Любая тайна, порожденная человеческим сознанием, им же может быть и раскрыта.

/Шерлок Холмс/

...едва ли разуму человека дано загадать такую загадку, которую разум другого его собрата, направленный должным образом, не смог бы раскрыть.

/Эдгар По/

Криптоанализ: современное состояние и перспективы развития

С.М. Авдошин, канд. техн. наук, проф.,
А.А. Савельева,
Государственный университет –
Высшая Школа Экономики

Одним из главных аспектов аудита безопасности информационных систем является оценка надежности используемых криптографических алгоритмов. В статье приведен обзор современных методов криптоанализа. Наряду с классическими методами, такими как метод полного перебора и метод Полларда, описываются атаки на симметричные и асимметричные криптосистемы: линейный и разностный анализ блочных шифров, субэкспоненциальные алгоритмы факторизации и дискретного логарифмирования и т.д. Особое внимание уделяется новому виду криптоанализа – атакам по побочным каналам. Последний раздел посвящен «экзотическим» методам криптоанализа, основанным на использовании нейронных сетей, генетических алгоритмов и квантовых компьютеров: несмотря на то, что в настоящее время эти методы не привели к сколько-нибудь серьезным прорывам во взломе шифров, нельзя исключать, что со временем их значение в криптологии может возрасти.

Cryptanalysis: Current State and Future Trends

Avdoshin S.M., Savelieva A.A.

An important aspect of information systems security auditing involves cryptographic algorithms strength evaluation. In this paper, we provide a survey of modern cryptanalysis techniques. Apart from classical approaches (such as brute-force attack and Pollard method), some techniques for breaking private-key and public-key cryptosystems are described as well (linear and differential block-cipher cryptanalysis, subexponential algorithms for the discrete logarithm and factoring, etc.). High emphasis is placed on a state-of the-art approach, side-channel attacks. The final part discusses some “eccentric” cryptanalysis techniques involving neural networks, genetic algorithms and quantum computers: no spectacular results have been obtained using these approaches; however, we cannot neglect that their significance in cryptology is yet to come.

Введение

Великий сыщик и великий писатель ошибались: их утверждения, вынесенные в эпиграф, были в середине 40х гг. прошлого века опровергнуты великим ученым и основоположником современной криптографии Клодом Шенноном. Он показал ([102]), что если на любой исходный текст наложить (т.е. сложить по модулю с текстом) ключ длины не меньшей, чем само сообщение, то такой шифр будет нераскрываемым: потенциальному злоумышленнику потребуется перебрать все возможные ключи и каждым из них попробовать расшифровать сообщение. Однако использование такого

способа шифрования, получившего название «одноразовых блокнотов», в большинстве случаев оказывается слишком дорогим и неоправданным. Это связано с тем, что нет смысла бороться за устойчивость системы защиты информации к взлому ниже некоторой «фоновой» вероятности, т.е. вероятности события, которое мы не в состоянии предотвратить [77]. Например, если вероятность выхода компании из бизнеса равна 2^{-30} (менее чем один из миллиона), то есть ли смысл для защиты информации, которая может нанести компании ущерб, сопоставимый с кризисом рынка, использовать алгоритм, вероятность вскрытия которого за приемлемое время составляет 2^{-200}?

Выбор необходимой степени защиты информации и средств ее обеспечения является важной задачей и должен учитывать ряд параметров: уровень секретности информации; ее стоимость; время, в течение которого она должна оставаться в тайне и т.д. Проблема защиты информационных ресурсов в настоящее время приобретает все более важное значение. Так, по данным отчета CSI/FBI Computer Crime and Security Survey 2005 [27], средний ущерб каждой компании, в которой в минувшем году была зафиксирована утечка конфиденциальных данных, составил 355,5 тыс. долларов (причем по сравнению с 2004 годом эта цифра возросла почти вдвое). По некоторым оценкам, экономические потери от злонамеренных атак на банковские системы по всему миру составляют ежегодно около 130 млрд. долларов.

Как известно [60], далеко не все присутствующие на рынке криптографические средства обеспечивают обещанный уровень защиты. Важность этой проблемы подчеркивается и в работе [99]. Системы и средства защиты информации (СЗИ) характеризуются тем, что для них не существует простых и однозначных тестов, позволяющих убедиться в надежной защите информации. Например, для проверки работоспособности системы связи достаточно провести ее испытания. Однако успешное завершение этих испытаний не позволяет сделать вывод о том, что встроенная в нее подсистема защиты информации тоже работоспособна. Задача определения эффективности СЗИ при использовании криптографических методов защиты зачастую более трудоемкая, чем разработка СЗИ, требует наличия специальных знаний и более высокой квалификации, чем задача разработки. Часто анализ нового шифра является новой научной, а не инженерной задачей.

Эти обстоятельства приводят к тому, что на рынке появляются средства криптографической защиты информации, про которые никто не может сказать ничего определенного. При этом нередко разработчики держат криптоалгоритм (как показывает практика, часто легко взламываемый) в секрете. Однако задача точного определения используемого криптоалгоритма не может быть гарантированно сложной хотя бы потому, что он известен разработчикам. Кроме того, если нарушитель нашел способ преодоления защиты, то не в его интересах об этом заявлять. В результате пользователи таких СЗИ попадают в зависимость как минимум от разработчика. Поэтому обществу должно быть выгодно открытое обсуждение безопасности СЗИ массового применения, а сокрытие разработчиками криптоалгоритма должно быть недопустимым [99].

Современная криптография — соревнование методов шифрования и криптоанализа. *Криптоанализом* (от греческого *kryptós* - "скрытый" и *analýein* - "ослаблять" или избавлять") называют науку восстановления (дешифрования) открытого текста без доступа к ключу. Фундаментальное допущение криптоанализа, впервые сформулированное Кирхгоффом [34], состоит в том, что секретность сообщения всецело зависит от ключа, т.е. весь механизм шифрования, кроме значения ключа, известен противнику. Как бы то ни было, секретность алгоритма не является большим препятствием: например, для определения типа программно реализованного криптографического алгоритма требуется лишь несколько дней инженерного анализа исполняемого кода.

Криптоанализ ставит своей задачей в разных условиях получить дополнительные сведения о ключе шифрования, чтобы значительно уменьшить диапазон вероятных

ключей. Результаты криптоанализа могут варьироваться по степени практической применимости. Так, криптограф Ларс Кнудсен [36] предлагает следующую классификацию успешных исходов криптоанализа блочных шифров в зависимости от объема и качества секретной информации, которую удалось получить:

- Полный взлом – криптоаналитик извлекает секретный ключ.
- Глобальная дедукция – криптоаналитик разрабатывает функциональный эквивалент исследуемого алгоритма, позволяющий зашифровывать и расшифровывать информацию без знания ключа.
- Частичная дедукция – криптоаналитику удается расшифровать или зашифровать некоторые сообщения.
- Информационная дедукция – криптоаналитик получает некоторую информацию об открытом тексте или ключе.

Однако взлом шифра совсем не обязательно подразумевает обнаружение способа, применимого на практике для восстановления открытого текста по перехваченному зашифрованному сообщению. В научной криптологии другие правила [57]. Шифр считается взломанным, если в системе обнаружено слабое место, которое может быть использовано для более эффективного взлома, чем метод полного перебора ключей (‘brute-force approach’). Допустим, для дешифрования текста методом полного перебора требуется перебрать 2^{128} возможных ключей; тогда изобретение способа, требующего для дешифрования 2^{110} операций по подбору ключа, будет считаться взломом. Такие способы могут требовать нереалистично больших объемов подобранного открытого текста или памяти ЭВМ. Под взломом понимается лишь подтверждение наличия уязвимости криптоалгоритма, свидетельствующее о том, что свойства надежности шифра не соответствуют заявленным характеристикам. Как правило, криптоанализ начинается с попыток взлома упрощенной модификации алгоритма, после чего результаты распространяются на полноценную версию: прежде чем браться за взлом, например, 16-раундовой версии DES, естественно для начала попытаться взломать шифр с меньшим количеством раундов, чем указано в его спецификации (например, 8-раундовую версию шифра).

Попытка криптоанализа называется *атакой*. Прежде чем классифицировать атаки, введем ряд обозначений: открытый текст будем обозначать буквой x, шифртекст - буквой y (в качестве x может выступать любая последовательность битов: текстовый файл, оцифрованный звук, точечный рисунок и т.д.). Пусть для зашифрования и расшифрования используются ключи k и k' соответственно (в симметричной криптографии $k = k'$); обозначим функцию зашифрования E_k, расшифрования - $D_{k'}$. Тогда выполняются соотношения $E_k(x) = y,\ D_{k'}(y) = x$.

Известны четыре основных типа криптоаналитических атак. В каждом случае предполагается (согласно фундаментальному допущению Кирхгоффа), что криптоаналитик знает используемый алгоритм шифрования.

- **Атака на основе только шифртекста.** Криптоаналитик располагает шифртекстами $y_1, \ldots, y_m$, полученными из неизвестных открытых текстов $x_1, \ldots, x_m$ различных сообщений. Требуется найти хотя бы один из $x_i, i = \overline{1,m}$ (или соответствующий ключ k_i), исходя из достаточного числа m криптограмм, или убедиться в своей неспособности сделать это. В качестве частных случаев возможно совпадение ключей: $k_1 = \ldots = k_m$ или совпадение открытых текстов: $x_1 = \ldots = x_m$.
- **Атака на основе открытого текста.** Криптоаналитик располагает парами $(x_1, y_1), \ldots, (x_m, y_m)$ открытых и соответствующим им зашифрованных текстов.

Требуется определить ключ k_i для хотя бы одной из пар. В частном случае, когда $k_1 = ... = k_m = k$, требуется определить ключ k или, убедившись в своей неспособности сделать это, определить открытый текст x_{m+1} еще одной криптограммы y_{m+1}, зашифрованной на том же ключе.

- **Атака на основе подобранного открытого текста** отличается от предыдущей лишь тем, что криптоаналитик имеет возможность выбора открытых текстов $x_1, ..., x_m$. Цель атаки та же, что и предыдущей. Подобная атака возможна, например, в случае, когда криптоаналитик имеет доступ к шифратору передающей стороны.
- **Атака на основе адаптивно подобранного открытого текста.** Это частный случай вышеописанной атаки с использованием подобранного открытого текста. Криптоаналитик может не только выбирать используемых шифруемый текст, но также уточнять свой последующий выбор на основе полученных ранее результатов шифрования.

Атаки с использованием известного или подобранного открытого текста встречаются чаще, чем можно подумать. Необходимым требованием к хорошему криптоалгоритму является способность противостоять таким атакам. Это означает, что рассекречивание некоторой информации, передававшейся по каналу связи в зашифрованном виде, не должно приводить к рассекречиванию другой информации, зашифрованной на этом ключе. Кроме того, указанное требование учитывает особенности эксплуатации аппаратуры и допускает некоторые вольности со стороны оператора или лиц, имеющих доступ к формированию засекреченной информации. В среде криптоаналитиков нельзя назвать неслыханными факты добычи открытого текста шифрованного сообщения или подкупа лица, которое должно будет зашифровать избранное сообщение. Применяются и «косвенные» методы осуществления атаки на основе подобранного шифртекста. Злоумышленник может убедить обладателя секретного ключа переслать некое сообщение, но в зашифрованной форме. Так, в [31] описан прием, использованный командованием военно-морского флота США во время Второй Мировой Войны перед битвой на Мидвее. Чтобы убедиться в правильности результатов работы по взлому японского военного шифра, криптоаналитики США попросили американский гарнизон, дислоцированный на Мидвее, сообщить по открытому незащищенному каналу о нехватке пресной воды. Спустя два дня было перехвачено секретное сообщение, в котором японцы, осуществлявшие мониторинг использованного канала, сообщали о проблемах с водой в некоем “AF”. Благодаря этому американцы узнали, “AF” – кодовое обозначение Мидвея в шифрограммах противника. Атаки на основе подобранных текстов считаются наиболее опасными.

Указанные криптоатаки относятся к классу *пассивных* [87]. Так классифицируются действия противника, который «пассивно изучает» шифрованные сообщения, может их перехватить и подвергнуть криптоанализу с целью получения информации об открытом тексте или ключе. Однако современные технические средства позволяют потенциальному противнику «активно» вмешиваться в процесс передачи сообщения. Обычно различают два типа *активных* атак, которые носят названия имитации и подмены сообщения. Атака имитации состоит в том, что противник «вставляет» в канал связи сфабрикованное им «шифрованное сообщение», которое на самом деле не передавалось от законного отправителя к получателю. При этом противник рассчитывает на то, что получатель воспримет это сообщение как подлинное (аутентичное). Атака подмены состоит в том, что противник, наблюдая передаваемое по каналу связи подлинное сообщение от отправителя, «изымает» его и заменяет поддельным. Различные шифры могут быть более или менее уязвимыми к активным атакам. Способность самого шифра (без использования дополнительных средств) противостоять активным атакам обычно называют

имитостойкостью шифра. Количественной мерой имитостойкости шифра служат вероятности успеха имитации и подмены соответственно. Эти вероятности определяют шансы противника на успех при навязывании получателю ложного сообщения.

Атаки можно также классифицировать по объему ресурсов, необходимых для их осуществления:

- Память – объем памяти, требуемый для реализации атаки;
- Время – количество элементарных операций, которые необходимо выполнить;
- Данные – необходимый объём открытых и соответствующим им зашифрованных текстов.

В некоторых случаях эти параметры являются взаимозависимыми: например, за счет увеличения памяти можно сократить время атаки.

Способность криптосистемы противостоять атакам криптоаналитика называется *стойкостью*. Количественно стойкость измеряется как сложность наилучшего алгоритма, приводящего криптоаналитика к успеху с приемлемой вероятностью. Универсальный метод прямого перебора множества всех возможных ключей позволяет получить оценку сверху для стойкости алгоритма шифрования. Проблема всей современной криптографии – это отсутствие нижней границы стойкости; длина ключа задаёт лишь общий объём пространства ключей, но всегда есть вероятность, ткнув пальцем в небо, угадать решение. Относительное ожидаемое безопасное время определяется как полупроизведение числа открытых ключей и времени, необходимого криптоаналитику для того, чтобы испытывать каждый ключ [95]. В зависимости от целей и возможностей криптоаналитика меняется и стойкость. Различают стойкость ключа (сложность раскрытия ключа наилучшим известным алгоритмом), стойкость бесключевого чтения, имитостойкость (сложность навязывания ложной информации наилучшим известным алгоритмом) и вероятность навязывания ложной информации. Аналогично можно различать стойкость собственно криптоалгоритма, стойкость протокола, стойкость алгоритма генерации и распространения ключей [99].

В зависимости от сложности взлома алгоритмы обеспечивают различные степени защиты. Во главу угла ставится принципиальная возможность получения по перехвату некоторой информации об открытом тексте или использованном ключе. Существуют *безусловно стойкие* (или *теоретически стойкие*), *доказуемо стойкие* и *предположительно стойкие* криптоалгоритмы.

Теоретически стойкие системы создают шифртексты, содержащие недостаточное количество информации для однозначного определения соответствующих им текстов (или ключей). В лучшем случае открытый текст может быть локализован в достаточно большом подмножестве множества всех открытых текстов, и его можно лишь «угадать» с ничтожно малой вероятностью. Для совершенного шифра открытый текст «локализуется» во всем множестве открытых текстов. Тем самым, для него сама задача расшифрования становится бессмысленной. Никакой метод криптоанализа, включая полный перебор ключей, не позволяет не только определить ключ или открытый текст, но даже получить некоторую информацию о них. Алгоритм безусловно стоек, если восстановление открытого текста невозможно при любом объеме шифртекста, полученного криптоаналитиком. Безопасность безусловно стойких криптоалгоритмов основана на доказанных теоремах о невозможности раскрытия ключа. Как уже говорилось, принципиально не раскрываемые шифры (например, совершенно секретные системы Клода Шеннона, в которых ключ не может использоваться повторно, а его размер больше либо равен объему текста) неудобны на практике (симметричные криптосистемы с разовым использованием ключа требуют большой защищенной памяти для хранения ключей, системы квантовой криптографии требуют волоконно-оптических каналов связи и являются дорогими, кроме того, доказательство их безопасности уходит из области математики в область физики [99]). В силу своей непрактичности и высокой ресурсозатратности абсолютно стойкие шифры применяются только в сетях связи с

небольшим объемом передаваемой информации, когда есть возможность обеспечить всех абонентов достаточным запасом случайных ключей и исключить возможность их повторного применения: обычно это сети для передачи особо важной государственной информации.

Стойкость *доказуемо стойких* криптоалгоритмов определяется сложностью решения хорошо известной математической задачи, которую пытались решить многие математики и которая является общепризнанно сложной. В качестве примера можно привести системы DH (Диффи-Хеллмана) [19] и RSA (Ривеста-Шамира-Адельмана) [53, 54], основанные на сложностях дискретного логарифмирования и разложения целого числа на множители соответственно. Достоинством доказуемо стойких алгоритмов является хорошая изученность задач, положенных в их основу, а недостатком - невозможность в случае необходимости оперативной доработки криптоалгоритмов, т.е. отсутствие гибкости. Повышение стойкости достигается увеличением размера математической задачи или ее заменой, что, как правило, влечет цепь изменений в аппаратуре, используемой для шифрования.

Предположительно стойкие криптоалгоритмы основаны на сложности решения частной математической задачи, которая не сводится к хорошо известным задачам и которую пытались решить один или несколько человек. Примерами могут служить шифры ГОСТ 28147-89 [83], AES [52], FEAL [63]. Предположительно стойкие криптоалгоритмы характеризует сравнительно малая изученность математических задач, на которых базируется их стойкость. Однако такие шифры обладают большой гибкостью, что позволяет при обнаружении слабых мест не отказываться от алгоритмов, а проводить их доработку.

Создание новых методов криптоанализа и повышение эффективности существующих методов необходимы как для анализа стойкости криптографических средств, так и для разработки методов их вскрытия. Каждый новый метод криптоанализа приводит к пересмотру безопасности шифров, к которым он применим.

Последнее десятилетие ознаменовалось резким ростом числа открытых работ по криптологии, а криптоанализ становится одной из наиболее активно развивающихся областей исследований. Появился целый арсенал математических методов, представляющих интерес для криптоаналитика.

Универсальные методы криптоанализа

Если целью криптоаналитика является раскрытие возможно большего числа шифров (независимо от того, хочет ли он этим нанести ущерб обществу, предупредить его о возможной опасности или просто получить известность), то для него наилучшей стратегией является разработка универсальных методов анализа [84, 99]. Но эта задача является также и наиболее сложной.

Метод полного перебора

Часто криптоаналитики вскрывают шифры на ЭВМ методом перебора ключей. В процессе криптоанализа приходится перебирать миллиард ключей со скоростью тысяча ключей в секунду.

