MODELING AND ANALYSIS OF INFORMATION SYSTEMS, VOL. 30, NO. 2, 2023
journal homepage: www.mais-journal.ru

ALGORITHMS

On Simplifying Expressions with Mixed Boolean-Arithmetic

Y. V. Kosolapov[1]

DOI: 10.18255/1818-1015-2023-2-140-159

[1]Southern Federal University, 105/42 Bolshaya Sadovaya str., Rostov-on-Don, 344006, Russia.

MSC2020: 93B11
Research article
Full text in Russian

Received April 3, 2023
After revision May 17, 2023
Accepted May 17, 2023

Mixed Boolean-Arithmetic expressions (MBA-expressions) with t integer n-bit variables are often used for program obfuscations. Obfuscation consists of replacing short expressions with longer equivalent expressions that seem to take the analyst more time to explore. The paper shows that to simplify linear MBA-expressions (reduce the number of terms), a technique similar to the technique of decoding linear codes by information sets can be applied. Based on this technique, algorithms for simplifying linear MBA-expressions are constructed: an algorithm for finding an expression of minimum length and an algorithm for reducing the length of an expression. Based on the length reduction algorithm, an algorithm is constructed that allows to estimate the resistance of an MBA-expression to simplification. We experimentally estimate the dependence of the average number of terms in a linear MBA-expression returned by simplification algorithms on n, the number of decoding iterations, and the power of the set of Boolean functions, by which a linear combination with a minimum number of nonzero coefficients is sought. The results of the experiments for all considered t and n show that if before obfuscation the linear MBA-expression contained $r = 1, 2, 3$ terms, then the developed simplification algorithms with a probability close to one allow using the obfuscated version of this expression find an equivalent one with no more than r terms. This is the main difference between the information set decoding technique and the well-known techniques for simplifying linear MBA-expressions, where the goal is to reduce the number of terms to no more than 2^t. We also found that for randomly generated linear MBA-expressions with increasing n, the average number of terms in the returned expression tends to 2^t and does not differ from the average number of terms in the linear expression returned by known simplification algorithms. The results obtained, in particular, make it possible to determine t and n for which the number of terms in the simplified linear MBA-expression on average will not be less than the given one.

Keywords: code obfuscation; MBA-expressions; simplification of MBA-expressions; decoding by information sets

INFORMATION ABOUT THE AUTHORS

Yury V. Kosolapov
corresponding author
orcid.org/0000-0002-1491-524X. E-mail: itaim@mail.ru
Associated professor.

For citation: Y. V. Kosolapov, "On Simplifying Expressions with Mixed Boolean-Arithmetic", *Modeling and analysis of information systems*, vol. 30, no. 2, pp. 140-159, 2023.

МОДЕЛИРОВАНИЕ И АНАЛИЗ ИНФОРМАЦИОННЫХ СИСТЕМ, ТОМ 30, № 2, 2023
сайт журнала: www.mais-journal.ru

ALGORITHMS

Об упрощении выражений со смешанной битовой и целочисленной арифметикой

Ю. В. Косолапов[1]

DOI: 10.18255/1818-1015-2023-2-140-159

[1]Южный федеральный университет, ул. Большая Садовая, 105/42, Ростов-на-Дону, 344006, Россия.

УДК 004.056.5
Научная статья
Полный текст на русском языке

Получена 3 апреля 2023 г.
После доработки 17 мая 2023 г.
Принята к публикации 17 мая 2023 г.

Выражения со смешанной булевой и целочисленной арифметикой (далее — MBA-выражения, от англ. Mixed Boolean-Arithmetic) от t целочисленных n-битных переменных часто находят применение при обфускации (запутывании) программного кода. Запутывание заключается в замене коротких выражений более длинными эквивалентными выражениями, на исследование которых, как представляется, аналитиком может быть затрачено больше времени. В работе показано, что для упрощения линейных MBA-выражений (сокращения количества слагаемых) может быть применена техника, аналогичная технике декодирования линейных кодов по информационным совокупностям. На основе этой техники в работе построены алгоритмы упрощения линейных MBA-выражений: алгоритм нахождения выражения с минимальным числом слагаемых и алгоритм сокращения числа слагаемых. На основе алгоритма сокращения числа слагаемых построен алгоритм, позволяющий оценить стойкость MBA-выражения к упрощению. В работе экспериментально оценена зависимость среднего числа слагаемых в линейном MBA-выражении, возвращаемом алгоритмами упрощения, от разрядности n, числа итераций декодирования и мощности набора булевых функций, по которому ищется линейная комбинация с минимальным числом ненулевых коэффициентов. Результаты экспериментов для всех рассмотренных t и n показывают, что если до обфускации линейное MBA-выражение содержало $r = 1, 2, 3$ слагаемых, то разработанные алгоритмы упрощения с вероятностью, близкой к единице, позволяют по обфусцированному варианту этого выражения найти эквивалентное с числом слагаемых не более r. В этом заключается главное отличие техники декодирования по информационным совокупностям от известных техник упрощения линейных MBA-выражений, в которых целью является сокращение числа слагаемых до не более чем 2^t. В работе также установлено, что для случайно сгенерированных линейных MBA-выражений с ростом n среднее число слагаемых в возвращаемом выражении стремится к 2^t и не отличается от среднего числа слагаемых в линейном выражении, возвращаемом известными алгоритмами упрощения. Полученные результаты, в частности, позволяют определить t и n, для которых количество слагаемых в упрощенном линейном MBA-выражении в среднем будет не менее заданного.

Ключевые слова: обфускация программного кода; MBA-выражения; упрощение MBA-выражений; декодирование по информационным совокупностям

ИНФОРМАЦИЯ ОБ АВТОРАХ

Юрий Владимирович Косолапов
автор для корреспонденции

orcid.org/0000-0002-1491-524X. E-mail: itaim@mail.ru
доцент, канд. техн. наук.

Для цитирования: Y. V. Kosolapov, “On Simplifying Expressions with Mixed Boolean-Arithmetic”, *Modeling and analysis of information systems*, vol. 30, no. 2, pp. 140-159, 2023.

Введение

Обфускация (или запутывание) программного кода — это преобразование, сохраняющее функциональность программы, с целью затруднения исследования кода аналитиком (человеком и/или автоматическими средствами). Несмотря на то, что не существует универсального обфусцирующего преобразования, стойкого в рамках модели «виртуального черного ящика» [1], исследования в области построения и анализа практически стойких обфусцирующих преобразований продолжаются. Одним из используемых на практике подходов является замена коротких арифметических и битовых выражений более длинными выражениями, содержащими как битовые операции над целыми числами (в том числе, битовые операции $\wedge$, $\vee$, $\oplus$ над целыми числами), так и арифметические (операции + и $\times$) [2]. Далее такие выражения будем называть MBA-выражениями (аббревиатура от английского названия Mixed Boolean-Arithmetic). Эвристическое обоснование запутывающих свойств такого подхода строится, с одной стороны, на отсутствии математических правил упрощения MBA-выражений, а с другой стороны, на том, что более длинные программы менее читабельны, чем короткие [3]. В области защиты программ от исследования MBA-выражения часто применяются для построения *непрозрачных (неявных) предикатов* (*opaque predicates*) [4]. Напомним, что непрозрачным предикатом $P(x_1, \dots, x_t)$ от t переменных называется логическая функция, *тождественно* равная нулю или единице (константный предикат). Обычно переменные $x_1, \dots, x_t$ являются целочисленными. Когда значение непрозрачного предиката известно разработчику, то этот предикат может использоваться для добавления, например, условного оператора, в котором всегда будет выполняться только одна из альтернатив, известная разработчику. Непрозрачный предикат обычно строится так, чтобы по его виду было вычислительно трудно распознать в нем константный предикат [5]. В этом случае аналитику потребуется затратить ресурсы (вычислительные и временные) на анализ обеих альтернатив. В конструкции условного оператора, изображенного на рис. 1, тождественно истинный непрозрачный предикат $P_T(x_1, \dots, x_t)$ используется для добавления недостижимого кода, на анализ и понимание которого аналитиком, как представляется, будут затрачены ресурсы.

```
if P_T(x_1, ..., x_t) = 1 then
  /* Всегда выполняемый код */
else
  /* Недостижимый код */
```

Fig. 1. Using an identically true opaque predicate

Рис. 1. Использование тождественно истинного непрозрачного предиката

Как отмечено в [5], в свободно распространяемых и коммерческих средствах обфускации используется сравнительно небольшой набор непрозрачных предикатов. Поэтому они могут быть распознаны аналитиком путем сравнения исследуемого предиката из программы с предикатами из словаря. Известно, например, что некоторые оптимизаторы могут распознавать непрозрачные предикаты и удалять недостижимый код. В частности, в [6] утверждается, что многие непрозрачные предикаты не устойчивы к оптимизации, применяемой в компиляторе clang. Для расширения разнообразия предикатов, устойчивых к распознаванию по словарю и/или оптимизации, могут использоваться MBA-выражения. Например, истинность предикатов $P(x_1, \dots, x_t)$ и $P(x_1, \dots, x_t) + E(x_1, \dots, x_t)$ совпадает, если $E(x_1, \dots, x_t)$ — тождественно равное нулю MBA-выражение. MBA-выражения и сами могут использоваться в роли непрозрачных предикатов. Например, если $E_1(x_1, \dots, x_t)$ и $E_2(x_1, \dots, x_t)$ — разные тождественно равные нулю MBA-выражения, то с помощью них, например, следующим образом могут быть построены тождественно равный нулю предикат $P_F(x_1, \dots, x_n)$ и тождественно равный единице предикат $P_T(x_1, \dots, x_n)$:

$$P_F(x_1, \dots, x_t) = (E_1(x_1, \dots, x_t) \neq E_2(x_1, \dots, x_t)),$$
$$P_T(x_1, \dots, x_t) = (E_1(x_1, \dots, x_t) \geqslant E_2(x_1, \dots, x_t)).$$

С появлением обфускации на основе MBA-выражений возникла и задача деобфускации таких выражений. Целью деобфускации MBA-выражений обычно является получение по выражению $E(x_1, \dots, x_t)$ эквивалентного ему MBA-выражения $E'(x_1, \dots, x_t)$, в котором количество переходов от арифметических вычислений к битовым (или наоборот) как можно меньше. Такие переходы в [7] названы *чередованиями*. Сокращение числа чередований в MBA-выражениях в рамках задачи деобфускации мотивировано следующим фактом. При деобфускации непрозрачных предикатов часто применяются SMT-решатели (аббревиатура от английского термина Satisfiability Modulo Theories), которые могут проверить, например, тождественную истинность или тождественную ложность выражения. Результаты сравнения трех известных SMT-решателей, проведенного в [8], показали, что чем больше чередований в MBA-выражении, на основе которого построен непрозрачный предикат, тем больше времени требуется SMT-решателю для получения ответа. В частности, при десяти чередованиях время решения одного предиката на основе MBA-выражения достигало 3 000 секунд. Эти результаты также подтверждаются результатами исследования [6], где заключается, что для деобфускации MBA-выражений недостаточно только SMT-решателей. Другими словами, перед применением SMT-решателя следует упрощать MBA-выражение $E(x_1, \dots, x_t)$, так как нахождение эквивалентного ему MBA-выражения $E'(x_1, \dots, x_t)$ с меньшим числом чередований может сократить время работы SMT-решателя. Наибольшего успеха исследователи достигли при деобфускации линейных MBA-выражений (линейная комбинация битовых подвыражений) [4, 8, 9]. Отметим, что эти результаты легли в основу плагина goo MBA для дизассемблера IDA Pro.

