МИНИСТЕРСТВО СРЕДНЕГО И ВЫСШЕГО ОБРАЗОВАНИЯ
РЕСПУБЛИКИ УЗБЕКИСТАН

НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ УЗБЕКИСТАНА
ИМЕНИ МИРЗО УЛУГБЕКА

На правах рукописи

УДК

Латыпов Азиз Усманович

# Исследование блочных алгоритмов криптологии

5А130202 Прикладная математика и информационные технологии

на соискание академической ученой степени
Магистра

# ДИССЕРТАЦИЯ

Ташкент 2013

## Аннотация

В данной работе, рассмотрены блочные алгоритмы шифрования, методы их построения и криптографического анализа. Проанализированы два основных подхода, используемые при проектировании блочных алгоритмов шифрования: сеть Фейстеля (Feistel Network, также известный как Luby-Rakoff Network) и подстановочно-перестановочная сеть (SP-Network). Самым важным элементом в обеих из этих схем является таблица замены, на которой и основывается стойкость всей схемы. Для оценки криптографической стойкости таблиц замены (в работе [3]) были введены специальные критерии: строгий лавинный эффект и независимость битов. Автором построен параллельный алгоритм высокой гранулярности для анализа таблиц замены генерируемых на параметрах для стандарта Республики Узбекистан <<Алгоритм шифрования данных>> O'zDSt 1105:2009 с использованием технологии MPI. Данный алгоритм в автоматическом режиме оценивает криптографическую стойкость таблиц замены, используемых в алгоритме шифрования O'zDSt 1105:2009, по указанным выше критериям. Также автором доказаны две теоремы, благодаря которым можно, во-первых, упростить алгоритмы анализа. Во-вторых, они позволяют понять связь между классическими методами криптографического анализа и критериями, налагаемыми на таблицы замен современных блочных шифров.

## Abstract

In this paper we consider the block encryption algorithms, methods of constructing them and their cryptoanalysis. Analyzed the two main approaches used in the design of block ciphers: Feistel network (Feistel Network, also known as Luby-Rakoff Network) and substitution-permutation network (SP-Network). The most important element in both of these schemes is the Substitution-Box (S-box) on which the resistance of the circuit based. To assess the cryptographic strength of the S-boxes specific criteria were introduced in paper [3]: strict avalanche effect and the independence of bits. The author has constructed a parallel high granularity algorithm for the analysis of S-boxes generated by the parameters of standard "Data encryption algorithm" O'zDSt 1105:2009 of the Republic of Uzbekistan using the MPI technology. This algorithm automatically evaluates the cryptographic security of S-boxes used in the encryption algorithm O'zDSt 1105:2009, according to the above criteria. Also the author of two theorems that help you, first of all, to simplify the analysis algorithms. Second, they allow us to understand the relationship between the classical methods of cryptanalysis and the criteria imposed for the S-boxes of contemporary block ciphers.

# Оглавление

# Введение

Одной из приоритетных задач в международном сообществе является обеспечение информационной безопасности. Развивается сотрудничество между государствами в данной области. В Республике Узбекистан особое внимание уделяется вопросам защиты государственной тайны и конфиденциальной информации. В функционировании, как коммерческих организаций, так и государственных предприятий первостепенную роль играет обеспечение мер, необходимых для защиты информации, представляющей научно-техническую, технологическую, производственную и финансово-экономическую ценность. В последнее время проблемы информационной безопасности приобрели особую значимость в связи с необходимостью усиления борьбы с проявлениями терроризма, приобретающего международные масштабы.

Изначально криптография изучала методы шифрования информации — обратимого преобразования открытого (исходного) текста на основе секретного алгоритма и/или ключа в шифрованный текст (шифротекст). Традиционная криптография образует раздел симметричных криптосистем, в которых зашифрование и расшифрование проводится с использованием одного и того же секретного ключа. Помимо этого раздела современная криптография включает в себя асимметричные криптосистемы, системы электронной цифровой подписи (ЭЦП), хеш-функции, управление ключами, получение

скрытой информации, квантовую криптографию.

Для современной криптографии характерно использование открытых алгоритмов шифрования, предполагающих использование вычислительных средств. Известно более десятка проверенных алгоритмов шифрования, которые при использовании ключа достаточной длины и корректной реализации алгоритма криптографически стойки.

Криптосистемы защиты информации обычно построены на основе композиции нескольких отображений, допускающих удобную аппаратную и/или программную реализацию. При этом криптосистема в целом реализует некоторое множество подстановок, а в некоторых случаях - группу подстановок. В то же время во многих криптосистемах защиты информации можно выделить функциональную часть, множество реализуемых отображений которой образует полугруппу или некоторое подмножество полугруппы, так как при построении криптографических алгоритмов используются не только обратимые (групповые), но и необратимые (полугрупповые) отображения. Например, в блочных криптосистемах полугрупповые преобразования могут использоваться при построении раундовых отображений. В поточных шифрах полугрупповые преобразования нередко используются в алгоритмах выработки гаммы. Например, функционирование алгоритма Solitaire описывается математической моделью автомата с частичными функциями переходов, порождающими полугруппу. Существуют классы генераторов гаммы, построенных на основе необратимых преобразований (например, генераторы самоусечения). Принципиальной идеей построения таких генераторов является усложнение слабых, в частности, линейных преобразований с целью повышения уровня защиты информации при несущественном усложнении реализации.

Таким образом, композиции обратимых и необратимых отображений сочетаются при построении криптографических алгоритмов защиты информации. Это обуславливает необходимость изучения криптографических свойств композиций преобразований информации, построенных с использованием как групповых, так и полугрупповых преобразований. Базовые критерии качества шифрующих отображений были сформулированы еще Клодом Шенноном в известном докладе 1949 года. Дальнейшая их конкретизация применительно к различным классам шифров привела к исследованию разнообразных криптографических свойств отображений информации. В качестве таких примитивов Х. Фейстель предложил использовать подстановки (S-блоки) и перестановки (P-блоки). Схемы, реализующие эти преобразования, делятся на две категории: Сеть Фейстеля и SP-сеть. Часто используемыми криптографическими примитивами являются также преобразования типа циклический сдвиг или гаммирование. Результаты этих исследований нашли отражение в многочисленных работах как отечественных, так и зарубежных специалистов (М. М. Глухов, Б. А. Погорелов, В. Н. Сачков, А. Шамир, М. Хеллман, Р. Рюппель и многие другие).

Имеется немало примеров, показывающих, что не всякая композиция отображений имеет хорошие криптографические свойства с точки зрения защиты информации. Поэтому для оценки уровня криптографической защиты информации важным является описание подмножеств слабых элементов полугруппы или группы, описывающей функционирование криптосистемы. Такое описание может быть использовано для построения тех или иных методов дешифрования.

Блочный шифр — разновидность симметричного шифра, оперирующего группами бит фиксированной длины — блоками, характерный размер которых

меняется в пределах 64 — 256 бит. Если исходный текст (или его остаток) меньше размера блока, перед шифрованием его дополняют. Фактически, блочный шифр представляет собой подстановку на алфавите блоков, которая, как следствие, может быть моно- или полиалфавитной. Блочный шифр является важной компонентой многих криптографических протоколов и широко используется для защиты данных, передаваемых по сети.

К достоинствам блочных шифров относят сходство процедур шифрования и расшифрования, которые, как правило, отличаются лишь порядком действий. Это упрощает создание устройств шифрования, так как позволяет использовать одни и те же блоки в цепях шифрования и расшифрования. Гибкость блочных шифров позволяет использовать их для построения других криптографических примитивов: генератора псевдослучайной последовательности, поточного шифра, имитовставки и криптографических хэшей.

Во многих странах приняты национальные стандарты шифрования. В 2001 году в США принят стандарт симметричного шифрования AES на основе алгоритма Rijndael с длиной ключа 128, 192 и 256 бит. Алгоритм AES пришёл на смену прежнему алгоритму DES, который теперь рекомендовано использовать только в режиме Triple DES. В Республике Узбекистан действует стандарт O'zDSt 1105:2009 [8], описывающий алгоритм блочного шифрования.

В настоящей работе общий подход к изучению слабых криптографических отображений. Актуальность данного направления исследований вытекает из необходимости исследования возможных слабостей отображений информации для широкого класса криптосистем с целью оценки уровня защиты информации.

Методы исследования: булева алгебра, теория групп, линейный анализ, диф-

ференциальный анализ, параллельные технологии, библиотека OpenMPI, лавинный эффект, критерий независимости битов, теория вероятностей.

Исследованость данной темы. К.Шеннон в своей статье “Communication Theory of Secrecy Systems”, опубликованной в 1949 г., ввёл понятия конфузии и диффузии в качестве методов, усложняющих статистический криптоанализ. Согласно Шеннону:

Диффузия — метод, при котором избыточность в статистике входных данных “распределяется” по всей структуре выходных данных. При этом большие объёмы выходных данных требуются для статистического анализа, скрывается структура исходного текста. Реализуется при помощи P-блоков.