Предположим, злоумышленнику известна одна или несколько пар (x, y). Пусть для простоты для любой пары (x, y) существует единственный ключ k, удовлетворяющий соотношению $E_k(x) = y$. Упорядочим множество возможных ключей (пространство ключей) и будем последовательно проверять ключи из K на выполнения равенства $E_k(x) = y$. Если считать проверку одного варианта ключа $k \in K$ за одну операцию, то полный перебор ключей потребует $|K|$ операций, где $|K|$ - число

элементов в множестве. Пусть ключ в схеме шифрования выбирается случайно и равновероятно из множества K. Тогда с вероятностью $\frac{1}{|K|}$ ключ будет угадан и трудоемкость метода полного перебора будет равна 1. Поэтому естественно в качестве оценки трудоемкости метода взять математическое ожидание случайной величины α, где α - число опробований до момента обнаружения использованного ключа. Поскольку α - равномерно распределенная случайная величина, то $M(\alpha) = \frac{|K|}{2}$.

Алгоритмы полного перебора допускают распараллеливание, что позволяет значительно ускорить нахождение ключа. Известно два направления в организации параллельного вычисления ключа [76].

Во-первых, построение конвейера. Пусть алгоритм соотношения $E_k(x) = y$ представим в виде детерминированной цепочки простейших действий (операций): $O_1, O_2, \ldots, O_N$.

Возьмем N процессоров $A_1, A_2, \ldots, A_N$, зададим их порядок и положим, что i-ый процессор выполняет три одинаковые по времени операции:

1) прием данных от $(i-1)$-го процессора;

2) выполнение операции O_i;

3) передача данных следующему $(i+1)$-му процессору.

Тогда конвейер из N последовательно соединенных, параллельно и синхронно работающих процессоров работает со скоростью $\frac{v}{3}$, где v - скорость выполнения одной операции процессором.

Второе направление распараллеливания состоит в том, что множество K разбивается на непересекающиеся подмножества $K_1, K_2, \ldots, K_Q$. Система из Q машин перебирает ключи так, что *i*-ая машина осуществляет перебор ключей из множества $K_i, i = \overline{1,Q}$. Система прекращает работу, если одна из машин нашла ключ. Самой большой сложностью в изложенном подходе является организация деления ключевого множества. Однако если организовать поиск ключа таким образом, что при каждом очередном опробовании каждый из N процессоров стартует со случайной точки, то время опробования увеличится, но схема значительно упростится. Как показано в работе [84], среднее число шагов опробования N процессорами (машинами) ключей из множества K в этом случае составляет $\frac{|K|}{N}$.

Реализация такого параллелизма предполагает различные решения. Самое очевидное решение - создание компьютерного вируса для распространения программы-взломщика в глобальной сети. Идея впервые была опубликована в [72]. Вирус должен использовать периоды простоя компьютера (по данным исследований, компьютер простаивает 70-90% времени) для осуществления перебора по множеству ключей. Рано или поздно один из зараженных компьютеров обнаружит искомый ключ (необходимо предусмотреть механизм оповещения злоумышленника); с ростом производительности компьютеров и скорости распространения вирусов угроза успешного исхода такой атаки растет.

В [51] описаны более оригинальные идеи распараллеливания вычислений: «Китайская лотерея», создание «криптоаналитических» водорослей и животных. Китайская лотерея предполагает, что в каждый радиоприемник и телевизор встроена микросхема, запрограммированная на автоматическую проверку различных множеств ключей после получения по эфиру пары открытый текст/шифртекст. Использование

биотехнологий в перспективе сделает возможным осуществление более эффективных атак. Рассмотрим вымышленное животное под названием “DESозавр”. Оно состоит из оптически прозрачных биологических клеток, умеющих тестировать возможные ключи. По какому-то широковещательному оптическому каналу в клетки передаются пары открытый текст/шифртекст. Решения переносятся к органам речи DESозавра специальными клетками, путешествующими по кровеносной системе животного. В доисторические времена средний динозавр состоял примерно из 10^{14} клеток (без микробов). Если каждая клетка может выполнять миллион шифрований в секунду, вскрытие 56-битового ключа займет $7 \cdot 10^{-4}$ сек, а 64-битового - не более 0,2 сек. Другой биологический подход заключается в создании методами генной инженерии криптоаналитических водорослей, умеющих вскрывать криптографические алгоритмы методом полного перебора. Водоросли могут покрывать большие пространства, что теоретически позволит создать некое подобие распределенного компьютера с огромным числом процессоров.

Теперь рассмотрим случай, когда криптоаналитик осуществляет атаку на основе только шифртекста. При осуществлении попытки определения ключа шифра по криптограмме путем ее расшифрования на разных ключах требуется некоторым образом анализировать выходные данные алгоритма и проверять их «осмысленность». Сегодня в качестве объекта шифрования может выступать графический файл или программа. В этом случае задача определения «осмысленности» выходных данных становится очень трудной. Рассмотрим более простой случай, а именно - защиту передаваемых текстовых сообщений. Когда известно, что открытый текст представляет собой предложение на естественном языке, проанализировать результат и опознать успешный исход дешифрования сравнительно несложно, тем более что нередко криптоаналитик располагает некоторой априорной информацией о содержании сообщения [87]. Требуется по небольшому отрезку текста решить, что собой представляет дешифрованный текст: осмысленное сообщение или набор случайных символов. Однако вручную выполнить анализ множества фрагментов дешифрированных текстов невозможно. Поэтому задачу выделения осмысленного текста (то есть обнаружение правильно дешифрированного текста) решают с помощью ЭВМ. В этом случае используют теоретические положения, разработанные в конце XIX века петербургским математиком Марковым А.А., - так называемые цепи Маркова [75].

Тем не менее, возможно, что несколько вариантов пройдут критерий на открытый текст. К. Шеннон привел следующий пример. Криптограмму WNAJW, полученную при использовании сдвигового шифра для шифрования текста на английском языке, порождают два открытых текста RIVER и ARENA, отвечающим ключам F (=5) и W (=22). При этом один из ключей является истинным, а другой – ложным. Для сдвигового шифра одинаковые криптограммы порождают и более длинные слова SULPHUR (сера) и PRIMERO (запал, учебник). Аналогичные примеры имеются и для русского языка: АГАТА – ОСОБО, БЕДНЯГ – КОНЧИМ, КОНОПЛЕЮ – ОТСТУПИВ и т.д. В [87] получены оценки числа ложных ключей для произвольного шифра замены. Так, среднее число ложных ключей θ_L относительно всех возможных шифртекстов длины L определяется формулой:

$$\theta_L = \sum_{v \in V^L} p(v) \cdot \theta_L(v),$$

где V^L - множество криптограмм длины L, $p(v)$ - вероятность появления криптограммы v, $\theta_L(v)$ - число ложных ключей, соответствующих данной криптограмме.

Противник заинтересован в получении некоторой вероятностной информации об исходном тексте сообщения. Например, известный факт написания текста на английском языке предоставляет криптоаналитику некоторую априорную информацию об этом сообщении даже до анализа шифровки. В этом случае он заранее знает, что слово

“HELLO” является более вероятным началом сообщения, чем набор букв “FGHKM”. Поэтому одной из целей криптоанализа может являться увеличение информации, относящейся к каждому возможному сообщению. Предположим, противник перехватил шифровку “ABCCD” и знает (или предполагает), что использованный шифр – это шифр простой замены. Анализ шифровки позволяет сделать вывод, что исходное сообщение состоит из пяти букв, причем на третьей и четвертой позициях стоит одна и та же буква, а остальные отличны от нее и различны между собой. Противник не может считать, что это сообщение “HELLO”, потому что имеются и другие возможные сообщения, например, “TEDDY”. Однако апостериорные вероятности таких открытых текстов возрастают относительно их априорных вероятностей. В то же время апостериорная вероятность таких открытых текстов, как “PEACE” или “GATES”, снижается до нуля вне зависимости от их априорных вероятностей. По Шеннону, криптосистема является совершенной, если после анализа закрытых текстов апостериорные вероятности возможных открытых текстов остаются такими же, какими были их априорные вероятности [102].

Можно подумать, что с ростом мощности компьютеров разрядность ключа, достаточная для обеспечения безопасности информации против атаки методом полного перебора, будет неограниченно расти. Однако это не так. Существуют фундаментальные ограничения вычислительной мощности, наложенные структурой вселенной: например, скорость передачи любого сигнала не может превышать скорость распространения света в вакууме, а количество атомов во Вселенной (из которых, в конечном счете, состоят компьютеры) огромно, но конечно. Так, например, в [18] описаны два фундаментальных ограничения:

1. Предел, основанный на выделяемой Солнцем энергии. Все вычисления потребляют энергию. Согласно принципам классической термодинамики и статистической механики, потребление энергии при осуществлении необратимого преобразования (вычисления) имеет порядок $k \cdot T$, где T – температура окружающей среды (по абсолютной шкале Кельвина), а k – постоянная Больцмана (равная $1.4 \cdot 10^{-23}$ Дж/ ^{o}K). Мощность излучения Солнца составляет приблизительно $3.86 \cdot 10^{26}$ Вт; таким образом, за весь свой предполагаемый период существования – 10 млрд. лет, или $3 \cdot 10^{17}$ секунд – Солнце выделит около 10^{44} Дж). Предположим, температура окружающей среды $T = 10^{-6}$ градусов, тогда энергозатраты на одну операцию составляют $1.4 \cdot 10^{-29}$ Дж. Значит, количество вычислительных операций, которые можно осуществить с использованием всей выделяемой солнцем энергии, равно выделяемой мощности, поделенной на количество энергии, необходимой для осуществления одной операции, т.е. всего 10^{73} операций. Такое количество операций потребовалось бы на взлом ключа из 73 десятичных цифр (или около 250 бит) методом прямого перебора при грубом предположении, что для проверки одного значения ключа необходима всего одна операция (на самом деле - сотни операций). Для справки, количество атомов солнечной системы – около 10^{60}.

2. Предел, основанный на массе Земли. Масса Земли составляет порядка $6 \cdot 10^{24}$ кг. Масса протона – $1.6 \cdot 10^{-27}$ кг, т.е. Земля содержит приблизительно $4 \cdot 10^{51}$ протонов. Сопоставим каждому протону отдельный компьютер и примем за скорость выполнения операции на таком компьютере время, за которое луч света проходит расстояние, равное диаметру этого протона ($\frac{10^{-15}\ \text{м}}{3 \cdot 10^{10}\ \text{м/с}} = \frac{1}{3} \cdot 10^{-25}$ с). Таким образом, каждый компьютер может выполнять $3 \cdot 10^{25}$ операций в секунду. Если все эти компьютеры будут работать параллельно, их суммарное быстродействие составит $4 \cdot 10^{51} \cdot 3 \cdot 10^{25}$ операций в секунду, т.е. 10^{77} операций в секунду. Возраст Вселенной приблизительно 10 млрд. лет, т.е. $3 \cdot 10^{17}$ секунд. За весь период существования Вселенной такие гипотетические

компьютеры размером с протон смогли бы выполнить $3 \cdot 10^{94}$ операций. При описанных в п. 1 предположений относительно количества операций, необходимых на проверку значения ключа, такое количество операций позволит взломать ключ длиной 95 десятичных цифр, или 320 бит.

Таким образом, минимальный размер ключа, необходимый для защиты информации от атак злоумышленника, будет расти по мере повышения быстродействия компьютеров; тем не менее, приведенные выше вычисления показывают, что существует возможность выбрать такую длину ключа, что атаку методом полного перебора будет осуществить в принципе невозможно, вне зависимости от повышения вычислительной мощности компьютеров или успехов в области классической теории алгоритмов.

Атака по ключам

Согласно [88], одной из причин ненадежности криптосистем является использование слабых ключей. Согласно уже упомянутому принципу Кирхгофа, стойкость криптоалгоритма определяется секретностью ключа. Слабый ключ – это ключ, не обеспечивающий достаточного уровня защиты или использующий в шифровании закономерности, которые могут быть взломаны. Обычно считается, что алгоритм симметричного шифрования должен по возможности не иметь слабых ключей. Если это невозможно, то количество слабых ключей должно быть минимальным, чтобы уменьшить вероятность случайного выбора одного из них. Тем не менее, все слабые ключи должны быть заранее известны, чтобы их можно было отбраковать в процессе создания ключа.

Генераторы случайных чисел – ещё одно уязвимое место криптографических систем [59]. Это означает, что, если для генерации ключей используется криптографический слабый алгоритм, независимо от используемого шифра вся система будет нестойкой. Качественный ключ, предназначенный для использования в рамках симметричной криптосистемы, представляет собой случайный двоичный набор. Если требуется ключ разрядностью n, в процессе его генерации с одинаковой вероятностью должен получаться любой из 2^n возможных вариантов. Генерация ключей для асимметричных криптосистем – процедура более сложная, т.к. ключи, применяемые в таких системах, должны обладать определенными математическими свойствами. Например, в случае системы RSA модуль шифрования представляет собой произведение двух больших простых чисел.

Рассмотрим понятие криптографически стойкого генератора псевдослучайных кодов, или, для краткости, псевдослучайного генератора, которое было введено Блюмом и Микали [25]. Пусть $g : \{0,1\}^n \to \{0,1\}^{q(n)}$ - функция, вычислимая за полиномиальное от n время, $q(n)$ - некоторый полином. Такая функция называется *генератором*. Генератор g является *псевдослучайным* [80], если порождаемые им последовательности неотличимы никаким полиномиальным вероятностным алгоритмом от случайных последовательностей той же длины $q(n)$. В 1989-1990 гг. Импальянцо, Левин и Луби [78] и Хостад [22] доказали, что псевдослучайные генераторы существуют тогда и только тогда, когда существуют односторонние функции.

С помощью псевдослучайных генераторов можно строить стойкие криптосистемы. Хорошие генераторы случайных чисел сложны в разработке, так как их надёжность часто зависит от особенностей аппаратного и программного обеспечения. В связи с этим ведется поиск способов построения эффективных псевдослучайных генераторов на основе различных криптографических предположений. Для генерации ключевой информации, предназначенной для использования в рамках симметричной криптосистемы, используются следующие методы (в порядке возрастания качества) [88]:

- Программная генерация, предполагающая вычисление очередного псевдослучайного числа как функции текущего времени, последовательности символов, введенных пользователем, особенностей его клавиатурного почерка и т.д.;
- Программная генерация, основанная на моделировании качественного псевдослучайного генератора с равномерным законом распределения;
- Аппаратная генерация с использованием качественного псевдослучайного генератора;
- Аппаратная генерация с использованием генераторов случайных последовательностей, построенных на основе физических генераторов шума и качественных псевдослучайных генераторов.

Показателем эффективности служит количество операций, затрачиваемых на вычисление каждого очередного бита псевдослучайной последовательности. Псевдослучайный генератор характеризуются периодом, разбросом, а также необходимостью его инициализировать. Малый период и плохой разброс относятся к недостаткам псевдослучайного генератора. В случае малого периода (когда псевдослучайных значений, вырабатываемых генератором, меньше, чем возможных значений ключа) злоумышленник может сократить время поиска ключа, перебирая не сами ключи, а псевдослучайные значения, и генерируя из них ключи. Невысокое качество программных методов формирования объясняется также необходимостью защиты от разрушающих программных воздействий.

Лучший способ генерации множества случайных битов – извлечение их из естественно случайных событий реального мира [58]. Часто такой метод требует наличия специальной аппаратуры, но можно реализовать его и на компьютерах. Дж. Б. Эгнью предложил генератор случайных битов, который можно встроить в СБИС [2]. Это конденсатор металл-изолятор-полупроводник. Два таких конденсатора помещаются рядом друг с другом, а случайный бит определяется разностью зарядов этих конденсаторов. Другой генератор случайных битов генерирует поток случайных битов, используя нестабильность частоты свободно колеблющегося осциллятора [21]. Коммерческая микросхема фирмы AT&T генерирует случайные числа, опираясь именно на это явление [5], а генератор М. Гьюда [28] собирает случайные числа из физических явлений (например, радиоактивного распада). В качестве случайных величин можно также рассматривать интервалы между нажатиями клавиш клавиатуры. Главный недостаток подобных систем – возможные закономерности в генерируемой последовательности. Используемые физические процессы могут быть случайны, однако использование измерительных инструментов может привести к появлению проблем: смещения, отклонения или корреляции между битами. Обойти эти недостатки можно, используя не один, а несколько случайных источников. В качестве случайных событий Брюс Шнайер предлагает рассматривать, например:

- моменты нажатия на клавиши;
- моменты поступления команд от мыши;
- номер сектора, время дня и задержку поиска для каждой дисковой операции;
- фактической положение курсора мыши;
- номер текущей строки развертки монитора;
- содержимое текущего выводимого на экран изображения;
- содержимое таблиц файловой системы FAT;
- момент окончания загрузки процессора;
- время поступления сетевых пакетов и т.д.

Применяя к этим событиям однонаправленную функцию, можно сохранять полученные случайные величины в накопителе (пуле) и при необходимости их извлекать.

Counterpane опубликовала новый класс атак на генераторы случайных чисел [33], основанный на работе компании над коммерческими моделями. Одна из самых неожиданных находок была в том, что определённые генераторы случайных чисел могут быть надёжными при использовании с одной целью, но ненадёжными для другой; обобщение анализа надёжности опасно.

Частотный анализ

На протяжении веков дешифрованию криптограмм помогает частотный анализ появления отдельных символов и их сочетаний. Вероятности появления отдельных букв в тексте сильно различаются. Для русского языка, например, буква "о" появляется в 45 раз чаще буквы "ф" и в 30 раз чаще буквы "э". Анализируя достаточно длинный текст, зашифрованный методом замены, можно по частотам появления символов произвести обратную замену и восстановить исходный текст.

В таблице 1 [75] приведены относительные частоты появления русских букв.

Буква	**Частота**	**Буква**	**Частота**	**Буква**	**Частота**	**Буква**	**Частота**
о	0.09	в	0.038	з	0.016	ж	0.007
е, ё	0.072	л	0.035	ы	0.016	ш	0.006
а	0.062	к	0.028	б	0.014	ю	0.006
и	0.062	м	0.026	ь,ъ	0.014	ц	0.004
н	0.053	д	0.025	г	0.013	щ	0.003
т	0.053	п	0.023	ч	0.012	э	0.003
с	0.045	у	0.021	й	0.01	ф	0.002
р	0.04	я	0.018	х	0.009		

Табл.1

Относительная частота появления пробела или знака препинания в русском языке составляет 0,174. Кроме того, порядок букв в словах и фразах естественного языка подчиняется определенным статистическим закономерностям. Частотный анализ также учитывает частоту появления различных буквосочетаний: например, пара стоящих рядом букв «ся» в русском языке более вероятна, чем «цы», а «оь» не встречается никогда. Для большинства естественных языков такая статистика документирована.

Эти принципы широко применяются в распространенных сегодня программах по подбору паролей. Возможные методы подбора пароля (могут применяться в совокупности) [86]:

- неоптимизированный перебор;
- перебор, оптимизированный по словарям вероятных паролей;
- перебор, оптимизированный на основе встречаемости символов и биграмм;
- перебор, ориентированный на информацию о подсистеме аутентификации ОС. Если ключевая система ОС допускает существование эквивалентных паролей, при переборе из каждого класса эквивалентности опробуется всего один пароль;
- перебор с использованием знаний о пользователе. Как правило, опробуются пароли, использование которых представляется наиболее вероятным.