В настоящей работе будет показано, как линейные MBA-выражения могут быть упрощены с использованием техники, применяемой при декодировании линейных кодов и дешифровании кодовых криптосистем. В ряде случае эта техника позволяет получить линейные MBA-выражения с меньшим числом чередований, чем при использовании известных техник деобфускации линейных MBA-выражений. Работа, помимо введения и заключения, содержит два раздела. В первом разделе приводится определение MBA-выражения и классификация таких выражений. Здесь же сформулирован и для полноты изложения доказан основной результат из [2], позволяющий построить множество тождественно равных нулю линейных MBA-выражений. В терминах из линейной алгебры дается определение эквивалентных линейных MBA-выражений и строятся рандомизированные алгоритмы генерации линейного MBA-выражения, эквивалентного заданному линейному MBA-выражению. Второй раздел посвящен построению и экспериментальному исследованию новых алгоритмов упрощения линейных MBA-выражений. В этом разделе приведены необходимые сведения о линейных кодах и показано, как связаны задачи декодирования линейных кодов и деобфускации линейных MBA-выражений.

1. Об обфускации MBA-выражениями

Пусть $\mathbb{R}$ — числовое кольцо. Носителем вектора $\mathbf{x} = (x_1, \dots, x_k) \in \mathbb{R}^k$ назовем множество $\operatorname{supp}(\mathbf{x}) = \{i : x_i \neq 0\}$, а весом Хэмминга $\operatorname{wt}(x)$ этого вектора — мощность его носителя: $\operatorname{wt}(x) = |\operatorname{supp}(x)|$.

1.1. MBA-выражения

Пусть $n \in \mathbb{N}$, $\mathbb{V} = \mathbb{Z}_{2^n}$. Для $B = \{0, 1\}$ и $t \in \mathbb{N}$ через $\beta : \mathbb{V} \longrightarrow B^n$ обозначим отображение, ставящее в соответствие целому числу из $\mathbb{V}$ его двоичную запись из B^n, причем для $\mathbf{a} = (a_{n-1}, \dots, a_0) \in B^n$ обратное отображение β^{-1} определим так: $\beta^{-1}(\mathbf{a}) = \sum_{i=0}^{n-1} 2^i a_i$. Под булевой функцией $f(\alpha_1, \dots, \alpha_t)$ от t переменных будем понимать произвольное отображение $f : B^t \longrightarrow B$. Тогда *битовым* выражением от t целочисленных n-битных переменных $x_1, \dots, x_t$, соответствующим булевой функции f,

называется отображение $e_f : \mathbb{V}^t \to \mathbb{V}$ вида

$$e_f(x_1, \dots, x_t) = \beta^{-1}\Big(f(\beta(x_1)_{n-1}, \dots, \beta(x_t)_{n-1}), \dots, f(\beta(x_1)_0, \dots, \beta(x_t)_0)\Big)$$
$$= \sum_{i=0}^{n-1} 2^i f(\beta(x_1)_i, \dots, \beta(x_t)_i), \tag{1}$$

где $\beta(x_j)_i$ — i-ая координата n-битного вектора $\beta(x_j)$. Из (1), в частности, вытекает, что булевой функции $f(\alpha_1, \dots, \alpha_t) \equiv 1$ соответствует битовое выражение $e_f(x_1, \dots, x_t) = \sum_{i=0}^{n-1} 2^i = 2^n - 1 = -1 \in \mathbb{V}$.

Обозначим через $\mathcal{F}_t$ множество всех булевых функций от t переменных. В соответствии с [4], *полиномиальным* MBA-выражением называется функция $E : \mathbb{V}^t \to \mathbb{V}$ от t целочисленных n-битных переменных $x_1, \dots, x_t \in \mathbb{V}$, имеющая вид

$$E(x_1, \dots, x_t) = \sum_{i \in I} a_i \prod_{f \in J_i} e_f(x_1, \dots, x_t), \tag{2}$$

где $I \subset \mathbb{N}$, $J_i \subset \mathcal{F}_t$, $a_i \in \mathbb{V} \setminus \{0\}$, $e_f(x_1, \dots, x_t)$ — битовое выражение. Операции сложения и умножения в (2) выполняются в кольце $\mathbb{V} = \mathbb{Z}_{2^n}$. Полиномиальное MBA-выражение называется *линейным*, если $|J_i| = 1$ для всех $i \in I$ в (2). Далее исследуются только линейные MBA-выражения, поэтому вместо представления (2) для таких выражений будем использовать представление

$$E(x_1, \dots, x_t) = \sum_{f \in J \subset \mathcal{F}_t} a_f e_f(x_1, \dots, x_t), \tag{3}$$

где $a_f \in \mathbb{V} \setminus \{0\}$. Множество J в (3) будем называть носителем выражения $E(x_1, \dots, x_t)$ и обозначать $\mathrm{supp}(E)$. Заметим, что количество чередований в линейном MBA-выражении, в соответствии с определением из [7], равно $|J|$. Для удобства $|J|$ будем называть *длиной* линейного MBA-выражения.

Для $n = 8$ и $t = 3$ примером полиномиального MBA-выражения является

$$E(x_1, x_2, x_3) = 7(x_1 \wedge x_2)(x_2 \vee x_3) - (x_1 \oplus x_2 \oplus x_3) + 120,$$

где $I = \{1, 2, 3\}$, $a_1 = 7$, $a_2 = -1$, $a_3 = -120$, $J_1 = \{f_1(\alpha_1, \alpha_2, \alpha_3) = \alpha_1 \wedge \alpha_2, f_2(\alpha_1, \alpha_2, \alpha_3) = \alpha_2 \vee \alpha_3\}$, $J_2 = \{f(\alpha_1, \alpha_2, \alpha_3) = \alpha_1 \oplus \alpha_2 \oplus \alpha_3\}$, $J_3 = \{f(\alpha_1, \alpha_2, \alpha_3) \equiv 1\}$. Для тех же n и t выражение вида

$$E(x_1, x_2, x_3) = 2(x_2 \vee x_3) + (x_2 \oplus x_3) \tag{4}$$

является линейным, где $J = \mathrm{supp}(E) = \{f_1(\alpha_1, \alpha_2, \alpha_3) = \alpha_2 \vee \alpha_3, f_2(\alpha_1, \alpha_2, \alpha_3) = \alpha_2 \oplus \alpha_3\}$, $a_{f_1} = 2$, $a_{f_2} = 1$.

Пусть $\{b_1, \dots, b_{2^t}\} = B^t$ — произвольным, но фиксированным образом упорядоченное множество двоичных векторов длины t. Так как $|\mathcal{F}_t| = 2^{2^t}$, то множеству $\mathcal{F}_t$ можно поставить в соответствие $2^t \times 2^{2^t}$-матрицу $\mathcal{T}_t$ из нулей и единиц, каждый столбец (столбец истинности) которой представляет собой значения соответствующей ему булевой функции на всех наборах из B^t. Для булевой функции $f \in \mathcal{F}_t$ через $\mathcal{T}_t[f]$ будем обозначать соответствующий столбец истинности. Для $N \geqslant 2^t$ и $F = \{\varphi_1, \dots, \varphi_N\} \subset \mathcal{F}_t$ через A_F обозначим $2^t \times N$-матрицу вида

$$A_F = (\mathcal{T}_t[\varphi_1] | \dots | \mathcal{T}_t[\varphi_N]), \tag{5}$$

элементы которой (0 и 1) рассматриваются как элементы кольца $\mathbb{V}$ и среди столбцов которой найдутся 2^t столбцов, образующих обратимую $2^t \times 2^t$-матрицу. Пусть $E(x_1, \dots, x_t)$ — линейное MBA-выражение вида (3), $J = \mathrm{supp}(E) \subseteq F$. Для $f \in J$ через $j(f)$ обозначим номер столбца $\mathcal{T}_t[f]$ в матрице A_F вида (5). Выражению $E(x_1, \dots, x_t)$ поставим в соответствие вектор $\mathcal{E}(E(x_1, \dots, x_t)) \in \mathbb{V}^N$ вида

$$\mathcal{E}(E(x_1, \dots, x_t)) = (\varepsilon_1, \dots, \varepsilon_N) \in \mathbb{V}^N, \varepsilon_l = \begin{cases} a_f, \text{если } l = j(f), f \in \mathrm{supp}(E), \\ 0, \text{иначе.} \end{cases}$$

Во введенных обозначениях сформулируем и докажем результат, доказанный впервые в [2].

Теорема 1. *Пусть A_F — матрица вида (5), $E(x_1, \dots, x_t)$ — MBA-выражение вида (3), $\mathrm{supp}(E) \subseteq F$. $E(x_1, \dots, x_t) \equiv 0$ тогда и только тогда, когда $\mathbf{y} = \mathcal{E}(E(x_1, \dots, x_t))$ является решением системы*

$$A_F \mathbf{y}^T = \mathbf{0}^T = (0, \dots, 0)^T \in \mathbb{V}^{2^t}. \tag{6}$$

Доказательство. Так как матрица $A_F \in \mathbb{V}^{2^t \times N}$ представима в виде

$$A_F = \begin{pmatrix} \varphi_1(0, \dots, 0) & \dots & \varphi_N(0, \dots, 0) \\ \varphi_1(0, \dots, 1) & \dots & \varphi_N(0, \dots, 1) \\ \vdots & \ddots & \vdots \\ \varphi_1(1, \dots, 1) & \dots & \varphi_N(1, \dots, 1) \end{pmatrix},$$

то

$$A_F \mathbf{y}^T = \begin{pmatrix} \sum_{j=1}^N y_j \varphi_j(0, \dots, 0) \\ \sum_{j=1}^N y_j \varphi_j(0, \dots, 1) \\ \vdots \\ \sum_{j=1}^N y_j \varphi_j(1, \dots, 1) \end{pmatrix}.$$

Следовательно,

$$A_F \mathbf{y}^T = \mathbf{0}^T \iff \forall \boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_t) \in B^t \ : \ \sum_{j=1}^N y_j \varphi_j(\alpha_1, \dots, \alpha_t) = 0. \tag{7}$$

Поэтому, из равенства $A_F \mathbf{y}^T = \mathbf{0}^T$, учитывая (1) и (7), получаем:

$$\begin{aligned} E(x_1, \dots, x_t) &= \sum_{f \in J \subset \mathcal{P}_t} a_f e_f(x_1, \dots, x_t) = \sum_{f \in J} a_f \sum_{i=0}^{n-1} 2^i f(\beta(x_1)_i, \dots, \beta(x_t)_i) \\ &= \sum_{i=0}^{n-1} 2^i \left(\sum_{f \in J} a_f f(\beta(x_1)_i, \dots, \beta(x_t)_i) \right) \\ &= \sum_{i=0}^{n-1} 2^i \left(\sum_{j=1}^N y_j \varphi_j(\beta(x_1)_i, \dots, \beta(x_t)_i) \right) = 0 \in \mathbb{V}. \end{aligned}$$

