Хорст Файстель.

Криптография и компьютерная безопасность.

Перевод Андрея Винокурова.

Перевод выполнен по изданию:
Horst Feistel, Cryptography and Computer Privacy, Scientific American, May 1973, Vol. 228, No. 5, pp. 15-23.

Компьютерные системы вообще и банки персональных данных в частности нуждаются в защите. Это может быть достигнуто шифрованием компьютерных данных и аутентификацией законных источников команд компьютеру.

В обществе растет беспокойство по поводу того, что компьютеры представляют сейчас или будут представлять в ближайшем будущем опасную угрозу тайне частной жизни. Поскольку многие компьютеры содержат персональные данные и доступны через удаленные терминалы, они являются непревзойденным средством накопления больших массивов информации об отдельных людях и группах людей. Специалисты считают, что вскоре станет технически осуществимым ведение подробного досье на каждого гражданина, тогда как до совсем недавнего времени данные для таких досье были рассеяны по многим местам, относящихся к тому же к юрисдикции совершенно разных ведомств. Однако я собираюсь показать в настоящей статье, что компьютерные системы все же могут быть приспособлены к защите хранящейся на них информации от всех людей, за исключением, естественно, тех, кому разрешен доступ к ним, путем зашифрования данных в формы, весьма устойчивые к попыткам взлома.

Традиционно людьми, больше всех нуждающимися в секретности, были военные и дипломаты. В их работе часто необходимы элементы неожиданности, а неожиданность такого рода всегда предполагает секретность. Что касается обычных людей, то какую бы потребность в секретности они не испытывали, это оставалось их личной проблемой и редко выносилось на публичное обсуждение; любовники и воры всегда сами, как могли обеспечивали свои потребности в секретной коммуникации. Это положение дел мало изменилось до самой середины XIX века – как раз примерно в то время для усовершенствования техники тайного письма были привлечены научные методы и способы мышления. Несмотря на это вплоть до нашего столетия способы, использованные для секретной коммуникации, продолжали оставаться процедурами, выполняемыми с помощью карандаша и бумаги.

Криптографическое зашифрование может быть достигнуто двумя совершенно различными путями: с помощью шифров и с помощью кодов. Различие между ними следующее: шифр всегда определяет символы подстановки для некоторого заданного набора букв алфавита. Будучи по своему характеру процедурой, завязанной на алфавит, шифр позволяет выразить все, что может быть напечатано на печатной машинке на каком-либо языке. Это означает, что посредством шифра можно выразить мысли, которые никогда до этого не высказывались и даже не предполагалось, что в будущем может возникнуть необходимость их выразить. С другой стороны, код является по своему внутреннему характеру семантическим. С помощью кода можно выразить только то, что было обдумано заранее и предусмотрено для передачи в виде секретного списка, такого, например, как кодовая книга.

Необходимо помнить, что в наше время слово “код” часто используется в значении, не связанном с криптографией. В таких случаях этот термин обычно имеет

широкий смысл, включающий наборы символы со специальным назначением. Так, говорят о кодах обнаружения и исправления ошибок, кодах сжатия данных, телекоммуникационных кодах, коммерческих кодах и кодах, включающих все виды замысловатых электрических сигналов и волновых форм.

Берясь в исследовательском центре Томаса Ватсона корпорации IBM за проблему безопасности, создаваемую современными компьютерами, мы отводили главную роль в ее решение использованию методов шифрования. Конечно, было бы невозможно охватить в настоящей статье весь предмет обеспечения конфиденциальности хранилищ данных и безопасности компьютерных операций. Я, однако, надеюсь показать вам тесно связанные с указанными проблемами принципы, которые лежат в основе шифрования данных и аутентификации их источников.

В современных межмашинных сетях того типа, который нужен для создания хранилищ данных, понятие секретности подразумевает нечто больше, чем просто сокрытие сообщений от посторонних. Хранилища данных включают в себя как составную часть терминал–компьютерные сети. Линии связи, соединяющие терминалы с компьютерными центрами полностью открыты не только для перехвата информации, но и для ее преднамеренного изменения и искажения. Таким образом, в добавок к и так достаточно очевидным аспектам секретности данных, необходимо обеспечить адекватную защиты от возможных злоупотреблений в системе. Простое обнаружение ошибок не может решить эту задачу. Здесь требуется, чтобы сама система сделала исключительно маловероятным то, что не допущенный к ней, но хитроумный и изощренный человек мог бы войти и прочитать либо исказить данные, или выполнить команды в такой системе.

Это жизненно важно, поскольку даже минимальное количество ложных данных, попавших в систему хранения данных случайно или введенных туда умышленно, может сделать ее работу бессмысленной. Компьютер без достаточной защиты легко обмануть, особенно человеку, который хорошо разбирается в принципах его работы. Действительно, терминалы могут быть использованы для хитроумно спланированных внесений ложных данных в систему. Чтобы предохранить чистоту банка данных, необходимо проверять подлинность законного источника и характер любых полученных от него данных, и делать это очень быстро и с высокой степенью надежности.

Давайте начнем рассмотрение вопроса с самого элементарного факта относительно шифров: вся криптография сводится к подстановкам. В своей простейшей форме подстановка может быть задана с помощью таблицы, как показано на следующей ниже иллюстрации. В левом столбце таблицы перечислены обычные буквы алфавита, из которых составлен открытый текст; в правой части – их эквиваленты, задаваемые шифром, и являющиеся подставляемыми значениями. Дилетанты часто бывают обескуражены огромным количеством способов выполнить такую подстановку или перестановок алфавита. Для латинского[1] алфавита, содержащего 26 букв, существует $1\times2\times3\times...\times26$ способов записать один алфавит подстановки. (Такое произведение называется 26-“факториал” и записывается 26! Факториал любого числа[2] n! есть произведение всех целых чисел от 1 до n.)

Число n! возможных перестановок таблицы с n входами составляет число возможных “ключей”. В данном случае, 26! – это очень большое число, больше 4×10^{26}. Но несмотря на это любая простая алфавитная подстановка легко может быть вскрыта статистическим анализом, основанным на определении частот символов в тексте. Если буква **Q** встречается чаще чем любая другая буква в достаточно длинном отрезке

---

[1] Автор называет его английским, но, как известно, британцы позаимствовали алфавит у латинян и весь мир называет его латинским.

[2] Любого целого положительного числа.

шифротекста, аналитик может быть вполне уверен, что эта буква заменяет ***E***, букву, которая чаще других встречается в текстах на английском языке.