Если программа перебора вначале подбирает наиболее вероятные пароли, а менее вероятные оставляет на потом, то перебор сокращается в десятки и сотни раз. В [80]

приводится ряд результатов, полученных при подборе пароля. Числа, указанные в первой колонке таблицы 2, соответствуют сложности полного перебора. Однако применялся оптимизированные перебор, а в первом случае пароль представлял собой два английских слова, записанных без пробела. Таким образом, время перебора сократилось во много раз. Во втором же случае пароль состоял из трех строчных английских букв, двух заглавных английских букв и одной цифры и был абсолютно бессмысленным.

Сложность перебора	Время перебора	Тип процессора
$2,08 \cdot 10^{11}$	15 минут	486DX/4-100
$5,68 \cdot 10^{10}$	8 часов	Pentium-120

Табл.2

Частотный криптоанализ использует статистические и лингвистические методы для получения дополнительной информации о ключе, а аналитические методы предполагают математическое изучение алгоритма шифрования. Каждый новый метод криптоанализа добавляет новые требования к алгоритмам шифрования. Так, частотный метод, в котором по распределению символов в шифртексте выдвигаются гипотезы о ключе шифрования, породил требование равномерного распределения символов в шифртексте. С ростом сложности алгоритмов постепенно стал доминировать математический подход. Такая тенденция проявилась особенно отчетливо во время Второй Мировой Войны, когда взлом шифров потребовал применения нетривиальных математических выкладок.

Метод Полларда

Многие задачи криптоанализа, например, логарифмирование в группе вычислимого порядка или поиск коллизий хэш-функции, сводятся к задаче о встрече на графе случайного отображения, которая решается алгоритмом Полларда [49]. Приведем схематическое описание метода Полларда, основываясь на статье [99]. Пусть на некотором конечном множестве M определено случайное отображение f. Применим это отображение поочередно ко всем элементам $x \in M$. Соединим пару $x, y \in M$ дугой, ведущей из x в y при условии, что $y = f(x)$. Полученный граф является *волновым графом первого рода* [73], или *функциональным* [100]. Каждая компонента связности функционального графа представляет собой деревья, вырастающие из цикла. В графе случайного отображения почти все вершины лежат на одном дереве. При обращении стрелок случайное дерево описывается критическим ветвящимся процессом [90]. Для случайного отображения f длина цикла равна и высота дерева в среднем равны $O(\sqrt{|M|})$.

Для нахождения ключа в криптоалгоритме, основанном на задаче логарифма на некоторой группе, достаточно решить задачу о встрече на графе случайного отображения. Для этого из двух разных стартовых точек x_0', x_0'' строятся траектория до входа в цикл. Затем одна из двух конечных точек, лежащих в цикле, фиксируется, а траектория другой продолжается до встречи с фиксированной точкой. Для функции f и точки входа x_0 длина траектории составляет $O(\sqrt{|M|})$ шагов. Типичный вид этой траектории содержит предельный цикл длины $O(\sqrt{|M|})$ и отрезок до входа в цикл примерно такой же длины. В качестве индикатора замыкания траектории Поллард предложил использовать равенство $x_i = x_{2i}$, где x_i - i-я точка траектории для входа x_0. Всегда найдется значение i, при

котором будет выполняться это равенство. Причем значение индекса i не превышает суммы длины пути до входа в цикл.

В качестве иллюстрации рассмотрим, каким образом можно использовать метод Полларда для нахождения коллизии (двух аргументов, дающих одинаковое значение хэш-функции). Такими аргументами будут элементы множества M, стрелки от которых под действием хэш-функции f ведут в точку входа в цикл. Параметром алгоритма является число v, определяющее доступную память (объем памяти ограничен величиной $O(v)$). Практически алгоритм сводится к нахождению точки входа в цикл и может быть сформулирован следующим образом:

1. Войти в цикл, используя равенство $x_i = x_{2i} = t$.

2. Измерить длину цикла m, применяя последовательно отображение f к элементу t до получения равенства $f^m(t) = t$.

3. Разбить цикл m на v интервалов одинаковой или близкой длины и создать базу данных, запомнив и упорядочив начальные точки интервалов.

4. Для стартовой вершины п.1 выполнять одиночные шаги до встречи с какой-либо точкой из базы данных п.3. Отметить начальную и конечную точки интервала, на котором произошла встреча.

5. Стереть предыдущую и создать новую базу данных из v точек, разбив интервал, на котором произошла встреча, на равные по длине части, запомнив и отсортировав начальные точки интервалов.

6. Повторить процедуры пп.4,5 до тех пор, пока не получится длина интервала, равная 1. Вычислить точку встречи в цикле, вычислить коллизию как пару вершин, одна из которых лежит в цикле, а другая - нет.

Этот алгоритм требует многократного выполнения $O(\sqrt{|M|})$ шагов до входа в цикл и выполнении сортировки базы данных. На каждом этапе при создании новой базы данных длина интервала сокращается в v раз, то есть количество повторов равно $O(\log_v |M|)$). Если $v << \sqrt{|M|}$ то сложностью сортировки базы данных можно пренебречь. Тогда сложность алгоритма равна $O(\sqrt{|M|} \cdot \log_v |M|)$.

Метод Полларда также применяется для решения задачи логарифма на циклической группе, поиска частично эквивалентных ключей и в некоторых других случаях. Применительно к задаче логарифмирования этот метод позволяет отказаться от использования большого объема памяти по сравнению с методом встречи посередине. Кроме того, пропадает необходимость сортировки базы данных. В результате временная сложность по сравнению с методом встречи посередине снижается на множитель $O(\log |M|)$. Сложность этого метода в данном случае составляет $O(\sqrt{|M|})$ шагов и требует памяти объема $O(1)$ блоков.

Метод «встречи посередине»

Другой популярный метод криптоанализа - алгоритм «встречи посередине» [99], требует такого же объема памяти, как метод Полларда, но при этом поддается эффективному распараллеливанию. Например, для логарифмирования в группе порядка p при параллельной работе n процессоров, где $n \gg p$, время работы алгоритма

уменьшается в n раз. Данный метод криптоанализа основан на *"парадоксе дней рождения"*. Известно, что если считать, что дни рождения распределены равномерно, то в группе из 24 человек с вероятностью 0,5 у двух человек дни рождения совпадут. В общем виде этот парадокс формулируется так: если $a\sqrt{b}$ предметов выбираются с возвращением из некоторой совокупности размером b, то вероятность того, что два из них совпадут, равна $1-e^{-\frac{a^2}{2}}$ (в описанном частном случае $b=365$ - количество дней в году, $a\sqrt{b}=24$, т.е. $a\approx 1,256$).

Если множество ключей криптоалгоритма замкнуто относительно композиции, т.е. для любых ключей k_i и k_j найдется ключ k_r такой, что результат шифрования любого текста последовательно на k_i и k_j равен криптограмме этого же числа на k_r, т.е. $F(k_j, F(k_i, x)) = F(k_r, x)$, то можно воспользоваться этим свойством. Пусть нам нужно найти ключ k_r. Тогда для нахождения ключа k_r необходимо найти эквивалентную ему пару ключей k_i, k_j.

Пусть известен открытый текст x и его криптограмма y. Для текста x строим базу данных, содержащую случайное множество ключей k' и соответствующих криптограмм $w = F(k', x)$, и упорядочиваем ее по криптограммам w. Объем базы данных выбираем $O(\sqrt{|\{k'\}|})$, где $|\{k'\}|$ - мощность множества ключей k'. Затем подбираем случайным образом ключи k'' для расшифровки текстов y и результат расшифрования $v = F(k'', y)$ сравниваем с базой данных. Если текст v окажется равным одной из криптограмм w, то ключ $k'k''$ эквивалентен искомому ключу k. Этот метод также применим, если множество ключей содержит достаточно большое подмножество, являющееся полугруппой.

Обозначим $\alpha = |\{k\}|$ - общее количество возможных ключей k. Временная сложность метода составляет $O(\sqrt{\alpha}\log\alpha)$. Множитель $\log\alpha$ учитывает сложность сортировки. Требуемая память равна $O(\sqrt{\alpha}\log\alpha)$ бит, или $O(\sqrt{\alpha})$ блоков (предполагается, что длина блока и длина ключа различаются на ограниченную константу).

Алгоритм является вероятностным. Однако существуют и детерминированный аналог этого алгоритма "giant step - baby step" с такой же сложностью, предложенный американским математиком Д. Шенксом [62].

Метод коллизий для хэш-функций

Электронно-цифровая подпись (ЭЦП) - это некоторая информация, позволяющая убедиться в подлинности документа, определить его автора, проверить время и дату подписания документа и т.д. (функциональный набор может быть различным).

Приведем классификацию атак на схемы электронной подписи [88]:

- **Атака на основе известного открытого ключа** – самая слабая из атак, практически всегда доступна противнику;
- **Атака на основе известных подписанных сообщений** – в распоряжении противника имеется некоторое число пар (M, C), где M - некоторое сообщение, а C - допустимая подпись для него, при этом противник не может влиять на выбор M;

- **Простая атака с выбором подписанных сообщений** – противник имеет возможность выбрать некоторое количество подписанных сообщений, при этом открытый ключ он получает после этого выбора;
- **Направленная атака с выбором сообщений** – выбирая подписанные сообщения, противник знает открытый ключ;
- **Адаптивная атака с выбором сообщений** – противник знает открытый ключ, выбор каждого следующего подписанного сообщения он может делать на основе знания допустимой подписи предыдущего выбранного сообщения.

Для контроля целостности сообщений, передаваемых по каналу связи, и генерации ЭЦП используются хэш-функции - это математические или иные функции, которые принимают на входе строку переменной длины (называемую прообразом) и преобразуют ее в выводную сроку фиксированной (обычно меньшей) длины, называемую значением хэш-функции или сверткой [58]. Более формально определение представлено в [84]. Пусть $A = \{A_1, \dots, A_m\}$ - алфавит, A^* - множество слов конечной длины в алфавите A, A^l - множество слов длины l, l - длина значения хэш-функции. Пусть определена функция $H : A^* \to A^l$, которая обладает тем свойством, что значения функции H на словах, которые даже при отличии друг от друга только в одном знаке дают значительно отличающиеся хэш-значения. Тогда, получив по каналу связи сообщение и его хэш, можно вычислить хэш-значение от сообщения, сравнить с полученным значением и проверить, не было ли сообщение искажено при передаче. Если функция H зависит также от симметричного ключа $k \in K$, то, помимо проверки целостности, добавление хэш-значения к сообщению подтверждает истинность сообщения. Такой способ подтверждения истинности называется кодом аутентификации (Message Autentification Code - MAC). Однако такое подтверждение истинности еще не является электронной подписью. Подтверждение истины называется подписью, если ее могут проверить все, не знающие секретного ключа. Для того чтобы код аутентификации стал электронной подписью сообщения, необходимо использовать хэш-функции с дополнительными свойствами (например, на основе асимметричной криптосистемы). Если k - секретный ключ, k' - открытый ключ, E_k и $D_{k'}$ - функции зашифрования и расшифрования соответственно, то электронной подписью документа $M \in A^*$ называется

$$C = D_{k'}(H(M))$$

Функции E_k и $D_{k'}$ - взаимно обратные, поэтому для проверки подписи под документом M достаточно вычислить $\widetilde{X} = H(M)$ и $\overline{X} = E_k(C)$. Если $\widetilde{X} = \overline{X}$, то автором документа M может являться только обладатель секретного ключа k'.

Перечислим основные свойства хэш-функций:

1. Если $M \neq M'$ хотя бы в одном знаке, то $H(M) \neq H(M^*)$ примерно на половине знаков. (могут найтись такие $M \neq M'$, что $H(M) = H(M^*)$, но выбор таких M и M' случайно маловероятен, а подбор труден).
2. Для произвольной строки $\alpha \in A^l$ трудно подобрать $M \in A^*$ такое, что $H(M) = \alpha$.
3. $H(M)$ вычисляется быстро.

Стойкость схемы ЭЦП зависит от стойкости используемых криптоалгоритмов и хеш-функций. Основная атака на хэш - это метод коллизий [84]. Пусть B - мошенник, а D - лицо, подвергающееся атаке. B готовит два документа M и M' так, что документ M пользователь D охотно подпишет, а B заинтересован в подписи D под документом M'. Тогда, подписав у D документ M, B получает подпись C, которая представляет

собой некоторую функцию F от хэша сообщения: $C = F(H(M))$. Если M' выбрано так, что $H(M) = H(M')$, то B может предъявить пару (M', C): тогда атака удалась.

Каким образом можно выбрать два таких сообщения? По свойству 2, подбор осуществить сложно.

Оказывается, с большой вероятностью злоумышленник может реализовать подбор такого сообщения следующим образом. Такой подбор основан на задаче о днях рождения (описан выше в разделе «Метод встречи посередине»). B выбирает два нужных сообщения M и M', при этом D может подписать сообщение M, но не подпишет сообщение M'. Варьируя в сообщениях интервалы, шрифты, формат или внося в текст незначительные и т.п., B получает n пар вариантов M и M' без изменения их смысла. И, следовательно, n пар соответствующих им значений хэш-функций:

$$H(M_1) \qquad H(M_1')$$
$$H(M_2) \qquad H(M_2')$$
$$\dots$$
$$H(M_n) \qquad H(M_n')$$

Сообщения $M_1, \dots, M_n$ отличаются слабо, а их хэш-функции - значительно, т.е. можно считать, что значения хэш-функций выбираются случайно, равновероятно и независимо друг от друга. Обозначим $N = |A^l| = |A|^l$, $l \to \infty$; тогда при выборе $n = t\sqrt{N}$ ($t > 0$ - некоторая константа) вероятность того, что имеется пара сообщений M_i и M_i', удовлетворяющих условию $H(M_i) = H(M_j')$, вычисляется по формуле $p = (1 - e^{-t^2})(1 + o(1))$ [84]. Поиск двух таких сообщений можно выполнить с использованием метода “встречи посередине” или метода Полларда, описанных выше.

Методы криптоанализа асимметричных криптосистем

Практически все используемые алгоритмы асимметричной криптографии основаны задачах факторизации и дискретного логарифмирования в различных алгебраических структурах. Несмотря на то, что принадлежность этих задач к классу NP-полных задач не доказана, на сегодняшний день не найден полиномиальный алгоритм их решения. Для криптоанализа асимметричных криптосистем можно применять методы «встречи посередине» и Полларда, описанные выше. Однако есть и другие методы, учитывающие специфику систем с открытым ключом. Они заключаются в решении математической задачи, положенной в основу асимметричного шифра.

Криптоанализ систем шифрования, основанных на сложности задачи дискретного логарифмирования

Задача дискретного логарифмирования считается более сложной, чем задача факторизации. Если будет найден полиномиальный алгоритм ее решения, станет возможным и разложение на множители (обратное не доказано).

Последние достижения теории вычислительной сложности показали, что общая проблема логарифмирования в конечных полях не может считаться достаточно прочным фундаментом. Наиболее эффективные на сегодняшний день алгоритмы дискретного логарифмирования имеют уже не экспоненциальную, а субэкспоненциальную временную сложность. Это алгоритмы “index-calculus”, использующие факторную базу. Первый такой

алгоритм был предложен Адлеманом [1] и имеет временную сложность $L_p\left[\frac{1}{2}; c\right]$ при вычислении дискретного логарифма в простом поле $\mathbb{Z}_p$ ($L_N\left[\gamma; c\right] = e^{(c+o(1))(\log N)^{\gamma}(\log\log N)^{1-\gamma}}$, где $0 < \gamma < 1, c = const, c > 0$). Идея использования факторной базы применялась и ранее, например, в [70]. На практике алгоритм [1] оказался недостаточно эффективным; Копперсмит, Одлыжко и Шреппель предложили алгоритм дискретного логарифмирования COS [17] с эвристической оценкой сложности $L_p\left[\frac{1}{2}; 1\right]$ операций. Алгоритм решета числового поля [56], предложенный Широкауэром, при $p > 10^{100}$ работает эффективнее различных модификаций метода COS; его временная сложность составляет порядка $L_p\left[\frac{1}{3}; (64/9)^{1/3}\right]$ арифметических операций.

Пусть G - мультипликативная абелева группа. Вычислить дискретный логарифм b по основанию a в группе G означает найти $x \in G$, при котором $a^x = b$. Свойства дискретного логарифма во многом схожи со свойствами обычного логарифма в поле действительных чисел. Например, выполняется тождество $\log_a(h \cdot j) \equiv \log_a(h) + \log_a(j) \pmod{|G|}$, где $|G|$ - порядок группы, a - образующая.

Основная идея методов “index-calculus” заключается в том, что если

$$\prod_{i=1}^{m} x_i = \prod_{j=1}^{n} y_j$$

для некоторых элементов конечного поля $\mathbb{Z}_p$, то

$$\sum_{i=1}^{m} \log_a x_i \equiv \sum_{j=1}^{n} \log_a y_j \mod (p-1) \qquad (1)$$

Получив достаточно много соотношений (1) (причем хотя бы одно из них должно включать элемент g, для которого $\log_a g$ известен), можно решить систему линейных уравнений относительно неизвестных $\log_a x_i$ и $\log_a y_j$ в кольце вычетов $\mathbb{Z}_{p-1}$ при условии, что количество неизвестных в уравнениях не слишком велико.

Самый простой подход к генерации соотношений вида (1) – выбрать произвольный элемент $g \in \mathbb{Z}_p$, вычислить $u = a^g \pmod p$ и с помощью перебора попытаться найти числа, удовлетворяющие соотношению:

$$u = \prod_{i=1}^{k} p_i^{\alpha_i},$$

где p_i - простые числа, такие, что $p_i < B$ для некоторой границы B. Если удалось найти такие числа, то u является гладким элементом с границей гладкости B [48].

Выделяются два основных этапа в работе алгоритмов: на первой, подготовительной стадии, формируется факторная база и на ее основе генерируется система линейных уравнений в кольце $\mathbb{Z}_{p-1}$; вид факторной базы (множество простых чисел, неприводимых многочленов или других объектов) и способы получения матрицы системы зависят от выбранного алгоритма. На второй стадии (которая является общей для рассматриваемых алгоритмов) требуется получить решение этой системы. Интенсивные

предварительные вычисления для каждого поля достаточно выполнить только один раз. Затем можно быстро вычислять различные дискретные логарифмы. Таким образом, все субэкспоненциальные методы дискретного логарифмирования в конечных полях сводятся к этой задаче решения систем линейных уравнений в кольцах вычетов. В работе [74] показаны пути повышения эффективности методов решения таких систем.

