Как устроен блочный шифр?

Андрей Винокуров (avin@chat.ru).

В настоящей статье вы найдете описание традиционной архитектуры построения блочных шифров, лежащей в основе наиболее известных современных несекретных шифров, таких, как Российский и американский стандарты шифрования. Статья была написана в апреле 1995 года, но по разным причинам не была тогда опубликована. В то время информация по архитектуре классических блочных шифров практически отсутствовала в Российской открытой печати, а русскоязычная терминология в этой области находилась еще в стадии становления. За прошедшее с тех пор время появилось огромное число материалов по обсуждаемой теме, поэтому сегодня статья может показаться слегка наивной. Однако ее коренная переделка заняла бы слишком много времени, проще написать новую статью, и поэтому автор параллельно с этой работой решил опубликовать текущую версию с незначительными изменениями. Статья не предполагает предварительного знакомства читателя с криптографией, однако, содержит достаточное число математических формул и подразумевает владение соответствующим математическим аппаратом.

С о д е р ж а н и е

Введение

Заканчивающееся 20-е столетие является веком не только электричества и атома, в еще большей степени оно может претендовать на то, чтобы называться веком тотальной информатизации и компьютеризации общества. С того самого момента, когда в его середине появились и начали победное шествие по планете устройства для обработки цифровых данных – компьютеры, возникла индустрия производства, обработки и потребления информации, которая в настоящее время стала неотъемлемой частью нашей жизни. Сейчас о технологическом уровне государств имеет смысл судить не по количеству выплавляемой стали или производимых комбайнов для уборки сахарной свеклы, а по совокупной мощностью всех вычислительных средств, приходящихся на одного жителя страны.

О важности информации в современном мире наиболее показательно свидетельствуют следующие факты: Во-первых, обладание определенным цифровым кодом может открыть доступ его владельцу к значительным материальным ценностям и услугам – такое положение вещей имеет место благодаря тому, что информатизация общества не обошла стороной банковско-финансовую сферу. Во-вторых, сложилась и необычайно окрепла индустрия информационных услуг – информация стала обыкновенным товаром, то есть объектом купли–продажи. Многие фирмы преуспевают только благодаря тому, что могут получить важные для их деятельности сведения всего на несколько часов или суток раньше своих конкурентов. В третьих, по оценкам зарубежных экономистов значительная доля западных фирм разорилась бы в течении нескольких дней после разглашения некоторой критически важной информации, лежащей в основе их деятельности.

Особый, нематериальный характер информации делает исключительно легким ее копирование и модифицирование, в силу чего она становится соблазнительным объектом различного рода злоупотреблений. Кроме того, довольно типичной является ситуация, когда нужную кому-либо информацию ее владельцы не согласились бы продать ни за какие деньги, и единственный способ ее получить – это украсть. Указанные причины привели к возникновению целой отрасли человеческой деятельности, основное назначение которой – добывать информацию любыми возможными и невозможными способами, – конечно же, речь идет о разведке. Профессия шпиона наряду с другими, прекрасно всем известными, является одной из древнейших на планете. С другой стороны, статистика неумолимо свидетельствует, что все большая доля всех преступлений совершается в сфере информационных технологий “белыми” и “синими воротничками”, использующими “бреши” информационных систем в своих корыстных целях. Действительно, сейчас, чтобы ограбить банк, не нужно бульдозером проламывать стены хранилищ и резать автогеном сейфы, достаточно узнать код, управляющий доступом к одному из банковских счетов. Все, что для этого требуется, – это компьютер и доступ к банковской сети, ну и конечно, некоторое количество серого вещества в черепной коробке. Прискорбно, но факт – число преступлений с использованием “высоких технологий” растет быстрыми темпами.

Высокая уязвимость информационных технологий к различным злоумышленным действиям породила острую необходимость в средствах противодействия этому, что привело к возникновению и развитию области защиты информации (ЗИ) как неотъемлемой части информационной индустрии. Древнейшей задачей сферы ЗИ является защита передаваемых сообщений от несанкционированного ознакомления с их содержимым, имеются свидетельства понимания людьми этой проблемы еще в древнем Египте и Вавилоне. Информация о способах ее решения в античном мире дошла до нас в виде ссылок на так называемый “код Цезаря” – простейший шифр, применяемый сначала Юлием Цезарем, а впоследствии и другими римскими императорами, для защиты своей переписки от излишне любопытных глаз. Однако вплоть до современности тайнопись была не ремеслом, а искусством, и как наука, криптография сложилась лишь в настоящем веке. Но еще долгое

время после этого шифровальные отделы были исключительной прерогативой дипломатических и разведывательных служб, ситуация кардинально изменилась только в последние десятилетия.

В настоящее время понятие “защита информации” (ЗИ) объединяет в себе множество самых различных значений – от резервирования питания для защиты информации от разрушения при возможных сбоях в питающей сети и охранников в дверях компьютерного зала, препятствующих входу посторонних лиц и выносу сотрудниками носителей информации, до генераторов помех, “глушащих” уносящие информацию излучения. Из всего многообразия методов ЗИ нас с вами интересуют лишь те, которые никак не связаны с характеристиками ее материальных носителей, а основаны на манипулировании самой информацией и используют лишь ее имманентные свойства. Эти методы относятся к ведению криптографии, которая переживает сейчас настоящий бум. На сегодняшний день известно большое количество задач, относящихся к сфере ЗИ, – такое обилие обусловлено тем, что информационные взаимодействия, развиваясь, приобретают все более сложный характер, соответственно становятся более разнообразными и изощренными угрозы в их сторону, а это в свою очередь приводит к возникновению новых задач. Если раньше все потребности в защите информации сводились к обеспечению секретности и подлинности передаваемых сообщений, то есть к их защите от чтения и внесения изменений посторонними лицами, то сейчас актуально гораздо большее число проблем. Среди новых задач отметим лишь две, наиболее известные: рассылка секретных ключей по незащищенным каналам связи (открытое распределение ключей) и подтверждение авторства сообщений (цифровая подпись). А ведь существует большое количество менее известных, но не менее важных задач ЗИ.

Соответственно классам решаемых задач в настоящее время сложились две области криптографии: классическая, или криптография с секретным ключом, и современная, или криптография с открытым ключом. История первой насчитывает тысячелетия, тогда как официальный возраст второй не перевалил еще за три десятка лет. Кратко остановимся на различиях между ними.

1. Классическая и современная криптография.

Классическая, или одно-ключевая криптография решает фактически лишь две задачи: защиту передаваемых сообщений от прочтения и от модификации посторонними лицами. Она опирается на использование симметричных алгоритмов шифрования, в которых за- и расшифрование отличаются только порядком выполнения и направлением некоторых простых шагов. Эти алгоритмы используют один и тот же секретный элемент (ключ), и второе действие (расшифрование) является простым обращением первого (зашифрования). Поэтому каждый из участников обмена может как зашифровать, так и расшифровать сообщение. По причине большой избыточности естественных языков непосредственно в зашифрованное сообщение чрезвычайно трудно внести осмысленное изменение, поэтому классическая криптография обеспечивает также защиту от навязывания ложных данных. Если же естественной избыточности оказывается недостаточно для надежной защиты сообщения от модификации, она может быть искусственно увеличена путем добавления к нему специальной контрольной комбинации, называемой имитовставкой.

Классическая схема шифрования прекрасно работает, пока между участниками информационного обмена есть взаимное доверие. Если же его нет, то могут возникать различные коллизии, так как из-за полной симметрии схемы в случае конфликта между сторонами для независимого наблюдателя нет возможности сделать однозначный вывод, кто из двух участников прав. Действительно, получатель может сам изготовить зашифрованное сообщение и затем объявить, что оно им получено от законного отправителя, а тот, в свою

очередь, может отказаться от авторства в действительности переданного им сообщения, объявив, что его сфабриковал сам получатель, благо соответствующая возможность у того имеется. В этих случаях независимый арбитраж, в функции которого входит разрешение конфликтов между участниками информационного процесса, не сможет определить, кто из них прав, а кто – нет. Приведенный факт означает, что рассматриваемая криптографическая схема не позволяет однозначно подтвердить или опровергнуть авторство сообщения. Кроме того, эта схема нуждается в специальной службе, занимающейся изготовлением секретных ключей и доставкой их участникам информационного обмена. Конечно, если участников обмена всего двое, то проблема невелика – роль такой службы может выполнять один из них или даже оба они попеременно. Но если система насчитывает сотни или даже тысячи связанных между собой узлов обработки информации, это небольшая проблема вырастает в большую головную боль.

Противоречие между ограничениями классической криптографии и постоянно возникающими новыми задачами привело к тому, что во второй половине семидесятых годов были разработаны принципиально новые подходы, позволяющие решить как перечисленные выше проблемы, так и большое число других. Основой послужило открытие так называемых асимметричных криптоалгоритмов, или методов, в которых процедуры прямого и обратного криптопреобразования выполняются на различных ключах и не имеют между собой очевидных и легко прослеживаемых связей, которые позволили бы по одному ключу определить другой. В такой схеме знание только ключа зашифрования не позволяет расшифровать сообщение, поэтому он не является секретным элементом шифра и обычно публикуется участником обмена для того, чтобы любой желающий мог послать ему шифрованное сообщение.

Как видим, современная криптография позволяет решить гораздо более широкий круг задач, чем криптография классическая. На заре ее развития высказывались даже мнения, что она за несколько лет полностью вытеснит свою предшественницу, однако этого не произошло по следующим причинам: Во-первых, алгоритмы с секретным ключом гораздо проще реализуются как программно, так и аппаратно. В силу этого при одинаковых характеристиках производительности и стойкости сложность, а значит и цена аппаратных средств, реализующих шифр с открытым ключом заметно выше цены аппаратуры, реализующей классический шифр, а при программной реализации на одном и том же типе процессора одноключевые шифры работают быстрее двухключевых. Во-вторых, надежность алгоритмов с открытым ключом в настоящее время обоснована гораздо хуже, чем надежность алгоритмов с секретным ключом и нет гарантии, что через некоторое время они не будут раскрыты, как это получилось с криптосистемой, основанной на задаче об укладке ранца. Поэтому для организации шифрованной связи в настоящее время применяются исключительно классические шифры, а методы современной криптографии используются только там, где они не работают, то есть для организации различных хитроумных протоколов типа цифровой подписи, открытого распределения ключей и игры в покер по переписке. Поскольку асимметричные криптографические алгоритмы не являются темой настоящей статьи, автор не будет более задерживаться на этом. Заинтересованный читатель может найти их описание и обсуждение в большом количестве источников, например, в [1,3,4,8].

Необходимо отметить, что в настоящее время публикуется значительное число научных и популярных работ по современной криптографии, тогда как немногочисленные публикации, в которых рассматриваются классические шифры, посвящены в основном или вопросам истории искусства тайнописи, или содержат описание конкретных алгоритмов без исследования общих принципов, лежащих в их основе. Такое положение вещей с одной стороны, является данью моде, а с другой – последствием чрезмерной засекреченности классической криптографии, во всяком случае, нормальным его назвать нельзя. Именно

поэтому в настоящей статье автор решил рассказать об общих принципах построения криптоалгоритмов с секретным ключом, точнее, одного из его классов, называемых блочными шифрами, на достаточно простом уровне, чтобы статья была понятна даже не очень подготовленному читателю, и вместе с тем с необходимой строгостью. Нет необходимости подробно расписывать достоинства рассматриваемых в статье принципов построения алгоритмов шифрования, достаточно лишь сказать, что они лежат в основе двух наиболее известных в России шифров из числа самых сильных, или, если вам так больше нравится, двух наиболее сильных шифров из числа самых известных – Российского и американского стандартов шифрования, алгоритмов ГОСТ 28147-89 и DES, а также большого числа менее известных шифров. Перейдем непосредственно к их изучению.

2. Основные понятия.

Существует несколько принципов построения криптоалгоритмов с секретным ключом – различают потоковые и блочные шифры, можно провести классификацию и по другим признакам. Чтобы рассказать о всех возможных типах шифров данного класса необходимо написать не статью, а довольно объемистую книгу или даже несколько книг. Поэтому автор не будет пытаться объять необъятное, и ограничится в данной статье рассказом об архитектуре, воплощенной в Российском и американском стандартах шифрования – при всех их различиях они похожи как братья, пусть даже и не близнецы. Быть может, изложенные в статье сведения вдохновят читателя с пытливым умом на создание собственного шифра – в этом нет ничего невозможного, ведь криптография в последние годы достигла таких успехов, что сейчас даже наименее развитые в науке страны, не говоря уже о крупных процветающих фирмах из развитых государств, могут позволить себе создать практически нераскрываемый шифр. Тому есть яркий пример – американский стандарт шифрования (DES) первоначально был разработан фирмой IBM для своих собственных нужд, и лишь затем, после некоторых переработок, был принят в качестве федерального стандарта США [2]. Достижения современной криптографии позволяют обеспечить настолько полную конфиденциальность при обработки информации, что даже самые ярые поборники приватности и невмешательства государства в личные дела граждан опасаются ее последствий. Так, Конгрессом Соединенных Штатов рассматриваются законодательные акты, разрешающие применение криптографических средств только под контролем соответствующих служб. При этом все процедуры обмена шифрованными сообщениями должны строиться таким образом, что бы при необходимости спецслужбы по решению судебных органов могли их дешифровать. Возможно, что достаточно скоро до этого дойдет дело и у нас. В этом направлении уже сделано несколько шагов – прочитайте указ президента России №334 от 3 апреля 1995 года “О мерах по соблюдению законности в области разработки, производства, реализации и эксплуатации шифровальных средств, а также предоставления услуг в области шифрования информации”, фактически устанавливающий государственный контроль за разработкой и использованием средств криптозащиты информации. Сомнений насчет того, откуда “уши растут” быть не может – ФАПСИ намерено быть монополистом на этом весьма привлекательном рынке аппаратов, программ и услуг, а государство не хочет, чтобы его граждане имели секреты от него.