Откр. текст	Шифр 1	Шифр 2		Откр. текст	Шифр 1	Шифр 2
A	F	○		N	M	⊖
B	E	□		O	B	∞
C	K	#		P	Z	⋒
D	J	△		Q	L	⬡
E	N	♡		R	V	◪
F	P	∃		S	X	◎
G	O	⚹		T	G	⌂
H	C	◧		U	T	∑
I	D	✠		V	R	⊠
J	Y	⊗		W	S	┼
K	U	▽		X	I	⧈
L	W	☽		Y	Q	☼
M	H	◐		Z	A	╳

Открытый текст

S	E	N	D	M	O	N	E	Y

Шифр 1

X	N	M	J	H	B	M	N	Q

Шифр 2

◎	♡	⊖	△	◐	∞	⊖	♡	☼

Подстановка – основная операция криптографии, здесь она проиллюстрирована двумя таблицами, объединенными в одну, определяющими замену букв исходного текста на их шифр-эквиваленты – другие буквы или произвольные символы.

Конечно, при замене букв алфавита подстановки на таинственно выглядящие символы никаких преимуществ не достигается. Аналитик может не беспокоиться о том, насколько сложными выглядят символы сообщения, он просто заменит их произвольным образом на буквы обычного алфавита или числа, и подвергает сообщение частотному анализу. Подход на основе “загадочных символов” демонстрирует, однако, гибкость шифра относительно возможных заменяющих символов. Так как таблицы подстановок, несмотря на их хорошо известную слабость, имеют основополагающее значение в криптографических разработках, давайте рассмотрим возможность использования действительно полезных символов подстановки.

Это и в самом деле может быть сделано. Двоичная система счисления, содержащая ровно две цифры, 0 и 1, идеально подходит для криптографического преобразования данных компьютерами. Используя ***n*** двоичных цифр можно записать 2^n различных двоичных кодов. Так, используя блок из 5 двоичных цифр, можно записать 2^5 или 32 различных комбинаций, – более чем достаточно для того, чтобы закодировать 26 букв алфавита (см. следующую ниже иллюстрацию). Если мы хотим поименовать или пометить большее количество элементов, нам придется увеличить наш запас двоичных чисел, просто увеличивая размер цифрового блока. Каждый раз, когда мы увеличиваем размер блока на одну цифру, мы удваиваем число возможных кодов. Следовательно, шестицифровой код обеспечивает 2^6 или 64 различных кодов, или достаточно для того, чтобы включить десятичные цифры, знаки пунктуации и т.д..

Вдобавок к той легкости, с которой двоичные цифры могут быть представлены в электрических схемах (например, наличием или отсутствием сигнала в проводнике), они имеют все полезные свойства обычных десятичных чисел. Так, двоичные числа могут складываться, вычитаться, умножаться и т.д. точно так же, как обычные десятичные числа. Мы покажем, что арифметические манипуляции с данными играют жизненно важную роль в криптографических методах, разработанных для компьютеров.

Открытый	Двоичный эквивалент	Двоичный шифр
A	00000	00100
B	00001	01001
C	00010	10001
D	00011	01010
E	00100	00011
F	00101	10011
...	...	...
X	10111	11101
Y	11000	01110
Z	11001	10110

Перевод в двоичную систему – необходимая в компьютерной криптографии операция. Например, двоичный шифр подстановки состоит из, первое, перевода каждой буквы алфавита в пятицифровое двоичное число, и, второе, из зашифрования двоичного эквивалента.

Теперь мы можем усложнить шифр принципиально иным способом. Вместо того, чтобы пользоваться единственной таблицей подстановок, мы можем использовать несколько таблиц в хаотичном, но определенном заранее порядке. Порядок использования таблиц и образует ключ. Если мы используем только две таблицы, обозначенные 0 и 1, то типовой ключ может быть, например, таким: 1101101. Теперь мы имеем дело с многоалфавитной подстановкой. Относительно этого нового источника сложности в шифре возникает следующий вопрос: возможно ли упростить таблицы подстановок, сделав их меньше? Простейшая двоичная подстановка, которую только можно выполнить, это замена одной двоичной цифры на другую, и в этом случае существует всего две различные таблицы подстановок. Давайте рассмотрим эти две таблицы, каждая из них будет соответствовать одному из двух основных типов ключа, как показано на следующей ниже иллюстрации. Таблица, помеченная “Ключ 1” заменяет нули на единицы и наоборот, а таблица, помеченная “Ключ 0” оставляет цифры неизменными. Существуют только две указанные возможности. Но оказывается, что тот же самый эффект можно получить с помощью операции, известной как сложение по модулю 2: две одинаковые цифры в результате такой операции дают 0, две различные – 1. Поэтому в рассматриваемом случае шаблонный ключ можно назвать также аддитивным ключом.[3]

Прежде чем продолжать дальше позвольте мне объяснить, что начиная с этого места мы будем молчаливо предполагать, что сообщение, которое мы хотим криптографически преобразовать, прежде всего переводится в последовательность двоичных цифр. Любой сорт информации, будь это письма, музыка или телевизионный сигнал, могут быть представлены в двоичном коде. Мы больше не будем беспокоиться о первоначальном смысле сообщений – мы будем иметь дело только с их двоичным представлением.

Теперь мы готовы рассмотреть что получается когда мы шифруем последовательность двоичных цифр, скажем, 0001010, преобразуя ее в другую последовательность, используя два ключа, **Ключ 1** и **Ключ 0**, в некотором произвольном

[3] Автор называет ключи по-разному в зависимости от способа их использования. Так, цифры **шаблонного ключа** используются для выбора одного из нескольких возможных однотипных преобразований по его номеру, а цифры **аддитивного ключа** просто прибавляются к цифрам преобразуемого блока данных.

порядке: 1101101. Принимая во внимание правило, что **Ключ 1** заменяет нули на единицы и наоборот, а **Ключ 0** оставляет цифры неизменными, мы получим следующее:

Сообщение:		0001010
Ключ:	+	1101101
Шифр:		1100111

Эта операция является сложением по модулю 2. Она имеет одно весьма удобное свойство: вычитание по модулю два есть то же самое, что и сложение, поэтому исходное сообщение может быть восстановлено просто прибавлением последовательности цифр ключа (она известна тому, кому направлено сообщение), к последовательности цифр шифра:

Шифр:		1100111
Ключ:	–	1101101
Сообщение:		0001010

а)

Ключ 0

Откры-тый	**Шифр**
0	0
1	1

Ключ 1

Откры-тый	**Шифр**
0	1
1	0

б)

Ключ	**Сообщение**	
	0	1
0	0	1
1	1	0

в)

Сообщение Ключ Шифр
0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 0

г)

“F” = 00101
Ключ = 10110
Шифр = 10011

Шифр = 10011
Ключ = 10110
00101 = “F”

