

Исследование поточного алгоритма шифрования типа «Раскольников» Вариант №4.

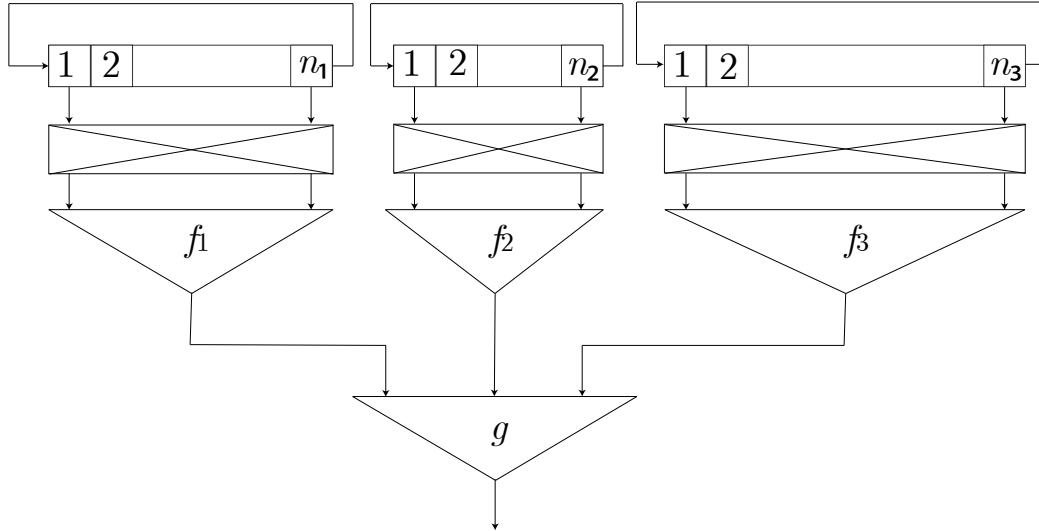
Роман Астраханцев, СКБ-171

25 мая 2022 г.

Содержание

1	Исследование ключевого множества	3
2	Исследование узлов алгоритма	3
2.1	Узлы РСЛОС	3
2.2	Функции усложнения	5
2.3	Комбинирующая функция	6
2.3.1	Вычисление многочлена Жегалкина	6
2.3.2	Характеристики	7
2.3.3	Наилучшее приближение линейной функцией	8
2.3.4	Строгий лавинный критерий	9
3	Методы восстановления ключа	9
3.1	Метод тотального опробования	9
3.2	Метод встречи по середине	10
3.3	Статистический метод	12
3.4	Метод частичного опробования с использованием аннигилятора	13
4	Выводы и рекомендации	15

Описание алгоритма



Поточный алгоритм шифрования типа «Раскольников» состоит из:

1. трех РСЛОС $L_i, i \in \overline{1,3}$, над полем \mathbb{F}_2 с характеристическими многочленами $F_1(x), F_2(x), F_3(x)$ степени соответственно n_1, n_2, n_3 .
2. трех коммутаторов (перестановок) $K_i, i \in \overline{1,3}, K_i \in S(l_i)$, на вход коммутатору K_i подаются значения РСЛОС L_i с индексами

$$1 = p_1 < p_2 < \dots < p_{l_i-1} < p_{l_i} = n_i$$

3. трех функций усложнения выхода РСЛОС $f_i(x_1, x_2, \dots, x_{l_i}), i \in \overline{1,3}$,
4. комбинирующей булевой функции $g(x, y, z)$, задаваемой формулой (\wedge – «и», \neg – отрицание, \vee – «или», \vee – «исключающее или»):

$$g(x, y, z) = (\neg x \vee \neg y) \wedge (x \vee y \vee \neg z) \wedge (\neg y \vee z)$$

Ключом являются:

- начальное заполнение регистра L_1
- коммутатор K_2
- начальное заполнение регистра L_3

1 Исследование ключевого множества

Множество ключей K состоит из всевозможных троек вида (k_1, k_2, k_3) , где

- k_1 - какое-то заполнение регистра L_1 длины n_1 ,
- k_2 - какая-то перестановка длины n_2 ,
- k_3 - какое-то заполнение регистра L_3 длины n_1 .