Конфузия — метод, при котором зависимость ключа и выходных данных делается как можно более сложной, в частности, нелинейной. При этом становится сложнее делать заключения о ключе по выходным данным, а также об исходных данных, если известна часть ключа. Реализуется при помощи S-блоков.

Важным криптографическим свойством для шифрования является лавинный эффект (англ. avalanche effect), который означает, что изменение значения малого количества битов во входном тексте или в ключе ведет к “лавинному” изменению значений выходных битов шифротекста. Другими словами это зависимость всех выходных битов от каждого входного бита. Термин лавинный эффект впервые введен Фейстелем в статье Cryptography and Computer Privacy, опубликованной в журнале Scientific American в Мае 1973 года. [1], хотя концептуальное понятие использовалось еще Шенноном. Если криптографический алгоритм не обладает лавинным эффектом в достаточной степени, криптоаналитик может сделать предположение о входной информации, основываясь на выходной информации. Таким обра-

зом, достижение лавинного эффекта является важной целью при разработке криптографического алгоритма [2].

Лавинный эффект является следствием хорошей конфузии и диффузии [9]. В данной работе автор сосредоточил свое внимание на проблеме оценки качества самой сложной части раундового криптографического преобразования – таблице замены (S-блок).

С. Е. Таварес и А. Ф. Вебстер в своей работе по изучению и описанию S-блоков [3] определили два критерия для оценкци *качества* криптографического преобазования:

1. строгий лавинный критерий,
2. критерий независимости битов.

Также принято учитывать такие требования к булевым функциям, участвующим в преобазовании как

1. нелинейность,
2. сбалансированность.

В рамках диссертации осуществлено применение полученных теоретических результатов: в криптографическом стандарте Республики Узбекистан O'zDSt 1105:2009 <<Алгоритм Шифрования Данных>> [8] таблицы замены являются перестановкой на 256 значений и порождаются в зависимости от битов расширенного ключа $k_{se}$ на параметрах $d$, $L$, $R$ по формуле

$$\left\{[((i+L) \mod 256)+1]^d\right\} \mod 256$$

где возведение в степень $d$ – является возведением в степень с параметром $R$. Всего таблиц замены, порождаемых различными значениями этих параметров – **4 161 600**.

Поэтому нашей основной целью было построение метода автоматической проверки *качества* таблиц замены используемых в алгортиме шифрования стандарта O'zDSt 1105:2009 по указанным выше критериям.

Алгоритм, используемых в стандарте Республики Узбекистан O'zDSt 1105:2009 <<Алгоритм Шифрования Данных>> [8], является относительно новым и поэтому мало исследован. Так нами построен параллельный алгоритм, позволяющий в автоматическом режиме оценивать таблицы замен для алгортима O'zDSt 1105:2009.

Полученный алгоритм позволяет анализировать произвольные таблицы замены. Например, можно модифицировать его, для исследования стойкости *всех* таблиц замены на 4 бита или некоторых конкретных классов функций. Так как S-блок является "сердцем" всей криптосистемы, то его исследование имеет большую практическую значимость.

Научная новизна работы характеризуется следующими результатами:

1. доказаны две теоремы, проливающие свет на взаимосвязь между классическими методами анализа и критериями, предъявляемыми к элементарным преобразованиям блочного алгоритма шифрования;

2. построен алгоритм, автоматически обеспечивающий высокий уровень масштабируемости и равномерную загрузку всех доступных машин;

3. с использованием данного алгоритма изучены таблицы замены криптографического стандарта Республики Узбекистан O'zDSt 1105:2009 <<Алгоритм Шифрования Данных>>.

Практическая значимость результатов определяется тем, что полученные результаты позволяют говорить о неприменимости прямых классических методов криптографического анализа к алгоритму O'zDSt 1105:2009.

Результаты диссертации по связи между классическими методами криптографического анализа могут использоваться для анализа конкретных криптографических систем защиты информации. В рамках диссертации осуществлено применение полученных теоретических результатов - полученные алгоритмы были запрограммированы с использованием технологии параллельных вычислений OpenMPI на кластере из Debian Linux машин.

Внедрение результатов исследований. Результаты диссертации использованы в Научно-Техническом Проекте Специального Назначения шифр <<Параллель>>.

Публикации и апробация работы. Результаты диссертации изложены в одной публикации:

1. Музафаров Х.А., Латыпов А.У, “Применение технологии параллельных вычислений MPI для анализа таблиц замены в криптографическом стандарте Республики Узбекистан O’z DSt 1105:2009 «алгоритм шифрования данных» по критерию строгого лавинного эффекта”, сборник тезисов, посвященный 95 летию НУУз им. М.Улугбека.

Структура и объём работы. Диссертация состоит из введения, глав 1 и 2, заключения и приложений. Глава 1 содержит основные формальные понятия, используемые в работе. В главе 2 содержатся полученные результаты, доказательства утверждений и описание построенного параллельного алогритма. В приложении №1 содержится листинг программы на языке Си++. Приложение №2 содержит описание двух современных методов криптографического анализа, которые были изучены автором дополнительно к классическим методам.

# Глава 1

# Основные понятия

Введем основные понятия и обозначения.

## 1.1 Определения

Основным объектом исследования является таблица замены, которая представляет из себя перестановку натуральных чисел (см. [7], глава V, параграф 3).

**Определение 1.** Таблицей замены *(lookup table) или* перестановкой *(substitution-box), будем называть биективное преобразоваие (bijective transformation)* $F$*,*

$$F\colon {\mathbb{F}_2}^n \to {\mathbb{F}_2}^n$$

*где* $\mathbb{F}_2$ *– конечное поле 2-го порядка (finite field of order 2).*

Можно считать, что $F$ представлена в виде системы булевых функций [1]:

$$F(x_1, x_2, \ldots, x_n) = (f_1, f_2, \ldots, f_n)$$

[1]Напомним, что *булевой функцией*, называется отображение $f\colon \{0,1\}^n \to \{0,1\}^n$.

$$\begin{cases} f_1 = f_1(x_1, x_2, \ldots, x_n) \\ f_2 = f_1(x_1, x_2, \ldots, x_n) \\ \qquad\qquad\qquad\qquad \ldots \\ f_n = f_1(x_1, x_2, \ldots, x_n) \end{cases} \tag{1.1}$$

Поэтому $i$-ю функцию будем обозначать $F^i$. Далее, $i$-ю компоненту вектора $x = (x_1, x_2, \ldots, x_n)$, будем обозначать $x^i$.

Обозначим $e_i = (0, 0, \ldots, 1, 0, \ldots, 0)$ – двоичный вектор, где единица стоит на $i$-ой позиции. Пусть $\Sigma_i = \{(x', x'') \colon x', x'' \in X, x'_i + x''_i = e_i\}$, $i = \overline{1, n}$, а $\Sigma = \bigcup_{i=1}^{n} \Sigma_i$.

Перестановка $F$ удовлетворяет строгому лавнному критерию (СЛК), если при изменении любого из $n$ входных битов, каждый выходной бит меняется с вероятностью $p = 1/2$ [3]. Дадим формальное опередеделение.

**Определение 2.** *Перестановка $F$, удовлетворяет строгому лавнному критерию (СЛК), если*

$$P(v_i^j = 1) = \frac{1}{2}, \forall j = \overline{1, n}$$

Множество функций, для которых выполняется строгий лавинный критерий, обозначим $SAC$.

Функция $f : \{0,1\}^n \to \{0,1\}^n$ удовлетворяет критерию независимости битов [3], если для любых $i, j, k \in \{1, 2, \ldots, n\}$, $j \neq k$, инвертирование бита $i$ на входе вызывает изменения битов $j$ и $k$, причем эти изменения независимы. Для измерения независимости двух выходных бит вводят коэффициент корреляции $\rho(v_i^j, v_i^k)$ между $j$ и $k$ битами лавинного вектора $v_i(x', x'') = F(x') + F(x'')$, $(x', x'') \in \Sigma_i$:

$$\rho(v_i^j, v_i^k) = \frac{cov(v_i^j, v_i^k)}{\sigma(v_i^j)\sigma(v_i^k)}$$

где $cov(\xi, \eta) = \mathbb{M}(\xi\eta) - \mathbb{M}\xi\mathbb{M}\eta$, $\sigma^2 = \mathbb{M}\xi^2 - (\mathbb{M}\xi)^2$, $\xi, \eta$ – случайные величины.

