритм расшифрования. Тогда наш алгоритм преобразует блок открытого текста $m = (m_h, m_l)$ в блок шифртекста $c = (c_h, c_l)$ по следующим правилам:

$$c_l = E(m_l, k) \in \mathbb{Z}_2^{63};\ a = T_i(k);\ c_h = m_h \oplus a \in \mathbb{Z}_2;$$

где $T_i(k)$ – i-й бит ключа k, а i – номер шифруемого информационного блока. Соответственно, алгоритм расшифрования имеет вид:

$$m_l = E^{-1}(c_l, k) \in \mathbb{Z}_2^{63};\ a = T_i(k);\ m_h = c_h \oplus a \in \mathbb{Z}_2.$$

В то же время знание i-го блока открытого текста позволяет получить i-й бит ключа:

$$T_i(k) = a = m_h \oplus c_h.$$

Знание t блоков открытого текста позволяет восстановить t битов ключа!

Рассмотренный выше алгоритм блочного шифрования имеет нелегальный канал утечки информации, который впрочем не является безопасным, так как им может воспользоваться любой знакомый со строением алгоритма.

Пример поточного шифра со встроенной лазейкой

Подавляющее большинство используемых на практике поточных шифров относится к разряду аддитивных синхронных поточных шифров, работающих по схеме:

шифрование:	*расшифрование:*
$y_i = x_i + k_i$	$x_i = y_i - k_i$

где x_i – знак открытого текста, y_i – соответствующий знак шифртекста, а k_i – соответствующий знак ключевого потока, который вырабатывается некоторым автономно работающим автоматом (running key generator – *RKG*), закон функционирования и начальное состояние которого определяется ключом системы k, имеющим существенно меньший объем относительно объема шифруемого сообщения (рис.6).

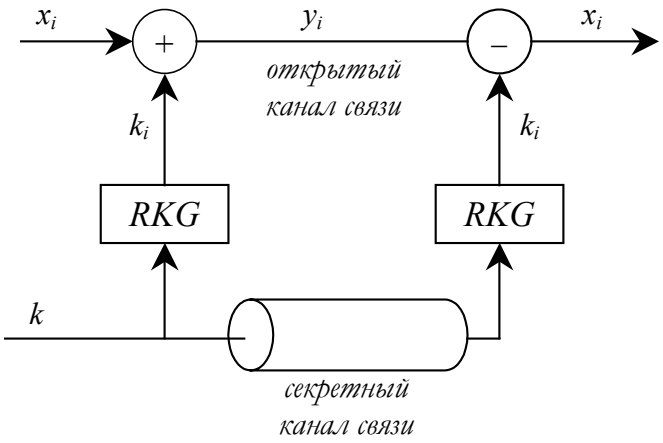


Рис. 6

В конструкции типичного RKG принято выделять управляющую часть (driving part) и комбинационную часть (combining part) (см. рис.7). Управляющая часть задает переход из одного внутреннего состояния автономного автомата в другое и обеспечивает большой период и хорошие статистические свойства выходной последовательности. Комбинационная часть непосредственно принимает участие в выработке знаков выходной последовательности и отвечает за ее сложность и непредсказуемость, не разрушая при этом хороших свойств, обеспечиваемых управляющей частью [15].

