ЖЭТФ, 2007, том 131, вып. 5, стр. 771–789

О СТРУКТУРЕ ОТКРЫТОГО КЛАССИЧЕСКОГО КАНАЛА СВЯЗИ В КВАНТОВОЙ КРИПТОГРАФИИ: КОРРЕКЦИЯ ОШИБОК, ЦЕЛОСТНОСТЬ И АУТЕНТИЧНОСТЬ

А. В. Тимофеев [c], ***Д. И. Помозов*** [c], ***А. П. Маккавеев*** [c], ***С. Н. Молотков*** [a,b,c*]

[a] *Академия криптографии Российской федерации,*
Московский государственный университет им. М. В. Ломоносова
119899, Москва, Россия

[b] *Институт физики твердого тела Российской академии наук*
142432, Черноголовка, Московская обл., Россия

[c] *Факультет вычислительной математики и кибернетики,*
Московский государственный университет им. М. В. Ломоносова
119899, Москва, Россия

Поступила в редакцию 6 июня 2006 г.

В квантовой криптографии используются два канала связи — квантовый и классический. (Физически это может быть один и тот же канал связи, например, оптоволоконная линия, которая выполняет функцию квантового канала, если по ней передаются однофотонные квантовые состояния, и классического, когда передаются вспомогательные классические сигналы.) Оба канала считаются несекретными и доступными для прослушивания. Все обмены вспомогательной информацией при коррекции ошибок в первичных ключах, балансировки интерферометра и т. д. происходят через открытый классический канал связи, который должен удовлетворять определенным требованиям. Обсуждаются требования, предъявляемые к классическому каналу связи.

PACS: 03.67.Db, 03.65.Bz

1. ВВЕДЕНИЕ

Квантовая криптография представляет собой новое направление в развитии средств конфиденциальной передачи информации. Более точно, квантовые криптографические системы представляют собой системы распределения секретных ключей между пространственно удаленными легитимными пользователями. Обеспечение секретного распространения ключей между удаленными пользователями играет принципиально важную роль в криптографии. Если бы существовал способ распространения (передачи) секретных ключей от одного легитимного пользователя к другому по открытому (не секретному) каналу связи с гарантией того, что в процессе передачи ключи не станут известны подслушивателю, то в этом случае возможна передача зашифрованных сообщений на этих ключах, которые принципиально не могут быть дешифрованы (взломаны) третьими лицами. Такие принципиально не дешифруемые системы шифрования называют абсолютно стойкими или системами шифрования в режиме одноразового блокнота (one time pad) [1–3].

Неформально шифр является абсолютно стойким, если [1, 2] выполнены следующие условия:

1) ключ секретен, т. е. известен только легитимным пользователям;

2) длина ключа в битах не меньше длины сообщения;

3) ключ случаен;

4) ключ используется только один раз.

В этом случае зашифрованное сообщение статистически независимо от исходного сообщения.

Принципиальная проблема при реализации криптосистем с одноразовыми ключами состоит в передаче (распространении) секретных ключей между про-

*E-mail: molotkov@issp.ac.ru

странственно удаленными легитимными пользователями. Ключ между удаленными пользователями должен передаваться при помощи какого-то физического сигнала через открытый канал связи (открытый означает доступный для подслушивания третьими лицами). Если оставаться в рамках классической физики, то в этом случае на уровне фундаментальных законов природы не существует запретов на измерение передаваемого сигнала без его возмущения. Поэтому принципиально невозможно гарантировать секретность ключа при его распространении.

Если же передавать ключи при помощи квантовых состояний, то возникает принципиально другая, более интересная ситуация. Квантовая криптография, основанная на фундаментальных запретах квантовой механики, открывает возможность передачи ключей при помощи квантовых состояний, секретность в этом случае гарантируется фундаментальными законами природы. Если можно передавать ключи и гарантировать их секретность на уровне законов природы, то в этом случае можно реализовать абсолютно стойкие системы шифрования с одноразовыми ключами, истоки которых восходят к работам Вернама, Котельникова и Шеннона [1–3]. Собственно идея квантовой криптографии как раз и была направлена на решение центральной проблемы криптографии — задачи распространения секретных ключей.

Впервые идея использовать квантовую механику для защиты информации была высказана Визнером в 1973 г. (идея «квантовых» денег). Данная работа была опубликована лишь спустя десятилетие после появления самой идеи [4]. Интересно отметить, что идеи использовать квантовую механику для защиты информации появились раньше, чем классическая криптография с открытым ключом [5, 6]. Начало квантовой криптографии было связано с появлением замечательной работы Беннета и Брассара в 1984 г., в которой был предложен первый криптографический протокол $BB84$, ставший впоследствии классическим [7].

Квантовая криптография, или распространение секретных ключей, в принципе позволяет реализовать абсолютно стойкие (не дешифруемые подслушивателем даже теоретически) системы шифрования с одноразовыми ключами. Секретность ключей в квантовой криптографии основана на фундаментальных запретах квантовой механики [8, 9]. Во-первых, неизвестное квантовое состояние не может быть скопировано (теорема «no cloning» [8]). Во-вторых, пара наблюдаемых, которым отвечают некоммутирующие эрмитовы операторы, не может быть достоверно одновременно различима, что является следствием соотношений неопределенности Гейзенберга [9]. Более формально, некоммутирующие операторы не могут иметь общей системы собственных векторов. В квантовой криптографии в качестве таких наблюдаемых выступают матрицы плотности информационных состояний, соответствующих классическим битам 0 и 1. Для чистых состояний одновременная ненаблюдаемость (достоверная неразличимость) матриц плотности эквивалентна неортогональности информационных квантовых состояний [9]. Сказанное означает, что не существует измерений, которые с вероятностью единица позволяют различать одно из пары неортогональных состояний так, чтобы после измерения система оказалась в исходном (невозмущенном) состоянии, в котором она была до измерения.

Таким образом, любое измерение, если оно дает информацию о передаваемых состояниях, неизбежно приводит к их возмущению, что позволяет детектировать любые попытки подслушивания в канале связи. Другими словами, подслушивание (соответственно, возмущение) передаваемых состояний должно неизбежно приводить к изменению статистики результатов измерений на приемном конце по сравнению со статистикой результатов измерений на невозмущенных состояниях. Искажение квантовых состояний возникает в неидеальном квантовом канале, что также приводит к изменению статистики результатов измерений. В квантовой криптографии принципиально невозможно отличить изменение статистики результатов по сравнению с идеальным случаем, возникающих за счет шума в канале или от действий подслушивателя, поэтому любые изменения статистики приходится относить на действия подслушивателя.

Если бы законы квантовой механики позволяли обнаруживать только сам факт возмущения передаваемых состояний, то это было бы мало интересно для целей криптографии, точнее, для передачи ключей. Квантовая механика позволяет не только обнаруживать возмущение состояний, но и связать изменение статистики результатов измерений с количеством информации, которое может быть получено подслушивателем при наблюдаемом изменении статистики отсчетов по сравнению с идеальным случаем. В квантовой криптографии кроме квантового канала связи (в реальных условиях это либо оптоволокно, либо открытое пространство), по которому передаются квантовые состояния, необходим также открытый классический канал связи. Классический открытый канал связи необходим для выяснения ле-

гитимными пользователями изменений статистики отсчетов и коррекции ошибок в первичном ключе, переданном по квантовому каналу связи.

Единственное требование, которое предъявляется к классическому каналу связи, состоит в том, что передаваемая открыто и доступная всем, включая подслушивателя, классическая информация не может быть изменена подслушивателем — должна сохранять целостность (так называемый «unjamable channel») [7]. Такой открытый классический канал является математической идеализацией, поскольку подобных каналов в природе не существует. Для сохранения целостности открыто передаваемых классических данных в реальных условиях необходимо использовать процедуры аутентификации и контроля целостности данных. Для подобных процедур в свою очередь требуется секретный ключ. Если в качестве открытого классического канала используется, например, интернет, то для целей сохранения целостности и аутентичности данных возможна генерация ключей по схеме Хеллмана–Диффи [5, 6]. Если же для открытого классического канала используется та же самая оптоволоконная линия, что и для квантового канала связи, то генерация ключей для аутентификации по схеме Хеллмана–Диффи оказывается неприемлемой из-за очевидной, так называемой атаки «man in the middle» (человек посередине).

В такой ситуации требуется небольшой стартовый ключ один раз при первом запуске системы. При последующих сеансах этот ключ выбрасывается и для аутентификации и сохранения целостности данных, передаваемых по классическому каналу, используется часть ключа, сгенерированного по квантовому каналу в предыдущем сеансе обмена. Остальная большая часть ключа, полученного по квантовому каналу, используется собственно для шифрования передаваемой информации. Если для аутентификации и сохранения целостности данных используются процедуры на основе ГОСТа Р 34.11-94 [10], то длина стартового ключа составляет 256 бит. При этом в течение нескольких секунд обмена может быть получен новый секретный ключ по квантовому каналу гораздо более длинный, чем исходный.

Разумеется, стартовый ключ мог бы быть использован для шифрования этим ключом нового ключа и передачи его второму легитимному пользователю. Однако при этом абсолютная секретность нового ключа гарантируется, лишь если его длина не более длины ключа, на котором он шифруется. То есть более длинный ключ, чем исходный, получить нельзя. В квантовой криптографии стартовый ключ не используется напрямую для передачи нового ключа, который генерируется по квантовому каналу связи. При этом число бит открытой информации, переданной по открытому классическому каналу на один бит нового секретного ключа, может быть сделано меньше единицы, поэтому возможно расширение ключа.