**Определение 3.** *Параметр независимости бит для функции $F$ находится как*

$$\chi = \max_{1 \le j,k \le n, j \ne k} \rho(v_i^j, v_i^k)$$

Этот параметр демонстрирует на сколько функция $F$ удовлетворяет критерию независимости битов. Он принимает значения в промежутке $[0, 1]$, и в наилучшем случае равен 0 и можно говорить о независимости, в наихудшем 1, когда имеется зависимость [3]. Говорят, что перестановка $F$ удовлетворяет критерию независимости битов, если при изменении любого входного бита любые два выходных бита изменяются независимо. Множество функций, для которых выполняется критерий независимости бит, то есть $\chi = 0$, обозначим $BIC$.

**Определение 4.** *Булева функция $f$ принадлежит классу линейных функций, если ее полином жегалкина имеет вид*

$$f(x_1, x_2, \ldots, x_n) = k_0 \oplus k_1 x_1 \oplus k_2 \oplus \ldots k_n x_n$$

*где константы $k_0, k_1, k_2, \ldots, k_n \in \{0, 1\}$. Будем обозначать $f \in LF$.*

**Определение 5.** *Будем говорить, что перестановка $F$ нелинейна, если система функций, образующих $F$, не содержит функий, которые принадлежат классу линейных булевых функций, то есть*

$$F^i \notin LF, \forall i = \overline{1, n}$$

**Определение 6.** *Сбалансированной булевой функцией называется такая булева функция $f$, которая на всей области определения функции принимает значение 0 ровно столько же раз как и значение 1. Будем обозначать $f \in BF$.*

**Определение 7.** *Будем говорить, что перестановка $F$ сбалансирована, если все булевы функции $f_1, f_2, \ldots, f_n$ в системе функций, образующих $F$, сбалансированы, то есть*

$$F^i \in BF, \forall i = \overline{1,n}$$

## 1.2 Алгоритмы криптографического анализа

### 1.2.1 Линейный анализ

Линейный анализ состоит из двух этапов: анализ компонент шифра (в основном S-блоков) и построение атаки. Нас будет интересовать только первый этап, а точнее анализ S-блоков. Коротко опишем общую схему анализа.

Таблица замены (S-блок) - есть некоторая перестановка $F$, которую нам нужно аппроксимировать, уравнениями следующего вида:

$$a_1x_1 \oplus a_2x_2 \oplus \ldots \oplus a_nx_n \oplus b_1y_1 \oplus b_2y_2 \oplus \ldots \oplus b_ny_n = v \qquad (1.2)$$

где $x \in {\mathbb{F}_2}^n, y = f(x), a_i, b_i \in \{0,1\}, i = \overline{1,n}$, значение $v \in \{0,1\}$.

При анализе мы перебираем всевозможные $x \in X$ для каждого набора $\{a_1, \ldots, a_n, b_1, \ldots, b_n\}$ и суммируем значения $v$ (статистику), получающиеся в результате. В случае, если функция $f$ линейно независима со всеми уравнениями (1.2), мы получим суммарное значение равное $\frac{1}{2}|X|$. Для проведения анализа нас интересует отклонение, которое рассчитывается по следующей формуле:

$$\epsilon_\gamma = \frac{\sum_{x \in X} \gamma(x, f(x))}{|X|} - 0.5 \qquad (1.3)$$

где $\gamma(x,y)$ – одно из уравнений (1.2). Чем больше число $|\epsilon_\gamma|$, тем больше вероятность прохождения уравнения $\gamma$.

Остальные детали атаки не имеют большого для нас значения.

Если существует уравнение $\gamma$, в котором как минимум 2 коэффициента $b_i, b_j = 1, i \neq j$, выполняющегося с вероятностью 1, то будем говорить, что перестановка $F$ *уязвима к линейному анализу*. Будем обозначать: $F \in LA$. С подробным описанием линейного анализа можно ознакомиться в оригинальной работе [18].

### 1.2.2 Дифференциальный анализ

Данный метод криптоанализа рассматривает рассеяние информации. Под рассеянием понимают, что распределение $P(a - b = F(a) - F(b))$ должно быть близко к равномерному. Отклонения от равномерного распределения являются уязвимостями операции $F$ относительно $-$.

Исследуют обычно нелинейные блоки шифра, так как линейные характерны как раз сохранением расстояния. Для этого перебирают все возможные дифференциалы, шифруют все возможные тексты с данным дифференциалом и строят распределение выходных дифференциалов.

$$d_{out} = F(x) + F(x + d_{in})$$

Здесь перебор по $x \in X, d_{in} \in D$, где $D = \{x | x = y \oplus z, y, z \in X\}$. В результате выбираются пары $(d_{in}, d_{out})$, обладающие наибольшей вероятностью, на базе которых и составляется атака.

Остальные детали атаки не имеют большого для нас значения.

Если все пары дифференциалов, встречаются с верояностью $\frac{1}{|X|}$ будем говорить, что функция $F$ *устойчива к дифференциальному анализу*. Будем обозначать: $F \in \overline{DA}$.

С подробным описанием дифференциального анализа можно ознакомиться в оригинальной работе [14], [15].

# Глава 2

# Полученные результаты

## 2.1 Теоремы и утверждения

**Утверждение 1.** *Для перестановки $F$, представленной системой функций (2.1), булева функция $f_1$ всегда сбалансирована.*

**Теорема 1.** *Пусть $F$ – перестановка, устойчивая к дифференциальному анализу, тогда для $F$ выполняется строгий лавинный критерий. То есть*

$$\overline{DA} \subseteq SAC$$

Будем говорить, что перестановка тривиальна, если

- существует $F^i \equiv c, c = const, c \in \{0, 1\}$,
- все $F^i(x_1, \ldots, x_n) = a_i \oplus x_j$, причем, все $x_j$ в правых частях встречаются ровно по одному разу, а $a_i, b_i$ – константы (0 или 1).

**Теорема 2.** *Пусть $F$ – не тривиальная перестановка, уязвимая к линейному анализу, тогда для $F$ не выполняется критерий независимости бит.*

## 2.2 Доказательства утверждений

*Доказательство.* (Утверждения 1) Согласно [7], в силу биективности подстановка $F = F(x_1, x_2, \ldots, x_n)$ может быть представлена в так называемом треугольном виде:

$$\begin{cases} f_1 = x_1 \oplus h_1 \\ f_2 = x_2 \oplus h_2(x_1) \\ \quad \ldots \\ f_n = x_n \oplus h_n(x_1, x_2, \ldots, x_{n-1}) \end{cases} \tag{2.1}$$

где $h_1, h_2, \ldots, h_n$ – булевы функции. Следовательно, $f_1$ представляет собой либо функцию $x_1$ либо $x1 \oplus 1$. Обе эти функции сбалансированы. □

*Доказательство.* (Теоремы 1) Если $F \in \overline{DA}$, тогда $\forall(d_{in}, d_{out}) = (x' + x'', F(x') + F(x''))$ верояность $P = \frac{1}{|X|}$. А следовательно, и для $x', x''$ таких, что $x' + x'' = e_i$ мы имеем $P(v(x', x'')^j = 1) = \sum_{d^j_{out}=1} P(d_{out}) = \sum_{d^j_{out}=1} \frac{1}{|X|}$. Так как все $d_{out}$ встречаются с вероятностью $\frac{1}{|X|}$, то бит на $j$-ой позиции будет равен единице ровно в $\frac{|X|}{2}$ раз. Следовательно, искомая вероятность равна $\frac{1}{2}$. □

*Доказательство.* (Теоремы 2) Если $F$ уязвима к линейному анализу, тогда сущетвует уравнение $\gamma(x, F(x))$ такое, что $P(\gamma) = 1$. Откуда, следует, что данная перестановка имеет вид

$$a_0 \oplus a_1x_1 \oplus a_2x_2 \oplus \ldots a_nx_n = b_0 \oplus b_1y_1 \oplus b_2y_2 \oplus \ldots b_ny_n$$

По критерию независимости бит, требуется, чтобы при инвертировании бита $x_i$, биты $y_j, y_k$ менялись независимо друг от друга. Предположим, что это требование выполняется и константы при $y_j, y_k$ не равны нулю. Так как если

константа при $y_j$ или $y_k$ равна нулю, мы имеем фиктивную переменную, и соответственно тривиальную перестановку.