Вернемся к сути проблемы: итак, пусть две стороны, которые мы назовем законными участниками обмена информацией, пытаются наладить секретную связь. Из них один является отправителем, а другой – получателем сообщения, хотя, конечно, в реальных ситуациях эти роли не закреплены за участниками жестко и каждому из них приходится быть как отправителем, так и получателем. Задача отправителя заключается в том, чтобы сформировать и отправить получателю сообщение $\boldsymbol{T}$. Задача получателя заключается в том, чтобы посланное сообщение получить и понять его содержание. Чтобы один только получатель мог ознакомиться с содержанием сообщения, отправителю необходимо

преобразовать его согласно некоторому алгоритму E таким образом, что бы ни кто, за исключением законного получателя и, быть может, самого отправителя, не мог восстановить его в прежнем виде. Это преобразование называется зашифрованием сообщения T, а то, что получается в результате этой процедуры, называется шифртекстом T'. Связь между исходным текстом T, шифртекстом T' и алгоритмом зашифрования E может быть символически выражена с помощью следующей формулы: $T' = E(T)$. Шифрованное сообщение T' передается отправителем получателю по каналу связи. Чтобы законный получатель мог ознакомиться с содержимым полученного сообщения, он должен предварительно его расшифровать, то есть применить к шифртексту T' алгоритм расшифрования D, в результате чего будет восстановлено исходное сообщение T. Таким образом, расшифрование данных осуществляется согласно уравнению $T = D(T')$. Чтобы обеспечить секретность связи, алгоритм расшифрования не должен быть известен посторонним лицам, поскольку его секретность определяет секретность организованной таким образом связи.

В нашей ситуации присутствует третья сторона, называемая согласно сложившейся традиции злоумышленником, которая желает воспрепятствовать осуществлению намерений законных участников обмена. В наиболее общем случае во исполнение своих намерений злоумышленник может перехватывать шифрованные сообщения и посылать сфабрикованные им самим сообщения одной из сторон от имени другой. Конечно, он не владеет алгоритмом расшифрования, – в этом случае все было бы предельно просто для него. Круг задач, которые может попытаться решить злоумышленник, шире, чем просто дешифрование сообщения – перечислим эти задачи, именуемые в криптографии угрозами:

- дешифровать сообщение T', то есть получить соответствующий ему открытый текст T полностью или частично, или хотя бы сделать некоторые заключения о содержании перехваченного сообщения, опираясь на обнаруженные в нем закономерности;
- сформировать на основе имеющихся у него данных сообщение $\tilde{T}'$, которое законный получатель принял бы за подлинное. При этом вовсе не обязательно, хотя, конечно, весьма желательно для злоумышленника, чтобы он сам мог понимать смысл составленного сообщения или даже мог сфабриковать любое выбранное им по своему усмотрению сообщение.

Под раскрытием шифра понимают успешную реализацию хотя бы одной из указанных угроз. Первая угроза, если будет осуществлена, нарушит секретность сообщения, а вторая нарушит его подлинность. Степень успеха в реализации перечисленных выше угроз также может быть различной. Если исключить случайную удачу, то успешная реализация угрозы секретности данных означает овладение злоумышленником алгоритмом расшифрования D, или построение функционально эквивалентного ему алгоритма D', позволяющего для любого зашифрованного сообщения T', или хотя бы для шифртекстов из некоторого класса, получить соответствующее открытое сообщение T. Аналогично, успешная реализация угрозы целостности данных означает овладение злоумышленником алгоритмом зашифрования E, или построение функционально эквивалентного ему алгоритма E', позволяющего для любого открытого сообщения T, или хотя бы для сообщений из некоторого класса, получить соответствующее зашифрованное сообщение T'. Или, в наименее успешном для злоумышленника случае, создание алгоритма, позволяющего построить такое зашифрованное сообщение T' без знания соответствующего ему открытого сообщения T, которое законный получатель принял бы за подлинное.

Для осуществления своих угроз злоумышленнику требуется выполнить некоторую достаточно интеллектуальную работу с перехваченными данными, в этом ему может помочь криптоаналитик. Нетрудно догадаться, что в задачу последнего входит криптоанализ, то есть анализ перехваченных сообщений с целью осуществления одной из перечисленных выше угроз. При этом аналитик может располагать следующей информацией:

- дешифруемый текст T', также могут иметься перехваченные ранее шифрованные сообщения

$(T_1', T_2', ..., T_n')$, зашифрованные с использованием того же самого алгоритма, что и T';

- открытые сообщения, соответствующие некоторым ранее перехваченным шифровкам: $(T_1, T_2, ..., T_m)$, $T_i' = E(T_i)$, $i = 1, ..., m$, где $m \le n$;
- возможность получить для произвольного выбранного злоумышленником открытого сообщения T соответствующий шифртекст $T' = E(T)$;
- возможность получить для произвольного выбранного злоумышленником шифрованного сообщения T' соответствующий открытый текст $T = D(T')$.

Соответственно этим трем возможностям различают следующие основные виды криптоанализа:

- анализ на основе только шифртекста;
- анализ на основе невыбранного открытого текста;
- анализ на основе выбранного открытого текста;
- анализ на основе выбранного шифртекста;

Первая возможность есть в распоряжении аналитика практически всегда. Вторая возможность также вполне реальна: например, разведке через агентуру удалось достать расшифровку одного из секретных сообщений. Более того, эта возможность весьма типична в тех сферах, где срок жизни секретной информации весьма мал, и она быстро предается огласке – при передаче рекламных материалов, различных репортажей, публикуемых впоследствии средствами массовой информации и т.д., словом везде, где надо опередить конкурентов всего на несколько часов или дней. Третья и четвертая возможности, хотя и кажутся экзотическими, тем не менее также могут иметь место – если сотрудник организации, работающий на другую сторону, имеет доступ к общим шифровальным средствам. Конечно, третий случай имеет смысл применительно к задаче дешифрования, а четвертый – навязывания ложных данных. Первые два актуальны для обеих угроз.

В рассматриваемой нами жизненной ситуации присутствует еще одна сторона, роль которой попытаемся выполнить мы сами. Это криптограф, в задачу которого входит снабдить законных участников обмена такими алгоритмами за- и расшифрования, чтобы злоумышленник не смог выполнить ни одну из своих угроз даже в самой благоприятной для него ситуации, то есть при возможности криптоанализа на основе произвольно выбранного открытого текста. Иными словами, задачей криптографа является разработка секретной и аутентичной системы передачи данных. Вообще говоря, это различные, хотя порой и связанные довольно тесно в конкретных шифрах свойства криптосистем. Поясним их смысл:

Аутентичность **есть защищенность криптографической системы от навязывания ложных данных.**

Секретность **есть защищенность криптографической системы от несанкционированного ознакомления с содержимым закрываемых системой данных.**

3. Шифры с секретным ключом.

Итак, наша задача заключается в том, чтобы снабдить участников обмена надежными алгоритмами шифрования. Первый вопрос, который мы себе зададим, – это решаема ли задача в принципе, и если да, то какую максимальную степень секретности мы можем обеспечить? Ответ на этот вопрос был найден Шенноном – теорема, носящая его имя, гласит, что существуют абсолютно стойкие шифры, то есть такие шифры, которые невозможно раскрыть, даже если криптоаналитик обладает неограниченным запасом времени и вычислительных ресурсов. Шенноном также было установлено, что условием абсолютной стойкости шифра является использование в алгоритме шифрования не меньшего количества секретной информации, чем содержится информации в шифруемом сообщении.

Как реализовать такой шифр? Прежде чем отвечать на этот вопрос вспомним, что все современные шифры базируются на принципе Кирхгофа (часто в переводах с английского его фамилия приводится также в транскрипции “Керхкофф”). Этот принцип гласит, что алгоритмы шифрования должны строиться таким образом, чтобы даже при их разглашении они все еще обеспечивали определенный уровень надежности. Вернемся к обсуждаемой теме – ясно, что криптоалгоритмы, построенные в соответствии с принципом Кирхгофа должны использовать при шифровании секретные данные, называемые ключом, которые и обеспечивают секретность сообщения в условиях открытости самого алгоритма. Другими словами, секретность шифра должна обеспечиваться секретностью ключа, а не секретностью алгоритма шифрования. Это означает, что алгоритмы E и D, введенные нами в рассмотрение в предыдущем разделе, используют секретный ключ K, и могут быть обозначены нами теперь как E_K и D_K. Тогда уравнения шифрования будут иметь следующий вид:

$T' = E_K(T)$ (зашифрование),
$T = D_K(T')$ (расшифрование).

Вспомним, что в шифрах рассматриваемого класса для за- и расшифрования используется один и тот же ключ. Ясно, что процедура расшифрования должна в любом случае приводить к правильному результату. Это означает, что каковы бы ни были допустимые блок данных T и ключ K, должно выполняться следующее равенство: $E_K(D_K(T)) = T$.

Вернемся к абсолютно стойким шифрам, реализующим принцип Кирхгофа. Так как вся их секретность сосредоточена в ключе K, применительно к ним требование Шеннона означает, что размер ключа шифрования не должен быть меньше размера шифруемого сообщения: $|K| \geq |T|$. Будем полагать, что они имеют одинаковый размер, равный N бит: $|K| = |T| = N$. Это минимум, при котором все еще возможна абсолютная стойкость. Для зашифрования сообщение T необходимо скомбинировать с ключом K с помощью некоторой бинарной операции “$\circ$” таким образом, чтобы полученный шифртекст зависел и от исходного текста T, и от ключа K. При этом уравнение зашифрования будет иметь следующий вид:

$T' = E_K(T) = T \circ K$.

Размер шифртекста при этом также равен N бит: $|T'| = |T| = |K| = N$. Для обеспечения абсолютной стойкости шифра количество секретной информации в ключе K должно быть максимально возможным для его размера. Это означает, что все биты ключа должны быть случайны с равновероятными значениями и статистически независимы. Такой ключ может быть получен только аппаратным способом, алгоритмически его выработать нельзя, так как в этом случае указанное требование будет нарушено и шифр перестанет быть абсолютно стойким.

Обсудим теперь требования, которым должна удовлетворять операция “$\circ$”. Во-первых, чтобы шифрование было обратимым, уравнение $T \circ K = T'$ должно быть однозначно разрешимо относительно T при любых значениях T' и K. Это означает, что у бинарной операции “$\circ$” должна существовать обратная, которую мы обозначим через “$\bullet$”, и каковы бы ни были N-битовые блоки данных T и K, всегда должно выполняться равенство $(T \circ K) \bullet K = T$. Во вторых, для обеспечения полной секретности шифра необходимо, чтобы разные ключи давали для одинаковых исходных текстов разные шифртексты. Это равносильно требованию однозначной разрешимости уравнения $T \circ K = T'$ относительно K. Так как секретность шифрованного сообщения целиком опирается на секретность ключа, во всем остальном операции “$\circ$” и “$\bullet$” могут выбираться из соображений удобства. В качестве таких операций может использоваться сложение и вычитание по модулю 2^N:

$T \circ K = (T + K) \bmod 2^N$, $T \bullet K = (T - K) \bmod 2^N$.

Осуществить вычисления над сообщением как единым целым может оказаться затруднительным по причине его значительного размера, поэтому целесообразно разбить сообщение и ключ на блоки меньшего размера и применить указанные операции к этим блокам:

$T = (T_1, T_2, ..., T_n)$, $K = (K_1, K_2, ..., K_n)$, $|K_i| = |T_i| = N_i$,

$T_i \circ K_i = (T_i + K_i) \bmod 2^{N_i}$, $T_i \bullet K_i = (T_i - K_i) \bmod 2^{N_i}$.

Если довести этот процесс дробления до логического конца, мы придем к операции побитового сложения по модулю 2, называемой также ***побитовым исключающим или:***

$T \circ K = T \bullet K = T \oplus K$.

Последняя операция оказалась обратной к самой себе и по этой причине, а также в силу своей простоты и легкости реализации (отдельные биты сообщения в ней обрабатываются независимо друг от друга), получила наибольшее распространение.

Кратко остановимся на достигнутом: шифр, который мы сейчас получили, называется одноразовой гаммой Вернама. Этот шифр обладает абсолютной стойкостью, которая, однако, оплачивается достаточно дорогой ценой – для шифрования сообщения нужен ключ такого же размера, предварительно доставленный отправителю и получателю.

Для шифрования двух различных сообщений нельзя применять одну и ту же последовательность элементов гаммы. Если это требование нарушено, всегда можно найти такую пару бинарных операций “$\otimes$” и “$\varnothing$”, которые, будучи примененными к блокам соответственно открытых и зашифрованных с использованием одного и того же элемента гаммы данных, дадут идентичные результаты:

$T_i' \otimes \tilde{T}_i' = T_i \varnothing \tilde{T}_i$.

Это сделает задачу криптоанализа тем более легкой, чем больше избыточности содержится в сообщении. Для текстов на естественных языках ввиду их колоссальной избыточности эта задача становится почти тривиальной, – отделить друг от друга такие сообщения не представляет особого труда. Такое разделение может быть выполнено методом перебора, причем на каждом шаге в переборе участвуют лишь несколько символов.

Если для наложения гаммы используется операция побитового суммирования по модулю 2, то в качестве бинарных операций, устраняющих влияние секретной гаммы на результат, может быть использована та же самая операция. Действительно, пусть, например, два блока шифруются с помощью одного и того же элемента гаммы Γ_i налагаемого на них операцией побитового сложения по модулю 2:

$T_i' = T_i \oplus \Gamma_i$,

$\tilde{T}_i' = \tilde{T}_i \oplus \Gamma_i$.

Выполним побитовое сложение блоков шифртекста по модулю 2:

$T_i' \oplus \tilde{T}_i' = (T_i \oplus \Gamma_i) \oplus (\tilde{T}_i \oplus \Gamma_i) = (T_i \oplus \tilde{T}_i) \oplus (\Gamma_i \oplus \Gamma_i) = (T_i \oplus \tilde{T}_i) \oplus 0 = T_i \oplus \tilde{T}_i$.

Как видим, результат совпадает с побитовой суммой по модулю 2 блоков открытого текста, что делает криптоанализ тривиальным.

Требование уникальной гаммы для каждого шифруемого сообщения делает шифрование с помощью одноразовой гаммы Вернама настолько накладным, что его использование экономически оправдано лишь в каналах связи для передачи сообщений исключительной важности, задействованных к тому же не слишком часто.

Барьер, перед которым мы остановились, непреодолим: достичь эффективности практической реализации криптоалгоритма можно только отказавшись от абсолютной стойкости. Шифр, не являющийся абсолютно стойким, можно раскрыть за конечное время,

точнее, за конечное число шагов, каждый из которых состоит в выполнении элементарной арифметической или логической операции. Однако ничто не мешает нам создать такой теоретически нестойкий шифр, для раскрытия которого требовалось бы выполнить настолько большое число операций, чтобы это было неосуществимо на современных и ожидаемых в не очень далекой перспективе вычислительных средствах за разумное время. Мерой практической стойкости шифров подобного типа может служить количество работы, необходимое для их раскрытия, выраженное в элементарных операциях или в длительности соответствующих вычислений на современных компьютерах. Отметим, что в реальных системах криптозащиты информации используются почти исключительно теоретически нестойкие шифры.