□

1.2. О генерации эквивалентных линейных MBA-выражений

Опишем множество эквивалентных линейных MBA-выражений, используя понятия линейной алгебры. Для матрицы (5) определим группу по сложению $C_F = \{\mathbf{y} \in \mathbb{V}^N \ : \ A_F \mathbf{y}^T = \mathbf{0}^T\}$, которая является также подмодулем $\mathbb{V}$-модуля $\mathbb{V}^N$. По каждому вектору $\mathbf{y} = (y_1, \dots, y_N) \in C_F$ может быть построено тождественно равное нулю линейное MBA-выражение вида

$$\mathcal{E}^{-1}(\mathbf{y}) = \sum_{i=1}^N y_i e_{\varphi_i}(x_1, \dots, x_t) = \sum_{i \in \mathrm{supp}(\mathbf{y})} y_i e_{\varphi_i}(x_1, \dots, x_t) \equiv 0 \in \mathbb{V}. \tag{8}$$

Отсюда, в частности, вытекает, что если для $i \in \mathrm{supp}(\mathbf{y})$ элемент y_i обратим в кольце $\mathbb{V}$, то

$$e_{\varphi_i}(x_1, \dots, x_t) = -y_i^{-1} \sum_{i \in \mathrm{supp}(\mathbf{y}) \setminus \{i\}} y_i e_{\varphi_i}(x_1, \dots, x_t). \tag{9}$$

Следовательно, левая часть равенства (9) (MBA-выражение без чередований) может быть *обфусцирована* с помощью правой части этого равенства (с помощью MBA-выражения, содержащего чередования).

По группе C_F построим фактор-множество

$$\mathbb{V}^N/C_F = \left\{ C(\mathbf{b}) = C_F + \mathbf{a} \,:\, \mathbf{a} \in \mathbb{V}^N, A_F\mathbf{a}^T = \mathbf{b}^T \in \mathbb{V}^{2^t} \right\}, |\mathbb{V}^N/C_F| = |\mathbb{V}|^{2^t}.$$

Из определения фактор-классов $C(\mathbf{b})$ вытекает, для любого $\mathbf{b} \in \mathbb{V}^{2^t}$ и любых $\mathbf{y}, \mathbf{y}' \in C(\mathbf{b})$ линейные МВА-выражения $\mathcal{E}^{-1}(\mathbf{y})$ и $\mathcal{E}^{-1}(\mathbf{y}')$ являются эквивалентными:

$$\mathcal{E}^{-1}(\mathbf{y}) = \sum_{i \in \mathrm{supp}(\mathbf{y})} y_i e_{\varphi_i}(x_1, \dots, x_t) = \sum_{j \in \mathrm{supp}(\mathbf{y}')} y'_j e_{\varphi_j}(x_1, \dots, x_t) = \mathcal{E}^{-1}(\mathbf{y}'). \tag{10}$$

Также получаем, что для заданного МВА-выражения в рамках заданной матрицы A_F существует $|C_F|$ эквивалентных выражений. Отметим, что в [8] вектор $A_F\mathbf{y}^T$ назван *подписью* МВА-выражения $E(x_1, \dots, x_t)$, где $\mathbf{y} = \mathcal{E}(E(x_1, \dots, x_t))$, и впервые доказано, что два линейных МВА-выражения эквивалентны, если равны их подписи.

К правой и левой частям равенства (10) могут быть прибавлены тождественно равные нулю МВА-выражения (не обязательно одинаковые для левой и правой части), при этом равенство сохранится. Алгоритм LinMBAGen (см. алгоритм 1) позволяет по заданному линейному МВА-выражению $E(x_1, \dots, x_t)$ найти эквивалентное ему линейное МВА-выражение заданной длины w.

Algorithm 1. LinMBAGen

Data: $E(x_1, \dots, x_t)$, $w \in [1, \dots, N]$
Result: $E'(x_1, \dots, x_t) \equiv E(x_1, \dots, x_t)$, $\mathrm{wt}(\mathcal{E}(E'(x_1, \dots, x_t))) \geqslant w$
/* Случайно выбрать тождественно равное нулю МВА-выражение */
1 Случайно выбрать $\mathbf{y} \in C_F$
/* Проверить условие на вес вектора */
2 **if** $\mathrm{wt}(\mathcal{E}(E(x_1, \dots, x_t)) + \mathbf{y}) \geqslant w$ **then**
3 **return** $\mathcal{E}^{-1}(\mathcal{E}(E(x_1, \dots, x_t)) + \mathbf{y})$
4 **else**
5 **go to** 1

Возвращаемое алгоритмом LinMBAGen выражение не содержит слагаемых с одинаковым битовым выражением. Поэтому, для того, чтобы длина выражения была как можно больше, матрица A_F должна иметь как можно больше разных столбцов. Так как для t переменных существует 2^{2^t} возможных булевых функций, то максимальная длина линейного МВА-выражения равна 2^{2^t}. Если исключить требование на неповторяемость битовых выражений, то с помощью алгоритма LinMBAGenAnyLen (см. алгоритм 2) может быть сгенерировано линейное МВА-выражение любой длины.

2. О деобфускации линейных МВА-выражений

В настоящей работе для деобфускации линейных МВА-выражений применяется техника, используемая при декодировании линейных кодов и дешифровании кодовых криптосистем. Поэтому сначала приведем необходимые сведения о линейных кодах.

2.1. Линейные коды

Напомним, что линейным $[m, k, d]_q$-кодом C над конечным полем $\mathbb{F}_q$ называется любое подпространство C размерности k пространства $\mathbb{F}_q^m$, для которого минимальный вес Хэмминга ненулевых векторов кода C равен d. Величина d называется *кодовым расстоянием* кода C. Из любых k линейно независимых векторов подпространства C может быть построена $k \times m$-матрица, называемая *порождающей* матрицей кода C. Если G — порождающая матрица кода C, то любая $(m - k) \times m$-матрица H

Algorithm 2. LinMBAGenAnyLen

Data: $E(x_1, \dots, x_t)$, $w \in \mathbb{N}$
Result: $E'(x_1, \dots, x_t) \equiv E(x_1, \dots, x_t)$, число слагаемых не менее w
1 $E'(x_1, \dots, x_t) = E(x_1, \dots, x_t)$
/* Случайно выбрать тождественно равное нулю MBA-выражение */
2 Случайно выбрать $\mathbf{y} \in C_F$
/* Проверить условие на количество слагаемых */
3 **if** $\mathrm{wt}(\mathcal{E}(E'(x_1, \dots, x_t))) + \mathrm{wt}(\mathbf{y}) \geqslant w$ **then**
4 **return** $E'(x_1, \dots, x_t) + \mathcal{E}^{-1}(\mathbf{y})$
5 **else**
6 $E'(x_1, \dots, x_t) = E'(x_1, \dots, x_t) + \mathcal{E}^{-1}(\mathbf{y})$
7 **go to** 2

ранга $m-k$, такая, что $GH^T = O$, называется *проверочной* матрицей кода C, где O — нулевая $k \times (m-k)$-матрица. Для любого вектора $\mathbf{z} \in \mathbb{F}_q^m$ вектор $\mathbf{s} = \mathbf{z}H^T$ называется *синдромом* вектора $\mathbf{z}$ относительно матрицы H. Для заданной проверочной матрицы H кода C, синдрома $\mathbf{s}$ и числа $t = \lceil (d-1)/2 \rceil$ задача нахождения вектора $\mathbf{z} \in \mathbb{F}_q^N$, такого, что $\mathrm{wt}(\mathbf{z}) \leqslant t$, называется задачей синдромного декодирования. Известно, что эта задача для случайно выбранного кода C является NP-полной [10].

2.2. Алгоритмы деобфускации линейных MBA-выражений

Предлагаемые в [8] и [4] способы упрощения линейного MBA-выражения $E(x_1, \dots, x_t)$ основаны на том, что матрица A_F содержит 2^t линейно независимых столбцов, поэтому подпись $A_F(\mathcal{E}(E(x_1, \dots, x_t)))^T$ может быть представлена как линейная комбинация не более чем 2^t столбцов матрицы A_F. Следовательно, любое линейное MBA-выражение может быть упрощено до линейного MBA-выражения длины не более 2^t. Вопрос о том, может ли быть получено MBA-выражение строго меньшей длины, чем 2^t, в работах [8] и [4] не изучался. Заметим, что отличительной особенностью работы [4] по сравнению с [8] является упрощение вычисления подписи. Именно в [4] показано, что для $E(x_1, \dots, x_t)$ можно напрямую (без разбора на составляющие битовые подвыражения) вычислить вектор подписи $\mathbf{s} = A_F\mathbf{y}^T$ на 2^t наборах значений t переменных (наименее значащие биты в этих переменных принимают все возможные значения, а все остальные биты имеют нулевое значение). В настоящей работе будет показано, что в ряде случаев можно получить упрощенное MBA-выражение с количеством слагаемых, существенно меньшим 2^t.

Задачу деобфускации линейного выражения $E(x_1, \dots, x_t)$ вида (3), как задачу поиска эквивалентного выражения с меньшим числом чередований (меньшей длиной), можно сформулировать следующим образом: найти такой вектор $\mathbf{z} \in \mathbb{V}^N$, чтобы 1) $A_F\mathbf{y}^T = A_F\mathbf{z}^T$ и 2) $\mathrm{wt}(\mathbf{z}) < \mathrm{wt}(\mathbf{y})$ для $\mathbf{y} = \mathcal{E}(E(x_1, \dots, x_t))$. Первое условие гарантирует, что MBA-выражения являются эквивалентными, а второе означает меньшую длину MBA-выражения $\mathcal{E}^{-1}(\mathbf{z})$. Вектор $\mathbf{s} = A_F\mathbf{y}^T$ является аналогом синдрома вектора $\mathbf{y}$ относительно матрицы A_F. Отличием является то, что элементами матрицы A_F являются элементы кольца $\mathbb{V}$, в общем случае не являющегося конечным полем. Поэтому вектор $\mathbf{s}$ будем называть $\mathbb{V}$-*синдромом* вектора $\mathbf{y}$ относительно матрицы A_F. Задачу упрощения линейного MBA-выражения $E(x_1, \dots, x_t)$ сформулируем следующим образом.