Из условия уязвимости имеем

$$y_j = a_0 \oplus a_1 x_1 \oplus a_2 x_2 \oplus \dots a_n x_n \oplus b_0 \oplus b_1 y_1 \oplus b_2 y_2 \oplus \dots b_n y_n$$

Так как остальные $y_i, i = \overline{1,n}, i \neq j, i \neq k$, являются функциями от $x_1, \dots, x_n$ и меняется здесь только $x_i$, то мы можем записать это выражение так

$$y_j = y_k \oplus g(x_i)$$

Для того, чтобы критерий независимости бит выполнялся, необходимо, чтобы $\rho(y_j, y_k)$ равнялось нулю, то есть

$$\mathbb{M} y_j y_k - \mathbb{M} y_j \mathbb{M} y_k = 0$$

Так как для булевых функций $\mathbb{M}y = 0P(y = 0) + 1P(y = 1)$, то нужно показать, что

$$P(y_j y_k = 1) = P(y_k = 1) P(y_j = 1)$$

Покажем, что это не так. Так как в общем случае

$$P(y_j y_k = 1) = P(y_k = 1) P(y_j = 1 | y_k = 1)$$

, то достаточно показать, что $P(y_j = 1 | y_k = 1) \neq P(y_j = 1)$. Иначе говоря, мы имеем

$$P(1 \oplus g(x_i) = 1) = P(y_j = 1)$$

что эквиваленто $y_j = 1 \oplus g(x_i)$. Отсюда следует, что $y_k \equiv 1$. Чего быть не может, так как $F$ - не тривиальна. $\square$

## 2.3 Вычислительный эксперимент

На базе гранта шифр “Параллель” при поддержке ректора НУУз был собран кластер, состоящий из сервера и 12-ти компьютеров (10 из них были предоставлены руководством университета). Собранный кластер очень хорошо подходит для проведения исследований по криптографии и для оценки качества создаваемых алгоритмов. Так как машины отличаются друг от друга производительностью, то сразу будут видны недостатки параллельного алгоритма.

Поэтому был создан параллельный алгоритм на базе технологии параллельных вычислений OpenMPI, обладающий высокой гранулярностью и масштабируемостью. Алгоритм автоматически, без модификации кода, подстраивается под предоставленные ему вычислительные ресурсы – количество вычислительных процессов. Кроме того, высокая гранулярность, гарантирует, что даже если в кластере все машины обладают различной производительностью, то ни одна не будет простаивать.

Предполагается, что имеется как минимум 2 вычислительных процесса и все они имеют номера (начиная с 0). Один из процессов (с номером 0) будет называться серверным процессом или просто сервером, а остальные вычислительными или рабочими процессами. Алгоритмы работы сервера и рабочих процессов отличаются. Сервер отвечает за то, чтобы исходная задача была разделена на гранулы определенного размера (задается при компиляции). Далее его алгоритм прост:

1. Пока нет запроса от клиентов он ничего не делает
2. Если пришел запрос от клиента с тэгом REQUEST_DATA
   (a) Сгенерировать новую задачу (гранулу)

(b) Передать задачу рабочему процессу

(c) Перейти к шагу 1

У рабочего процесса, алгоритм несколько отличается:

1. Запросить новую задачу (гранулу), то есть отправить серверному процессу свой номер с тэгом REQUEST_DATA

2. Если получена новая задача и размер ее больше нуля

   (a) Для каждого задания из задачи вызвать функцию eval()

3. Иначе, останов

Для того, чтобы не потерять все данные при скачке электричества или его отключении, данные с некоторой периодичностью (задаваемой при компилляции) “сбрасываются” в log-файл с помощью функций из файла “log.h”.
Эксперимент состоял в том, что генерировались все возможные таблицы замены криптографического стандарта Республики Узбекистан O’zDSt 1105:2009 <<Алгоритм Шифрования Данных>> [8]. Здесь таблицы замены являются перестановкой на 256 значений и порождаются в зависимости от битов расширенного ключа $k_{se}$ на параметрах $d$, $L$, $R$ по формуле

$$\left\{[((i+L) \mod 256)+1]^d\right\} \mod 256$$

где возведение в степень $d$ – является возведением в степень с параметром $R$. Всего таблиц замены, порождаемых различными значениями этих параметров – **4 161 600**.
Для каждой из этих таблиц выполнялась функция оценки качества, eval(), которая применяя линейный и дифференциальный метод, автоматически

оценивала возможность прохождения этих атак. Так как мы уже показали, что если таблица подвержена линейному анализу, то для нее не выполнен критерий независимости бит. А если подвержена дифференциальному анализу – не выполняется строгий лавинный эффект.

Полученные результаты, позволяют утверждать, что таблицы замены имеют устойчивость как к линейному так и дифференциальному анализу, при количестве раундов больше 4-х.

# Заключение

В процессе исследований, выполненных в диссертационной работе, получены следующие результаты:

1. доказаны две теоремы, проливающие свет на взаимосвязь между классическими методами анализа и критериями, предъявляемыми к элементарным преобразованиям блочного алгоритма шифрования;

2. построен параллельный алгоритм, автоматически обеспечивающий высокий уровень масштабируемости и равномерную загрузку всех доступных машин;

3. Проведен вычислительный эксперимент:

    - с использованием данного алгоритма изучены таблицы замены криптографического стандарта Республики Узбекистан O'zDSt 1105:2009 <<Алгоритм Шифрования Данных>>.

Практическая значимость результатов определяется тем, что полученные результаты позволяют говорить о неприменимости прямых классических методов криптографического анализа к алгоритму O'zDSt 1105:2009 уже после 4-х раундов.

Также полученные алгоритмы могут быть использованы для исследования другого произвольного класса таблиц замены.

# Приложение №1. Листинг программы

Описание алгоритма содержится в главе 2.3. Ниже представлены файлы программы на языке C++, которые реализуют данные идеи. Так как кластеры, в основном строятся на базе операционный систем Linux, то мы будем использовать инструментарий разработки GNU Make.
Прокомментируем кратко содержание каждого файла.

- “Makefile” – это основной “конфигурационный” файл “проекта”, который использует утилита make для сборки целевого исполняемого файла.

- “type.h” – здесь содержатся введенные нами типы данный, используемые во всем проекте.

- “main.cpp” – файл содержащий главную функцую (точка входа программы). Здесь происходит инициализации и финализация подсистемы MPI. Также здесь определяется количество доступных нам вычислительных процессов и основной алгоритм сервера и рабочих процессов.

- Файлы “log.h” и “log.cpp” – это влючаемый файл и имплементация

(implementation) функций, отвечающих за сброс данных о ходе выполнения программы. Здесь предолагается, что “логи” будут “сбрасываться” в стандартный поток вывода (stdout) процесса с номером 0 (сервера).

- Файлы “eval.h” и “eval.cpp” – это влючаемый файл и имплементация (implementation) функций, отвечающих за обработку каждого задания. Здесь предполагается выполнение линейного и дифференциального анализа (сами функции анализа находятся в файле “analysis.cpp”).

Предполагается, что все необходимое, для работы и сбоки программ для системы OpenMPI уже установлено и проверено.
Для сборки нужно выполнить комманду

```
bash> make parallel
```

для запуска программы на счет достаточно дать комманду

```
bash> mpirun parallel -n N --hostfile hostfile
```

где hostfile – это файл, содержащий информацию о принадлежащих кластеру машин и N – сколько процессов нужно запустить на счет.
Дале приведено содержание всех файлов, кроме “analysis.h” и “analysis.cpp”. Эти фалы содержат секретную информацию, с которой можно ознакомиться в [4].

# Makefile

```
CXX = mpic++
CXXFLAGS = -Wall
CPPFLAGS =
HEADERFILES= log.h analysis.h eval.h type.h
SRCMODULES = log.cpp analysis.cpp eval.cpp
OBJMODULES = $(SRCMODULES:.cpp=.o)

%.o: %.cpp %.h
  $(CXX) $(CXXFLAGS) -c $< -o $@

parallel: main.cpp $(OBJMODULES)
  $(CXX) $(CXXFLAGS) $^ -o $@

clean:
  rm -f *.o

run:
  mpirun --hostfile hosts -n 2 ./parallel
```

# type.h

```
#ifndef TYPE_H_DEFINED
#define TYPE_H_DEFINED

typedef unsigned char byte_t;
```

```
#endif //TYPE_H_DEFINED
```