Хороший шифр должен быть устойчив к статистическому и алгоритмическому анализу, то есть не должно существовать легко обнаружимых статистических связей между открытым и шифрованным текстом и зависимости между ними должны быть достаточно сложными для того, что бы их можно было обнаружить путем анализа. Есть четкая граница между хорошо и не очень хорошо спроектированными блочными шифрами: первый невозможно раскрыть способом, более эффективным чем полный перебор по всем возможным значениям ключа, для раскрытия второго могут оказаться пригодными и более эффективные способы. Как же строить стойкие шифры? Наука об этом засекречена практически везде, где имеют дело с криптографией – публикуются лишь соображения, носящие самый общий характер и не содержащие никакой конкретики. Приобщиться к этим тайным знаниям можно, лишь работая в соответствующем подразделении соответствующей специальной службы. Ну а мы с вами попытаемся рассмотреть проблему исключительно с позиции здравого смысла. По большому счету, можно предложить лишь два фундаментальных подхода к построению шифра с секретным ключом:

- вырабатываемый элемент гаммы Γ_i не зависит от шифруемого блока данных T_i;
- шифрование массива информации T осуществляется путем манипулирования с ним самим;

Первый подход очевидным образом вытекает из шифра Вернама. Все отличие заключаются в том, что в нем гамма сама по себе не является ключевым элементом, а только вырабатывается из ключа K фиксированного размера с помощью некоторого набора функций f_i: $\Gamma_i = f_i(K)$, или, точнее, одной универсальной функции $\Gamma_i = f(i,K)$, – хотя с точки зрения математики это одно и то же, с алгоритмической точки зрения это разные вещи. Требование практической реализуемости шифра в виде аппарата или программы для ЭВМ приводит к необходимости возможности его описания в форме алгоритма с конечным числом возможных состояний, наиболее общей моделью которого может служить конечный автомат. Таким образом, генератор гаммы может быть определен с помощью следующих соотношений как конечный автомат без входа:

$S_i = \Omega_K(S_{i-1})$, $\Gamma_i = \Theta_K(S_i)$, (простое гаммирование),

или как конечный автомат со входом, если вырабатываемый блок гаммы зависит от предшествующего блока шифртекста и/или открытого текста:

$S_i = \Omega_K(S_{i-1}, T_{i-1}, T'_{i-1})$, $\Gamma_i = \Theta_K(S_i)$, (гаммирование с обратной связью).

Первое из двух соотношений определяет правило изменения состояний, второе – правило выработки выходного элемента, то есть элемента гаммы. Ясно, что для обеспечения секретности шифра оба эти правила должны или сами быть секретными, или зависеть от значения секретного ключа K. Шифры, построенные по данной схеме, называются потоковыми, так как в них используется поток гаммы, вырабатываемый генератором. Зашифрование (расшифрование) осуществляется путем простого наложения элементов гаммы на блоки открытого (шифрованного) текста с помощью соответствующих бинарных операций: $T'_i = T_i \circ \Gamma_i$, $T_i = T'_i \bullet \Gamma_i$. В зависимости от используемых операций наложение

гаммы может осуществляться как побитово, так и блоками иного размера. Основная сложность при реализации данного подхода заключается в разработке действительно качественного источника криптостойкой гаммы.

Второй подход к построению шифров с секретным ключом не содержит в себе никаких намеков на возможные способы его реализации. В следующем разделе статьи мы попытаемся нащупать эти способы сами.

4. Базовая идея блочного шифра.

Отправной точкой в реализации рассматриваемого подхода является идея вырабатывать гамму для зашифрования сообщения … из самого сообщения! Однако сделать это непосредственно невозможно, так как при этом возникают трудности с расшифрованием: пусть гамма вырабатывается из шифруемого блока согласно уравнению $\Gamma=f(T)$, где f – некоторая функция. При этом уравнение зашифрования будет иметь следующий вид: $T'=E_K(T)=T\circ\Gamma=T\circ f(T)$. Для расшифрования сообщения его получатель должен решить это уравнение относительно T: $T\circ f(T)=T'$. Если функция f сложна и нелинейна, что требуется для обеспечения достаточной стойкости шифра, данная задача практически неразрешима.

Тем не менее, при разработке новой криптографической схемы нам не хотелось бы отказываться от ранее использованных и хорошо зарекомендовавших себя решений, к числу которых относится наложение гаммы на данные для их шифрования. Как же выйти из того затруднительного положения, в котором мы оказались? Попытаемся решить хотя бы часть проблемы, для чего представим шифруемый блок данных T размера $|T|=N$ в виде пары блоков меньшего размера: $T=(T_0,T_1)$, $|T_0|=N_0$, $|T_1|=N_1$, $N_0+N_1=N$, где T_0 будет обозначать младшую, а T_1 – старшую часть массива T. Выполним зашифрование старшего блока с помощью младшего, используя при этом некоторую функцию f, отображающую N_1-битовый блок данных на N_0-битовый, и обратимую бинарную операцию “$\circ$” над N_0-битовыми блоками данных. Полученное шифрующее преобразование обозначим через Γ_f°. Уравнение этого преобразования будет следующим:

$$\Gamma_f^{\circ}(T)=\Gamma_f^{\circ}(T_0,T_1)=(T_0,T_1\circ f(T_0)).$$

Для Γ_f° легко построить обратное, или расшифровывающее преобразование $\Gamma_f^{\bullet}$:

$$\Gamma_f^{\bullet}(T)=\Gamma_f^{\bullet}(T_0,T_1)=(T_0,T_1\bullet f(T_0)).$$

Действительно, если бинарные операции “$\circ$” и “$\bullet$” взаимно обратные, каков бы ни был N-битовый блок данных $T=(T_0,T_1)$, всегда справедливо следующее равенство:

$$\Gamma_f^{\bullet}(\Gamma_f^{\circ}(T))=\Gamma_f^{\bullet}(\Gamma_f^{\circ}(T_0,T_1))=\Gamma_f^{\bullet}(T_0,T_1\circ f(T_0))=(T_0,(T_1\circ f(T_0))\bullet f(T_0))=(T_0,T_1)=T.$$

Ясно, что назначение функции f заключается в том, чтобы маскировать зависимость между блоком T_0 и гаммой для шифрования блока T_1, которая из него вырабатывается. Для этого функция f должна быть секретным элементом нашего шифра – мы пока закроем глаза на это нарушение принципа Кирхгофа. Отметим очень важный факт: наша схема работоспособна при любой функции f, после расшифрования мы всегда получим те же самые данные, которые были перед зашифрованием.

Прежде чем перейти к изучению дальнейшего материала, читателям, не очень сильным в математике, стоит познакомиться с некоторыми математическими понятиями, а именно, с понятием композиции отображений и преобразований. Те же, кто владеет этим материалом в достаточной степени, могут не читать следующие абзацы, напечатанные мелким шрифтом.

Все рассматриваемые в данной части статьи преобразования являются операторами в множестве блоков данных, то есть функциями, принимающими в качестве аргумента и выдающими в качестве результата блоки данных.

1. Назовем композицией преобразований A и B такое преобразование $C = AB$, что каков бы ни был блок данных T, всегда выполняется равенство $C(T) = B(A(T))$. Таким образом, по определению композиции $AB(T) = B(A(T))$.
2. Для композиции преобразований справедлив ассоциативный закон, то есть для любых преобразований A, B, C справедливо тождество: $A(BC) = (AB)C$. Действительно, каким бы ни был блок данных T, справедливо следующее равенство:
   $(A(BC))(T) = BC(A(T)) = C(B(A(T))) = C(AB(T)) = ((AB)C)(T)$.
   Поэтому в выражении для композиции трех и более преобразований скобки излишни: $A(BC) = (AB)C = ABC$.
3. Среди всех преобразований существует одно особое, называемое тождественным и обозначаемое нами через I. Отличительной особенностью данного преобразования является то, что оно оставляет свой аргумент неизменным: каков бы ни был блок данных T, всегда справедливо $I(T) = T$. Очевидно, что композиция любого преобразования A с тождественным дает в результате это же преобразование A: $IA = AI = A$. Действительно, для любого блока данных T справедливы следующие равенства: $AI(T) = I(A(T)) = A(T)$ и $IA(T) = A(I(T)) = A(T)$.
4. Преобразование B называется обратным к преобразованию A, если их композиция является тождественным преобразованием, то есть если выполняется условие $AB = I$ или для произвольного блока данных T справедливо равенство $B(A(T)) = T$. Преобразование A называется обратимым, если существует обратное к нему преобразование, обозначаемое A^{-1}: $AA^{-1} = I$.

С изложенной только что точки зрения шифрующее преобразование должно быть обратимым, то есть должно выполняться свойство $\Gamma_f^{\circ}\Gamma_f^{\bullet} = I$. Следует отметить, что данное свойство выполняется всегда, какую бы функцию f мы не использовали в нашем преобразовании, если только бинарные операции "$\circ$" и "$\bullet$" являются взаимно обратными.

Теперь вспомним про то, что предложенная нами схема решила лишь половину проблемы, так как младшая часть T_0 блока T осталась незашифрованной. Но эта проблема имеет очевидное решение: на следующем шаге необходимо зашифровать часть T_0 блока T, используя элемент гаммы, выработанный из части T_1' с использованием некоторой другой функции g, отображающей N_0-битовые блоки данных на N_1-битовые. Теперь обе части исходного блока окажутся зашифрованными. По ряду причин, однако, принято, что шифруемая на каждом таком шаге и используемая для выработки гаммы части находятся на фиксированных позициях внутри блока – традиция предписывает вырабатывать гамму из младшей части и накладывать ее на старшую. По крайней мере, дело обстоит именно так в ГОСТе, DESe, и всех других шифрах, построенных по их образу и подобию. Для того, чтобы обеспечить указанное свойство, между шагами шифрования необходимо выполнить перестановку частей блока, помещающую соответствующие части на надлежащие места.

По соображениям обеспечения максимальной криптостойкости и эффективности реализации шифра целесообразно взять размеры старшей и младшей частей шифруемого блока одинаковыми: $N_0 = N_1 = N/2$. Именно такое условие выбрали разработчики большинства шифров рассматриваемой архитектуры. В этом случае между шагами алгоритма части шифруемого блока просто меняются местами. Однако существуют шифры, не подчиняющиеся этому правилу, о них будет сказано несколько слов ниже в настоящей статье.

Обозначим через S операцию перестановки старшей и младшей частей массива информации: $S(T) = S(T_0, T_1) = (T_1, T_0)$. Очевидно, что операция S является обратной для самой себя: $S^2 = S \cdot S = I$. Действительно, для произвольного блока данных $T = (T_0, T_1)$ справедливо равенство:

$S^2(T) = S^2(T_0, T_1) = S(S(T_0, T_1)) = S(T_1, T_0) = (T_0, T_1) = T$.

Тогда наша новая схема шифрования может быть представлена композицией более простых шагов:

$\Gamma^{\circ}_{f,g} = \Gamma^{\circ}_{f} \cdot S \cdot \Gamma^{\circ}_{g}$.

При этом обратное, или расшифровывающее преобразование может быть представлено следующим соотношением:

$\Gamma^{\bullet}_{g,f} = \Gamma^{\bullet}_{g} \cdot S \cdot \Gamma^{\bullet}_{f}$.

Действительно, справедливо следующее равенство:

$$\Gamma^{\circ}_{f,g} \cdot \Gamma^{\bullet}_{g,f} = (\Gamma^{\circ}_{f} \cdot S \cdot \Gamma^{\circ}_{g}) \cdot (\Gamma^{\bullet}_{g} \cdot S \cdot \Gamma^{\bullet}_{f}) = \Gamma^{\circ}_{f} \cdot S \cdot \Gamma^{\circ}_{g} \cdot \Gamma^{\bullet}_{g} \cdot S \cdot \Gamma^{\bullet}_{f} = \Gamma^{\circ}_{f} \cdot S \cdot (\Gamma^{\circ}_{g} \cdot \Gamma^{\bullet}_{g}) \cdot S \cdot \Gamma^{\bullet}_{f} = \Gamma^{\circ}_{f} \cdot S \cdot I \cdot S \cdot \Gamma^{\bullet}_{f} =$$
$$= \Gamma^{\circ}_{f} \cdot (S \cdot I) \cdot S \cdot \Gamma^{\bullet}_{f} = \Gamma^{\circ}_{f} \cdot S \cdot S \cdot \Gamma^{\bullet}_{f} = \Gamma^{\circ}_{f} \cdot (S \cdot S) \cdot \Gamma^{\bullet}_{f} = \Gamma^{\circ}_{f} \cdot I \cdot \Gamma^{\bullet}_{f} = (\Gamma^{\circ}_{f} \cdot I) \cdot \Gamma^{\bullet}_{f} = \Gamma^{\circ}_{f} \cdot \Gamma^{\bullet}_{f} = I.$$

Операции наложения гаммы в шагах шифрования блока, вообще говоря, могут быть различными, однако особого смысла в этом во время создания шифров не видели.

Как уже отмечалось выше, существуют шифры, в которых шифруемый блок данных делится на две не одинаковые по размеру части. Если элемент гаммы вырабатывается из младшей части блока, имеющей размер N_0 и накладывается на старшую часть, имеющую размер N_1, то схема более устойчива к криптоанализу когда выполняется условие $N_1 < N_0$. При этом просто менять местами части блока между шагами шифрования уже не годится, необходимо после каждой такой перестановки заново разбивать блок на части. Обычно в такой ситуации используют циклический сдвиг (вращение) блока на N_1 бит влево или вправо, а при расшифровании между шагами необходимо будет повернуть блок на такое же число битов в обратную сторону. В указанном случае выражения для преобразований зашифрования и расшифрования блока будут следующими:

$\Gamma^{\circ}_{f,g} = \Gamma^{\circ}_{f} \cdot R^{\rightarrow}_{N_1} \cdot \Gamma^{\circ}_{g}$,

$\Gamma^{\bullet}_{g,f} = \Gamma^{\bullet}_{g} \cdot R^{\leftarrow}_{N_1} \cdot \Gamma^{\bullet}_{f}$,

где через $R^{\leftarrow}_{m}$ и $R^{\rightarrow}_{m}$ обозначены операторы вращения блока данных на m бит влево и вправо соответственно. Поскольку за один шаг алгоритма шифруется $N_1 < N/2$ битов блока, то для зашифрования всего блока потребуется более двух шагов. Точное значение числа минимально требуемых шагов в таком алгоритме равно $\lceil N/N_1 \rceil$, где через $\lceil x \rceil$ обозначен результат округления числа x до ближайшего целого в сторону увеличения. Из соображения простоты реализации шифра обычно выбирают N_1 так, чтобы размер блока N делился на него без остатка. Как правило, если $N = 64$, то берут $N_1 = 16$ или $N_1 = 8$.