Тогда общее число ключей будет равно

$$|K| = 2^{n_1} \cdot n_2! \cdot 2^{n_3}$$

Приведём пример параметров n_1, n_2, n_3 , при котором $|K| > 2^{64}$. Пусть $n_1 = 25, n_2 = 10, n_3 = 25$, тогда

$$|K| = 2^{25} \cdot 10! \cdot 2^{25} > 2^{25} \cdot 2^{21} \cdot 2^{25} > 2^{71} > 2^{64}$$

2 Исследование узлов алгоритма

2.1 Узлы РСЛОС

Регистр сдвига с линейной обратной связью или *РСЛОС* — это блок, который генерирует двоичные псевдослучайные периодические последовательности, которые называются линейными рекуррентными последовательностями (ЛРП).

Широкое распространение в криптографических приложениях линейных регистров сдвига над конечными полями \mathbb{F}_{2^n} и кольцами вычетов обусловлено целым рядом факторов. Среди них можно отметить:

- использование только простейших операций сложения и умножения, аппаратно реализованных практически на всех вычислительных средствах;
- высокое быстродействие создаваемых на их основе криптографических алгоритмов;
- большое количество теоретических исследований свойств линейных рекуррентных последовательностей (ЛРП), свидетельствующих об их удовлетворительных криптографических свойствах

В данном шифре в ключ входят только заполнения регистра сдвига, а вид характеристического многочлена, является параметром построения шифра. Значит, нужно подобрать такие функции обратной связи, чтобы для любого **ненулевого** заполнения ЛРП была максимального периода. В этом нам помогут следующие теоремы и следствие из них.

Теорема 2.1. Пусть u — ЛРП над полем \mathbb{F}_q с реверсивным минимальным многочленом $F(x)$ степени m и $q^m > 2$. Тогда следующие утверждения эквивалентны:

- (1) u — ЛРП максимального периода;

(2) многочлен $F(x)$ неприводим над \mathbb{F}_q , и его корень α в минимальном поле разложения \mathbb{F}_{q^m} над \mathbb{F}_q есть примитивный элемент поля.

Теорема 2.2. *Неприводимый многочлен $F(x)$ примитивен в том и только в том случае, когда для любого простого числа p , делящего $q^m - 1$, многочлен $x^{\frac{q^m-1}{p}}$ не сравним с 1 по модулю многочлена $F(x)$.*

Следствие 2.3. *Если $F(x)$ – неприводимый многочлен над полем \mathbb{F}_2 степени m , и $2^m - 1$ – простое число, то $F(x)$ – примитивный многочлен.*

Исходя из утверждений выше, для того чтобы линейная рекуррентная последовательность порядка m над полем из q элементов имела максимальный период, необходимо и достаточно, чтобы ее минимальный многочлен был примитивным многочленом.

Более конкретно, над полем \mathbb{F}_2 необходимо реверсивный минимальный многочлен $F(x)$ на основе простых чисел Мерсенна. В этом случае для любого **ненулевого** заполнения ЛРП получается максимального периода.

Приведём конкретный пример многочленов $F_1(x)$, $F_2(x)$, $F_3(x)$ для заданных нашим алгоритмом РСЛОС L_1, L_2, L_3 . Для их генерации будем использовать втроенные функции пакета Wolfram Mathematica

Пусть $n_1 = 31, n_2 = 13, n_3 = 19$ ($2^{31} - 1, 2^{13} - 1$ и $2^{19} - 1$ — это известные числа Мерсенна).