Подход с небольшим стартовым ключом является более предпочтительным, чем использование алгоритмов асимметричной криптографии с открытым ключом, поскольку в этом случае можно свести к минимуму число раундов обмена по открытому каналу связи в процессе «чистки» и усиления секретности ключа (privacy amplification) [11].

Основная задача теории сводится к выяснению длины секретного ключа, который может быть получен при наблюдаемых изменениях статистики результатов измерений на приемном конце по сравнению со статистикой на невозмущенных состояниях. Как правило, величиной, которая характеризует отклонение статистики измерений от идеальной, является наблюдаемая вероятность ошибки на приемном конце. Точнее, вероятности того, что переданный бит был 0, а зарегистрирован как 1, и наоборот. Такая ситуация имеет место для широко используемого протокола $BB84$, хотя возможны и другие критерии для изменения статистики, которые используют несколько параметров. Перед выяснением вероятности ошибки через открытый канал происходит сравнение базисов на приемной и передающей стороне (для протокола $BB84$ [7]) или раскрытие позиций на приемной стороне, где имел место результат измерений с неопределенным исходом (для протокола $B92$ [9]). Оценка вероятности ошибки получается путем сравнения через открытый канал части переданной последовательности по квантовому каналу, в дальнейшем раскрытая часть отбрасывается.

Следующий этап любого квантового криптографического протокола распространения ключей состоит в коррекции ошибок в нераскрытой части последовательности у легитимных пользователей посредством обмена информацией через открытый канал связи. Легитимных пользователей обычно называют Alice и Bob, а подслушивателя — Eve (от английского «eavesdropper»). В результате коррекции ошибок остается последовательность бит меньшей длины и уже одинаковая у Alice и Bob. Слово «одинаковая» означает, что последовательности совпадают с вероятностью сколь угодно близкой к единице, $1-2^{\nu}$ (например, $1-2^{-200}\approx 1-10^{-70}$). Напомним, что число атомов в видимой части Вселенной оценивается как 10^{77}). Параметр ν выбирается легитимными пользователями.

После «чистки» первичного ключа у подслушивателя имеется строка бит или регистр квантовой памяти с состояниями или и то, и другое вместе. Последний шаг при получении финального секретного ключа состоит в усилении секретности [11] — сжатии «очищенного» ключа при помощи так называемых универсальных функций хэширования второго рода, которые сами являются случайными величинами (two universal hash functions [12]) для уже одинаковой последовательности у Alice и Bob. Случайно выбираемая функция хэширования открыто сообщается одним из легитимных пользователей через открытый канал связи и считается известной всем, включая подслушивателя. Сжатая последовательность бит является общим секретным ключом для легитимных пользователей, про который гарантируется, что подслушиватель имеет о ключе сколь угодно малую информацию по некоторому заданному легитимными пользователями параметру секретности.

Одним из требований к процедурам коррекции ошибок и усиления секретности ключа является сохранение как можно большего числа бит в финальном ключе. Еще одно требование состоит в минимизации числа раундов обменов по открытому каналу связи в пересчете на один бит в финальном секретном ключе. При коррекции ошибок в первичном ключе задача легитимных пользователей состоит не только в исправлении ошибок, но также в оценке верхней границы информации, которую может получить об оставшемся ключе подслушиватель из обменов по открытому каналу связи. Ниже подробно рассматриваются процедуры коррекции ошибок в первичном ключе, сжатие «очищенного» ключа и требования к открытому каналу связи, через который осуществляется обмен открытой информацией при этих процедурах.

2. КОРРЕКЦИЯ ОШИБОК («ЧИСТКА» КЛЮЧА)

В результате передачи квантовых состояний и измерений на приемном конце будет получена строка бит, т. е. легитимные пользователи Alice и Bob имеют битовые строки, причем строка бит у Bob содержит ошибки. В зависимости от количества ошибок в строке подслушиватель может иметь информацию о содержащихся в ней битах. Поэтому, прежде чем использовать эту строку в качестве секретного ключа, ее необходимо предварительно обработать. Обработка состоит из следующих шагов.

1. Оценка вероятности ошибки (это понадобится для более эффективного исправления ошибок).

2. Исправление ошибок («чистка»).

3. Проверка того, что все ошибки исправлены.

4. Удаление информации, выданной в процессе чистки.

5. Удаление информации, полученной подслушивателем в результате работы квантовой части протокола (процедура усиления секретности — хэширование при помощи универсальных хэш-функций второго рода [12]).

Далее подробно рассматривается каждый из этих шагов.

Подчеркнем здесь еще раз, что все обмены информацией через открытый классический канал связи между легитимными пользователями должны происходить с сохранением целостности и аутентичности данных. Этому вопросу посвящена вторая часть работы.

2.1. Оценка вероятности ошибки

Вначале биты первичного ключа $b_1, \ldots, b_t$ случайным образом перемешиваются по следующему алгоритму [13].

1. Присвоить $j = t$.

2. Сгенерировать случайное число $1 < k < j$.

3. Поменять местами биты b_k и b_j.

4. Уменьшить j на единицу. Если $j > 1$, вернуться к шагу 2.

Это равномерным образом распределит ошибки в ключе.

После этого сравним некоторое случайно выбранное количество бит ключа (достаточной величиной является $N = 30000$) и подсчитаем вероятность ошибки как отношение числа ошибочных бит к числу раскрытых.

Случайный выбор бит для проверки возможен по двум вариантам.

1. Последовательность проверочных бит выбирается на одной стороне, например, на приемной стороне Bob при помощи физического генератора случайных чисел (true random number generator). В этом случае требуется пересылка номеров позиций проверочных бит с одной стороны на другую.

2. Последовательность бит выбирается согласованно сразу на приемной и передающей сторонах при помощи генератора псевдослучайных чисел. В этом случае для старта генератора псевдослучайных чисел требуется небольшой стартовый ключ. Если используется генератор на основе ГОСТ 28147-89 [14], то длина ключа составляет 64 бита. В

Рис. 1.

этом случае пересылки позиций проверочных бит не требуется.

На этом этапе нам достаточно приближенной оценки ошибки, точную оценку мы получим после того, как исправим все ошибки. После этого раскрытые биты удаляются из ключа. Процесс передачи через открытый канал связи схематически изображен на рис. 1.

2.2. Оценка погрешности

Оценим количество бит, которое нам необходимо раскрыть для получения достаточно точной оценки. Благодаря случайной перестановке бит мы можем считать, что вероятность ошибки в каждом бите независима и равна p. Вероятность того, что оценка будет отличаться от истинной вероятности p больше, чем на δ, равна

$$\begin{aligned} P\left(\left|\frac{n}{N}-p\right|>\delta\right) &= P(|n-Np|>N\delta) = \\ &= P(n-Np>N\delta)+P(Np-n>N\delta) = \\ &= P(n>N(p+\delta))+P(n<N(p-\delta)) = \\ &= 1-P(N(p-\delta)<n<N(p+\delta)) = \\ &= 1-\sum_{i=\lceil N(p-\delta)\rceil}^{\lfloor N(p+\delta)\rfloor} C_N^i p^i (1-p)^{N-i}. \quad (1) \end{aligned}$$

Рассчитанная вероятность приведена на графике (рис. 2) для $\delta = 0.02$ и $p = 0.5$ (при $p = 0.5$ достигается максимум P). При $N = 30000$ вероятность ошибиться в оценке более чем на 2 % равна $4.41141 \cdot 10^{-12}$.

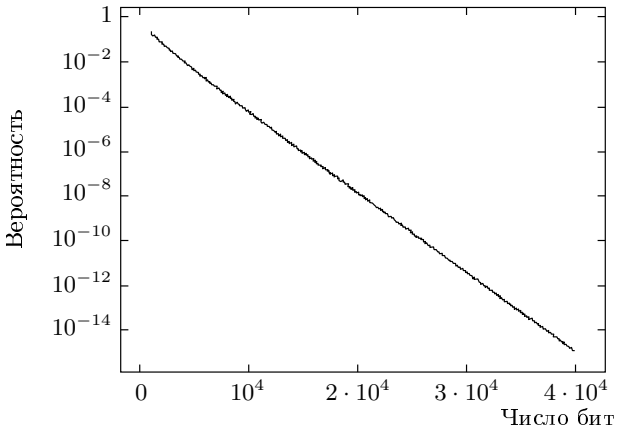

Рис. 2. Зависимость вероятности оценки истинной ошибки более чем на 2 % от числа бит, раскрытых для оценки ошибки

3. ИСПРАВЛЕНИЕ ОШИБОК

Следующая стадия протокола заключается в коррекции ошибок посредством обмена классической информацией через открытый классический канал связи. Исправление ошибок на приемной стороне, по своей сути, является классической процедурой. Принципиальное отличие этой процедуры от обычных методов коррекции ошибок в классической теории информации состоит в сохранении конфиденциальности. То есть в результате исправления ошибок подслушиватель, который при коррекции ошибок имеет доступ ко всей передаваемой информации, не должен увеличить информацию об «очищенном» ключе по сравнению с той, которую он имел до исправления ошибок легитимными пользователями. При этом часть бит в первичном ключе в процессе «чистки» легитимным пользователям приходится отбрасывать. Второе требование относится к эффективности процедуры и состоит в том, чтобы «очищенный» ключ оставался как можно большей длины.

Наша цель — максимально эффективно исправить все ошибки в первичном ключе. Критериев эффективности можно привести несколько.

1. Все ошибки в первичном ключе должны быть исправлены с вероятностью, сколь угодно близкой к единице (экспоненциально близкой к единице по параметру, который выбирается легитимными пользователями). Если после «чистки» в ключе остаются ошибки, этот ключ придется отбросить и начать процесс заново.

