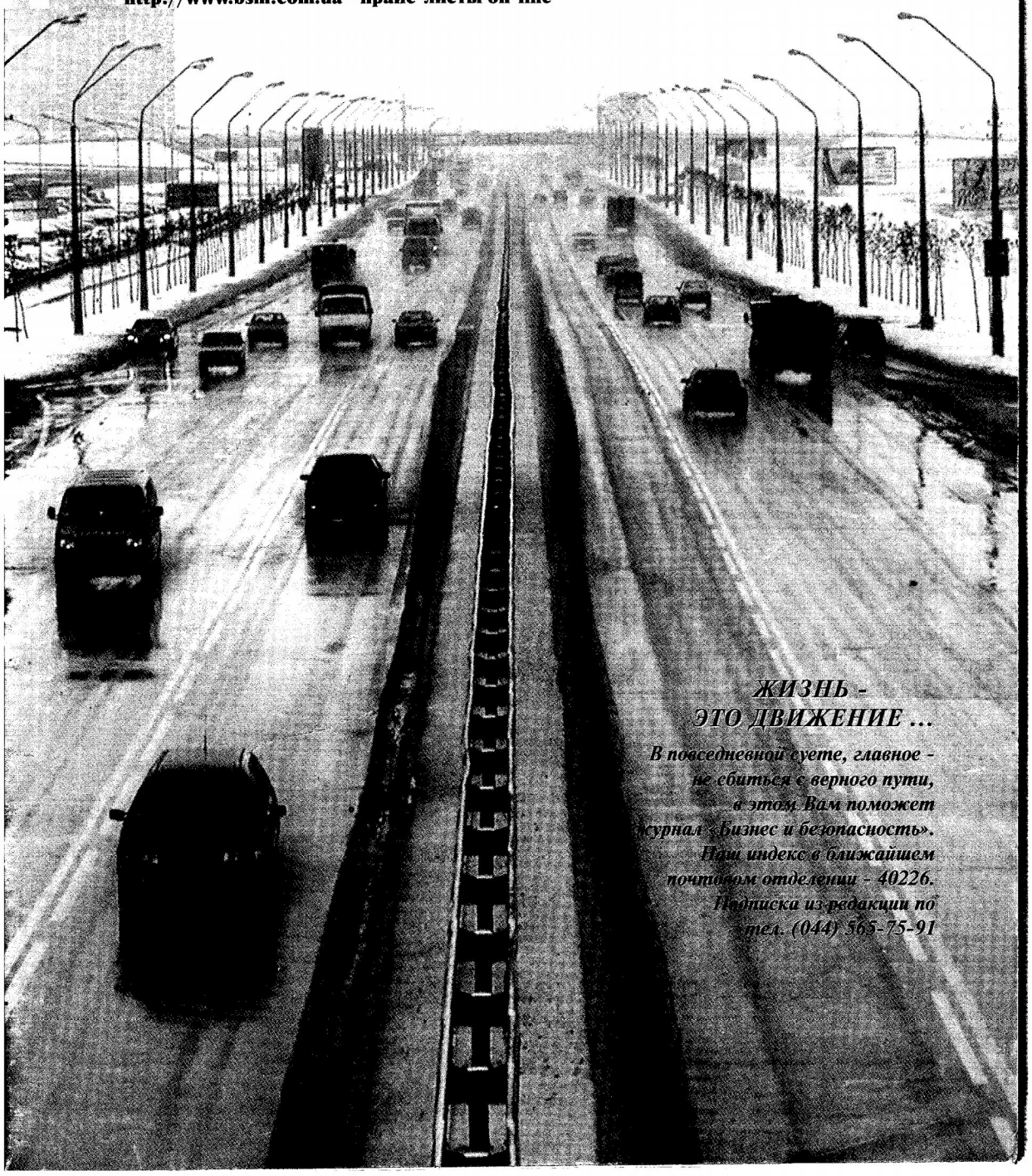
Бизнес и безопасность

6/2001

Бизнес и безопасность

http://www.bsm.com.ua - прайс-листы on-line



ПРИЧИНЫ НЕНАДЕЖНОСТИ ПРИКЛАДНЫХ КРИПТОГРАФИЧЕСКИХ ПРОГРАММ

В современном программном обеспечении криптоалгоритмы широко применяются не только для задач шифрования данных, но и для аутентификации и проверки целостности. На сегодняшний день существуют хорошо известные криптоалгоритмы (как с симметричными, так и с несимметричными ключами), криптостойкость которых либо доказана математически, либо основана на необходимости решения математически сложной задачи.