Следует отметить, что шифров, построенных по такому принципу, намного меньше чем шифров, в которых блок делится на две одинаковые по размеру части. Обусловлено это тем, что в них за один шаг шифруется меньшее количество битов, и, соответственно, требуется больше шагов. По такому принципу, например, построен шифр под названием “албер”, созданный в недрах одного из многочисленных НИИ в системе бывшего 8-го Главного Управления КГБ СССР, и, поговаривают, даже претендовавший на место Российского стандарта шифрования, для него $N_1 = 8$.

5. Шифр простой замены.

Для того, чтобы блочная схема шифрования была устойчива к криптоанализу, она должна обладать свойствами перемешивания и рассеивания. Это означает, что каждый бит исходного текста должен влиять на все биты шифртекста, причем характер этого влияния не должен быть прослеживаем ни статистически, ни алгоритмически. Это требование важно именно для блочных шифров по следующей причине: для раскрытия шифра по алгоритмической линии криптоаналитик может попытаться в явном виде вывести соотношения, связывающие входы и выходы алгоритма. Для потокового шифра это

бесполезно, потому что эти соотношения целиком определяются элементами гаммы, которые различны для различных блоков шифруемых данных. А вот чтобы криптоанализ не увенчался успехом для блочного шифра, требуется, чтобы характер влияния входных данных на выходные был достаточно сложным для того, чтобы его можно было выявить путем анализа массивов входных и выходных данных. Это, в частности, подразумевает отсутствие статистической зависимости битов выходного блока от битов входного, что означает на практике, что каким бы образом мы ни изменили блок открытых данных $\boldsymbol{T}$, все биты блока шифрованных данных $\boldsymbol{T'} = \boldsymbol{E_K}(\boldsymbol{T})$ с вероятностью $1/2$ независимо друг от друга изменят свое значение. Но построенная в предыдущем разделе схема шифрования не обладает таким свойством. Действительно, пусть зашифрованию подвергается блок данных $\boldsymbol{T} = (\boldsymbol{T}_0, \boldsymbol{T}_1)$, тогда в результате мы получим:

$$\boldsymbol{T'} = \boldsymbol{\Gamma}^{\circ}_{f,g}(\boldsymbol{T}) = \boldsymbol{\Gamma}^{\circ}_{f,g}(\boldsymbol{T}_0, \boldsymbol{T}_1) = (\boldsymbol{\Gamma}^{\circ}_{f} \cdot \boldsymbol{S} \cdot \boldsymbol{\Gamma}^{\circ}_{g})(\boldsymbol{T}_0, \boldsymbol{T}_1) = (\boldsymbol{S} \cdot \boldsymbol{\Gamma}^{\circ}_{g})(\boldsymbol{\Gamma}^{\circ}_{f}(\boldsymbol{T}_0, \boldsymbol{T}_1)) = (\boldsymbol{S} \cdot \boldsymbol{\Gamma}^{\circ}_{g})(\boldsymbol{T}_0, \boldsymbol{T}_1 \circ \boldsymbol{f}(\boldsymbol{T}_0)) =$$
$$= \boldsymbol{\Gamma}^{\circ}_{g}(\boldsymbol{S}(\boldsymbol{T}_0, \boldsymbol{T}_1 \circ \boldsymbol{f}(\boldsymbol{T}_0))) = \boldsymbol{\Gamma}^{\circ}_{g}(\boldsymbol{T}_1 \circ \boldsymbol{f}(\boldsymbol{T}_0), \boldsymbol{T}_0) = (\boldsymbol{T}_1 \circ \boldsymbol{f}(\boldsymbol{T}_0), \boldsymbol{T}_0 \circ \boldsymbol{g}(\boldsymbol{T}_1 \circ \boldsymbol{f}(\boldsymbol{T}_0))) = (\boldsymbol{T}'_0, \boldsymbol{T}'_1)$$

Рассмотрим младшую часть зашифрованного блока данных: $\boldsymbol{T}'_0 = \boldsymbol{T}_1 \circ \boldsymbol{f}(\boldsymbol{T}_0)$. Нетрудно заметить, что зависимость битов части $\boldsymbol{T}'_0$ блока $\boldsymbol{T'}$ от битов части $\boldsymbol{T}_1$ достаточно тривиальна и не удовлетворяет высказанным выше требованиям. Поэтому сконструированное нами криптопреобразование $\boldsymbol{\Gamma}^{\circ}_{f,g} = \boldsymbol{\Gamma}^{\circ}_{f} \cdot \boldsymbol{S} \cdot \boldsymbol{\Gamma}^{\circ}_{g}$ недостаточно сложно для того, чтобы можно было без натяжек назвать его криптографическим. Но оно может быть весьма простым способом распространено на произвольное число шагов $\boldsymbol{n}$: пусть заданы функции $\boldsymbol{f}_1,\ldots,\boldsymbol{f}_n$ отображающих на себя множество $N/2$-битовых блоков данных, и пара взаимно обратных бинарных операций "$\circ$" и "$\bullet$" в этом же самом множестве. Тогда шифрующее и расшифровывающее преобразования могут быть определены следующим образом:

$$\boldsymbol{\Gamma}^{\circ}_{f_1, f_2, \ldots, f_n} = \boldsymbol{\Gamma}^{\circ}_{f_1} \cdot \boldsymbol{S} \cdot \boldsymbol{\Gamma}^{\circ}_{f_2} \cdot \boldsymbol{S} \cdot \ldots \cdot \boldsymbol{S} \cdot \boldsymbol{\Gamma}^{\circ}_{f_n},$$
$$\boldsymbol{\Gamma}^{\bullet}_{f_n, \ldots, f_2, f_1} = \boldsymbol{\Gamma}^{\bullet}_{f_n} \cdot \boldsymbol{S} \cdot \ldots \cdot \boldsymbol{S} \cdot \boldsymbol{\Gamma}^{\bullet}_{f_2} \cdot \boldsymbol{S} \cdot \boldsymbol{\Gamma}^{\bullet}_{f_1}.$$

Индукцией по числу основных шагов $\boldsymbol{n}$ можно показать, что второе преобразование обратно первому:

$$\boldsymbol{\Gamma}^{\circ}_{f_1, f_2, \ldots, f_n} \cdot \boldsymbol{\Gamma}^{\bullet}_{f_n, \ldots, f_2, f_1} = \boldsymbol{I}.$$

Чтобы последовательно реализовать в шифре принцип перемешивания и рассеивания, целесообразно представить шифрующее преобразование в виде достаточно большого числа ($\boldsymbol{n}$) шагов каждый из которых представляет собой реализацию относительно несложного шифра. Так, в Российском стандарте шифрования число таких шагов равно 32, а в американском – 16, но сами шаги в нем несколько сложнее. Есть криптоалгоритмы подобной архитектуры с меньшим числом шагов $\boldsymbol{n}$, например – FEAL [7], для различных вариантов которого $\boldsymbol{n} = 4$ или $\boldsymbol{n} = 8$.

Как уже отмечалось, вид бинарных операций наложения гаммы "$\circ$" и "$\bullet$" не столь важен с точки зрения стойкости шифра, поэтому, как правило, берется самый простой для практической реализации вариант – операция ***побитового исключающего или*** "$\oplus$". Это позволяет реализовать прямую и обратную процедуру преобразования однотипным образом, они будут отличаться только порядком использования шифрующих функций $\boldsymbol{f}_1,\ldots,\boldsymbol{f}_n$. Для более полного рассеивания информации исходных данных криптопреобразование может быть дополнено начальной ($\boldsymbol{Y}_0$) и конечной ($\boldsymbol{Y}_1$) перестановками битов или другими простыми и очевидным образом обратимыми преобразованиями. В результате мы получим следующие шифрующие преобразования:

$$\boldsymbol{E}^{\boldsymbol{Y}_0, \boldsymbol{Y}_1}_{f_1, f_2, \ldots, f_n} = \boldsymbol{Y}_0 \cdot \boldsymbol{\Gamma}^{\oplus}_{f_1} \cdot \boldsymbol{S} \cdot \boldsymbol{\Gamma}^{\oplus}_{f_2} \cdot \boldsymbol{S} \cdot \ldots \cdot \boldsymbol{S} \cdot \boldsymbol{\Gamma}^{\oplus}_{f_n} \cdot \boldsymbol{Y}_1,$$
$$\boldsymbol{D}^{\boldsymbol{Y}_0, \boldsymbol{Y}_1}_{f_1, f_2, \ldots, f_n} = \boldsymbol{Y}_1^{-1} \cdot \boldsymbol{\Gamma}^{\oplus}_{f_n} \cdot \boldsymbol{S} \cdot \ldots \cdot \boldsymbol{S} \cdot \boldsymbol{\Gamma}^{\oplus}_{f_2} \cdot \boldsymbol{S} \cdot \boldsymbol{\Gamma}^{\oplus}_{f_1} \cdot \boldsymbol{Y}_0^{-1}.$$

Как обычно, через $\boldsymbol{Y}^{-1}$ обозначается обратное к $\boldsymbol{Y}$ преобразование. Чтобы сохранить однотипность прямого и обратного криптопреобразований, необходимо обеспечить выполнение условий $\boldsymbol{Y}_0^{-1} = \boldsymbol{Y}_1$, $\boldsymbol{Y}_1^{-1} = \boldsymbol{Y}_0$, то есть подстановки $\boldsymbol{Y}_0$ и $\boldsymbol{Y}_1$ должны быть взаимно обратными. Получаем следующие формулы для прямого и обратного криптопреобразований:

$$\widetilde{\boldsymbol{E}}^{\boldsymbol{Y}}_{f_1,f_2,\dots,f_n} = \boldsymbol{Y}\cdot\boldsymbol{\Gamma}^{\oplus}_{f_1}\cdot\boldsymbol{S}\cdot\boldsymbol{\Gamma}^{\oplus}_{f_2}\cdot\boldsymbol{S}\cdot\ldots\cdot\boldsymbol{S}\cdot\boldsymbol{\Gamma}^{\oplus}_{f_n}\cdot\boldsymbol{Y}^{-1},$$

$$\widetilde{\boldsymbol{D}}^{\boldsymbol{Y}}_{f_1,f_2,\dots,f_n} = \boldsymbol{Y}\cdot\boldsymbol{\Gamma}^{\oplus}_{f_n}\cdot\boldsymbol{S}\cdot\ldots\cdot\boldsymbol{S}\cdot\boldsymbol{\Gamma}^{\oplus}_{f_2}\cdot\boldsymbol{S}\cdot\boldsymbol{\Gamma}^{\oplus}_{f_1}\cdot\boldsymbol{Y}^{-1}.$$

Преобразование с дополнительной перестановкой битов обладает более высокой криптостойкостью по сравнению с аналогичным преобразованием без перестановки только в том случае, когда она является секретным элементом шифра. Действительно, если это не так, то первым действием криптоаналитика будет подвергнуть все имеющееся в его распоряжении блоки данных, как открытые, так и зашифрованные, этой перестановке $\boldsymbol{Y}$, и таким образом свести исходную задачу к задаче раскрытия шифра без перестановки. С этой точки зрения начальная и конечная битовые перестановки в алгоритме DES являются не более, чем украшениями и не оказывают заметного влияния на его криптостойкость.

Теперь вспомним про нарушение принципа Кирхгофа, заключающегося в использовании в качестве секретных элементов функций шифрования $\boldsymbol{f_i}$. Для того, чтобы наш алгоритм шифрования его не нарушал, слегка изменим схему использования функций $\boldsymbol{f_i}$ – сделаем их зависящими от секретного ключа $\boldsymbol{K}$: $\boldsymbol{f_i}(\boldsymbol{T}) = \boldsymbol{f_i}(\boldsymbol{T},\boldsymbol{K})$, где $\boldsymbol{f_i}$ –известная (открытая) функция, а $\boldsymbol{K}$ – секретный элемент шифра (ключ). Как правило, все функции $\boldsymbol{f_i}$ одинаковым образом используют преобразуемый блок данных $\boldsymbol{T}$ и различаются лишь схемой использования ключа $\boldsymbol{K}$, то есть можно записать: $\boldsymbol{f_i}(\boldsymbol{T},\boldsymbol{K}) = \boldsymbol{f}(\boldsymbol{T},\boldsymbol{\varphi_i}(\boldsymbol{K})) = \boldsymbol{f}(\boldsymbol{T},\boldsymbol{K_i})$, где через $\boldsymbol{K_i} = \boldsymbol{\varphi_i}(\boldsymbol{K})$ мы обозначили код, вырабатываемый из ключа $\boldsymbol{K}$ и используемый на $\boldsymbol{i}$-ом шаге шифрования, который мы назовем для краткости “шаговым ключом”.

Нам осталось обсудить последнюю деталь в построении блочного шифра. До сих пор мы не требовали, чтобы размер шифруемого блока $\boldsymbol{T}$ был постоянным, однако теперь нам придется это сделать по следующим причинам:

- Многие элементы шифра, такие, как перестановки битов и функции замены битовых групп в массиве данных предполагают, что преобразуемый блок имеет фиксированный размер. Хотя в принципе можно задавать правило построения таких элементов для блоков произвольного размера, применять это правило на практике было бы чрезвычайно неудобно.
- Если бы было возможно произвольно увеличивать размер шифруемого блока, это привело бы к ситуации, когда блок данных большого размера шифруется за один проход на ключе гораздо меньшего размера, и такая схема становится менее устойчивой к попыткам алгоритмического криптоанализа.

В силу указанных причин шифрованию по рассмотренной нами схеме должны подвергаться только блоки данных фиксированного размера, именно поэтому данный шифр называется блочным. При необходимости сообщение $\boldsymbol{T}$ разбивается на несколько блоков одинакового размера, которые шифруются независимо друг от друга:

$$\boldsymbol{T} = (\boldsymbol{T}_1,\boldsymbol{T}_2,\dots,\boldsymbol{T_n}),\ \boldsymbol{T'} = (\boldsymbol{T}_1',\boldsymbol{T}_2',\dots,\boldsymbol{T_n'}) = (\boldsymbol{E_K}(\boldsymbol{T}_1),\boldsymbol{E_K}(\boldsymbol{T}_2),\dots,\boldsymbol{E_K}(\boldsymbol{T_n})).$$

По этой причине данную схему шифрования называют шифром простой замены, так как в конечном итоге она сводится к замене одних значений блоков данных на другие. Для построенного нами алгоритма шифрования, очевидно, существует проблема, если размер шифруемого текста не кратен размеру блока криптоалгоритма. Этот, и ряд других вопросов будут обсуждены в следующем разделе.