Алгоритмов, осуществляющих дискретной логарифмирование на эллиптических кривых в общем случае хотя бы с субэкспоненциальной сложностью, на сегодняшний день не существует. Тем не менее, известны работы И.А. Семаева [101], в одной из которых рассматривается метод, идейно близкий методам логарифмирования в конечном поле Адлемана [1]. В другой работе для эллиптических кривых специального вида (накладываются некоторые условия на модуль арифметики и на мощность группы точек) И.А. Семаев указал способ сведения с полиномиальной сложностью задачи логарифмирования в группе точек эллиптической кривой к задаче логарифмирования в некотором расширении простого поля. При этом используется так называемое спаривания Вейля [98], после чего можно применять известные субэкспоненциальные методы. Аналогичные результаты опубликованы за рубежом [23].

Криптоанализ систем шифрования, основанных на сложности задачи факторизации

Поиском эффективных способов разложения целых чисел на множители занимаются очень давно. Наиболее очевидный метод разложения числа n на множители – перебор простых делителей не превышающих $\sqrt{n}$.

Существует также другой переборный метод (метод Ферма) [80], который основан на представлении числа n в виде разности квадратов:

$$n = a^2 - b^2 = (a+b)(a-b).$$

Ферма предложил, вычисляя $НОД(n, a-b)$, попытаться найти нетривиальный делитель n. Он предложил способ, позволяющий найти требуемое представление. Если разлагаемое число имеет два не очень различающиеся по величине множителя, этот способ позволяет определить их быстрее, чем простой перебор делителей. Лежандр обратил внимание на то, что при таком подходе достаточно получить сравнение

$$a^2 \equiv b^2 \bmod n \qquad (2)$$

Не каждая пара чисел, удовлетворяющая ему, позволяет разложить n на множители. Для нахождения чисел, связанных соотношением (2), Лежандр использовал *непрерывные дроби* [79]. В работе [45] описано некоторое усовершенствование метода Ферма.

По сложности как простой переборный метод, так и метод Ферма оцениваются величиной $O(\sqrt{n})$, однако последний метод может оказаться эффективнее, если делители числа n близки друг к другу.

Для достаточно больших чисел n время работы таких алгоритмов становится неприемлемым. Прежде чем приступать к факторизации таких чисел, следует убедиться в том, что они действительно составные. Для этого лучше всего использовать один из вероятностных тестов на простоту, например, алгоритм Миллера—Рабина [91].

Обзор методов факторизации с экспоненциальной сложностью для больших чисел можно найти, например, в работах [68, 79]. Эти методы используются в сочетании с субэкспоненциальными методами факторизации, которые будут описаны далее, и применяются, как правило, для предварительного отделения небольших простых делителей у факторизуемого числа.

Существуют и более эффективные методы разложения чисел на множители – это методы, имеющие субэкспоненциальную оценку сложности. Так, вероятностный алгоритм Ленстры [42] для факторизации целых чисел с помощью эллиптических кривых имеет среднюю оценку временной сложности $e^{((2+o(1))\log p \log\log p)^{1/2}} \log^2 n$, где p - минимальный простой делитель n. При замене p на $\sqrt{n}$ получаем субэкспоненциальную оценку сложности $L_n\left[\frac{1}{2};2\right]$. Этот метод может быть эффективно использован для отделения небольших простых делителей. Преимуществом также является использование небольшого объема памяти.

Рассмотрим еще один субъкспоненциальный алгоритм факторизации – метод Диксона [20]. Пусть $n \in \mathbb{N}$ - число, которое мы хотим разложить на множители, $0 < a < 1$ - некоторая постоянная, значение которой будет определено ниже. *Факторной базой* будем называть множество простых чисел p_i таких, что $2 \le p_i \le L^a$ (где $L = e^{\sqrt{\log n \log\log n}}$). Обозначим символом k количество простых чисел в факторной базе, а $Q(m)$ - наименьший неотрицательный *вычет в классе* $m^2 (\mathrm{mod}\, n)$ [79].

Случайным перебором ищем числа $m_1,...,m_{k+1}$ такие, что $1 < m_i < n,\ Q(m_i) = p_1^{\alpha_{i,1}}...p_k^{\alpha_{i,k}}$ $(i = \overline{1,k+1})$. $m_i^2 \equiv Q(m_i) (\mathrm{mod}\, n)$ являются гладкими числами, т.к. могут быть представлены в виде произведения небольших простых чисел (определение гладкого элемента дано выше, в разделе «**Криптоанализ систем шифрования, основанных на сложности задачи дискретного логарифмирования**»). Обозначим $\vec{v}_i = (\alpha_{i,1},...\alpha_{i,k}) \in \mathbb{Z}^k$ - векторы показателей в разложении $Q(m_i)$ на множители ($\mathbb{Z}^k$ - векторное пространство столбцов длины $n \in \mathbb{N}$ с координатами из $\mathbb{Z}$).

Решая систему линейных уравнений $x_1\vec{v}_1 + ... + x_1\vec{v}_{k+1} \equiv \vec{0} (\mathrm{mod}\, 2)$ в векторном пространстве $\mathbb{Z}_2^k$ (k-мерное булево пространство), находим значения $x_1,...,x_{k+1} \in \{0,1\}$, не все из которых равны нулю (такое решение существует, поскольку число уравнений k меньше числа неизвестных).

Для вычисленных значений $x_1,...,x_{k+1}$ справедливо соотношение:

$$(m_1^{x_1}...m_{k+1}^{x_{k+1}})^2 \equiv p_1^{\sum_{i=1}^{k+1} x_i\alpha_{i,1}}...p_k^{\sum_{i=1}^{k+1} x_i\alpha_{i,k}} \quad (\mathrm{mod}\, n)$$

Обозначим $X = m_1^{x_1}...m_{k+1}^{x_{k+1}}$, $Y = \prod_{j=1}^{k} p_j^{\left(\sum_{i=1}^{k+1} x_i\alpha_{i,j}\right)/2}$ (числа $\left(\sum_{i=1}^{k+1} x_i\alpha_{i,j}\right) / 2$ - целые по определению x_i). Получаем соотношение $X^2 \equiv Y^2 (\mathrm{mod}\, n)$. Далее проверяем условие $1 < \mathrm{НОД}(X \pm Y, n) < n$. В случае успеха мы разложили n на множители (вероятность успеха не меньше $1/2$ - доказательство приведено в [79]). В случае неудачи возвращаемся в начало алгоритма и ищем другие значения m_i.

При выборе границы факторной базы равной $L^2 = e^{2\sqrt{\log n \log\log n}}$ временная сложность алгоритма Диксона оценивается как $L_n\left[\frac{1}{2};2\right]$. Существует несколько стратегий, позволяющих повысить эффективность алгоритма Диксона [50]:

- Стратегия LP (использование больших простых чисел)
- Стратегия PS (применение алгоритма Полларда—Штрассена)
- Стратегия EAS (стратегия раннего обрыва)

В алгоритме Бриллхарта-Моррисона [15] случайный выбор m из алгоритма Диксона заменяется надет детерминированное определение очередного значения m, для которого мы ищем разложение $m^2 (\mathrm{mod}\, n)$ на простые множители из факторной базы. Этот выбор m делается с помощью непрерывных дробей для числа $\sqrt{n}$. Данный метод факторизации был наиболее популярным до появления в 1981 г. алгоритма квадратичного решета Померанца [50]. Покажем его основную идею, следуя работе [80].

В качестве факторной базы выберем ограниченные по величине некоторым параметром простые числа p_i, такие, что $\left(\frac{n}{p_i}\right) = 1$, где $\left(\frac{\alpha}{\beta}\right)$ - так называемый *символ Якоби* [79, 80]. Обозначим буквой s количество таких чисел. Обозначим $m = \left\lfloor \sqrt{n} \right\rfloor$, $Q(t) = (t+m)^2 - n \equiv H(t)^2$, где $H = t + \left\lfloor \sqrt{n} \right\rfloor$. При малых целых значениях t величина $Q(t)$ будет сравнительно невелика. На следующем шаге, вместо того чтобы перебирать числа t и раскладывать соответствующие значения $Q(t)$ на множители, алгоритм сразу же отсеивает «ненужные» значения t, оставляя только те, для которых $Q(t)$ имеет делители среди элементов базы факторной базы:

$$A = Q(t) = \prod_{j=1}^{s} p_j^{\gamma_j},$$

т.е. $Q(t)$ раскладывается в факторной базе. Тогда, обозначая $B = H(t)$, получаем сравнение $B^2 = A^2 (\mathrm{mod}\, n)$, и, накопив достаточно много таких соотношений, проводим исключение переменных и строим соотношение $X^2 \equiv Y^2 (\mathrm{mod}\, n)$ так же, как в алгоритме Диксона.

Эвристическая оценка сложности усовершенствованного алгоритма квадратичного решета составляет $L_n\left[\frac{1}{2}; 1\right]$ арифметических операций. Рекордное значение для факторизованных этим методом чисел составляет 129-значное RSA-число n [6]. Метод квадратичного решета следует применять для факторизации, если число n не превосходит 10^{110}. Для чисел большей величины следует использовать алгоритмы решета числового поля [41]. Решето числового поля по сути не является алгоритмом: это метод вычисления, состоящий из нескольких этапов, и каждый из этих этапов обслуживается несколькими алгоритмами [79]. Описание этого метода слишком объемно, чтобы уместиться в рамках данной статьи. На сегодняшний день алгоритмы решета числового поля и квадратичное решето Померанца – самые быстрые из известных алгоритмов факторизации больших чисел.

Методы криптоанализа симметричных криптосистем

Методы криптоанализа блочных шифров

Блочный шифр представляет собой отображение векторных пространств над полем из двух элементов вида $F: \mathbb{Z}_2^m \times \mathbb{Z}_2^n \to \mathbb{Z}_2^m$, где ключ $k \in \mathbb{Z}_2^n$, а блоки открытого и зашифрованного текста $X_i, Y_i \in \mathbb{Z}_2^m$. Идея, лежащая в основе большинства итерационных блочных шифров, состоит в построении криптографически стойкой системы путем последовательного применения относительно простых криптографических преобразований. Принцип многоразового шифрования с помощью простых криптографических преобразований был впервые предложен Шенноном в работе [102]: он использовал с этой целью преобразования перестановки и подстановки. Первое из этих

преобразований переставляет отдельные символы преобразуемого информационного блока, а второе – заменяет каждый символ (или группу символов) из преобразуемого информационного блока другим символом из того же алфавита (соответственно группой символов того же размера и из того же алфавита). Узлы, реализующие эти преобразования, называются, соответственно, ***P-блоками*** (*P-box, permutation box*) и ***S-блоками*** (*S-box, substitution box*).

Наибольший прогресс в разработке методов раскрытия блочных шифров был достигнут в самом конце XX века и в основном связан с появлением в начале 90-х годов двух методов – метода разностного криптоанализа и метода линейного криптоанализа.

Статистический метод

Задачей статистического метода криптоанализа является разработка алгоритмов определения неизвестного ключа (или части ключа) $k \in \mathbb{Z}_2^n$. Рассмотрим базовые принципы и понятия статистического метода для блочных шифров; более развернутые описания представлены в работах [92] и [40], использованных в данной статье.

Реализации статистического метода криптоанализа для ряда блочных шифров позволяют получать оценки эффективности алгоритмов определения секретного ключа лучше, чем оценки метода полного перебора ключей. Входом алгоритма является некоторое число пар $(X_i, Y_i), i = 1, \ldots, N$ открытого и шифрованного текста, полученных в результате применения отображения F с ключом k. Такие пары будем называть *материалом* [92] и обозначим буквой M. Объем материала соответствует числу пар (X_i, Y_i): $|\mathrm{M}| = N$. Предполагается, что открытые тексты $X_i, i = 1, \ldots, N$ выбраны случайно, равновероятно и независимо из всего пространства $\mathbb{Z}_2^m$.

Важнейшей частью статистических методов анализа являются *процедуры статистической классификации* (ПСК) [40], предназначенные для поиска неизвестного параметра по доступным случайным наблюдениям. Функции распределения вероятностей для наблюдений зависят от этого параметра. Идея ПСК заключается в том, что, если эти распределения вероятностей различны, то при достаточно большом числе наблюдений можно с определенной долей уверенности определить закон распределения наблюдений, а, значит, и искомый параметр.

Доступными случайными наблюдениями в нашем случае является материал M, а неизвестным параметром - часть ключа или некоторые линейные комбинации битов ключа. Обозначим множество, в котором принимает значения неизвестный параметр, через $\Gamma, |\Gamma| = s \le 2^n$.

Каждая ПСК определяет разбиение всего пространства наблюдений на $T > 1$ непересекающихся областей: $\mathrm{M} = \mathrm{M}_1 \cup \mathrm{M}_2 \cup \ldots \cup \mathrm{M}_T; \mathrm{M}_i \cap \mathrm{M}_j = \varnothing$ при $i \ne j, \ i, j \in \{1, \ldots, T\}$. Области $\mathrm{M}_i, i \in \{1, \ldots, T\}$, называют *областями принятия решений* [92], причем для заданного наблюдения $m \in \mathrm{M}$ сложность алгоритма определения номера $i(m)$, такого, что $m \in \mathrm{M}_{i(m)}$, считается малой. Для каждой области M_i ПСК также определяет упорядоченный список объема $s' \ (1 \le s' \le s)$ элементов множества $\Gamma: \gamma_{i,1}, \gamma_{i,2}, \ldots, \gamma_{i,s'}$, при этом $\gamma_{i,j_1} \ne \gamma_{i,j_2}$ при $j_1 \ne j_2$.

Для определения неизвестного параметра из Γ выполняются следующие действия. Сначала по полученному наблюдению $m \in \mathrm{M}$ нужно определить номер области принятия решений $i(m)$. Затем последовательно перебирают параметры из $\Gamma: \gamma_{i(m),1}, \gamma_{i(m),2}, \ldots, \gamma_{i(m),s'}$ и проверяют, является ли значение j – го параметра ($j = \overline{1, s'}$) искомым или нет. Алгоритм проверки включает два этапа:

1. доопределение оставшейся части ключа;
2. проверка, правильно ли определен весь ключ.

Первый шаг отсутствует, если в качестве неизвестного параметра выступает весь ключ. В этом случае $\Gamma = \mathbb{Z}_2^n$. Как правило, для доопределения оставшейся части ключа используется полный перебор все оставшихся неизвестными битов ключа. Если параметром является часть ключа или невырожденная линейная комбинация битов ключа и $\Gamma = \mathbb{Z}_2^{n^*}, 1 \le n^* \le n$, потребуется перебрать 2^{n-n^*} вариантов.

Проверка того, правильно ли доопределен весь ключ, осуществляется следующим образом: для пар $(X_i, Y_i) \in \mathrm{M}, X_i, Y_i \in \mathbb{Z}_2^m, i \in \{1, \dots, N\}$ открытого и шифрованного текста из доступного материала M проверяют, выполнены или нет равенства: $F(X_i, k^*) = Y_i$, где $k^* \in \mathbb{Z}_2^n$ - опробуемый вариант всего ключа. Ложный ключ, как правило, отсеивается уже на первых шагах проверки. На самом деле проверку достаточно осуществить для d первых пар открытого и зашифрованного текста, где число d таково, что для любого набора из d различных открытых текстов $X_i, i = 1, \dots, d$ и для любых двух различных ключей $k_1 \ne k_2 \in \mathbb{Z}_2^n$ найдется такой номер $j \in \{1, .., d\}$, что $F(X_j, k_1) \ne F(X_j, k_2)$ (минимальное d_0, удовлетворяющее этому условию, называют *расстоянием единственности шифра* F [92]).

Алгоритмы определения ключа сравнивают по трем параметрам: N - объем используемого материала, Q_0 - средняя трудоемкость работы алгоритма и π_0 - надежность алгоритма. Q_0 и π_0 зависят от того, какие открытые тексты были случайно выбраны, и от искомого ключа. Трудоемкость соответствует математическому ожиданию числа шагов алгоритма при случайном выборе открытых текстов и случайном, равновероятном и независимом от выбора открытых текстов выборе ключа. Надежность алгоритма равна математическому ожиданию вероятности того, что процедура выдаст правильный результат в предположении, что ключ выбран в пространстве $\mathbb{Z}_2^n$ случайно, равновероятно и независимо от выбора открытых текстов. Между параметрами Q_0 и π_0 существует прямая зависимость: чем выше надежность, тем больше трудоемкость, и наоборот.

Метод разностного анализа

При написании этого раздела использовались материалы семинара А.Е. Жукова по спектральным характеристикам булевых функций.

Метод разностного анализа сочетает в себе обобщение идеи общей линейной структуры с применением вероятностно-статистических методов исследования. Этот метод относится к атакам по выбранному открытому тексту. Попытки применить разностный анализ к известному открытому тексту в большинстве случаев приводили к резкому увеличению требуемого материала. Метод был разработанный в 1990 году израильскими математиками Э. Бихамом и А. Шамиром [11]. Д. Копперсмит утверждает [16], что этот метод был известен команде разработчиков DES алгоритма еще в начале 70-х годов, но был засекречен. Идея, близкая к методу дифференциального анализа, была опубликована до работы Э. Бихама и А. Шамира в 1990 году С.Мерфи [47].

Пусть некоторый блочный шифратор с длиной блока m задается отображением $F: \Xi \times (K_1 \times \dots \times K_R) \to Y$, где $F = F_R \circ F_{R-1} \circ \dots \circ F_2 \circ F_1$. При этом $k_i \in K_i$ получаются по некоторой схеме из общего ключа k или выбирается независимо и равновероятно для

каждого цикла. Пространство открытых текстов Ξ снабжено групповой операцией $\otimes$, и для каждого $X \in \Xi$ в Ξ существует элемент $X^{-1} \in \Xi$, обратный к X относительно операции $\otimes$. Выходной информационный блок $(i-1)$-го цикла является входным блоком i-го цикла, т.е. $X(i) = Y(i-1)$, для $i = \overline{2,R}$; открытый текст $X = X(1)$, зашифрованный текст $Y = Y(R)$.

Пусть одноцикловое преобразование F_j - криптографически слабое. Сделанное предположение вполне допустимо (идея Шеннона о суперпозиции простых шифров для получения сложного шифра [102]). Отметим, что под *слабым криптографическим преобразованием* $\mathrm{F}:\Xi \times K \to \Xi$ мы будем понимать такое криптографическое преобразование $F(X,k) = Y$, для которого по известным величинам $Y = F(X,k)$, $Y^* = F\left(X^*,k\right)$ и $\Delta X = X \otimes \left(X^*\right)^{-1}$ можно, не зная X и X^*, определить множество $K', |K'| << |K|$, такое, что $k \in K'$.

Пусть X и X^* - открытые тексты. Два открытых текста определяют последовательность разностей $\Delta X(0), \Delta X(1), ..., \Delta X(R)$, где $\Delta X(0) = \Delta X = X \otimes \left(X^*\right)^{-1}$; $\Delta X(i) = X(i+1) \otimes \left(X^*(i+1)\right)^{-1}$, $(i = \overline{1,R-1})$; $\Delta X(R) = Y \otimes \left(Y^*\right)^{-1}$. Тогда для любого $1 \le i \le R$ и любой пары (α, β) можно определить вероятность $P_{\alpha\beta}^{(i)} = \mathrm{P}\left(\Delta X(i) = \beta \mid \Delta X(0) = \alpha\right)$ при условии, что вход X и все одноцикловые ключи k_i выбраны случайно, независимо и равновероятно. Пара (α, β) возможных значений вектора $\left(\Delta X(0), \Delta X(i)\right)$ называется *дифференциалом i-го цикла* [84].

