Подход к автоматизации дифференциальных методов анализа

Тавокин Михаил Александрович Бондакова Ольга Сергеевна

ООО «СФБ Лаб»

РТУ МИРЭА

РусКрипто'2025

Mikhail.Tavokin@sfblaboratory.ru

Вступление

Цель – показать возможности использования MILP при построении Rebound-атаки на функции хэширования, использующие блочные шифры с нетипичными размерами блока.

Актуальность – семейство функций хэширования «MORA-Z» является перспективным семейством параметризуемых низкоресурсных криптографических алгоритмов.

Архитектура хэш-функции «MORA-Z»

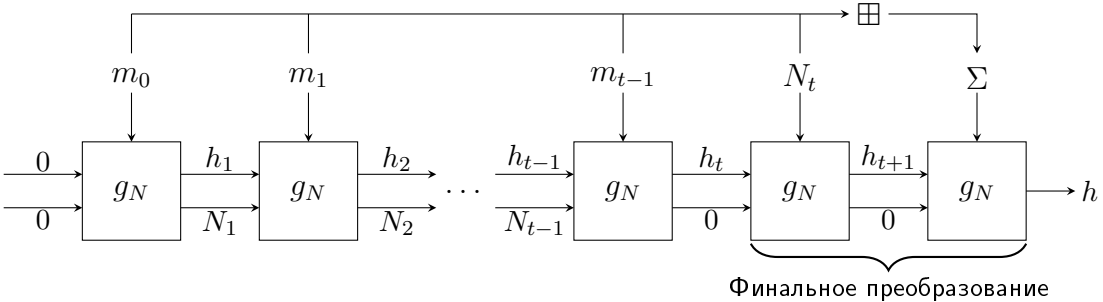
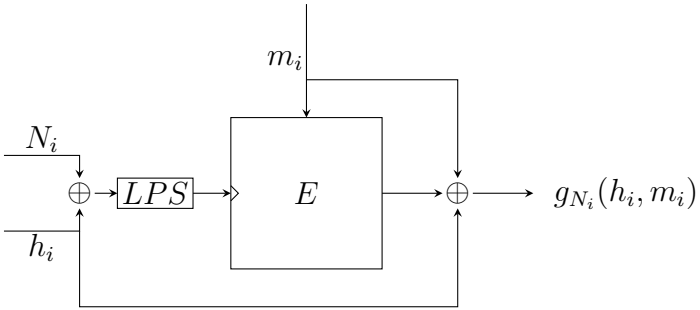


Схема функции сжатия g



$$E(K, m) = X[k_{r+1}]LPSX[k_r] \dots LPSX[k_1](m)$$

Различия алгоритмов. Часть 1

Алгоритм	Размер внутреннего состояния, $a \times b$	Размер хэш-значения, Z бит	Число раундов, r
MORA-64	4×4	64	10
MORA-80	5×4	80	15
MORA-96	6×4	96	15
MORA-128	8×4	128	20
MORA-160	10×4	160	20

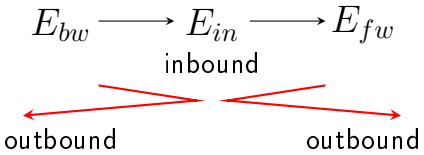
Различия алгоритмов. Часть 2

Подстановка $\pi = (15, 9, 1, 7, 13, 12, 2, 8, 6, 5, 14, 3, 0, 11, 4, 10)$ для всех алгоритмов одинаковая.

Перестановка τ является уникальной для каждого алгоритма:

Алгоритм	Перестановка τ
MORA-64	(0, 4, 8, 12, 1, 5, 9, 13, 2, 6, 10, 14, 3, 7, 11, 15)
MORA-80	(0, 4, 8, 12, 1, 5, 9, 16, 2, 6, 13, 17, 3, 10, 14, 18, 7, 11, 15, 19)
MORA-96	(0, 4, 8, 12, 1, 5, 9, 16, 2, 6, 13, 20, 3, 10, 17, 21, 7, 14, 18, 22, 11, 15, 19, 23)
MORA-128	(0, 4, 8, 12, 1, 5, 9, 16, 2, 6, 13, 20, 3, 10, 17, 24, 7, 14, 21, 28, 11, 18, 25, 29, 15, 22, 26, 30, 19, 23, 27, 31)
MORA-160	(0, 4, 8, 12, 1, 5, 9, 16, 2, 6, 13, 20, 3, 10, 17, 24, 7, 14, 21, 28, 11, 18, 25, 32, 15, 22, 29, 36, 19, 26, 33, 37, 23, 30, 34, 38, 27, 31, 35, 39)