Задача 1. *Для заданной $2^t \times N$-матрицы A_F и заданного линейного MBA-выражения $E(x_1, \dots, x_t)$ с $\mathbb{V}$-синдромом $\mathbf{s} = A_F \times \left(\mathcal{E}(E(x_1, \dots, x_t))\right)^T$ найти вектор $\mathbf{z} \in C_F(\mathbf{s})$ минимального веса Хэмминга.*

Рассмотрим линейное MBA-выражение $E(x_1, \dots, x_t)$ и соответствующий вектор $\mathbf{y} = \mathcal{E}(E(x_1, \dots, x_t))$. Предположим, что существует такой вектор $\mathbf{z}$ минимального веса $\mathrm{wt}(\mathbf{z}) = r$, что $A_F\mathbf{y}^T = A_F\mathbf{z}^T$. Поиск такого вектора перебором имеет сложность $O((|\mathbb{V}|-1)^r \binom{N}{r})$. И даже для небольших значений r такой

перебор может потребовать значительных вычислительных ресурсов. Например, при $|\mathbb{V}| = 2^{32}$, $N = 15$ и $r = 3$ сложность поиска будет не менее 2^{104}.

В настоящей работе предлагается способ упрощения, закаляющийся в применении метода *декодирования по информационным совокупностям* (Information Set Decoding) [11]. Для описания этого метода нам потребуется следующее утверждение.

Утверждение 1. *Пусть $N \geqslant 2^t$, S — обратимая над $\mathbb{V}$ квадратная $2^t \times 2^t$-матрица, Q — перестановочная $N \times N$-матрица, причем $S \cdot A_F \cdot Q = [E_{2^t}|P]$, где E_{2^t} — единичная $2^t \times 2^t$-матрица. Тогда $\mathcal{L}(G) = \mathcal{C}_F$, где*

$$G = [-P^T|E_{N-2^t}]Q^T, \tag{11}$$

$\mathcal{L}(G)$ — линейная оболочка, натянутая на строки матрицы G.

Доказательство. Из равенства

$$SA_FG^T = SA_FQ[-P^T|E_{N-2^t}]^T = [E_{2^t}|P] \times [-P^T|E_{N-2^t}]^T = [-P + P] = O$$

вытекает, что $\mathcal{L}(G)$ содержится во множестве решений системы $SA_F\mathbf{y}^T = \mathbf{0}^T$, которое, в силу обратимости S, совпадает с $\mathcal{C}_F$. Следовательно, $\mathcal{L}(G) \subseteq \mathcal{C}_F$. Система уравнений $[E_{2^t}|P]\mathbf{y}^T = \mathbf{0}^T \in \mathbb{V}^{2^t}$ имеет $N - 2^t$ независимых неизвестных и 2^t зависимых. Следовательно эта система имеет $|\mathbb{V}^{N-2^t}|$ решений. Заметим, что матрица G порождает множество также из $\mathbb{V}^{N-2^t}$ векторов. Поэтому $\mathcal{L}(G) = \mathcal{C}_F$. □

Для $\mathcal{C}_F$ *информационной совокупностью* назовем множество $\tau \subset \{1, \dots, N\}$, $|\tau| = N - 2^t$, для которого столбцы матрицы G, такой, что $\mathcal{L}(G) = \mathcal{C}_F$ (например, вида (11)) с номерами из τ образуют квадратную обратимую матрицу над кольцом $\mathbb{V}$. Матрицу, образованную столбцами матрицы G с номерами из τ, будем обозначать G_τ. Вектор $\mathbf{y} = \mathcal{E}(E(x_1, \dots, x_t))$ может быть представлен в виде

$$\mathbf{y} = \mathbf{c} + \mathbf{z}, \mathrm{wt}(\mathbf{z}) = r, \tag{12}$$

где $\mathbf{c} = \mathbf{m}G \in \mathcal{C}_F$, так как очевидно, что

$$A_F\mathbf{y}^T = A_F(\mathbf{c} + \mathbf{z})^T = A_F\mathbf{c}^T + A_F\mathbf{z}^T = A_F\mathbf{z}^T.$$

Пусть τ — информационная совокупность, причем $\tau \cap \mathrm{supp}(\mathbf{z}) = \varnothing$. Следовательно, $\mathbf{y}_\tau = \mathbf{c}_\tau = \mathbf{m}G_\tau$. Откуда $\mathbf{z} = \mathbf{y} - \mathbf{c}_\tau G_\tau^{-1}G = \mathbf{y} - \mathbf{y}_\tau G_\tau^{-1}G$. Поэтому задача нахождения вектора $\mathbf{z}$ веса r (задача 1) может быть сведена к нахождению информационной совокупности, не пересекающейся с носителем вектора $\mathbf{z}$. Этот подход и его модификации (см., например, [12]) используются в теории кодирования для исправления ошибок при помощи кодов умеренной длины, если не известны более эффективные алгоритмы; а в теории асимметричных кодовых криптосистем декодирование по информационным совокупностям является наиболее эффективным способом дешифрования сообщений без знания секретного ключа [13]. Отличительной особенностью рассматриваемого здесь алгоритма декодирования по информационным совокупностям является лишь то, что матрица G определена над коммутативным кольцом $\mathbb{V}$, а не над конечным полем $\mathbb{F}_q$. Способ нахождения вектора веса не более r за заданное количество итераций I реализован в виде алгоритма ISD (см. алгоритм 3). Сложность этого алгоритма составляет

$$O\left(\min\left\{\binom{N}{N-2^t} \cdot \binom{N-r}{N-2^t}^{-1}, I\right\} \cdot (N-2^t)^3\right) \tag{13}$$

и соответствует среднему количеству попыток нахождения информационной совокупности, не пересекающейся с носителем вектора $\mathbf{z}$ в представлении (12), умноженному на $(N - 2^t)^3$ — сложность

Algorithm 3. ISD

Data: $\mathbf{y} \in \mathbb{V}^N$, $G \in \mathbb{V}^{(N-2^t)\times N}$, $r, I \in \mathbb{N}$
Result: ($\mathbf{z} \in \mathbb{V}^N$, $\mathbf{y} - \mathbf{z} \in \mathcal{L}(G)$, $\mathrm{wt}(\mathbf{z}) \leqslant r$) или $\perp$

```
1  i = 0
2  i = i + 1
   /* проверить счетчик максимального числа итераций алгоритма; счетчик используется
      для предотвращения зацикливания                                              */
3  if i = I then
4      return ⊥
5  выбрать τ ⊂ {1, ..., N}, |τ| = N − 2^t
   /* проверить, что τ является информационной совокупностью: τ — информационная
      совокупность, если элемент det(G_τ) обратим в V                              */
6  if det(G_τ) необратим в V then
7      go to 2
8  найти вектор z = y − y_τ G_τ^{-1} G
   /* проверить условие на вес найденного вектора                                  */
9  if wt(z) ⩽ r then
10     return z
11 else
12     go to 2
```

нахождения обратной матрицы G_τ^{-1}. Например, при $|\mathbb{V}| = 2^{32}$, $N = 15$, $r = 3$, $t = 2$ сложность составляет порядка $\min\{3, I\} \cdot 11^3$.

Заметим, что в задачах криптографии и теории кодирования, где используется некоторый $[m, k, d]_q$-код, входной вектор $\mathbf{y}$ алгоритма ISD обычно такой, что вектор минимального веса с таким же синдромом единственен. Поэтому для корректного декодирования или расшифрования сообщения применяется алгоритм ISD, где $r = \lceil (d-1)/2 \rceil$. Сложность (13) алгоритма ISD не является полиномиальной и пока нет оснований предполагать, что задача 1 менее сложна, чем задача синдромного декодирования для линейного кода. Однако, при упрощении линейного MBA-выражения $E(x_1, \ldots, x_t)$ успешным упрощением можно считать не только случай решения задачи 1, но и когда в $\mathcal{C}(\mathbf{s})$ найден такой вектор $\mathbf{z}$, что $\mathrm{wt}(\mathbf{z}) < \mathrm{wt}(\mathcal{E}(E(x_1, \ldots, x_t)))$. Другими словами, под успешным упрощением можно понимать сокращение числа чередований в MBA-выражении. Поэтому задача упрощения линейного MBA-выражения может быть сформулирована так.

Задача 2. *Для заданной $2^t \times N$-матрицы A_F, заданного линейного MBA-выражения $E(x_1, \ldots, x_t)$ с $\mathbb{V}$-синдромом $\mathbf{s} = \mathcal{E}(E(x_1, \ldots, x_t)) \times A_F^T$ и заданного $I \in \mathbb{N}$ найти за I итераций такой вектор $\mathbf{z} \in C_F(\mathbf{s})$, что* $\mathrm{wt}(\mathbf{z}) < \mathrm{wt}(\mathcal{E}(E(x_1, \ldots, x_t)))$.

Алгоритм ISDMinSearch (см. алгоритм 4), предназначенный для решения задачи 2, возвращает вектор $\mathbf{z}$ с наименьшим весом среди I опробованных. Сложность алгоритма ISDMinSearch составляет

$$O\left(I \cdot (N - 2^t)^3\right). \tag{14}$$

Отметим, что алгоритмы ISD и ISDMinSearch могут быть распараллелены, так опробование разных наборов τ может выполняться независимо. Алгоритм MBASimplify (см. алгоритм 5) возвращает по линейному MBA-выражению $E(x_1, \ldots, x_t)$ эквивалентное ему линейное MBA-выражение меньшей длины и возвращает $E(x_1, \ldots, x_t)$, если при заданных входных параметрах не нашлось MBA-

Algorithm 4. ISDMinSearch

Data: $\mathbf{y} \in \mathbb{V}^N$, $G \in \mathbb{V}^{(N-2^t)\times N}$, $I \in \mathbb{N}$
Result: $\mathbf{z} \in \mathbb{V}^N$, $\mathbf{y} - \mathbf{z} \in \mathcal{L}(G)$, $\mathrm{wt}(\mathbf{z}) \longrightarrow \min$

1 $i = 0$, $mw = \mathrm{wt}(\mathbf{y})$, $\mathbf{z} = \mathbf{y}$
2 $i = i + 1$
/* проверить счетчик максимального числа итераций алгоритма */
3 **if** $i = I$ **then**
4 вернуть $\mathbf{z}$
5 выбрать $\tau \subset \{1, \dots, N\}$, $|\tau| = N - 2^t$
/* проверить, что τ является информационной совокупностью */
6 **if** $\det(G_\tau)$ *необратим в* $\mathbb{V}$ **then**
7 **go to** 2
8 найти вектор $\mathbf{x} = \mathbf{y} - \mathbf{y}_\tau G_\tau^{-1} G$
9 **if** $\mathrm{wt}(\mathbf{x}) \leqslant mw$ **then**
10 $mw = \mathrm{wt}(\mathbf{x})$, $\mathbf{z} = \mathbf{x}$
11 **else**
12 **go to** 2

выражения меньшей длины. Заметим, что если в линейном MBA-выражении $E(x_1, \dots, x_t)$ не более g слагаемых вида $a_f e_f(x_1, \dots, x_t)$, то шаг 1 алгоритма 5 имеет вычислительную сложность не более $O(g2^t)$, так как по каждому слагаемому $a_f e_f(x_1, \dots, x_t)$ прямыми вычислениями битового выражения $e_f(x_1, \dots, x_t)$ на всех 2^t наборах из B^t может быть построен столбец истинности $\mathcal{T}_t[f]$. В алгоритме MBASimplify предполагается, что выражение $E(x_1, \dots, x_t)$ не содержит слагаемых с одинаковыми битовыми выражениями вида $e_f(x_1, \dots, x_t)$. Если же такие имеются, то перед выполнением алгоритма следует привести подобные слагаемые. Так как матрица A_F строится рандомизированно, то алгоритм MBASimplify может быть выполнен несколько раз, что позволит выбрать наилучшее решение. Другим способом поиска наилучшего решения является увеличение параметров I и R, однако увеличение R может замедлить работу алгоритма ISDMinSearch, так как в этом случае растут размеры матрицы G_τ, для которой в алгоритме ISDMinSearch находится обратная матрица.