Выберем пару (α, β), для которой величина $P_{\alpha\beta}^{(R-1)}$ принимает максимальное значение, и пару $\left(X, X^*\right)$, такую, что $\Delta X = \alpha$. Для одноциклового шифра F_R, полагая $\Delta X(R-1) = \beta$ и зная истинные значения $Y = F(X,k)$ и $Y^* = F\left(X^*,k\right)$, определим множество вероятных одноцикловых ключей *K′*. Если теперь эту процедуру провести для различных пар $\left(X, X^*\right)$, удовлетворяющих условию $\Delta X = \alpha$, то ключи, наиболее часто встречающиеся в множествах *K′*, можно считать кандидатами в истинный ключ *R*-го цикла шифрования. Ключ всей системы находим с помощью перебора оставшихся неизвестными разрядов ключа системы или с использованием особенностей процедуры выработки цикловых ключей из ключа всей системы.

Для того, чтобы описанная процедура приводила к корректным результатам, необходимо, чтобы для данной системы шифрования выполнялась

Гипотеза о статистической эквивалентности:

$$\mathrm{P}_{\alpha\beta}^{(\mathrm{R}\text{-}1)}\left(\Delta X(R-1) = \beta \middle| \Delta X(0) = \alpha\right) \approx \mathrm{Prob}\left(\Delta X(R-1) = \beta \middle| \Delta X(0) = \alpha, k_1 = \omega_1, ..., k_{R-1} = \omega_{R-1}\right)$$

для почти всех значений частей ключа, используемых в циклах шифрования $(\omega_1, ..., \omega_{R-1})$, где $\mathrm{Prob}(\theta)$ обозначает вероятность события θ.

Возможность эффективного применения метода разностного анализа существенно зависит от выбора групповой операции, относительно которой определяются разности Δ. Чаще всего в качестве таковой выбирается операция сложения булевых векторов. Однако

в отдельных случаях неудачный выбор операции может приводить к нарушению гипотезы о статистической эквивалентности, в результате чего становится невозможным вычисление вероятностей.

Эффективность метода разностного анализа существенно зависит от выбора характеристики, с помощью которой он проводится. Свойства подстановки P на разностный анализ систем типа DES не влияют. В то же время, даже изменение порядковой нумерации S-блоков (без изменения их строения) может сильно ослабить DES. При неудачном подборе этой нумерации DES-16 раскрывается за 2^{46} опробований. Стойкость системы DES к методу разностного анализа может также уменьшаться при замене операции векторного сложения на другие арифметические операции, при замене S-блоков на случайно выбранные и даже при внесении минимальных изменений в один S-блок.

Почти сразу после появления первых работ по разностному криптоанализу начались поиски условий, при которых та или иная криптографическая система остается устойчивой по отношению к этому методу. Так как разностный анализ основан на использовании неравновероятности в распределении значений разности двух шифртекстов полученных из пары открытых текстов, имеющих некоторую фиксированную разность, то очевидно, что если все возможные значения разностей двух шифртекстов будут появляться с близкими (в идеале – с равными) вероятностями, то метод разностного анализа не сможет работать.

Как показано в [92], метод разностного анализа развивался в следующих направлениях:

1. Вскоре после изобретения метода разностного анализа было предложено использовать разностные характеристики для случая, когда операцию покоординатного суммирования векторов из $\mathbb{Z}_2^{32}$ (для DES-алгоритма) по модулю 2 заменяют на операцию суммирования этих векторов по модулю 2^{32}, рассматривая их как 2-адическую запись целых чисел [9].
2. В 1993 году израильские математики Бен-Аройа и Бихам [8] предложили искать разностные характеристики, считая, что ключ принимает не все возможные значения, а лишь значения из некоторого подмножества. Этот метод получил название «*метод условных дифференциалов*».
3. В 1994 году датский математик Ларс Кнудсен [37] предложил строить по аналогии с обычным разностным методом криптоанализа метод усеченных дифференциалов. Идея Кнудсена заключается в том, чтобы "следить" в разностной характеристике лишь за частью бит векторов, участвующих в соотношении.
4. В 1994 году швейцарский криптограф Лаи [37, 39] показал, что для построения метода криптографического анализа вместо пар можно использовать разностные характеристики высших порядков: $\bigoplus_{a \in L} F(x \oplus a) = d$, где L - пространство в $\mathbb{Z}_2^n$, $\dim L \geq 2$.
5. В 1998 году Бихам, Бирюков и Шамир [10] заметили, что для построения метода криптографического анализа можно использовать дифференциалы, имеющие не повышенную вероятность появления, а пониженную, еще лучше - нулевую (*невозможные дифференциалы*). Именно этим методом была обнаружена слабость в криптографическом алгоритме Skipjack [65] - первом и пока единственным алгоритме, авторство которого официально признано Агентством Национальной Безопасности США.
6. Для шифров итерационного типа редко удается построить разностную характеристику, имеющую гарантированно большую вероятность. В 1999 году

Вагнер [69] предложил использовать не пары, а четверки открытых текстов с заданным набором разностей. Для построения таких четверок Вагнер разработал специальный метод, названный «*методом прямоугольника*». Развитие этого метода получило название «*метода бумеранга*».

Метод линейного анализа

Подобно разностному анализу, линейный криптоанализ является комбинированным методом, сочетающим в себе поиск линейных статаналогов для уравнений шифрования, статистический анализ имеющихся открытых и шифрованных текстов, использующий также методы согласования и перебора. Этот метод исследует статистические линейные соотношения между отдельными координатами векторов открытого текста, соответствующего шифртекста и ключа и использует эти соотношения для определения статистическими методами отдельных координат ключевого вектора.

На сегодняшний день метод линейного криптоанализа позволил получить наиболее сильные результаты по раскрытию ряда итерационных систем блочного шифрования, в том числе и системы DES. В отличие от метода разностного анализа, метод линейного криптоанализа в неявном виде появился еще в работе [47], где он успешно применялся при анализе системы блочного шифрования FEAL [63]. В работе же [43] этот подход был впервые четко формализован, а затем успешно применен к анализу системы DES.

Пусть имеется блочный шифр $F: \mathbb{Z}_2^n \times \mathbb{Z}_2^k \to \mathbb{Z}_2^n$, отображающий при фиксированном ключе $k \in \mathbb{Z}_2^k$ вектор открытого текста $p \in \mathbb{Z}_2^n$ в вектор шифртекста $c \in \mathbb{Z}_2^n$. Предполагается, что открытый текст выбирается случайно и равновероятно, а подключи в каждом раунде независимы друг от друга. Сложность атаки связана с количеством необходимых известных открытых текстов, т.к. для любой пары (открытый текст, шифртекст) требуется небольшое количество вычислений для реализации алгоритма.

Эта атака может быть проведена, если имеется только шифртекст. Мацуи были получены следующие результаты:

- если известно, что открытый текст на английском языке, представленный в ASCII кодах, то при 8-раундовый DES вскрывается при 2^{29} известных шифртекстах;
- если открытый текст является случайным ASCII кодом, то при 8-раундовый DES вскрывается при 2^{37} известных шифртекстах.

Метод линейного криптоанализа является частным случаем общего статистического метода и основан на использовании выражений вида

$$\langle Y(t_1), a \rangle \oplus \langle Y(t_2), b \rangle = \langle k, c \rangle \qquad (3),$$

где $a, b \in \mathbb{Z}_2^m, c \in \mathbb{Z}_2^n, 0 \le t_1 < t_2 \le R$ (при этом пару векторов (a, b) называют линейной характеристикой), $\langle g, h \rangle = \sum_j g_j \cdot h_j$ - скалярное произведение векторов g и h одинаковой размерности. В методе линейного криптоанализа используют только те выражения типа (3), которые выполняются с вероятностью, отличной от $\frac{1}{2}$. Вероятность, с которой выполняется равенство (3), удобно записывать в виде

$$P\left(\langle Y(t_1), a \rangle \oplus \langle Y(t_2), b \rangle = \langle k, c \rangle\right) = \frac{1}{2} + \varepsilon \qquad (4)$$

где $-\frac{1}{2} \le \varepsilon \le \frac{1}{2}$. Число ε называют преобладанием (deviation, bias).

Рассмотрим простейший (но не самый эффективный) вариант метода линейного криптоанализа - алгоритм определения одного бита ключа, называемый также «алгоритм 1 Мацуи» [92]. Этот алгоритм использует выражение типа (3) при $t_1 = 0$ и $t_2 = R$, т.е. подставляются непосредственно открытый и зашифрованный тексты:

$$\langle X_i, a\rangle \oplus \langle Y_i, b\rangle = \langle k, c\rangle, \qquad i = 1, \ldots, N$$

где (X_i, Y_i) - пары открытого и зашифрованного текста, известные криптоаналитику. Метод линейного криптоанализа в данном варианте определяет линейную комбинацию $\langle K, c\rangle$ битов ключа. Подсчитывается мощность множества:

$$N_0 = \left| \left\{ i \mid \langle X_i, a\rangle \oplus \langle Y_i, b\rangle = 0, i = \overline{1, N} \right\} \right|$$

Процедура статистической классификации разбивает все пространство наблюдений M на две области M_0 и M_1 ($\mathrm{M} = \mathrm{M}_0 \cup \mathrm{M}_1$): к области M_0 относят все те наблюдения, для которых $N_0 \geq N/2$, а к области M_1 - те наблюдения, для которых $N_0 < N/2$. Алгоритм использует принцип максимума правдоподобия (этот алгоритм называют «алгоритмом 1 Мацуи»): если $N_0 \geq N/2$ и в равенстве (4) $\varepsilon > 0$ или $N_0 < N/2$ и в равенстве (4) $\varepsilon < 0$, то алгоритм выдает значение $\langle k, c\rangle = 0$, в противном случае - $\langle k, c\rangle = 1$. Остальные биты ключа определяют методом перебора.

Улучшенный алгоритм 2 Мацуи [92] представляет собой естественное обобщение алгоритма 1 и заключается в использовании выражений вида (3) при $t_2 > R$ и/или $t_1 > 0$.

Таким образом, метод линейного криптоанализа сводится к построению таких векторов $a, b \in \mathbb{Z}_2^m, c \in \mathbb{Z}_2^n$, чтобы модуль ε в выражении (4) был как можно больше. При этом надежность и трудоемкость метода определяются этой величиной $|\varepsilon|$, т.е. для построенных векторов $a, b \in \mathbb{Z}_2^m, c \in \mathbb{Z}_2^n$ надо уметь оценивать $|\varepsilon|$ в выражении (4).

Эту задачу метод линейного криптоанализа решает, используя итерационную структуру блоковых шифров. Первым шагом при этом является анализ в каждом раунде алгоритма шифрования нелинейных элементов. Для каждого нелинейного элемента вычисляют для всех $a' \in \mathbb{Z}_2^k, b' \in \mathbb{Z}_2^l$ вероятности $P\left(\langle X', a'\rangle = \langle S(X'), b'\rangle\right) = \frac{1}{2} + \varepsilon'$ при случайном равновероятном выборе X' из $\mathbb{Z}_2^k$. Для получения оценки на преобладание для всего алгоритма рассматривают последовательности «согласованных» линейных соотношений для соседних раундов (т.е. соотношения, когда выходная линейная комбинация предыдущего раунда совпадает с входной линейной комбинацией последующего раунда).

За годы развития линейного криптоанализа был предложен ряд его обобщений. В работе [32] предложена идея использовать несколько линейных соотношений (этот метод называют методом кратного приближения), но конкретной конструкции, как это реализовать, не предложено, за исключением случая, когда во все линейные соотношения входит одна и та же линейная комбинация битов ключа. И лишь недавно появилась статья [12], в которой предложен статистический расчет метода кратных приближений. В работе [38] предложено использовать нелинейные приближения вместо линейных, однако не указано, как эффективно искать такие приближения и как рассчитывать их преобладания. В работе [35] предложено использовать линейный метод криптоанализа в сценарии атаки с выбором открытого текста. Также заслуживает внимания метод решеточного криптоанализа, предложенного Ростовцевым в статье [97]. Метод должен быть

эффективен для тех шифров, в которых не каждый разряд ключа сцеплен с каждым разрядом шифртекста, например, для шифров с псевдослучайным выбором слов ключа.

Методы криптоанализа поточных шифров

Поточные шифры преобразуют открытый текст в шифртекст шифрованный побитово. Простейшая реализация поточного шифра представлена на рис. 1. Генератор гаммы выдает поток битов: $k_1, k_2, ..., k_i$, называемый *ключевым потоком*, или *бегущим ключом*, или *гаммой*. Гамма шифра и поток битов открытого текста $p_1, p_2, ..., p_i$ подвергаются операции XOR, в результате чего создается поток битов шифртекста $c_1, c_2, ..., c_i$, где $c_j = p_j \oplus k_j$ (этот режим шифрования называется *гаммированием*). При расшифровании для восстановления битов открытого текста над битами шифртекста и той же самой гаммой тоже выполняется операция XOR: $p_j = c_j \oplus k_j$. Надежность схемы всецело зависит от генератора гаммы. Если он создает бесконечную строку нулей, шифртекст, очевидно, будет совпадать с открытым текстом, и вся операция бессмысленна.

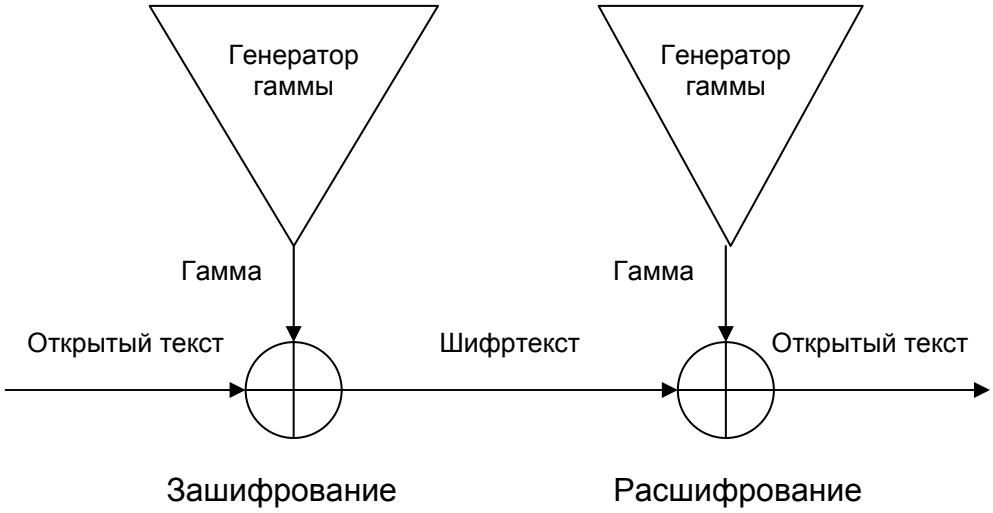

Рис. 1

Значительная часть предлагаемых поточных схем состоит из унифицированных узлов и блоков, напоминая тем самым выставку поделок из детского конструктора [96]. Основными деталями этого "криптографического конструктора" являются:

- регистры сдвига (на битах или двоичных векторах определенной размерности) с обратной связью (обычно линейной),
- дискретные (главным образом, булевы) функции усложнения,
- запоминающие устройства,
- узлы, реализующие неравномерное движение.

Регистры сдвига обеспечивают большой период и хорошие статистические свойства последовательностей, в то время как остальные детали "конструктора" используются для внесения элементов нелинейности в законы функционирования поточной схемы.

В *синхронных* поточных шифрах гаммирующая последовательность формируется независимо от потока открытого текста при зашифровании и шифртекста при расшифровании. Главное свойство синхронного поточного шифра – отсутствие эффекта размножения ошибок. Это свойство ограничивает возможность обнаружения ошибки при

расшифровании. Кроме того, противник имеет возможность производить управляемые изменения шифртекста, совершенно точно зная, какие изменения в результате произойдут в соответствующем открытом тексте.

Самосинхронизирующиеся поточные шифры, благодаря свойству восстанавливать информацию после потери синхронизации, являются наиболее распространенным видом шифрования в дипломатических, военных и промышленных системах связи. В таких шифрах каждый бит гаммы представляет собой функцию фиксированного числа предыдущих битов шифртекста [58]. Наиболее распространенный режим функционирования самосинхронизирующихся поточных шифров – режим обратной связи по шифртексту.

Поточные шифры почти всегда работают быстрее и обычно требуют для своей реализации гораздо меньше программного кода, чем блочные шифры. Наиболее известный поточный шифр был разработан Р. Ривестом; это шифр RC4, который характеризуется переменным размером ключа и байт-ориентированными операциями. На один байт требуется от 8 до 16 действий, программная реализация шифра выполняется очень быстро. Независимые аналитики исследовали шифр, и он считается защищенным. RC4 [71] используется для шифрования файлов в таких изделиях, как RSA SecurPC. Он также применяется для защиты коммуникаций, например, для шифрования потока данных в Интернет-соединениях, использующих протокол SSL.

Хотя подавляющее большинство существующих шифров с секретным ключом с определенностью могут быть отнесены или к поточным или к блочным шифрам, теоретически граница между этими классами остается довольно размытой. Так, например, допускается использование алгоритмов блочного шифрования в режиме поточного шифрования (например, режимы CBF и OFB для алгоритма DES или режим гаммирования для алгоритма ГОСТ 28147-89 [83]).

Структура поточного ключа может иметь некоторые уязвимые места, которые позволят нападающему получить некоторую дополнительную информацию о ключе. Наиболее очевидно, если *период поточного ключа* (т.е. количество бит, после которых поточный ключ начинает повторяться) слишком мал, то нападающий может применить обнаруженные части поточного ключа для дешифрации других частей закрытого текста.

Другая опасность состоит в том, что одна или несколько внутренних выходных последовательностей – часто выходы отдельных регистры сдвига с линейной обратной связью – могут иметь корреляцию с объединенной гаммой и атакованы при помощи линейной алгебры. Такие атаки называют «корреляционными атаками», или атаками «разделяй и властвуй» [58]. Основная идея состоит в выявлении определенной корреляции между выходом генератора и одной из его составных частей. Используя эту информацию, можно собирать данные о других промежуточных выходах до тех пор, пока генератор не будет взломан. К поточным шифрам также применим линейный и дифференциальный криптоанализ.

В обзоре [96] отмечаются следующие особенности работ, посвященных криптографическим исследованиям поточных шифров.

1. Если подавляющее число работ, посвященных блочным шифрам, ориентировано на анализ и синтез DES-подобных алгоритмов, то для поточных шифров нет такого "центра притяжения". Синтезные решения и методы "взлома" поточных шифров отличаются значительным разнообразием.