Размер блоков шифрования определяется разработчиком шифра из условия достижения необходимой криптостойкости. Выбор недостаточного размера блоков сделает возможным криптоанализ на основе статистических закономерностей. С другой стороны, неоправданное увеличение размера блока сделает шифр громоздким и неудобным для применения, поэтому в данном вопросе необходимо искать компромисс. В подавляющем

большинстве известных автору шифров рассматриваемой архитектуры используется размер блока, равный 64 битам.

В заключение подведем некоторый итог наших изысканий: Для определения блочного шифра необходимо задать следующее:

1. числовые параметры алгоритма:
   - размер шифруемого блока данных $|T| = N$;
   - размер ключа $|K| = N_K$;
   - размер "шагового" ключа $|K_i| = N'_K$;
   - число шагов шифрования (раундов) n;
2. функцию шифрования f, принимающую в качестве аргумента и возвращающую в качестве значения $N/2$-битовые блоки данных. Функция шифрования зависит также от N'_K-битового секретного блока данных – "шагового" ключа;
3. набора функций φ_i, $1 \le i \le n$ для выработки "шаговых" ключей K_i из исходного ключа шифрования K: $K_i = \varphi_i(K)$;
4. перестановки битов Y в N-битовом блоке данных;

Зашифровывающее и расшифровывающее криптографические преобразования строятся как композиции описанных выше простых шагов Γ_i, S, Y:

$$\tilde{E}_K^Y = Y \cdot \Gamma_1 \cdot S \cdot \Gamma_2 \cdot S \cdot \ldots \cdot S \cdot \Gamma_n \cdot Y^{-1},$$

$$\tilde{D}_K^Y = Y \cdot \Gamma_n \cdot S \cdot \ldots \cdot S \cdot \Gamma_2 \cdot S \cdot \Gamma_1 \cdot Y^{-1}.$$

Здесь Γ_i и S обозначают соответственно шаг шифрующего преобразования и перестановку старшей и младшей половин шифруемого блока:

$$\Gamma_i(T) = \Gamma_i(T_0, T_1) = (T_0, T_1 \oplus f(T_0, K_i)) = (T'_0, T'_1) = T',$$
$$S(T) = S(T_0, T_1) = (T_1, T_0).$$

Как было отмечено выше, данная схема может быть обобщена на случай разделения шифруемого блока данных T на два неодинаковых по размеру подблока. Кроме того, могут быть добавлены дополнительные преобразования данных в начале, конце, или на промежуточных этапах криптопреобразования, и также вместо функции ***побитового исключающего или*** можно использовать какую-либо другую для наложения гаммы на части шифруемого блока. Шифр, который мы с вами только что построили, называется шифром простой замены, так как по сути его действие заключается в том, что шифруемые блоки данных заменяются на блоки шифртекста независимо от других блоков.

6. Российский стандарт шифрования.

Настало время рассказать об алгоритме шифрования, являющемся Российским стандартом. Этот стандарт имеет обозначение ГОСТ 28147-89 из которого явствует, что он был принят в 1989 году. Его основу составляют процедуры зашифрования и расшифрования по алгоритму простой замены 64-битовых блоков данных, и уже с их использованием строятся другие алгоритмы шифрования – как это сделать, обсуждается в следующих разделах. В настоящей статье рассматривается только криптоалгоритм ГОСТ 28147-89 [6], описание некоторых других шифров с подобной архитектурой можно найти в литературе [2,7,8], поэтому автор решил не останавливаться на них. Сейчас мы рассмотрим ГОСТ по намеченной выше схеме:

1. определим числовые характеристики алгоритма:
   - размер шифруемого блока равен 64 битам: $|T| = 64$;
   - размер ключа равен 256 битам $|K| = 256$;
   - на каждом шаге алгоритма используется 32-битовый ключевой элемент: $|K_i| = 32$;

- число повторений основного шага (число раундов) $n = 32$;

2. функция шифрования определяется следующим образом:
   - исходными данными для функции шифрования являются 32-битовые элемент данных T и "шаговый" ключ K_i;
   - блок данных T и "шаговый" ключ K_i складываются по модулю 2^{32}: $S = (T + K_i) \bmod 2^{32}$;
   - полученный 32-битовый блок данных интерпретируется как массив из восьми 4-битовых групп $S = (S_1, S_2, ..., S_8)$, $|S_i| = 4$, и в каждой группе выполняется подстановка с помощью соответствующего узла замен: $S'_i = h_{i,S_i}$. В результате мы получаем следующий блок данных: $S' = (S'_1, S'_2, ..., S'_8) = (h_{1,S_1}, h_{2,S_2}, ..., h_{8,S_8})$;
   - результат предыдущего шага вращается на 11 битов влево, то есть в сторону старших разрядов: если $S' = b_{31}b_{30}...b_{21}b_{20}...b_1b_0$, то $T' = R^{\leftarrow}_{11}(S') = b_{20}...b_1b_0b_{31}b_{30}...b_{21}$;
   - полученный блок данных T' является значением функции шифрования: $T' = f_i(T, K)$;

3. при шифровании используется следующая секретная (ключевая) информация:
   - **ключ** – 256-битовый массив информации, структурированный в массив из восьми 32-битовых элементов, которые мы пронумеруем в порядке их использования в цикле шифрования: $K = (k_1, k_2, ..., k_8)$, $|K| = 256$, $|k_i| = 32$;
   - **таблица замен** – набор из восьми – по числу битовых групп, на которые разбивается 32-битовый блок данных при вычислении функции шифрования – **узлов замен**: $H = (H_1, H_2, ..., H_8)$. Каждый **узел замен** представляет из себя подстановку в 16-элементном множестве всех возможных значений 4-битовых кодов, и таким образом может быть представлен в виде массива из 16 различных 4-битовых чисел: $H_i = (h_{i,0}, h_{i,1}, ..., h_{i,15})$, $|h_{i,j}| = 4$, $0 \le h_{i,j} \le 15$, для любых i, j, k $(j \ne k)$, должно выполняться условие $h_{i,j} \ne h_{i,k}$. Размер каждого узла замен равен $|H_i| = 8 \cdot |h_{i,j}| = 64$ бита, а размер всей таблицы замен $|H| = 8 \cdot |H_i| = 8 \cdot 64 = 512$ бит или 64 байта;

4. определим порядок использования ключа при шифровании, то есть зададим функцию φ_i выработки "шагового" ключа из исходного ключа шифрования K: $K_i = \varphi_i(K)$. В качестве "шаговых" ключей K_i в ГОСТе берутся определенные элементы k_j ключа K: $K_i = k_j$. Однако таких элементов всего 8, а шагов в цикле шифрования 32, значит, необходимо задать функцию $\pi(i)$, отображающую 32-элементное множество шагов в цикле шифрования в 8-элементное множество частей ключа: $K_i = k_{\pi(i)}$, $1 \le i \le 32$, $1 \le \pi(i) \le 8$ Зададим эту функцию в табличном и формульном виде:

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$\pi(i)$	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
i	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
$\pi(i)$	1	2	3	4	5	6	7	8	8	7	6	5	4	3	2	1

$$\pi(i) = \begin{cases} i, & 1 \le i \le 8; \\ i - 8, & 9 \le i \le 16; \\ i - 16, & 17 \le i \le 24; \\ 33 - i, & 25 \le i \le 32; \end{cases}$$

Иными словами, в ходе цикла зашифрования элементы ключа используются последовательно друг за другом, при этом ключ "просматривается" четыре раза – первые три в прямом направлении, четвертый – в обратном. Читатели, хорошо усвоившие материал предыдущих разделов, без труда сообразят, что в цикле расшифрования элементы ключа используются в обратном по отношению к циклу зашифрования порядке, то есть сначала ключ просматривается один раз в прямом направлении, затем три раза в обратном.

Теперь поговорим о таблице замен. Она является аналогом блоков подстановки (S-boxes) американского стандарта, но, в отличие от последнего, не фиксирована и является секретным элементом шифра. Следует отметить, что алгоритм шифрования обратим при

любом заполнении узлов замен, даже если они и не задают правильные подстановки в 16-элементном множестве, то есть содержат повторяющиеся элементы замены: $h_{i,j} = h_{i,k}$ при некоторых $i, j, k\ (j \neq k)$. Однако такая замена ведет к потере информации о заменяемом значении, и, как следствие, к снижению криптостойкости шифра, поэтому возможность этого не предусматривается в ГОСТе.

Отметим, что основным секретным элементом российского шифра является ключ. Алгоритм должен оставаться криптостойким даже в случае раскрытия таблицы замен, что, однако, имеет место не при всех ее возможных значениях. Существование "слабых" таблиц замен, то есть таблиц, при использовании которых криптостойкость шифра существенно снижается, не вызывает сомнения – примером здесь может служить тривиальная таблица, при использовании которой всякое значение заменяется на него же самого. Однако в широких кругах специалистов – криптографов доподлинно неизвестно, как много есть подобных таблиц и существует ли критерий, позволяющий для конкретной таблицы однозначно определить, является ли она слабой или нет. Очевидно, что если такие критерии и существуют, они тщательно засекречены, никакие данные по этому вопросу, равно как и по вопросу качества ключей, не публиковались в открытой печати.

Однако дело с выбором ключевых данных обстоит не так уж плохо – с вероятностью, незначительно отличающейся от единицы, случайным образом выбранные ключевые данные не приведут к катастрофическому снижению криптостойкости шифра, и стоимость его раскрытия по-прежнему останется достаточно высокой и вряд ли окажется по зубам какой-либо коммерческой структуре, пусть даже и очень крупной. Ну а госслужбы при необходимости смогут получить всю интересующую их информацию и не прибегая к дешифрованию. По этой причине для обычного пользователя, стоимость защищаемой информации которого не превышает суммы порядка нескольких сотен тысяч долларов США, вполне достаточно будет хорошего статистического качества ключевой информации, заключающегося в следующем:

- ключ должен быть массивом из 256 независимых битов с равновероятными значениями;
- таблица замен должна быть набором из восьми независимых узлов, каждый из которых является случайной подстановкой в 16-элементном множестве;

В добавлении следует сделать следующее замечание: если функции, выражающие биты результата замены через биты исходного блока получаются достаточно простыми, это будет ослаблять шифр. Так, безобидный на первый взгляд узел замен $H_i = (9, 8, 3, 10, 12, 13, 7, 14, 0, 1, 11, 2, 4, 5, 15, 6)$ на самом деле является слабым, так как оставляет второй и третий биты группы неизменными, что становится очевидным, если записать его в табличной форме с двоичными значениями аргумента и функции:

S	0000	0001	0010	0011	0100	0101	0110	0111
H(S)	1001	1000	0011	1010	1100	1101	0111	1110
S	1000	1001	1010	1011	1100	1101	1110	1111
H(S)	0000	0001	1011	0010	0100	0101	1111	0110

Кроме этого, таблица замен может содержать обходные пути других типов, позволяющие расшифровать сообщение более эффективным образом, чем полным перебором по возможным значениям ключа – но это уже тема не для журнальной статьи.

7. Недостатки режима простой замены.

Использование блочного шифра в режиме простой замены имеет ряд недостатков, отражающихся на стойкости и удобстве использования шифра. Первый и самый серьезный недостаток простой замены заключается в том, что в этом режиме зашифрование одинаковых блоков исходного текста дает в результате идентичные блоки шифртекста, что

облегчает задачу криптоаналитика. Действительно, на основе только шифрованных данных он может сделать некоторые заключения о свойствах исходного текста, что, конечно же, не является достоинством шифра.

Приведем типичный пример: пусть зашифрованию подвергается информация на гибком магнитном диске. Редко встречается ситуация, когда данные занимают весь диск, как правило значительная его часть остается свободной. Часть дискеты, никогда не содержавшая полезной информации, обычно заполнена фиксированными кодами, записанными туда при форматировании, и при ее зашифровании мы получим фиксированные блоки шифртекста. Тогда, анализируя данную дискету, криптоаналитик сможет определить с точностью до нескольких байтов размер массива полезной информации, содержащейся на ней, что в ряде случаев, например, если формат и содержание сообщения связаны с размером соответствующего файла, может облегчить ему задачу дешифрования.

Можно предложить модификацию схемы шифрования по алгоритму простой замены, устраняющую данный недостаток – для этого перед зашифрованием сообщения надо выполнить его рандомизацию. Это действие заключается в том, что блоки исходного текста модифицируются индивидуальным образом, например, комбинируются с данными, вырабатываемыми датчиком псевдослучайных чисел (ПСЧ) с помощью некоторой бинарной операции "$\circ$", имеющей обратную операцию "$\bullet$". Зашифрование и расшифрование будет выполняться согласно следующим уравнениям:

$$T_i' = E_K(T_i \circ \Gamma_i),$$

$$T_i = D_K(T_i') \bullet \Gamma_i.$$

где Γ_i – блоки данных, вырабатываемые датчиком ПСЧ. Как уже отмечалось ранее, в качестве операции наложения/снятия гаммы наиболее всего подходит процедура ***побитового исключающего или***. Период повторения последовательности ПСЧ $\{\Gamma_i\}$, вырабатываемых датчиком, должен превышать максимально возможное число блоков в сообщении.

Однако данный протокол несостоятелен, если криптоаналитик обладает возможностью анализа на основе выбранного открытого текста, то есть если он может получить любое выбранное им открытое сообщение зашифрованным на том же ключе и с использованием тех же элементов Γ_i, что и дешифруемый текст. Действительно, предположим, что криптоаналитик располагает такой возможностью. Тогда он может подвергнуть зашифрованию массив, состоящий из одних блоков–заполнителей и сравнить полученный результат с анализируемым шифртекстом:

$U_i = U$, где U – вероятный блок–заполнитель,

$U_i' = E_K(U_i \circ \Gamma_i)$.

Если выполняется условие $U_i' = T_i'$, это означает, что $T_i = U$, и наш криптографический протокол не выполняет возложенных на него задач.