## main.cpp

```
// Подключение заголовочного файла MPI
#include <mpi.h>
#include <malloc.h>

#include "log.h"
#include "eval.h"

typedef enum {
  REQUEST_DATA,
  REQUEST_ANS
} Tag;

static const int SERVER_PROC = 0;
static const unsigned int n = 256;

int main(int argc, char **argv)
{
  int myid, numprocs;
  double startwtime = 0.0, endwtime;
  int namelen;
  char processor_name[MPI_MAX_PROCESSOR_NAME];
```

```
RequestAnswer ra;
MPI_Status stat;

// Инициализация подсистемы MPI
MPI_Init(&argc, &argv);
// Получить размер коммуникатора MPI_COMM_WORLD
// (общее число процессов в рамках задачи)
MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
// Получить номер текущего процесса в рамках
// коммуникатора MPI_COMM_WORLD
MPI_Comm_rank(MPI_COMM_WORLD, &myid);
MPI_Get_processor_name(processor_name, &namelen);

// Вывод номера потока в общем пуле
fprintf(stdout, "Process␣%d␣of␣%d␣is␣on␣%s\n", \
  myid, numprocs, processor_name);

fflush(stdout);

if(myid==0)//server process
{
  int L=0,R=0,d=3;
  bool finished=false;
  int reqGenerated;

  startwtime = MPI_Wtime();
```

```
printfLog(myid, "Server␣(%d)␣started␣at␣%lf\n",
                        myid, startwtime);

//MPI_Bcast(&count, 1, MPI_INTEGER, 0, MPI_COMM_WORLD);

//loop
do{
  //wait for data request
  int reqId;
  MPI_Recv((void*)&reqId,1,MPI_INTEGER, MPI_ANY_SOURCE,
    (int)REQUEST_DATA, MPI_COMM_WORLD, &stat);

  printfLog(myid, "Receved␣data␣request␣\
from␣%d\n", reqId);

  //generate data
  ra.reset();
  for(reqGenerated=0;
    reqGenerated<MAX_TASKS_PER_REQUEST;
    reqGenerated++)
    {
    ra << Task(L,R,d);
    //next params
    d+=4;
    if( d > 255 ) {
```

```
            R++;
            d=3;
          }
          if( R > 255 ) {
            L++;
            R=0;
          }
          if( L > 255 ){
            finished=true;
          }
        }
        //send it
        if( ra.size()>0 ){
          printfLog(myid, "Sending request ans to \
%d of size %d\n", reqId, ra.size());

          MPI_Send(&ra, sizeof(ra), MPI_BYTE, reqId,
          (int)REQUEST_ANS, MPI_COMM_WORLD);
        }
      }while(!finished);

      //MPI_Reduce(&tmp2, &tmp1, 1, MPI_INTEGER,
                    MPI_MAX, 0, MPI_COMM_WORLD);

      endwtime = MPI_Wtime();
      printf("Time = %f\n", endwtime-startwtime);
```

```
    fflush(stdout);
  }
  else //worker process
  {
    int i;
    double *A = (double*)malloc(n * n * sizeof(double));
    printf("&A=%X\n", (unsigned int)A);
    do{
      //request new data to process
      printfLog(myid, "Sending␣request␣to␣\
%d\n", SERVER_PROC);

      MPI_Send(&myid, 1, MPI_INTEGER,
          SERVER_PROC,
        (int)REQUEST_DATA, MPI_COMM_WORLD);

      //get request answer
      MPI_Recv((void*)&ra,sizeof(ra),MPI_BYTE,
          SERVER_PROC,
        (int)REQUEST_ANS, MPI_COMM_WORLD, &stat);

      printfLog(myid, "Receved␣request␣ans␣from␣\
%d␣of␣size␣%d\n", SERVER_PROC, ra.size());
      if( !ra.size() ) //request answer is empty!
        break;
```

```
        printfLog(myid, "Evaluating..␣\
ra.size()=%d\n", ra.size());

        //evaluate
        for(i=0; i<ra.size(); i++){
          evalTask(myid, ra[i], n, A);
        }

      }while(true);
    }

    MPI_Finalize();
    return 0;
}
```

## log.h

```
#ifndef LOG_H_DEFINED
#define LOG_H_DEFINED

#include <stdarg.h>

void printfLog(int myid, const char *format, ...);
void printfVector(int myid, double *a, int n);
void printfMatrix(int myid, double *A, int n);
void printfError(int myid, const char *format, ...);
```

```
#endif //LOG_H_DEFINED
```

## log.cpp

```
#include "log.h"

#include <string.h>
#include <stdio.h>

void printfLog(int myid, const char *format, ...)
{
    va_list argptr;
    va_start(argptr, format);
    fprintf(stdout, "Process␣%d:␣", myid);
    vfprintf(stdout, format, argptr);
}

void printfVector(int myid, double *a, int n)
{
  fprintf(stdout, "Process␣%d:␣", myid);
   for (int j = 0; j <= n; j++)
     fprintf(stdout, "%.2f␣", a[j]);
  fprintf(stdout, "\n");
}

void printfMatrix(int myid, double *A, int n)
{
```

```
  fprintf(stdout, "Process␣%d:␣", myid);
  for (int i = 0; i <= n; i++){
    for (int j = 0; j <= n; j++)
      fprintf(stdout, "%.2f␣", A[i*n+j]);
    fprintf(stdout, "\n");
  }
}

void printfError(int myid, const char *format, ...)
{
    va_list argptr;
    va_start(argptr, format);
    fprintf(stdout, "Process␣%d␣\
encountered␣a␣problem:␣", myid);
    vfprintf(stdout, format, argptr);
}
```

## eval.h

```
#ifndef EVAL_H_DEFINED
#define EVAL_H_DEFINED

struct Task
{
  int L, R, d;
  Task(int _L, int _R, int _d)
  : L(_L), R(_R), d(_d)
```

```
  {}
  Task() {}
};

///

#define MAX_TASKS_PER_REQUEST 5

class RequestAnswer
{
  int reqSize;
  Task data[MAX_TASKS_PER_REQUEST];
public:
  void operator<<(Task t){data[reqSize++]=t;}
  Task &operator[](int i){return data[i];}
  int size() {return reqSize;}
  void reset() {reqSize=0;}
  RequestAnswer() : reqSize(0) {}
};

///

int evalTask(int myid, Task t, int n, double *A);

#endif //EVAL_H_DEFINED
```

## eval.cpp

```
#include "eval.h"
#include "analysis.h"
#include "log.h"

byte_t s_b[256];

double r=0,
  lin_min=1.0e+35, dif_min=1.0e+35,
  lin_max=1000000, dif_max=1000000;

int
  d_lin_max=-1,
  L_lin_max=-1,
  R_lin_max=-1,
  d_dif_max=1000,
  L_dif_max=1000,
  R_dif_max=1000;

//eval() should request for new data after %d steps
#define REQUEST_NEW_DATA_AFTER 5

//eval() should flush log to screen after %d solved tasks
#define FLUSH_LOG_AFTER 2
```

```
static unsigned long long step = 0;

int evalTask(int myid, Task t, int n, double *A)
{
  int i;
  int L=t.L, R=t.R, d=t.d;
  printfLog(myid, "L = %d, R = %d, d = %d\n", L, R, d);

  for (i = 0; i < 256; i++)
    s_b[i] = power((i + L) % 256 + 1, d, R);

  printfLog(myid, "Analysing..\n", r);

  r = diff_analysis(s_b, n, A);
  printfLog(myid, "Dif T = %lf\n", r);
  if (r < dif_min)
    dif_min = r;

  //Linear analysis
  r = linear_analysis(s_b, n, A);
  printfLog(myid, "Lin T = %lf\n", r);
  if (r < lin_min)
    lin_min = r;

  if ( dif_min < dif_max ){
    dif_max = dif_min;
```

```
      L_dif_max = L;
      d_dif_max = d;
      R_dif_max = R;
    }
    if ( lin_min < lin_max ){
      lin_max = lin_min;
      L_lin_max = L;
      d_lin_max = d;
      R_lin_max = R;
    }

    if( step % FLUSH_LOG_AFTER == 0 ){
      printfLog(myid, "L = %d\n", L);
      printfLog(myid, "lin_min in T = %lf; \
dif_min if T = %lf\n",
        lin_min, dif_min);
      printfLog(myid, "MaxMin Lin = %lf \
(d = %d, L = %d, R = %d)\n",
        lin_max, d_lin_max, L_lin_max, R_lin_max);
      printfLog(myid, "MaxMin Dif = %lf \
(d = %d, L = %d, R = %d)\n",
        dif_max, d_dif_max, L_dif_max, R_dif_max);
    }
    step++;

    return 0;
```

```
}
```

# Приложение №2. Методы криптографического анализа блочных алгоритмов шифрования

# Атака отражением (Reflection Attack)

Атака отражением использует высокий уровень самоподобия алгоритма шифрования. Мы конструируем новую функцию, которая совпадает с исходной функцией шифрования, на большом подмножестве входных текстов. Будем использовать отклонения в распределении вероятностей неподвижных точек одной из выбранных промежуточных функций, расширяя ее свойства с использованием конкретных инволюций.

**Лемма 1.** *Пусть* $i, j$ *таковы, что* $i < j$, $i + j \leq r$. *Предположим, что* $F_{i-t} = F_{j+t}^{-1}$, *для всех* $t = \overline{1, i}$. *Тогда верны следующие утверждения:*

1. *Если* $y = F[i-t, i-1](x)$ *и* $y \in U(F[i, j])$, *тогда*

$$x \in U(F_K[i-t, j+t]), \forall t = \overline{2, i-1}$$

2. *Если* $x \in U(F[i-t, j+t])$, *для некоторого* $t, 1 < t < i$, *то* $y = F[i-t, i-1](x)$, $y \in U(F[i, j])$.