К наиболее известным из них относятся **DES** *(www.rsa.com/rsalabs/newfaq/q64.html)*, **ГОСТ** *(www.jetico.sci.fi/gost.htm)*, **RSA** *(www.rsa.com/rsalabs/newfaq/q8.html)*. Таким образом, они не могут быть вскрыты иначе, чем полным перебором или решением указанной задачи.

С другой стороны, в компьютерном и околокомпьютерном мире все время появляется информация об ошибках или «дырах» в той или иной программе (в т.ч. применяющей криптоалгоритмы), или о том, что она была взломана *(cracked)*. Это создает недоверие как к конкретным программам, так и к возможности вообще защитить что-либо криптографическими методами не только от спецслужб, но и от простых хакеров.

Поэтому знание истории атак и «дыр» в криптопрограммах, а также понимание причин, по которым они имели место, является одним из необходимых условий разработки защищенных систем.

Выделим следующие причины ненадежности криптографических программ:

1. Невозможность применения стойких криптоалгоритмов;
2. Ошибки в реализации криптоалгоритмов;
3. Неправильное применение криптоалгоритмов;
4. Человеческий фактор.

Невозможность применения стойких криптоалгоритмов

Эта группа причин является наиболее распространенной из-за следующих факторов.

Малая скорость стойких криптоалгоритмов

Это основной фактор, затрудняющий применение хороших алгоритмов, например, в системах «тотального» шифрования или шифрования «на лету». В частности, программа **Norton DiskReet**, хотя и имеет реализацию DES, при смене пользователем ключа может не перешифровывать весь диск, т.к. это займет слишком много времени. Аналогично, программа компрессии «на лету» **Stacker** фирмы *Stac Electronics (www.stac.com/)* имеет опцию закрытия паролем компрессируемых данных. Однако она не имеет физической возможности зашифровать этим паролем свой файл, обычно имеющий размеры в несколько сот мегабайт, поэтому она ограничивается очень слабым алгоритмом и хранит хэш-функцию от пароля вместе с защищаемыми данными. Величина криптостойкости этой функции оказалась равной 2^8.

Экспортные ограничения

Это причина, связанная с экспортом криптоалгоритмов или с необходимостью приобретать патент или права на них. В частности, из США запрещен экспорт криптоалгоритмов *(www.cypher.net/info/itar.html)* с длиной ключа более 56 бит. Очевидно, что такая криптостойкость не может считаться надежной при современных вычислительных мощностях.

Известные примеры программ, подверженных экспортным ограничениям - это последние версии *броузеров (browser)* Интернета, в частности **Netscape Navigator** фирмы *Netscape Communications (/home.netscape.com/)* и **Internet Explorer** фирмы *Microsoft (www.microsoft.com)*. Они предоставляют шифрование со 128-битным ключом для пользователей на территории США и с 40-битным ключом для всех остальных.

Также в эту группу попадает последняя версия архиватора ARJ 2.60 (www.arjsoftware.com/), известного своим слабым алгоритмом шифрования архивов. Теперь пользователи на территории США могут использовать криптостойкий алгоритм ГОСТ. Комизм ситуации в том, что, хотя этот алгоритм является российским, даже россияне по законам США все равно не могут воспользоваться им в программе ARJ.

Использование собственных криптоалгоритмов

Незнание или нежелание использовать известные алгоритмы, как ни парадоксально, также имеет место быть, особенно в программах типа Freeware и Shareware, например, в архиваторах.

Как уже говорилось, архиватор ARJ (до версии 2.60 включительно) использует (по умолчанию) очень слабый алгоритм шифрования. Казалось бы, что в данном случае использование его допустимо, т.к. архивированный текст должен быть совершенно неизбыточен и статистические методы криптоанализа здесь не подходят. Однако после более детального изучения оказалось, что в архивированном тексте присутствует (и это оказывается справедливым для любых архиваторов) некоторая неслучайная информация – например, таблица Хаффмана и другая служебная информация. Поэтому, точно зная или предсказав с некоторой вероятностью значение этих служебных переменных, можно с той же вероятностью определить и соответствующие символы пароля.

