ДИФФЕРЕНЦИАЛЬНЫЙ АНАЛИЗ 3 РАУНДОВ ШИФРА КУЗНЕЧИК

Аннотация

В январе 2016 года в Российской Федерации вступил в силу новый стандарт блочного шифрования ГОСТ Р 34.12-2015. В его состав вошли два алгоритма шифрования. Первый шифр Магма был ранее известен под именем ГОСТ28147-89 (или просто ГОСТ). Второй алгоритм получил название Кузнечик. Кузнечик – это новый симметричный алгоритм шифрования, построенный по принципу SP-сети. До сих пор нет публикаций о дифференциальных свойствах алгоритма «Кузнечик». Нами впервые исследованы свойства основных операций и предложен метод дифференциального анализа трех раундов алгоритма шифрования «Кузнечик». Мы исследовали дифференциальные свойства нелинейной функции S и линейной функции L и выяснили, что возможна ситуация, когда 1 ненулевой байт разности в результате функции L разворачивается в 16 ненулевых байт, проходит через блок замены S, и затем сворачивается снова в 1 ненулевой байт. Разработанная схема позволяет затрагивать активный S-блок минимальное количество раз. В результате для предложенной схемы вероятность нахождения правильных пар текстов составляет 2^{-108} зашифрований. Также мы разработали алгоритм нахождения секретного ключа, сложность которого составляет $6*2^{-120}$зашифрований. Таким образом, общая сложность анализа, включая поиск правильных пар текстов и поиск битов секретного ключа шифрования составляет $2^{-108} + 6*2^{-120}$ зашифрований.

В результате выполнения исследований разработан, реализован и протестирован алгоритм шифрования Кузнечик, разработан метод дифференциального анализа трех раундов алгоритма Кузнечик, а также для данного алгоритма разработана схема нахождения правильных пар текстов для дифференциального анализа трех раундов шифрования. Программа выполнены на языке программирования C++ в среде разработки Microsoft Visual Studio C++. Результаты исследований подробно описаны, показаны на примерах, структурированы в хронологической последовательности и наглядно отображены в виде иллюстраций, таблиц и схем в тексте данной статьи. В заключении статьи приведены теоретические расчеты о времени, требуемом для проведения атаки с использованием самых мощных суперкомпьютеров мира при использовании как простой реализации, так и реализации с использованием специальных таблиц предвычислений.

Введение

Алгоритм шифрования Кузнечик был выбран в качестве стандарта ГОСТ Р 34.12-2015 и официально вступил в силу 1 января 2016 года, поэтому исследование его надежности на сегодняшний день является *актуальной задачей*. Описание алгоритма шифрования Кузнечик, как части нового принятого стандарта шифрования содержится в документе технического комитета по стандартизации ТК 26 «Криптографическая защита информации» в ГОСТ Р 34.12–2015 [1]. Есть несколько статей, посвященных различным способам реализации алгоритма Кузнечик, в том числе и с использованием специальных таблиц предвычислений [2]

На сегодняшний день уже можно встретить несколько работ, посвященных анализу надежности шифра Кузнечик. На конференции «CRYPTO 2015» Алекс Бирюков, Лео Перрин и Алексей Удовенко представили доклад, в котором говорится о том, что

несмотря на утверждения разработчиков, значения S-блока шифра Кузнечик и хэш-функции Стрибог не являются (псевдо)случайными числами, а сгенерированы на основе скрытого алгоритма, который им удалось восстановить методами обратного проектирования [3, 4]. Riham AlTawy и Amr M. Youssef описали атаку «встречи посередине» на 5 раундов шифра Кузнечик, имеющую вычислительную сложность 2^{140} и требующую 2^{153}памяти и 2^{113} данных [5]. В работе [2] рассматриваются подходы к анализу алгоритма Кузнечик с использованием слайдовой атаки. Однако при этом показано, что рассматриваемая степень самоподобия никогда не может быть достигнута в силу используемой в шифре функции для генерации раундовых подключей. В настоящий момент нет публикаций с информацией о дифференциальных характеристиках алгоритма шифрования Кузнечик, а также описывающих возможность применения данного метода анализа к какому-либо сокращенному количеству раундов шифра Кузнечик.

Метод дифференциального криптоанализа был впервые предложен Эли Бихамом и Ади Шамиром, которые применили его к анализу алгоритма шифрования DES [6, 7]. В своей работе они показали, что алгоритм DES оказался довольно устойчивым к данному методу криптоанализа, и любое малейшее изменение структуры алгоритма делает его более уязвимым. При этом им все же удалось значительным образом понизить сложность анализа алгоритма DES, сократив ее с 2^{56} до 2^{36}. Однако требование такого же количества пар открытых и шифрованных текстов (2^{36}) оставило эту атаку теоретической для шифра DES. Несмотря на это, метод оказался очень удачным. Его применение возможно не только к симметричным блочным шифрам, но также и к поточным шифрам [8], а также к функциям хеширования [9].

В настоящей статье мы предлагаем метод построения трехраундового дифференциала для алгоритма шифрования Кузнечик. Разработанная схема анализа основана на использовании дифференциальных свойств преобразований S иLалгоритма Кузнечик и предназначена для того, чтобы можно было определить правильную пару текстов для дальнейшего анализа, целью которого является определение секретного ключа шифрования. Схема разработана таким образом, чтобы затрагивать активные нелинейные компонены (S-блоки) минимальное количество раз. В результате для предложенной схемы вероятность нахождения правильных пар текстов составляет 2^{-108} зашифрований. Также мы разработали алгоритм нахождения секретного ключа, сложность которого составляет $6*2^{-120}$зашифрований с использованием шифра Кузнечик.Таким образом, общая сложность анализа, включая поиск правильных пар текстов и поиск битов секретного ключа шифрования составляет $2^{-108} + 6*2^{-120}$ зашифрований.Сложность предложенной схемы является достаточно высокой в сравнении с возможностями современных вычислительных средств, но гораздо более низкой чем сложность компрометации ключа методом полного перебора.

Статья организована следующим образом. В первой части статьи содержится описание алгоритма шифрования Кузнечик. Во второй частиприводится исследование дифференциальных свойств трех основных преобразований шифра Кузнечик: операции сложения по модулю два, нелинейного S-блока замены, линейного преобразования L.В третьей части содержится описание процесса построения диференциальной характеристики для трех раундов шифра Кузнечик. Ключевым моментом данного раздела является не использование дифференциальных свойств рассмотренных преобразований по отдельности, а исследование работы данных преобразований в совокупности. Именно данное наблюдение позволяет нам построить трехраундовые характеристики таким

образом, чтобы в процесс было вовлечено минимально количество S-блоков замены, тем самым обеспечивая максимальное значение вероятности. В четвертой части содержится описание разработанного и реализованного алгоритма поиска правильных пар текстов для проведения анализа шифра. В пятой части содержится описание процесса компрометации ключа вследствие использования разработанного алгоритма анализа трех раундов шифра Кузнечик. В шестом разделе приведены теоретические расчеты о времени, требуемом для проведения атаки с использованием самых мощных суперкомпьютеров мира при использовании как простой реализации, так и реализации с использованием специальных таблиц предвычислений.

1. Описание алгоритма шифрования Кузнечик

Алгоритм шифрования Кузнечик представляет собой симметричный блочный шифр с длиной блока равной 128 бит и длиной ключа равной 256 бит. Шифр Кузнечик построен по принципу SP-сети, что позволяет выполнить преобразования над всем входным блоком целиком, а не только над его частью.

Процесс шифрования состоит из девяти раундов, каждый из которых включает в себя три преобразования: сложение по модулю 2 (xor) с раундовым ключом, замена с помощью блоков подстановок (S) и линейное преобразование (L).Один раунд шифрования показан на рисунке 1. Раундовые ключи, их десять, вырабатываются на основе 256-битного мастер-ключа. Первые два ключа получаются путем разбиения мастер-ключа пополам, а последующие восемь при помощи восьми раундов сети Фейстеля. В каждом раунде осуществляются: xor ключа с раундовой константой, преобразование с помощью блока подстановок, линейное преобразование. Раундовую константу получаем из применения к номеру раунда линейного преобразованияL. Подробное описание работы шифра, а также численные примеры, моделирующие процесс шифрования, можно найти по ссылке [1].

Функция расшифрования выполняется в обратном порядке, снизу вверх, с сипользованием раундовых преобразований, инверсных к тем, что использовались при зашифровании.

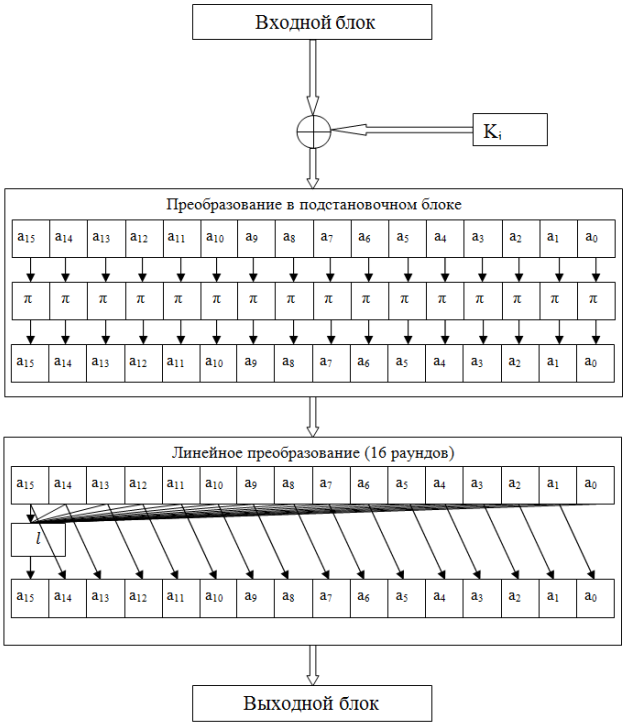

Рисунок 1. – Один раунд шифрования

2. Анализ дифференциальных свойств основных компонентов шифра Кузнечик

Прежде чем мы приступим к рассмотрению дифференциальных свойств основных компонентов шифра Кузнечик, введем несколько понятий и обозначений, которыми мы будем оперировать в дальнейшем. Метод дифференциального криптоанализа рассматривает тексты не по отдельности, а в парах. Необходимо проследить как меняется разность двух текстов при их прохождении через раунды шифрования. В качестве меры несходства рассматривается результат сложения по модулю два, который принято называть ***дифференциалом***. Так, пара текстов на входе в алгоритм шифрования X и X' называется ***входным дифференциалом*** и обозначается как $\Delta X=X \oplus X'$. Пара текстов на выходе из алгоритма шифрованияY и Y', называется ***выходным дифференциалом*** и соответственно обозначается как ΔY. Под ***дифференциальной характеристикой*** будем понимать наиболее вероятную пару значений входного и выходного дифференциалов. ***Правильная пара текстов*** – это такая пара (открытый, шифрованный текст): (X, Y)и (X’, Y’), для которой $\Delta X=X \oplus X'$ и $\Delta Y=Y \oplus Y'$.***Неактивный S-блок*** – это такой блок замены, через который проходит значение разности, равное нулю. ***Активный S-блок*** – это такой блок замены, через который проходит значение разности, неравное нулю. Задача дифференциального анализа по сути сводится к тому, чтобы подобрать такое сочетание входного и выходного дифференциалов, для которого общее количество активных S-блоков будет минимальным (но отличным от нуля) при прохождении через все раунды шифрования. Итак, рассмотрим вначале дифференциальные свойства каждого из преобразований по отдельности.

Операция сложения данных с секретным ключом по модулю два (xor). Еще Э. Бихамом и А. Шамиром [6, 7],было показано, что ключ не может повлиять на знаение дифференциала. Это связано с тем, что мы рассматриваем два параллельных процесса шифрования, где используется один и тот же секретный ключ. Это значит, что благодаря

свойствам операции сложения по модулю два,биты секретного ключа будут взаимно уничтожены – $X \oplus K_i \oplus X' \oplus K_i$. $= X \oplus X' = \Delta X$.