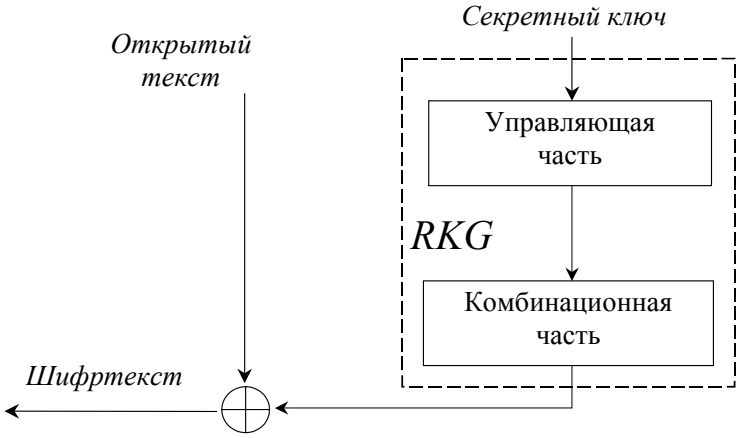


Рис. 7

Рассмотрим поточный шифр, элемент алфавита и секретный ключ которого являются n-битными словами. В качестве управляющей части будем использовать блочный алгоритм шифрования с n-битным информационным блоком и n-битным ключом, работающий по схеме, представленной на рис.8. Входом блочного алгоритма шифрования является секретный ключ k. Ключом блочного алгоритма шифрования является значение его выхода на предыдущей итерации. Начальный ключ блочного шифра – выбирается произвольно (например, нулевой вектор).

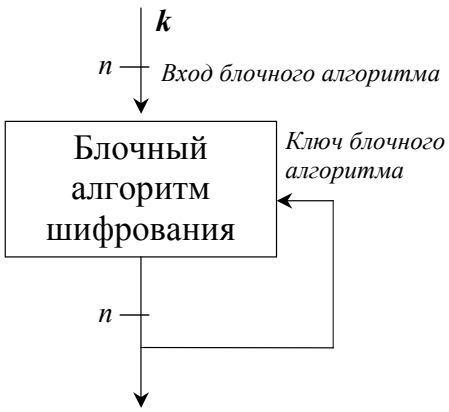


Рис. 8

В качестве комбинационной части будем использовать однонаправленную функцию. В частности, для этой цели можно выбрать конечный автомат или с бинарным или с *n*-битным алфавитом. Подобный выбор представляется вполне оправданным, особенно учитывая сложность задачи обращения конечного автомата*), который в этом контексте можно рассматривать как однонаправленную функцию.

Явная структура предлагаемой поточной криптосистемы представлена на рис.9. В данном случае, однонаправленная функция не позволяет при известных открытом и зашифрованном текстах вычислить значение выхода управляющей части и получить значение секретного ключа поточной криптосистемы.

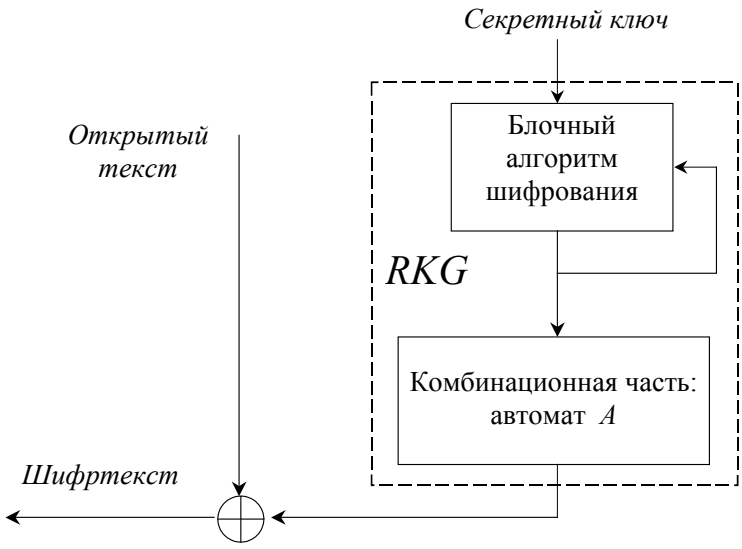


Рис. 9

Внутренняя структура криптосистемы

В качестве «однонаправленной» функции в данной криптосистеме на самом деле выбирается однонаправленная функция с лазейкой. Для ее обращения разработчик может использовать секретный ключ, известный только ему. Это обеспечивает защиту встроенной в поточную криптосистему лазейки, а следовательно, и эксклюзивного преимущества разработчика по взлому данной криптосистемы. Даже после успешного изучения структуры криптосистемы и обнаружения в ней SETUP-механизма, третья сторона не сможет использовать данную лазейку в собственных интересах, так как шифрование производится с использованием открытого ключа разработчика системы, а

*) т.е. построения автомата A^{-1}, который любую выходную последовательность автомата A перерабатывает в соответствующую входную последовательность.