2. Алгоритм должен выдавать подслушивателю как можно меньше информации о ключе. В среднем на каждый бит ключа приходится порядка 10^3–10^4 посланных состояний, так что каждый бит дорог.

3. Время выполнения алгоритма должно быть минимально. В нашем случае это не очень важно, так как передача квантовых состояний занимает намного большее время, чем последующая их обработка.

4. Необходимо минимизировать сетевой трафик. Обеспечить неизменность (не модифицируемость) посланных по сети (открытому классическому каналу связи) бит в процессе коррекции ошибок в первичном ключе. Другими словами, требуется обес-

печить целостность и аутентичность передаваемых данных через открытый канал связи (см. разд. 7).

Существуют два основных класса алгоритмов для распределенной коррекции ошибок: это алгоритмы, основанные на кодах коррекции ошибок (кодирование «вперед»), и итерационные алгоритмы, основанные на проверках четности, которые используют двусторонние обмены информацией между легитимными пользователями. Коды коррекции ошибок дают меньшее время выполнения и меньший трафик (по крайней мере, по числу посланных по сети пакетов). Итерационные алгоритмы более эффективны — они выдают меньше информации, но требуют пересылки большего количество пакетов маленького размера (по несколько бит) по сети и обычно работают дольше[1)].

Наиболее простая итерационная процедура коррекции ошибок — это бисективный поиск ошибок, который сводится к разбиению первичного ключа на случайные непересекающиеся блоки и вычислению четностей этих блоков. Четности множеств сравниваются как на приемном, так и на передающем конце через открытый канал. После раскрытия четности какого-либо множества один из случайно выбранных бит отбрасывается. При несовпадении четностей размер блока уменьшается вдвое, и процесс повторяется. Поскольку блоки не пересекаются, выбрасывание бит не представляет труда. Такая процедура сохраняет конфиденциальность — подслушиватель не получает дополнительной информации при «чистке» первичного ключа. Однако такая процедура крайне неэффективна, поскольку остается достаточно мало бит в «очищенном» ключе (например, при вероятности ошибки в 10 % в первичном ключе в «очищенном» ключе остается не более 10 % от исходной длины).

Наиболее эффективным в смысле длины «очищенного» ключа на сегодняшний день, по-видимому, является каскадный метод коррекции ошибок, впервые предложенный в работе [15]. В исходном варианте каскадного метода биты четности отдельных и в общем случае пересекающихся подмножеств запоминаются и используются на следующих проходах. В процессе работы метода никакие биты не выбрасываются, поэтому метод не сохраняет конфиденциальность. Оценить информацию, которую получает подслушиватель, когда раскрываются биты четности набора возникающих на разных проходах пересекающихся подмножеств, достаточно сложно.

Сохранить конфиденциальность и эффективность метода можно, если в конце «чистки» первичного ключа отбрасывать некоторое количество бит. Поскольку возникающие на каждом проходе множества бит пересекаются, процедура выбрасывания не является тривиальной. В нашей работе [16] был предложен простой регулярный способ сохранения конфиденциальности.

4. КАСКАДНЫЙ МЕТОД КОРРЕКЦИИ ОШИБОК (CASCADE)

4.1. Исходные данные

1. «Сырой ключ» (первичный ключ — raw key) у Alice и Bob содержит ошибки. Количество ошибок зависит от наличия подслушивателя и свойств канала, но не должно превышать критического значения, в этом случае гарантируется извлечение секретного ключа. Величина критической ошибки зависит не только от используемого квантового протокола распределения ключей, но и от эффективности процедуры коррекции ошибок через открытый канал связи[2)].

2. Вероятность ошибки в бите ключа равна p.

4.2. Результат

Ключ, не содержащий ошибок, получается с вероятностью, не меньше чем $1-2^{-M}$ (в нашем примере $M = 30$).

4.3. Алгоритм

Алгоритм Cascade заключается в четырех проходах по ключу. На каждом проходе ключ случайным образом разбивается на блоки определенного размера с помощью хэш-функций и в этих блоках ищутся ошибки по описанному ниже алгоритму. В результате работы алгоритма подслушиватель может получить биты четности для некоторого набора

[1)] Хотя по сравнению со временем передачи квантовых состояний время работы алгоритма все равно существенно меньше.

[2)] Точнее, от избыточности процедуры коррекции ошибок – информации, выдаваемой в открытый канал связи. Например, для коррекции ошибок Bob может послать всю свою битовую строку через открытый канал к Alice. При этом все ошибки будут исправлены, но всю строку придется отбросить. Избыточность такой процедуры равна единице. Данный пример показателен, потому что из него явно видно, что критическая ошибка зависит не только от квантовой части протокола, но и от эффективности процедуры коррекции ошибок. Даже при сколь угодно малой вероятности ошибки невозможно будет извлечь секретный ключ.

множеств бит ключа. Эта информация удаляется из ключа с помощью алгоритма, описанного в разд. 5.

4.3.1. Определение размеров блоков

Обозначим через N_k размер блока на k-м проходе. Исходя из вероятности ошибки на бит ключ разбивается на блоки определенного размера. Размер блока N_1 на первом проходе находится следующим образом [15].

Обозначим через $\delta_1(j)$ вероятность того, что после первого прохода в блоке останется $2j$ ошибок:

$$\delta_1(j) = C_{N_1}^{2j} p^{2j} (1-p)^{N_1-2j} + C_{N_1}^{2j+1} p^{2j+1} (1-p)^{N_1-2j-1}.$$

Обозначим через E_1 математическое ожидание числа ошибок после первого прохода:

$$E_1 = 2 \sum_{j=1}^{\left[\frac{N_1}{2}\right]} j\delta_1(j) = N_1 p - \frac{(1-(1-2p)^{N_1})}{2}.$$

Присвоим N_1 значение максимального целого числа, удовлетворяющего следующим неравенствам:

$$\sum_{l=j+1}^{\left[\frac{N_1}{2}\right]} \delta_1(l) \le \frac{1}{4}\delta_1(j),\ 1 \le j < \left[\frac{N_1}{2}\right],$$

$$E_1 \le -\frac{\ln\frac{1}{2}}{2}.$$

Далее, пусть $N_i = 2N_{i-1}$. В этом случае вероятность ошибки экспоненциально убывает с числом проходов [15].

4.3.2. Выбор хэш-функций при разбиении на блоки

Единственное требование к хэш-функциям состоит в том, чтобы они обеспечивали максимально близкий к случайному выбор бит для отдельных блоков. Хэш-функции удобно выбрать в виде [12]

$$f_k(i) = ((mx+n) \mod p) \mod b,$$

где $b = N/N_k$ — число блоков, p — простое число, большее N, а числа m и n выбираются случайным образом, так что $0 < m < p$ и $0 \le n < p$. Такие функции называются универсальными случайными однородными хэш-функциями второго рода.

Bob генерирует восемь случайных 32-х битных чисел (по два числа на функцию, четыре функции по числу проходов) и посылает их Alice (рис. 3).

4.3.3. Разбиение ключа на блоки

На каждом проходе алгоритма ключ разбивается на блоки с помощью соответствующей хэш-функции $f_k(i)$: в блок с номером m попадают биты, для номеров которых выполняется равенство

$$f_k(i) = m.$$

4.3.4. Сравнение бит четности

Alice и Bob вычисляют биты четности для каждого блока B_j на текущем проходе:

$$S^j = \bigoplus_{i \in B_j} b_i.$$

Alice посылает полученные биты четности Bob (рис. 4). Bob складывает их со своими и посылает результат Alice (рис. 5). При этом подслушиватель получает N/N_k бит информации о ключе.

4.3.5. Поиск ошибок на текущем проходе

Если у блока с номером i не совпадают биты четности, то это означает, что в этом блоке нечетное число ошибок. Пусть m — число таких блоков. Alice и Bob находят по одной ошибке в каждом из этих блоков бисективным поиском, выдавая Eve $m \log N_k$ бит информации.

1. Для каждого такого блока Alice посылает Bob четность первой половины блока (рис. 6).

2. Bob складывает биты четности со своими и возвращает результат Alice (рис. 7). Если четности не совпали, значит ошибка в первой половине блока, иначе — во второй.

3. Alice делит половину блока с ошибкой пополам и снова посылает Bob четность первой половины. Так продолжается до тех пор, пока ошибка не будет локализована. Тогда Bob изменяет значение ошибочных бит.

4.3.6. Поиск ошибок на предыдущих проходах

Заметим, что на первом проходе этот шаг пропускается.

1. Если Bob изменил значение бита b_i на k-м проходе, это означает, что у всех блоков, содержавших

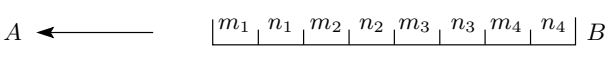

Рис. 3.

этот бит, изменилась четность. Alice и Bob составляют множество блоков M, состоящее из всех блоков на проходах $1, \ldots, k-1$, у которых изменилась четность.

2. Находится наименьший номер прохода, в котором есть блоки, поменявшие четность. Во всех блоках этого прохода делением пополам ищутся ошибки так же, как на шаге, описанном в разд. 4.3.5.

3. Составляется новое множество M, состоящее уже из блоков на проходах $1, \ldots, k$. Если это множество не пусто, переходим к шагу 2.

4.3.7. Окончание прохода

Если еще не прошло четыре прохода, то Alice и Bob увеличивают размер блока вдвое, увеличивают номер прохода на единицу и переходят к шагу, описанному в разд. 4.3.3.