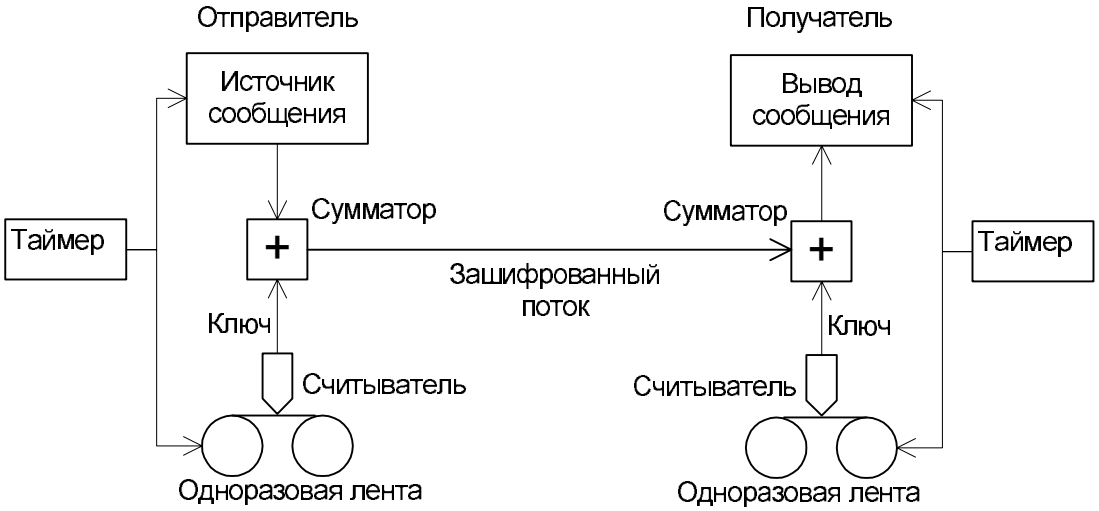
Двоичная подстановка в своей простейшей форме позволяет только две возможности: (**а**) в одной таблице (Ключ 0) открытые цифры равны цифрам шифра, во второй (Ключ 1) цифры шифра противоположны открытым цифрам. В таком варианте двоичной подстановки две таблицы подстановки могут быть заменены единственной таблицей сложения по модулю 2 (**б**); в этом случае двоичные подстановки эквивалентны двоичному сложению (**в**). Ключ шифрования для двоичного сложения по модулю 2 может быть произвольной последовательностью единиц и нулей. Чтобы зашифровать двоичное представление буквы сообщения, прибавляют цифры ключа к каждой цифре сообщения. При расшифровании вычитают цифры (это то же самое. что и сложение по модулю 2).

Сразу же возникает вопрос: имеет ли рассмотренный простой шифр какое-либо практическое значение? Поскольку этот шифр, в сущности, использует лишь две таблицы подстановки минимального размера, очевидно, что мы должны переключаться между ними часто, и делать это случайным образом, то есть прибавлять к данным случайную последовательность ключевых цифр. Предположим что мы делаем это, – тогда ответ на наш вопрос достаточно неожиданный: мы имеем потенциально нераскрываемый шифр. С точки зрения теории информации этот шифр делает следующее: к каждому биту информации сообщения прибавляется один бит информации (а точнее, дезинформации!) ключа. Этого достаточно, чтобы полностью разрушить любую структуру, которую исходное сообщение могло бы иметь, если только цифры ключа взяты в случайном, скажем, определяемом подбрасыванием монеты порядке, ключевая последовательность имеет такую же длину, как и сообщение, и никогда не повторяется.

Почему эта система действительно невскрываемая? На самом деле, это даже вообще не “система”. Рассмотренное криптографическое преобразование является не более чем случайным прибавлением одной цифры, и в такой же степени тривиально. Стойкость метода следует исключительно из того факта, что для каждой цифры сообщения мы полностью и случайным образом меняем ключ. Это единственный класс шифров, для которого можно доказать невскрываемость в абсолютном смысле этого слова.

Даже если оппонент осуществляет попытку вскрыть систему грубой силой, например, опробованием всех возможных прибавляемых ключей (2^6 или 64, в случае нашего 6-битового сообщения), он получит все возможные открытые тексты, включая тот, который мы в действительности зашифровали. Так, если мы зашифровали имя “Том” (что потребовало бы, как минимум, пятнадцати двоичных цифр), аналитик нашел бы среди своих вариантов расшифрования все английские трехбуквенные имена, такие, как Джо (Joe), Джим (Jim), Джоб (Job) и так далее, включая Тома (Tom), но никаких намеков на то, какое имя является правильным. Даже сам бог или дьявол, который мог бы опробовать все возможные ключи в одно мгновение, не мог бы внести сюда никакой определенности. Эта система хорошо известна и используется на практике под разными именами, такими, как система Вернама или одноразовый блокнот, всеми крупными правительствами.

В реальных системах сначала подготавливают две идентичные ленты со случайными цифрами ключа, ленты могут быть любого типа – телетайпные, перфорированные, магнитные или какие-нибудь еще. Одна остается у отправителя, а другая передается “неперехватываемым” образом например, курьером с охраной, законному получателю. Когда отправитель хочет передать сообщение, он сначала преобразует его в двоичную форму и помещает в устройство, которое к каждой цифре сообщения прибавляет по модулю два цифры, считанные с ключевой ленты, как показано на следующем ниже рисунке. На принимающей стороне кодированное сообщение записывается и пропускается через машину, похожую на устройство, использованное для шифрования, которое к каждой двоичной цифре сообщения прибавляет (вычитает) по модулю два цифры, считанные с ключевой ленты, получая таким образом открытый текст. При этом, естественно, ключевая лента должна продвигаться абсолютно синхронно со своим дубликатом, используемым для зашифрования.

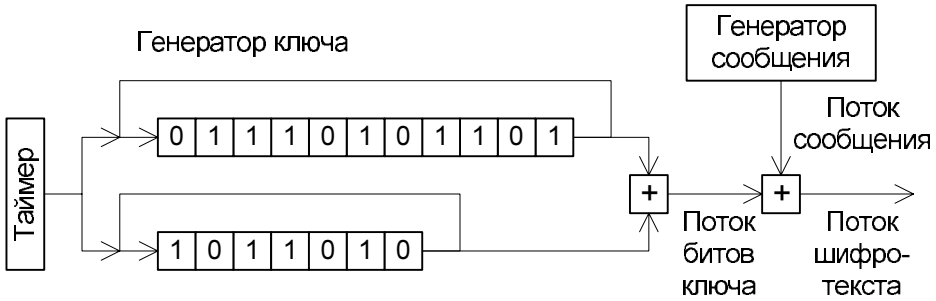


Система “одноразовая лента” требует, чтобы на передатчике и на приемнике были ленты с одним и тем же ключом, синхронизированные посредством таймера. Цифры сообщения и цифры ключа складываются по модулю 2, полученный в результате зашифрованный поток передается через канал связи, после чего ключ вычитается из данных (прибавляется по модулю 2). Для трафика большого объема обширные запасы ключевых цифр должны быть заблаговременно доставлены получателю и храниться у него.

Фундаментальный недостаток системы Вернама заключается в том, что для каждого бита переданной информации получателю необходимо хранить один заранее подготовленный бит ключевой информации. Более того, эти биты должны следовать в случайной последовательности, и эта последовательность не может быть использована вторично. Если необходимо шифровать трафик большого объема, это является довольно серьезным ограничением. В силу данного требования система Вернама используется только для передачи сообщений наивысшей секретности.