Пример неприводимых многочленов соответствующих степеней из $\mathbb{F}_2[x]$.

$$F_1(x) = x^{31} + x^{30} + x^{29} + x^{28} + x^{27} + x^{24} + x^{21} + x^{19} + \\ + x^{18} + x^{13} + x^{12} + x^{10} + x^4 + x^3 + 1$$

$$F_2(x) = x^{13} + x^{12} + x^{11} + x^8 + x^7 + x^6 + x^5 + x^4 + x^2 + x + 1$$

$$F_3(x) = x^{19} + x^{15} + x^{13} + x^{12} + x^{10} + x^9 + x^5 + x^4 + x^2 + x + 1$$

В этом случае любое **ненулевое** заполнение каждого из L_1, L_2, L_3 даст нам максимальный период на каждом из регистров. Кроме того, каждый регистр сдвига был выбран разной длинны для того, чтобы исключить случай совместного заикливания двух ЛРП. Иными словами итоговый период совместной работы всех трёх регистров будет равна

$$N_{L_1 L_2 L_3} = \text{НОК}(2^{n_1} - 1, 2^{n_2} - 1, 2^{n_3} - 1) = \text{НОК}(2^{31} - 1, 2^{13} - 1, 2^{19} - 1) \approx 2^{63}$$

Это означает, что вектор начальных заполнений (u_1, u_2, u_3) регистров L_1, L_2, L_3 вернётся в сам в себя после $N_{L_1 L_2 L_3} \approx 2^{63}$ совместных тактов работы всех трёх регистров.

При этом мощность ключевого множества по-прежнему будет удовлетворять условию $|K| > 2^{64}$:

$$|K| = 2^{31} \cdot 13! \cdot 2^{19} > 2^{31} \cdot 2^{32} \cdot 2^{19} > 2^{82} > 2^{64}$$

2.2 Функции усложнения

Для усложнения аналитической сложности выходной последовательности РСЛОС используются функции усложнения. «Фильтрующая» функция f должна выбираться так, чтобы выходная последовательность имела распределение, близкое к равномерному распределению, и высокую линейную сложность.

Ниже представлены основные криптографические характеристики нелинейных преобразований, используемые в поточных шифрах.

Свойство 1. Функция f фильтрующего генератора должна быть сбалансированной, т.е. $|\{x : f(x) = 0\}| = 2^{n-1}$.

Свойство 2. У функции f фильтрующего генератора должны отсутствовать запреты.

Для выполнения свойства 2 достаточно, чтобы функция f была линейна по крайней переменной. Есть более общий критерий определения имеет ли булева функция запрет или нет.

Теорема 2.4. *Булева функция не имеет запрета тогда и только тогда, когда она сильно равновероятна.*

Свойство 3. У функции f фильтрующего генератора должна иметь высокую алгебраическую степень.

Свойство 4. Функция f фильтрующего генератора должна иметь высокую нелинейность, т.е. не иметь эффективных линейных статистических аналогов.

Свойство 5. Функция f фильтрующего генератора должна быть k -равновероятной для максимально возможного значения k .

Свойство 6. Функция f фильтрующего генератора (а также функция $f + 1$) не должны иметь аннигиляторов малой алгебраической степени.

На практике помимо достижения перечисленных условий необходимо дополнительно исследовать весь блок фильтрующего генератора целиком. Например, фильтрующий генератор с нелинейной функцией усложнения может представляться в виде ЛРП с меньшим (чем исходная ЛРП) периодом.