4.3.8. Проверка идентичности «очищенного» ключа

Перед использованием ключа нужно убедиться, что все ошибки исправлены. Alice и Bob генерируют M (в нашем примере $M = 30$) одинаковых случайных битовых строк, по длине равных длине ключа. Alice вычисляет скалярные произведения строк на ключ

$$S_i = \bigoplus_{i=1}^{N} b_i \wedge l_i$$

и посылает полученные M бит Bob (рис. 8). Bob сравнивает их со своими, и, если они совпадают, ключ принимается. В противном случае ключ отбрасывается (рис. 9).

Пусть в ключе остались ошибки. Обозначим один из ошибочных бит за b_i. Тогда все возможные случайные строки разбиваются на два равных класса:

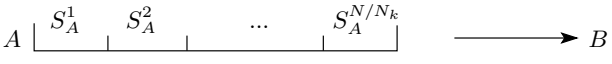

Рис. 4.

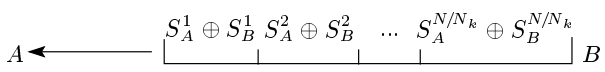

Рис. 5.

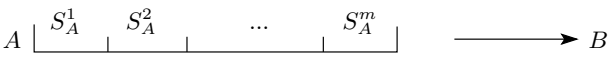

Рис. 6.

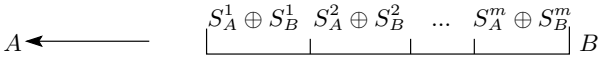

Рис. 7.

Рис. 8.

строки, в которых бит с номером i равен единице, и строки, в которых этот бит равен нулю. Если в строке этот бит равен единице, скалярные произведения строки на ключ у Alice и Bob не совпадут. Следовательно, вероятность обнаружить ошибку таким способом равна $1/2$. Поскольку строки независимы, вероятность того, что ключ с ошибками пройдет проверку, равна 2^{-M}. Иначе говоря, в случае принятия ключа как одинакового у Alice и Bob, вероятность того, что он у них различается в точности, равна $1 - 2^{-M}$.

4.4. Статистика

На рис. 10 приведена зависимость числа раскрытых бит от наблюдаемой вероятности ошибки на приемном конце.

5. УДАЛЕНИЕ БИТ ЧЕТНОСТИ

В принципе можно обойтись и без этого шага и удалять всю выданную информацию на последней стадии усиления секретности «очищенного» ключа с помощью хэширования, но поскольку существует детерминированный алгоритм, гарантированно удаляющий всю информацию, мы предпочли выделить его в отдельный шаг.

5.1. Исходные данные

Исходные данные состоят в следующем.

1. Ключ длины N.

2. Набор множеств бит ключа $X = X_1, \ldots, X_m$, у которых Eve известны биты четности.

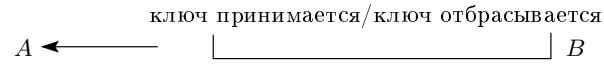

Рис. 9.

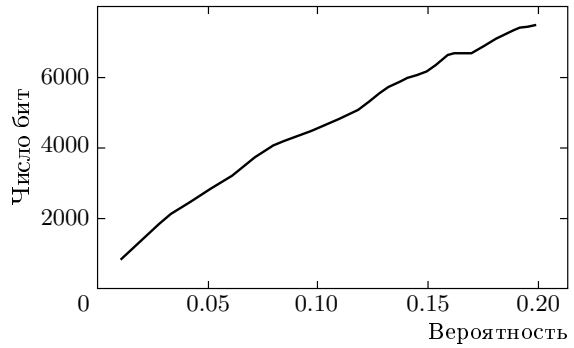

Рис. 10. Зависимость числа переданных по открытому каналу бит от вероятности ошибки (для ключа длиной 10000 бит)

5.2. Результат

В результате получается новый ключ, о котором Eve имеет ту же информацию, которую она имела до процесса «чистки» ключа.

5.3. Алгоритм

5.3.1. Предварительная обработка множеств

Рассмотрим набор множеств X, полученный в ходе работы алгоритма, описанного в разд. 4. В нашем примере он состоит из $M = 30$ множеств, полученных на шаге, описанном в разд. 4.3.8, множеств, полученных в результате разбиения ключа на блоки (разд. 4.3.3), и множеств, полученных в результате поиска ошибок (разд. 4.3.5).

В процессе поиска одной ошибки генерируется набор вложенных множеств (рис. 11). Их можно заменить на эквивалентный набор непересекающихся множеств (рис. 12). После этой операции множества, полученные на каждом из четырех проходов, не будут пересекаться и каждый бит ключа будет принадлежать не более чем четырем множествам (не считая $M = 30$, полученным на шаге проверки ключа).

Рис. 11.

Рис. 12.

Переупорядочим множества $X_1, \ldots, X_m$, расположив их по возрастанию мощности:

$$|X_1| < |X_2| < \ldots < |X_m|.$$

5.3.2. Составление матрицы

Составим битовую матрицу M из N строк и m столбцов. Бит M_i^j равен единице, если i-й бит ключа принадлежит множеству X_j, и нулю в противном случае. Полученная матрица сильно разрежена, и большая часть единиц сконцентрирована в правой части.

5.3.3. Нахождение линейно-независимых векторов

Нам нужно найти такой набор бит, что, придавая этим битам различные значения, можно получить все возможные наборы четностей $p_1, \ldots, p_m$. Изменяя значение бита b_i, мы изменим четность всех множеств, у которых $V_k^i = 1$. Изменяя значения нескольких бит $b_{i_1}, \ldots, b_{i_l}$, мы изменим четность всех множеств X_j, таких что бит j вектора $V_{i_1} + \ldots + V_{i_l}$ равен единице. Следовательно, нам нужно найти максимальный набор линейно-независимых векторов V_i и удалить соответствующие биты.

Данная процедура реализуется методом Гаусса с запоминанием перестановки строк. Для того чтобы сохранить разреженность матрицы, на каждом шаге в качестве ведущей строки выбирается строка с наименьшим весом.

5.3.4. Удаление бит

Все биты, которым соответствуют линейно-независимые строки, удаляются из ключа. После этой стадии гарантируется, что подслушиватель не может иметь бóльшую информацию об «очищенном» ключе, чем он имел до этой стадии. До этой стадии подслушиватель имел некоторую информацию, которую он получил из квантового канала связи при передаче информационных квантовых состояний. Получение этой информации неизбежно приводило к возмущениям передаваемых квантовых состояний и возникновению ошибок на приемной стороне у Bob. Теперь эти ошибки исправлены легитимными пользователями, а информация подслушивателя при этом не увеличилась.

6. УСИЛЕНИЕ СЕКРЕТНОСТИ КЛЮЧА (СЖАТИЕ УНИВЕРСАЛЬНЫМИ ОДНОРОДНЫМИ ХЭШ-ФУНКЦИЯМИ)

Процедура усиления секретности [11] сводится к сжатию «очищенного» ключа при помощи хэш-функций специального вида (так называемые универсальные однородные хэш-функции второго рода (two-universal hash functions [12])). По определению, множество таких функций $\{G\}$ состоит из универсальных однородных хэш-функций второго рода

$$y = g(x) : \{0,1\}^N \to \{0,1\}^R \ (x \in \{0,1\}^N, y \in \{0,1\}^R),$$

если вероятность того, что при разных x_1 и x_2 и случайном выборе g из $\{G\}$ в соответствии с равномерным распределением, удовлетворяет условию

$$\Pr(g(x_1) = g(x_2)) \le 2^{-R}.$$

Отметим, что сама хэш-функция является случайной величиной. Удобно реализовать такие функции на основе модулярной арифметики полиномов над полем $GF(2)$ (см. ниже в разд. 6.1).

После сжатия «очищенного» ключа такими хэш-функциями распределение ключей является сколь угодно близким к равномерному (точнее, экспоненциально близким по любому наперед заданному параметру секретности).

Если при коррекции ошибок используется метод с выбрасыванием раскрытых проверочных бит, то после коррекции ошибок информация подслушивателя об «очищенном» ключе не изменяется. То есть подслушиватель имеет лишь ту информацию, которую он получил о ключе из квантового канала связи. Для устранения этой информации о ключе (точнее, для уменьшения количества бит до любой наперед заданной величины) используется сжатие ключа. Для определения степени сжатия ключа необходимо иметь надежную оценку верхней границы количества бит информации, которую может иметь подслушиватель, о ключе при наблюдаемом потоке ошибок p на приемном конце у Bob.

Кроме того, степень сжатия «очищенного» ключа зависит от эффективности процедуры исправления ошибок, точнее, от ее избыточности. Использование случайных кодовых слов (шенноновский предел) позволяет получить максимальное количество бит информации, равное $nH(p)$ (здесь n — длина последовательности, $H(p) = 1 + p\log p + (1-p)\log(1-p)$, p — вероятность ошибки). Соответственно, избыточность случайного кода равна $n(1 - H(p))$. Шенноновский случайный код обладает максимальным минимальным расстоянием (минимальной избыточностью) по сравнению с другими. Другими словами,

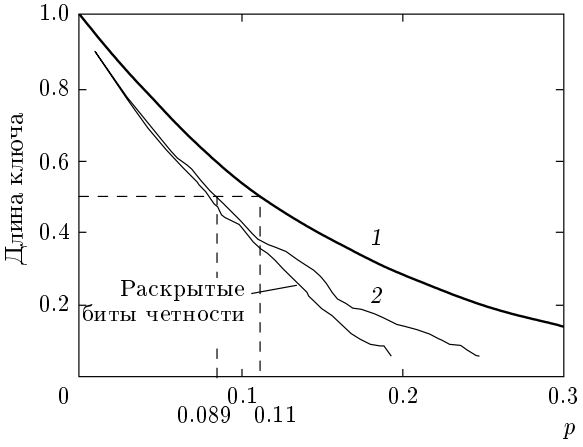

Рис. 13. Зависимость конечной длины ключа от вероятности ошибки. Для сравнения приведена избыточность (отброшенные биты четности) для случайного кода Шеннона (*1*) и процедуры Cascade (*2*)

раскрытая информация в битах при коррекции ошибок не может быть меньше, чем $n(1 - H(p))$. Чем больше избыточность кода, исправляющего ошибки, тем меньше длина финального ключа, а также тем меньше критическая величина ошибки, до которой гарантируется секретность ключа.