Чтобы обойти проблему предварительной передачи получателю сообщения секретного ключа большого объема, инженеры и изобретатели придумали много остроумных схем генерации очень длинных потоков псевдослучайных цифр из нескольких коротких потоков в соответствии с некоторым алгоритмом. Получателя шифрованного сообщения при этом необходимо снабдить точно таким же генератором, как и у отправителя. Конечно, такой алгоритм предполагает использование систематических процедур, добавляющих регулярности в шифротекст, обнаружение которых может помочь аналитику дешифровать сообщение.

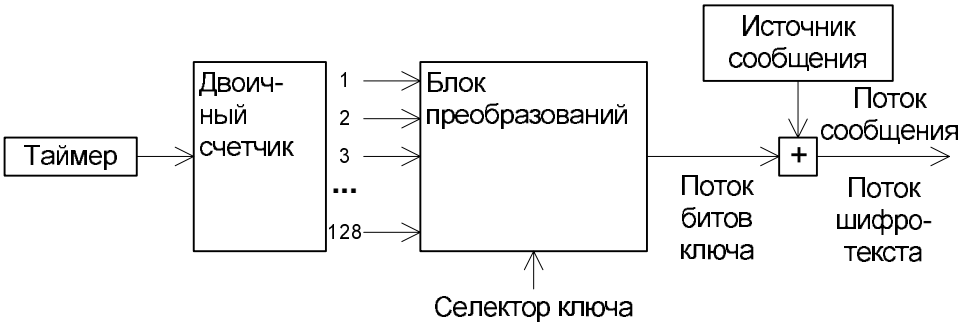
Один из основных методов построения подобных генераторов заключается в использовании двух или более битовых лент, считанные с которых данные побитно складываются для получения “смешанного” потока. Например, простая одноразовая лента может быть заменена двумя циклическими лентами, длины которых являются простыми или взаимно простыми числами. Так как в этом случае длины лент не имеют общих множителей, полученный из них поток имеет период повторения, равный произведению их длин: две ленты, имеющие длину 1000 и 1001 соответственно, дают в результате составной поток, который не повторяется на первых 1000×1001, или 1001000 цифрах. Ленты циркулируют через сумматор, который складывает по модулю два считанных с них цифры, как показано на следующем ниже рисунке. Выход сумматора служит ключом, используемым для зашифрования сообщения. Поэтому важно, чтобы составной поток превышал по длине все вместе взятые сообщения, которые могут быть переданы за разумный период времени.



Система “псевдослучайная лента” использует два замкнутых участка ленты с цифрами ключа или битами, которые складываются друг с другом для того, чтобы получить псевдослучайный поток битов ключа, прибавляемых по очереди к битам сообщения в точности так же, как при использовании одноразовой ленты. Расшифрование осуществляется с помощью ключа, идентичным образом сгенерированного на приемнике. Таким образом несколько коротких лент заменяют одну длинную, но получающиеся при этом внутренние периодичности могут помочь аналитику в раскрытии шифра.

Поскольку побитовый сумматор является линейным устройством, он изначально криптографически слаб, но может быть усилен большим количеством различных способов. Можно нагромождать усложнение на усложнение, вводя цепочки обратной связи, возможно, связанные каким-либо образом с передаваемым сообщением, или вводя такие нелинейные математические операции, как подстановки в блоках цифр подходящего размера. Несекретная криптографическая литература содержит много

очаровательных конструкций генераторов псевдослучайных последовательностей, все из которых могут быть, в принципе, сведены к одной базовой схеме, изображенной на следующем ниже рисунке. Тем или иным способом они вырабатывают псевдослучайные числа, выполняя сложные математические операции над каким-либо образом упорядоченной последовательностью входных чисел, таким образом преобразуя их способом, который, как предполагается, должен поставить в тупик аналитика.



Псевдослучайный поток может быть получен и более сложным образом. В этом обобщенном представлении его генератора двоичный счетчик обеспечивает входными числами блок преобразований, который вырабатывает цифры ключевого потока, прибавляемые далее к потоку цифр сообщения.

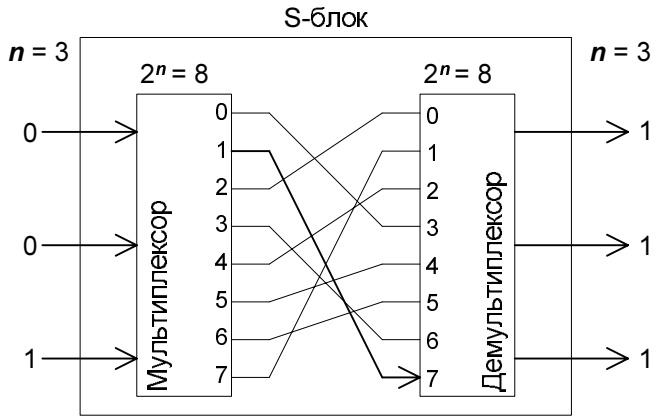
Читатель может быть удивлен и обескуражен узнав тот факт, что класс шифров Вернама – единственный класс шифров, для которого может быть доказана невскрываемость в абсолютном смысле этого термина, – является несовершенным в другом смысле: он сам по себе не обеспечивает защиты против искусного мошенничества с трафиком, не обладающим избыточностью.

Независимо от того, закодировано ли сообщение с использованием истинно случайных цифр или псевдослучайной последовательности цифр, в подобном побитовом зашифровании одиночная ошибка, возникшая при передаче сообщения, остается в рамках одной цифровой позиции; ошибка не “размножается”, не распространяется на остаток сообщения. Этот шифр не вносит межсимвольных зависимостей. Когда сообщение написано на естественном языке, таком, как английский, контекст естественной избыточности позволяет человеку, читающему текст, легко обнаруживать случайные ошибки. Так, если некоторые из 5 бит, представляющих букву E, оказались искаженными таким образом, что соответствующая группа битов стала представлением буквы F (так например, что слово SECRET превратилось в SECRFT), то читатель–человек немедленно обнаружил бы ошибку.

Совершенно иная ситуация при использовании компьютеров. Передаваемые данные здесь могут не содержать избыточности, например, если они полностью числовые, и в этой ситуации ошибка всего в одной цифре может вызвать целый каскад вычислительных погрешностей. Изучение проблемы показало, что простые коды обнаружения ошибок не подходят для защиты целостности компьютерных данных от возможных подтасовок со стороны экспертов–людей. В этой ситуации необходимо не просто обнаружение ошибок, а защищенная криптографическими методами аутентификация. Неожиданно оказалось, что это лучше всего достигается, если основывать решение на определенных принципах, внутренне присущих шифрующим структурам. Вместо того, чтобы пытаться изменить концепцию потока, давайте бросим свежий взгляд на концепцию всей криптографии: замену блоков или цифр сообщения.