Rebound-атака



Rebound-атака – алгоритм криптографического анализа функций хэширования, которые используют *XSPL* блочные шифры. Цель – поиск коллизий.

$$E = E_{fw} \circ E_{in} \circ E_{bw}$$

Этапы атаки

	Этап атаки	Метод автоматизации
1	Построение паттерна дифференциального пути	MILP
2	Inbound-фаза	Перебор
3	Outbound-фаза	MILP

MILP

MILP (англ. Mixed Integer Linear Programming – смешанное целочисленное линейное программирование) — это метод оптимизации, который используется для решения задач, использующих как вещественные, так и целочисленные переменные.

Сущность MILP

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \leq b_1, & a_{ij} \in \mathbb{R}, \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \leq b_2, & b_i \in \mathbb{R}, \\ \vdots & \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \leq b_m, & x_j \in \mathbb{Z}, i \in \overline{1,m}, j \in \overline{1,n} \end{cases}$$

Решением задачи линейного программирования является поиск таких решений $x_1, \ldots, x_n, n \in \mathbb{N}$, при которых с учетом ограничений осуществляется поиск либо минимума, либо максимума заданной **целевой функции** в зависимости от поставленной задачи.

$$z = \sum_{j=1}^{n} c_j x_j \to max\,(min), z \in \mathbb{Z}$$

Моделирование первого этапа атаки

Переменные:

1. $\mathbb{S}_{r,k} \in \{0,1\}$ – статус активности ячейки (задействованной в S преобразовании) $k = 4 \cdot i + j$ (строка i, столбец j). 1 активна, 0 – в противном случае;
2. $\mathbb{M}_{r,j} \in \{0,1\}$ – статус активности строки j до преобразования L. 1 активна, 0 – в противном случае.

Неравенства для первого этапа

Основное неравенство для каждого раунда r:

$$5 \cdot \mathbb{M}_{r,i} \leq \sum_j \mathbb{S}_{r,\tau(4i+j)} + \sum_j \mathbb{S}_{r+1,4i+j} \leq 8 \cdot \mathbb{M}_{r,i}$$

Условие ненулевой суммы активных ячеек:

$$\sum_k \mathbb{S}_{0,k} \geq 1$$

Совпадение промежуточных состояний:

$$\mathbb{S}_{0,k} = \mathbb{S}_{r,k}$$

Целевая функция первого этапа

Целевой функцией при моделировании первого этапа является функция подсчета числа активных ячеек. На данном этапе производится поиска минимума данной целевой функции:

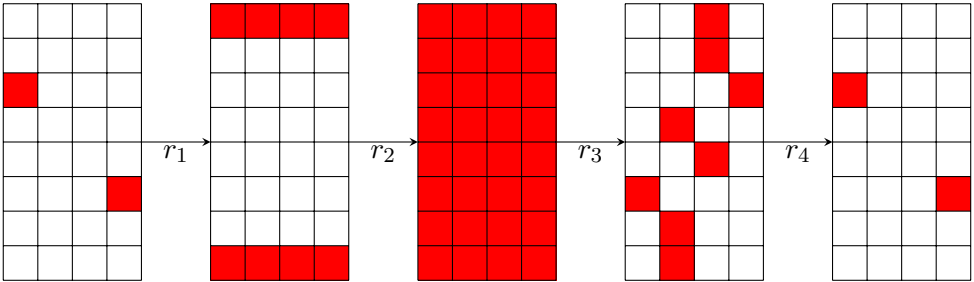
$$\sum_{r,k} \mathbb{S}_{r,k} \to min$$

Технические характеристики

- Процессор: Intel core i5 9300HF (2.4 ГГц, 4 ядра, 8 потоков);
- ОЗУ: 16 ГБ;
- Версия интерпретатора Python: `Python 3.9.0`;
- Версия компилятора C: `GNU GCC 12`;
- MILP-решатель: `IBM ILOG CPLEX Interactive Optimizer` версии `22.1.0.0`.

