Таблица 2

Верхние оценки числа кубит и глубины схемы

Параметры NTRU	Количество кубит	Глубина схемы
$N = 1$, $q = 2$	105	332
$N = 2$, $q = 2$	266	428
$N = 8$, $q = 4$	3742	3498
$N = 256$, $q = 128$	4138013	2945510

Таким образом, в работе получены верхние оценки сложности реализации квантового оракула из алгоритма Гровера для реализации гибридного квантово-классического алгоритма на основе GaussSieve, который может быть использован для атак на криптосистемы, стойкость которых зависит от решения задачи SVP. Проанализирована сложность реализации квантового оракула для атаки на постквантовую криптосистему NTRU. На сегодняшний день количество кубит, с которыми оперирует квантовый компьютер, не превосходит 76 [7]. Из полученных оценок следует, что предложенная модель квантового оракула не может быть реализована на квантовом компьютере даже для самых малых параметров NTRU, так как ещё не существует квантового компьютера, оперирующего достаточным количеством кубит. В рамках дальнейшей работы предлагается оптимизировать квантовую схему оракула, получить необходимые оценки для реализации оракула данного класса, а также проанализировать другие известные классические атаки на постквантовые криптосистемы с целью изучения возможности их ускорения с помощью квантовых вычислений.

ЛИТЕРАТУРА

1. *Laarhoven T., Mosca M., and van de Pol J.* Finding shortest lattice vectors faster using quantum search // Des. Codes Cryptogr. 2015. V. 77. No. 2–3. P. 375–400.
2. *Micciancio D. and Voulgaris P.* Faster exponential time algorithms for the Shortest Vector roblem // 21st Ann. ACM Symp. Discrete Algorithms (SODA). 2010. P. 1468–1480.
3. *Grover L. K.* A fast quantum mechanical algorithm for database search // 28th Ann. ACM Symp. Theory Comput. (STOC). 1996. P. 212–219.
4. *Nielsen M. A. and Chuang I. L.* Quantum Computation and Quantum Information. Cambridge: Cambridge University Press, 2010.
5. `https://csrc.nist.gov/Projects/post-quantum-cryptography/round-3-submissions`.
6. *Chen C., Danba O., Hoffstein J., et al.* NTRU Algorithm Specifications and Supporting Documentation. `https://ntru.org/`, 2019.
7. *Zhong H.-S., Wang H., Deng Y.-H., et al.* Quantum computational advantage using photons. Science. 2020. V. 370. Iss. 6523. P. 1460–1463.

УДК 519.7 DOI 10.17223/2226308X/14/14

КРИПТОАНАЛИТИЧЕСКАЯ ОБРАТИМОСТЬ ФУНКЦИЙ ДВУХ АРГУМЕНТОВ

Н. Ю. Бердникова, И. А. Панкратова

Предложены тесты криптоаналитической обратимости всех возможных типов для произвольных функций от двух аргументов. Сформулированы алгоритмы построения функции восстановления и генерации обратимых функций; посчитано количество обратимых функций некоторых типов.

Ключевые слова: *обратимость функции по переменной, криптоаналитическая обратимость, тест обратимости, функция восстановления.*

Понятие криптоаналитической обратимости функции введено Г. П. Агибаловым в [1, 2] как обобщение, с одной стороны, понятия «обычной» обратимости функции, с другой — криптоаналитической обратимости конечного автомата [3]. Обобщение понятия обратимости функции сделано в двух направлениях:

— обращение *по переменной*: по значению функции восстанавливается не весь набор значений аргументов, а только значение некоторой переменной;
— использование кванторов: восстановление значения переменной возможно не обязательно *для всех* значений остальных переменных.

Определение 1 [2]. Функция $g(x_1, \ldots, x_n)$ *обратима по переменной* x_k, $k \in \{1, \ldots, n\}$, *типа* $K_1 \ldots K_n$, где $K_i \in \{\exists, \forall\}$ и $K_k = \forall$, если существует *функция восстановления* f, такая, что верна формула

$$K_1 x_n \ldots K_n x_n \big(f(g(x_1, \ldots, x_n)) = x_k\big).$$

Понятно, что если функция обратима по всем переменным типа $\forall\forall \ldots \forall$, то она обратима в классическом смысле.