На рис. 13 для сравнения приведена избыточность для шенноновских случайных кодов и процедуры Cascade с выбрасыванием.

В качестве примера оценим длину финального ключа при потоке ошибок p для квантового протокола распространения ключей $BB84$ [7].

Воспользуемся следствием из фундаментальной теоремы об усилении секретности [11]. Пусть

$$E : \{0,1\}^{n_{AB}} \to \{0,1\}^t$$

— произвольная функция, описывающая стратегию Eve в том смысле, что для произвольной строки бит длиной n_{AB} Eve известно не более t бит. Здесь n_{AB} — битовая строка у Alice и Bob после исправления ошибок. Пусть s — параметр секретности ($s < n_{AB} - t$). Далее пусть

$$G : \{0,1\}^N \to \{0,1\}^R$$

— универсальная однородная функция хэширования второго рода [11, 12], которая сама является случайной величиной. Тогда взаимная информация Eve о секретном ключе $K = G(X)$

$$I(K; GZ) \le \frac{2^{-s}}{\ln 2}, \qquad (2)$$

где $Z : \{0,1\}^t$ — строка Eve, согласованная со строкой легитимных пользователей X в том смысле, что эта строка могла произойти из строк X как $Z = E(X)$. Наша цель состоит в вычислении длины финального ключа R, о котором Eve имеет экспоненциально малую по s информацию. Для этого потребуется знание энтропии Реньи второго рода, которая в свою очередь выражается через вероятность коллизий.

В случае квантовой криптографии (протокол $BB84$) Eve может различить не более $2^{n\overline{C}(\rho)/2}$ строк ($\overline{C}(\rho)$ — классическая пропускная способность квантового канала связи [17]) из общего множества $2^{nH_{Cascade}(p)}$ (здесь $2^{nH_{Cascade}(p)}$ — число возможных строк у Alice и Bob после исправления ошибок процедурой Cascade). Другими словами,

$$n_{AB} = nH_{Cascade}(p)$$

— длина битовой строки, которая остается после исправления ошибок процедурой Cascade с выбрасыванием (здесь n — длина строки до исправления ошибок).

Условная вероятность определяется отношением общего числа строк к числу областей декодирования у Eve:

$$P(X|Z=z) = \frac{2^{n\overline{C}(\rho)/2}}{2^{nH_{Cascade}(p)}} = \\ = 2^{-n(H_{Cascade}(p)-\overline{C}(\rho)/2)} = a_z. \quad (3)$$

Фактически $1/a_z$ — доля строк, таких что $z = E(X)$. То есть с каждой частичной строкой Eve согласовано множество строк, определяемое формулой (1). Для вероятности коллизий получаем

$$P_c(X|Z=z) = \sum_{X:\{z=E(X)\}} P^2(X|Z=z) = \\ = 2^{-n(H_{Cascade}(p)-\overline{C}(\rho)/2)} = \frac{1}{a_z}a_z^2. \quad (4)$$

Энтропия Реньи второго рода равна

$$R(X|Z=z) = -\log P_c(X|Z=z) = \\ = n\left(H_{Cascade}(p) - \frac{\overline{C}(\rho)}{2}\right). \quad (5)$$

Согласно теореме об усилении секретности [11], находим

$$H(K|G,Z=z) \geq R - \frac{2^{r-R(X|Z=z)}}{\ln 2} > \\ > R - \frac{2^R}{a_z \ln 2}. \quad (6)$$

Для взаимной информации между строками Eve и секретным ключом у Alice и Bob, с учетом того, что

$$P_Z(z) = 2^{-n\overline{C}(\rho)/2} = 2^{-nH_{Cascade}(p)}/a_z,$$

имеем

$$I(K;GZ) = H(K) - H(K|GZ) \leq \\ \leq R - \sum_{z\in\{0,1\}^t} P_Z(z)H(K|G,Z=z) \leq \\ \leq \sum_{z\in\{0,1\}^t} a_z 2^{-nH_{Cascade}(p)}\frac{2^R}{a_z \ln 2} = \\ = \frac{2^{-nH_{Cascade}(p)+t+R}}{\ln 2} = \frac{2^{-s}}{\ln 2}. \quad (7)$$

Секретный ключ

$$K = G(X) \in \{0,1\}^R$$

имеет длину

$$R = n\left(H_{Cascade}(p) - \frac{\overline{C}(\rho)}{2}\right) - s. \quad (8)$$

Для протокола $BB84$ $\overline{C}(\rho) = 1$, поэтому «чистка» первичного ключа процедурой Cascade с выбрасыванием обеспечивает секретность финального ключа, если процент ошибок не превышает $p_{Cascade} \approx 8.9\,\%$. При этом $p_{Cascade}$ определяется как корень уравнения

$$H_{Cascade}(p_{Cascade}) = \frac{\overline{C}(\rho)}{2}.$$

При коррекции ошибок случайными кодовыми словами (шенноновский предел), например, протокол $BB84$ секретен до $p_c \approx 11\,\%$ [18–20].

Перейдем к описанию реализации процедур сжатия ключа.

6.1. Основной алгоритм

6.1.1. Исходные данные

Исходные данные состоят в следующем.

1. Ключ K длины N.
2. Длина финального (хэшированного) ключа R.

6.1.2. Результат

В результате имеем fK — финальный ключ.

6.1.3. Алгоритм

1. Выбираем неприводимый полином $P(x)$ степени $N+1$ из таблицы. Полиномы проверялись на неприводимость по алгоритму, описанному в разд. 6.3.

2. Генерируем случайную строку бит X длины N.

3. Перемножаем K и X как полиномы над полем $GF(2)$. Сложность соответствует $O(N^{\log 3})$, согласно методу Карацубы (см. ниже разд. 6.2).

4. Вычисляем первые R бит частного KX/P в поле $GF(2)$. Это и есть новый ключ. Сложность соответствует $O(R)$ из-за того, что в $P(x)$ мало единиц (три или пять).

6.2. Метод Карацубы

6.2.1. Исходные данные

Имеем A, B — полиномы степени $2n$.

6.2.2. Результат

X — произведение AB.

6.2.3. Алгоритм

1. Представляем полиномы A и B в виде

$$A = x^n A_0 + A_1, \tag{9}$$

$$B = x^n B_0 + B_1. \tag{10}$$

2. Представляем произведение AB следующим образом:

$$AB = (x^n A_0 + A_1)(x^n B_0 + B_1) = (x^n + x^{2n}) A_0 B_0 - \\ - x^n (A_1 - B_0)(B_1 - A_0) + (X^n + 1) A_1 B_1. \tag{11}$$

Заметим, что в этом выражении имеются три операции умножения полиномов степени n, а также операции сдвига и XOR.

3. Рекурсивно продолжаем процесс, пока степень умножаемых многочленов не станет достаточно маленькой.

4. Произведения полиномов маленькой степени определяем по заранее сгенерированной таблице умножения.

6.3. Проверка полинома на неприводимость

6.3.1. Исходные данные

Имеем полином $P(x)$ степени n.

6.3.2. Результат

1, если полином неприводим, иначе 0.

6.3.3. Алгоритм

Проверяем, что

$$x^{2^d} - x \equiv 1 \pmod{P(x)} \; \forall \; d \in (1, n-1)$$

следующим образом.

1. Инициализация:

$$u = x, \quad i = 1.$$

2. Возводим u в квадрат:

$$u = u^2 \bmod P(x).$$

3. Вычисляем

$$g = gcd(u - x, P(x)).$$

Если $g \neq 1$, то полином приводим, иначе переходим к следующему шагу.

4. Увеличиваем i на единицу, если $i = n$, то полином неприводим, иначе возвращаемся к шагу 2.

6.4. Таблица неприводимых полиномов

Из полной таблицы приведем для примера неприводимые полиномы для нескольких степеней:

1. $n = 100$ — $P(x) = x^{100} + x^{15} + 1$;
2. $n = 1000$ — $P(x) = x^{1000} + x^5 + x^4 + x^3 + 1$;
3. $n = 5000$ — $P(x) = x^{5000} + x^{17} + x^{15} + x^7 + 1$;
4. $n = 10000$ — $P(x) = x^{10000} + x^{19} + x^{13} + x^9 + 1$.

7. ЦЕЛОСТНОСТЬ И АУТЕНТИЧНОСТЬ ДАННЫХ

7.1. Общая идеология открытого классического канала связи

Кратко опишем методы сохранения целостности и аутентичности информации, передаваемой по открытому классическому каналу связи[3)].

[3)] Физически открытые (доступные для подслушивателя) квантовый канал связи и классический канал связи могут представлять собой одну и ту же оптоволоконную линию. Единственное отличие состоит в том, что когда линия работает как квантовый канал, через нее передаются однофотонные состояния (ослабленное лазерное излучение до среднего числа фотонов 0.1 в импульсе на длине волны 1550 нм). Если канал работает для передачи лазерных импульсов для синхронизации на длине волны 1310 нм или для передачи вспомогательных данных, например, при коррекции ошибок, то канал связи работает как классический.