Мы будем называть любой шифр, который преобразует ***n*** цифр сообщения в ***n*** цифр шифрограммы блочным шифром. Например, блочным будет такой шифр, который преобразует код 00000, представляющий по нашему соглашению букву A в открытом тексте, в, скажем, 11001, эквивалент A для шифротекста, по некоторому ключу

перестановки, в точности, как это задает таблица подстановок. Чтобы увидеть, как такое двоичное преобразование выполняется электронным устройством, давайте рассматривать подстановки только в группах из трех двоичных цифр, как это показано на следующем ниже рисунке.



Вход	Выход
000	011
001	111
010	000
011	110
100	010
101	100
110	101
111	001

Блок подстановок, в отличие от потоковых устройств, имеет более общий характер и включает как линейные, так и нелинейные преобразования: он не просто прибавляет нули и единицы к цифрам входа, но может заменить любой входной блок цифр на любой выходной блок. Реально он состоит из двух коммутаторов – один преобразует двоичное число из n цифр в одну цифру по основанию 2^n, другой выполняет обратное преобразование. Блок, таким образом, содержит 2^n внутренних соединений коммутаторов, которые могут быть выполнены $2^n!$ различными способами[4]. Это означает, что в случае изображенного на данном рисунке блока с $n = 3$ существует 40320 различных вариантов разводки или таблиц, подобных той, что изображена на рисунке. Блок такого типа с $n = 128$ сделал бы анализ практически неосуществимым, однако его технологически невозможно создать.

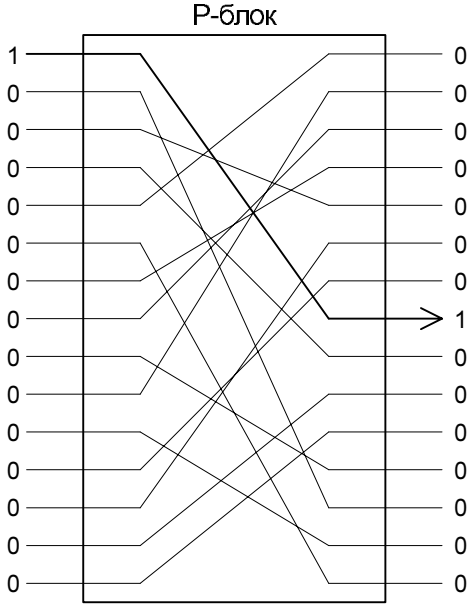
С помощью трех двоичных цифр можно представить восемь элементов: 2^3 равно восьми. Устройство, выполняющее подстановку, состоит из двух коммутаторов. Первый преобразует последовательность из трех двоичных цифр в соответствующее восьмеричное значение, подавая сигнал ровно на одну из восьми выходных линий. Эти восемь выходов могут быть соединены с восемью входами второго переключателя любым из 8! или 40320 способов. Мы вольны выбирать, который из 40320 различных вариантов соединения или коммутации проводов между первым и вторым переключателем использовать. Роль второго переключателя – преобразовать вход, представленный одной цифрой по основанию 8, обратно в трехцифровой двоичных выход.

Если бы устройство подстановки было построено для обработки пятицифрового двоичного входа, оно могло бы быть использовано для зашифрования алфавита из 32 символов. Количество возможных соединений двух переключателей было бы тогда 32!. Это могло бы показаться невероятно большим количеством ключей, но к созданному таким образом шифру все же необходимо относиться как к очень слабому: он не может противостоять частотному анализу. Эта слабость не является его неотъемлемым свойством; описанное устройство с математической точки зрения определяет наиболее общее возможное преобразование. Оно включает для любого заданного размера входа-выхода любой возможный обратимый шифр, который когда-либо был, или даже просто мог бы быть изобретен; математики могли бы сказать, что он представляет полную симметричную группу. Он полностью "несистематический": одно соединение переключателей ничего не говорит оппоненту относительно всех других соединений. Слабость не является внутренне присущим данному шифру свойством, она обусловлена

[4] В оригинале – $n!$ различными способами, что, безусловно, является опечаткой.

выбранным размером блока. Несмотря на большое количество ключей "каталог" возможных входов или выходов очень мал: в нем всего лишь 32 элемента. Здесь необходим, такой большой каталог, чтобы для любого оппонента было практически невозможно записать его. Если бы у нас, например, был блок со 128 входами и выходами, аналитику было бы необходимо одолеть 2^{128} (или больше 10^{38}) возможных блоков цифр – настолько огромное число, что частотный анализ здесь просто неосуществим. К несчастью, устройство подстановок со 128 входами также потребовало бы 2^{128} внутренних соединений между первым и вторым переключателями, что технологически нереализуемо. Это фундаментальная дилемма криптографии. Мы знаем, что является идеалом, но мы не можем достичь его на практике.

Однако существует преобразование, которое легко реализовать для большого набора входов. Практически возможно, например, построить блок со, скажем, 128 входными и выходными выводами, которые внутри соединены обычными проводами, как показано на следующем ниже рисунке. Для такого "блока перестановок" с n выходами имеется $n!$ возможных вариантов коммутации проводов, каждый из которых определяется отдельным ключом. Он легко может быть построен для $n = 128$. Хотя это обеспечит нам большое количество возможных ключей (128!), что весьма полезно, мы теперь столкнемся с новой трудностью. Путем использования набора специально сконструированных сообщений можно целиком определить ключ такой системы всего за $n - 1$ (в данном случае 127) попыток[5]. Этот прием заключается в том, чтобы использовать серию сообщений, содержащих одну единственную единицу в $n - 1$ различных позициях; позиция единицы в выходном блоке "выдаст" использованное в устройстве подключение провода. Слабость простого блока перестановок заключается в том, что он опять является линейной системой.



Блок перестановок может "обслужить" очень много линий, но он всего лишь изменяет положение цифр. Оппонент может узнать его внутреннюю коммутацию, подавая ему на вход одиночные единицы и наблюдая, на каком выходе эти единицы появляются.

Нам необходим компромисс, который бы, как минимум, приближался по характеристикам к общей системе. Мы пришли к идее составного шифра, в котором два или более шифра скомбинированы таким образом, что результирующая система обладает большей стойкостью, чем каждая из составляющих ее систем в отдельности. Как раз перед первой мировой войной исследовались различные громоздкие шифры, использующие несколько этапов шифрования. Первым действительно успешным образцом была, вероятно, система, изобретенная немцами, известная как ADFCVX–

[5] Реально это можно сделать даже за меньшее число попыток, если надлежащим образом формировать входные данные для каждого теста.

система. Нам необходимо здесь отметить только, что она соединяла "дробления" с "перестановками". В этой процедуре сообщение разбивалось на сегменты и эти сегменты переставлялись на другие места. Важный факт, на который следует обратить внимание, заключается в том, что составной шифр, составленный из блочных шифров, опять является блочным шифром; цель здесь, конечно, состоит в том, чтобы шифр вел себя, насколько это возможно, подобно общему шифру замен.