Линейное преобразование L. Преобразование L в данном алгоритме является линейным. Это значит, что оно преобразовывает значение разности однозначно, с вероятностью, равной 1. При этом аналогично преобразованию MixColumn в алгоритма шифрования AES, оно перемешивает все биты исходного состояния. Поэтому, если у вас на входе в преобразование L был всего один ненулевой байт, то на выходе из L преобразования скорее всего все байты будут ненулевые. Легко показать, что для преобразования L выполняется следующее свойство:

$$L(a) \oplus L(b) = L(a \oplus b)$$

Нелинейное преобразование S. Дифференциалы, поступающие на вход S-блока замены обозначим как ΔA, а дифференциалы, получаемые на выходе S-блока замены обозначим как ΔC. Для анализа алгоритма нам необходимо составить таблицу, которая будет отражать вероятности появления различных значений ΔC для заданного значения ΔA. Алгоритм построения подобных таблиц можно найти в работах Э. Бихама и А. Шамира [6, 7]. Итоговая таблица получается довольна большая, она содержит 256 строк и 256 столбцов. Часть полученной таблицы вероятностей показана в таблице 1.

Таблица 1 – Часть таблицы вероятностей, составленной для блока подстановок S в процессе дифференциального анализа

ΔC / ΔA	0	1	2	…	3e	3f	…	fe	ff
0	256/256	0/256	0/256	…	0/256	0/256	…	0/256	0/256
1	0/256	0/256	2/256	…	2/256	4/256	…	0/256	2/256
2	0/256	4/256	0/256	…	0/256	0/256	…	4/256	2/256
3	0/256	2/256	0/256	…	6/256	2/256	…	0/256	0/256
…	…	…	…	…	…	…	…	…	…
a5	0/256	0/256	0/256	…	0/256	2/256	…	2/256	0/256
a6	0/256	4/256	2/256	…	0/256	0/256	…	0/256	2/256
…	…	…	…	…	…	…	…	…	…
fe	0/256	0/256	2/256	…	2/256	0/256	…	2/256	0/256
ff	0/256	2/256	2/256	…	0/256	4/256	…	0/256	0/256

Строки табл. 1 обозначают входные значения разности ΔA в S-блок, а столбцы – соответствующие выходные значения разности ΔC, получаемые на выходе S-блока. На пересечении строк и столбцов отображается количество пар дифференциалов ΔA/ΔC имеющих данные входную и выходную разности. В результате проведенного анализа было установлено, что вероятность соответствия для значений ΔA/ΔC может быть равна 2/256, 4/256, 6/256 и 8/256, а также может быть равной нулю. При этом ненулевое значение разности на входе S-блока никогда не будет преобразовано к нулевой разности на выходе.

Если в результате проведения анализа удается найти правильную пару текстов, то это дает нам возможность предположить, что разности проходили через раунды шифрования именно так, как мы рассматривали их при построении раундовых дифференциалов. Комбинация дифференциалов ΔA и ΔC позволяет предположить значения $A \oplus K_i$ и $A' \oplus K_i$. При известных A и A' это позволяет определить K_i.

3. Построение дифференциальной характеристики для трех раундов шифра Кузнечик

Итак, в предыдущем разделе было отмечено, что в силу перемешивания байтов в результате применения преобразования L, если на входе в преобразование L был всего один ненулевой байт, то на выходе из L преобразования скорее всего все байты будут ненулевые. Это означает, что если в первом раунде шифрования будет использован один активный S-блок, то во втором раунде шифрования после L-преобразования первого раунда, уже все 16 блоков будут активными. А это означает резкое падение значения вероятности при прохождении значений разности через S-блок. Даже, если в соответствии с табл. 1 рассматривать наибольшее значение вероятности, равное 8/256 = $\frac{1}{2^5}$, при 16 активных S-блоках значение вероятности изменится в $\left(\frac{1}{2^5}\right)^{16} = \frac{1}{2^{80}}$ раз. А это означает, что в последующем раунде скорее всего также будут задействованы все 16 блоков замены. Даже при самом благоприятном значении вероятности каждого из рассматриваемых блоков, каждый раунд будет сокращать значение вероятности в $\frac{1}{2^{80}}$ раз. Таким образом, уже через три раунда вероятность дифференциала будет равна (при самых благоприятных вероятностях, что еще также требует проверки) $\frac{1}{2^5} * \frac{1}{2^{80}} * \frac{1}{2^{80}} = \frac{1}{2^{165}}$. После четвертого раунда вероятность приблизится к вероятности полного перебора и станет равна $\frac{1}{2^{245}}$, что сделает дальнейший анализ бесполезным. Отметим, что данные значения вероятностей посчитаны с учетом самых благоприятных значений вероятностей. Хотя на практике еще требуется проверить, могут ли они быть достигнуты одновременно. Тем не менее такой подход в расчете вероятностей явно дает понять бессмысленность данных расчетов.

Наша идея заключалась в том, чтобы использовать свойства S и L преобразований вместе. Изначально появилась гипотеза, а что если один активный S-блок в первом раунде шифрования приведет к 16 активным S-блокам во втором раунде шифрования, а в третьем раунде шифрования мы сможем опять получить один активный S-блок? Схематично это можно представить так, как сделано на рис. 2.

Выше было показано, что сложение данных с раундовым подключом не изменяет значение дифференциала. В таком случае наша идея построения трехраундового дифференциала сводилась к следующему.

Первый раунд: На вход алгоритма подается входной дифференциал ΔX, у которого все байты за исключением одного имеют нулевые значения. Это же значение дифференциала поступает на вход нелинейного биективного преобразования S, где изменяется на другой байт в соответствии с таблицей вероятностей, построенной для данного нелинейного преобразования. Таким образом мы получаем один активный S-блок. Разность на выходе преобразования S по-прежнему содержит один ненулевой байт. После этого значение разности подается на вход линейного преобразования L, в

результате которого на выходе получается значение разности, в которой все 16 байтов имеют ненулевые значения.

Второй раунд: Дифференциал, полученный в результате преобразования Lпервого раунда остается неизменным после сложения со вторым раундовым подключом. Таким образом, для нелинейного преобразования Sвторого раунда все 16 байтов окажутся активными. В результате работы S-блока каждый байт разности может быть преобразован к другому значению в соответствии с таблицей вероятностей. На вход линейного преобразования L второго раунда поступит значение разности, у которого все 16 байтов будут иметь ненулевое значение. После L-преобразования эти 16 ненулевых байтов будут преобразованы к значению разности, которое содержит всего один ненулевой байт.

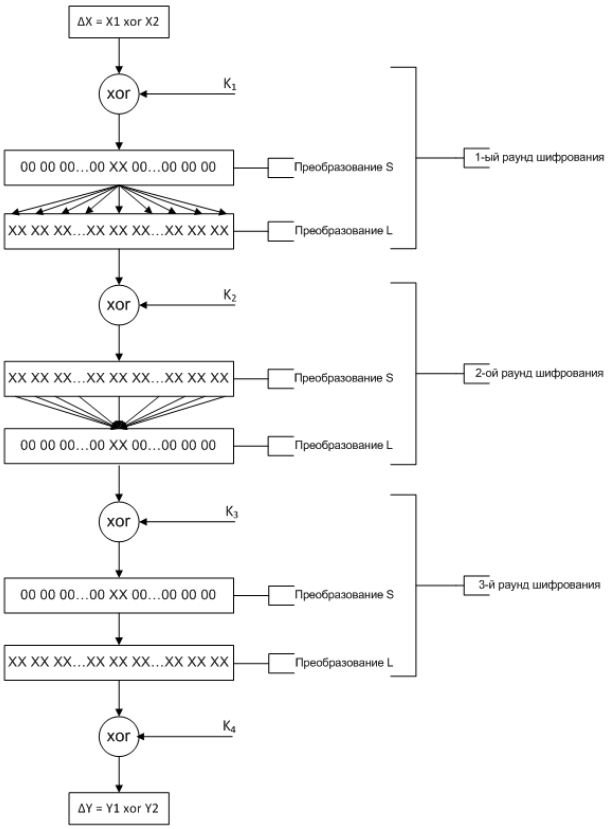

Рисунок 2 – Схема дифференциального анализа трех раундов шифрования

Третий раунд: Полученное во втором раунде значение дифференциала складывается с третьим раундовым ключом, который по аналогии с первым и вторым раундом не влияет на значение дифференциала. Таким образом в S-преобразовании мы получаем всего один активный S-блок. Ненулевой байт разности в соответствии с таблицей вероятностей меняется на другое вероятное значение. Затем в результате преобразования Lразность с одним ненулевым байтом преобразуется к значению разности, которое опять содержит 16 ненулевых байтов.

Таким образом, для рассматриваемой схемы анализа мы получаем всего 18 активных S-блоков вместо 33, как было рассмотрено ранее. Даже если предположить, что мы сможем подобрать значения дифференциалов, которые будут соответствовать рис. 2, с минимальными значениями вероятностей для каждого из активных S-блоков 2/256 = $\frac{1}{2^7}$, мы получим трехраундовый дифференциал с вероятностью в $\left(\frac{1}{2^7}\right)^{18} = \frac{1}{2^{126}}$, что меньше, чем

ранее предположенное значение $\frac{1}{2^{165}}$. Забегая вперед, хотим сказать, что нам удалось получить трехраундовую характеристику, вероятность появления которой равна $\frac{1}{2^{108}}$, что означает, что при использовании активных S-блоков были задействованы не самые минимальные вероятности.

Итак, для того, чтобы построить дифференциал, схема которого представлена на рис. 2, нам необходимо было рассмотреть работу S и L преобразований в совокупности. В общем виде схематично это можно представить так, как сделано на рис. 3. Необходимо чтобы в первом раунде один ненулевой байт дифференциала после L преобразовывался в 16 ненулевых байт (левая часть рисунка). Во втором раунде нам необходимо, чтобы 16 ненулевых байтов дифференциала после L преобразовывались в 1 ненулевой байт. Или, если пойти от обратного, то нас интересуют такие случаи, когда один ненулевой байт дифференциала после L_{inv} преобразовывался в 16 ненулевых байт (правая часть рисунка). При этом дифференциалы левой и правой части должны быть связаны преобразованием S, то есть в таблице вероятностей должна иметься хотя бы минимальная вероятность 2/256 преобразования значения разности из левой части рис. 3 к правой части.

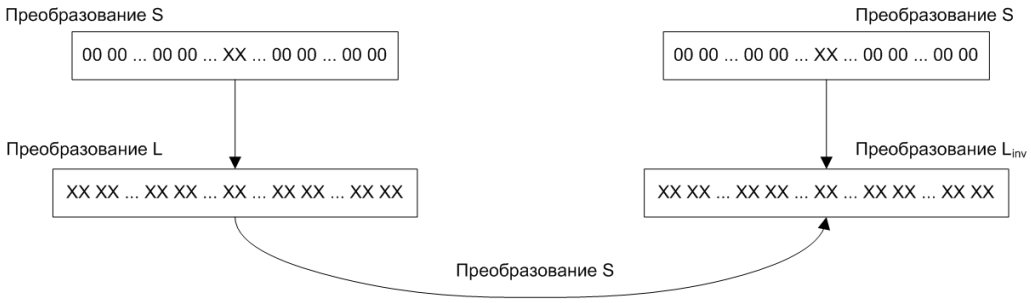

Рисунок 3 – Зависимость преобразований Si L

Таким образом, предложенная нами схема взаимосвязей S и L преобразований подразумевает, что через преобразования L и L_{inv} нужно пропустить шестнадцать различных вариантов заполнения одного байта от 00 до FF, который в результате данного преобразования разложится на 16 байт. Сформулируем предложенный нами алгоритм действий по шагам:

Первый шаг: на вход преобразования L подается блок, в котором всего один байт имеет ненулевое значение. Через преобразование L пройдет 16 различных вариантов заполнения каждого возможного байта от 00 до FF. Каждое однобайтовое значение раскладывается с помощью преобразования L на значение, состоящее из 16-ти байтов. В итоге мы получаем возможные значения разностей на входе преобразования S второго раунда. Всего таких значений будет 255*16=4080. Это возможные значения на выходе L преобразования первого раунда (левая часть рис. 3). Эти же значения будут поступать на вход S-блока второго раунда шифрования. Обозначим их как ΔA.