Далее, использование слабых алгоритмов часто приводит к успеху *атаки по открытому тексту*. В случае архиватора ARJ, если злоумышленнику известен хотя бы один файл из зашифрованного архива, он с легкостью определит *(www.ssl.stu.neva.ru/psw/DOWNLOAD/solvepwd.com, www.microsoft.com/rus)* пароль архива и извлечет оттуда все остальные файлы (криптостойкость ARJ при наличии открытого текста – 2^8).

Неправильная реализация криптоалгоритмов

Несмотря на то, что в этом случае применяются криптостойкие или сертифицированные алгоритмы, эта группа причин приводит к нарушениям безопасности криптосистем из-за их неправильной реализации.

Уменьшение криптостойкости при генерации ключа

Эта причина с весьма многочисленными примерами, когда криптопрограмма либо обрезает пароль пользователя, либо генерирует из него данные, имеющие меньшее количество бит, чем сам пароль. Примеры:

1. Во многих (старых) версиях UNIX пароль пользователя обрезается до 8 байт перед хэшированием. Любопытно, что, например, **Linux 2.0**, требуя от пользователей ввода паролей, содержащих обязательно буквы и цифры, не проверяет, чтобы 8-символьное начало пароля также состояло из букв и цифр. Поэтому пользователь, задав, например, достаточно надежный пароль *passwordIsgood19*, будет весьма удивлен, узнав, что хакер вошел в систему под его именем с помощью элементарного пароля *password*.

2. Novell Netware позволяет пользователям иметь пароли до 128 байт, что дает (считая латинские буквы без учета регистра, цифры и спецсимволы) $68^{[128}$~2^{779} комбинаций. Но при этом, во-первых, хэш-функция получает на входе всего лишь 32-байтовое значение, что ограничивает эффективную длину пароля этой же величиной.

3. Полностью аналогичная ситуация происходит с архиватором **RAR** *(www.creabel.com/softronic/rar_archiver.htm)* версий 1.5х – выбор пароля больше 10 символов не приводит к росту времени, необходимого на его вскрытие.

Если длина пароля «сверху» в этом случае определяется реализацией криптоалгоритмов, то ограничение на длину «снизу» уже связано с понятием единицы информации или энтропии.

Отсутствие проверки на слабые ключи

Некоторые криптоалгоритмы (в частности, DES, IDEA) при шифровании со специфическими ключами не могут обеспечить должный уровень криптостойкости. Такие ключи называют *слабыми (weak)*. Для **DES** известно 4 слабых и 12 *полуслабых (semi-weak)* ключей. И хотя вероятность попасть в них равняется ~$2 \cdot 10^{-16}$, для серьезных криптографических систем пренебрегать ей нельзя.

Мощность множества слабых ключей **IDEA** *(www.rsa.com/rsalabs/newfaq/q77.html)* составляет ни много ни мало – 251 (впрочем, из-за того, что всего ключей 2^{128}, вероятность попасть в него в $3 \cdot 10^7$ раз меньше, чем у DES).

Недостаточная защищенность от РПС

РПС (разрушающие программные средства) – это компьютерные вирусы, троянские кони, программные закладки и т. п. программы, способные перехватить секретный ключ или сами нешифрованные данные, а также просто подменить алгоритм на некриптостойкий. В случае если программист не предусмотрел достаточных способов защиты от РПС, они легко способны нарушить безопасность криптосистемы. Особенно это актуально для операционных систем, не имеющих встроенных средств защиты или средств разграничения доступа типа MS DOS или Windows 95:

1. Перехват пароля. Как пример можно привести самый старый способ похищения пароля, известный еще со времен больших ЭВМ, когда *программа - «фантом»* эмулирует приглашение ОС, предлагая ввести имя пользователя и пароль, запоминает его в некотором файле и прекращает работу с сообщением *«Invalid password»*. Для MS DOS и Windows существует множество закладок для чтения и сохранения паролей, набираемых на клавиатуре (через перехват соответствующего прерывания), например, при работе утилиты **Diskreet v. 6.0**.

2. Троянский конь в электронной почте. Последним примером *(www.avp.ru/russian/news/press/newvir.htm)* служат, имевшие место в июне 1998 года, попытки проникновения троянского коня через электронную почту. В письмо были вложены картинка и EXE-файл FREECD.EXE, который за то время, пока пользователь развлекался с письмом, расшифровывал пароли на соединение с провайдером (Dial-Up), и отправлял их на адрес ispp@usa.net.