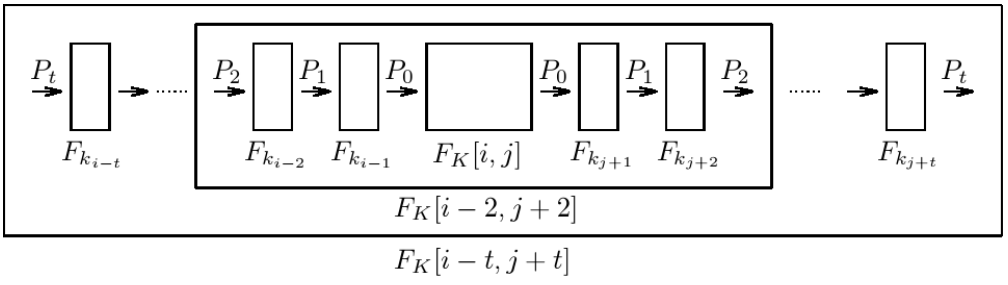


Рис. 1: Свойсво отображений (см. лемму 1)

**Следствие 2.1** (1)**.** *При выполнении условий леммы 1, функция шифрования* $E_K$ *эквивалентна функции* $F[i+j, r]$ *на множестве* $F^{-1}[1, i-1]\left(U\left(F[i, j]\right)\right)$

То есть, существует другая функция которая совпадает с исходной функцией на специальном подмножестве данных. Эта функция, обладает меньшим количеством раундов, поэтому "вероятно"она будет слабее. Тогда, криптоаналитик може восстановить ключи $k_{i+j}, \ldots, k_r$ решая систему уравнений вида

$$F[i+j, r](x) = E_K(x) = y \tag{2}$$

Таким обрзом определены следующие три параметра, которые задают сложность атаки:

1. $m$ – количество пар $(x, y)$, необходимых для решения уравнения (2). Под решением мы подразумеваем нахождение единственного решения, если все уравнения верны и отсутствие решения в противном случае.

2. $|U(F_K[i, j])|$ – размерность пространства неподвижных точек функции $F_K[i, j]$

3. $P(F[1, i-1](x) \in U(F[i, j])|E_K(x))$ – вероятность того, что $F_K[1, i-1](x)$ находится в $U(F_K[i, j]$, для заданного $E_K(x)$.

Вероятность того, что $F[1, i-1](x)$ находится в $U(F[i, j]$, для заданного $E_K(x)$, равняется $\frac{|U(F[i,j])|}{2^n}$ для случайно выбранного $x$. Однако, для заданных конкретных значений $E_K(x)$, эта вероятность может значительно отличаться как в большую так и меньшую сторону в зависимости от структуры шифра.

**Теорема 3.** *Пусть $m$ уравнений необходимо для решения уравнений (2) и $T$ – это неоходимое время (то есть количество операций шифрования).*

1. *Тогда атака для раскрытия ключей $k_{i+j}, \ldots, k_r$ – имеет временную сложность $\frac{m \cdot 2^n}{|U(F[i,j])|}$.*

2. *Пусть вероятности* $P(F[1,i-1](x) \in U(F[i,j])|E_K(x))$ *уже подсчитаны, и выбраны* $l$ *– наиболее вероятных из* $\frac{m \cdot 2^n}{|U(F[i,j])|}$*, такие что*

$$\sum_{s=1}^{l} P(F[1,i-1](x) \in U(F[i,j])|E_K(x)) \approx m$$

*Тогда временная сложность атаки ограничена сверху числом* $C_l^m \cdot T$.

# Атака отражением на Сеть Фейстеля

Пусть откртые тексты $x$ из ${\mathbb{F}_2}^n$, заданы в виде пары $x = (x_0, x_1), x_0, x_1 \in {\mathbb{F}_2}^{n/2}$. Тогда сеть фестеля может быть представлена как рекурсиваня функция

$$x_i = R_{k_{i-1}}(x_{i-1}) \oplus x_{i-2}$$

где функция $R\colon {\mathbb{F}_2}^{n/2} \to {\mathbb{F}_2}^{n/2}$ – функция шифрования, а $\oplus$ – побитовое сложение по модулю 2. Тогда $i$-ый раунд может быть представлен как

$$(x_i, x_{i+1}) = F_{k_i}(x_{i-1}, x_i) = (x_i, R_{k_i}(x_i) \oplus x_{i-1}), i < r$$

Где выходным шифрованным текстом является $(x_{r+1}, x_r)$.

**Утверждение 2.** *Для заданного натруального числа* $n < r$*, предполагая, что* $k_{m-i} = k_{m+i}, \forall i, 1 \leq i \leq \min r - m, m - 1$*. Пусть* $x = (x_0, x_1)$ *– исходное сообщение и* $x_0, x_1, \ldots, x_r, x_{r+1}$ *– поток шифрования. Тогда если* $R_{k_m}(x_m) = 0$*, то* $x_{m-i} = x_{m+i}, \forall i, 1 \leq i \leq minr - m, m - 1$*. И наоборот, если* $x_{m-i} = x_{m+i}$ *и* $x_{m-i+1} = x_{m+i-1}$ *для некоторого* $i$*, то* $R_{k_m}(x_m) = 0$.

Предложение 2 было известно давно при изучении цикловых структур DES. Поэтому, идея использования неподвижных точек в ключах DES хорошо

исследована. Однако, эти исследования были сфокусированы на алгебраических свойствах перестановок в DES и их коротких циклах, а не на развитии методов восстановления ключа. Следующее следствие поможет выявить обратную стороно давно известного феномена.

**Следствие 3.1** (2)**.** *Предположим, что каждый раундовый ключ* $k_i$ *задает ранудовую функцию* $R_i$*. Пусть* $x = (x_0, x_1)$ *– шифруемое сообщение и* $x_0, x_1, \ldots, x_r, x_{r+1}$ *– поток шифрования. Предположим, что количество ранудов* $r$ *– четное число, то есть* $r = 2r'$*, и* $k_{r'-i} = k_{r'+i}, \forall i, 1 \leq i < r'$*. Тогда* $P(x_0 = x_r) = 2^{-\frac{n}{2}} - 2^{-n}$ *и* $P(R_{r'}(x_{r'}) = 0 | x_0 = x_r) = \frac{1}{2-2^{-\frac{n}{2}}}$.

**Теорема 4.** *Условия такие же как в следствии 3.1. Тогда из* $x_0 = x_r$ *следует, что уравнение*

$$x_1 = R_r(x_r) \oplus x_{r+1}$$

*выполняется с вероятностью* $\frac{1}{2-2^{-\frac{n}{2}}}$.

Заметим, что параметры в теореме 4 все известны, кроме ключа последнего раунда. Пусть $(x_0, x_1)$ – открытый текст, а $(x_{r+1}, x_r)$ – соответствующий шифро-текст. Тогда по теореме 4, мы имеем следующую атаку: будем шифровать тексты, и выбирать из них те, для которых $x_0 = x_r$. Если раундовые ключи удовлетворяют условию $k_{\frac{r}{2}-i} = k_{\frac{r}{2}+i}$, тогда соответствующие уравнения, $x_1 = R_r(x_r) \oplus x_{r+1}$ будут выполняться с вероятностью равной приблизительно половине выбранных текстов. Вероятней всего, эти уравнения легко разрешимы. Решая их мы восстановим последний раундовый ключ. Можно атаку применять несколько раз подбирая различные параметры или используя сведения о процедуре распределения ключей по раундам.

# Атка скольжением (Slide Attack)

На рис. 2 изображен процес зашифрования $n$-битного открытого текста $x_0$; пусть $x_i = F_i(x_{i-1}, k_i)$.

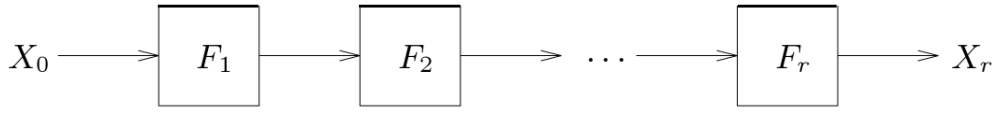


Рис. 2: Типичный алгоритм блочного шифрования

Для простоты будем считать, что все $F_i$ одинаковы, то есть все раундовые ключи совпадают и обозначим это преобрзование просто $F$.

Пусть $F$ очень слабо против атаки с двумя известными парами текст – шифр-текст. То есть, мы будем называть $F$ **слабой** перестановкой, если имея два уравнения $F(x_1, k) = y_1$ и $F(x_2, k) = y_2$ – "просто" извлеч ключ $k$. Это не формальное определение, так как степень "простоты" может меняться от шифра к шифру. В качестве $F$ не обязательно брать функцию шифрования на 1 раунд, можно взять 1.5 рануда или более.