Шаг второй: Аналогичен первому шагу за тем исключением, что вместо L преобразования используется преобразование L_{inv}. Таким образом, мы получаем еще 4080 значений. Это возможные значения на входе L преобразования второго раунда (правая часть рис. 3). Эти же значения будут появляться на выходе S-блока второго раунда шифрования. Обозначим их как ΔC.

Шаг третий: В результате выполнения первого и второго шагов мы имеем дифференциалы ΔA и ΔC на входе и выходе преобразования S второго раунда соответственно. Так как для каждого значения ΔA не все значения ΔC равновероятны, нам нужно определить, какие именно поступающие на вход блока подстановки S дифференциалы ΔA из 4080 значений будут иметь ненулевую вероятность быть полученными в виде значений ΔC в результате преобразования. Для этого необходимо каждое из 4080 значений ΔA разбить на 16 байт и подать на вход таблицы вероятностей. Аналогично разбить на байты каждое значение ΔC. Для некоторых 16-ти байтных дифференциалов ΔA из 4080 значений по таблице найдутся такие дифференциалы ΔC, которые будут иметь ненулевую вероятность. Всего, в результате рассмотрения всех возможных комбинаций, нами было получено 13 пар значений ΔA/ΔC, которые показаны в табл.2.

Я нашла еще вероятности для каждого байта составляющего ΔA/ΔC - они нужны?

Таблица 2 – Найденные пары значений ΔA / ΔC

ΔA	ΔC
f3ab8c55c199996fc5a4f2381976846	51ac91f0df2470190ad86a256131163
1a76bc71665284b01a3e595982599369	ba5a9d5e6d2b6431ac6b9cb72dc5a7a1
5cfbaa318fd91c774940bef22a5f86	1f8355405427f8e7d8c71cc07f2288c6
ac8ea817121caa3445efd4b9c43e875f	c6e355f95177d1bd58f9a4145283a143
522cbe6cbcf88eaf4963f28f8f29e62	353cceb5eab273db5790cc909cfa9
2bc22a75e57cbd804b35bf31ee5167	54be1b26f4dbe5b1f6a2a66a61e384d
5f28ebaf31b588b3f8f23923e399f0ef	b4eba9c9151b1fbd907f4f4d419fcdaa
7dbda6246e653ba46ec427fafa9a462f	5d87c43030f77ec08a9f25c0b8413318
efd9d1a17499185749edf1d1d6ee4c	bbd9f4a6c11bf5e154a2c52495e1ddd9
8fb8e26a91ccf9b72d6d5dce4f9ad4a9	b86cc61926ba16e11d19dce66e78450
6a7c998e18379bf720d423721d3c7e63	c6fc783ae4466b402df56e30dff1f8d4
337ddbfefc424a38d45ae559c2d2cd36	fd58a4739c68ed296855b6723e9fb3
5c9ce97665a2afd9172a54a881949c	3a3d1b8c2658ea7f8a958b2f866ecb5b

Из найденных значений ΔA/ΔC легко можно определить значения дифференциалов на входе и выходе 3-раундового алгоритма Кузнечик ΔX/ΔY, а также вероятность их появления. Для первой пары значений ΔA/ΔC с наибольшей вероятностью равной $\frac{6}{256}$ на вход алгоритма подано значение ΔX = 00000000002000000000000000000000, а на выходе получены три равновероятных значения ΔY_1 = e07d0b92e148a1f4774154a6b06e99f0, ΔY_2 = 0d38eb47ba753eb564333cbda803ac2c, ΔY_3 = acb714a9647d96d76b3a89ab83f1bd71. Однобайтовые значения, которые развернулись на данные 16-байтовые значения были получены с максимальной вероятностью равной $\frac{6}{256}$.

4. Алгоритм нахождения правильных пар текстов для трехраундовой характеристики

В общем случае, если вам известно значение дифференциала и его вероятность, то в алгоритме поиска правильных пар текстов нет ничего сложного. В общем случае алгоритм поиска правильных пар текстов сводится к следующему:

1. Сгенерировать случайным образом текст X.
2. Вычислить текст X': $X' = X \oplus \Delta X$
3. Зашифровать X, получить шифр Y
4. Зашифровать X', получить шифр Y'
5. Если $\Delta Y = Y \oplus Y'$, то (X,Y) и (X', Y') – правильная пара текстов, иначе вернуться к шагу 1.

Сложность анализа будет определяться вероятностью нахождения таких дифференциалов. Мы сделали теоретические расчеты, какое количество времени уйдет в среднем на поиск одной правильной пары текстов при использовании самых мощных суперкомпьютеров мира. Данные расчеты будут приведены в разделе 6.

Как бы там ни было, для нас произвести такие вычислительные расчеты не представляется возможным. Однако, чтобы подтвердить работоспособность предложенной схемы дифференциалов для трех раундов шифрования мы решили пойти «от обратного». То есть, зная, как разности должны преобразовываться при прохождении через раунды шифрования не найти перебором, а подобрать тексты и секретные ключи, которые будут соответствовать полученной нами раундовой характеристики. Сделано это было с одной целью – показать, что такие значения текстов существуют, а значит предложенная нами схема – работоспособна.

Как видно из табл. 2, всего было найдено 13 пар значений $\Delta A/\Delta C$. Из этих значений можно вычислить, какие значения дифференциалов были поданы на вход алгоритма шифрования, а так же какие будут получены на выходе. Для этого нужно провести ряд обратных преобразований.

Первый шаг: поиск открытых текстов. ΔA подается на вход преобразования L_{inv}, обратного к преобразованию L первого раунда, в результате $L_{inv}(\Delta A)$ получаем выход преобразования S первого раунда. Затем, предыдущий результат подается на вход преобразования S_{inv} первого раунда, обратного к преобразованию S, в результате $S_{inv}(L_{inv}(\Delta A))$ получим дифференциал, подаваемый на вход алгоритма.

Второй шаг: поиск текстов по итогам трех раундов. ΔC подается на вход преобразования L второго раунда, получаем $L(\Delta C)$. Затем полученный результат подается на вход блока замены третьего раунда, где в результате перестановки S по построенной табл. 1, содержащей значения вероятностей, мы получаем $S(L(\Delta C)$. В конечном итоге, последний найденный результат поступает на вход преобразования L, где претерпевает изменения в соответствии с алгоритмом. Таким образом мы получаем требуемый выход, содержащий искомый дифференциал ΔY.

Если найти байты, составляющие дифференциалы ΔA / ΔC, собрать из них составляющие их тексты и пропустить через трехраундовые преобразования, то при определенных значения секретного ключа, по разработанному и вышеописанному

способу, правильная пара текстов образует нужные значения дифференциала. Для доказательства работоспособности предложенного метода были получены исходные значения, часть которых изображена в таблице 3 и итоговые значения, часть которых изображена в таблице 4. Где A и A’ – тексты, составляющие найденный дифференциал ΔA, а C и C’ – тексты, составляющие дифференциал ΔC. Если принять мастер-ключ как K= 8899aabbccddeeff0011223344556677fedcba98765432100123456789abcdef, то раундовые ключи K_1, K_2, K_3 и K_4 равны следующим значениям:

– K_1=8899aabbccddeeff0011223344556677;
– K_2=fedcba98765432100123456789abcdef;
– K_3=db31485315694343228d6aef8cc78c44;
– K_4=3d4553d8e9cfec6815ebadc40a9ffd04.

Таблица 3 – Исходные тексты

$\mathbf{K_1 \oplus (S_{inv}(L_{inv}(A \oplus K_2)))}$	$\mathbf{K_1(S_{inv}(L_{inv}(A' \oplus K_2)))}$
8998079702eaede8422655046852245d	8998079702fbede8422655046852245d
8998079702eaede8422655046852cf5d	8998079702fbede8422655046852cf5d
8998079702eaede8422655046852825d	8998079702fbede8422655046852825d

Таблица 4 –Значения по итогам трех раундов

$\mathbf{K_4 \oplus (L(S(C)))}$	$\mathbf{K_4 \oplus (L(S(C')))}$
8ccf9f63b68216e7e87fb65cfce4364f	27ac6bf0c98ce6b78ed52e43f34d17cb
7152f68de9e8354c0db9cb0594ee651e	da31021e96e6c51c6b13531a9b47449a
8ef7bcaec365c4890540ec622239d595	2594483dbc6b34d963ea747d2d90f411

5. Алгоритм поиска секретного ключа на основе анализа правильных пар текстов

Метод дифференциального криптоанализа, как правило, сводится к компрометации раундовых ключей с целью дешифрования информации. Мало найти правильные пары текстов, нужно еще уметь использовать их для того, чтобы восстаносить информацию о битах секретного ключа. Метод криптоанализа на основе ранее найденных правильных пар текстов заключается в следующем.

При рассмотрении трех раундов шифрования, мы используем 4 раундовых ключа (рис. 2). Особенностью алгоритма Кузнечик является то, что первыми двумя раундовыми ключами являются половинки 256-битного мастер-ключа, на основе которого вырабатываются остальные раундовые подключи. Учитывая этот факт, предложенный нами алгоритм предусматривает нахождение ключа K_1, затем K_2, из которых вырабатываются остальные ключи K_3 и K_4. Схема предложенного алгоритма приведена на рис. 4.

Как было сказано в части 4 данной статьи, при проведении обратных операций над найденным ΔA, а именно $S_{inv}(L_{inv}(\Delta A))$, мы получим дифференциал, поступающий на вход преобразования S первого раунда, а при $L_{inv}(\Delta A)$ мы получим выход того же преобразования. Таким образом, если подать на вход алгоритма, изображенного на рис. 2, ранее подобранную пару текстов X и X’, то эти тексты при сложении с числом,

получившимся в итоге при $S_{inv}(L_{inv}(\Delta A))$,дадут значение ключа K_1.Так как значение $S_{inv}(L_{inv}(\Delta A))$ состоит из одного байта (8 бит) и имеет вероятность по таблице вероятностей равное 6, то это даст нам 6 возможных значений одного байта ключа K_1. Таким образом, количество всех возможных вариантов 128-битного ключа K_1будет равно $2^{120}\times 6$.