Наличие зависимости во времени обработки ключей

Это сравнительно новый аспект недостаточно корректной реализации криптоалгоритмов. Многие криптосистемы неодинаково быстро обрабатывают разные входные данные. Это происходит как из-за аппаратных (разное количество тактов на операцию, попадание в процессорный кэш и т. п.), так и программных причин (особенно при оптимизации программы по времени). Время может зависеть как от ключа шифрования, так и от шифруемых данных.

Ошибки в программной реализации

Пока программы будут писаться людьми, этот фактор всегда будет иметь место. Хороший пример – ОС **Novell Netware 3.12**, где, несмотря на достаточно продуманную систему аутентификации, при которой, по заявлениям фирмы Novell, «нешифрованный пароль никогда не передается по сети», удалось найти ошибку в программе **SYSCON v. 3.76**, при которой пароль именно в открытом виде попадает в один из сетевых пакетов. Этого не наблюдается ни с более ранними, ни с более поздними версиями этой программы, что позволяет говорить именно о чисто программистской ошибке. Эта ошибка проявляется, только если супервизор меняет пароль кому-либо. Видимо, каким-то образом, в сетевой пакет попадает клавиатурный буфер.

Наличие люков

Причины наличия люков в криптопрограммах очевидны – разработчик хочет иметь контроль над обрабатываемой в его системе информацией и оставляет для себя возможность расшифровывать ее, не зная ключа пользователя. Возможно также, что они используются для отладки и по какой-то причине не убираются из конечного продукта. Естественно, что это рано или поздно становится известным достаточно большому кругу лиц и ценность такой криптопрограммы становится почти нулевой. Самыми известными примерами

Скорость полного перебора на Pentium 166

Криптосистема	Скорость, паролей/сек.
ARJ 2.50	350 000
RC5 - 56 бит	150 000
LM-хэш	50 000
Novell Netware 3.x	25 000
MS Office 97	15 000
UNIX - crypt()	15 000
RAR 2.0	1000
UNIX - MD5	500

здесь являются **AWARD BIOS** (до версии 4.51PG) с его универсальным паролем *«AWARD_SW»* и **СУБД Paradox** *(www.borland. com/paradox/)* фирмы *Borland International (www.borland.com)*, также имеющей «суперпароли» *«jIGGAe»* и *«nx6bprx»*.

Вплотную к наличию люков в реализации (очевидно, что в этом случае они используют явно нестойкие алгоритмы или хранят ключ вместе с данными) примыкают алгоритмы, дающие возможность третьему лицу читать зашифрованное сообщение, как это сделано в нашумевшем проекте **CLIPPER** *(epic.org/crypto/clipper/)*, где третьим лицом выступает государство.

Недостатки датчика случайных чисел (ДСЧ)

Хороший, математически проверенный и корректно реализованный ДСЧ также важен для криптосистемы, как и хороший, математически стойкий и корректный криптоалгоритм, иначе его недостатки могут повлиять на общую криптостойкость системы. При этом для моделирования ДСЧ на ЭВМ обычно применяют датчики псевдослучайных чисел (ПСЧ), характеризующиеся периодом, разбросом, а также необходимостью его *инициализации (seed)*. Применение ПСЧ для криптосистем вообще нельзя признать удачным решением, поэтому хорошие криптосистемы применяют для этих целей физический ДСЧ (специальную плату), или, по крайней мере, вырабатывают число для инициализации ПСЧ с помощью физических величин (например, время нажатия на клавишу).

Неправильное применение криптоалгоритмов

Эта группа причин приводит к тому, что оказываются ненадежными криптостойкие и корректно реализованные алгоритмы.

Малая длина ключа

Это самая очевидная причина. Возникает вопрос: как стойкие криптоалгоритмы могут иметь малую длину ключа? Видимо, вследствие двух факторов:

1. Некоторые алгоритмы могут работать с переменной длиной ключа, обеспечивая разную криптостойкость. Задача разработчика в том, чтобы выбрать необходимую длину, исходя из желаемой криптостойкости и эффективности. Иногда могут возникнуть иные обстоятельства – такие, как экспортные ограничения.