Вся вспомогательная информация, передаваемая легитимными пользователями по открытому классическому каналу, считается доступной и известной подслушивателю. Единственное требование, предъявляемое к открытому классическому каналу связи, состоит в том, что передаваемая информация не может быть модифицирована за время передачи вспомогательной информации (фактически несколько секунд). Применительно к квантовой криптографии такой канал называют unjamable channel.

В классической криптографии обеспечение целостности данных достигается при помощи криптографических хэш-функций. Общая идея использования хэш-функций для сохранения целостности информации сводится к тому, что для каждого набора данных вычисляется значение хэш-функции (обычно это значение называют кодом аутентификации сообщений или имитовставкой (КАС)). Удобно использовать ключевые хэш-функции на основе ГОСТа Р 34.11-94 [10]. Для такой функции требуется небольшой общий для легитимных пользователей стартовый ключ (256 бит). В принципе возможно применение бесключевых хэш-функций с использованием асимметричных криптографических схем. Однако, если в качестве квантового и классического каналов связи физически используется одна и та же оптоволоконная линия, такой подход оказывается неприемлемым из-за слабости по отношению к атаке «man in the middle».

Первое требование к каналу состоит в том, что нелегитимный пользователь не может изменять информацию, проходящую через канал. Вторым требованием является невозможность перепосылки нелегитимным пользователем посланной ранее информации. Все эти требования выполняются путем применения хэширующих и шифрующих функций классической криптографии. Передача по классическому каналу сводится к формированию пакетов из потока передаваемых данных (рис. 14*а*).

Как отмечалось выше, для обеспечения неизменности данных применяется хэширование. Процесс хэширования сводится к применению алгоритма на основе ГОСТа Р 34.11-94 к пакетам данных. Результаты записываются в поле «Хеш». Для использования алгоритма ГОСТа Р 34.11-94 требуется секретный ключ размером 256 бит, полученный из предыдущих сессий передачи ключа или заданный заранее (рис. 14*б*).

Хеширование не обеспечивает защиту информации от перепосылки пакетов, поэтому в тело пакета добавлена служебная информация, содержащая в себе номер текущего пакета. Служебная информация хэшируется, как и остальные данные пакета, так что нелегитимный пользователь не может ее изменить (рис. 14*в*). Фактически данная служебная информация является уникальным номером пакета и аналогична метке времени. Такие метки имеют «сквозную» случайную нумерацию до следующей смены ключей по квантовому каналу связи.

Шифрование не является необходимым требованием, применяемым для классического канала в квантовой криптографии, однако для универсальности программного обеспечения удобно реализовать шифрование по ГОСТу 28147-89 в режиме сцепления блоков (cypher block chaining).

Вспомогательная информация, которая используется при коррекции ошибок, балансировка интерферометра и т. д., и собственно сами конфиденциальные сообщения обрабатываются той же самой программой, но при этом используются разные ключи. Поэтому в пакет также записывается идентификатор ключевой пары, применяемый для хэширования и шифрования. Идентификатор хэшируется, как и остальные данные пакета, но не шифруется (рис. 14*в*), что позволяет легитимному пользователю выбрать правильную ключевую пару.

7.2. Требования к обмену данными по открытому каналу связи

Математическое и программное обеспечение для передачи данных по открытому классическому каналу связи должно обеспечивать следующую функциональность.

1. Обеспечивать создание соединения между двумя легитимными пользователями.

2. Содержать сервисное программное обеспечение для вспомогательных целей (настройки приемной части интерферометра, тестирования электроники и т. д.).

3. Обеспечивать коррекцию ошибок в первичном ключе, переданном по квантовому каналу связи, а также его дальнейшее сжатие легитимными пользователями до финального секретного ключа по установленному соединению.

4. Обеспечивать передачу текстовых сообщений в режиме электронной почты с хэшированием и шифрованием на ключах, полученных по квантовому каналу связи.

5. Обеспечивать передачу конфиденциальных сообщений (файлов) с использованием ключей, переданных по квантовому каналу связи.

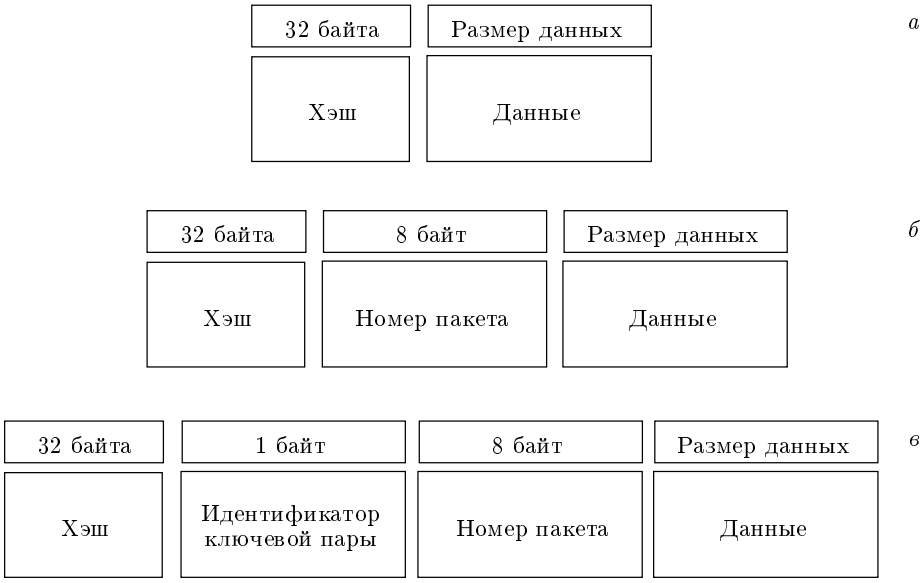

Рис. 14.

7.2.1. Создание соединения между двумя легитимными пользователями

Для обеспечения передачи данных создается соединение на уровне «Приложений и служб» стека протоколов TCP/IP (Transmission Control Protocol/Internet Protocol). Данными может быть как вспомогательная информация при генерации ключей, так и зашифрованная информация на ключах, переданных по квантовому каналу. Для установки соединения используется протокол управления передачей с контролем доставки TCP. Используется клиент-серверная модель программы.

Программно-реализованный алгоритм соединения выглядит следующим образом.

1. Сервер инициализируется и становится готовым к соединению с клиентами. Клиенты могут подсоединяться к серверу на порт, открытый при инициализации.

2. Клиент запрашивает у TCP открытия соединения с сервером по указанному IP-адресу и порту.

3. Клиентская TCP получает начальный порядковый номер и посылает сегмент синхронизации.

4. Когда поступает сегмент синхронизации, серверная TCP получает свой начальный порядковый номер и посылает свой сегмент синхронизации клиенту.

5. Клиентская TCP, получив от сервера сегмент синхронизации, посылает номер первого посланного сервером байта.

6. Клиентская TCP указывает программному обеспечению на открытие соединения.

7. Серверная TCP указывает программному обеспечению на открытие соединения.

7.2.2. Обеспечение целостности и аутентичности передаваемых данных

Предполагается, что в момент первого запуска системы клиент и сервер (Alice и Bob) имеют два одинаковых стартовых секретных ключа по 256 бит, которые используются при обмене открытой вспомогательной информацией при генерации ключей по квантовому каналу связи. Один ключ используется для шифрования данных, а второй — для хэширования.

После генерации по квантовому каналу новых секретных ключей два начальных ключа выбрасываются и заменяются двумя новыми при каждом сеансе генерации ключей. Остальные секретные ключи, также полученные из квантового канала связи, используются уже для шифрования сообщений до следующего сеанса генерации новых ключей, когда в этом возникнет необходимость. При этом используется общий алгоритм шифрования и хэширования как для вспомогательных данных, так и конфиденциальных информационных сообщений.

Для открытой вспомогательной информации не требуется шифрования, достаточно использовать хэширование, обеспечивающее целостность и аутентичность открытых данных. Однако для дости-

жения универсальности программного обеспечения при передаче открытой вспомогательной информации используется шифрование. Такой подход обеспечивает как секретность, так и целостность и аутентичность всех передаваемых данных. Шифрование и хэширование данных происходит после установки соединения между клиентом и сервером. На основе установленного TCP-соединения создаются несколько виртуальных каналов, каждый из которых использует свой собственный ключ. Таким образом, в каждый текущий момент легитимные пользователи имеют набор секретных ключей, которые используются для обмена по открытому каналу связи.

Реализовано программное обеспечение, работа которого определяется следующим алгоритмом.

1. На уровне приложения формируется поток данных для передачи.

2. Поток данных преобразуется в пакеты по следующему алгоритму.

2.1. Если размер потока данных меньше 65532 байт, то переходим на шаг 2.3.

2.2. Формируем пакет из первых 65532 байт потока данных и удаляем их из потока, переходим на шаг 2.1.

2.3. Если размер потока данных равен нулю, то переходим на шаг 2.6.

2.4. Если размер потока данных при делении на 8 не дает остаток 4, то добавляем в поток данных такое количество случайных байт, чтобы размер потока данных при делении на 8 давал остаток 4.

2.5. Формируем пакет из потока данных.

2.6. Выход.

3. К каждому пакету добавляется поле «Размер», содержащее его размер в байтах.

4. К каждому пакету добавляется поле «Флаги». К первому пакету ставится маркер начала цепочки данных. К последнему пакету ставится маркер окончания цепочки данных. К остальным пакетам ставится маркер середины цепочки данных. Поле «Флаг» может одновременно содержать маркер «Начало» и маркер «Конец», если цепочка данных содержит один единственный пакет.

5. К каждому пакету добавляется системное поле «Идентификатор протокола».

6. К каждому пакету добавляется системное поле «Идентификатор последовательности» (seqID).