В работе Rachwalik и др. (2012) были найдены несколько функций усложнения порядков 25 и 27. Работа была нацелена на поиск таких функций, что обеспечивают наибольший период выходной последовательности. Кроме того у перечисленных в работе булевых функций были исследованы некоторые статистические свойства выходных последовательностей. Возьмём эти функции усложнения за основу, приняв некоторые переменные за единицу.

$$f_1 = x_0 \oplus x_6 \oplus x_{11} \oplus x_{14} \oplus x_{16} \oplus x_{17} \oplus x_{18} \oplus x_{19} \oplus x_{23} \oplus \\ \oplus x_4 x_{19} \oplus x_4 x_{21} \oplus x_5 x_{22} \oplus x_9 x_{19} \oplus x_1 x_{17} x_2 \oplus x_5 x_7 x_{18} \oplus x_5 x_{12} x_{19}$$

$$f_2 = x_0 \oplus x_3 \oplus x_6 \oplus x_{10} \oplus x_6 x_9 \oplus x_6 x_{10} \oplus x_4 x_9 x_{10}$$

$$f_3 = x_0 \oplus x_8 \oplus x_9 \oplus x_{10} \oplus x_{11} \oplus x_{10} x_{14} \oplus x_4 x_{16} \oplus x_{11} x_{16} \oplus x_1 x_5 x_7$$

2.3 Комбинирующая функция

2.3.1 Вычисление многочлена Жегалкина

С помощью быстрого преобразования Фурье вычислим коэффициенты многочлена Жегалкина для булевой функции g .

Моном	x	y	z	g	g_1	g_2	g_3
1	0	0	0	1	1	1	1
z	0	0	1	0	1	1	1
y	0	1	0	0	0	1	1
yz	0	1	1	1	1	0	0
x	1	0	0	1	1	1	0
xz	1	0	1	1	0	0	1
xy	1	1	0	0	0	1	0
xyz	1	1	1	0	0	0	0

В итоге получаем

$$g(x, y, z) = 1 \oplus y \oplus z \oplus xz$$

2.3.2 Характеристики

Вычислим характеристики комбинирующей функции, которые позволят нам определить возможность применения различных методов атак.

Из вида многочлена жегалкина видно, что **степень нелинейности** функции g равна 2.

Булева функция $h = xy$ является **аннигилятором** функции g , поскольку

$$hg = (1 \oplus y \oplus z \oplus xz)xy = xy \oplus xy \oplus xyz \oplus xyz = 0$$

Из предыдущего пункта видно, что **вес** функции $|g(x,y,z)| = 4$. Это означает, что функция g является **сбалансированной** (равновероятной)

$$P(g = 0) = \frac{4}{8} = P(g = 1)$$

Исследуем функцию g на корреляционную иммунность. Для этого будет поочередно фиксировать значения переменных и считать вероятность получения 0 до тех пока она не станет отличной от 0.5.

$$P(g(x,y,z) = 0 \mid x = 0) = P(1 \oplus y \oplus z = 0) = \frac{1}{2}$$

$$P(g(x,y,z) = 0 \mid x = 1) = P(1 \oplus y = 0) = \frac{1}{2}$$

$$P(g(x,y,z) = 0 \mid y = 0) = P(1 \oplus z \oplus xz = 0) = \frac{3}{4}$$

$$P(g(x,y,z) = 0 \mid y = 1) = P(z \oplus xz = 0) = \frac{1}{4}$$

$$P(g(x,y,z) = 0 \mid z = 0) = P(1 \oplus y = 0) = \frac{1}{2}$$

$$P(g(x,y,z) = 0 \mid z = 1) = P(y \oplus x = 0) = \frac{1}{2}$$

Из вычислений выше видно, что фиксация переменной y вероятность отклоняется от исходной. А значит функция g **не** является **корреляционно-иммунной**.

Вспомним, что произвольная булева функция f является k -устойчивой тогда и только тогда, когда она корреляционно-иммунная порядка k и равновероятна. Из рассуждений выше следует, что функция g также **не** является **устойчивой**.

Исследуем функцию g на наличие запретов. Для этого в пакете Wolfram Mathematica будем перебирать всевозможные выходные последовательности разной длины и пытаться решить систему уравнений относительно ячеек регистра сдвига. В результате такого исследования были найдены следующие невозможные выходные гаммы.

001000...	1001000...
111000...	0111000...
0001000...	1111000...

Из наличия запретов следует, что функция g **не** является **сильно-равновероятной**.