2.3. Экспериментальная оценка ISD и ISDMinSearch

В работе проведено экспериментальное исследование алгоритмов ISD и ISDMinSearch. В соответствии с (13) и (14), вычислительная сложность этих алгоритмов зависит от t — количества переменных в линейном MBA-выражении, N — количества столбцов в матрице A_F, I — количества итераций поиска. Сложность ISD также зависит от r — максимального веса искомого вектора $\mathbf{z}$ в (12). Поэтому параметры N, t, I и r выбраны так, чтобы эксперименты могли быть проведены в среде Sage Math 9.2 на компьютере x64 AMD FX(tm)-8350 Eight-Core Processor 4.00 GHz с восьмиядерным процессором и 16ГБ оперативной памяти. В качестве кольца $\mathbb{V}$ были рассмотрены $\mathbb{Z}_2$, $\mathbb{Z}_{2^2}$, $\mathbb{Z}_{2^4}$, $\mathbb{Z}_{2^8}$, $\mathbb{Z}_{2^{16}}$ и $\mathbb{Z}_{2^{32}}$. Заметим, что кольца $\mathbb{Z}_{2^8}$, $\mathbb{Z}_{2^{16}}$ и $\mathbb{Z}_{2^{32}}$ соответствуют типам unsigned char, unsigned short и unsigned int языков программирования C/C++, поэтому результаты для этих колец могут быть использованы, например, при обфускации или деобфускации программ на этих языках. Для случая $t = 2$ количество всех булевых функций от двух переменных равно $2^{2^2} = 16$. Поэтому в качестве матрицы A_F взята (4×15)-матрица, состоящая из всех ненулевых столбцов истинности таблицы $\mathcal{T}_2$, так как такие размеры матрицы позволяют провести вычислительный эксперимент на выбранном компьютере за приемлемое время.

Algorithm 5. MBASimplify

Data: $E(x_1, \dots, x_t)$, I, R
Result: $E'(x_1, \dots, x_t) \equiv E(x_1, \dots, x_t)$
/* Построить множество булевых функций f, соответствующих $e_f(x_1, \dots, x_t)$ в (3) */
По $E(x_1, \dots, x_t)$ построить $F \subset \mathcal{F}_t$
/* Построить матрицу A_F^0 */
for $f \in F$ **do**
    Построить столбец $\mathcal{T}_t[f]$
Из столбцов $\mathcal{T}_t[f]$, $f \in F$, построить матрицу A_F^0
/* Случайно достроить матрицу A_F^0 до A_F */
Выбрать случайно R разных $(0,1)$-столбцов, не содержащихся в A_F^0, и приписать их к матрице A_F^0, образовав $(2^t \times N)$-матрицу A_F
/* Подготовить входные данные для ISDMinSearch */
По матрице A_F построить $(N - 2^t) \times N$-матрицу G (см. утверждение 1)
$\mathbf{y} = \mathbf{0} \in \mathbb{V}^N$
for $f \in F$ **do**
    /* $\mathcal{E}(e_f(x_1, \dots, x_t))$ — вектор веса 1, в котором координата равна 1 в позиции, соответствующей столбцу $\mathcal{T}_t[f]$ в матрице A_F */
    $\mathbf{y} = \mathbf{y} + a_f \mathcal{E}(e_f(x_1, \dots, x_t))$
$\mathbf{z} = \mathsf{ISDMinSearch}(\mathbf{y}, G, I)$
return $\mathcal{E}^{-1}(\mathbf{z})$

2.3.1. Исследование алгоритма ISD

В рамках анализа ISD оценена вероятность нахождения вектора минимального веса при $t = 2, 3, 4$ и $N \approx 2^{t+2}$, а в случае $t = 3$ эта вероятность оценена в зависимости от $N \in \{16, 32, 48, 64\}$. Во всех экспериментах для $2^t \times N$-матрицы A_F полного ранга вида (5), случайно выбранной из кольца $\mathbb{Z}_n^{2^t \times N}$, используя утверждение 1, сначала находилась $(N - 2^t) \times N$-матрица G полного ранга, для которой $GA_F^T = O$. Далее из линейной оболочки $\mathcal{L}(G)$ было выбрано случайным образом сто векторов, для каждого из которых случайно и независимо от других векторов генерировался и прибавлялся вектор веса $r \in \{1, \dots, 2^t\}$. Фактически, для ста случайно сгенерированных линейных MBA-выражений с r слагаемыми был выполнен алгоритм LinMBAGen с параметром $w = 0$ (для каждого входного MBA-выражения шаг 1 алгоритма выполнялся только один раз). Для каждого из полученных векторов был выполнен алгоритм ISD с целью поиска вектора с таким же синдромом и веса не более r. Результаты экспериментов показаны в таблицах 1–6, где δ — доля входных векторов $\mathbf{y}$, для которых был найден вектор с заданным ограничением на вес, $\mathrm{wt}(\mathbf{y})$ — средний вес входного вектора, $\mathrm{wt}(\mathbf{z})$ — средний вес первого найденного алгоритмом ISD вектора, удовлетворяющего условию на вес (в эксперименте, если вектор с заданным условием за I итераций не был найден, то возвращался вектор с наименьшим найденным весом). В таблицах 1, 3 и 6 представлены результаты эксперимента, целью которого было исследование вероятности нахождения вектора минимального веса при $N/2^t \approx \mathrm{const}$ и $t = 2, 3, 4$, а в таблицах 2, 3, 4 и 5 представлены результаты исследования этой вероятности при $t = \mathrm{const}(t = 3)$ и $N = 16, 32, 48, 64$.

Из таблиц 1, 3 и 6 видно, что если вес r вектора, соответствующего MBA-выражению $E(x_1, \dots, x_t)$, принадлежит множеству $\{1, 2, 3, 2^t\}$, то с близкой к единице вероятностью алгоритмом ISD, а значит, и алгоритмом ISDMinSearch, по обфусцированному MBA-выражению LinMBAGen$(E(x_1, \dots, x_t), w)$ или LinMBAGenAnyLen$(E(x_1, \dots, x_t), w)$ для любого $w \in \mathbb{N}$ будет найден вектор веса не более r, соответствующий MBA-выражению $E'(x_1, \dots, x_t)$, тождественно равному $E(x_1, \dots, x_t)$. Для других значений r

Table 1. Finding the minimum weight vector by the algorithm ISD for $\mathbb{V} \in \{\mathbb{Z}_2, \mathbb{Z}_{2^2}, \mathbb{Z}_{2^4}, \mathbb{Z}_{2^8}, \mathbb{Z}_{2^{16}}, \mathbb{Z}_{2^{32}}\}$ when $t = 2$, $N = 2^{2^t} - 1 = 15$, $r \in \{1, \dots, 4\}$, the number of experiments for the given weight r — 100.

Таблица 1. Поиск вектора минимального веса алгоритмом ISD для $\mathbb{V} \in \{\mathbb{Z}_2, \mathbb{Z}_{2^2}, \mathbb{Z}_{2^4}, \mathbb{Z}_{2^8}, \mathbb{Z}_{2^{16}}, \mathbb{Z}_{2^{32}}\}$ при $t = 2$, $N = 2^{2^t} - 1 = 15$, $r \in \{1, \dots, 4\}$, количество экспериментов для заданного веса r — 100.

r	$\mathbb{Z}_2$			$\mathbb{Z}_{2^2}$			$\mathbb{Z}_{2^4}$		
	δ	wt(y)	wt(z)	δ	wt(y)	wt(z)	δ	wt(y)	wt(z)
1	1.0	7.50	0.45	1.0	11.11	0.72	1.0	14.16	0.90
2	**1.0**	**7.16**	**1.13**	**1.0**	**11.21**	**1.68**	**1.0**	**14.14**	**1.90**
3	1.0	7.39	1.71	1.0	11.15	2.43	1.0	14.20	2.94
4	1.0	7.65	1.99	1.0	11.12	3.00	1.0	14.14	3.76

r	$\mathbb{Z}_{2^8}$			$\mathbb{Z}_{2^{16}}$			$\mathbb{Z}_{2^{32}}$		
	δ	wt(y)	wt(z)	δ	wt(y)	wt(z)	δ	wt(y)	wt(z)
1	1.0	14.94	1.00	1.0	14.99	1.00	1.0	15.0	1.00
2	**1.0**	**14.94**	**2.00**	**1.0**	**14.99**	**2.00**	**1.0**	**15.0**	**2.00**
3	1.0	14.96	2.96	1.0	14.99	2.97	1.0	15.0	3.00
4	1.0	14.93	3.88	1.0	14.99	3.94	1.0	15.0	3.94

Table 2. Finding the minimum weight vector by the algorithm ISD for $\mathbb{V} \in \{\mathbb{Z}_2, \mathbb{Z}_{2^2}, \mathbb{Z}_{2^4}, \mathbb{Z}_{2^8}, \mathbb{Z}_{2^{16}}, \mathbb{Z}_{2^{32}}\}$ when $t = 3$, $N = 16$, $r \in \{1, \dots, 8\}$, the number of experiments for the given weight r — 100.