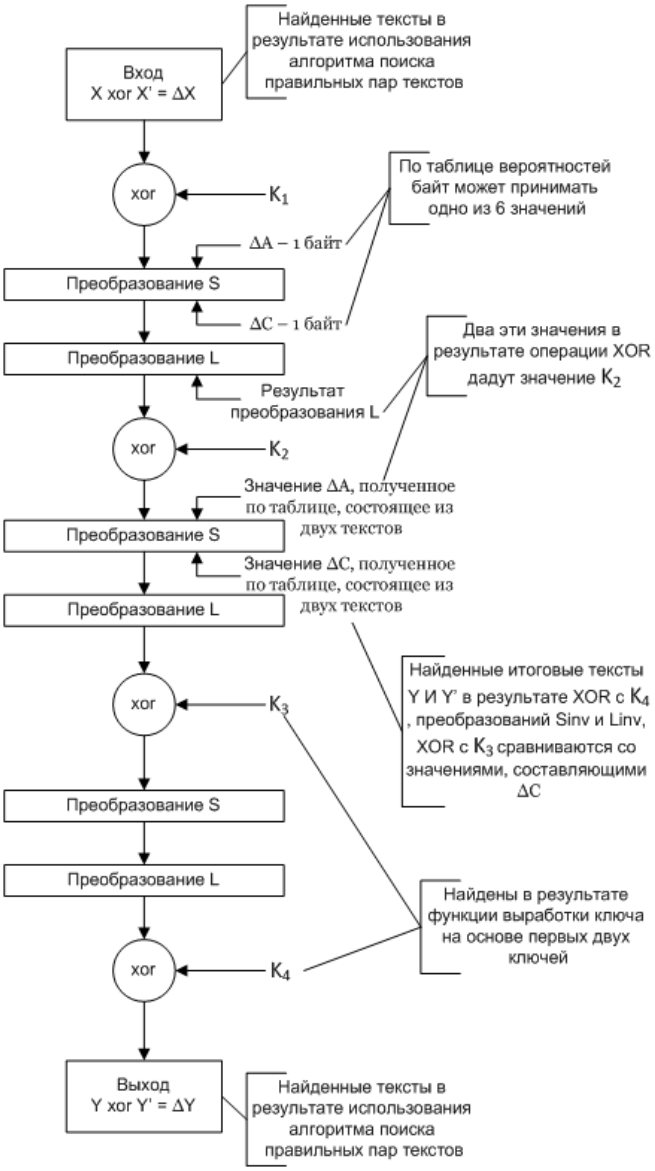

Рисунок 4 – Алгоритм поиска битов ключа

Для проверки одного возможного варианта ключа K_1 нужно произвести несколько операций. Исходя из рис.4, сначала подаем на вход два подобранных текста X и X’, затем складываем их с одним возможным ключом K_1 из общего количества возможных ключей K_1. Затем полученные значения проходят преобразование S первого раунда, затем преобразование L. Результаты преобразования складываются со значениями, составляющими дифференциал ΔA, поступающий на вход преобразования S второго раунда. На данном этапе получаем одно из возможных значений ключа K_2. Объединяем вместе ключи K_1 и K_2, получаем 256-битный мастер-ключ и из него вырабатываем ключи K_3 и K_4.

Следующим шагом берем два возможных текста Y и Y’, полученных в результате применения алгоритма нахождения правильных пар текстов для дифференциального криптоанализа, и складываем их с ключом K_4. Теперь, полученные значения претерпевают преобразования L_{inv} и S_{inv} третьего раунда и складываются с ключом K_3. Затем эти же значения претерпевают преобразование L_{inv} второго раунда и сравниваются со значениями, составляющими дифференциал ΔC. Если значения равны, то ключи сохраняются как вероятно правильные. Если значения не равны, то алгоритм действий повторяется заново: два текста X и X’ складываются со следующим ключом K_1, из всех возможных значений K_1. И все последующие действия выполняются по точно такой же схеме. Фактически опробование ключей сводится к выполнению всех операций трехраундового зашифрования. Поэтому в общей сложности алгоритм будет задействовать максимум $2^{120}\times6$ зашифрований. При этом будут найдены все 4 ключа, что представляет собой гораздо меньшее число, чем поиск ключа полным перебором, где общее возможное количество ключей представляет число, равное 2^{128} только лишь для первого ключа K_1, количество значений для поиска второго ключа K_2 полным перебором также равно 2^{128}, что в общем случае позволяет оценить сложность поиска ключа полным перебором как 2^{256}.

Таким образом, сложность нахождения ключей с помощью разработанного метода криптоанализа составляет $2^{120}\times6$ 3-раундовых зашифрований. Общую сложность проведения дифференциального криптоанализа трех раундов шифрования можно оценить как $2^{120}\times6+2^{108}$ зашифрований.

Куда этот раздел - в заключение (как написано в аннотации) или в 6-ом пункте оставить (как написано во введении)??

6. Примерная оценка временных затрат, требуемых для анализа алгоритма Кузнечик на современных суперкомпьютерах из ТОП 500.

Выполнение операций по разработанному нами алгоритму дифференциального криптоанализа трех раундов алгоритма Кузнечик можно оценить примерные временные затраты, которые потребуются при проведении анализа, например, на современных суперкомпьютерах из ТОП 500. Для оценки были выбраны первые 10 компьютеров из общего рейтинга [10].

Примерные временные затраты можно высчитать с помощью пропорции. Если на используемом нами компьютере частота 1,8 ГГц и при этом один блок данных считается за 0,0022 сек, то с частотой одного ядра суперкомпьютера равной, например, 1.45 ГГц один блок данных будет посчитан примерно за 0,0027 сек. Для расчета сложности анализа нужно разделить значение общей сложности на количество ядер компьютера, а затем, для подсчета времени анализа, умножить полученное значение на количество секунд обсчета одного блока. Результаты расчетов приведены в таблице 5.

Таблица 5 – Расчет временных затрат

№ п/п	Суперкомпьютер и его характеристики	Примерные временные затраты
1	Sunway TaihuLight – 10,649,600 cores, 1.45GHz	1 блок – 0,0027 сек Сложность анализа – $2^{85}+6\times2^{92}$ Время анализа – $(2^{85}+6\times2^{92})\times0,0027$
2	Tianhe-2 (MilkyWay-2) – 3,120,000 cores,	1 блок – 0.0018 сек

	2.2 GHz	Сложность анализа – $2^{87}+6\times2^{99}$ Время анализа – $(2^{87}+6\times2^{99})\times0{,}0018$
3	Piz Daint – 361,760 cores, 2.6 GHz	1 блок – 0.0015 сек Сложность анализа – $2^{90}+6\times2^{102}$ Время анализа – $(2^{90}+6\times2^{102})\times0{,}0018$
4	Titan – 560,640 cores, 2.2 GHz	1 блок – 0.0018 сек Сложность анализа – $2^{89}+6\times2^{101}$ Время анализа – $(2^{89}+6\times2^{101})\times0{,}0018$
5	Sequoia – 1,572,864 cores, 1.60 GHz	1 блок – 0.0025 сек Сложность анализа – $2^{88}+6\times2^{100}$ Время анализа – $(2^{88}+6\times2^{100})\times 0.0025$
6	Cori – 622,336 cores, 1.4 GHz	1 блок – 0.0028 сек Сложность анализа – $2^{89}+6\times2^{101}$ Время анализа – $(2^{89}+6\times2^{101})\times0{,}0028$
7	Oakforest – 556,104 cores, 1.4 GHz	1 блок – 0.0028 сек Сложность анализа – $2^{89}+6\times2^{101}$ Время анализа – $(2^{89}+6\times2^{101})\times0{,}0028$
8	K computer – 705,024 cores, 2.0 GHz	1 блок – 0.0019 сек Сложность анализа – $2^{89}+6\times2^{101}$ Время анализа – $(2^{89}+6\times2^{101})\times0{,}0019$
9	Mira – 786,432 cores, 1.60 GHz	1 блок – 0.0025 сек Сложность анализа – $2^{89}+6\times2^{101}$ Время анализа – $(2^{89}+6\times2^{101})\times0{,}0025$
10	Trinity – 301,056 cores, 2.3 GHz	1 блок – 0.0017 сек Сложность анализа – $2^{90}+6\times2^{102}$ Время анализа – $(2^{90}+6\times2^{102})\times0{,}0017$

Где найти Ромину публикацию в англоязычном варинте, чтобы вставить ее в список источников?

Также возможно оценить временные затраты при использовании высокоскоростной реализации алгоритма шифрования Кузнечик с использованием таблиц предвычислений. Частота машины, на которой реализовывались высокоскоростные вычисления 2.67GHz, а скорость обработки информации 120 МБ/сек. Следовательно 1 блок обрабатывается примерно 0,000001 секунд. Результаты временных рассчетов показаны в таблице 6.

Таблица 6 – Расчет временных затрат с использованием высокоскоростной реализации алгоритма Кузнечик

№ п/п	Суперкомпьютер и его характеристики	Примерные временные затраты
1	Sunway TaihuLight – 10,649,600 cores, 1.45GHz	1 блок – 0.0000018 сек Сложность анализа – $2^{85}+6\times2^{92}$ Время анализа – $(2^{85}+6\times2^{92})\times 0.0000018$
2	Tianhe-2 (MilkyWay-2) – 3,120,000 cores, 2.2 GHz	1 блок – 0.0000012 сек Сложность анализа – $2^{87}+6\times2^{99}$ Время анализа – $(2^{87}+6\times2^{99})\times 0.0000012$
3	Piz Daint – 361,760 cores, 2.6 GHz	1 блок – 0,000001 сек Сложность анализа – $2^{90}+6\times2^{102}$ Время анализа – $(2^{90}+6\times2^{102})\times 0{,}000001$
4	Titan – 560,640 cores, 2.2 GHz	1 блок – 0.0000012 сек Сложность анализа – $2^{89}+6\times2^{101}$

		Время анализа – $(2^{89}+6\times2^{101})\times 0.0000012$
5	Sequoia – 1,572,864 cores, 1.60 GHz	1 блок – 0.0000017 сек Сложность анализа – $2^{88}+6\times2^{100}$ Время анализа – $(2^{88}+6\times2^{100})\times 0.0000017$
6	Cori – 622,336 cores, 1.4 GHz	1 блок – 0.0000018 сек Сложность анализа – $2^{89}+6\times2^{101}$ Время анализа – $(2^{89}+6\times2^{101})\times 0.0000018$
7	Oakforest – 556,104 cores, 1.4 GHz	1 блок – 0.0000018 сек Сложность анализа – $2^{89}+6\times2^{101}$ Время анализа – $(2^{89}+6\times2^{101})\times 0.0000018$
8	K computer – 705,024 cores, 2.0 GHz	1 блок – 0.0000013 сек Сложность анализа – $2^{89}+6\times2^{101}$ Время анализа – $(2^{89}+6\times2^{101})\times 0.0000013$
9	Mira – 786,432 cores, 1.60 GHz	1 блок – 0.0000017 сек Сложность анализа – $2^{89}+6\times2^{101}$ Время анализа – $(2^{89}+6\times2^{101})\times 0.0000017$
10	Trinity – 301,056 cores, 2.3 GHz	1 блок – 0.0000012 сек Сложность анализа – $2^{90}+6\times2^{102}$ Время анализа – $(2^{90}+6\times2^{102})\times 0.0000012$

Таким образом, из полученных результатов оценки временных характеристик, при использовании алгоритма с высокоскоростной реализацией при анализе с помощью суперкомпьютеров из ТОП 500 скорость анализа существенно повышается.

Заключение

В результате работы над данным проектом нами впервые были исследованы и получены дифференциальные свойства алгоритма шифрования Кузнечик. На основе проведенных нами исследований была выявлена связь между преобразованиями S и L, которая позволила разработать алгоритм дифференциального криптоанализа трех раундов шифра Кузнечик, ранее никем не предложенный в открытых литературных источниках.

Для предложенной схемы дифференциального криптоанализа нами был разработан алгоритм нахождения правильных пар текстов для анализа шифра. На основе трехраундового дифференциала мы разработали алгоритм нахождения секретного ключа с гораздо меньшей сложностью, чем сложность при поиске ключа полным перебором. Разработанные алгоритмы позволяют оценить общую сложность проведения анализа, которая составляет $2^{-108} + 6*2^{-120}$.

Результаты работы включают в себя: разработанный, реализованный и протестированный алгоритм шифрования Кузнечик, разработанный алгоритм дифференциального анализа трех раундов алгоритма Кузнечик, а также разработанную для данного алгоритма схему нахождения правильных пар текстов для дифференциального анализа трех раундов шифрования. Программа реализована на языке программирования C++ в среде разработки Microsoft Visual Studio C++. Все промежуточные и конечные результаты исследований подробно описаны, показаны на примерах, структурированы в хронологической последовательности и наглядно отображены в виде иллюстраций, таблиц и схем в тексте данной статьи. Проведены теоретические расчеты о времени, требуемом для проведения атаки с использованием самых мощных суперкомпьютеров мира при использовании как простой реализации, так и реализации с использованием специальных таблиц предвычислений

Результаты работы использованы при выполнении исследовательских работ по гранту РФФИ №17-07-00654-а «Разработка и исследование последовательных и параллельных алгоритмов анализа.