2.3.3 Наилучшее приближение линейной функцией

С помощью быстрого преобразования Фурье вычислим наилучшее приближение для булевой функции g .

Многочлен	x	y	z	g	g_1	g_2	g_3
1	0	0	0	1	1/2	1/2	1/2
z	0	0	1	0	1/2	0	0
y	0	1	0	0	1/2	0	1/4
$y \oplus z$	0	1	1	1	-1/2	1/2	1/4
x	1	0	0	1	1	1/2	0
$x \oplus z$	1	0	1	1	0	0	0
$x \oplus y$	1	1	0	0	0	1/2	-1/4
$x \oplus y \oplus z$	1	1	1	0	0	0	1/4

Коэффициент Уолша-Адамара на нулевом наборе переменных будет $1 - 2 \cdot (1/2) = 0$. В итоге получаем, что булева функция

$$\tilde{g}(x, y, z) = y \oplus (y \oplus z) \oplus (x \oplus y \oplus 1) \oplus (x \oplus y \oplus z) = y \oplus 1$$

является наилучшим линейным приближением булевой функции g . Посчитаем вероятность того, что значения этих функций совпали.

$$p(g = \tilde{g}) = \frac{3}{4}$$

2.3.4 Строгий лавинный критерий

Исследуем насколько сильно изменяются значения булевой функции $g(x, y, z)$ при малом изменении значений входных переменных. Для этого рассчитаем производные по направлениям координатных векторов в V_3

x	y	z	g	$D_x g$	$D_y g$	$D_z g$
0	0	0	1	0	1	1
0	0	1	0	1	1	1
0	1	0	0	0	1	1
0	1	1	1	1	1	1
1	0	0	1	0	1	0
1	0	1	1	1	1	0
1	1	0	0	0	1	0
1	1	1	0	1	1	0

Таблица выше демонстрирует, что производная по направлению y **не сбалансирована**. Это значит, что булева функция g **не** удовлетворяет **строгому лавинному критерию**. Более того, это означает что булева функция g **не** удовлетворяет **критерию распространения**.

3 Методы восстановления ключа

Зафиксируем параметры согласно размышлениям из пунктов 2.1 и 2.2 и исследуем применимость разных методов для восстановления ключа. Для каждого алгоритма вычислим его характеристики.

3.1 Метод тотального опробования

Самым наивным вариантом восстановления ключа будет алгоритм тотального опробования, который применим абсолютно к любому шифру. Идея алгоритма заключается в переборе всевозможных ключей и сопоставлении полученной гаммы с известным материалом.

Через $G(K, n)$ будем обозначать выработку гаммы Γ длины n представленным поточным алгоритмом шифрования на ключе K . Количество материала, необходимое для однозначного определения ключа составляет $m = \lceil \log_2(|K|) \rceil = 83$ бита.

Алгоритм 1: Метод тотального опробования

Вход: Пара открытый и шифрованный текст $P, C \in V_{83}$

Выход: Ключ шифрования K

1. Для каждого $k_1 \in V_{31}$
 2. Для каждого $k_2 \in S(13)$
 3. Для каждого $k_3 \in V_{19}$
 4. Сформировать ключ $K = (k_1, k_2, k_3)$
 5. Вычислить $\Gamma = G(K, 83)$
 6. Если $P \oplus C = \Gamma$, то
 7. Закончить алгоритм и вернуть K
-

Вероятность работы алгоритма $p(A) = 1$, поскольку алгоритм гарантированно находит ключ.

Средняя трудоёмкость алгоритма $S(A) = \frac{2^{31} \cdot 13! \cdot 2^{19} + 1}{2} \approx 2^{82}$ использований алгоритма выработки гаммы. Она же равна **трудоёмкости по Шеннону** $Q(A)$.

Объём памяти необходимый для работы алгоритма $M(A) = O(1)$ бит, поскольку алгоритм хранит только локальные переменные.