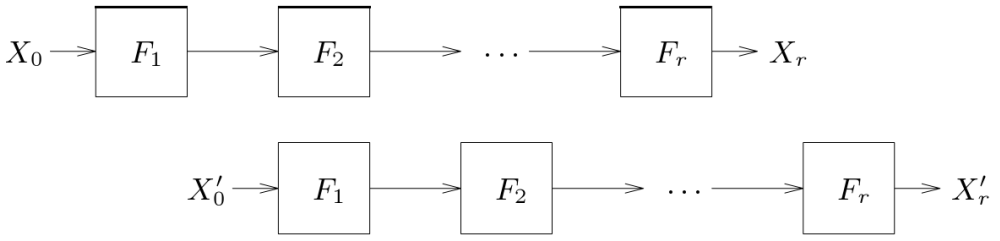


Рис. 3: Эскиз атаки скольжением

На рис. 3 изображена идея атаки скольжением против такого шифра. Идея состоит в том, чтобы "проскользить"одной копией процесса шифрования относительно другой копии, так чтобы эти два процесса отличались на фазу в один раунд. Пусть $x_0$ и $x'_0$ обозначают пару открытых текстов, пусть также $x_i = F_i(x_{i-1})$ и $x'_i = F_i(x'_{i-1})$.

В этих обозначениях, мы сопоставим $x_1$ с $x'_0$ и $x_{i+1}$ с $x'_i$.

Далее, мы предположим, что $F_i = F_{i+1}$ для всех $i \geq 1$. Основное наблюдение в том, что если $x_1 = x'_0$, то у нас также $x_r = x'_{r-1}$. Поэтому мы будем рассматривать пары $(P, C)$, $(P', C')$ такие, что $F(P) = P'$ и $F(C) = C'$. Будем их называть **slid–парами**.

Имея это наблюдение можно провести атаку следующим образом. Мы получаем $2^{n/2}$ известных пар текст–шифртекст $(P_i, C_i)$ и ищем среди них slid-пары. Из парадокса дней рождений следует, что мы (вероятно) получим хотябы одну пару с индексами $i, i'$ такую, что $F(P_i) = P_{i'}$, что и дает нам, искомую пару.

Так используя всего одну slid-пару мы можем восстановить около $n$ бит ключа.

# Атка скольжением на Сеть Фейстеля

Здесь, мы покажем как slide-атака может быть оптимизирована для применения к шифрам на базе сети Фейстеля.

**Атака с известными текстами.** В случае сетей Фейстеля, ранудовая функция $F((l, r)) = (r \oplus f(l), l)$ модифицирует только половину своего входа. Поэтому, условие $F(x) = x'$ может быть распознано простым сравнением левых частей $x$ с павыми частями $x'$, и фильтрующее условие исключит все кроме $2^{-n/2}$ непрвильных пар.

Это наблюдение позволяет нам уменшить временную сложность атаки с известными текстами – $2^{n/2}$ известных текстов и $2^{n/2}$ предварительных вычислений. Поэтому потенциальные slid-пары могут быть идентифицированы с

использованием таблицы поиска (или сортированного списка) с $2^{n/2}$ элементами: мы отсортируем пары известных текстов $(P_i, C_i)$ на основании левых пловин $P_i$ и $C_i$, и для каждого $j$ мы будем искать совпадение правой части $P'_j$ и $C'_j$.

Используя эту технику мы ожидаем найти одну slid-пару при всего лишь однократном возможном возникновении ошибки; ошибочные совпадения могуть быть легко устранены во второй фазе. Искомые slid-пары дают нам около $n$ битов информации о ключе; если это не даст всего ключа, мы можем поискать еще slid-пар, либо воспользоваться перебором.

**Атака с выбранными текстами.** Для сети Фейстеля, сложность по данным может быть уменьшена до $2^{n/4}$ пар текстов, при условии, что аналитик может выбирать тексты. Идея в том, чтобы использовать специально выбранные структуры (эта техника была верпвые исользована Бихамом в его работе о related-key криптоанализе). Фиксируем произвольное $n/2$-битное значение $x$. Мы будем выбирать множество $2^{n/4}$ открытых текстов $P_i = (x_i, y_i)$ варьируя над множеством из $2^{n/4}$ значений для $y_i$. Затем мы возьмем второй набор из $2^{n/4}$ текстов в виде $P'_j = (y'_j, x)$ варьируя над другим множеством из $2^{n/4}$ значений для $y'_j$. Это позволяет получить $2^n$ пар текстов, где правая часть будет появляться с вероятностью $2^{-n/2}$ (точнее, когда $f(x) = y_j \oplus y'_j$). Поэтому мы ожидаем нати хотя бы одну slid-пару. Эти slid-пары могут быть распознаны с использованием $n/2$-битного фильтрующего условия над шифротекстами, и затем мы можем использовать ее для вскрытия $n$ битов ключевого материала.

**Атака с вероятными текстами.** Когда открытые тексты имеют некоторую избыточность, количество требуемых открытых текстов может быть

значительно уменьшено. Эта техника является продолжением идеи статьи Бирюкова и Кушилевитза на использование такой избыточности текстов в дифференциальном анализе. Рассмотрим сначала очень простую модель: источник открытых текстов вырабатывает блоки, в которых 4 самых старших разряда в каждом байте всегда равны нулю. Тогда результирующий открытый текст имеет энтропию равную $n/2$. В этом случае, можно применить атаку с приблизительно $2^{3n/8}$ шифротекстами. Что находится между атакой с выбранными текстами и атакой с известными текстами. Наблюдение состоит в том, что для каждого фиксированного значения $x$, которое может быть левой половиной открытого текста, мы ожидаем получить приблизительно $2^{3n/8-n/4} = 2^{n/8}$ открытых текстов в виде $P_i = (x, y_i)$, вместе с другими $2^{n/8}$ открытыми текстами вида $P'_j = (y'_j, x)$. Каждый $x$ дает около $2^{n/4}$ пар текстов. Причем $x$ может принимать $2^{n/4}$ значений. Предположив, что $f$ действует как случаяная перестановка, каждая такая пара дает $2^{-n/2}$-ный шанс для формирования slid-пары, поэтому в сумме мы ожидаем получить одну slid-пару среди всех $2^{3n/8}$ шифротекстов. Эта атака может быть преобразована к атаке на основе только шифротекстов с сравнительно небольшим увеличением сложности. Предположим, что условие $f(u) = v, f(u') = v'$ единственным образом определяет ключ, и восстановление ключа по $u, u', v, v'$ потребует $O(1)$ времени. Тогда мы можем найти ключ с использованием $2^{3n/8+1}$ шифротекстов и $O(2^{n/2})$ действий. Сначала, мы отметим, что $n/2$-битное фильтрующее условие над шифротекстами дает нам множество из $2^{n/4+2}$ потенциальных slid-пар, из которых около 4-х будут правильными (остальные – это ошибочные). Список потенциальных slid-пар может быть определен с помощью $O(2^{3n/8})$ шагов с помощью хеширования или сортировки. Затем, мы делаем предположение о корректности slid-пары $C_i, C'_j$.

В третьих, для каждой из оставшихся slid-пар мы высчитываем значение ключа с использованием уравнений $F(C_i) = C'_i, F(C'_j) = C'_j$, и сохраняем эти $n$-битные значения в таблице. Мы ищем коллизии в этой таблице. Если наше предположение о $C_i, C'_j$ дало нам правильную slid-пару, правильный ключ будет выбран 3 раза. С друго стороны, по порадоксу дней рождений, неправильные slid-пары будут выбраны с очень маленькой вероятностью. Это прводит к атаке, которая требует $O(2^{n/2})$ действий ($2^{n/4}$ предположений о $C_i, C'_j$, выполнение $O(2^{n/4})$ действий для каждого предположения) и требует $2^{n/4+2}$ памяти и около $2^{3n/8+1}$ шифротекстов.

Конечно, это всего лишь пример. В общем, мы ожидаем, что количество текстов требуемых для нахождения первой slid-пары будет приблизительно

$$2^{n/4} \left( \sum_x P(r = x) \cdot P(l = x) \right)^{-1/2}$$

Кроме того, конкретные детали проведения атаки сильно зависят от распределения открытых текстов.

# Литература


[1] Horst Feistel, “Cryptography and Computer Privacy.” Scientific American, Vol. 228, No. 5, 1973.

[2] William Stallings, “Cryptography and network security: principles and practice”, Prentice Hall, 1999, стр. 80.

[3] Isl VERGILI, Melek D. YUCEL VOL.9, “Avalanche and Bit Independence Properties for the Ensembles of Randomly Chosen n X n S-Boxes”, Turk J Elec Engin, 2001, p. 137.