7. Полученный пакет шифруется в соответствии с алгоритмом ГОСТа 28147-89 со сцеплением блоков шифротекста с ключом K_1, полученным по идентификатору пользователя и идентификатору ключевой пары из внутрипрограммного сервиса распространения ключей.

8. К каждому пакету добавляется поле «Идентификатор ключевой пары».

9. К каждому пакету добавляется его значение хэш-функции, полученной по алгоритму ГОСТа Р 34.11-94 с ключом K_2, полученным по идентификатору пользователя и идентификатору ключевой пары из внутрипрограммного сервиса распространения ключей.

Формирование пакета данных на передающей стороне представлено на рис. 15. Разворачивание пакета данных на приемной стороне представлено на рис. 16.

7.2.3. Описание формата данных

Поскольку система квантовой криптографии является распределенной системой, все служебные обмены информацией происходят через тот же самый открытый канал связи, т. е. могут реализовываться различные типы протоколов: пересылка зашифрованных сообщений, коррекция ошибок, балансировка интерферометра Маха–Цендера на приемной и передающей сторонах и т. д. Для этого введено специальное поле.

«Идентификатор протокола» — поле, отвечающее за текущий протокол. Реализовано три типа протоколов обмена.

1. Протокол передачи данных «Идентификатор протокола 1». Используется для передачи текстовых сообщений или файлов между легитимными пользователями. Формат потока данных следующий.

1.1. Номер команды.

1.1.1. Передача текстового сообщения — 1.

1.1.2. Передача файла — 2.

1.2. Размер пересылаемых данных (текстового сообщения или файла) в байтах.

1.3. Пересылаемые данные (текстовое сообщение или файл).

2. Протокол балансировки приемной части интерферометра, «Идентификатор протокола 2».

3. Протокол коррекции ошибок в первичном ключе, сгенерированном по квантовому каналу связи и сжатом в дальнейшем до финального секретного ключа, «Идентификатор протокола 3».

«Идентификатор ключевой пары» — поле, указывающее на то, какая пара ключей используется.

«Идентификатор последовательности» (seqID) — данный идентификатор по сути играет роль метки времени для пакета данных и гарантирует невозможность для подслушивателя вторичной переposылки пакетов.

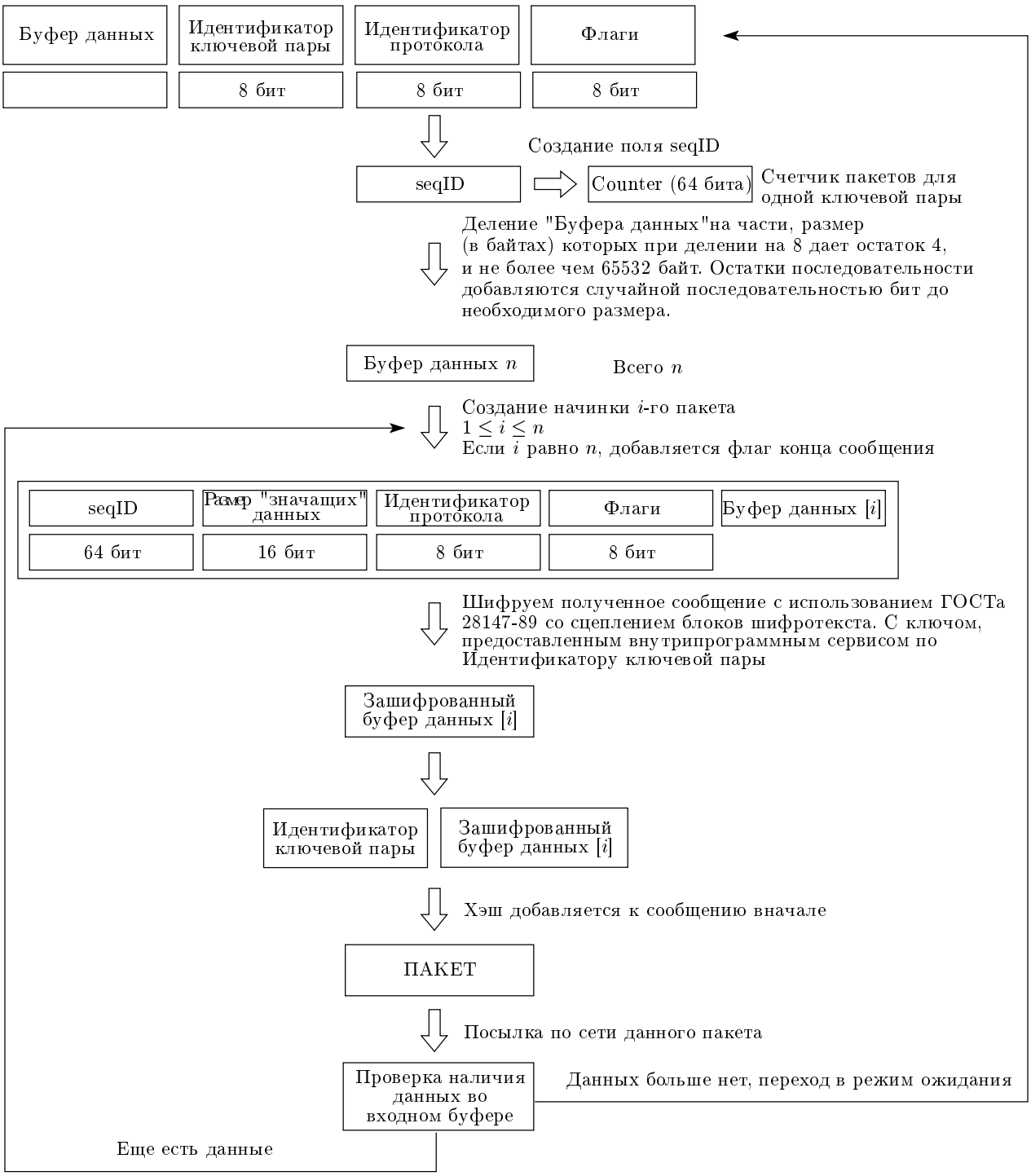

Рис. 15. Внутренняя структура пакета данных при передаче («сборка» пакета данных)

Поскольку в каждый момент времени у легитимных пользователей имеется набор секретных ключей, они могут сгенерировать пару идентичных псевдослучайных чисел (в нашей реализации при помощи алгоритма ГОСТа 28147-89) без обмена информацией через открытый канал связи. Поле seqID имеет длину 64 бита. Первые 32 бита — номер пакета, вторые 32 бита — псевдослучайная последова-

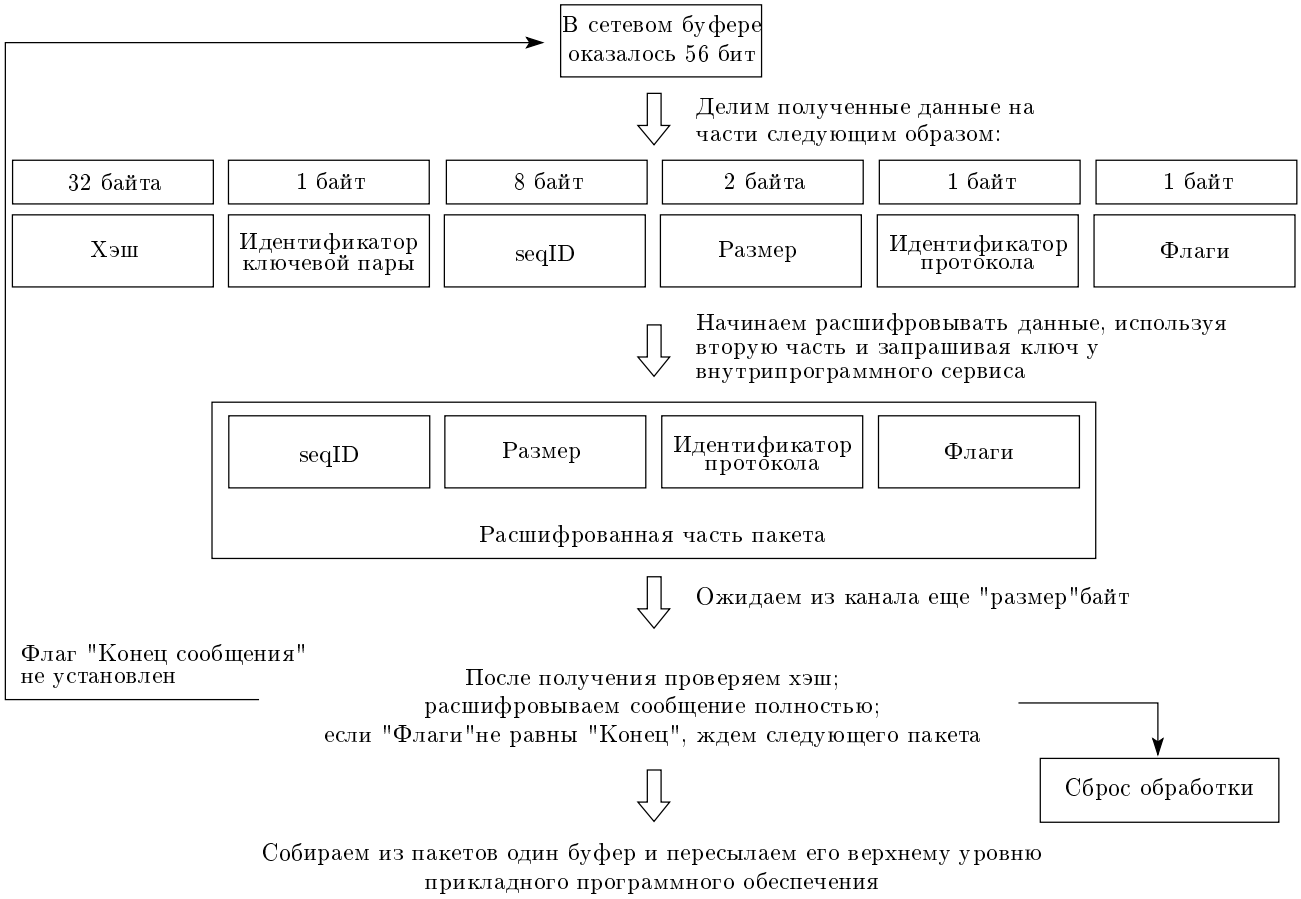

Рис. 16. Внутренняя структура пакета на приемной стороне («разборка» пакета данных)

тельность, играющая роль метки времени. В пакете шифруются следующие поля: seqID, размер значащих данных, идентификатор протокола, флаги, буфер данных (данные).