Библиографический список

1 Криптографическая защита информации Блочные шифры ГОСТ Р 34.12–2015 [Электронный ресурс].– URL: http://tc26.ru/en/standard/gost/GOST_R_34_12_2015_ENG.pdf

2 E.A. Ishchukova, L.K. Babenko, M.V. Anikeev Fast Implementation and Cryptanalysis of GOST R 34.12-2015 Block Ciphers // 9th International Conference on Security of Information and Networks, SIN 2016, Newark, Nj, 20-22 July 2016. – P. 104 – 111.

3 AlexBiryukov, LéoPerrin, AlekseiUdovenko. Reverse-Engineering the S-Box of Streebog, Kuznyechik and STRIBOBr1 (Full Version) (2016). https://eprint.iacr.org/2016/071.pdf

4 Alex Biryukov, Léo Perrin, Aleksei Udovenko. Reverse Engineering the S-Box of Streebog, Kuznechik and Stribob (2015).
http://crypto.2015.rump.cr.yp.to/1ea2c6c01144e0e7f6b14b324c5e4562.pdf

5- Riham AlTawy and Amr M. Youssef. A Meet in the Middle Attack on Reduced Round Kuznyechik (17 апреля 2015) https://eprint.iacr.org/2015/096.pdf

6 Biham, E., and Shamir, A., "Differential cryptanalysis of DES-like cryptosystems", Journal of Cryptology, Vol. 4, No. 1, pp. 3-72 (full issue), 1991.
http://www.cs.bilkent.edu.tr/~selcuk/teaching/cs519/Biham-DC.pdf

7 Biham, E., and Shamir, A., "Differential cryptanalysis of the full 16-round DES", Advances in cryptology, proceedings of CRYPTO'92, pp. 487-496, 1992.

8 Eli Biham, Orr Dunkelman, Differential Cryptanalysis in Stream Ciphers // Cryptology ePrint Archive, Report 2007/218, 2007. http://eprint.iacr.org/

9 EliBiham, AdiShamir Differential Cryptanalysis of Hash Functions // Differential Cryptanalysis of The Data Encryption Standard, Springer, 1993, ISBN 978-1-4613-9314-6. – P. 133 – 148

10 Статистика по суперкомпьютерам – ТОП 500[Электронный ресурс].– URL: https://www.top500.org/lists/2017/06/

DIFFERENCIAL ANALYSIS OF 3 ROUNDS OF CIPHER KUZNYECHIK

Annotation

In January 2016, a new standard of block-by-block encryption entered into force in the Russian Federation GOST R 34.12-2015. It includes two algorithms of encryption. The first cipher was previously known under the name GOST28147-89 (or simply GOST). The second algorithm was called Kuznyechik. Kuznyechik is a new symmetric algorithm of encryption, created on the principle of SP-network. Up to now there are no publications about the differential properties of the algorithm «Kuznyechik». We are the first to examine the properties of main operations and suggest a method of differential analysis of three rounds of encryption cipher «Kuznyechik». We examined the differential properties of the non-linear function S and the linear function L and found out that it's possible a situation when, 1 non-zero byte difference, being a result of the function L, is expanded into 16 non-zero bytes, then it passes through the replacement block S, and then collapses again into 1 non-zero byte. The developed scheme allows to affect the active S-block a minimum number of times. As a result, for the suggested scheme the possibility of finding the correct pairs of texts is equal to 2^{-108} encryptions. We also developed the algorithm of finding a secret key, the complexity of which is equal to $6*2^{-120}$ encryptions. In this way, the total complexity of the analysis, including searching for the correct pairs of texts and bits of the secret encryption key is equal to $2^{-108} + 6*2^{-120}$ encryptions.

As a result of the researches was developed and tested the encryption algorithm Kuznyechik, was developed a method of differential analysis of three rounds of algorithm Kuznyechik, and for this algorithm was created a scheme of finding the correct pairs of texts for the differential analysis of three round of encryption. The program is realized in the programming language C++ in the development framework Microsoft Visual Studio C++. The results of the researches are described in detail, are shown in examples, are structured in chronological order and are visually displayed in the form of illustrations, tables and diagrams in the text of this article. The article concludes with theoretical calculations of the time required to conduct an attack using the most powerful supercomputers in the world, using both simple implementation and implementation with the use of special precomputation tables.

Introduction

The encryption algorithm Kuznyechik was chosen as the standard GOST R 34.12-2015 and officially came into force on January 1, 2016, that's why today the research of its reliability is a *relevant objective*. The description of the encryption algorithm Kuznyechik, as a part of the new admitted encryption standard, is contained in document of the technical standardization committee TC 26 «Cryptographic protection of information» in GOST R 34.12–2015 [1]. There are several articles, dedicated to various ways of realization of the algorithm Kuznyechik, including using the special precomputation tables. [2]

Up to date you can already find several studies dedicated to the analysis of the reliability of the cipher Kuznyechik. At the international conference «CRYPTO 2015» Alex Biryukov, Léo Perrin and Aleksei Udovenko presented a report, in which was said that, despite the developer's statements, the values of S-block of cipher Kuznyechik and hash function Stribog aren't (pseudo) random numbers, but they are generated on the basis of hidden algorithm, which was restored by them using the methods of reverse engineering [3, 4]. Riham AlTawy and Amr M. Youssef described the meet-in the middle attack on five rounds of cipher Kuznyechik, that has a

computational complexity of 2^{140}, and requires 2^{153} of memory and 2^{113} of data [5]. In the work [2] are discussed the approaches to the analysis of the algorithm Kuznyechik using a slide-attack. But it was also shown that the considered degree of self-similarity can never be achieved because of the function used in cipher for the generation of round sub keys. Nowadays there are no publications with information about the differential characteristics of encryption algorithm Kuznyechik, and also the publications that describe the possibility of using of this analysis method with minimized number of rounds of cipher Kuznyechik.

The method of differential cryptanalysis for the first time was proposed by Eli Biham and Adi Shamir, who applied it to the analysis of the encryption algorithm DES [6, 7]. In their studies, they showed that the algorithm DES is quite robust to this method of cryptanalysis, and any smallest change of the algorithm's structure makes it more vulnerable. However, they still managed to reduce significantly the complexity of analysis of the algorithm DES, reducing it from 2^{56} to 2^{36}. But requirement of the same number of the open and encrypted pairs of texts (2^{36}) left this attack theoretical for cipher DES. Despite this, the method proved to be very successful. It can be implemented not only to the symmetrical blocking ciphers, but also to the stream ciphers [8], and to the hashing functions [9].

In the present article, we suggest a method of constructing of three-round differential for encryption algorithm Kuznyechik. The developed analysis scheme is based on the use of differential properties of S and L conversions of the algorithm Kuznyechik, and is designed to be able to determine the correct pair of texts for further analysis, which scope is the determination of secret encryption key. The scheme is developed to affect the active nonlinear components (S-blocks) a minimum number of times. As a result, for the suggested scheme the probability of finding the correct pairs of texts is equal to 2^{-108} encryptions. Also, we developed an algorithm of finding a secret key, the complexity of which is equal to $6*2^{-120}$ encryptions, using the cipher Kuznyechik. In this way, the total complexity of the analysis, including searching for the correct pairs of texts and bits of the secret encryption key is equal to $2^{-108} + 6*2^{-120}$ encryptions. The complexity of the suggested scheme is quite high in comparison with possibilities of modern computational tools, but it is much lower than the complexity of key compromising with the full enumeration method.

The article is organized in the following way. The first part of the article contains the description of the encryption algorithm Kuznyechik. The second part contains the study of differential properties of three main conversions of cipher Kuznyechik: exclusive disjunction operation, nonlinear replacement S-block, linear conversion L. The third part contains the description of constructing process of differential characteristic for three rounds of the cipher Kuznyechik. The key point of this section isn't the separate use of the differential properties of considered conversions, but a study of the work of these conversions in their entirety. Precisely this research allows to construct the three-round characteristics in such way, that in the process will be involved a minimum number of replacement S-blocks, in that way providing the maximum probability value. The fourth part describes the developed and implemented algorithm for finding the right pairs of texts for analysis of the cipher. The fifth part describes the process of key compromising due to use of the developed algorithm of analysis of three rounds of cipher Kuznyechik. The sixth part contains *theoretical calculations of the time required to conduct an attack using the most powerful supercomputers in the world, using both simple implementation and implementation with the use of special precomputation tables.*

1. Description of encryption algorithm Kuznyechik

The encryption algorithm Kuznyechik is a symmetric block cipher with a block length equal to 128 bits and key length equal to 256 bits. The cipher Kuznyechik is developed on the principle of SP-network, which allows to perform the conversions with entire input block, and not only over some part of it.

The encryption process consists of nine round, and every round includes three conversions: exclusive disjunction (xor) with round key, substitution with replacement blocks (S) and linear conversion (L). One round of encryption is shown in the Figure 1. Ten round keys are based on the master key of 256 bits. The first two keys are obtained by splitting the master key in a half, and the next eight keys are obtained with eight rounds of Feistel network. In each round are taken: key xor with a round constant, conversion with a substitution block, linear conversion. The round constant is obtained from the application to the round's number a linear conversion L. A detailed description of the cipher's work, and numerical examples which model the encryption process, can be found under the link [1].

The decryption function is executed in the reverse order, from bottom to top, using the round conversion, inversed to those that were used during the encryption.

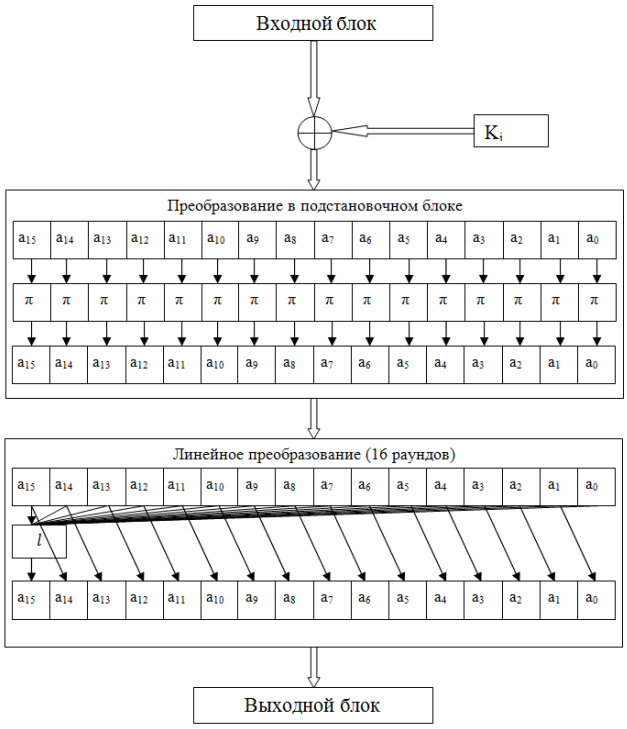

Figure 1. – One round of encryption

2. Analysis of the differential proprieties of main components of the cipher Kuznyechik

Before we begin to consider the differential proprieties of main components of the cipher Kuznyechik, we will speak about several notions and indications, which we will operate in future. The method of differential cryptanalysis not separately, but in pairs. It's necessary to control how is changing the diversity of two texts during their pass through the encryption rounds. As a measure of dissimilarity is considered the result of exclusive disjunction, which is called the ***differential***. Thus, a pair of texts at the entrance to the encryption algorithm X and X' is called the ***input differential*** and is denoted like $\Delta X=X \oplus X'$. A pair of texts at the egress from

the encryption algorithm Y and Y', is called the ***output differential*** and is denoted like ΔY. By a ***differential characteristic,*** we mean the most probable pair of values of the input and output differentials. ***The right pair of texts*** is a pair (open encrypted text): (X, Y) and (X', Y') for which $\Delta X=X\oplus X'$ and $\Delta Y=Y\oplus Y'$. The ***inactive S-block*** is a replacement block, through which passes the value of diversity, which is equal to zero. The ***active S-block*** is a replacement block, through which passes the value of diversity, which isn't equal to zero. The task of differential analysis is come down to the selection of a combination of input and output differentials, for which the total number of active S-blocks would be minimum (but nonzero) passing through all rounds of encryption. So, first, let's consider the differential properties for every conversion.