Между первой и второй мировыми войнами интерес к составным шифрам практически полностью пропал благодаря успешному развитию роторных или проводно-дисковых машин, которые принадлежат к общему классу генераторов псевдослучайных последовательностей. Типичная роторная машина имела клавиатуру, напоминающую клавиатуру пишущей машинки. Каждая буква шифровалась с помощью нескольких дисков, работающих в определенной последовательности, для очередной шифруемой буквы диски приводились в другое положение с использованием нерегулярного и зависящего от ключа алгоритма. Сообщение расшифровывалось идентичной машиной с точно таким же установленным ключом.

Современный интерес к составным шифрам возник благодаря статье Клода Шеннона, озаглавленной "Теория связи в секретных системах", которая была опубликована в техническом журнале корпорации Bell (*Bell System Technical Journal*) в 1949 году. В разделе, посвященном практической разработке шифров, Шеннон ввел в рассмотрение понятие "перемешивающего преобразования", которое предполагает особый способ использования результатов преобразования. В добавок к изложению интуитивных принципов, ведущих, как он полагал, к созданию стойких шифров, он ввел в обиход понятия "перемешивания" и "рассеивания". Его статья открыла практически неограниченные возможности по разработке и исследованию шифров.

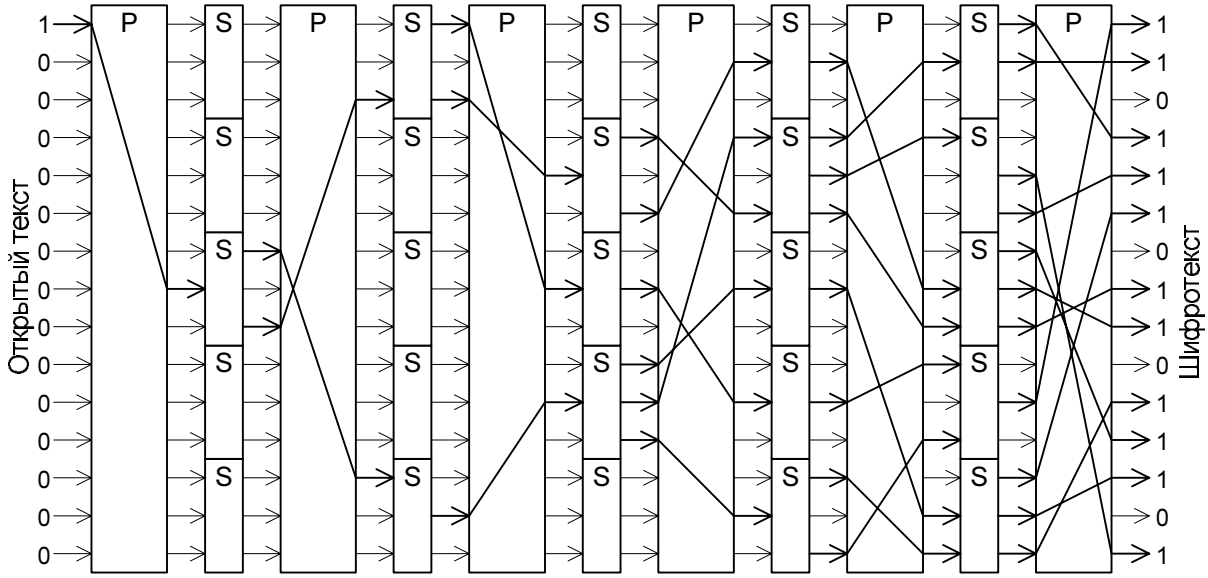
Способ, которым следует сочетать принципы перемешивания и рассеивания для получения криптографической стойкости, можно описать следующим образом: Мы видели, что перестановки общего вида не могут быть реализованы для больших значений n, скажем, для $n = 128$, и поэтому мы должны ограничиться схемами подстановки, имеющими практический размер. В системе, разработанной в IBM и названной "Люцифер" мы выбрали для блоков подстановки $n = 4$. Хотя это может показаться слишком маленьким числом, такая подстановка может оказаться вполне эффективной, если ключ подстановки, или схема коммутации проводников, выбрана верно. В Люцифере нелинейная подстановка эффективно обеспечивает определенную степень перемешивания.

Мы также видели, что линейный блок перестановок легко построить даже для $n = 128$. Число входных и выходных клемм равно n. Будучи "тасователем" цифр в чистом виде, устройством, которое просто перемещает цифры с места на место без изменения количества единиц и нулей, блок перестановок является естественным распределителем перемешивания, таким образом, он может обеспечить оптимальное рассеивание.

В системе "Люцифер" входные данные пропускаются через чередующиеся уровни блоков, которые мы обозначим символами **P** и **S**. **P** обозначает блок перестановок, в котором n является большим числом (128 или 64), и **S** обозначает блок подстановок, в котором n мало (4). Несмотря на то, что **P**-блоки в отдельности и **S**-блоки в отдельности составили бы слабую систему, их стойкость в комбинации друг с другом весьма значительна.

Мы проиллюстрируем меру стойкости подобных конструкций на примере изображенного ниже на рисунке устройства, в котором для простоты **P**-блоки имеют размер $n = 15$, а **S**-блоки $n = 3$. Если мы изобразим этот бутерброд из блоков, "озадаченный" специально сконструированным входным числом, составленным из 14-ти нулей и одной единственной единицы, то легко будет увидеть перемешивание и

рассеивание в работе. Первый блок, **P**, передает единственную единицу на вход некоторого блока **S**, который, будучи нелинейным устройством, может преобразовать единицу в трехцифровой выход, содержащий в потенциале целых три единицы. В показанном на диаграмме варианте он вырабатывает две единицы. Следующий блок, **P**, тасует две единицы и передает их на вход двух различных **S**-блоков, которые вместе имеют потенциал по выработке целых шести единиц[6]. Дальше диаграмма говорит сама за себя: по мере того, как входной блок данных проходит через последовательные уровни, узор из сгенерированных единиц расширяется и дает в результате непредсказуемый каскад цифр. Конечный результат, получающийся на выходе всей цепочки, будет содержать в среднем половину нулей и половину единиц, в зависимости от ключей перестановки, использованных в различных **P**- и **S**-блоках.