Результат первого этапа

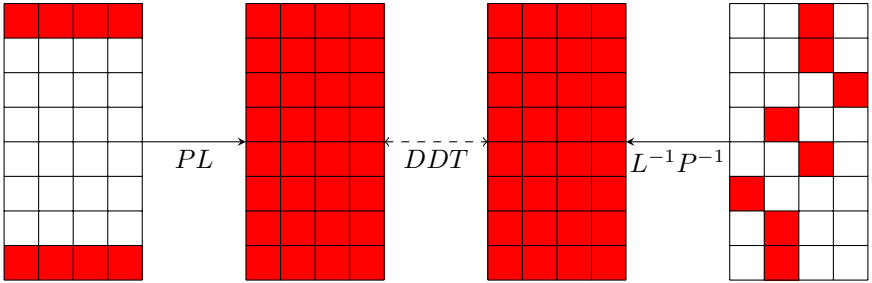
Всюду далее будут демонстрироваться результаты моделирования для алгоритма MORA-128 на 4 раунда.



$$E_{in} = L \circ P \circ S \circ X[k_3] \circ L \circ P$$
$$E_{bw} = S \circ X[k_2] \circ L \circ P \circ S \circ X[k_1], E_{fw} = X[k_5] \circ L \circ P \circ S \circ X[k_4]$$

Второй этап атаки

Второй этап заключается в «состыковке» возможных пар дифференциалов на концах состояний композиции E_{in}.



Моделирование третьего этапа. Преобразование S. Часть 1

Пусть π – биективная подстановка, имеющая вход (x_0, x_1, x_2, x_3) и выход (y_0, y_1, y_2, y_3), где $x_i, y_i \in V_2, i \in \overline{0,3}$.

$$
\begin{cases}
x_0 - A \leq 0 \\
x_1 - A \leq 0 \\
x_2 - A \leq 0 \\
x_3 - A \leq 0 \\
x_0 + x_1 + x_2 + x_3 - A \geq 0 \\
4(x_0 + x_1 + x_2 + x_3) - (y_0 + y_1 + y_2 + y_3) \geq 0 \\
4(y_0 + y_1 + y_2 + y_3) - (x_0 + x_1 + x_2 + x_3) \geq 0
\end{cases}
$$

Моделирование третьего этапа. Преобразование S. Часть 2

Пусть мы хотим смоделировать подстановку π, в которой переход осуществляется с определенной вероятностью:

$$p = P[(x_0, x_1, x_2, x_3) \to (y_0, y_1, y_2, y_3)]$$

Согласно таблице DDT, возможны три ненулевые вероятности: $2^{-3}, 2^{-2}$ и 1. Закодируем эту информацию в двух битах (π_0, π_1):

$$(\pi_0, \pi_1) = (0, 0) \to p = 1$$
$$(\pi_0, \pi_1) = (0, 1) \to p = 2^{-3}$$
$$(\pi_0, \pi_1) = (1, 0) \to p = 2^{-2}$$

Моделирование третьего этапа. Преобразование S. Часть 2

Закодируем вход, выход и вероятность перехода в единый двоичный вектор:

$$\xi = (x_0, x_1, x_2, x_3, y_0, y_1, y_2, y_3, \pi_0, \pi_1)$$

Для представления входного вектора как систему линейных неравенств, необходимо воспользоваться H-представлением.

$$\begin{cases} (\gamma_{0,0}, \gamma_{0,1}, \ldots, \gamma_{0,9}) \cdot \xi + \gamma_{0,10} \leq 0 \\ \vdots \\ (\gamma_{t-1,0}, \gamma_{t-1,1}, \ldots, \gamma_{t-1,9}) \cdot \xi + \gamma_{t-1,10} \leq 0 \end{cases}$$

где $\gamma_{i,j} \in \mathbb{Z}$ – искомые переменные, $j \in \overline{0, 10}, i \in \overline{0, t-1}$,
t означает общее число неравенств.
Поиск $\gamma_{i,j}$ осуществляется с помощью программы MILES.

Моделирование третьего этапа. Преобразование P

Поскольку перестановка является линейным преобразованием и влияет лишь на расположение ячеек, то в построении отдельных неравенств не нуждается. Его можно «склеить» со следующим преобразованием.