2. Некоторые алгоритмы разрабатывались весьма давно, когда длина используемого в них ключа считалась более чем достаточной для соблюдения нужного уровня защиты.

С резким скачком производительности вычислительной техники сначала столкнулся алгоритм **RSA**. В марте 1994 была закончена длившаяся в течение 8 месяцев факторизация числа из 129 цифр. Для этого было задействовано 600 добровольцев и 1600 машин, связанных посредством электронной почты.

Прогресс в решении проблемы факторизации во многом связан не только с ростом вычислительных мощностей, но и с появлением в последнее время новых эффективных алгоритмов. На сегодняшний день, в принципе, реально факторизовать 512-битные числа. Если вспомнить, что такие числа еще недавно использовались в программе **PGP** *(www.pgp.com/)*, то можно утверждать, что это самая быстро развивающаяся область криптографии и теории чисел.

29 января 1997 фирмой *RSA Labs (www.rsa.com/rsalabs/)* был объявлен конкурс *(www.rsa.com/rsalabs/97challenge/)* на вскрытие симметричного алгоритма RC5 *(www.rsa.com/rsalabs/newfaq/q76.html)*. 40-

битный ключ был вскрыт *(www.rsa.com/rsalabs/new-faq/q76.html)* через 3.5 часа после начала конкурса! (Для этого даже не потребовалось связывать компьютеры через Интернет – хватило локальной сети из 250 машин в Берклевском университете). Через 313 часов был вскрыт и 48-битный ключ *(www.42.org/challenge/)*. Таким образом, всем стало очевидно, что длина ключа, удовлетворяющая экспортным ограничениям, не может обеспечить даже минимальной надежности.

Параллельно со вскрытием RC5 был брошен вызов и столпу американской криптографии – алгоритму **DES**, имеющему ключ в 56 бит. И он пал *(www.rsa.com/des/)* 17 июня 1997 года, через 140 дней после начала конкурса (при этом было протестировано около 25 % всех возможных ключей и затрачено примерно 450 MIPS-лет). Это было, безусловно, выдающееся достижение, которое означало фактическую смерть DES как стандарта шифрования.

Ошибочный выбор класса алгоритма

Это также весьма распространенная причина, при которой разработчик выбирает пусть и хороший, но совершенно неподходящий к его задаче алгоритм. Чаще всего это выбор шифрования вместо хэширования или выбор симметричного алгоритма вместо алгоритма с открытыми ключами.

Примеров здесь множество – это почти все программы, ограничивающие доступ к компьютеру паролем при его включении или загрузке, например, **AMI BIOS**, хранящий вместо хэша пароля его зашифрованный вариант, который, естественно, легко дешифруется *(www.ssl.stu.neva.ru/psw/DOWNLOAD/amipswd.com)*.

Во всех сетевых процедурах аутентификации естественно применять асимметричную криптографию, которая не позволит подобрать ключ даже при полном перехвате трафика. Однако такие алгоритмы (из сетевых ОС) пока реализуют только **Novell Netware 4.x**, остальные же довольствуются (в лучшем случае!) стандартной схемой «запрос-отклик», при которой можно вести достаточно быстрый перебор по перехваченным значениям «запроса» и «отклика».

Человеческий фактор

В любой критической системе ошибки человека-оператора являются чуть ли не самыми дорогостоящими и распространенными. В случае с криптопрограммами, непрофессиональные действия пользователя сводят на нет самый стойкий криптоалгоритм и самую корректную его реализацию и применение.

В первую очередь, это связано с выбором паролей. Очевидно, что короткие или осмысленные пароли легко запоминаются человеком, но они гораздо проще для вскрытия. Использование длинных и бессмысленных паролей, безусловно, лучше с точки зрения криптостойкости, но человек обычно не может их запомнить и записывает на бумажке, которая потом либо теряется, либо попадает в руки злоумышленника.

В последние годы много внимания уделяется разрешению этого противоречия, но рекомендации по выбору хороших паролей выходят за рамки этой статьи.

Именно из того, что неискушенные пользователи обычно выбирают либо короткие, либо осмысленные пароли, существуют два метода их вскрытия: *атака полным перебором* и *атака по словарю*.

В связи с резким ростом вычислительных мощностей, атаки полным перебором имеют гораздо больше шансов на успех, чем раньше. Скорость перебора паролей для различных криптосистем приведена в таблице.