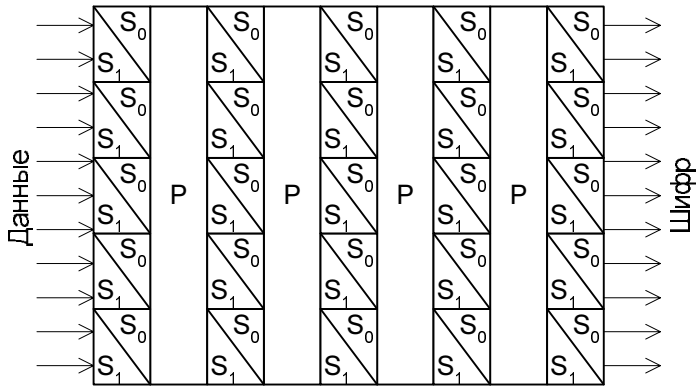
Составная шифрующая система объединяет S-блоки и P-блоки в единую конструкцию. P-блоки имеют большое количество входов (на рисунке показано 15), а S-блоки – столько входов, сколько подобные устройства в состоянии обслужить – три в данном случае. P-блоки тасуют цифры, обеспечивая рассеивание. S-блоки предусматривают нелинейные подстановки и таким образом обеспечивают перемешивание. В этом упрощенном примере вход системы включает одну единицу и четырнадцать нулей. Поскольку S-блоки нелинейны, они потенциально увеличивают число единиц, тогда как P-блоки просто двигают единицы с места на место. Результатом может быть непредсказуемая лавина единиц.

Очень важный факт заключается в том, что все выходные цифры потенциально стали очень сложными функциями всех входных цифр. Поскольку все блоки имеют независимые ключи, поток вырабатываемых цифр и окончательный результат не могут быть предсказаны. Целью разработчика, конечно, является сделать предельно трудным для оппонента проследить цепочки назад и, таким образом, реконструировать ключи в **P**- и **S**-блоках.

В реальной системе “Люцифер” мы сочли целесообразным внести в описанную схему несколько важных изменений. Например, **S**-блок, являясь достаточно общим преобразованием, может случайно быть снабжен таким ключом, что будет вести себя в

[6] Здесь Файстель допустил принципиальную ошибку. Написав в статье и изобразив на диаграмме (см. следующую ниже диаграмму), что число единиц в выходе каскада будет постепенно увеличиваться от каскада к каскаду по крайней мере на первых шагах преобразования, он неявно предположил, что **S**–блоки, получающие на входе блок из одних нулей, на выходе также выдают все нули. Безусловно, такие блоки явились бы весьма слабыми. На самом деле, при использовании правильно спроектированных блоков статистический баланс нулей и единиц установится уже на выходе первого слоя **S**–блоков.

точности как **P**-блок, и в этом случае вся система будет не более стойкой, чем один слой **P**, который может быть раскрыт описанной выше процедурой "щекотания". Чтобы избежать такого результата блоки обоих типов, **S** и **P**, снабжаются постоянными ключами, которые должны быть сильными; эти постоянные ключи будут известны каждому, кто имеет доступ к системе. Следовательно, нам необходим другой способ использовать ключи, при этом желательно, чтобы они могли быть представлены двоичными числами. Этого можно достигнуть построением "бутерброда", в котором каждый **S**-блок содержит два различных постоянных ключа, и, таким образом, может быть представлен двумя возможными различными состояниями, $\mathbf{S}_0$ и $\mathbf{S}_1$. Последовательность этих состояний для любого отдельного "бутерброда" составляет управляемую ключом структуру, неизвестную потенциальному оппоненту. Эта структура может быть представлена двоичным ключом, который в сущности указывает, которую из двух таблиц подстановки следует использовать, в точности как в случае двухтабличной подстановки, обсуждавшейся ранее (см. следующую ниже иллюстрацию; на диаграмме изображены 25 **S**-блоков, и таким образом, ключ имеет длину 25 цифр, в реальных устройствах может быть много больше **S**-блоков, и соответственно более длинный ключ.) Цифры ключа могут быть загружены в ключевой регистр криптографического устройства и могут быть записаны на ключевую магнитную карту, закрепленную за законным пользователем системы. Когда два состояния **S**-блоков используются подобным образом, результирующая криптограмма показывает наличие существенных межсимвольных зависимостей, которые делают все цифры выхода сложными функциями не только всех цифр входа, но и всех цифр ключа. Мы, таким образом, показали, что система устойчива к попыткам проникновения с помощью математических методов анализа.



Возможность использования индивидуальных ключей необходима, потому что ключи, встроенные в S- и P-блоки, известны всем, кто имеет доступ к системе. Эта возможность может быть обеспечена способом, показанным на данном рисунке. Для каждого S-блока в системе существуют два возможных состояния, S_0 и S_1. Порядок, в котором они используются, определяется двоичным ключом, как показано на нижней части иллюстрации.

Шаблон из S_0 и S_1.

$S_1\ S_0\ S_1\ S_0\ S_0\ S_0\ S_1\ S_0\ S_1\ S_1\ S_1\ S_1\ S_1\ S_0\ S_1\ S_1\ S_0\ S_1\ S_0\ S_1\ S_1\ S_1\ S_0\ S_1\ S_0$

Двоичный "шаблонный" ключ.

1 0 1 0 0 0 1 0 1 1 1 1 1 0 1 1 0 1 0 1 1 1 0 1 0

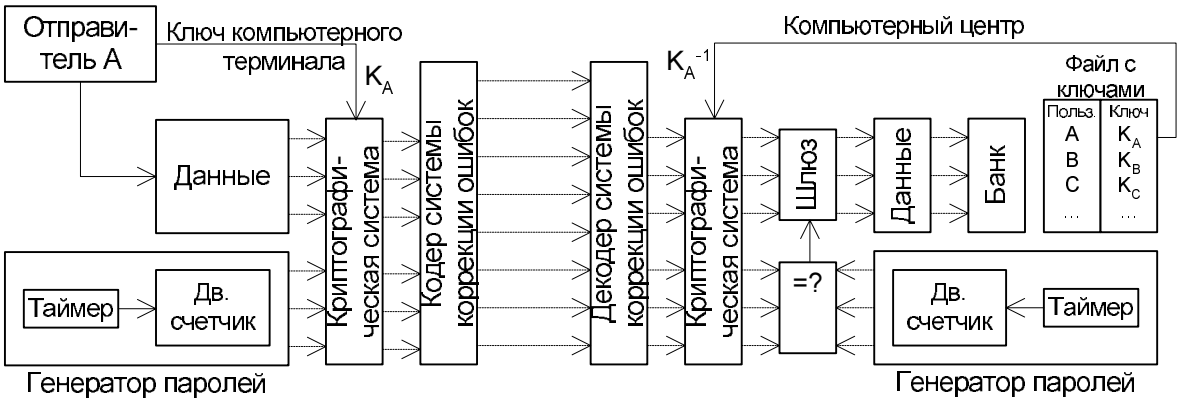
Хотя сильная межсимвольная зависимость является необходимым (но не достаточным) показателем криптографической стойкости, она имеет и оборотную сторону: она влечет за собой очень большую чувствительность к шуму или помехам во время передачи. Погрешность в единственной цифре может через эффект "обратной лавины" привести к полному искажению расшифрованных данных. Современные средства коммуникации делают, однако, проблему менее актуальной по крайней мере для невоенного использования.