для взлома криптосистемы необходимо знать секретный ключ разработчика. Таким образом, стойкость SETUP-механизма определяется стойкостью используемой функции шифрования с открытым ключом.

В качестве однонаправленной функцией с лазейкой предлагается использовать малоизвестный алгоритм шифрования с открытым ключом, основанный на конечных автоматах, называемый FAPKC (Finite Automaton Public Key Cryptosystem). В основании таких криптосистем лежат задача обратимости конечного автомата и задача декомпозиции конечного автомата. И та и другая задача в настоящее время считаются вычислительно сложными. Основная идея подобных криптосистем состоит в том, что шифруемая информация поступает на вход обратимого конечного автомата, который является композицией нескольких простых обратимых конечных автоматов (см. рис.10). Для расшифровки зашифрованной информации необходимо построить автомат, обратный шифрующему. Однако, если не знать его декомпозиции на простые, задача его обращения оказывается вычислительно сложной. Таким образом, открытым ключом криптосистемы является шифрующий автомат-композиция. Секретным ключом является обратный автомат, который легко строится, если известны простые автоматы, входящие в состав шифрующего автомата (рис.11).

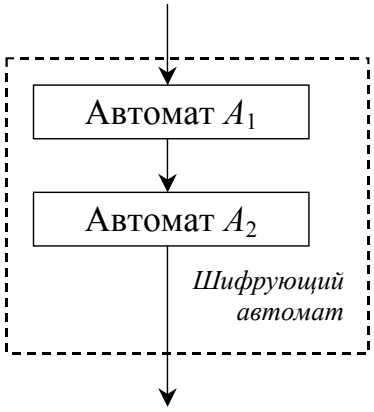


Рис. 10

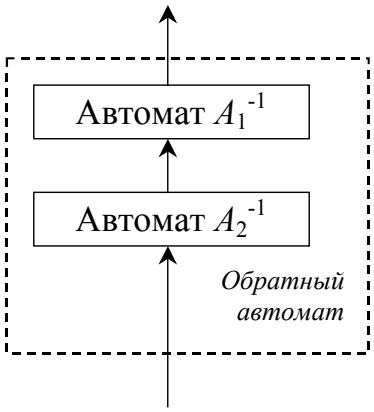


Рис. 11

Впервые такая криптосистема была представлена в 1985 г., некоторые ее модификации были опубликованы в 1986, а затем, в 1995 г. [2], [9], [10]. Являясь поточным шифром, FAPKC не требует разбиения открытого текста на блоки. Кроме того, он имеет высокую скорость (много выше, чем у RSA) и прост в реализации.

Для взлома построенной поточной криптосистемы рассмотрим сначала ее скрытую структуру (рис.10). Понятно, что разработчик знает соответствующие обратные автоматы для расшифрования A_1^{-1} и A_2^{-1}. Их можно легко получить из автоматов A_1 и A_2,

являющихся секретным ключом для нашей лазейки. Таким образом, если известны хотя бы два подряд стоящих *n*-битных блоков открытого текста, секретный ключ поточной криптосистемы находится с помощью алгоритма, приведенного на рис.12.