2. Если для блочных шифров нет "канонической" теории их синтеза и анализа, то для поточных шифров теория построения методов "взлома" сформировалась; также установлен набор требований, которым должны удовлетворять "хорошие" схемы. В идейном плане методы взлома поточных шифров сводятся к одному из следующих двух подходов:

- использование статистических связей (корреляционные атаки);
- линеаризация (сведение задачи поиска ключа к решению системы линейных уравнений).

Соответственно, от стойких поточных схем требуется:

- большие периоды выходных последовательностей;
- хорошие статистические свойства выходных последовательностей ("постулаты Соломона Голомба" [26]);
- нелинейность (точнее, высокая линейная сложность) выходных последовательностей.

3. Исследования поточных схем протекают более динамично, чем исследования блочных шифров. В то время как исследования DES-алгоритма до недавнего времени шли без видимых продвижений, область поточного шифрования испытала множество "взлетов и падений", некоторые схемы, вначале казавшиеся стойкими, "рухнули" при последующем исследовании.
4. Вопросам исследования поточных шифров уделяется больше внимания в европейских криптографических центрах, в то время как в США больший "крен" делается в сторону блочных шифров.

Криптографические исследования поточных шифров явились источником ряда задач для фундаментальных направлений дискретной математики:

1. Задача анализа свойств регистров сдвига с линейной обратной связью стимулировали исследования линейных рекуррентных последовательностей над полями и кольцами.
2. Задача поиска статистических связей между входом и выходом узла, реализующего дискретную (булеву) функцию, и построение функций с заданными свойствами.

Криптоанализ по побочным каналам

В последнее время одним из самых актуальных направлений криптоанализа стало осуществление атак, использующих особенности реализации и рабочей среды. *Атаки по сторонним, или побочным, каналам* — это вид криптографических атак, использующих информацию, полученную по сторонним или побочным каналам. Под *информацией из побочных каналов* понимается информация, которая может быть получена с устройства шифрования и не является при этом ни открытым текстом, ни шифртекстом. При подготовке этого раздела использовались материалы доклада А.Е.Жукова на конференции РусКрипто2006 [85].

Почти все осуществленные на практике удачные атаки на криптосистемы используют слабости в реализации и размещении механизмов криптоалгоритма. Такие атаки основаны на корреляции между значениями физических параметров, измеряемых в разные моменты во время вычислений (потребление энергии, время вычислений, электромагнитное излучение и т.п.), и внутренним состоянием вычислительного устройства, имеющим отношение к секретному ключу. На практике атаки по побочным каналам на много порядков более эффективны, чем традиционные атаки, основанные только на математическом анализе. При этом атаки по побочным каналам используют особенности реализации (поэтому их иногда называют также называют атаками на реализацию - implementation attacks) для извлечения секретных параметров,

задействованных в вычислениях. Такой подход менее обобщённый, поскольку привязан к конкретной реализации, но зачастую более мощный, чем классический криптоанализ.

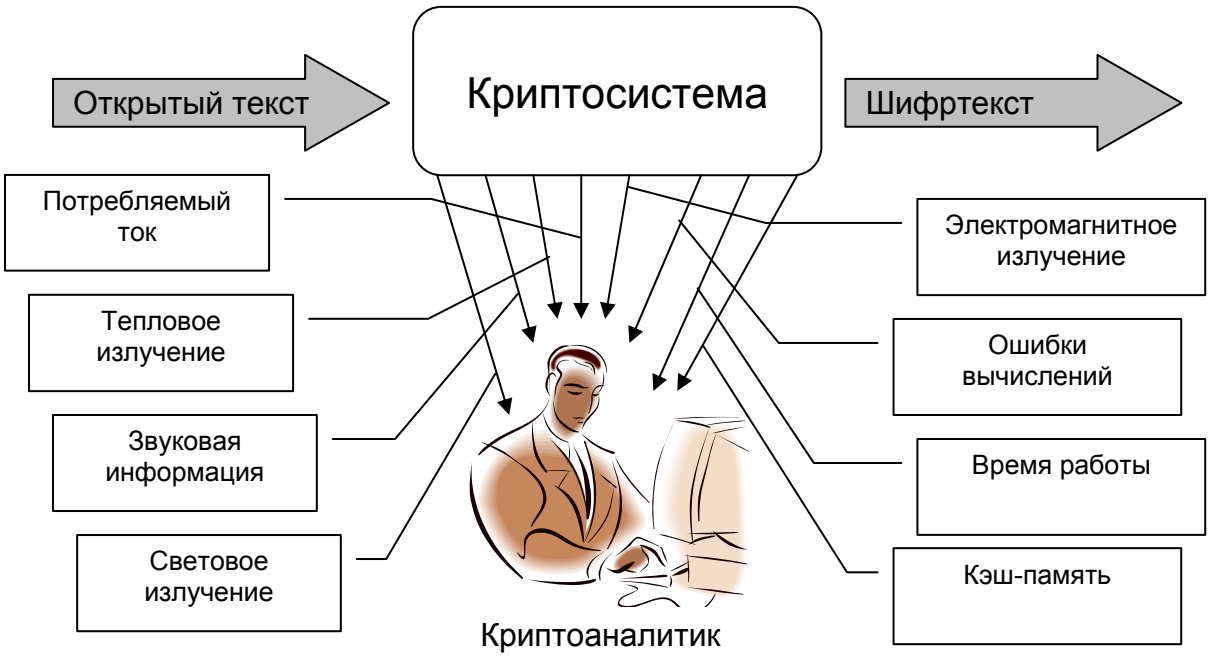

Рис. 2

В последние годы резко возросло количество криптографических атак, использующих особенности реализации и рабочей среды. Например, противник может отслеживать энергию, потребляемую смарт-картой, когда она выполняет операции с закрытым ключом, такие, как расшифрование или генерация подписи. Противник может также замерять время, затрачиваемое на выполнение криптографической операции, или анализировать поведение криптографического устройства при возникновении определённых ошибок. Побочную информацию на практике собрать порой несложно, поэтому нужно обязательно учитывать такую угрозу при оценке защищённости системы.

Атаки по побочным каналам классифицируются по следующим трём типам:

- по контролю над вычислительным процессом: *пассивные* и *активные*.
- по способу доступа к модулю: *агрессивные* (invasive), *полуагрессивные* (semi-invasive) и *неагрессивные* (non-invasive).
- по методу, применяемому в процессе анализа: простые – simple side channel attack (*SSCA*) и разностные – differential side channel attack (*DSCA*).

На сегодняшний день выделено более десяти побочных каналов. Атаки различаются по виду используемого побочного канала (рис. 2): атаки по времени исполнения (Timing Attacks), атаки по энергопотреблению (Power Analysis Attacks), атаки по ошибкам вычислений (Fault Attacks), атаки по электромагнитному излучению (ElectroMagnetic Analysis), атаки по ошибкам в канале связи (Error Message Attacks). Существуют и более изощренные виды атак: атаки по кэш-памяти (Cache-based Attacks), акустические атаки (Acoustic Attacks), атаки по световому излучению (Visible Light Attacks).

Атака по времени

Атака по времени – способ получения какой-либо скрытой информации путем точного измерения времени, которое требуется пользователю для выполнения криптографических операций. Это – самая первая из атак по побочным каналам, появившаяся в гражданской криптографии. Зачастую время обработки данных в

криптосистемах немного изменяется в зависимости от входных значений (например, открытого текста или шифртекста). Это является следствием оптимизации производительности и широкого круга иных причин. Атака по времени основана на измерении времени, необходимого модулю шифрования для выполнения операции шифрования. Эта информация может вести к раскрытию информации о секретном ключе. Например, тщательно измеряя время, требуемое для выполнения операций с секретным ключом, атакующий может найти точное значение экспоненты в алгоритме Diffie-Hellman [19].

Атаки по времени наделали много шума в прессе в 1995 году: закрытые ключи RSA могут быть восстановлены измерением относительных интервалов времени, затраченных на произведение криптографических операций. Эти атаки были успешно применены к карточкам с микропроцессорами и другим средствам надёжной идентификации, а также к серверам электронной коммерции в Сети.

Атаки по мощности

Атака по анализу мощности пригодна в основном для аппаратной реализации криптографических средств и успешно применяется при взломе смарт-карт и других систем, в которых хранится секретный ключ.

Чтобы измерить потребляемую схемой мощность, необходимо последовательно с цепью питания или заземления подключить резистор малого сопротивления (например, 50 Ом). Падение напряжения, деленное на сопротивление, даст силу тока. Современные лаборатории располагают оборудованием, способным производить цифровые измерения напряжения на исключительно высоких частотах (более 1 ГГц) и с превосходной точностью (ошибка менее 1%).

Атака по мощности может быть разделена на простую (Simple Power Analysis, SPA) и разностную (Differential Power Analysis, DPA). Целью SPA является получение информации о конкретных выполняемых инструкциях в системе и о конкретных обрабатываемых данных. В общем случае SPA может дать как сведения о работе устройства, так и информацию о ключе. Для осуществления этой атаки криптоаналитик должен располагать точными данными о реализации устройства. Этот метод использует непосредственные данные измерений, собранные во время выполнения криптографических операций. Согласно [13], простая атака по мощности для смарт-карт обычно занимает несколько секунд, в то время как разностная атака по мощности может занять несколько часов.

В отличие от простых атак, разностные атаки, основанные на анализе потребляемой мощности, подразумевают не только визуальное представление потребляемой мощности, но также статистический анализ и статистические методы исправления ошибок для получения информации о ключах. Более того, DPA зачастую не нуждается в данных о конкретной реализации и в качестве альтернативы использует статистические методы анализа. Разностный анализ мощности – одно из самых мощных средств для проведения атак, использующих побочные каналы, причем эта атака требует очень маленьких затрат.

Атаки по ошибкам вычислений

Ошибки аппаратного обеспечения, появляющиеся во время работы соответствующего криптографического модуля, или ошибочные выходные данные могут стать важными побочными каналами и иногда существенно увеличивают уязвимость шифра к криптоанализу. Криптоанализ на основе формирования случайных аппаратных ошибок - это вид нападения на шифры в случае, когда предполагаемый нарушитель имеет возможность оказать на шифратор внешнее физическое воздействие и вызвать одиночные ошибки в процессе шифрования одного блока данных. Атаки по ошибкам на

криптографические алгоритмы изучаются с 1996 года, и с того времени почти все криптографические алгоритмы были подвержены атакам такого вида.

Осуществимость атаки по ошибкам (или, по крайней мере, ее эффективность) зависит от возможностей злоумышленника вызывать ошибки в системе специально или пользоваться сбоями естественного происхождения. Ошибки наиболее часто происходят из-за скачков напряжения, сбоев часов или из-за излучений различных типов. Рассмотрение вопроса стойкости к этому методу особенно актуально для шифраторов, применяемых в интеллектуальных электронных карточках. В основном ошибки классифицируются по следующим аспектам:

- Точность, которую нарушитель может достичь при выборе времени и места, где появляется ошибка во время работы криптографического модуля.
- Длина данных, на которые влияет ошибка; например, только один бит.
- Постоянство ошибки; является ли ошибка кратковременной или постоянной.
- Тип ошибки; такие как изменение одного бита; изменение одного бита, но только в одном направлении (например, с 1 на 0); изменение бита на случайное значение и др.

В общем, успешная атака по ошибкам на криптографические модули или устройства требует двух шагов: шаг создания ошибки и шаг использования ошибки. Ошибки могут быть вызваны в смарт-картах путем внешнего влияния на нее и помещения ее в неправильные условия. Некоторые из них – аномальное и внезапное понижение или повышение напряжения, частоты, температуры, излучения, освещения и др.

Разностный анализ по ошибкам состоит в изучении результата работы алгоритма шифрования в нормальных и ненормальных условиях при одном и том же входе (открытом тексте). Ненормальные условия обычно получаются созданием ошибки в процессе (кратковременная ошибка) или перед процессом (постоянная ошибка) работы. Разностный анализ по ошибкам широко изучены с теоретической точки зрения и кажутся применимыми почти ко всем симметричным криптосистемам.

Для ГОСТ 28147-89 была показана возможность раскрытия ключа и таблиц подстановок с помощью криптоанализа на основе формирования случайных аппаратных ошибок [93]. DES, RC5 и другие шифры также являются уязвимыми по отношению к этому виду криптоанализа, поэтому при их использовании необходимо обеспечить защиту аппаратуры от навязывания сбоев.

Атаки по электромагнитному излучению

Выполнение вычислительных операций на компьютере сопряжено с выделением электромагнитного излучения. Измеряя и анализируя это излучение, нарушитель может получить значительную информацию о выполняющихся вычислениях и используемых данных. Атаки по электромагнитному анализу могут быть также разделены на две большие категории: простые (SEMA) и дифференциальные (DEMA).

Стойкость российского и американского стандартов на симметричную криптографию.

DES, Triple DES и AES. В 1973-74 гг. Национальное Бюро Стандартов США (NBS) опубликовало документы, содержащие требования к криптографическому алгоритму, который мог бы быть принят в качестве стандарта шифрования данных в государственных и частных учреждениях. В 1976 г. в качестве такового стандарта был утвержден алгоритм, разработанный фирмой IBM. В 1977 г. этот стандарт был официально опубликован и

вступил в силу как федеральный стандарт шифрования данных – Data Encryption Standard или сокращенно DES [4].

В самом схематичном виде DES представляет собой 16-цикловой итерационный блочный шифр. DES работает с блоками данных разрядностью 64 бита с использованием 56-разрядного ключа. Применяемые преобразования – поразрядное сложение по модулю два, подстановки и перестановки. Алгоритм выработки 48-битовых цикловых ключей из 56-битового ключа системы и ряд преобразований служат для обеспечения необходимого перемешивания и рассеивания перерабатываемой информации, однако при анализе DES чаще всего играют не самую существенную роль.

В 1999 г. на конференции, организованной RSA, компания Electronic Frontier Foundation взломала ключ DES менее чем за 24 часа. Одной из замен DES, получившей широкое распространение, стал алгоритм Triple DES. В этом случае алгоритм DES выполняется трижды, при этом используются 3 ключа, каждый из которых состоит из 56 битов (что, по сути, соответствует использованию 168-битового ключа). Тем не менее, криптоаналитики обнаружили способ, позволяющий сделать атаку прямого перебора эквивалентной атаке на 108-битовый ключ. Второй проблемой является значительное снижение скорости зашифрования и расшифрования данных.

В ответ на проблемы с длиной ключа и производительностью, проявившиеся в Triple DES, многие криптографы и компании разработали новые блочные шифры. Наиболее популярными предложениями стали алгоритмы RC2 и RC5 корпорации RSA Data Security, IDEA компании Ascom, Cast компании Entrust, Safer компании Cylink и Blowfish компании Counterpane Systems. Коммерческие альтернативы DES получили определенное распространение, но ни одна из них не стала стандартом.

В 2001 г. на смену DES и Triple DES пришел стандарт AES (Advanced Encryption Standard), действующий и по сей день. Шифр AES основан на алгоритме Rijndael [52], разработанном бельгийцами Д. Дейменом и В. Райменом. Он быстрый, простой, защищенный, универсальный и хорошо подходит для реализации на смарт-картах. Rijndael – это итерационный блочный шифр, имеющий архитектуру «Квадрат». Шифр имеет переменную длин у блоков и различные длины ключей. Длина ключа и длина блока могут быть равны независимо друг от друга 128, 192 или 256 битам. В стандарте AES определена длина блока, равная 128 битам.

ГОСТ 28147-89 [83]. Отечественный стандарт шифрования носит официальное название «Алгоритм криптографического преобразования ГОСТ 28147-89». Как явствует из его номера, стандарт был принят в СССР в 1989 г. Если охарактеризовать алгоритм ГОСТ в самом общем виде, то он является блочным шифром, построенным по схеме Фейстеля с 32 циклами шифрования. Длина информационного блока – 64 бита, длина ключа – 256 бит.

Основные отличия алгоритма ГОСТ от алгоритма DES – в строении функции, которая осуществляет отображение $\mathbb{Z}_2^{32} \times \mathbb{Z}_2^{48} \to \mathbb{Z}_2^{32}$, и алгоритме выработки цикловых ключей. И в том и в другом случае преобразования, используемые в алгоритме ГОСТ, проще для программной реализации.

В статье [81] рассматривается устойчивость алгоритмов ГОСТ и AES к известным видам криптоанализа, в особенности - к линейному и разностному методу.

По оценкам разработчиков шифра Rijndael, уже на четырех раундах шифрования этот алгоритм приобретает достаточную устойчивость к указанным видам криптоанализа. Теоретической границей, за которой линейный и дифференциальный виды криптоанализа теряют смысл, является рубеж в 6-8 раундов в зависимости от размера блока. Согласно спецификации, в шифре предусмотрено 10-14 раундов. Следовательно, шифр Rijndael устойчив к указанным видам криптоанализа с определенным запасом.

Дать оценку устойчивости алгоритма ГОСТ 28147-89 к конкретным видам криптоанализа невозможно без спецификации узлов замен, так как качество этого шифра

существенным образом зависит от качества использованных узлов. Однако исследования близких по архитектуре шифров с заданными таблицами подстановок (DES) показали, что криптоанализ шифра с 16 раундами в принципе осуществим, однако требует очень большого числа исходных данных, а при 20-24 раундах становится теоретически бесполезным. ГОСТ предусматривает 32 раунда шифрования, и этого количества хватает с запасом, чтобы успешно противостоять указанным видам криптоанализа.

Исследования [81] показывают, что российский стандарт не уступает по стойкости американскому AES. Делается вывод, что оба сравниваемых шифра обладают достаточной стойкостью к известным видам криптоанализа. Молдовян, напротив, утверждает [93], что ГОСТ является устаревшим алгоритмом шифрования из-за его подверженности атакам на основе аппаратных ошибок. К стандарту, который придет на смену ГОСТ 28147-89, автор предъявляет следующие требования:

- высокое быстродействие при аппаратной и программной реализации,
- стойкость ко всем известным видам криптоанализа, включая атаку на основе формирования аппаратных ошибок,
- невысокая стоимость аппаратной реализации,
- полная открытость алгоритма.

Использование новых технологий в криптоанализе

Этот раздел посвящен самым необычным и достаточно спорным подходам к исследованию криптографических систем. На данный момент эти методы не привели к сколько-нибудь серьезным прорывам во взломе шифров и представляют в большей степени академический интерес, чем практический. Тем не менее, эти методы заслуживают внимания хотя бы из-за своей оригинальности; кроме того, не исключено, что со временем их значение в криптологии возрастет.

Нейронные сети

Криптосистему можно рассматривать как «черный ящик», т.е. устройство или программу, о внутренней структуре которой ничего не известно, но, подавая сигналы команды или данные на вход, можно получить реакцию на выходе. Задача криптоанализа – идентификация этой системы, т.е. определение ее структуры на основе сигналов, поступающих на ее вход и получаемых на выходе. Одним из инструментов решения этой задачи могут являться нейронные сети, теория которых изложена в [82].