Более того, сильные взаимозависимости между цифрами могут принести удивительную и неожиданную пользу: поскольку система так чувствительна к изменениям и так резко реагирует на них, она автоматически становится идеальным средством обнаружения таких изменений, как произошедших случайно, так и сделанных

умышленно. И мы, таким образом, одним выстрелом убиваем двух зайцев – имеем одновременно высокую секретность сообщений и неподдающийся обману сигнализатор ошибок.

Чтобы извлечь пользу из этой дополнительной особенности шифра нам необходимо всего лишь зарезервировать место для “пароля” внутри заданного блока цифр сообщения. Пароль – это последовательность цифр, автоматически вводимая в поток цифр сообщения передающей аппаратурой без какого-либо участия лица, использующего систему. Роль пароля заключается в том, чтобы сообщить приемной аппаратуре, что сообщение не было преднамеренно искажено или серьезно испорчено шумом в процессе передачи. Процесс зашифрования оставляет оппонента в неведении, как биты сообщения и пароля отображены в криптограмме. Если цифры пароля не могут быть восстановлены декодером на принимающем конце без ошибок, сообщение отвергается.

Решающую роль в этой схеме играет генератор пароля, который должен быть как на приемнике, так и на передатчике, как показано на следующем ниже рисунке. Генератор пароля на самом деле является не чем иным, как двоичным таймером или счетчиком, определяющим время или порядковый номер сообщения в двоичной записи, и добавляющим эту группу цифр к каждому блоку цифр передаваемого сообщения. Мы подразумеваем, что в определенный момент времени, скажем в 8:00, таймеры на обоих концах канала передачи были синхронизированы и их частоты одинаковы.



Полная система объединяет генератор пароля, составную криптографическую систему, такую, как “бутерброд” из S- и P-блоков, показанный на предыдущей иллюстрации, и систему коррекции ошибок. Генератор паролей вырабатывает свежий парольный блок для каждого блока данных. Отправитель, используя свой персональный ключ, вводит свои данные. Цифры пароля и данных станут неотслеживаемыми после того, как будут зашифрованы в соответствии с ключом. Дополнительные цифры кода коррекции ошибок добавляются к данным перед передачей и изымаются сразу после приема. Криптографическая система компьютерного центра расшифровывает передачу в соответствии с инвертированным ключом отправителя, который выбирается из специального защищенного файла, хранимого в центре, и извлекает цифры пароля. Если они совпадут с цифрами пароля, сгенерированного в компьютере, шлюз открывается и входные данные передаются в хранилище как имеющие законный источник.

Теперь давайте посмотрим, как “парольная схема аутентификации” обеспечивает безопасность работы членам сообщества пользователей централизованного хранилища данных, которые имеют доступ к большому центральному компьютеру. Каждый участник имеет свой собственный секретный ключ, возможно, представленный в форме последовательности двоичных цифр, записанной на магнитную карту. Ключи всех пользователей хранятся в подходящим образом защищенной форме в центральном компьютере. Предположим, что пользователь с ключом **A** хочет передать сообщение на центральный компьютер. Он вставляет карточку, на которой записан его ключ, в напоминающий пишущую машинку терминал, располагающийся на его рабочем столе, секунду или две ждет сигнала, что линия свободна, и начинает набирать свое сообщение.

Сообщение автоматически разделяется на блоки цифр (скажем, по 64 цифры), которые на каждом тике двоичного таймера объединяются с паролем (который также может иметь размер 64 цифры), соответствующим выходу таймера в этот момент времени. Результирующий блок из 128 цифр подвергается зашифрованию, для чего пропускается через сэндвич из P- и S-блоков, который полностью перемешивает цифры пароля и цифры данных.

Поскольку результирующая криптограмма очень чувствительна к ошибкам передачи, она усиливается с помощью подходящего кода исправления ошибок, который должен соответствовать шумовым характеристикам используемых телефонных линий. Добавление этого кода удлиняет блок, содержащий цифры пароля и сообщения, еще на несколько цифр. Результирующий блок шифрограммы предваряется адресом отправителя в открытом виде и передается на центральный компьютер. Когда сообщение доходит до адресата, ключ K_A, принадлежащий пользователю **A**, отыскивается в списке и его обращение[7] загружается в декодер для того, чтобы расшифровать криптограмму.

Теперь решающей проверкой является, будет ли совпадать пароль из полученной криптограммы с паролем, локально выработанным двоичным таймером на принимающей стороне. При отсутствии искажений и если криптограмма была действительно зашифрована на ключе пользователя **A** выход декодера будет состоять из блока цифр данных и блока цифр пароля. Это считается достаточным свидетельством в пользу того, что криптограмма действительно создана участником **A**. Данные принимаются системой как имеющие законный источник.

Что же случится, если произошло искажение данных? Если оно было вызвано спорадическими всплесками шума в линии передачи, код исправления ошибок устранит его и сообщение успешно пройдет аутентификацию. Если, однако, искажения имеют такую природу, что не могут быть устранены кодом коррекции ошибок, то даже одна неверная цифра произведет эффект лавины в декодирующем устройстве и превратит весь выход в мусор. Пароли больше не будут совпадать. Система воспримет сообщение как имеющее подозрительное происхождение и отвергнет его.

Парольный тест сработал бы так же надежно, если кто-либо записал бы перехваченное сообщение и повторно передал бы его позже, когда пароль перестал быть действительным. Конечно, использование неверного ключа является причиной для немедленной отбраковки сообщения. Представляется, что предложенная система устойчива к любой мыслимой попытке обмануть ее. Каждая двоичная цифра пароля обеспечивает один бит аутентифицирующей информации. Если пароль имеет длину в n цифр, то оппонент имеет лишь один шанс из 2^n (или один шанс из 2^{64}, если $n = 64$) сгенерировать любым выбранным им способом такую криптограмму, которая при расшифровании случайно даст истинный пароль. Число 2^{64} равно примерно 10^{19}. Невозможно аутентифицировать данные более эффективно.[8]

Нет причин, которые бы препятствовали распространению системы, описанной только что для одного пользователя, на всех пользователей сети для обеспечения их безопасной работы. В нашей лаборатории, как и в других местах продолжается также исследование и других криптографических подходов. Было бы удивительно, если бы криптография, будучи традиционным средство обеспечения конфиденциальности связи, не смогла бы обеспечить безопасность сообществу пользователей банков данных.

---

[7] В статье об этом нигде не сказано явно, но понятие “обращение” ключа подразумевает, что алгоритмы за- и расшифрования идентичны и различаются только использованными ключами, при этом “обращением” ключа зашифрования является такой ключ, что последовательное зашифрование сообщения на этих двух ключах оставляет его неизменным.

[8] Понятно, что это чисто эмоциональное утверждение. Если взять больший размер пароля, то можно аутентифицировать более эффективно, то есть с меньшей вероятностью успешной подделки сообщения.