Моделирование третьего этапа. Преобразование L. Часть 1

Пусть $(x_0, x_1, \ldots, x_{15})$ – векторная строка матрицы внутреннего состояния, G^{bin} – значение битовой матрицы, $(y_0, y_1, \ldots, y_{15})$ – векторная строка, полученная в результате умножения $(x_0, x_1, \ldots, x_{15})$ на G^{bin}. Формулировка значений $y_k, k \in \overline{0, 15}$ выглядит следующим образом.

$$y_k = x_0 \cdot G^{bin}_{0,k} \oplus x_1 \cdot G^{bin}_{1,k} \oplus \ldots \oplus x_{15} \cdot G^{bin}_{15,k}$$

где $G^{bin}_{i,j}$ - элемент из строки i и столбца j матрицы G^{bin}.

Моделирование третьего этапа. Преобразование L. Часть 2

Для построения системы моделируется операция «n-XOR», то есть $a_0 \oplus a_1 \oplus \ldots \oplus a_n = b$, где $a_0, \ldots, a_n, b \in \{0,1\}$ с применением вспомогательных переменных $d_i \in \{0,1\}, i \in \mathbb{N}$. Если n чётное, то неравенство в общем случае выглядит следующим образом:

$$\sum_{i=0}^{n} a_i + b = (n+2)d_1 - \left(nd_2 + (n-2)d_3 + \ldots + 2d_{\frac{n}{2}+1}\right)$$

В случае, если n нечётное неравенство выглядит так:

$$\sum_{i=0}^{n} a_i + b = (n+1)d_1 - \left((n-1)d_2 + (n-3)d_3 + \ldots + 2d_{\frac{n-1}{2}+1}\right)$$

Моделирование третьего этапа. Целевая функция

В случае Rebound-атаки максимизируется вероятность дифференциального пути фазы **Outbound**. Пусть каждой паре $(\pi_{i,0}, \pi_{i,1})$ на раунде i поставим в соответствии значение $-\log_2 p_i$. Тогда в MILP модели целевая функция будет построена следующим образом:

$$\sum_i 2\pi_{i,0} + 3\pi_{i,1} \to min$$

Результат третьего этапа

В итоге получен дифференциальный путь фазы **Outbound** с вероятностью 2^{-51}.

$\alpha = \Delta_L^0$	0x00000000d00000000000000a00000000
Δ_S^1	0x0000000040000000000000700000000
Δ_L^1	0xbd7b0000000000000000000000a7e9
Δ_S^2	0x761100000000000000000000000001111
INBOUND	
Δ_L^3	0x00900050000a0f0000b0c0000c000500
Δ_S^4	0x00a0005000090d0000c050000c000500
$\beta = \Delta_L^4$	0x00000000d00000000000000a00000000

Результаты для остальных алгоритмов

Алгоритм\Кол-во раундов	2	3	4	5	6	7
MORA-64	2^{-9}	2^{-16}	2^{-21}	2^{-66}	2^{-76}	$\times$
MORA-80	2^{-10}	2^{-24}	2^{-41}	2^{-68}	2^{-94}	$\times$
MORA-96	2^{-15}	2^{-29}	2^{-40}	$\times$	2^{-92}	2^{-121}
MORA-128	$\times$	2^{-43}	2^{-51}	$\times$	$\times$	$\times$
MORA-160	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$

Таблица: Результат поиска дифференциальных путей для разных алгоритмов с 2 по 7 раунды внутреннего блочного шифра

Проблематика. Outbound-фаза

r\Алгоритм	64	80	96	128	160
2	240/1335	300/1668	360/2001	×	×
3	680/2540	850/3174	1020/3808	1360/5076	×
4	1120/3742	1400/4681	1680/5613	2240/7479	×
5	1560/4959	1950/6191	2340/7423	3120/9887	×
6	2000/6164	2500/7701	3000/9233	×	×
7	×	×	3660/11045	×	×

Таблица: Количество переменных / неравенств в модели MILP в зависимости от числа раундов и алгоритма

Перспективы

- Перенос вычислений на графическом процессоре (GPU);
- Оптимизация перебора на фазе Inbound;
- Подходы к оптимизации MILP-моделей.

Вопросы?