Exclusive disjunction operation with a secret key (xor). E. Biham and A. Shamir have already shown that the key can't affect the differential value. This is due to the fact that we're considering two parallel encryption processes, where is used the same secret key. This means that due to the properties of exclusive disjunction operation, the bits of secret key will be mutually destroyed – $X\oplus K_i\oplus X'\oplus K_i. =X\oplus X' = \Delta X$.

Linear conversion L. The conversion L in this algorithm is linear. This means that it transforms the value of diversity one-to-one, with a probability equal to 1. Therewith like the conversion of MixColumn in the encryption algorithm AES, it mixes all the bits of initial state. Therefore, if on the entrance to the L conversion you had only one nonzero byte, then at the output of L conversion all the bytes will be nonzero. It's easy to show that for the L conversion is realized the following property:

$$L(a)\oplus L(b) = L(a\oplus b)$$

Nonlinear conversion S. Let's denote the differentials that arrive on the input of the replacement S-block like ΔA, and the differentials received at the output of replacement S-block let's denote like ΔC. For the algorithm's analysis, it is necessary to complete the table, which will display the possibilities of appearance of different values of ΔC for the determined value ΔA. It is possible to find the algorithm of the creation of such tables in the studies of E. Biham and A. Shamir [6, 7]. The resulting table is big, and contains 256 rows and 256 columns. One part of the obtained table of probabilities is shown in the Table 1.

Table 1 – Part of the table of probabilities, created for the substitution block S in the process of the differential analysis.

ΔC ΔA	0	1	2	…	3e	3f	…	fe	ff
0	256/256	0/256	0/256	…	0/256	0/256	…	0/256	0/256
1	0/256	0/256	2/256	…	2/256	4/256	…	0/256	2/256
2	0/256	4/256	0/256	…	0/256	0/256	…	4/256	2/256
3	0/256	2/256	0/256	…	6/256	2/256	…	0/256	0/256
…	…	…	…	…	…	…	…	…	…
a5	0/256	0/256	0/256	…	0/256	2/256	…	2/256	0/256
a6	0/256	4/256	2/256	…	0/256	0/256	…	0/256	2/256
…	…	…	…	…	…	…	…	…	…
fe	0/256	0/256	2/256	…	2/256	0/256	…	2/256	0/256

ff	0/256	2/256	2/256	…	0/256	4/256	…	0/256	0/256

The rows of the Table 1 denote the input values of diversity ΔA in S-block, and columns denote the corresponding output values of the diversity ΔC, obtained at the output of S-block. At the intersection of rows and columns is showed a number of pairs of the differentials ΔA/ΔC that have a given input and output diversity. As a result of the analysis was established that the probability of matching for values ΔA/ΔC can be equal to 2/256, 4/256, 6/256 and 8/256, and also can be equal to zero. Herewith a nonzero value of the diversity at the input of the S-block will be never transformed into a zero diversity at the output.

If the analysis succeeds in finding the right pair of texts, then this gives us a possibility to assume, that the diversities have passed through the encryption rounds exactly as we have considered them during the construction of round differentials. The combination of differentials ΔA and ΔC allows to suppose the values $A \oplus K_i$ and $A' \oplus K_i$. With known A and A' it allows to determine K_i.

3. Construction of differential characteristic for three rounds of the cipher Kuznyechik

So, in the previous section it was noted that due to the byte mixing because of using of the conversion L, if at the input at the conversion L there was only one nonzero byte, so at the output of the conversion L most probably all the bytes would be nonzero. In means that if at the first encryption round will be used one active S-block, then in the second round of encryption, after L-conversion of first round, all 16 blocks will become active. And this means a sharp decrease of the probability value when passing the values of diversity through the S-block. Even if in accordance with the Table 1 consider the highest value of probability, equal to 8/256 = $\frac{1}{2^5}$, with 16 active S-blocks the probability value will change in $\left(\frac{1}{2^5}\right)^{16} = \frac{1}{2^{80}}$ times. And this means, that in the next round, most likely, will be also involved all 16 replacement blocks. Even with the most favorable value of probability of each considered block, every round will reduce the probability value in $\frac{1}{2^{80}}$ times. In this way, after three rounds the probability of differential will be equal to (under the most favorable probabilities, which also requires verification) $\frac{1}{2^5} * \frac{1}{2^{80}} * \frac{1}{2^{80}} = \frac{1}{2^{165}}$. After the fourth round the probability will approach the probability of exhaustive enumeration and will be equal to $\frac{1}{2^{245}}$, which will make useless the future analysis. We want to note, that these probability values have been calculated considering the most favorable values of probability. Although in practice it is still necessary to check whether they can be achieved simultaneously. Nevertheless, such an approach in the calculation of probabilities makes it clear that these calculations are pointless.

Our idea was based on using the properties of S and L conversions together. Initially there was a hypothesis, that if one active S-block in the first encryption round will lead to the 16 active S-blocks in the second encryption round, and in the third round of encryption we can get again only one active S-block? Schematically this can be represented as done in the Figure 2.

It was shown above that adding data with a round sub key doesn't change a differential value. In this case, our idea of constructing of a three-round differential boils down to following.

First round: At the algorithm's input is given an input differential ΔX, that have all bytes except one with zero values. The same differential value is given at the input of nonlinear bi-active conversion S, where it is changed into another byte in accordance with the probability table, created for this nonlinear conversion. In this way, we get one active S-block. The diversity at the output of the conversion S as before contains one nonzero byte. After this the diversity value is given at the input of linear conversion L, as a result of which at the output we get the diversity value with all 16 nonzero bytes.

Second round: The differential, obtained from the L conversion of the first round, remains unchanged after adding to the second-round sub key. Therefore, for nonlinear conversion S of a second round all the 16 bytes will be active. As a result of work of the S-block each byte of the diversity can be converted into the other value in accordance with the probability table. At the input of linear conversion L of the second round will arrive the diversity value, that has all 16 bytes with nonzero value. After the L-conversion these 16 nonzero bytes will be converted at the diversity value, that contains only one nonzero byte.

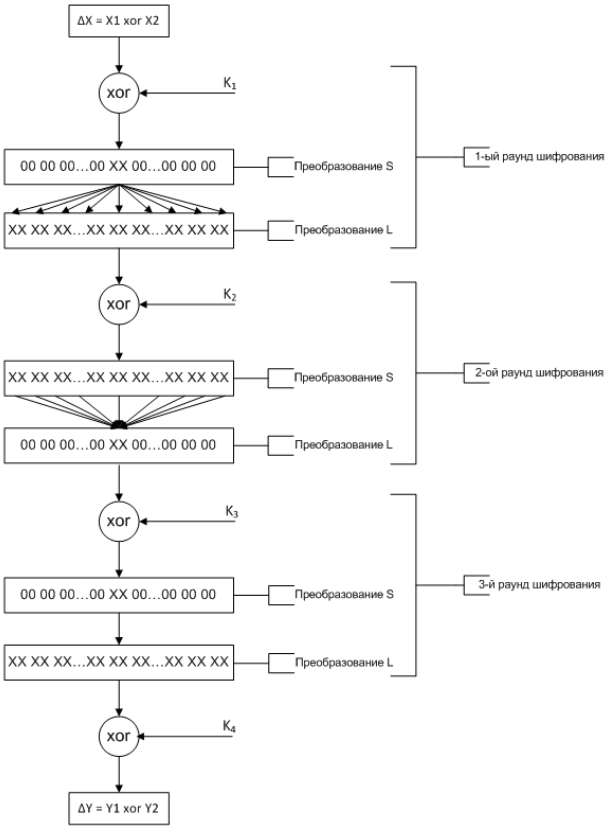

Figure 2 – Scheme of the differential analysis of three encryption rounds

Third round: The value of the differential obtained in the second round is added to the third-round key, which by analogy with first and second rounds doesn't affect the differential value. Therefore, in the S-conversion we get only one active S-block. Nonzero byte of diversity in accordance with the probability table is changed into other probable value. Then, as a result of the conversion L the diversity with one nonzero byte is converted to a diversity value, which again consists of 16 nonzero bytes.

Therefore, for the considered scheme of analysis we obtain only 18 active S-blocks instead of 33 as was considered before. Even if we assume that we can select the differential values that will correspond to the Figure 2, with the minimum probability values for each of

active S-blocks 2/256 = $\frac{1}{2^7}$, we will get a three-round differential with probability equal to $\left(\frac{1}{2^7}\right)^{18} = \frac{1}{2^{126}}$, what is less, that the value assumed before $\frac{1}{2^{165}}$. Looking ahead, we want to say that we have managed to get a three-round characteristic, the appearance probability of which is equal to $\frac{1}{2^{108}}$, what means, that during the use of S-blocks were involved not the most minimal probabilities.

So, in order to construct a differential, the scheme of which is shown in the Figure 2, we need to consider of work of the S and L conversions together. In a general form, this can be represented schematically as done in the Figure 3. It's necessary that in the first round one nonzero byte of differential after L can be converted into 16 nonzero bytes (left part in the Figure). In the second round it's necessary, that 16 nonzero bytes of differential after L can be converted into 1 nonzero byte. Or, if we go from the opposite, we are interested in such cases when one nonzero byte of differential after L_{inv} can be converted into 16 nonzero bytes (right part in the Figure). Herewith, the differentials of the right and the left parts must be connected with S conversion, in other words, in the probability table must be presented at least a minimum probability of 2/256 conversion of diversity value from the left part of the Figure 3 to its right part.

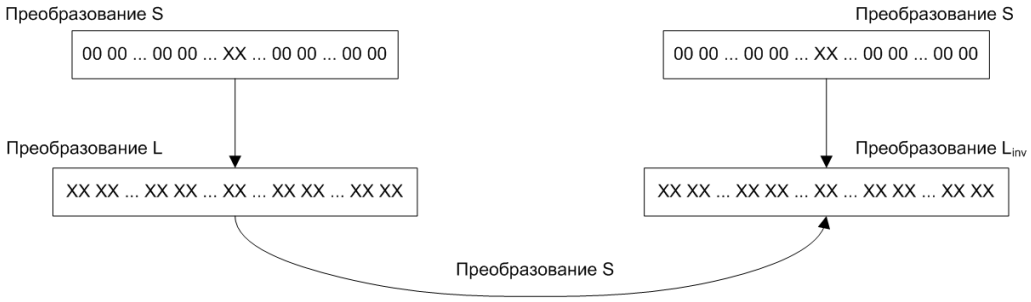

Figure 3 – Reliance of conversions S and L

Thus, the suggested scheme of relations between S and L conversions implies, that through the conversions L and $L_{inv,}$ it's necessary to pass sixteen different variants of filling one byte from 00 to FF, which as a result of this conversion will be decomposed into 16 bytes. Let's formulate the suggested algorithm in steps:

First step: at the input of the conversion L is given a block, where only one byte has a nonzero value. Through the L-conversion will pass 16 different variants of filling of every possible byte from 00 to FF. Each one-byte value will be decomposed through the conversion L into a value, that contains 16 bytes. As a result, we get probable diversity values at the input of the S-conversion of the second round. The total of these values will be 255*16=4080. These are probable values at the output of the L-conversion of the first round (left part of the Figure 3). The same values will arrive at the input of the S-block of the second encryption round. Let's denote them like ΔA.