[4] Музафаров Х.А., Жураев Г.У., Икрамов А.А., Латыпов А.У., “Отчет по проекту шифр <<Параллель>> за 2012 год”, Ташкент, 2012 г.

[5] Колмогоров А. Н., Фомин С. В. “Элементы теории функций и функционального анализа”, ФМЛ, 7е изд., 2004. (стр. 83-86)

[6] Яблонский С. В. “Введение в дискретную математику”. Высшая школа, Москва, 2002.

[7] Носов В. А., “Основы дискретной математики”. Издание второе, переработанное и дополненное, Москва 1997

[8] Алгоритм шифрования информации Oz’STD 1105:2009, *Государствен-*

*ный стандарт Узбекистана* (официальное издание), утвержден и введен в действие постановлением Узбекского агентства стандартизации, метрологии и сертификации (агентство <<Узстандарт>>) от 28.09.2009 №05-163.

[9] C. E. Shannon, “Communication Theory of Secrecy Systems”, Vol 28, Oct 1949. — С. 656-715.

[10] Ишматова Ю.А., *Групповые свойства алгебры с параметром*, Конференция <<Ломоносов-2010>>, филиал МГУ им. Ломоносова в Ташкенте, 2010 г.

[11] *Алгоритм криптографического преобразования ГОСТ 28147-89*, ИПК Издательство Стандартов, Москва, 1996 г.

[12] Кемени Дж. Дж., Снелл Дж. Л. *Конечные цепи Маркова.* — М.: Наука. 1970. — 272 с.

[13] M. Matsui, “Linear Cryptanalysis Method for DES Cipher”, Advances in Cryptology - EUROCRYPT ’93 (Lecture Notes in Computer Science no. 765), Springer-Verlag, pp. 386-397, 1994.

[14] E. Biham and A. Shamir, “Differential Cryptanalysis of DES-like Cryptosystems”, Journal of Cryptology, vol. 4, no. 1, pp. 3-72, 1991.

[15] E. Biham and A. Shamir, “Differential Cryptanalysis of the Data Encryption Standard”, Springer-Verlag, 1993.

[16] H.M. Heys and S.E. Tavares, “Substitution-Permutation Networks Resistant to Differential and Linear Cryptanalysis”, Journal of Cryptology, vol. 9, no.1, pp. 1-19, 1996.

[17] National Bureau of Standards, “Data Encryption Standard”, Federal Information Processing Standard 46, 1977.

[18] M. Matsui, “The First Experimental Cryptanalysis of the Data Encryption Standard”, Advances in Cryptology - CRYPTO ’94 (Lecture Notes in Computer Science no. 839), Springer-Verlag, pp. 1-11, 1994.

[19] National Institute of Standards, Advanced Encryption Standard (AES) web site: www.nist.gov/aes.

[20] J. Daemen and V. Rjimen, “AES Proposal: Rijndael”, First Advanced Encryption Standard (AES) Conference, California, Aug. 1998. (See also [6].)

[21] L. Keliher, “Linear and Differential Cryptanalysis of SPNs”, unpublished.

[22] L. Knudsen, “Block Ciphers: A Survey”, State of the Art in Applied Cryptography: Course on Computer Security and Industrial Cryptography (Lecture Notes in Computer Science no. 1528), Springer-Verlag, pp. 18-48, 1998.

[23] D.R. Stinson, Cryptography: Theory and Practice, CRC Press, 1995.

[24] B. Schneier, Applied Cryptography: Protocols, Algorithms, and Source Code in C, 2nd ed., John Wiley & Sons, 1995.

[25] A. J. Menezes, P.C. van Oorschot, and S.A. Vanstone, Handbook of Applied Cryptography, CRC Press, 1997.

[26] W. Stallings, Cryptography and Network Security: Principles and Practices, 2nd ed., Prentice Hall, 1999.

[27] H. Feistel, “Cryptography and Computer Privacy”, Scientific American, vol. 228, no. 5, pp. 15-23, 1973.

[28] K. Nyberg, “Linear Approximations of Block Ciphers”, Advances in Cryptology - EUROCRYPT ’94 (Lecture Notes in Computer Science no. 950), Springer-Verlag, pp. 439-444, 1995.

[29] X. Lai, J.L. Massey, and S. Murphy, “Markov Ciphers and Differential Cryptanalysis”, Advances in Cryptology - EUROCRYPT ’91 (Lecture Notes in Computer Science no. 547), Springer-Verlag, pp. 17-38, 1991.

[30] E. Biham, “On Matsui’s Linear Cryptanalysis”, Advances in Cryptology - EUROCRYPT ’94 (Lecture Notes in Computer Science no. 950), Springer-Verlag, pp. 341-355, 1995.

[31] F. Chabaud and S. Vaudenay, “Links Between Differential and Linear Cryptanalysis”, Advances in Cryptology - EUROCRYPT ’94 (Lecture Notes in Computer Science no. 950), Springer-Verlag, pp. 356-365, 1995.

[32] M. Hellman and S. Langford, “Differential-Linear Cryptanalysis”, Advances in Cryptology - CRYPTO ’94 (Lecture Notes in Computer Science no. 839), Springer-Verlag, pp. 26-39, 1994.

[33] L.R. Knudsen, “Truncated and Higher Order Differentials”, Fast Software Encryption (Lecture Notes in Computer Science no. 1008), Springer-Verlag, pp. 196-211, 1995.

[34] E. Biham, A. Biryukov, and A. Shamir, “Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials”, Advances in Cryptology -

EUROCRYPT ’99 (Lecture Notes in Computer Science no. 1592), Springer - Verlag, pp. 55-64, 1996.

[35] M.J.B. Robshaw and B.S. Kaliski, “Linear Cryptanalysis Using Multiple Approximations”, Advances in Cryptology - CRYPTO ’94 (Lecture Notes in Computer Science no. 839), Springer-Verlag, pp. 1-11, 1994.

[36] L. Knudsen and M.J.B. Robshaw, “Nonlinear Approximations in Linear Cryptanalysis”, Advances in Cryptology - EUROCRYPT ’96 (Lecture Notes in Computer Science no. 1070), Springer-Verlag, pp. 224-236, 1996.

[37] K. Nyberg, “Differentially Uniform Mappings for Cryptography”, Advances in Cryptology - EUROCRYPT ’93 (Lecture Notes in Computer Science no. 765), Springer-Verlag, pp. 55-64, 1994.

[38] E. De Win, A. Bosselaers, B. Preneel, J. Daemen, and V. Rijmen, “The Cipher SHARK”, Fast Software Encryption (Lecture Notes in Computer Science no. 1039), Springer-Verlag, pp. 99-112, 1996.

[39] A.M. Youssef, S. Mister, and S.E. Tavares, “On the Design of Linear Transformations for Substitution Permutation Encryption Networks”, Workshop on Selected Areas of Cryptography (SAC ’96): Workshop Record, pp. 40-48, 1997.

[40] M. Matsui, “On Correlation Between the Order of S-boxes and the Strength of DES”, Advances in Cryptology - EUROCRYPT ’94 (Lecture Notes in Computer Science no. 950), Springer-Verlag, pp. 366-375, 1995.

[41] Шнайер Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си, 2-е издание — М.: Триумф, 2002, 14.1

[42] Грегори Р. Эндрюс. *Основы многопоточного, параллельного и распределенного программирования*. <<Вильямс>>, 2003, 512 стр.

[43] Камерон Хьюз, Трейси Хьюз. *Параллельное и распределенное программирование с использованием C++*. – <<Вильямс>>, 2004, 672 стр.

[44] В. Д. Корнеев. *Параллельное программирование в MPI*. <<Регулярная и хаотическая динамика>>, 2003

[45] Ольга Стесик, Сергей Немнюгин. *Параллельное программирование для многопроцессорных вычислительных систем*. <<BHV-СПб>>, 2002, 400 стр.

[46] Intel Corporation. Developing Multithreaded Applications: A Platform Consistent Approach.

[47] Rohit Chandra, Leonardo Dagum, Dave Kohr, Dror Maydan, Jeff McDonald, Ramesh Menon. *Parallel Programming in OpenMP*. <<Morgan Kaufmann>>, 2001

[48] Корнеев В. В. *Параллельные вычислительные системы*. – М.: <<Нолидж>>, 1999, 320 с.

[49] Богачев К. Ю. *Основы параллельного программирования*. <<Бином. Лаборатория знаний>>, 2003, 342 стр.

[50] Воеводин В. В, Воеводин Вл. В. *Параллельные вычисления*. – <<БХВ-Петербург>>, 2002, 609 с.

[51] Prabhu, C.S.R., *Grid And Cluster Computing*, <<Phi Learning>>, 2009, 256 p.