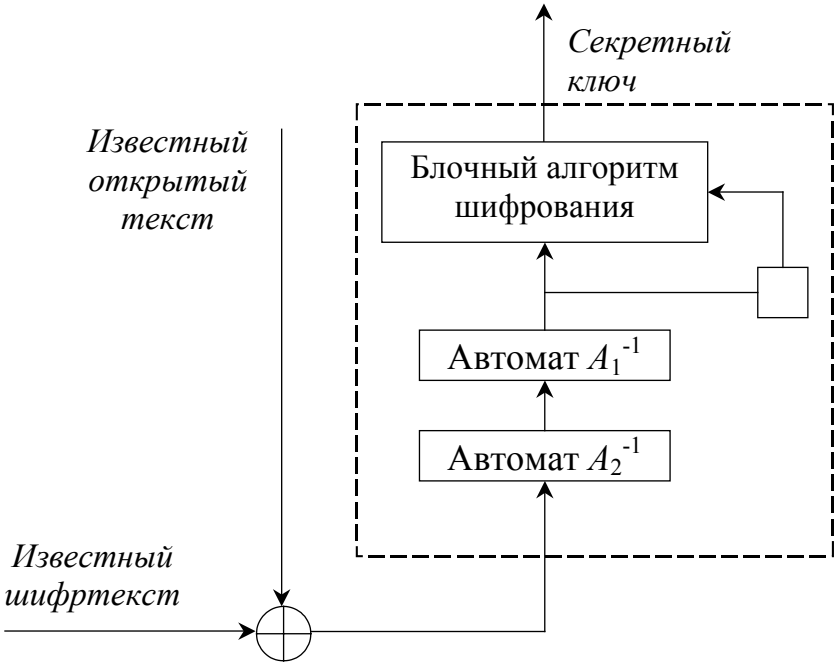


Рис. 12

Тестирование статистических свойств криптосистемы

RKG в поточных шифрах должен вести себя максимально подобно генератору истинно случайных последовательностей. Для оценки качества выходной ключевой последовательности, вырабатываемой нашим RKG, воспользуемся пакетом программ, рекомендуемым Национальным Институтом Стандартов и Технологий США (NIST) для оценки качества псевдослучайных последовательностей [16].

В результате многократного тестирования установлено, что выходная последовательность RKG всегда проходит тестирование, т.е. что по критериям NIST получающаяся выходная последовательность RKG нашего поточного шифра статистически не отличается от случайной последовательности и, следовательно, построенная поточная криптосистема со встроенной лазейкой с *формальной точки зрения* удовлетворяет этому критерию стойкости поточных криптосистем.

Таким образом, в данную поточную криптосистему была встроена лазейка с универсальной защитой (SETUP-механизм), основанная на однонаправленной функции шифровании с открытым ключом алгоритма FAPKC. Проведенное по указанным

критериям тестирование алгоритма показало, что разработанная криптосистема имеет формально хорошие криптографические характеристики и с этих позиций может считаться достаточно качественной. Однако, знание секретного ключа, необходимого для обращения однонаправленной функции шифрования, позволяет свести сложность нахождения секретного ключа поточной криптосистемы на нет.

Выводы

Встроенный SETUP-механизм полностью компрометирует рассматриваемые криптосистемы по отношению к ее разработчику (причем, только по отношению к нему). Эти атаки требуют единственного вмешательства в криптосистему. Особенную опасность представляют SETUP-механизмы для smart-карт, т.к. генерация ключей в них всегда происходит без участия пользователя. В случае с блочными шифрами можно также использовать каналы утечки, работающие при генерации и передаче так называемых Initial Vectors, которые используются в режимах OFB, CFB, CBC [14].

Многие разработчики криптографических протоколов считают, что хорошим правилом является встраивание внутрь самих протоколов механизмов случайности (т.е. датчиков случайных чисел). Но если устройство шифрования выбирает "случайные числа" самостоятельно, то, как было показано выше, разработчику криптосистемы не составляет большого труда организовать скрытый канал.

Общие рекомендации:

1. Перед использованием криптографического примитива, его структура должна быть тщательно изучена и оценена. Нельзя безоговорочно доверять аппаратным компонентам с заданной спецификацией (необходима проверка реализации на соответствие спецификации, а также изучение самой спецификации). Даже программные реализации могут быть опасны, особенно, если исходных код и документация разработчика недоступны для проверки.
2. Прохождение тестов по формальным критериям не гарантирует отсутствия скрытых лазеек в исследуемой криптосистеме. По-видимому, для любого фиксированного набора критериев можно построить криптосистему с лазейкой, которая, тем не менее, будет удовлетворять этим критериям. В частности, нельзя принимать решение о доверии лишь на основании обширного статистического исследования, так как оно не является альтернативой изучению структуры системы и ее реализации. Таким образом, для анализа криптосистемы

недостаточно формальной проверки, а необходимо проведение комплексного анализа ее структуры с привлечением специалистов в области криптографии.