Second step: Is similar to the first step, except that instead of L-conversion is used a conversion L_{inv}. In this way, we get other 4080 values. these are probable values at the input of L-conversion of the second round (right part of the Figure 3). The same values will arrive at the input of the S-block of the second encryption round. Let's denote them like ΔC.

Third step: As a result of the first and second steps, we have the differentials ΔA and ΔC at the input and output of the S-conversion of the second round. As for each value ΔA not all the values ΔC are the same, we must determine, what differentials ΔA that arrive at the input of the replacement S-block from 4080 values will have a nonzero probability to be obtained as ΔC values due to conversion. For this it's necessary to decompose each of 4080 value ΔA into 16 bytes and put them at the input of the probability table. Then in the same way we must decompose into bytes each value ΔC. For some 16-bytes differentials ΔA from 4080 values in accordance with the table will be found the differentials ΔC with nonzero probability. In total, as a result of considering all possible combinations, we obtained 13 pairs of values ΔA/ΔC, which are shown in the Table 2.

I've found other probabilities for each byte of ΔA/ΔC – do they need?

Table 2 – Found pairs of values ΔA / ΔC

ΔA	ΔC
f3ab8c55c199996fc5a4f2381976846	51ac91f0df2470190ad86a256131163
1a76bc71665284b01a3e595982599369	ba5a9d5e6d2b6431ac6b9cb72dc5a7a1
5cfbaa318fd91c774940bef22a5f86	1f8355405427f8e7d8c71cc07f2288c6
ac8ea817121caa3445efd4b9c43e875f	c6e355f95177d1bd58f9a4145283a143
522cbe6cbcf88eaf4963f28f8f29e62	353cceb5eab273db5790cc909cfa9
2bc22a75e57cbd804b35bf31ee5167	54be1b26f4dbe5b1f6a2a66a61e384d
5f28ebaf31b588b3f8f23923e399f0ef	b4eba9c9151b1fbd907f4f4d419fcdaa
7dbda6246e653ba46ec427fafa9a462f	5d87c43030f77ec08a9f25c0b8413318
efd9d1a17499185749edf1d1d6ee4c	bbd9f4a6c11bf5e154a2c52495e1ddd9
8fb8e26a91ccf9b72d6d5dce4f9ad4a9	b86cc61926ba16e11d19dce66e78450
6a7c998e18379bf720d423721d3c7e63	c6fc783ae4466b402df56e30dff1f8d4
337ddbfefc424a38d45ae559c2d2cd36	fd58a4739c68ed296855b6723e9fb3
5c9ce97665a2afd9172a54a881949c	3a3d1b8c2658ea7f8a958b2f866ecb5b

From the found values ΔA/ΔC it's easy to determine the values of differentials at the input and output of 3-round algorithm Kuznyechik ΔX/ΔY, and the probability of their appearance. For the first pair of values ΔA/ΔC with the highest probability equal to $\frac{6}{256}$ at the algorithm's input is given a value ΔX = 00000000002000000000000000000000, and at the output are obtained three values with the same probability ΔY_1 = e07d0b92e148a1f4774154a6b06e99f0, ΔY_2 = 0d38eb47ba753eb564333cbda803ac2c, ΔY_3 = acb714a9647d96d76b3a89ab83f1bd71. One-byte values, which were decomposed into 16-byte values, was obtained with the maximum probability equal to $\frac{6}{256}$.

4. Algorithm of finding of the right pairs of texts for a three-round characteristic

In general case, if you know the value of the differential and its probability, then in the search algorithm for the right pairs of texts there is nothing difficult. The algorithm of searching for the right pairs of texts boils down to following:

1. Randomly generate the text X.
2. Determine the text X': $X' = X \oplus \Delta X$
3. Encrypt the text X, obtain the cipher Y
4. Encrypt the text X', obtain the cipher Y'
5. If $\Delta Y = Y \oplus Y'$, then (X,Y) and (X', Y') – is a right pair of texts, otherwise turn to step 1.

The complexity of the analysis will be determined by the probability of finding such differentials. We have made theoretical calculations of how much time will be spent on searching for one correct pair of texts using the most powerful supercomputers in the world. These calculations will be described in the section 6.

Whatever it was, for us it's impossible to perform such computational calculations. But in order to confirm the efficiency of the proposed differential scheme for three rounds of encryption we decided to go «from the reverse». Knowing how diversities should be converted when passing through the rounds of encryption, not by searching, but selecting the texts and secret keys, that will correspond to the obtained round characteristic. It was done with only purpose – to show that such values of text exist, and the suggested scheme is workable.

A scan be seen from the Table 2, were found 13 pairs of values $\Delta A/\Delta C$. From these values it's possible to determine, which values of the differentials were given at the input of the encryption algorithm, and which will be obtained at the output. For this it's necessary to perform some reverse conversions.

First step: searching for the open texts. ΔA is given at the input of L_{inv} conversion, reversed with conversion L of the first round, as a result $L_{inv}(\Delta A)$ we get an output of the S conversion of the first round. Then, the previous result is given at the input of S_{inv} conversion of the first round, reversed with conversion S, as a result $S_{inv}(L_{inv}(\Delta A))$ we get a differential given at the input of the algorithm.

Second step: searching for texts based on three rounds. ΔC is given at the input of conversion L of the second round, and we get $L(\Delta C)$. Then the obtained result is given at the input of the replacement block of the third round, where as a result of rearrangement of S in accordance with the created Table 1, which contains the probability values, we get $S(L(\Delta C)$. Ultimately, the last obtained result as given at input of the conversion L, where is changed in accordance with the algorithm. In this way, we get a necessary output, which contains the researched differential ΔY.

If find the bytes, which constituting the differentials ΔA / ΔC, compile the texts composing them and pass them through three-rounds conversions, then for certain values of a secret-key, according to the developed and described method, the correct pair of texts will form the required values of the differential. to prove the operability of the proposed method, were obtained the initial values, some of which are shown in the Table 3 and the final values, some of which are shown in the Table 4. Where A and A' are the texts, constituting the found differential ΔA, so C and C' are the texts composing the differential ΔC. If consider the master key like K=

8899aabbccddeeff0011223344556677fedcba98765432100123456789abcdef, than round keys K_1, K_2, K_3 and K_4 will be equal to following values:

– K_1=8899aabbccddeeff0011223344556677;
– K_2=fedcba98765432100123456789abcdef;
– K_3=db31485315694343228d6aef8cc78c44;
– K_4=3d4553d8e9cfec6815ebadc40a9ffd04.

Table 3 – Initial texts

$K_1 \oplus (S_{inv}(L_{inv}(A \oplus K_2)))$	$K_1(S_{inv}(L_{inv}(A' \oplus K_2)))$
8998079702eaede8422655046852245d	8998079702fbede8422655046852245d
8998079702eaede8422655046852cf5d	8998079702fbede8422655046852cf5d
8998079702eaede8422655046852825d	8998079702fbede8422655046852825d

Table 4 – Values based on three rounds

$K_4 \oplus (L(S(C)))$	$K_4 \oplus (L(S(C')))$
8ccf9f63b68216e7e87fb65cfce4364f	27ac6bf0c98ce6b78ed52e43f34d17cb
7152f68de9e8354c0db9cb0594ee651e	da31021e96e6c51c6b13531a9b47449a
8ef7bcaec365c4890540ec622239d595	2594483dbc6b34d963ea747d2d90f411

5. Algorithm of searching for the secret key based on the analysis of the right pairs of texts

The method of differential cryptanalysis, as a rule, boils down to compromising round keys for the purpose of information decryption. It's not enough to find the right pairs of texts, it's still needed to be able to use them to restore the information about the bits of a secret key. The method of cryptanalysis based on the previously found right pairs of texts boils down to following.

Considering three rounds of encryption, we use 4 round keys (Figure 2). A feature of the algorithm Kuznyechik is that the first two round keys are the halves of a 256-bit master-key, on the basis of which are produced other round sub keys. Considering the fact, that suggested algorithm provides for finding of key K_1, and then key K_2, from which are produced other keys K_3 and K_4. The scheme of suggested algorithm is shown in the Figure 4.

As it was said in the 4th part of this article, when performing reverse operations on the found ΔA, more specifically $S_{inv}(L_{inv}(\Delta A))$, we get a differential, which arrives at the input of the conversion S of the first round, and with $L_{inv}(\Delta A)$ we get the output of the same conversion. In this way, if give at the input of the algorithm, shown on the Figure 2, the previously selected pair of texts X and X', than these texts while adding to a number, obtained from $S_{inv}(L_{inv}(\Delta A))$, give the value of the key K_1.So as the value of $S_{inv}(L_{inv}(\Delta A))$ consists from one byte (8 bit) and has a probability equal to 6 in accordance with the probability table 6, than it gives us 6 probable values of one byte of the key K_1. Therefore, a number of all possible variants of 128-bit key K_1 is equal to $2^{120} \times 6$.

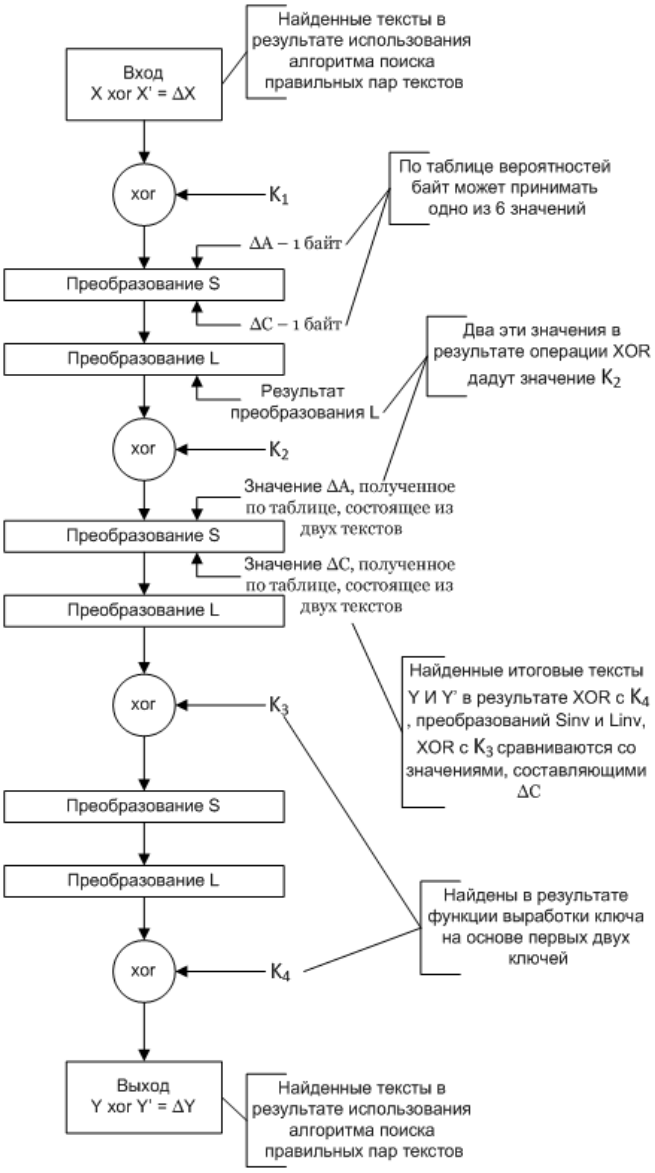

Figure 4 – Algorithm of searching for key bits

For checking of one possible variant of key K_1, it's necessary to perform several operations. Proceeding from the Figure 4, first we give at input two selected texts X and X', and then add them with one possible key K_1 from the total number of possible keys K_1. Then the obtained values pass through the S conversion of the first round, and then through the L-conversion. The conversion results are added with values, composing the differential ΔA, that arrives at the input of the S-conversion of the second round. On this stage, we get one of possible values of key K_2. Then we put together the keys K_1 and K_2, and obtain the 256-bit master-key, from which we produce the keys K_3 and K_4.