3.2 Метод встречи по середине

Прежде чем применить метод встречи по середине, необходимо представить преобразование исходного шифра как двойное последовательное шифрование. Иными словами если алгоритм зашифрования $E_K(P)$ открытого текста P на ключе K можно представить в виде $E'_{k_1}(E''_{k_2}(P))$, то можно применить метод встречи по середине. Например, выход булевой функции $g(x, y, z) = 1 \oplus y \oplus z \oplus xz$, где x, y, z - выходы фильтрующих функций соответствующих блоков шифра, может быть представлен, как сложение двух булевых функций $h_1 = 1 \oplus y$ и $h_2 = z \oplus xz$.

Наиболее эффективным с точки зрения трудоёмкости этот метод становится тогда, когда алгоритм зашифрования $E_k(P)$ удалось представить в виде $E'_{k_1}(E''_{k_2}(P))$ так, что длина $k_1 \approx$ длине k_2 .

Теоретически блок коммутатора (перестановка) K_2 можно было бы разделить на 2 блока коммутатора меньшей длины с общим входом. Поскольку блок РСЛОС не является ключевым, то он может быть задан в виде таблицы.

Однако для упрощения рассмотрим разделение так, что E'_{k_1} – это центральный блок, а E''_{k_2} – это правый и левый блоки. Для однозначного отделения ключа k_2 достаточно 33 бит материала, иными словами исходный материал (83 бита) разделить как 33 и 50 бит соответственно.

Разделим исходный алгоритм шифрования $E_K(P)$, где P – открытый

текст длины n и напишем текст работы алгоритма.

$$\begin{aligned}
 E_K(P) &= P \oplus G(K, n) = \\
 &= P \oplus g(x, y, z) = (P \oplus 1 \oplus y) \oplus z \oplus xz = E''_{k_2}(P) \oplus z \oplus xz = \\
 &= E'_{k_1}(E''_{k_2}(P))
 \end{aligned}$$

Алгоритм 2: Метод встречи по середине

Вход: Пара открытый и шифрованный текст $P, C \in V_{83}$

Выход: Ключ шифрования K

1. Разделить исходный материал на $P_1, C_1 \in V_{33}$ и $P_2, C_2 \in V_{50}$
 2. Для каждого $k_2 \in S(13)$
 3. Вычислить $\Pi = E''_{k_2}(P_1)$
 4. Занести в ячейку с адресом Π ключ k_2
 /* в среднем в каждой ячейке памяти будет лежать около
 1 ключа */
 5. Для каждого $k_1 \in V_{31}$
 6. Для каждого $k_3 \in V_{19}$
 7. Сформировать ключ $\tilde{k}_1 = (k_1, k_3)$
 8. Вычислить $\Pi = E'^{-1}_{k_1}(C_1)$
 9. Если ячейка с адресом Π не пуста то
 10. Достать из неё ключ k_2
 11. Сформировать ключ $K = (k_1, k_2, k_3)$
 12. Доопробовать ключ K на материале P_2, C_2
 13. Если доопробование успешно то
 14. Закончить алгоритм и вернуть K
-

Вероятность работы алгоритма $p(A) = 1$, поскольку алгоритм гарантированно находит ключ.

Средняя трудоёмкость алгоритма $S(A) = 13! \cdot s_1 + \frac{2^{31} \cdot 2^{19} + 1}{2} \cdot s_2$, где s_1 – сложность алгоритма зашифрования E'' , s_2 – сложность алгоритма расшифрования E'^{-1} . Средняя трудоёмкость равна **трудоёмкости по Шеннону** $Q(A) = S(A) \approx 2^{33} \cdot s_1 + 2^{49} \cdot s_2$.