Таблица 2. Поиск вектора минимального веса алгоритмом ISD для $\mathbb{V} \in \{\mathbb{Z}_2, \mathbb{Z}_{2^2}, \mathbb{Z}_{2^4}, \mathbb{Z}_{2^8}, \mathbb{Z}_{2^{16}}, \mathbb{Z}_{2^{32}}\}$ при $t = 3$, $N = 16$, $r \in \{1, \dots, 8\}$, количество экспериментов для заданного веса r — 100.

r	$\mathbb{Z}_2$			$\mathbb{Z}_{2^2}$			$\mathbb{Z}_{2^4}$		
	δ	wt(y)	wt(z)	δ	wt(y)	wt(z)	δ	wt(y)	wt(z)
1	1.0	8.07	0.49	1.0	11.98	0.81	1.00	14.94	0.86
2	1.0	8.01	1.03	1.0	11.88	1.58	1.00	15.04	1.76
3	1.0	8.17	1.63	1.0	12.12	2.43	1.00	14.96	2.71
4	1.0	7.82	2.44	1.0	11.81	3.35	0.99	14.96	3.64
5	1.0	8.01	3.08	1.0	11.99	4.12	0.94	14.94	4.76
6	1.0	8.25	3.37	1.0	12.19	5.00	1.00	14.91	5.55
7	1.0	8.09	3.87	1.0	11.74	5.66	1.00	14.81	6.56
8	1.0	8.00	3.72	1.0	12.02	5.82	1.00	14.95	7.39

r	$\mathbb{Z}_{2^8}$			$\mathbb{Z}_{2^{16}}$			$\mathbb{Z}_{2^{32}}$		
	δ	wt(y)	wt(z)	δ	wt(y)	wt(z)	δ	wt(y)	wt(z)
1	1.00	15.93	1.00	1.00	16.0	1.00	1.00	16.0	1.00
2	1.00	15.93	1.98	1.00	16.0	2.00	1.00	16.0	2.00
3	1.00	15.95	2.96	1.00	16.0	3.00	0.99	16.0	3.04
4	1.00	15.93	3.96	0.97	16.0	4.07	0.96	16.0	4.14
5	1.00	15.94	4.97	0.93	16.0	5.19	0.93	16.0	5.21
6	0.97	15.93	5.95	0.80	16.0	6.32	0.77	16.0	6.45
7	1.00	15.98	6.98	0.87	16.0	7.10	0.73	16.0	7.25
8	1.00	15.94	7.91	1.00	16.0	7.92	1.00	16.0	7.99

вероятность успеха значительно зависит от кольца. Например, при $t = 2$ и $\mathbb{V} = \mathbb{Z}_2$ алгоритм ISD почти всегда находит вектор с заданным ограничением на вес. Но уже для кольца $\mathbb{V} = \mathbb{Z}_{2^2}$ при $r = 5$ и $r = 6$ алгоритм не находит векторы веса с заданным ограничением на вес с вероятностями соответственно 0.21 и 0.31. Для колец, начиная с $\mathbb{Z}_{2^4}$, имеются значения r, для которых алгоритм ни для одного входного вектора (при заданных ограничениях на количество итераций) не нашел вектор веса не более r. Также видно, что для каждого r с ростом мощности кольца растет и средний

Table 3. Finding the minimum weight vector by the algorithm ISD for $\mathbb{V} \in \{\mathbb{Z}_2, \mathbb{Z}_{2^2}, \mathbb{Z}_{2^4}, \mathbb{Z}_{2^8}, \mathbb{Z}_{2^{16}}, \mathbb{Z}_{2^{32}}\}$ when $t = 3$, $N = 32$, $r \in \{1, \ldots, 8\}$, the number of experiments for the given weight r — 100.

Таблица 3. Поиск вектора минимального веса алгоритмом ISD для $\mathbb{V} \in \{\mathbb{Z}_2, \mathbb{Z}_{2^2}, \mathbb{Z}_{2^4}, \mathbb{Z}_{2^8}, \mathbb{Z}_{2^{16}}, \mathbb{Z}_{2^{32}}\}$ при $t = 3$, $N = 32$, $r \in \{1, \ldots, 8\}$, количество экспериментов для заданного веса r — 100.

r	$\mathbb{Z}_2$			$\mathbb{Z}_{2^2}$			$\mathbb{Z}_{2^4}$		
	δ	wt(y)	wt(z)	δ	wt(y)	wt(z)	δ	wt(y)	wt(z)
1	1.0	16.30	0.51	1.00	24.11	0.82	1.00	29.93	0.93
2	1.0	16.16	1.06	1.00	24.09	1.56	1.00	29.85	1.88
3	1.0	16.14	1.97	0.99	23.97	2.49	0.98	29.92	2.87
4	1.0	16.28	2.69	1.00	24.06	3.54	0.93	29.89	4.04
5	1.0	16.40	3.31	1.00	23.90	4.53	0.99	29.90	4.77
6	1.0	16.02	4.03	1.00	23.75	5.31	1.00	29.93	5.80
7	1.0	16.38	3.85	1.00	23.84	5.84	1.00	29.78	6.64
8	1.0	16.40	3.79	1.00	24.29	6.16	1.00	29.93	7.52

r	$\mathbb{Z}_{2^8}$			$\mathbb{Z}_{2^{16}}$			$\mathbb{Z}_{2^{32}}$		
	δ	wt(y)	wt(z)	δ	wt(y)	wt(z)	δ	wt(y)	wt(z)
1	1.00	31.80	1.00	1.00	32.0	1.00	1.00	32.0	1.00
2	1.00	31.80	2.00	1.00	32.0	1.99	1.00	32.0	1.98
3	0.99	31.78	3.02	0.97	32.0	3.11	0.97	32.0	3.15
4	0.67	31.79	5.20	0.70	32.0	5.12	0.63	32.0	5.39
5	0.36	31.81	6.86	0.42	32.0	6.69	0.18	32.0	7.42
6	0.52	31.84	6.91	0.45	32.0	7.05	0.28	32.0	7.39
7	0.99	31.86	6.99	0.63	32.0	7.35	0.61	32.0	7.37
8	1.00	31.82	7.95	1.00	32.0	7.98	1.00	32.0	7.99

Table 4. Finding the minimum weight vector by the algorithm ISD for $\mathbb{V} \in \{\mathbb{Z}_2, \mathbb{Z}_{2^2}, \mathbb{Z}_{2^4}, \mathbb{Z}_{2^8}, \mathbb{Z}_{2^{16}}, \mathbb{Z}_{2^{32}}\}$ when $t = 3$, $N = 48$, $r \in \{1, \ldots, 8\}$, the number of experiments for the given weight r — 100.

Таблица 4. Поиск вектора минимального веса алгоритмом ISD для $\mathbb{V} \in \{\mathbb{Z}_2, \mathbb{Z}_{2^2}, \mathbb{Z}_{2^4}, \mathbb{Z}_{2^8}, \mathbb{Z}_{2^{16}}, \mathbb{Z}_{2^{32}}\}$ при $t = 3$, $N = 48$, $r \in \{1, \ldots, 8\}$, количество экспериментов для заданного веса r — 100.

r	$\mathbb{Z}_2$			$\mathbb{Z}_{2^2}$			$\mathbb{Z}_{2^4}$		
	δ	wt(y)	wt(z)	δ	wt(y)	wt(z)	δ	wt(y)	wt(z)
1	1.0	24.52	0.51	1.00	36.20	0.79	1.00	45.08	0.91
2	1.0	24.48	1.19	1.00	36.37	1.58	0.99	45.03	1.91
3	1.0	24.29	2.04	0.98	36.37	2.58	0.91	45.12	3.28
4	1.0	24.19	2.68	1.00	36.07	3.58	0.72	45.00	4.77
5	1.0	24.31	3.23	1.00	36.41	4.49	0.94	45.15	5.09
6	1.0	24.39	3.71	1.00	36.18	4.96	1.00	45.06	5.88
7	1.0	24.21	3.64	1.00	36.34	5.79	1.00	45.00	6.76
8	1.0	24.45	4.23	1.00	36.27	6.05	1.00	45.08	7.53

r	$\mathbb{Z}_{2^8}$			$\mathbb{Z}_{2^{16}}$			$\mathbb{Z}_{2^{32}}$		
	δ	wt(y)	wt(z)	δ	wt(y)	wt(z)	δ	wt(y)	wt(z)
1	1.00	47.87	1.00	1.00	48.0	0.98	1.00	48.0	1.00
2	0.99	47.86	2.03	1.00	48.0	1.94	1.00	48.0	2.00
3	0.73	47.86	4.26	0.75	48.0	4.10	0.79	48.0	3.93
4	0.30	47.84	6.63	0.31	48.0	6.58	0.36	48.0	6.39
5	0.26	47.85	7.15	0.20	48.0	7.32	0.14	48.0	7.47
6	0.39	47.89	7.21	0.24	48.0	7.48	0.26	48.0	7.46
7	1.00	47.84	7.00	0.63	48.0	7.36	0.57	48.0	7.42
8	1.00	47.82	7.94	1.00	48.0	7.99	1.00	48.0	7.99

Table 5. Finding the minimum weight vector by the algorithm ISD for $\mathbb{V} \in \{\mathbb{Z}_2, \mathbb{Z}_{2^2}, \mathbb{Z}_{2^4}, \mathbb{Z}_{2^8}, \mathbb{Z}_{2^{16}}, \mathbb{Z}_{2^{32}}\}$ when $t = 3$, $N = 64$, $r \in \{1, \dots, 8\}$, the number of experiments for the given weight r — 100.

Таблица 5. Поиск вектора минимального веса алгоритмом ISD для $\mathbb{V} \in \{\mathbb{Z}_2, \mathbb{Z}_{2^2}, \mathbb{Z}_{2^4}, \mathbb{Z}_{2^8}, \mathbb{Z}_{2^{16}}, \mathbb{Z}_{2^{32}}\}$ при $t = 3$, $N = 64$, $r \in \{1, \dots, 8\}$, количество экспериментов для заданного веса r — 100.

r	$\mathbb{Z}_2$			$\mathbb{Z}_{2^2}$			$\mathbb{Z}_{2^4}$		
	δ	wt(y)	wt(z)	δ	wt(y)	wt(z)	δ	wt(y)	wt(z)
1	1.0	31.62	0.44	1.0	48.06	0.75	1.00	60.05	0.95
2	1.0	31.68	1.09	1.0	48.00	1.63	0.99	60.00	1.97
3	1.0	31.79	1.80	1.0	47.98	2.53	0.75	60.01	3.97
4	1.0	31.69	2.80	1.0	47.85	3.57	0.66	59.99	4.98
5	1.0	31.44	3.43	1.0	48.14	4.58	0.95	60.04	5.04
6	1.0	31.79	3.85	1.0	48.29	5.19	1.00	59.79	5.87
7	1.0	31.42	3.77	1.0	48.07	5.75	1.00	59.95	6.75
8	1.0	31.74	3.81	1.0	48.09	5.95	1.00	59.97	7.49

r	$\mathbb{Z}_{2^8}$			$\mathbb{Z}_{2^{16}}$			$\mathbb{Z}_{2^{32}}$		
	δ	wt(y)	wt(z)	δ	wt(y)	wt(z)	δ	wt(y)	wt(z)
1	1.00	63.77	0.99	1.00	64.0	1.00	1.00	64.0	1.00
2	1.00	63.74	1.98	0.99	64.0	2.04	0.99	64.0	2.06
3	0.58	63.74	4.96	0.55	64.0	5.09	0.65	64.0	4.58
4	0.15	63.76	7.29	0.26	64.0	6.80	0.24	64.0	6.86
5	0.12	63.77	7.57	0.14	64.0	7.49	0.13	64.0	7.51
6	0.33	63.76	7.29	0.23	64.0	7.05	0.20	64.0	7.58
7	1.00	63.76	6.99	0.62	64.0	7.38	0.64	64.0	7.34
8	1.00	63.76	7.96	1.00	64.0	7.99	1.00	64.0	7.99

вес возвращаемого вектора. Это свидетельствует о том, что чем больше кольцо, тем менее вероятно по линейному MBA-выражению $E(x_1, \dots, x_t)$ произвольной длины найти эквивалентное MBA-выражение длины не более r. Для примера в выделенной в таблице 1 строке при $r = 2$ средний вес найденного вектора для $\mathbb{V} = \mathbb{Z}_2$ равен 1.13, но уже, начиная с $\mathbb{V} = \mathbb{Z}_{2^8}$, средний вес равен двум.