Чтобы избавиться от указанного недостатка, протокол шифрования должен обеспечить уникальность последовательности элементов гаммы Γ_i. Как это можно осуществить? Как правило, все алгоритмы выработки ПСЧ имеют некоторый набор числовых параметров, однозначно определяющий получаемую выходную последовательность $\{\Gamma_i\}$. Это позволяет обеспечить ее уникальность для различных процессов шифрования одним из следующих двух способов:

- сделать числовые параметры алгоритма выработки ПСЧ Γ_i секретными. Этот способ нельзя назвать хорошим, так как он приводит к возрастанию общего объема секретной ключевой информации, то есть, по своей сути, предполагает использование дополнительного алгоритма шифрования.
- обеспечить уникальность используемого набора числовых параметров соответствующим построением шифрователя. Этот способ также нельзя считать приемлемым, так как шифрователь в этом случае должен содержать некоторый блок, обеспечивающий получение уникального вектора начальных

параметров для выработки Γ_i, что усложнит его структуру. Кроме того, в этом случае придется запретить шифрование по алгоритму простой замены без использования элементов Γ_i, так как если этого не сделать, криптоаналитик может выполнить комбинирование блока–заполнителя с элементами Γ_i самостоятельно, подвергнув зашифрованию последовательность блоков $\{U_i\}$, где $U_i = U \circ \Gamma_i$.

Как видим, данный способ рандомизации исходного сообщения сам по себе порождает больше проблем, чем позволяет решить, и поэтому его использование, хотя и возможно в принципе, не получило сколько-нибудь значительного распространения. Более того, в дальнейшем мы увидим, что при шифровании по методу гаммирования возникают сходные проблемы.

Второй недостаток, имеющий место при использовании блочного шифра в режиме простой замены, обусловлен проблемой неполных блоков, или “хвостов”. Так как в данном режиме криптопреобразованию подвергаются лишь блоки фиксированного размера, возникает проблема, если размер шифруемого сообщения не кратен размеру блока используемого криптоалгоритма. Суть проблемы заключается в том, чем и как дополнять “хвосты” до полноразмерных блоков, чтобы это было удобно в использовании и не снижало криптостойкости шифра. Рассмотрим возможные пути решения данной проблемы:

- можно дополнить “хвост” фиксированными данными – например, нулями. Это, однако, существенно снизит криптостойкость последнего блока, так как криптоаналитику, обладающему информацией о способе дополнения “хвоста”, потребуется выполнить перебор по множеству возможных значений этого блока, гораздо меньшему, чем полное пространство значений блока.
- можно дополнять “хвосты” данными из полных блоков. В целом это неплохое решение, но оно не работает, если неполный блок – единственный, то есть если длина сообщения меньше размера блока шифра. Кроме того, используя данный подход, мы рано или поздно столкнемся с ситуацией, когда для дополнения “хвоста” будут использованы фиксированные части сообщения, обладающие недостаточной неопределенностью для криптоаналитика. Так, например, многие сообщения начинаются с грифа, который может принимать всего два – три возможных значения, а различных форм его записи в текстовом виде может быть максимум несколько десятков вариантов. Если для дополнения “хвоста” используется часть такого блока, то складывается ситуация, подобная рассмотренной в предыдущем пункте – “хвост” может стать легкой добычей криптоаналитика.
- применение для дополнения “хвоста” отдельного постоянного секретного элемента не подходит по той же самой причине, что и первый способ – использование данного элемента по частям делает его уязвимым к криптоанализу. Такая схема оказывается беззащитной перед анализом на основе выбранного открытого текста. Кроме того, данный способ увеличивает общий объем секретной (ключевой) информации, что весьма нежелательно.
- пожалуй, наилучший возможный способ заключается в том, чтобы использовать для дополнения “хвоста” данные с аппаратного датчика случайных чисел. Здесь нужен датчик, вырабатывающий действительно случайные числа, имеющие высокое статистическое качество. По этой причине различные программные датчики ПСЧ здесь не подходят. В силу этого часто такой способ оказывается неприемлемым по экономическим соображениям, так как требует наличия на каждом компьютере-шифрователе весьма дорогостоящей аппаратной компоненты.

Как мы видим, вторая проблема шифрования в режиме простой замены также не имеет эффективного и вместе с тем экономичного решения. Кроме того, эта проблема создает еще одну, чисто техническую сложность – после зашифрования в режиме простой замены все разряды полученного блока станут значащими, а это значит, что вместе с шифртекстом необходимо теперь хранить размер (число битов или байтов) последнего, неполного блока исходного текста, что приводит к изменению его размера, а это в некоторых случаях нежелательно.

Третий недостаток шифрования в режиме простой замены заключается в его неустойчивости перед модификацией сообщения, заключающейся в перестановке блоков шифртекста. Действительно, если мы вносим изменения в блок шифртекста “наобум” то скорее всего после расшифрования он окажется “запорченным”, то есть его содержимое будет бессмысленным. Причина этого заключается в том, что все естественные и искусственные языки имеют огромную избыточность, а изменения, внесенные в блок

шифртекста случайным, то есть непредсказуемым для нас образом, влияют на соответствующие расшифрованные данные и вероятность получить имеющий смысл результат крайне мала. Другое дело, если мы просто меняем местами, удаляем или дублируем блоки шифрованного сообщения. В этом случае мы можем получить результат, имеющий некоторый смысл, и такое нарушение целостности данных может оказаться незамеченным, если не принять специальных мер.

В силу изложенных выше соображений использование блочного шифра в режиме простой замены может быть рекомендовано только для шифрования небольших по объему массивов данных, размер которых кратен размеру блока используемого блочного криптоалгоритма и которые не содержат повторяющихся блоков. Этому набору требований вполне удовлетворяют лишь массивы ключевой информации. Именно поэтому DES не рекомендует, а Российский стандарт ГОСТ 28147-89 прямо запрещает использовать шифрование в режиме простой замены для данных, не являющихся ключевыми.

8. Гаммирование.

Предложенная в предыдущем разделе процедура шифрования по методу простой замены с использования датчика псевдослучайных чисел может быть слегка изменена, что освободит ее от ряда недостатков. Зашифрованию по алгоритму простой замены следует подвергать сами блоки, вырабатываемые датчиком ПСЧ, и уже их использовать в качестве гаммы, комбинируемой с блоками данных с помощью бинарных операций "$\circ$" и "$\bullet$":

$T_i' = T_i \circ E_K(\Gamma_i)$ (зашифрование),

$T_i = T_i' \bullet E_K(\Gamma_i)$ (расшифрование),

где блоки гаммы Γ_i размером $|\Gamma_i| = |T| = N$ вырабатываются с помощью датчика псевдослучайных чисел, являющегося, как правило, рекуррентным алгоритмом:
$\Gamma_{i+1} = \gamma(\Gamma_i)$, где γ – некоторая функция, реализуемая алгоритмически.

Какие же требования должны быть предъявлены к ПДПСЧ, чтобы обеспечить достаточное качество шифра? Прежде всего отметим, что не требуется ни криптостойкости, ни хороших статистических параметров для вырабатываемой последовательности, так как они обеспечиваются в дальнейшем зашифрованием блоков по алгоритму простой замены.

1. Необходимо, чтобы период повторения генерируемой последовательности ПСЧ $\{\Gamma_i\}$ был достаточно большим, лучше – максимально возможным. По крайней мере, он должен превосходить наибольшее возможное количество блоков в шифруемом массиве данных. Если исходить только из этого требования, наиболее подходящим было бы использование простейшего из возможных датчиков, определяемого следующим соотношением: $\Gamma_{i+1} = (\Gamma_i + 1) \bmod 2^N$, или попросту $\Gamma_i = i \bmod 2^N$. Но дело в том, что этот датчик ПСЧ не обеспечивает выполнение второго достаточно важного требования к ПДПСЧ:
2. Необходимо, чтобы соседние или близкие по расположению элементы последовательности $\{\Gamma_i\}$ достаточно, то есть как минимум, в нескольких битах, отличались друг от друга. Было бы крайне желательно, чтобы различия между ними были в каждом байте. Для простейшего же генератора, предложенного в пункте 1, половина пар соседних значений отличается лишь в одном бите: $\Gamma_{2i+1} = \Gamma_{2i} \oplus 1$.

Смысл первого требования становится очевидным исходя из того, что метод гаммирования по своей сути требует одноразовой гаммы, иначе он легко вскрывается по алгоритмической линии. Если же период повторения вырабатываемой гаммы недостаточно велик, различные части одного и того же длинного сообщения могут оказаться зашифрованными с помощью одинаковых участков гаммы. Это на много порядков снижает стойкость шифра, делая его легкой добычей криптоаналитика. Второе требование является

менее очевидным, и, вообще говоря, имеет место только для шифров вполне определенных архитектур, в которых шаг шифрования является комбинацией нескольких сравнительно простых преобразований, в ходе каждого из которых различия в шифруемых блоках данных увеличиваются весьма незначительно. Поэтому, если мы подвергнем шифрованию два блока, отличающиеся только в единственном двоичном разряде, серьезные различия между ними появятся только через несколько таких простых шагов, что несколько снижает стойкость шифра и облегчает задачу криптоаналитика. Конечно, в случае ГОСТа это снижение не так велико, но разработчики алгоритма решили подстраховаться. Датчик ПСЧ, входящий в Российский стандарт на шифрование, предполагает независимую обработку старшей и младшей половин вырабатываемого блока $\Gamma_i = (\Gamma_{i,0}, \Gamma_{i,1})$:

$\Gamma_{i+1,0} = (\Gamma_{i,0} + C_0) \bmod 2^{32}$,
$\Gamma_{i+1,1} = (\Gamma_{i,1} + C_1 - 1) \bmod (2^{32} - 1) + 1$,

где константы C_0 и C_1 имеют следующие значения в 16-ричной системе счисления: $C_0 = 1010101_{16}$, $C_1 = 1010104_{16}$.

Такой ДПСЧ ненамного сложнее простейшего, однако удовлетворяет всем перечисленным выше требованиям: период повторения вырабатываемой с его помощью последовательности достаточно велик и близок к максимальному, и получаемые с его помощью блоки различаются как минимум в одном бите в каждом байте.

Отметим, что данный алгоритм прост как в аппаратной, так и в программной реализации, – вспомним, что все целочисленные вычисления в ЭВМ ведутся по модулю 2^N, где N-разрядность машинного слова. Первое из двух соотношений реализуется на всех без исключения 32-разрядных типах процессоров за одну команду, второе, хотя и выглядит более грозно, чем первое, на самом деле реализуется ненамного сложнее него. Это становится очевидным, если его записать в следующей форме:

$$\Gamma_{i+1,1} = \begin{cases} \Gamma_{i,1} + C_1 = (\Gamma_{i,1} + C_1) \bmod 2^{32}, & \Gamma_{i,1} + C_1 < 2^{32} \\ \Gamma_{i,1} + C_1 - 2^{32} + 1 = (\Gamma_{i,1} + C_1) \bmod 2^{32} + 1, & \Gamma_{i,1} + C_1 \geq 2^{32} \end{cases}$$

На всех известных автору 32-разрядных процессорах рекуррентные соотношения для выработки очередного элемента реализуется всего за три машинные команды:

```
add     G0,01010101h
add     G1,01010104h
adc     G1,0
```

Команды даны в мнемонике фирмы Intel для производимого ею процессора iAPX386, G0,G1 – 32-х битовые элементы данных – регистры или двойные слова в памяти, соответственно младшая и старшая половины 64-битового блока данных, из которого после зашифрования по алгоритму простой замены получается блок гаммы.

Очевидно, что при использовании режима гаммирования, как и в любом другом случае, когда используется рекуррентный датчик ПСЧ, необходим дополнительный элемент данных Γ_0, первый в рекуррентной последовательности $\{\Gamma_i\}$. Чтобы осложнить некоторые возможные способы криптоанализа, в Российском стандарте шифрования элемент Γ_0 получается зашифрованием по алгоритму простой замены блока данных S такого же размера $|S| = |\Gamma_0| = 64$, называемого синхропосылкой или начальным заполнением: $\Gamma_0 = E_K(S)$. Для шифрования используются блоки гаммы, начиная с Γ_1. Синхропосылка должна быть известна принимающей сообщение стороне, но требование уникальности гаммы шифрования, однозначно определяемой комбинацией **синхропосылка + ключ**, приводит к необходимости использовать уникальную синхропосылку для каждого передаваемого сообщения. Так как ее секретность не требуется для обеспечения стойкости сообщения, синхропосылка обычно передается в открытом виде вместе с зашифрованным сообщением.

Очевидно, что гаммирование является вариантом потокового шифрования, то есть в настоящем разделе мы построили механизм, позволяющий создавать потоковые шифры на основе блочных. Соответственно этому предложенный нами алгоритм шифрования обладает всеми преимуществами и недостатками потоковых шифров. Прежде всего отметим, что гаммирование свободно от многих недостатков простой замены.

Во-первых, одинаковые блоки исходного текста после зашифрования дадут различные блоки шифртекста, что не облегчает задачу криптоанализа на основе только шифртекста.

Во-вторых, как и во всех поточных шифрах, отсутствует проблема неполного блока данных. Действительно, при зашифровании последнего в сообщении блока данных $\boldsymbol{T_n}$ неполного размера $|\boldsymbol{T_n}| < \boldsymbol{N}$ из выработанного для этого блока гаммы мы можем взять фрагмент такого же размера. Этот подход очевиден, если операция наложения гаммы является побитовой, но может быть применен и в иных случаях. Важно, что при этом мы получим блок шифртекста того же размера, что и блок исходных данных. Таким образом, зашифрование по методу гаммирования не изменяет размер шифруемых данных, что в ряде случаев бывает важным.

В третьих, факты любых манипуляций как над частями криптоблоков, так и над целыми блоками, такие, как их перестановка, удаление, дублирование, становятся очевидными после расшифрования – это проявляется тем в большей степени, чем больше избыточности содержится в шифруемом сообщении. Особенно ярко этот эффект проявляется для текстовых файлов – если таким манипуляциям подвергнуть шифрованное сообщение, составленное на одном из естественных языков, то после расшифрования мы получим полную белиберду.

Режиму гаммирования в гораздо большей степени, чем режиму простой замены, присуще свойство нераспространения ошибки. Если во втором случае ошибка распространяется в пределах всего измененного криптографического блока непредсказуемым для злоумышленника образом, то в первом случае ее влияние на соответствующий блок открытого текста сравнительно легко прогнозируется, что делает возможной направленную модификацию блоков шифртекста. Если модификации подвергнуть бит в массиве данных, зашифрованном гаммированием с использованием операции ***побитового исключающего или***, то после расшифрования измененным окажется лишь тот же самый бит в открытом тексте.