Объём памяти, который необходим для успешной работы алгоритма $M(A) = 2^{33} \cdot 13! + O(1) \approx 2^{66} + O(1)$ бит, поскольку алгоритму требуется хранить таблицу ключей из $S(13)$. Для описания одного ключа из $S(13)$ требуется 33 бита (2^{33} – это размерность пространства памяти). Поэтому в среднем будет в каждой 33-битной ячейке памяти храниться не более одного ключа. К сожалению, в современных реалиях хранение таблицы размером 2^{66} бит не представляется возможным.

3.3 Статистический метод

В основе статистического метода лежит построение некоторой статистики, на которой строится предположение о правильности ключа или его части. Если эта статистика превышает некоторый статистический порог T (который зависит от α и β , статистических ошибок 1-го и 2-го рода соответственно), то можно с вероятностью $p = 1 - \alpha$ полагать, что ключ или его часть подобраны верно.

В пункте 2.3.2 было отмечено, что комбинирующая функция g не обладает корреляционной иммунностью, а в подпункте 2.3.3 был найден линейный статистический аналог функции g , вероятность совпадения с которым больше $1/2$. Это позволяет провести атаку либо с использованием **корреляционного метода**, либо с использованием **линейного статистического аналога**. Рассмотрим статистический метод на примере корреляционной зависимости.

Вспомним, что

$$p(g = \bar{y}) = \frac{6}{8} = \frac{3}{4}$$

Это означает, что можно построить такую статистику, которая бы отражала корреляционную зависимость выходов гаммы и центрального блока. Функция корреляции последовательностей $\{\gamma_i\}_{i=1}^N$ и $\{\theta_i\}_{i=1}^N$ длины N

$$C(\gamma, \theta) = \sum_{i=1}^N (-1)^{\gamma_i + \theta_i}$$

Наблюдение выше позволяет использовать выход последовательности центрального блока, как индикатор того, коррелирует ли с ней выходная гамма всего шифра. Идея метода заключается в том, чтобы для всевозможных ключей центрального блока вырабатывать выходные значения этого блока и считать функцию корреляции между полученной и взламываемой гаммой. Если эта функция превышает некоторый статистический порог T (который зависит от длины N выработанной гаммы и α и β , статистических ошибок 1-го и 2-го рода соответственно), то можно с вероятностью $p = 1 - \alpha$ полагать, что ключ центрального блока подобран верно.

Количество материала в данном случае состоит из двух частей: материал для статистического исследования корреляции и материал для доопробования. Чем больше будет материала для статистического исследования, тем точнее покажет себя алгоритм, поэтому примем это N , количество бит, за параметр алгоритма. Материал же для доопробования можно полагать равным 50 бит (достаточным для однозначного определения ключа).

Обозначим за $G'(k, n)$ выработку гаммы длины n центальным блоком на ключе k .

Алгоритм 3: Корреляционный метод

Вход: Пары открытый и шифрованный текст

$$P_1, C_1 \in V_N, P_2, C_2 \in V_{50}$$

Выход: Ключ шифрования K

1. Вычислить последовательность $\gamma = P_1 \oplus C_1$
 2. Для каждого $k_2 \in S(13)$
 3. Вычислить последовательность $\theta = G'(k_2, N)$
 4. Вычислить $C = C(\gamma, \theta)$
 5. Если $C > T$ то
 6. Для каждого $k_1 \in V_{31}$
 7. Для каждого $k_3 \in V_{19}$
 8. Сформировать ключ $K = (k_1, k_2, k_3)$
 9. Доопробовать ключ K на материале P_2, C_2
 10. Если доопробование успешно то
 11. Закончить алгоритм и вернуть K
-

Как уже отмечалось ранее **вероятность работы** алгоритма $p(A) = 1 - \alpha$, где α – это статистическая ошибка 1-го рода, зависящая от N .

Средняя трудоёмкость алгоритма при выбранных α и β

$$S(A) = (1 - \alpha) \cdot \left(\frac{13! + 1}{2} \cdot N + \beta \frac{13! + 1}{2} \frac{2^{31} \cdot 2^{19} + 1}{2} \right) + \\ + \alpha (13! \cdot N + \beta \cdot 2^{31} \cdot 2^{19}).$$

Объём памяти необходимый для работы алгоритма $M(A) = O(1)$ бит, поскольку алгоритм хранит только локальные переменные.