Результаты в таблицах 2–5 показывают, что с ростом N доля δ при $t = 3$ не возрастает, а для большинства колец только убывает, при этом средний вес возвращаемого вектора с ростом N меняется несущественно (как в большую, так и в меньшую стороны). Поэтому представляется, что значение N порядка $O(2^{t+2})$ может являться достаточным для алгоритма ISD.

2.3.2. Исследование алгоритма ISDMinSearch

Для алгоритма ISDMinSearch при $t = 2, \dots, 5$ и $I = 1000$ оценен средний вес возвращаемого вектора $\mathbf{z}$, когда входной вектор $\mathbf{y}$ выбран случайно и равновероятно из $\mathbb{V}^N$. Результаты оценки среднего веса представлены в таблице 7. Также при тех же t исследована зависимость среднего веса вектора, возвращаемого алгоритмом ISDMinSearch, от количества итераций $I \in \{10, 100, 1000, 10000\}$. Именно для случайно выбранного вектора веса N алгоритм ISDMinSearch вызывался с числом итераций 10000, в ходе выполнения которого фиксировался минимальный вес после $i \in \{10, 100, 1000, 10000\}$ итераций. Всего для каждого t было случайно выбрано по 10 векторов веса N. Результаты показаны в таблице 8. Кроме этого оценена зависимость среднего веса вектора, возвращаемого алгоритмом ISDMinSearch, от параметра N. Эксперименты проведены для $t = 3$ при $N = 16, 32, 48, 64$, а результаты представлены в таблице 9.

Результаты из таблицы 7 показывают, что для случайно выбранного вектора из $\mathbb{V}^N$ с ростом мощности кольца растет и средний вес wt(z) найденного алгоритмом ISDMinSearch вектора при фиксированном числе итераций I. В таблице 7 выделены ячейки, в которых количество слагаемых

Table 6. Finding the minimum weight vector by the algorithm ISD for $\mathbb{V} \in \{\mathbb{Z}_2, \mathbb{Z}_{2^2}, \mathbb{Z}_{2^4}, \mathbb{Z}_{2^8}, \mathbb{Z}_{2^{16}}, \mathbb{Z}_{2^{32}}\}$ when $t = 4$, $N = 64$, $r \in \{1, \ldots, 16\}$, the number of experiments for the given weight r — 100.

Таблица 6. Поиск вектора минимального веса алгоритмом ISD для $\mathbb{V} \in \{\mathbb{Z}_2, \mathbb{Z}_{2^2}, \mathbb{Z}_{2^4}, \mathbb{Z}_{2^8}, \mathbb{Z}_{2^{16}}, \mathbb{Z}_{2^{32}}\}$ при $t = 4$, $N = 64$, $r \in \{1, \ldots, 16\}$, количество экспериментов для заданного веса r — 100.

r	$\mathbb{Z}_2$			$\mathbb{Z}_{2^2}$			$\mathbb{Z}_{2^4}$		
	δ	wt(y)	wt(z)	δ	wt(y)	wt(z)	δ	wt(y)	wt(z)
1	1.00	31.53	0.50	1.00	48.32	0.77	1.00	60.08	0.92
2	1.00	31.49	1.04	1.00	48.35	1.46	1.00	60.05	1.83
3	0.99	31.31	1.56	0.99	48.48	2.23	0.99	60.10	2.91
4	1.00	31.60	2.46	0.92	48.34	3.45	0.72	59.91	6.84
5	1.00	31.44	3.80	0.79	48.43	5.26	0.29	60.07	11.75
6	1.00	31.38	4.60	0.69	48.28	7.06	0.04	60.10	14.56
7	1.00	31.89	5.86	0.94	48.35	6.69	0.01	60.19	14.79
8	1.00	31.84	6.61	1.00	48.30	7.71	0.01	59.77	14.82
9	1.00	31.35	6.99	1.00	48.14	8.62	0.00	59.94	14.88
10	1.00	31.81	7.43	1.00	48.42	9.42	0.12	59.72	14.40
11	1.00	32.37	7.77	1.00	48.44	10.32	0.54	59.74	12.82
12	1.00	31.51	7.88	1.00	48.55	11.02	1.00	60.02	11.80
13	1.00	31.95	7.87	1.00	48.28	11.46	1.00	59.69	12.80
14	1.00	31.34	7.93	1.00	48.76	11.76	1.00	59.82	13.61
15	1.00	31.57	7.86	1.00	48.33	11.98	1.00	59.95	14.31
16	1.00	31.57	8.44	1.00	47.89	11.87	1.00	59.92	14.92

r	$\mathbb{Z}_{2^8}$			$\mathbb{Z}_{2^{16}}$			$\mathbb{Z}_{2^{32}}$		
	δ	wt(y)	wt(z)	δ	wt(y)	wt(z)	δ	wt(y)	wt(z)
1	1.00	63.66	1.00	1.00	63.99	1.00	1.00	64.0	1.00
2	1.00	63.67	2.00	1.00	63.99	2.00	1.00	64.0	2.00
3	0.98	63.66	3.26	0.99	63.99	3.13	0.98	64.0	3.26
4	0.54	63.67	9.44	0.64	63.99	8.32	0.57	64.0	9.15
5	0.10	63.67	14.81	0.14	63.99	14.46	0.12	64.0	14.67
6	0.04	63.68	15.54	0.06	63.99	15.40	0.03	64.0	15.70
7	0.00	63.68	15.92	0.00	63.99	15.99	0.00	64.0	16.00
8	0.00	63.76	15.97	0.00	64.00	16.00	0.01	64.0	15.92
9	0.00	63.71	15.97	0.00	64.00	16.00	0.00	64.0	16.00
10	0.00	63.71	15.94	0.00	64.00	16.00	0.00	64.0	16.00
11	0.00	63.74	15.96	0.00	64.00	16.00	0.00	64.0	16.00
12	0.00	63.73	15.90	0.00	63.99	16.00	0.00	64.0	16.00
13	0.00	63.76	15.94	0.00	63.99	16.00	0.00	64.0	16.00
14	0.41	63.78	15.13	0.00	63.99	16.00	0.00	64.0	16.00
15	1.00	63.75	14.95	0.08	63.99	15.92	0.02	64.0	15.98
16	1.00	63.80	15.95	1.00	64.00	16.00	1.00	64.0	16.00

в упрощенном MBA-выражении больше 10. Этот порог выбран на основании результатов работы [8], где экспериментально установлено, что при числе чередований 10 и более время работы SMT-решателей превышает 3 000 секунд. Следовательно, по выделенным ячейкам можно определить размер кольца $\mathbb{V}$ и количество переменных t, для которых время анализа SMT-решателем случайного линейного MBA-выражения (с заданным синдромом) будет в среднем не менее 3 000 секунд. Например, если для тождественно истинного предиката P_T на рис. 1 нелинейные MBA-вы-

Table 7. Finding the low weight vector for $\mathbb{V} \in \{\mathbb{Z}_2, \mathbb{Z}_{2^2}, \mathbb{Z}_{2^4}, \mathbb{Z}_{2^8}, \mathbb{Z}_{2^{16}}, \mathbb{Z}_{2^{32}}\}$ when $t \in \{2, 3, 4, 5\}$, $N = 4 \cdot 2^t$, $r \in \{1, \ldots, 2^t\}$, number of iterations 1000 of the algorithm ISDMinSearch, number of randomly chosen vectors — 100.

Таблица 7. Поиск вектора малого веса для $\mathbb{V} \in \{\mathbb{Z}_2, \mathbb{Z}_{2^2}, \mathbb{Z}_{2^4}, \mathbb{Z}_{2^8}, \mathbb{Z}_{2^{16}}, \mathbb{Z}_{2^{32}}\}$ при $t \in \{2, 3, 4, 5\}$, $N = 4 \cdot 2^t$, $r \in \{1, \ldots, 2^t\}$, количество итераций 1000 алгоритма ISDMinSearch, количество случайно выбираемых векторов — 100.

t	$\mathbb{Z}_2$		$\mathbb{Z}_{2^2}$		$\mathbb{Z}_{2^4}$		$\mathbb{Z}_{2^8}$		$\mathbb{Z}_{2^{16}}$		$\mathbb{Z}_{2^{32}}$	
	wt(y)	wt(z)	wt(y)	wt(z)	wt(y)	wt(z)	wt(y)	wt(z)	wt(y)	wt(z)	wt(y)	wt(z)
2	7.39	0.93	11.29	1.79	14.18	2.91	14.92	3.84	15.0	4.00	15.0	4.0
3	16.41	1.99	24.16	3.23	30.00	5.24	31.86	6.91	32.0	7.98	32.0	8.0
4	31.81	3.47	47.99	6.74	60.01	**11.44**	63.74	**14.59**	64.0	**15.94**	64.0	**16.0**
5	61.30	**11.00**	94.20	**19.50**	120.10	**26.70**	127.70	**30.80**	128.0	**32.00**	128.0	**32.0**

Table 8. Dependence of the average weight of the vector returned by the algorithm ISDMinSearch on the number of iterations for 10 randomly chosen vectors $\mathbf{v}$ of the weight N.

Таблица 8. Зависимость среднего веса возвращаемого алгоритмом ISDMinSearch вектора от количества итераций для 10 случайно выбранных векторов $\mathbf{v}$ веса N.

$\mathbb{V}$	I $(t = 2)$				I $(t = 3)$				I $(t = 4)$				I $(t = 5)$			
	10	10^2	10^3	10^4	10	10^2	10^3	10^4	10	10^2	10^3	10^4	10	10^2	10^3	10^4
$\mathbb{Z}_2$	1.1	1.0	1.0	1.0	3.4	1.8	1.8	1.8	6.4	4.0	3.6	3.6	15.6	11.2	7.4	6.4
$\mathbb{Z}_{2^2}$	2.1	1.7	1.7	1.7	6.0	3.5	3.1	3.1	11.0	8.4	6.8	5.8	21.6	17.6	15.8	15.0
$\mathbb{Z}_{2^4}$	3.4	3.0	2.9	2.9	7.1	5.8	5.3	4.8	14.5	12.7	11.4	10.3	29.2	26.4	24.6	23.4
$\mathbb{Z}_{2^8}$	4.0	3.9	3.8	3.8	7.9	7.4	6.9	6.7	15.6	15.1	14.6	14.0	31.8	31.0	30.0	29.6
$\mathbb{Z}_{2^{16}}$	4.0	4.0	4.0	4.0	10.4	8.0	8.0	7.8	20.8	16.0	16.0	15.5	32.0	32.0	32.0	32.0
$\mathbb{Z}_{2^{32}}$	4.0	4.0	4.0	4.0	12.8	8.0	8.0	8.0	16.0	16.0	16.0	16.0	32.0	32.0	32.0	32.0

Table 9. Dependence of the average weight of the vector returned by the algorithm ISDMinSearch on N for 10 randomly chosen vectors $\mathbf{v}$ of the weight N.