Указанное свойство гаммирования довольно неприятно, так как по своей сути означает, что злоумышленник, воздействуя только на шифртекст, может по своему усмотрению менять любые разряды в соответствующем открытом тексте. Насколько это может быть опасно, поясним на следующем примере: пусть почтальон – злоумышленник передает шифрованное сообщение, которое он не может расшифровать, однако может внести в него произвольные изменения. Таким почтальоном может быть, например, работник филиала фирмы, обслуживающий коммуникационную ЭМВ, предназначенную для передачи сообщений в головное отделение фирмы. Пусть он передает некий документ, составленный в виде текстового файла, зашифрованного в режиме гаммирования, в котором, как он знает, с 13 позиции начинается запись некоторого числа, например, суммы его премиального вознаграждения. Вполне может оказаться, что ему известна вся сумма или, хотя бы ее некоторая часть – предположим, что запись числа начинается с символа ‘1’, и злоумышленник это знает. Тогда для него не составит труда переправить эту цифру на другую, например – на ‘9’. Все что для этого требуется – это заменить 13-й байт сообщения $\boldsymbol{B}_{13}$ на новое значение, вычисляемое следующим образом: $\boldsymbol{B}'_{13} = \boldsymbol{B}_{13} \oplus$ ‘1’ $\oplus$ ‘9’. Через ‘1’ и ‘9’ мы обозначили коды цифр 1 и 9 соответственно в той системе кодировки, в которой подготовлен текст сообщения. Если это ASCII, то достаточно инвертировать всего один бит сообщения: $\boldsymbol{B}'_{13} = \boldsymbol{B}_{13} \oplus 8$. После расшифрования получатель получит сообщение, в котором

вместо верной цифры 1 стоит цифра 9, и больше никаких изменений нет! Очевидно, что данная подмена не может быть обнаружена, если в передаваемом массиве данных нет никакой дополнительной информации, кроме зашифрованного исходного сообщения. Приведенный пример еще раз иллюстрирует тот факт, что обеспечение секретности сообщения (его шифрование) само по себе не обеспечивает его аутентичности, то есть подлинности. В чем же причина такой особенности гаммирования? Дело в том, что этот режим не обеспечивает весьма желательного для всех шифров перемешивания битов сообщения, то есть влияния одного бита исходного текста на несколько битов шифртекста.

Другая особенность гаммирования, осложняющая механизм его использования – это необходимость обеспечения уникальности гаммы. Так как гамма однозначно определяется ключом и синхропосылкой, эта пара должна быть уникальна для каждого шифрованного документа, что приводит при постоянстве ключа к необходимости назначения каждому сообщению уникальной синхропосылки. Поэтому любой протокол шифрования (алгоритм шифрования более высокого уровня) полностью неустойчив к анализу на основе произвольно выбранного открытого текста, если криптоаналитик может по своему усмотрению назначать синхропосылку или использовать для шифрования сообщения метод простой замены.

9. Гаммирование с обратной связью.

При синтезе алгоритма блочного шифрования мы уже сталкивались с идеей вырабатывать гамму для шифрования очередного блока данных с использованием одного или нескольких предыдущих блоков данных.

$\Gamma_{i+1} = f(T_i)$, или, более обще:
$\Gamma_{i+1} = f(T_1, T_2, ..., T_i)$.

У этой схемы шифрования, так же как и у обычного гаммирования, есть проблема получения первого элемента гаммы. Способ ее решения вполне традиционен: исходный текст дополняется фиктивным несекретным блоком T_0, единственное назначение которого – быть основой для выработки первого блока гаммы: $\Gamma_1 = f(T_0)$.
Как и при гаммировании без ОС, этот блок должен быть частью шифрованного сообщения:

$T' = (T_0, T'_1, T'_2, ..., T'_n)$.

Если мы решили вырабатывать блоки гаммы из блоков исходного текста, то нам необходимо скрыть при этом любые статистические закономерности этого текста. Самый очевидный и уже опробованный путь для этого – использовать шифрующее преобразование, то есть алгоритм простой замены:

$\Gamma_i = E_K(T_i)$,
$T'_i = T_i \oplus \Gamma_i = T_i \oplus E_K(T_{i-1})$.

Но предложенный подход в его первоначальном виде обладает многими недостатками алгоритма простой замены, для преодоления которых мы, собственно, и затеяли различные усовершенствования схемы шифрования. Обратим внимание на то, что при его использовании блок шифртекста зависит только от соответствующего и предшествующего ему блоков исходного текста:

$T'_i = T_i \oplus E_K(T_{i-1}) = f(T_i, T_{i-1})$.

Это означает, что при зашифровании массива одинаковых блоков данных соответствующие блоки шифртекста, начиная со второго из них, будут также идентичны – это становится очевидным из рассмотрения уравнений зашифрования:

$T'_i = f(T_i, T_{i-1})$,
$T'_{i+1} = f(T_{i+1}, T_i)$,
$T_{i-1} = T_i = T_{i+1} \rightarrow T'_i = T'_{i+1}$.

Как видим, по данному свойству предложенный нами алгоритм шифрования ничуть не лучше простой замены. Однако его очень легко можно усовершенствовать, если для выработки гаммы использовать блоки шифртекста, а не соответствующего открытого текста:

$\Gamma_i = E_K(T'_{i-1})$,
$T'_i = T_i \oplus \Gamma_i = T_i \oplus E_K(T'_{i-1})$ (зашифрование),
$T_i = T'_i \oplus \Gamma_i = T'_i \oplus E_K(T'_{i-1})$ (расшифрование).

Теперь каждый блок шифртекста зависит не только от соответствующего, но и от всех предшествующих блоков исходного текста:

$T'_i = T_i \oplus E_K(T'_{i-1}) = f(T_i, T'_{i-1})$, но
$T'_{i-1} = T_{i-1} \oplus E_K(T'_{i-2}) = f(T_{i-1}, T'_{i-2})$, поэтому
$T'_{i-1} = f(T_i, T'_{i-1}) = f(T_i, T_{i-1}, T'_{i-2}) = \ldots = f(T_i, T_{i-1}, \ldots, T_1, T_0)$, где
T_0 – синхропосылка.

Выполненное нами усовершенствование делает гаммирование с обратной связью свободным от недостатков простой замены. Действительно, как внесение изменений в блоки шифртекста, так и манипулирование целыми блоками приводит к непредсказуемым для злоумышленника изменениям в расшифрованном тексте, в результате чего он теряет возможность оказывать такие воздействия целенаправленно.

Как мы показали ранее, на каждый блок шифртекста при зашифровании оказывают влияние все предшествующие блоки исходного текста и синхропосылка:

$T'_i = f(T_i, T_{i-1}, \ldots, T_1, T_0)$.

Несколько иным, хотя весьма похожим образом на блоки открытого текста, получаемые при расшифровании, влияют блоки шифртекста. Для уяснения характера этого влияния еще раз запишем уравнения расшифрования для двух смежных блоков данных:

$T_i = T'_i \oplus E_K(T'_{i-1})$,
$T_{i+1} = T'_{i+1} \oplus E_K(T'_i)$.

Из этих уравнений становится ясно, что каждый блок шифртекста при расшифровании оказывает влияние лишь на два блока открытого текста: на соответствующий ему блок, и на следующий за ним блок. Причем в первом случае это влияние носит такой же характер, как и при использовании обычного гаммирования, а во втором – как при простой замене. Таким образом, если злоумышленник внесет изменения в блок шифртекста, они отразятся на текущем и следующем блоках сообщения.

Проявление этого влияния удобно пояснить на примере из предыдущего раздела: если сообщение было зашифровано по методу гаммирования с обратной связью и в него были внесены указанные в примере изменения, то после расшифрования получатель увидит следующую картину:

- в измененном блоке все также, как и при обычном гаммировании – вместо верной цифры один стоит навязанная злоумышленником девятка, и больше никаких изменений нет;
- в блоке, следующем за измененным, картина такая же, как и при простой замене – блок представляет собой случайную бессмысленную комбинацию битов;
- все прочие блоки остались неизменными;

Таким образом, режим гаммирования с обратной связью имеет те же преимущества, что и обычное гаммирование, и вместе с тем свободен от его недостатков. Гаммирование с ОС устойчиво к перестановкам блоков и к направленным модификациям шифртекста, если только модифицируется не последний блок. Иными словами, злоумышленник, вносящий изменения в шифртекст, не может точно рассчитать их влияние на открытый текст, если, конечно, не обладает секретным ключом. С другой стороны, для этого режима как для частного случая гаммирования, отсутствует проблема “хвоста” – последнего неполного блока данных и в отличие от гаммирования без ОС не так остра проблема уникальности

синхропосылки. Поясним последнее замечание: действительно, при обычном гаммировании вырабатываемая гамма однозначно определяется синхропосылкой и ключом: $\Gamma_i = f(i, T_0, K)$, тогда как при гаммировании с ОС она зависит еще и от шифруемых данных: $\Gamma_i^{oc} = f(i, T_0, T_1, ..., T_{i-1}, K)$. Поэтому, если два сообщения, не содержащих одинаковых блоков, зашифрованы с использованием одного и того же ключа и синхропосылки, то равенство $T_i' \oplus \tilde{T}_i' = T_i \oplus \tilde{T}_i$ выполняется только при $i = 1$, что облегчает задачу дешифрования лишь первого, но никак не последующих блоков сообщения. Это снижает остроту проблемы, но, конечно же, не снимает ее полностью. Если криптоаналитик обладает возможностью анализа на основе произвольного открытого текста, включая выбор синхропосылки, такая схема шифрования неустойчива к анализу. Таким образом, и в этом случае необходимо позаботиться об оригинальности синхропосылки.

Завершая рассказ о гаммировании отметим, что Российский стандарт шифрования не предусматривает никаких других режимов шифрования, кроме описанных выше. Американский же стандарт определяет еще несколько режимов с обратной связью, однако они невыгодно отличаются от описанных выше тем, что за одно обращение к процедуре простой замены шифруется порция данных τ, по размерам меньшая, чем полный 64-битовый блок данных алгоритма DES, $|\tau| < 64$, что, конечно же, снижает быстродействие шифрователь без сколько-нибудь заметного повышения его стойкости. Кроме того отметим, что все сказанное в данном разделе статьи останется верным, если вместо алгоритма простой замены по ГОСТ использовать какой-либо другой алгоритм простой замены – им могут быть DES, FEAL, албер и т.д. – стойкость получаемого таким образом потокового шифра полностью определяется стойкостью использованного шифра простой замены.

10. Имитозащита.

Как мы уже неоднократно демонстрировали в примерах, шифрование само по себе не может защитить передаваемые данные от внесения изменений. Это является отражением того факта, что секретность и аутентичность – суть независимые свойства криптографических систем. Таким образом, при организации шифрованной связи нужно отдельно позаботиться об **имитозащите** передаваемых данных.

> ***Имитозащита*** **есть защита системы шифрованной связи или иной криптосистемы от навязывания ложных данных.**

Так как мы рассматриваем системы защиты информации, обеспечивающие необходимые характеристики только путем манипулирования самой информацией безотносительно свойств ее материальных носителей, обеспечение аутентичности сообщения может быть достигнуто только закладкой в него определенной избыточности, которая с достаточной степенью достоверности позволяла бы обнаружить все внесенные в него несанкционированным образом изменения. Самый простой и очевидный способ это сделать – добавить к сообщению дополнительный код, называемый контрольной комбинацией или имитовставкой, зависящий от его содержания:

$\tilde{T} = (T, Y)$, где $Y = f(T)$.

При получении сообщения принявшая его сторона проверяет выполнение условия $Y = f(T)$, и если оно справедливо, сообщение считается подлинным, в противном случае оно отвергается как ложное.

> ***Имитовставка*** **есть контрольная комбинация, добавляемая к передаваемому сообщению с целью защиты системы от навязывания ложных данных и зависящая от его содержания.**

Простейшим способом вычисления контрольной комбинации может служить контрольная сумма блоков сообщения по модулю некоторого числа – обычно размер контрольной комбинации равен размеру блоков данных:
$f(T) = (T_1 + T_2 + \ldots + T_n) \bmod 2^N$, где $N = |T_i|$ – размер блоков.

Однако очевидно, что данный способ вычисления контрольной комбинации не годится для имитозащиты, так как его подделка не представляет особой проблемы. Чтобы разобраться в том, какими свойствами должен обладать алгоритм выработки имитовставки, рассмотрим возможные угрозы аутентичности данных:

- злоумышленник может попытаться изменить данные $T \to \tilde{T}$ таким образом, чтобы их имитовставка осталась неизменной: $f(T) = f(\tilde{T})$;
- злоумышленник может попытаться снабдить сфабрикованное им сообщение $\tilde{T}$ правильно вычисленной контрольной комбинацией $f(\tilde{T})$, выдав тем самым его за подлинное;

Любая практическая схема имитозащиты должна обеспечивать эффективную защиту от двух перечисленных выше угроз, чего нельзя сказать об имитовставке – контрольной сумме.

Для вычисления такой контрольной комбинации может быть использована так называемая **вычислительно необратимая функция**. Функция $Y = f(T)$ называется **вычислительно необратимой**, если определение таких значений T, для которых справедливо уравнение $f(T) = Y$, то есть вычисление обратной функции $T = f^{-1}(Y)$ вычислительно невозможно. На практике это означает, что такое значение T не может быть определено более эффективным способом, чем перебором по множеству всех его возможных значений. Размер имитовставки Y $|Y| = N$ должен исключать сколько-нибудь значимую вероятность успешной подделки сообщения, которая в наихудшем для аналитика случае равна $p = 2^{-N}$.

Ясно, что понятие вычислительной необратимости чрезвычайно трудно формализуемо и поэтому его выполнение практически невозможно проверить для конкретных алгоритмов. Схема имитозащиты, основанная на необратимой функции, устойчива по отношению к первой угрозе. Действительно, чтобы ее реализовать, злоумышленник должен подобрать сообщение T под заданную контрольную комбинацию Y, для чего ему необходимо решить уравнение $f(T) = Y$ относительно T, что по нашему предположению невозможно за приемлемое время. Однако данная схема имитозащиты не защищает от второй угрозы. Чтобы сделать ее устойчивой к ней, можно передавать имитовставку вместе с сообщением зашифрованной на том же самом или другом ключе. Контрольная комбинация, вычисленная таким образом, в зарубежной литературе называется **кодом обнаружения манипуляций** с данными (Manipulation Detection Code), и сокращенно обозначается **MDC**. Для реализации данного метода необходима вычислительно необратимая функция сжатия данных. Слово “сжатие” присутствует в названии функции потому, что независимо от размера сообщения размер контрольной комбинации всегда постоянен, и, естественно, в большинстве практических случаев меньше размера сообщения. Однако соответствующая область математики настолько сложна, что вычислительная необратимость пока формально не доказана ни для одного используемого на практике алгоритма сжатия данных. В Российском стандарте на шифрование и имитозащиту данных подход на основе **MDC** не нашел отражения.