Однако вернемся на несколько лет назад, когда вычислительной мощности для полного перебора всех паролей не хватало. Хакерами был тогда придуман остроумный метод, основанный на том, что в качестве пароля человеком выбирается существующее слово или какая-либо информация о себе или своих знакомых (имя, дата рождения и т. п.). Ну, а поскольку в любом языке не более 100000 слов, то их перебор займет весьма небольшой промежуток времени. От 40 до 80 % существующих паролей может быть угадано с помощью простой схемы, называемой «атака по словарю». (Кстати, до 80 % этих паролей может быть угадано с использованием словаря размером всего 1000 слов!) Даже вирус Морриса (в 1988 г.!) применял такой способ, тем более что в UNIX «под рукой» часто оказывается файл-словарь, обычно используемый программами--корректорами. Что же касается «собственных» паролей, то файл */etc/passwd* может дать немало информации о пользователе: его входное имя, имя и фамилию, домашний каталог. Вирус Морриса с успехом пользовался следующими предположениями:

- в качестве пароля берется входное имя пользователя;
- пароль представляет собой двойной повтор имени пользователя;
- то же, но прочитанное справа налево;
- имя или фамилия пользователя;
- то же, но в нижнем регистре.

Сегодня пользователи уже понимают, что выбирать такие пароли нельзя, но до тех пор, пока с компьютером работает человек, эксперты по компьютерной безопасности не дождутся использования таких простых и радующих душу паролей, как *34jXs5U@bTa!6*. Поэтому даже искушенный пользователь хитрит и выбирает такие пароли, как *hope1, user1997, pAsSwOrD, toor, roottoor, parol, gfhjkm, asxz*. Видно, что все они, как правило, базируются на осмысленном слове или на простом его преобразовании (прибавить цифру, прибавить год, перевести через букву в другой регистр, записать слово наоборот, прибавить записанное наоборот слово, записать русское слово латинскими буквами, набрать русское слово на клавиатуре с латинской раскладкой, составить пароль из рядом расположенных на клавиатуре клавиш и т. п.).

Поэтому не надо удивляться, если такой «хитрый» пароль будет вскрыт хакерами. Они не глупее самих пользователей, и уже вставили в свои программы те правила, по которым может идти преобразование слов.

Заключение

Можно выделить 4 основных группы причин ненадежности криптографических программ:

- применение нестойких алгоритмов;
- неправильная реализация криптоалгоритмов;
- неправильное применение криптоалгоритмов;
- человеческий фактор.

При этом видна четкая параллель между ними и причинами нарушения безопасности вычислительных систем.

По вышеописанным причинам имелись или имеются проблемы в безопасности у всех классов программных продуктов, использующих криптоалгоритмы, будь то операционные системы, криптопротоколы, клиенты и сервера, их поддерживающие, офисные программы, пользовательские утилиты шифрования, популярные архиваторы.

Для того, чтобы грамотно реализовать свою криптопрограмму, необходимо не только ознакомиться с ошибками других и понять причины, по которым они произошли, но и, возможно, применять особые защитные приемы программирования и специализированные средства разработки.

Медведев Н.Г.,
профессор, д.т.н.
Москалик Д.В.,
инженер- программист,
Европейский университет

Литература.

1. Теория и практика обеспечения информационной безопасности. Под редакцией Зегжды П.Д. – М., Яхтсмен, 1996. (www.ssl.stu.neva.ru/ssl/koi/publications/book-6.html)

2. П. Кочер. Временной анализ реализаций Диффи-Хеллмана, RSA, DSS и других систем (www.ssl.stu.neva.ru/psw/crypto/timing.html)

3. Mark W. Eichin, Jon A. Rochils. With Microscope and Tweezers: An Analysis of the Internet virus of November 1988. (coast.cs.purdue.edu/pub/doc/morris_worm/mit.PS.Z)

4. Б. Шнайер, П. Мюдж. Криптоанализ протокола PPTP от Microsoft (www.ssl.stu.neva.ru/psw/crypto/pptp.html).

5. Eli Biham, Lars R. Knudsen. Cryptanalysis of the ANSI X9.52 CBCM Mode. Proceedings of Eurocrypt'98. (www.cs.technion.ac.il/users/wwwb/cgi-bin/tr-get.cgi/1998/CS/CS0928.ps.gz)