К зашифрованным данным добавляется поле идентификатора ключевой пары. Фактически это лишь ссылка для приемной стороны на разные пары ключей, которые используются передающей стороной либо для зашифровывания и хэширования вспомогательных данных, либо для зашифровывания и хэширования информационных сообщений. Это поле не шифруется, а лишь хэшируется, поскольку в противном случае приемная сторона не сможет понять, какую пару ключей использовать при расшифровке. Ссылка занимает 8 бит, остальные 56 бит оставлены как резервные. Далее весь пакет подписывается при помощи ключевой функции хэширования по алгоритму ГОСТа Р 34.11.-94.

7.2.4. Схема работы программного комплекса квантовой криптографии

Программный комплекс работает в двух режимах.

1. Сервер ожидает соединения клиентов.

2. Клиент устанавливает соединение с сервером.

Обе части работают одинаково, используя схему программы, предложенную на рис. 17. Предложенная схема реализована в нотации UML (Unified Modeling Language). На рисунке использованы следующие обозначения: прямоугольные блоки с названиями — модули программного комплекса; линии между модулями — логическая связь; «1» и «*» — количество экземпляров данного модуля (один или любое количество), создаваемое по отношению к связанному модулю.

В схеме используются следующие модули. «Service Provider» — абстрактный модуль, созданный для унификации интерфейсов модулей «App», «Clean» и «Data Transfer».

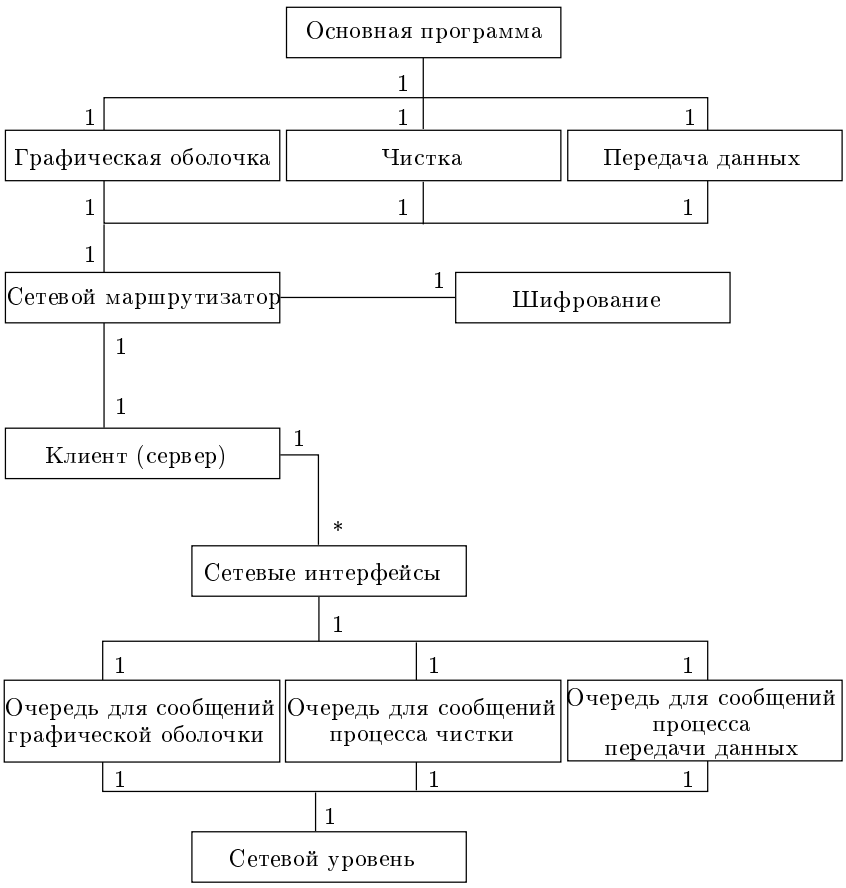

Рис. 17.

«App» — модуль, обеспечивающий управление аппаратной частью комплекса. «Clean» — модуль, обеспечивающий процедуру «чистки» первичного ключа, сгенерированного по квантовому каналу связи. «Data Transfer» — модуль, обеспечивающий передачу текстовых данных и файлов. «Network Broker» — основной модуль, обеспечивающий взаимосвязь уровня передачи данных с графическим интерфейсом пользователя. Данным модулем проводится фильтрация потока данных по клиентам (в случае сервера) и по интерфейсу передачи, обеспечивая параллельную работу разных интерфейсов передачи данных для каждого клиента. Данный модуль управляет модулем «Crypto Provider» и обеспечивает взаимодействие всех последующих уровней приложения с этим модулем. «Crypto Provider» — модуль, обеспечивающий хранение ключевых пар каждого клиента. Данный модуль предоставляет последующим уровням программы необходимые данные для шифрования потока данных. «Server» — модуль, обеспечивающий сетевое соединение клиентов. «Client» — модуль, обеспечивающий сетевое соединение к серверу. «Interface» — модуль, обеспечивающий фильтрацию потока данных по протоколу. «App. Queue» — модуль, обеспечивающий поддержку очереди сообщений для протокола управления и настройки аппаратной части комплекса. «Clean Queue» — модуль, обеспечивающий поддержку очереди сообщений для протокола чистки «сырого» ключа. «Data Transfer Queue» — модуль, обеспечивающий поддержку очереди сообщений для протокола передачи текстовых сообщений и файлов. «Socket» — модуль, отвечающий за передачу потока данных через сетевой интерфейс путем фрагментирования его на пакеты определенной длины. Каждый пакет, используя ключи, полученные от модуля «Crypto Provider», шифруется и хэшируется.

8. ЗАКЛЮЧЕНИЕ

Таким образом, в работе рассмотрены требования и построены процедуры, обеспечивающие

распределенную коррекцию ошибок, сжатие «очищенных» ключей через открытый классический канал связи с сохранением целостности и аутентичности данных.

Работа поддержана государственным контрактом № 02.435.11.1004 Федерального агентства по науке и инновациям (Роснаука) Министерства образования и науки, а также частично РФФИ (проекты №№ 05-02-08306-офи-а и 05-02-17387).

ЛИТЕРАТУРА

1. В. А. Котельников, Отчет (1941).

2. C. E. Shannon, Bell Syst. Tech. J. **28**, 658 (1949).

3. G. S. Vernam, J. Amer. Inst. Elect. Eng. **55**, 109 (1926).

4. S. Wiesner, SIGACT News **15**, 78 (1983).

5. W. Diffie and M. E. Hellman, IEEE Trans. on Inform. Theory **IT-22**, 644 (1976).

6. R. L. Rivest, A. Shamir, and L. Adleman, Commun. ACM **21**, 120 (1978).

7. C. H. Bennett and G. Brassard, *Proc. of IEEE Int. Conf. on Comput. Sys. and Sign. Proces.*, Bangalore, India, December 1984, p. 175.

8. W. K. Wootters and W. H. Zurek, Nature **299**, 802 (1982).

9. C. H. Bennett, Phys. Rev. Lett. **68**, 3121 (1992); C. H. Bennett, G. Brassard, and N. D. Mermin, Phys. Rev. Lett. **68**, 557 (1992).

10. Государственный стандарт Российской Федерации, ГОСТ Р 34.11-94, *Криптографическая защита информации. Функция хэширования.*

11. C. H. Bennett, G. Brassard, C. Crépeau, and U. Maurer, IEEE Trans. on Inform. Theory **41**, 1915 (1995).

12. J. L. Carter and M. N. Wegman, J. Comput. Syst. Sci. **18**, 143 (1979).

13. Д. Кнут, *Искусство программирования для ЭВМ, т. 3, Сортировка и поиск*, Мир, Москва (1978) [D. E. Knuth, *The Art of Computer Programming, v. 3, Sorting and Searching*, Addison-Wiley, Massachusetts (1973)].

14. Государственный стандарт СССР, ГОСТ 28147-89, *Системы обработки информации. Защита криптографическая.*

15. G. Brassard and L. Salvail, Lect. Notes in Comp. Sci. **765**, 410 (1994).

16. А. В. Тимофеев, С. Н. Молотков, Письма в ЖЭТФ **82**, 868 (2005).

17. А. С. Холево, Проблемы передачи информации **8**, 63 (1972); **15**, 3 (1979); УМН **53**, 193 (1998); *Введение в квантовую теорию информации*, сер. *Современная математическая физика, вып. 5*, МЦНМО, Москва (2002).

18. D. Mayers and A. Yao, E-print archives quant-ph/9802025.

19. E. Biham, M. Boyer, P. O. Boykin, T. Mor, and V. Roychowdhury, E-print archives quant-ph/9912053.

20. P. W. Shor and J. Preskill, E-print archives quant-ph/0003004.