Чтобы злоумышленник не смог реализовать ни одну из перечисленных выше доступных ему угроз по нарушению целостности данных, достаточно обеспечить невозможность вычисления им контрольной комбинации $f(T)$ для любого текста T. Для этого просто надо сделать алгоритм ее вычисления $f(T)$ секретным элементом криптосистемы. Последовательное проведение в жизнь принципа Кирхгофа приводит к схеме, в которой функция f сама по себе не секретна, но зависит от вектора секретных параметров – ключа K:

$I = f(K, T) = f(K, T_1, \ldots, T_n)$, где

$T = (T_1, \ldots, T_n)$ – исходный текст, разбитый на блоки фиксированного размера. Выработанная таким способом контрольная комбинация получила название **кода аутентификации сообщений** (Message Autentification Code) и сокращенно обозначается **MAC**. В отечественной литературе она иногда (не вполне корректно) именуется криптографической контрольной суммой. Как нам реализовать эту функцию $f(K, T)$? В нашем распоряжении имеется алгоритм криптографического преобразования $I = E_K(T)$ (простой замены) ,который обладает необходимыми для построения подобных функций свойствами, однако позволяет работать только с блоками текста фиксированного размера $|T| = N$.

Первая идея, которая приходит в голову, это зашифровать контрольную сумму, вычисляемую обычным образом. Однако несостоятельность такого подхода очевидна – он позволяет нам защититься от второй из приведенных угроз, однако не вполне защищает от первой. Действительно, злоумышленник, располагающий некоторым перехваченным ранее сообщением как в открытой, так и в зашифрованной форме с имитовставкой, без труда может навязать такой криптосистеме ложное сообщение – для этого он фабрикует сообщение с точно такой же контрольной суммой, что и имеющееся у него, и снабжает его той же самой имитовставкой. Единственное, что может помешать злоумышленнику в этом, это невозможность корректно зашифровать сфабрикованное сообщение. Однако нам нельзя надеяться на это – требование обеспечения аутентичности сообщения независимо от его секретности остается в силе.

Наш первый подход к задаче построения алгоритма вычисления имитовставки оказался неудачным. Попробуем слегка его улучшить: будем сначала шифровать блоки сообщения, и лишь затем находить контрольную сумму полученных шифроблоков:

$$f(T) = (E_K(T_1) + E_K(T_2) + \ldots + E_K(T_n)) \bmod 2^N.$$

Данный подход устойчив по тем линиям, по которым вскрывался предыдущий, однако и он имеет очевидное слабое место: никак не реагирует на возможную перестановку блоков шифртекста. Для преодоления этого недостатка можно попытаться вместо суммирования использовать другую, некоммутативную функцию комбинирования блоков, однако это порождает множество различных неприятных проблем и поэтому данный подход не нашел практического применения.

Как же нам поступить в сложившейся ситуации? Тут самое время вспомнить про режим гаммирования с обратной связью – в этом режиме каждый блок шифртекста зависит от всех блоков исходного текста от первого до соответствующего ему. Значит, последний блок шифртекста зависит от всех блоков исходного текста, секретного ключа и, вдобавок, устойчив к возможным перестановкам блоков шифртекста, перед которыми спасовал предыдущий вариант. Таким образом, мы можем попытаться использовать его в качестве имитовставки:

$$I = T'_n = f(T_n, T'_{n-1}) = f(T_n, T_{n-1}, T'_{n-2}) = \ldots = f(T_n, T_{n-1}, \ldots, T_1, T_0).$$

При этом, однако, следует обратить внимание на две вещи:

- синхропосылка для вычисления имитовставки не нужна – преобразование можно начинать непосредственно с первого блока данных: $I = T_n \oplus E_K(T_{n-1} \oplus E_K(\ldots E_K(T_1)\ldots))$;
- данная схема уязвима по последнему блоку – действительно, итоговый результат для имитовставки образуется побитовым суммированием по модулю 2 последнего блока данных с некоторым кодом, зависящим от всех предыдущих блоков: $I = T'_n = T_n \oplus E_K(T'_{n-1})$ – это значит, что злоумышленник может сфабриковать произвольное сообщение, и затем добавить к нему еще один блок, вычисленный по формуле $T_n = I \oplus E_K(T'_{n-1})$, чтобы контрольная комбинация для него была равна заданному значению I;

Данные замечания легко учесть – надо лишь изменить порядок использования на каждом шаге операций ***побитового исключающего или*** и простой замены, – тогда алгоритм вычисления имитовставки будет выглядеть следующим образом:

$I = E_K(T_n \oplus E_K(T_{n-1} \oplus \dots E_K(T_1)\dots)))$.

Использование в качестве имитовставки последнего блока, полученного при шифровании массива данных гаммированием с обратной связью, или какой-либо функции от него может сделать некоторые криптографические протоколы несостоятельными. Это имеет место, если для зашифрования текста и выработки имитовставки используется один и тот же алгоритм – гаммирование с обратной связью, и один и тот же ключ. Покажем, что эта схема не обеспечивает имитостойкости. Пусть сообщение $T = (T_1, T_2, \dots, T_n)$ зашифровано на ключе K по алгоритму гаммирования с ОС: $T' = (T_1', T_2', \dots, T_n')$. Пусть в качестве кода аутентификации используется вычисленный по тому же самому алгоритму и на том же самом ключе код I, очевидно $T = T_n'$. Тогда полное шифрованное сообщение с кодом аутентификации имеет следующий вид: $\tilde{T}' = (T', I) = (T_1', T_2', \dots, T_n', T_n')$. Если злоумышленник внесет изменения в шифртекст, то тем не менее наша схема аутентификации не обнаружит изменений, хотя, полученный при расшифровании текст будет весьма далек от истинного. Причина этого кроется в том, что критерием подлинности сообщения здесь служит равенство последнего блока шифртекста контрольной комбинации, а это обеспечить сравнительно легко. Ситуация кардинально не улучшится, если вместо непосредственно последнего блока шифртекста в качестве имитовставки использовать функцию от него, полученную пусть даже с помощью алгоритма простой замены: $I = E_K(T_n')$.

Таким образом, криптографические схемы, в которых имитозащита не отделена от шифрования, то есть в качестве кода аутентификации используется часть шифртекста или функция, вычисляемая с использованием только такой части, могут оказаться неустойчивыми к манипуляции с шифртекстом. Отсюда следует, что существуют два пути устранения указанного недостатка:

- использовать для вычисления имитовставки различные криптоалгоритмы, или различные режимы одного и того же алгоритма;
- использовать для шифрования и вычисления имитовставки различные ключи;

Использование двух ключей вместо одного не совсем удобно, поэтому создатели ГОСТа пошли по первому пути – они разработали отдельный алгоритм E_K', предназначенный исключительно для вычисления имитовставки. Если для шифрования простой заменой на каждом шаге используется цикл 32-З, выражаемый в терминах композиции простых преобразований (см. раздел 4 настоящей статьи) следующим образом:
$E_K = \Gamma_{f_1} S \Gamma_{f_2} S \dots S \Gamma_{f_8} S \Gamma_{f_1} S \Gamma_{f_2} S \dots S \Gamma_{f_8} S \Gamma_{f_1} S \Gamma_{f_2} S \dots S \Gamma_{f_8} S \Gamma_{f_8} S \dots S \Gamma_{f_2} S \Gamma_{f_1}$,то при вычислении имитовставки используется упрощенный цикл 16-З, содержащий первые 16 шагов цикла 32-З: $E_K' = \Gamma_{f_1} S \Gamma_{f_2} S \dots S \Gamma_{f_8} S \Gamma_{f_1} S \Gamma_{f_2} S \dots S \Gamma_{f_8} S$. Упрощенный алгоритм вычисления имитовставки позволяет достигнуть примерно вдвое большего быстродействия по сравнению с режимами шифрования.

Другие отличия алгоритма вычисления имитовставки от алгоритма гаммирования с обратной связью обусловлены приведенными ранее причинами и заключаются в следующем:

- не используется синхропосылка;
- комбинирование кода с данными с помощью операции $\oplus$ осуществляется не после, а до шага криптопреобразования: $I_0 = 0$, $I_i = E_K'(I_{i-1} \oplus T_i)$, $i = 1,2,\dots,n$.

Второе свойство устраняет уязвимость имитовставки по последнему блоку сообщения.

Отметим, что в качестве имитовставки может браться не весь, а только часть полученного блока I_n: $I = (I_n)_{1\dots L}$, где $L = |I|$ – размер имитовставки. Как правило, это 32 младших бита блока I_n. Размер имитовставки L определяет вероятность успешного навязывания ложных данных, которая равна $p = 2^{-L}$. В заключении данного раздела отметим,

что в режиме вычисления имитовставки ГОСТ разрешает добавлять в начало текста его учетные параметры, такие, как дата последней модификации файла и его размер, и т.д.: $\boldsymbol{I}(\boldsymbol{T}) = \boldsymbol{I}(\boldsymbol{U},\boldsymbol{T})$, где $\boldsymbol{U}$ – вектор учетных параметров.

Заключение.

На этом автор вынужден поставить точку, хотя без рассмотрения осталось большое число важных вопросов, в частности, совсем не были затронуты вопросы реализации Российского стандарта в виде аппарата или программы для ЭВМ, представляющие в последнее время наибольший интерес для широкого круга специалистов. Тем не менее автор надеется, что данная статья будет полезна всем интересующимся криптографическими методами защиты информации. В качестве дополнения к статье прилагается набор уравнений для всех режимов криптопреобразований по ГОСТ 28147-89, являющийся по сути формальным и достаточно полным описанием стандарта.

Литература.

1. Дж. Л. Месси. Введение в современную криптологию. ТИИЭР, т.76, №5, Май 88, М, Мир, 1988, с.24–42.
2. М. Э. Смид, Д. К. Бранстед. Стандарт шифрования данных: прошлое и будущее. ТИИЭР, т.76, №5, Май 88, М, Мир, 1988, с.43–54.
3. У. Диффи. Первые десять лет криптографии с открытым ключом. ТИИЭР, т.76, №5, Май 88, М, Мир, 1988, с.54–74.
4. А. Н. Лебедев. Криптография с открытым ключом и возможности ее практического применения. Тем. Сб. «Защита информации», вып. 2, М.1992, с.129–147.
5. А. В. Спесивцев и др. Защита информации в персональных компьютерах. М., Радио и связь, 1992, с.140–149.
6. ГОСТ 28147–89. Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования.
7. С. Мафтик. Механизмы защиты в сетях ЭВМ. М., Мир, 1992.
8. В. Жельников. Криптография от папируса до компьютера. М., ABF, 1996.
9. А. Винокуров. “ГОСТ не прост, а очень прост.” М. “Монитор”, 1995, №1, с. 60-73.

Приложение 1. Уравнения криптографических преобразований по ГОСТ 28147-89.

Режим	Уравнение зашифрования	Уравнение расшифрования	Дополнительная информация
Простая замена	$T_i' = E_K^{(32)}(T_i),\ 1 \le i \le n$	$T_i = D_K^{(32)}(T_i'),\ 1 \le i \le n$	—
Гаммирование	$T_i' = T_i \oplus E_K^{(32)}(\Gamma_i),\ 1 \le i \le n$	$T_i = T_i' \oplus E_K^{(32)}(\Gamma_i),\ 1 \le i \le n$	$\Gamma_0 = E_K(S),\ \Gamma_i = (\Gamma_{i,0}, \Gamma_{i,1}),\ 1 \le i \le n$ $\Gamma_{i,0} = (\Gamma_{i-1,0} + C_0) \bmod 2^{32},\ 1 \le i \le n$ $\Gamma_{i,1} = (\Gamma_{i-1,1} + C_1 - 1) \bmod (2^{32} - 1) + 1,$ $1 \le i \le n,\ C_0 = 1010101_{16},\ C_1 = 1010104_{16}.$
Гаммирование с обратной связью	$T_i' = T_i \oplus E_K^{(32)}(T_{i-1}'),\ 1 \le i \le n$	$T_i = T_i' \oplus E_K^{(32)}(T_{i-1}'),\ 1 \le i \le n$	$T_0' = S.$
Выработка имитовставки	—	—	$I_0 = 0,\ I_i = E_K^{(16)}(I_{i-1} \oplus T_i),\ 1 \le i \le n,$ $I = (I_n)_{1\ldots L}$

Пояснения:

T_i, T_i' – i-тый блок открытого и зашифрованного текста соответственно;

S – синхропосылка – массив данных размером $|S| = 64$ бит, несекретна;

I – имитовставка – массив данных размером L не более 64 бит: $L = |I| < 64$;

$E_K^{(32)} = (\Gamma_1 S \Gamma_2 S \ldots S \Gamma_8 S)^3 (\Gamma_8 S \ldots S \Gamma_2 S \Gamma_1)$ – преобразование цикла зашифрования (32-З);

$D_K^{(32)} = (\Gamma_1 S \Gamma_2 S \ldots S \Gamma_8 S)(\Gamma_8 S \ldots S \Gamma_2 S \Gamma_1)^3$ – преобразование цикла расшифрования (32-Р);

$E_K^{(16)} = (\Gamma_1 S \Gamma_2 S \ldots S \Gamma_8 S)^2$ – преобразование цикла выработки имитовставки (16-З);

$S(T_0, T_1) = (T_1, T_0)$ – операция перестановки старшей и младшей 32-битовых половин преобразуемого блока данных;

$\Gamma_i(T_0, T_1) = (T_0, T_1 \oplus f_H(T_0, K_i))$ – операция одного шага преобразования 64-битового блока данных, $1 \le i \le 8$;

$f_H(t, k) = R_{11}^{\leftarrow}(C_H((t + k) \bmod 2^{32}))$ – функция шифрования, t – преобразуемый блок данных, k – используемый на шаге элемент ключа – 32-битовые блоки данных: $|t| = |k| = 32$;

$R_{11}^{\leftarrow}$ – операция вращения (циклического сдвига) 32-битового блока данных на 11 бит влево (в сторону старших битов);

C_H – операция преобразования 32-битового блока данных, заключающаяся в поблочной подстановке 4-битовых групп данных по таблице замен $H = \{h_{i,j}\}_{1 \le i \le 8, 1 \le j \le 16}$: $C_H(t) = C_H(t_1, t_2, \ldots, t_8) = (h_{1,t_1}, h_{2,t_2}, \ldots, h_{8,t_8})$;