Таблица 9. Зависимость среднего веса возвращаемого алгоритмом ISDMinSearch вектора от N для 10 случайно выбранных векторов $\mathbf{v}$ веса N.

	I $(N = 16)$				I $(N = 32)$				I $(N = 48)$				I $(N = 64)$			
	10	10^2	10^3	10^4	10	10^2	10^3	10^4	10	10^2	10^3	10^4	10	10^2	10^3	10^4
$\mathbb{Z}_2$	4.0	2.9	2.9	2.9	3.4	1.8	1.8	1.8	3.5	1.9	1.8	1.8	3.6	1.9	1.9	1.9
$\mathbb{Z}_{2^2}$	5.4	4.0	3.9	3.9	6.0	3.5	3.1	3.1	4.8	3.6	3.1	3.1	4.6	3.3	3.0	3.0
$\mathbb{Z}_{2^4}$	6.5	5.8	5.6	5.6	7.1	5.8	5.3	4.8	11.2	5.9	5.0	4.7	12.8	5.8	5.0	4.4
$\mathbb{Z}_{2^8}$	8.8	7.7	7.0	6.9	7.9	7.4	6.9	6.7	7.9	7.6	6.8	6.7	7.9	7.4	7.0	6.5
$\mathbb{Z}_{2^{16}}$	8.0	7.9	7.9	7.9	10.4	8.0	8.0	7.8	8.0	8.0	8.0	7.9	19.2	8.0	8.0	8.0
$\mathbb{Z}_{2^{32}}$	8.0	8.0	8.0	8.0	12.8	8.0	8.0	8.0	8.0	8.0	8.0	8.0	8.0	8.0	8.0	8.0

ражения $E_1(x_1, \ldots, x_t)$ и $E_2(x_1, \ldots, x_t)$ содержат линейные MBA-подвыражения, выбранные случайно и равновероятно из некоторых фактор-классов $\mathbb{V}^N/C_F$, то при $t \geqslant 4$ и $\mathbb{V} \in \{\mathbb{Z}_{2^8}, \mathbb{Z}_{2^{16}}, \mathbb{Z}_{2^{32}}\}$ можно ожидать, что время решения SMT-модулем предиката P_T будет не менее 3000 секунд. Подчеркнем, что выражения $E_1(x_1, \ldots, x_t)$ и $E_2(x_1, \ldots, x_t)$ следует выбирать нелинейными, иначе, согласно [4], тождественную истинность или ложность предикатов на основе этих выражений можно установить менее, чем за секунду, сравнивая соответствующие им подписи ($\mathbb{V}$-синдромы).

Результаты эксперимента, целью которого была оценка зависимости среднего веса возвращаемого вектора от числа итераций I, показаны в таблице 8. Анализ этих результатов показывает, что для колец $\mathbb{Z}_2$, $\mathbb{Z}_{2^2}$ и $\mathbb{Z}_{2^4}$ при $I = 1000$ наблюдается замедление в падении среднего веса возвращаемого вектора. Именно, в этом случае разница в среднем весе для $I = 1000$ и $I = 10000$ при всех рассмотренных значениях t не превышает 1.2. Представляется, что значения I больше 10000 могут с несущественной вероятностью уменьшить вес возвращаемого вектора ценой существенно боль-

шей вычислительной сложности. Для колец $\mathbb{Z}_{2^8}$, $\mathbb{Z}_{2^{16}}$ и $\mathbb{Z}_{2^{32}}$ замедление в падении среднего веса возвращаемого вектора наблюдается уже при $I = 100$.

Из таблицы 9 видно, что при $I = 10^4$ для колец $\mathbb{Z}_2$, $\mathbb{Z}_{2^2}$, $\mathbb{Z}_{2^4}$ и $\mathbb{Z}_{2^8}$ средний вес возвращаемого вектора несущественно уменьшается с ростом N, когда входной вектор алгоритма ISDMinSearch выбирается случайно и равновероятно из $\mathbb{V}^N$; для колец $\mathbb{Z}_{2^{16}}$ и $\mathbb{Z}_{2^{32}}$ с ростом I средний вес для всех рассмотренных N колеблется около $2^t = 8$. Можно предположить, что при деобфускации случайных линейных MBA-выражений (не только при $t = 3$) для небольших колец значение N следует выбирать побольше, чтобы уменьшить число слагаемых в MBA-выражении. Однако в этом случае возрастает вычислительная сложность алгоритма ISDMinSearch, так как в нем приходится находить обратные $(N - 2^t) \times (N - 2^t)$-матрицы, что требует порядка $O((N - 2^t)^3)$ операций.

Для всех рассмотренных t при $r = 1$ для колец малой мощности средний вес вектора, возвращаемого алгоритмом ISD, меньше единицы. Это означает, что в ряде случаев вес возвращаемого вектора равен нулю. Откуда следует, что некоторые обфусцируемые в экспериментах с помощью алгоритмов LinMBAGen и LinMBAGenAnyLen MBA-выражения принадлежали $\mathcal{C}_F$. Поэтому, перед обфускацией MBA-выражения $E(x_1, \ldots, x_t)$ (например, при генерации множества непрозрачных предикатов) следует проверить, что $\mathcal{E}(E(x_1, \ldots, x_t)) \notin \mathcal{C}_F$. Более общей рекомендацией может являться сравнение веса вектора, возвращаемого алгоритмом ISDMinSearch с заданным весом. Эта проверка может быть выполнена с помощью алгоритма CheckMBAComplexity (см. алгоритм 6).

Algorithm 6. CheckMBAComplexity

Data: $E(x_1, \ldots, x_t)$, I_1, I_2, r, R
Result: $T \in \{0, 1\}$, $T = 0$ — вес вектора, возвращаемого ISDMinSearch, меньше r, $T = 1$ — вес не меньше r

```
1 i = 0, T = 1
2 i = i + 1
3 if i < I_2 then
      /* Попытаться упростить MBA-выражение за I_1 итераций с использованием R случайно
         выбранных добавочных столбцов                                              */
4     z = E(MBASimplify(E(x_1, ..., x_t), I_1, R))
      /* Проверить условие на вес вектора                                           */
5     if wt(z) ⩾ r then
          /* Для очередной проверки вес возвращенного вектора не менее r; перейти
             к следующей проверке (проверка с другой достроенной матрицей A_F)     */
6         go to 2
7     else
          /* Среди I_2 случайно достроенных матриц A_F нашлась, по крайней мере, одна,
             для которой алгоритм ISDMinSearch вернул MBA-выражение с менее чем r
             слагаемыми                                                             */
8         T = 0
9 return T
```

Заключение

При всех рассмотренных t и $N = O(2^{t+2})$ предлагаемый способ упрощения линейных MBA-выражений на основе техники декодирования по информационным совокупностям позволяет существенно сократить длину выражения в случае, когда вес вектора $\mathbf{z}$ в (12) не превышает 3. В случае $\mathrm{wt}(\mathbf{z}) > 3$ для колец $\mathbb{Z}_{2^8}$, $\mathbb{Z}_{2^{16}}$ и $\mathbb{Z}_{2^{32}}$ средний вес возвращаемого вектора практически всегда равен 2^t.

В этом же случае количество итераций поиска может быть порядка $I = 100$, так как разумное увеличение числа итераций практически не увеличивает вероятность найти вектор меньшего веса. Для меньших колец при $I \geqslant 1000$ с большой вероятностью может быть получен вектор веса меньше 2^t. Заметим, что целью известных алгоритмов упрощения (см. [8] и [4]) является получение MBA-выражения длиной порядка 2^t. Таким образом, в рамках задачи деобфускации MBA-выражений (не только линейных), предлагаемый способ может быть применен как предварительный шаг перед применением метода из [9], в котором эквивалентность MBA-выражений проверяется с помощью SMT-решателей. А так как время решения зависит от количества чередований в MBA-выражении, то предлагаемый способ позволит сократить это время. Способ упрощения линейных MBA-выражений на основе декодирования по информационным совокупностям, как представляется, можно использовать не только при деобфускации кода, но и при его *оптимизации*. Кроме того, такая оптимизация может быть полезной в среде символьного исполнения KLEE, где также актуальна задача сокращения времени выполнения запросов к SMT-решателю.

References

[1] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang, “On the (im)possibility of obfuscating programs”, in *Advances in Cryptology — CRYPTO 2001*, ser. Lecture Notes in Computer Science, Springer, vol. 2139, 2001, pp. 1–18.

[2] Y. Zhou, A. Main, Y. X. Gu, and H. Johnson, “Information hiding in software with mixed boolean-arithmetic transforms”, in *Information Security Applications. WISA 2007*, ser. Lecture Notes in Computer Science, Springer, vol. 4867, 2007, pp. 61–75.

[3] S. Gulwani, O. Polozov, and R. Singh, “Program synthesis”, *Foundations and Trends in Programming Languages*, vol. 4, no. 1-2, pp. 1–119, 2017.

[4] B. Reichenwallner and P. Meerwald-Stadler, “Efficient deobfuscation of linear mixed boolean-arithmetic expressions”, in *Proceedings of the 2022 ACM Workshop on Research on offensive and defensive techniques in the context of Man At The End (MATE) attacks*, 2022, pp. 19–28.

[5] L. Zobernig, “Mathematical aspects of program obfuscation”, Ph.D. dissertation, The University of Auckland, 2020. [Online]. Available: https://researchspace.auckland.ac.nz/handle/2292/53400.

[6] P. Garba and M. Favaro, “Saturn-software deobfuscation framework based on LLVM”, in *Proceedings of the 3rd ACM Workshop on Software Protection*, 2019, pp. 27–38.

[7] N. Eyrolles, “Obfuscation with mixed boolean-arithmetic expressions: Reconstruction, analysis and simplification tools”, Ph.D. dissertation, Université Paris-Saclay, 2017, 126 pp.

[8] D. Xu, B. Liu, W. Feng, J. Ming, Q. Zheng, J. Li, and Q. Yu, “Boosting SMT solver performance on mixed-bitwise-arithmetic expressions”, in *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation*, 2021, pp. 651–664.

[9] B. Liu, J. Shen, J. Ming, Q. Zheng, J. Li, and D. Xu, “MBA-Blast: Unveiling and simplifying mixed boolean-arithmetic obfuscation”, 2021, pp. 1701–1718.

[10] E. Berlekamp, R. McEliece, and H. Van Tilborg, “On the inherent intractability of certain coding problems (corresp.)”, *IEEE Transactions on Information Theory*, vol. 24, no. 3, pp. 384–386, 1978.

[11] E. Prange, "The use of information sets in decoding cyclic codes", *IRE Transactions on Information Theory*, vol. 8, no. 5, pp. 5–9, 1962.

[12] C. Peters, "Information-set decoding for linear codes over $\mathbb{F}_q$", in *Post-Quantum Cryptography. PQCrypto 2010*, ser. Lecture Notes in Computer Science, Springer, vol. 6061, 2010, pp. 81–94.

[13] V. Weger, N. Gassner, and J. Rosenthal, *A survey on code-based cryptography*, 2022. arXiv: 2201.07119v3 [cs.CR].