Искусственная нейронная сеть— это математическая модель, а также устройства параллельных вычислений, представляющие собой систему из соединенных и взаимодействующих между собой простых процессоров (искусственных нейронов). Такие процессоры обычно исключительно просты, особенно в сравнении с процессорами, используемыми в персональных компьютерах. Каждый процессор подобной сети имеет дело только с сигналами, которые он периодически получает, и сигналами, которые он периодически посылает другим процессорам. Тем не менее, будучи соединёнными в достаточно большую сеть с управляемым взаимодействием, такие локально простые процессоры вместе способны выполнять довольно сложные задачи. Понятие возникло при изучении процессов, протекающих в мозге при мышлении, и при попытке смоделировать эти процессы. Полученные модели называются искусственными нейронными сетями (ИНС). Схема простой нейросети изображена на рис. 3. Черным обозначены входные элементы, белым — выходной элемент.

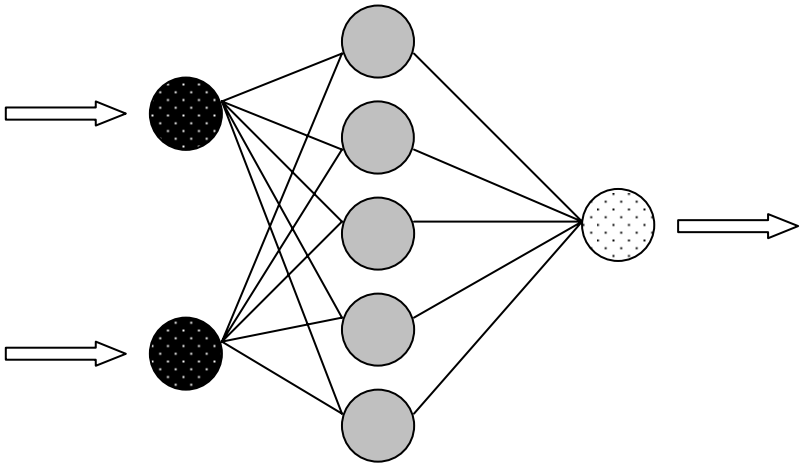

Рис. 3

Брюс Шнайер в своей книге «Прикладная криптография» [58] склоняется к пессимизму в отношении к применимости нейронных сетей в криптоанализе: «Процесс взлома не оставляет места обучению: вы либо раскрываете ключ, либо нет. (По крайней мере, это верно при вскрытии любого надежного алгоритма). Нейронные сети хорошо работают в структурированных средах, допускающих обучение, но не в высокоэнтропийном, предположительно случайном мире криптографии». Тем не менее, исследования в этом направлении продолжаются.

В статье [Al-Ubaidi] разработана так называемая «атака черного ящика» на классические и поточные криптосистемы, основанная на построении модели «нейро-распознавателя черного ящика» (Black-Box Neuro Identifier). Преследуются две цели: во-первых, определение ключа на основе открытых и соответствующим им зашифрованных текстов; во-вторых, создание нейро-модели исследуемой криптосистемы.

Идентификация системы заключается в определении правил функционирования системы-«черного ящика» по входным и выходным данным и аппроксимации неизвестной функции моделью нейронной сети. Первым этапом является выбор нейронной модели, которая характеризуется определенной архитектурой и алгоритмом обучения. Выбор осуществляется методом проб и ошибок. Множество известных пар открытых текстов и криптограмм разделяется на два подмножества, одно из которых используется для обучения сети, а другое – для проверки соответствия полученной модели заданному критерию точности. Оптимальной считается модель, имеющая минимальное число нейронов и при этом удовлетворяющая критерию.

При помощи описанной системы в работе [3] осуществлялся криптоанализ классического полиалфавитного шифра Вижинера и поточных шифров. При установке порога ошибки на уровне 10^{-5} удалось получить модель криптосистем, выдающую результат со 100%-й точностью. К преимуществам такого подхода по сравнению с другими современными методами относится независимость результата от используемого естественного языка и его статистических характеристик, поскольку для получения знаний система использует адаптивный обучающий процесс.

В статье [29] говорится о применении нейронных сетей для взлома DES. В процессе обучения использовалось 2240 пар открытых текстов и криптограмм, что позволило получать результаты с точностью до 98%. В планы авторов также входит криптоанализ AES с использованием разработанной стратегии.

Генетические алгоритмы

Один из первых шифров на основе задачи об укладке ранца был предложен Меркли и Хеллманом в 1978 [46]. Это была одна из первых попыток создания системы шифрования с открытым ключом. Несмотря на то, что проблема укладки ранца относится к классу NP-полных, было показано, что большинство версий алгоритма являются нестойкими. В 1983 г. Брикел предложил способ взлома криптосистемы на основе ранца низкой плотности [14]. Год спустя Шамир разработал полиномиальный алгоритм для атаки на исходную «рюкзачную» криптосистему [61]. После этого было предложено множество других систем на основе алгоритма укладки ранца: несколько последовательных рюкзаков, рюкзаки Грэм-Шамира (Garham-Shamir) и др. [58]. Для всех этих систем были разработаны методы вскрытия. В статье [66] предлагается еще один метод криптоанализа шифров на основе алгоритма укладки ранца; отличительной особенностью такого подхода является его универсальность, т.е. возможность применения к любой версии «рюкзачной» криптосистемы, а также простота работы. В метод базируется на использовании генетических алгоритмов.

Генетические алгоритмы были разработаны Джоном Холландом и представляют собой модификацию так называемого «эволюционного программирования» [30]. Идея Холланда заключалась в том, чтобы создать алгоритм поиска на основе механизмов естественного отбора, известного из биологии, или алгоритм «направленного» случайного поиска. На этапе инициализации процедуры создается популяция возможных решений. На основе этой популяции выводится новое поколение решений, которое, в свою очередь, служит «исходным материалом» для очередного поколения, и т.д. Цикл генетического алгоритма в общем виде представлен на рис. 4 и включает стадии отбора, скрещивания и мутации. Лучшие представители поколения отбираются для воспроизводства популяции; таким образом, по предположению, каждое новое поколение должно содержать лучшие решения, чем предыдущее. Во многих случаях это соответствует действительности [30].

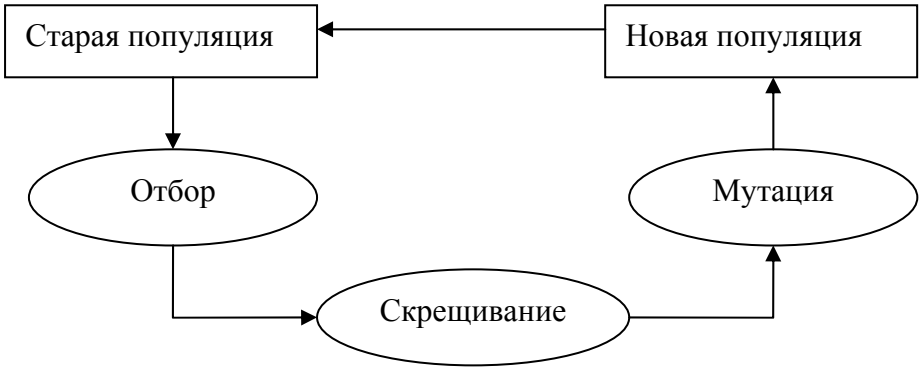

Рис.4

Популяция состоит из набора бинарных строк. Каждая бинарная строка представляет решение проблемы и называется хромосомой. Первой стадией генетического алгоритма является отбор. В процессе отбора определяются строки, которые будут использоваться при создании нового поколения. «Родители» выбираются произвольно, однако «лучшие особи» популяции имеют больший шанс оказаться выбранными. Таким образом алгоритм продвигается в самом перспективном направлении поиска. Следующая стадия – скрещивание. Скрещивание заключается в том, что для пары отобранных строк длины r каждая выбирается произвольным образом число $s \in \{1, ..., r\}$. «Родители» обмениваются битами от $s+1$ до r; таким образом получаются хромосомы потомков (рис. 4).

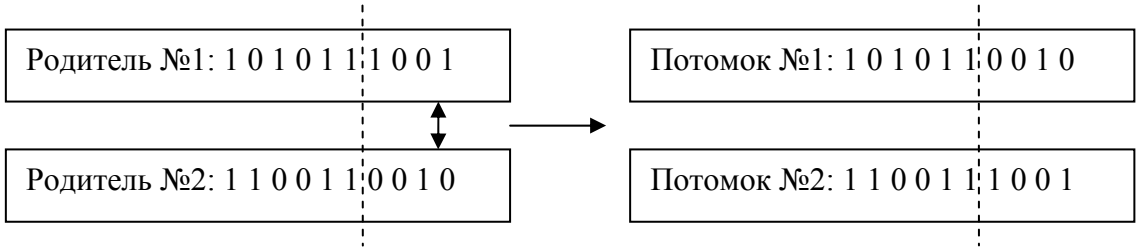

Рис.5

Заключительная стадия – мутация. При инициализации алгоритма устанавливается фиксированная маленькая вероятность мутации, которой подвергаются вновь образовавшиеся хромосомы (рис.5).

Эти стадии повторяются до достижения условия выхода из цикла (таким условием может быть, к примеру, превышение максимального количества популяций).

Рассмотрим формулировку задачи об укладке ранца. Дано множество предметов различного веса; спрашивается, можно ли положить некоторые из этих предметов в ранец так, чтобы его вес стал равен определенному значению? Более формально задача формулируется так: дан набор значений $M_1, M_2, \ldots, M_n$ и суммарное значение S, требуется вычислить значения b_i такие, что:

$$S = b_1 M_1 + b_2 M_2 + \ldots + b_n M_n$$

Здесь b_i может быть либо нулем, либо единицей. Значение $b_i = 1$ означает, что i-й предмет кладут в рюкзак, а $b_i = 0$ - не кладут. Отсюда очевидным образом вытекает представление содержимого рюкзака в виде хромосом, биты которых соответствуют значениям b. Функция выбора «лучших хромосом» оценивает близость веса конкретного рюкзака к заданному числу. Значения функции располагаются в диапазоне [0; 1], где 1 означает точное совпадение с искомым весом. Если вес одного рюкзака превышает целевое значение S на некоторое число x, а вес другого, напротив, меньше требуемого на то же число x, то «лучшим» считается последний рюкзак. Более формально эта функция описывается ниже:

1. Вычислить максимальное расхождение, которое может возникнуть между произвольной хромосомой и целевым значением S: $\Delta_{\max} = \max(S, \tilde{S} - S)$, где $\tilde{S}$ - сумма всех компонентов, которые можно использовать при укладке рюкзака
2. Вычислить вес рюкзака, соответствующего текущей хромосоме, и обозначить S'
3. Если $S' \le S$, то «качество» хромосомы оценивается значением: $\alpha = 1 - \sqrt{\frac{|S' - S|}{S}}$
4. Если $S' > S$, то $\alpha = 1 - \sqrt{\frac{|S' - S|}{\Delta_{\max}}}$

Общий алгоритм, примененный в статье для криптоанализа задачи об укладке ранца, имеет вид:

1. Создается случайная популяция двоичных хромосом
2. Для каждой хромосомы вычисляется значение α (функция оценки)
3. На основе полученных коэффициентов происходит естественный отбор
4. К выбранным на 3-м этапе особям применяется скрещивание
5. Потомки подвергаются мутации
6. Новая популяция анализируется, выделяются «лучшие хромосомы»

Процесс прервется, когда количество поколений превысит определенное заданное число; «лучшие хромосомы» будут использованы для взлома шифра. Для простого шифра подстановки, исследованного в статье, алгоритм показал хорошие результаты: для достижения оптимальной точки, т.е. секретного ключа, алгоритму потребовалось исследовать в среднем не более 2% всего ключевого пространства.

Генетические алгоритмы успешно применяются в криптоанализе перестановочных и подстановочных шифров [44, 67].

Квантовые компьютеры

С помощью квантового компьютера можно проводить вычисления, которые не реализуемы на сегодняшних (классических) компьютерах [89, 94]. В квантовой физике состояние частицы характеризуется так называемой волновой функцией ψ, которая принимает комплексные значения, называемые амплитудами. Аргументом волновой функции является время, а также некоторый набор физических параметров (например, координаты частицы). В теории алгоритмов принято оперировать конечными объектами. Поэтому от принятых в физике бесконечномерных моделей перейдем к конечномерным, считая аргументы волновой функции дискретными.

Для простоты рассмотрим случай, когда волновая функция $\psi(x,t)$ зависит только от координаты x и времени t, которое мы пока зафиксируем. Пусть сначала x может принимать только два значения: 0 и 1. Это соответствует случаю, когда частица может находиться только в двух различных точках. Можно посмотреть на этот случай с другой стороны. Предположим, в нашем распоряжении только один бит для хранения информации о местонахождении частицы, которая находится где-то на отрезке $[0;1]$. Тогда мы примем $x=0$, если она находится в левой половине этого отрезка, и $x=1$, если в правой. Рассмотрим две вспомогательные функции: $|0\rangle$ и $|1\rangle$. Первая равна 1 при $x=0$ и нулю при $x=1$, а вторая наоборот. Тогда любую волновую функцию $\psi(x)$ можно единственным образом записать в форме $\lambda_0|0\rangle+\lambda_1|1\rangle$. Эта запись соответствует разложению вектора двумерного комплексного пространства по базису $|0\rangle,|1\rangle$. Если считать эти базисные вектора ортогональными и единичной длины, то получим, что волновая функция есть вектор двумерного комплексного пространства с выделенным ортонормированным базисом. Такая частица называется *квантовым битом*, или *кубитом*.

Единственный способ узнать, в какой точке находится частица в данным момент времени – это измерить ее волновую функцию. Измерение даст нам любой из векторов $|0\rangle,|1\rangle$ - каждый с вероятностью $|\lambda_o|^2$ и $|\lambda_1|^2$ соответственно. λ_o и λ_1 — это *амплитуды*, связанные условием нормировки: $|\lambda_0|^2+|\lambda_1|^2=1$ (суммарная вероятность равна единице).

Система из n кубитов имеет пространство состояний размерности 2^n. Именно этот экспоненциальный рост пространства состояний в зависимости от числа частиц даёт преимущество в скорости вычислений на квантовых компьютерах в сравнении с классическими.

В 1994 году Питер Шор открыл так называемый «ограниченно-вероятностный» алгоритм факторизации [64], который позволяет разложить на множители число N за полиномиальное время $O((\log N)^3)$, затратив $O(\log N)$ места на квантовом компьютере. В большинстве алгоритмов, включая алгоритм Шора, используется стандартный способ сведения задачи разложения к задаче поиска периода функции. Шор использует квантовый параллелизм для получения суперпозиции всех значений функции за один шаг.

Затем он производит квантовое преобразование Фурье, результатом которого, как и для классического преобразовании Фурье, является функция, аргумент которой кратен величине, обратной периоду. С высокой вероятностью измерение состояния возвращает период, который, в свою очередь, служит для разложения целого числа N. Вышесказанное раскрывает суть квантового алгоритма в очень упрощённом виде. Проблема заключается в том, что квантовое преобразование Фурье основано на быстром преобразовании Фурье и, таким образом, в большинстве случаев дает только приблизительный результат.

Алгоритм Шора разложения чисел на множители явился, пожалуй, главным достижением в области квантовых вычислительных алгоритмов. Это был не только крупный успех математики. Именно с этого момента началось усиленное финансирование работ по созданию квантовых компьютеров.

Эффективность алгоритма Шора была поставлена под сомнение японскими учеными из компании SHARP. Дело в том, что как сам Шор, так и все математики, работающие в области квантовых алгоритмов, говорят о количестве операций, хотя практически важно именно время расчета, которое и определили японцы [55]. Дискретное преобразование Фурье (ДПФ) на квантовом компьютере выполняется как чередующиеся друг за другом преобразования Адамара [94] над отдельными кубитами и операции условного поворота фазы в одном кубите j в зависимости от состояния другого кубита к на угол $\theta = \frac{\pi}{2^{k-j}}$. Если число кубитов равно n, то минимальный угол равен $\frac{\pi}{2^{n-1}}$. Пусть на эту операцию мы затрачиваем время $\tau_{\min}$, тогда на операцию поворота на угол $\frac{\pi}{2}$ для соседних кубитов мы затратим время порядка $\tau = \tau_{\min} 2^n$. Экспоненциальная зависимость, полученная японскими учеными, сводит на нет преимущества алгоритма Шора во времени расчета.

Однако сотрудник ФТИАН Леонид Федичкин указал на опубликованную в 1996 году работу финских авторов [7]. Они исследовали влияние шума на точность ДПФ. Как показано в [55], такое преобразование требует экспоненциально большого (по отношению к числу кубитов n) динамического диапазона углов θ. Что будет, если поворот на малые углы забивается шумом? Оказалось, что требуемая точность операции фазового сдвига в состоянии кубита допускает устранение операции поворота фазы на малые углы. Расчеты Федичкина показывают, что исключение этой операции сохраняет полиномиальную зависимость времени выполнения алгоритма Шора от количества кубитов n.

Так как алгоритм Шора работает только на квантовом компьютере, в настоящее время не существует технических средств, позволяющих за полиномиальное время разложить достаточно большое число на множители. Таким образом, самым важным вопросом остаётся создание квантового компьютера. Алгоритм Шора чрезвычайно прост и довольствуется гораздо более скромным аппаратным обеспечением, чем то, которое понадобилось бы для универсального квантового компьютера. Поэтому вероятно, что квантовое устройство для разложения на множители будет построено задолго до того, как весь диапазон квантовых вычислений станет технологически осуществимым. На сегодняшний день есть конкретные результаты. Так, IBM продемонстрировала использование созданного в лабораториях компании семикубитового квантового компьютера для факторизации чисел по алгоритму Шора. Хотя решённая им задача вряд ли способна поразить воображение (компьютер верно определил, что делителями числа 15 являются числа 5 и 3), это самое сложное вычисление за всю историю квантовых компьютеров.

Заключение

Чтобы снизить вероятность непредсказуемого “обвала” вновь разработанного криптоалгоритма, необходимо заблаговременное проведение криптографических исследований. Разработка любого шифра предусматривает оценку его стойкости к достаточно разнообразным типам криптоаналитических нападений. Как относиться к заявляемым оценкам стойкости с учетом того, что их получение обычно является довольно сложной задачей? Это зависит от того, кто дает оценку [93]. Стойкость шифра рассматривается как разработчиком, так и критиком (криптоаналитиком). Оценки разработчика шифра можно считать корректными, если он делает некоторые допущения в пользу криптоаналитика. Оценки разработчика будут опровергнуты, если кто-либо укажет другой способ криптоанализа, для которого вычислительная сложность получается меньше заявляемой.

Оценки критика являются корректными, если он не занижает значение стойкости по предлагаемому им лучшему методу криптоанализа. Оценки критика будут опровергнуты, если кто-либо найдет и укажет принятые криптоаналитиком существенные допущения, учет которых приводит к значительному увеличению сложности предлагаемого криптоаналитического нападения. Таким образом, если криптоаналитик предлагает корректный вариант атаки, который вычислительно реализуем по оценкам, то практическая проверка должна быть положительной.