3. Хорошую защиту от наличия скрытых лазеек в криптоалгоритмах дает композиция (каскадирование) криптопреобразований, имеющих происхождение из различных источников.
4. Важен контроль за случайностью, необходимо отличать ее от псевдослучайности. Так совершенно необходимо, чтобы алгоритмы выработки случайных величин, используемых в криптографических примитивах, были открыты для пользователя. Это позволит сравнить имеющуюся реализацию с заявленной. Если имеется программное обеспечение для выработки ключей – оно должно быть абсолютно надежным и доверенным. В случае smart-карт хорошим выходом будет возможность использовать посторонний источник случайных чисел.
5. Лучше если источник случайности, генератор ключей и алгоритм, использующий их – три раздельных компоненты. При этом исключена возможность их обхода, сами они – из надежного источника, а каналы, связывающие их, не допускают утечку информации.
6. Наиболее надежным является вариант использования собственной реализации какого-либо общепринятого алгоритма, так как не может быть никаких гарантий отсутствия недокументированных возможностей в сторонней реализации.

Литература

1. Шнайер Б. *Прикладная криптография*. М.: ТРИУМФ, 2002.
2. Dai Z.D., Ye D.F., Lam K.Y. *Weak Invertibility of Finite Automata and Cryptoanalysis on FAPKC.* ASIACRYPT’98, Lect. Notes Comput. Sci. v.1514 (1998), pp. 227-241.
3. *Digital Signature Standard (DSS)*. Federal Information Processing Standard (FIPS) Publication 186, National Institute of Standards and Technology (NIST), US Department of Commerce, Washington D.C., December, 1998.
4. Rijmen V., Preneel B. *A family of trapdoor ciphers*. FSE’97, Lect. Notes Comput. Sci. v.1267 (1997), pp. 139-148.
5. Simmons G.J. *The subliminal channel and digital signatures.* EUROCRYPT’84, Lect. Notes Comput. Sci. v.209 (1985), pp. 51-57.
6. Simmons G.J. *A secure subliminal channel (?).* CRYPTO’85, Lect. Notes Comput. Sci. v.218 (1986), pp. 33-41.
7. Simmons G.J. *Subliminal communication is easy using the DSA.* EUROCRYPT’93, Lect. Notes Comput. Sci. v.765 (1994), pp. 218-232.
8. Simmons G.J. *Subliminal Channels: Past and Present.* European Transactions on Telecommunication, v. 4, N 4 (1994), pp. 459-473.
9. Tao R.C. and Chen S.H., *A Finite Automaton Public Key Cryptosystem and Digital Signatures*. Chinese J. of Computer, 1985(8), pp.401-409.

10. Tao R.J. and Chen S.H. and Chen X.M. *FAPKC3: a new finite automaton public key cryptosystem.* Laboratory for Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100080, China, June, 1995. ISCAS–LCS–95–07.
11. Young A., Yung M. *The Dark Side of Black-Box Cryptography.* CRYPTO'96, Lect. Notes Comput. Sci. v., 1109 (1997), pp. 89-103.
12. Young A., Yung M. *Kleptography: using Cryptography against Cryptography*. EUROCRYPT'97, Lect. Notes Comput. Sci. v.1233, (1998), pp. 62-74.
13. Young A., Yung M. *Monkey – Black-Box Symmetric Ciphers Designed for MONopolizing KEYs*. FSE'98, Lect. Notes Comput. Sci. v.1372 (1998), pp. 122-133.
14. Варфоломеев А.А., Жуков А.Е., Мельников А.Б., Устюжанин Д.Д. *Блочные криптосистемы. Основные свойства и методы анализа стойкости.* М.: МИФИ, 1998
15. Варфоломеев А.А., Жуков А.Е., Пудовкина М.А. *Поточные криптосистемы. Основные свойства и методы анализа стойкости.* М.: ПАИМС, 2000
16. *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications.* NIST Special Publication 800-22, May 15, 2001