Стоит так же отметить, что на этапе доопробования для нахождения оставшейся части ключа также, можно использовать разные методы. Например, успешное использование метода встречи по середине может уменьшить среднюю трудоёмкость до $O(N \cdot (2^{33} + 2^{31} + 2^{19}))$

3.4 Метод частичного опробования с использованием аннигилятора

В пункте 2.3.2 было замечено, что функция $h = xy$ является аннигилятором комбинирующей функции g . Заметим, что функция h зависит только от выходов левого и центрального блоков. Это означает, что на позициях, где выходная гамма равна 1, значение функции $h = 0$ (из определения аннигилятора). Это позволяет успешно воспользоваться **методом частичного опробования**.

Перебирая всевозможные ключи левого и центрального блока мы можем определить является ли совместный ключ двух блоков валидным, если на всех позициях, где выходная гамма равна 1 функция h равна нулю. Стоит заметить, что для успешного применения метода достаточно 83 бит **материала** с тем условием, что среди них будет 64 бита будут единицами.

Будем обозначать через $G_x(K_x, n)$ и $G_y(K_y, n)$ вычисление n бит выходной гаммы левого и центрального блоков соответственно.

Алгоритм 4: Метод частичного опробования с использованием аннигилятора

Вход: Пары открытый и зашифрованный текст $P, C \in V_{83}$

Выход: Ключ шифрования K

1. Вычислить $\Gamma = P \oplus C$
 2. Для каждого $k_1 \in V_{31}$
 3. Вычислить $\Gamma_x = G_x(k_1, 83)$
 4. Для каждого $k_2 \in S(13)$
 5. Вычислить $\Gamma_y = G_y(k_2, 83)$
 6. Вычислить $\Gamma_{xy} = \Gamma_x \& \Gamma_y$
 7. Проверить, что на всех местах, где в последовательности Γ стоят единицы в последовательности Γ_{xy} стоят нули.
 8. **Если проверка успешна то**
 9. **Для каждого** $k_3 \in V_{19}$
 10. Сформировать ключ $K = (k_1, k_2, k_3)$
 11. Доопробовать ключ K на материале P, C
 12. **Если доопробование успешно то**
 13. Закончить алгоритм и вернуть K
-

Вероятность работы алгоритма $p(A) = 1$, поскольку алгоритм гарантированно находит ключ.

Средняя трудоёмкость алгоритма $S(A) = \frac{2^{31} \cdot 13! + 1}{2} s_1 + \frac{2^{19} + 1}{2} s_2$, где s_1 – сложность алгоритма выработки гаммы Γ_{xy} и проверки согласованности её с материалом, s_2 – сложность алгоритма доопробования.

Средняя трудоёмкость равна **трудоёмкости по Шеннону** $Q(A) = S(A) \approx 2^{63} \cdot s_1 + 2^{18} \cdot s_2$.

Объём памяти необходимый для работы алгоритма $M(A) = O(1)$ бит, поскольку алгоритм хранит только локальные переменные.

4 Выводы и рекомендации

Из разобранных методов для восстановления ключа лучше всего подходят **корреляционный метод** и **метод частичного опробования с использованием аннигилятора**. В зависимости от количества материала или имеющихся вычислительных мощностей можно отдать предпочтение одному или другому методу.

Для повышения стойкости алгоритма шифрования стоит выбрать более стойкую комбинирующую функцию. А именно, стоит выбрать функцию $g(x, y, z)$, аннигилятор которой зависит от всех выходных блоков. Также стоит гарантировать корреляционную иммунность хотя бы 1 порядка. Кроме того универсальным способом увеличения стойкости алгоритма является увеличение мощности ключевого множества за счёт увеличения длины регистров РСЛОС.