В обоих случаях риск того, что оценки будут скомпрометированы, тем меньше, чем больше специалистов анализировали алгоритм, чем выше их квалификация и чем больше времени они уделили анализу. Поэтому открытая публикация криптоалгоритмов, их исследование и публичное обсуждение являются необходимыми.

Для уменьшения возможного ущерба, вызванного несвоевременной заменой криптоалгоритма, потерявшего свою стойкость, желательна периодическая перепроверка стойкости криптоалгоритма. То обстоятельство, что любую задачу отыскания способа раскрытия некоторой конкретной криптосистемы можно переформулировать как привлекательную математическую задачу, при решении которой удается использовать многие методы той же теории сложности, теории чисел и алгебры, привело к раскрытию многих криптосистем. С развитием математики и средств вычислительной техники стойкость криптоалгоритма может только уменьшаться. Если влияние роста мощности компьютеров на стойкость алгоритмов еще можно предсказать с той или иной степенью точности (до настоящего момента каждое десятилетие скорость вычислений вырастала на порядок), то оценить перспективы научного прогресса не под силу даже ученым-криптографам с мировым именем. Так, в 1977 году Рон Ривест заявил, что разложение на множители 125-разрядного числа потребует 40 квадриллионов лет [24]. Однако уже в 1994 г. было факторизовано число, состоящее из 129 двоичных разрядов! Как видно, предсказания – дело неблагодарное, поэтому в данной статье авторы ограничились изложением фактов, касающихся современного состояния и тенденций развития криптоанализа. Хочется надеяться, что этот обзор позволил заинтересованному читателю получить общее представление о теме; более подробная информация доступна в источниках, использованных при написании данной статьи.

Список литературы

1. **Adleman L.** A Subexponential Algorithm for the Discrete Logarithm with Application to Cryptography // Proc. IEEE 20-th Annual Symposium on Foundations of Computer Science (FOCS), 1979. P. 55—60.
2. **Agnew G.B.** Random Sources for Cryptographic Systems // Advances in Cryptology – EUROCRYPT'87 Proceedings, Springler-Verlag, 1988. P. 77-81.
3. **Al-Ubaidy M. K. I.** Black-box attack using neuro-identifier // Cryptologia, Oct 2004.
4. ANSI X3.92. American National Standard for Data Encryption Algorithm (DEA). American National Standards Institute, 1981.
5. AT&T. T7001 Random Number Generator // Data Sheet, Aug 1986
6. **Atkins D., Graff M., Lenstra A.K., Leyland P.C.** The magic words are squeamish ossifrage // Advances in cryptology — ASIACRYPT'94. Wollongong, 1994.
7. **Barenco A., Ekert A., Suominen K.-A., Törmä P.** Approximate Quantum Fourier Transform and Decoherence // Quantum Physics, abstract quant-ph/9601018.
8. **Ben-Aroya I., Biham E.** Differential Cryptanalysis of Lucifer // CRYPTO'93, Springer. P. 187-199.
9. **Berson T.A.** Differential Cryptanalysis Mod with Applications to MD5", EUROCRYPT'92, Lecture Notes in Computer Science, v. 658, Springer, 1992, pp. 71-80.
10. **Biham E., Biryukov A., Shamir A.** Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials // EUROCRYPT'99, 1999, pp.12-23.
11. **Biham E., Shamir A.** Differential Cryptanalysis of DES-like cryptosystems // Journal of Cryptology. 1991. V.4. No. 1. P. 3-72.
12. **Biryukov A., De Canniere C., Quisquarter M.** On Multiple Linear Approximations // 2004. Available via http://www.iacr.eprint-archiv.
13. **Black J., Urtubia H.** Side-channel attacks on symmetric encryption schemes: the case for authenticated encryption // Proc of 11th USENIX Security Symposium, 2002. P. 327-338.
14. **Brickell E**. Solving Low Density Knapsacks // Advances in Cryptology: Proceedings of CRYPTO. New York: Plenum Press, 1984. P. 25 -37.
15. **Brillhart J., Morrison M.A.** A method of factoring and the factorization of F7 // Math. Comp. 1975. V. 29. P. 183—205.
16. **Coppersmith D.** The data encryption standard (DES) and its strength against attacks // Technical Report RC 18613 (81421), IBM Research Division, December, 1992.
17. **Coppersmith D., Odlyzko A., Schroeppel R.** Discrete logarithms in GF(p) // Algorithmica. 1986. V. 1. P. 1—15.
18. **Dam K.W., Lin H. S.** Cryptography's Role in Securing the Information Society. National Academy Press. Washington, D.C. 1996.
19. **Diffie W., Hellman M.** New directions in cryptography // IEEE Trans. Inform. Theory, Vol 22, 6 (1976). P. 644–654.
20. **Dixon J.** Asymptotically fast factorization of integers // Mathematics of computation, vol. 36, no. 153. P. 255-260, 1981.
21. **Fairfield R.C., Mrotenson R.L., Koulthart K.B.** An LSI Random Number Generator (RNG) // Advances in Cryptology: Proceedings of CRYPTO 84, Springler-Verlag, 1985. P. 115-143.
22. **Fiat A., Shamir A.** How to prove yourself: practical solutions to identification and signature problems // Proc. Crypto'86, Lect. Notes in Comput. Sci. V. 263, 1987. P. 186-194.

23. **Frey G., Ruck H.-G.** A Remark Concerning m-Divisibility and Discrete Logarithm in the Divisor Class Group of Curves // In Mathematics of Computation, Vol. 62, pp. 865-874, 1994.
24. **Gardner M.** A New Kind of Cipher That Would Take Millions of Years to Break // Scientific American, v.237, n.8, Aug 1977. P..120 – 124.
25. **Goldwasser S., Micali S., Rivest R.** A secure digital signature scheme // SIAM J. Comput. V. 17, No 2, 1988. P. 169-192.
26. **Golomb S.W.** Shift Register Sequences. Holden-Day, San Francisco, 1967.
27. **Gordon L.A., Loeb M.P., Lucyshyn W., Richardson R.** CSI/FBI Computer Crime and Security Survey 2005 // Computer Security Institute Publications, 2005.
28. **Gude M.** Concept for a high-Performance Random Number Generator Based on Physical Random Phenomena // Frequenz, v. 39, 1985. P. 187-190.
29. **Haranadh Gavara, Harendra Kumar Mishra, Surendra Kumar Y.** Cryptanalysis using Neural Networks // Available via netlab.cs.iitm.ernet.in/cs650/2006/TermPapers/Group9.pdf
30. **Holland J.** Adaptation in Natural and Artificial Systems. Ann Arbor MI: University of Michigan Press. 1975.
31. **Kahn D.** The Codebreakers: The Story of Secret Writing. MacMillan, New York, 1967.
32. **Kaliski B.S., Robshaw M.J.B.** Linear cryptanalysis using multiple approxiamtions // In Proceedings of Advances of Cryptology - CRYPTO`94, (Desmedt Y. Ed.), Lect. Notes in Comp. Sci. Springler-Verlag, 1994. V. 950. P. 26-39.
33. **Kelsey J., Schneier B., Wagner D., Hall C.** Cryptanalytic Attacks on Pseudorandom Number Generators // Fast Software Encryption, Fifth International Workshop Proceedings (March 1998), Springer-Verlag, 1998, pp. 168-188. Available via: http://www.counterpane.com/pseudorandom_number.html
34. **Kerckhoffs A.** La cryptographie militaire // Journal des sciences militaires, vol. IX. P. 5-38, Jan. 1883, (P. 161-191, Feb. 1883).
35. **Knudsen L.R., Mathiassen J.E.** A chosen-plaintext linear attack on DES // In Proceedings of Fast Software Encryption - FSE'2000, (Schneier B. ed.), Lect. Notes in Comp. Sci. Sprinler-Verlag, 2001. V. 1978. P. 262-272.
36. **Knudsen L.R.** Block Ciphers – Analysis, Design, Applicatons // Ph.D. dissertation, Aarhus Unversity, Nov 1994.
37. **Knudsen L.R.** Truncated and Higher Order Differentials // Fast Software Encryption, Second International Workshop, Belgium, December, 1994, Springer. P. 196-211.
38. **Knudsen L.R., Robshaw M.J.B.** Non-Linear Approximation in Linear Cryptanalysis // In Proceedings of Advances of Cryptology - EUROCRYPT'96, (Maurer U. ed.), Lect. Notes in Comp. Sci. Springer-Verlag, 1996. V. 1070. P. 224-236.
39. **Lai X.** Higher Order Derivatives and Differential Cryptanalysis // Communications and Cryptography, Kluwer Academic Publishers, 1994. P. 227-233.
40. **Lehmann E.L.** Testing statistical hypotheses. John Wiley, 1959. (Русский перевод: Леман Э. Проверка статистических гипотез. М.: Наука, 1979.)
41. **Lenstra A.K., Lenstra H.W., Manasse M. S., Pollard J.M.** The factorization of the ninth Fermat number // Math. Comp. 1993. V. 61 (203). P. 319—349.
42. **Lenstra H.W.** Factoring integers with elliptic curves // Ann. Math. 1987. V. 126. P. 649—673.
43. **Matsui M., Yamagishi A.** A new method for known plaintext attack of FEAL cipher // In Proceedings of Advances in Cryptology - EUROCRYPT'92. Lect. Notes in Comp. Sci. Berlin: Springler-Verlag, 1992. V. 658. P. 1-91.
44. **Matthews R.** The use of genetic algorithms in cryptanalysts // Cryptologia. 17(2), 1993. P. 187-201.

45. **McKee J.** Speeding Fermat's factoring method // Math. Comp. 1999. V. 68 (228). P. 1729—1737.
46. **Merkle R.C., Hellman M.E.** Hiding Information and Signatures in Trapdoor Knapsacks // IEEE transactions on Information Theory. V. 24, n. 5, Sep 1978. P. 525-530.
47. **Murphy S.** The cryptanalysis of FEAL-4 with 20 chosen plaintexts // Journal of Cryptology, 1990, v. 3, No. 2. P. 145-154.
48. **Odlyzko A.M.** Discrete logarithms: The past and the future. AT&T Labs – Research, 1999.
49. **Pollard J.** Monte Carlo methods for index computatiom (mod p) // Math. Comp., v. 32, 1978. P. 918–924.
50. **Pomerance C.** Analysis and comparision of some integer factoring algorithms // Computational methods in number theory. V. 1 / H.W. Lenstra and R. Tijdeman, editors. Amsterdam, 1982. P. 89—139.
51. **Quisquarter J.-J., Desmedt Y.G.** Chinese Lotto as an Exhaustive Code-Breaking Machine // Computer, v.24, n.11, Nov 1991. P. 14-22.
52. RIJNDAEL description. Submission to NIST by Joan Daemen, Vincent Rijmen // Available via http://csrc.nist.gov/encryption/aes/round1/docs.htm
53. **Rivest R.L., Shamir A., Adleman L.M.** A Method for Obtaining Digital Signatures and Public-Key Cryptosystems // Communications of the ACM, v.21, n.2, Feb 1978. P. 120-126.
54. **Rivest R.L., Shamir A., Adleman L.M.** On Digital Signatures and Public Key Cryptosystems // MIT Laboratory for Computer Science, Technical Report, MIT/LCS/TR-212, Jan1979.
55. **Saito A., Kioi K., Akagi Y., Hashizume N., Ohta K.** Actual computational time-cost of the Quantum Fourier Transform in a quantum computer using nuclear spins. // Quantum Physics, abstract quant-ph/0001113.
56. **Schirokauer O.** Discrete logarithms and local units. Phil. Trans. R. Soc. Lond. A. 1993. V. 345. P. 409—423.
57. **Schneier B.** A self-study course in block-cipher cryptanalysis // Cryptologia, v.24, n.1, Jan 2000. P. 18-34.
58. **Schneier B.** Applied Cryptography Second Edition: protocols, algorithms and source code in C. John Wiley & Sons Inc., 1996. (Русский перевод: Шнайер Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си. М.: ТРИУМФ, 2002.)
59. **Schneier B.** Security Pitfalls in Cryptography // Cryptography Consultant Counterpane Internet Security, Inc., 2001.
60. **Schneier B.** Snake Oil, Crypto-Gram // February, 1999. Available via http://www.counterpane.com/Crypto-Gram.html
61. **Shamir A.** A Polynomial -Time Algorithm for Breaking the Basic Merkle -Hellman Cryptosystem // Proceedings of the 23rd IEEE Symposium on the Foundations of Computer Science, 1982. P. 145 – 152.
62. **Shanks D.** Class Number, a Theory of factorization, and Genera // Proc. Sympos. Pure Math., vol. 20, Amer. Math. Soc., Providence, R.I., 1971.
63. **Shimizu R., Miyaguchi S.** Fast Data Encipherment Algorithm FEAL // Transactions of IEICE of Japan, v. J70-D, n. 7, Jul 87. P. 1413-1423.
64. **Shor P. W.** Algorithms for Quantum Computation: Discrete Logarithms and Factoring // In Proceedings, 35th Annual Symposium on Foundations of Computer Science, Santa Fe, NM, November 20–22, 1994, IEEE Computer Society Press. P. 124–134.
65. SKIPJACK and KEA Algorithm Specifications // Version 2.0, 1998. Available via: http://csrc.nist.gov/CryptoToolkit/skipjack/skipjack-kea.htm

66. **Spillman R.** Cryptanalysis of knapsack ciphers using genetic algorithms // Cryptologia, 17(1), 1993.
67. **Spillman R., Janssen M., Nelson B., Kepner M.** Use of a Genetic Algorithm in the Cryptanalysis of Simple Substitution Ciphers // Cryptologia. 17(1), 1993. P. 31-44.
68. **Voorhoeve M.** Factorization algorithms of exponential order // Computational methods in number theory. V. 1 / H.W. Lenstra and R. Tijdeman, editors. Amsterdam, 1982. P. 79—88.
69. **Wagner D.** The Boomerang Attack // Proceedings of Fast Software Encryption'99, Springer Verlag, Lecture Notes in Computer Science, v. 1636, 1999. P. 156-170.
70. **Western A.E., Miller J.C.P.** Tables of indices and primitive roots. Cambridge University Press, 1968. (Royal Society Mathematical Tables; V. 9).
71. What is RC4? // RSA Security Laboratories; available via http://www.rsasecurity.com/rsalabs/node.asp?id=2250
72. **White S.R.** Covert Distributed Processing with Computer Viruses. // Advances in Cryptology - CRYPTO'89 Proceedings, Springler-Verlag, 1990. P. 616-619.
73. **Авдошин С.М., Белов В.В.** Обобщенный метод «волны» для решения экстремальных задач на графах // ЖВМиМФ, 1979, 19, №3. – с. 739-755.
74. **Авдошин С.М., Савельева А.А.** Алгоритм решения систем линейных уравнений в кольцах вычетов // Информационные технологии. 2006. № 2. с.50-54.
75. **Алексеев А.** Криптография и криптоанализ: вековая проблема человечества. // Опубликовано: http://infocity.kiev.ua/hack/content/hack008.phtml
76. **Амамия М., Танака Ю.** Архитектура ЭВМ и искусственный интеллект. М.: Мир, 1993.
77. **Баричев С.** Основной вопрос криптографии // Chief Inforamtion Officer – руководитель информационной службы. #5 (37), 2005, с. 93-95.
78. **Варновский Н.П.** О стойкости схем электронной подписи с аппаратной поддержкой. Технический отчет. Лаборатория МГУ по математическим проблемам криптографии, 1997.
79. **Василенко О.Н.** Теоретико-числовые алгоритмы в криптографии. М.: МЦНМО, 2004.
80. Введение в криптографию / Под общей ред. В.В. Ященко // СПб.: Питер, 2001.
81. **Винокуров А., Применко Э.** Сравнение российского стандарта шифрования, алгоритма ГОСТ 28147-89, и алгоритма Rijndael, выбранного в качестве нового стандарта шифрования США // «Системы безопасности», М., изд. «Гротэк», 2001, №№1,2.
82. **Галушкин А.** Теория нейронных сетей. М.:ИПРЖР, 2000.
83. ГОСТ 28147-89. Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования.
84. **Грушо А.А., Применко Э.А., Тимонина Е.Е.** Анализ и синтез криптоалгоритмов. Курс лекций. Йошкар-Ола: МФ МОСУ, 2000.
85. **Жуков А.Е.** Криптоанализ по побочным каналам (Side Channel Attacks). // Материалы конференции РусКрипто – 2006. // Опубликовано: http://ruscrypto.ru/sources/publications/
86. **Загнетко А.** Информация доступная и недоступная. // Опубликовано: 19.06.2006: http://pda.cio-world.ru/?action=article&id=273907
87. **Зубов А.Ю.** Криптографические методы защиты информации. Совершенные шифры: Учебное пособие. М.: Гелиос АРВ, 2005.
88. **Иванов М.А.** криптографические методы защиты информации в компьютерных системах и сетях // М.: КУДИЦ-ОБРАЗ, 2001.

89. **Китаев А., Шень А., Вялый М.** Классические и квантовые вычисления. М.: МЦНМО, 1999.
90. **Колчин В. Ф.** Случайные отображения. М.: Наука, 1984.
91. **Кормен Т., Лейзерсон Ч., Ривест Р.** Алгоритмы: построение и анализ. М.: МЦНМО, 1999.
92. **Логачев О.А., Сальников А.А., Ященко В.В.** Булевы функции в теории кодирования и криптологии. М.: МЦНМО, 2004.
93. **Молдовян Н.** Каким быть новому стандарту шифрования? //"Компьютерра" №2 от 18.01.2000.
94. **Ожигов Ю.И.** Квантовые вычисления. Учебно-методическое пособие. М.: МГУ, факультет ВМиК, 2003.
95. **Осипян В.О., Осипян К.В.** Криптография в задачах и упражнениях. М.: Гелиос АРВ, 2004.
96. Основные тенденции развития открытой криптографии (обзор по заказу cryptography.ru) // Опубликовано: geo.com.ru
97. **Ростовцев А.Г.** Решеточный криптоанализ // Безопасность информационных технологий. - М.: Изд. МИФИ, 1997, №3. - с. 53-55.
98. **Ростовцев А.Г., Маховенко Е.Б.** Теоретическая криптография. СПб.: АНО НПО Профессионал, 2005.
99. **Ростовцев А.Г., Михайлова Н.В.** Методы криптоанализа классических шифров // 1998. Опубликовано: http://crypto.hotbox.ru/download/cryptoan.zip
100. **Сачков В.Н.** Комбинаторные методы дискретной математики. М.: Наука, 1977.
101. **Семаев И. А.** О сложности вычисления логарифмов на эллиптических кривых // Вторая международная конференция по теории чисел и ее приложениям, Тула, 1993.
102. **Шеннон К.Э.** Работы по теории информации и кибернетике // М.: И.Л., 1963.