The next step is to take two possible texts Y and Y', obtained as a result of using algorithm of finding of right pairs of texts for the differential cryptanalysis, and add them to the key K_4. Now, obtained values are changed with L_{inv} and S_{inv} conversions of the third round and then they are added to the key K_3. Then the same values are changed with L_{inv} conversion of the second round and are compared with the values, composing the differential ΔC. If the values are equal, then the keys are saved as probably correct. If the values are not equal, then the algorithm of the actions is repeated anew: two texts X and X' are added to the next key K_1, from all possible values of K_1. And all the future actions are performed on the same scheme. In fact, key

sampling boils down to the performing of all operations of three-round encryption. That's why in total the algorithm will use a maximum of $2^{120}\times6$ encryptions. In this case all 4 keys will be found, which is much smaller number, than a full-case search of the key, where the total possible number of keys represents a number equal to 2^{128} for only first key K_1, a number of values for searching for the second key K_2 by the full search is also equal to 2^{128}, which in general makes it possible to estimate the complexity of searching for the key by the full search as 2^{256}.

Therefore, the complexity keys finding by developed method of cryptanalysis is equal to $2^{120}\times6$ of 3-round encryptions. The total complexity of the performing of differential cryptanalysis of three rounds of the encryption can be estimated as $2^{120}\times6+2^{108}$ of encryptions.

Where have I put this section – in the conclusion (as gived in the annotation) or remain it in the 6'th section (as given in the introduction)??

6. An approximate estimate of the time required to analyze the algorithm Kuznyechik on modern supercomputers from TOP 500.

Performing operations based on the developed algorithm of differential cryptanalysis of three rounds of the algorithm Kuznyechik, we can estimate the approximate time costs required for the analysis, for example, on modern supercomputers from TOP 500. For evaluation were selected first 10 computers from the overall rating. [10].

Approximate time costs can be calculated using the proportions. If on the used computer the frequency is equal to 1,8 GHz and one unit of data is calculated in 0,0022 sec, then with the frequency of supercomputer's core equal to, for example, 1.45 GHz, one unit of data will be calculated in 0,0027 sec. To calculate the complexity of the analysis you need to divide the value of total complexity by the number of cores of the computer, and then, for calculation of the time of analysis, you need to multiply the obtained value by the number of seconds of calculation of one block. The results of calculations are given in the Table 5.

Table 5 – Calculation of time costs

N°	Supercomputer and its characteristics	Approximate time costs
1	Sunway TaihuLight – 10,649,600 cores, 1.45GHz	1 block – 0,0027 sec Complexity of analysis – $2^{85}+6\times2^{92}$ Time of analysis – $(2^{85}+6\times2^{92})\times0{,}0027$
2	Tianhe-2 (MilkyWay-2) – 3,120,000 cores, 2.2 GHz	1 block – 0.0018 sec Complexity of analysis – $2^{87}+6\times2^{99}$ Time of analysis – $(2^{87}+6\times2^{99})\times0{,}0018$
3	Piz Daint – 361,760 cores, 2.6 GHz	1 block – 0.0015 sec Complexity of analysis – $2^{90}+6\times2^{102}$ Time of analysis – $(2^{90}+6\times2^{102})\times0{,}0018$
4	Titan – 560,640 cores, 2.2 GHz	1 block – 0.0018 sec Complexity of analysis – $2^{89}+6\times2^{101}$ Time of analysis – $(2^{89}+6\times2^{101})\times0{,}0018$
5	Sequoia – 1,572,864 cores, 1.60 GHz	1 block – 0.0025 sec Complexity of analysis – $2^{88}+6\times2^{100}$ Time of analysis – $(2^{88}+6\times2^{100})\times 0.0025$
6	Cori – 622,336 cores, 1.4 GHz	1 block – 0.0028 sec Complexity of analysis – $2^{89}+6\times2^{101}$ Time of analysis – $(2^{89}+6\times2^{101})\times0{,}0028$
7	Oakforest – 556,104 cores, 1.4 GHz	1 block – 0.0028 sec Complexity of analysis – $2^{89}+6\times2^{101}$

		Time of analysis – $(2^{89}+6\times2^{101})\times0,0028$
8	K computer – 705,024 cores, 2.0 GHz	1 block – 0.0019 sec Complexity of analysis – $2^{89}+6\times2^{101}$ Time of analysis – $(2^{89}+6\times2^{101})\times0,0019$
9	Mira – 786,432 cores, 1.60 GHz	1 block – 0.0025 sec Complexity of analysis – $2^{89}+6\times2^{101}$ Time of analysis – $(2^{89}+6\times2^{101})\times0,0025$
10	Trinity – 301,056 cores, 2.3 GHz	1 block – 0.0017 sec Complexity of analysis – $2^{90}+6\times2^{102}$ Time of analysis – $(2^{90}+6\times2^{102})\times0,0017$

Where can I find Roma's publication in English, to insert it into the list of sources?

It is also possible to estimate the time costs when using the high-speed realization of the encryption algorithm Kuznyechik with use of the precomputation tables. The frequency of the machine, on which were performed high-speed calculations 2.67GHz, and the information processing speed is 120 MB/sec. Consequently, 1 block is processed approximately in 0,000001 seconds. The results of time calculations are shown in the Table 6.

Table 6 – Calculation of time costs using a high-speed realization of the algorithm Kuznyechik

N°	Supercomputer and its characteristics	Approximate time costs
1	Sunway TaihuLight – 10,649,600 cores, 1.45GHz	1 block – 0.0000018 sec Complexity of analysis – $2^{85}+6\times2^{92}$ Time of analysis – $(2^{85}+6\times2^{92})\times$ 0.0000018
2	Tianhe-2 (MilkyWay-2) – 3,120,000 cores, 2.2 GHz	1 block – 0.0000012 sec Complexity of analysis – $2^{87}+6\times2^{99}$ Time of analysis – $(2^{87}+6\times2^{99})\times$ 0.0000012
3	Piz Daint – 361,760 cores, 2.6 GHz	1 block – 0,000001 sec Complexity of analysis – $2^{90}+6\times2^{102}$ Time of analysis – $(2^{90}+6\times2^{102})\times$ 0,000001
4	Titan – 560,640 cores, 2.2 GHz	1 block – 0.0000012 sec Complexity of analysis – $2^{89}+6\times2^{101}$ Time of analysis – $(2^{89}+6\times2^{101})\times$ 0.0000012
5	Sequoia – 1,572,864 cores, 1.60 GHz	1 block – 0.0000017 sec Complexity of analysis – $2^{88}+6\times2^{100}$ Time of analysis – $(2^{88}+6\times2^{100})\times$ 0.0000017
6	Cori – 622,336 cores, 1.4 GHz	1 block – 0.0000018 sec Complexity of analysis – $2^{89}+6\times2^{101}$ Time of analysis – $(2^{89}+6\times2^{101})\times$ 0.0000018
7	Oakforest – 556,104 cores, 1.4 GHz	1 block – 0.0000018 sec Complexity of analysis – $2^{89}+6\times2^{101}$ Time of analysis – $(2^{89}+6\times2^{101})\times$ 0.0000018
8	K computer – 705,024 cores, 2.0 GHz	1 block – 0.0000013 sec Complexity of analysis – $2^{89}+6\times2^{101}$ Time of analysis – $(2^{89}+6\times2^{101})\times$ 0.0000013
9	Mira – 786,432 cores, 1.60 GHz	1 block – 0.0000017 sec Complexity of analysis – $2^{89}+6\times2^{101}$ Time of analysis – $(2^{89}+6\times2^{101})\times$ 0.0000017
10	Trinity – 301,056 cores, 2.3 GHz	1 block – 0.0000012 sec Complexity of analysis – $2^{90}+6\times2^{102}$ Time of analysis – $(2^{90}+6\times2^{102})\times$ 0.0000012

Therefore, from the obtained results of the estimation of time characteristics, when using the algorithm with high-speed realization with analysis with supercomputers from TOP 500 the speed of analysis essentially increases.

Conclusion

As a result of work on this project we for the first time studied and obtained the differential properties of the encryption algorithm Kuznyechik. On the basis of our studies was found a connection with S and L conversions, which allowed to develop the algorithm of differential cryptanalysis of three rounds of the cipher Kuznyechik, previously unnamed in open literary sources.

For the suggested scheme of differential cryptanalysis, we developed the algorithm of finding the right pairs of texts for the cipher analysis. On the basis of the three-round differential we developed the algorithm of finding the secret key with less complexity, than complexity when using a key by full searching. The developed algorithms allow to estimate the total complexity of the analysis, which is equal to $2^{-108} + 6*2^{-120}$.

As a result of the researches was developed and tested the encryption algorithm Kuznyechik, was developed a method of differential analysis of three rounds of algorithm Kuznyechik, and for this algorithm was created a scheme of finding the correct pairs of texts for the differential analysis of three round of encryption. The program is realized in the programming language C++ in the development framework Microsoft Visual Studio C++. The results of the researches are described in detail, are shown in examples, are structured in chronological order and are visually displayed in the form of illustrations, tables and diagrams in the text of this article. The theoretical calculations were also performed on the time required to conduct an attack using the most powerful supercomputers in the world both using the normal implementation and implementation with use of special precomputation tables.

The results of the work were used in the performing of research works under the grant RFFI N°17-07-00654-a «Development and studying of sequential and parallel analysis algorithms».

Bibliographic list

1 Cryptographically protection of information Block ciphers GOST R 34.12–2015 [Internet source].– URL: http://tc26.ru/en/standard/gost/GOST_R_34_12_2015_ENG.pdf

2 E.A. Ishchukova, L.K. Babenko, M.V. Anikeev Fast Implementation and Cryptanalysis of GOST R 34.12-2015 Block Ciphers // 9th International Conference on Security of Information and Networks, SIN 2016, Newark, Nj, 20-22 July 2016. – P. 104 – 111.

3 AlexBiryukov, LéoPerrin, AlekseiUdovenko. Reverse-Engineering the S-Box of Streebog, Kuznyechik and STRIBOBr1 (Full Version) (2016). https://eprint.iacr.org/2016/071.pdf

4 Alex Biryukov, Léo Perrin, Aleksei Udovenko. Reverse Engineering the S-Box of Streebog, Kuznechik and Stribob (2015).
http://crypto.2015.rump.cr.yp.to/1ea2c6c01144e0e7f6b14b324c5e4562.pdf

5- Riham AlTawy and Amr M. Youssef. A Meet in the Middle Attack on Reduced Round Kuznyechik (April 17, 2015) https://eprint.iacr.org/2015/096.pdf

6 Biham, E., and Shamir, A., "Differential cryptanalysis of DES-like cryptosystems", Journal of Cryptology, Vol. 4, No. 1, pp. 3-72 (full issue), 1991.
http://www.cs.bilkent.edu.tr/~selcuk/teaching/cs519/Biham-DC.pdf

7 Biham, E., and Shamir, A., "Differential cryptanalysis of the full 16-round DES", Advances in cryptology, proceedings of CRYPTO'92, pp. 487-496, 1992.
8 Eli Biham, Orr Dunkelman, Differential Cryptanalysis in Stream Ciphers // Cryptology ePrint Archive, Report 2007/218, 2007. http://eprint.iacr.org/
9 EliBiham, AdiShamir Differential Cryptanalysis of Hash Functions // Differential Cryptanalysis of The Data Encryption Standard, Springer, 1993, ISBN 978-1-4613-9314-6. – P. 133 – 148
10 Statistics on supercomputers – TOP 500 [Internet source].– URL: https://www.top500.org/lists/2017/06/