Для каждого типа обратимости возникают следующие задачи:

1) разработка теста обратимости;
2) разработка алгоритма построения функции восстановления;
3) разработка алгоритмов генерации обратимых функций (возможно, с дополнительными условиями — равновероятность генерации любой функции из класса; генерация функций с заданными свойствами и т. п.);
4) подсчёт или оценка количества обратимых функций.

Рассмотрим некоторые из этих задач для случая $n = 2$, т. е. для функций вида

$$g : D_1 \times D_2 \to D,$$

где D_1, D_2, D — произвольные множества.

Введём обозначения: $|M|$ — мощность множества M (конечного или бесконечного); D_g — множество значений функции g; G_a — множество значений подфункции, полученной из g фиксацией переменной $x_1 = a$, $a \in G_1$:

$$D_g = \{g(x_1, x_2) : x_1 \in D_1, x_2 \in D_2\}, \quad G_a = \{g(a, x_2) : x_2 \in D_2\}.$$

Очевидно, что необходимым условием обратимости функции $g(x_1, x_2)$ по переменной x_k, $k \in \{1, 2\}$, является $|D_g| \geqslant |D_k|$; будем всюду считать, что оно выполнено.

1. Обратимость типа $\forall\forall$

Условие обратимости функции $g(x_1, x_2)$ по переменной x_k, $k \in \{1, 2\}$, в этом случае записывается так:

$$\exists f \forall x_1 \forall x_2 \big(f(g(x_1, x_2)) = x_k\big).$$

Тест обратимости является частным случаем (при $n = 2$) леммы 1 из [3].

Утверждение 1 (тест обратимости типа $\forall\forall$). Функция $g(x_1, x_2)$ обратима типа $\forall\forall$ по переменной x_k, $k \in \{1, 2\}$, если и только если

$$\forall x_1 \forall x_2 \forall y_1 \forall y_2 \big(x_k \neq y_k \Rightarrow g(x_1, x_2) \neq g(y_1, y_2)\big).$$

Без ограничения общности (поскольку одноимённые кванторы перестановочны) далее будем считать, что $k = 1$, и переформулируем тест более конструктивным образом.

Утверждение 2. Функция $g(x_1, x_2)$ обратима типа $\forall\forall$ по переменной x_1, если и только если

$$\forall a, b \in D_1 \big(a \neq b \Rightarrow G_a \cap G_b = \varnothing\big). \tag{1}$$

Функция восстановления $f : D_g \to D_1$ строится по формуле

$$\forall x_1 \in D_1 \forall x_2 \in D_2 \big(f(g(x_1, x_2)) = x_1\big),$$

функциональность отношения f следует из условия (1).

Можно предложить следующий **алгоритм 1** генерации обратимой функции g:

1. Построить произвольное разбиение множества D на классы $D^{(a)}$, $a \in D_1$.
2. Для всех $a \in D_1$:
   2.1) для каждого $x_2 \in D_2$ выбрать в качестве $g(a, x_2)$ случайное значение из множества $D^{(a)}$.

Корректность алгоритма 1 следует из выполнения для построенной функции g условия (1); его полнота — из произвольности выбора разбиения на шаге 1 и значений функции на шаге 2.1.

Если множества D_1 и D конечны и $|D| = |D_1| = m$, то количество функций, обратимых типа $\forall\forall$ по переменной x_1, равно $m!$.

2. Обратимость типа $\forall\exists$

Условие обратимости типа $\forall\exists$ функции $g(x_1, x_2)$ по переменной x_1:

$$\exists f \forall x_1 \exists x_2 \big(f(g(x_1, x_2)) = x_1\big).$$

Утверждение 3 (тест обратимости типа $\forall\exists$). Функция $g : D_1 \times D_2 \to D$ обратима типа $\forall\exists$ по переменной x_1, если и только если существует такое отображение $\varphi : D_1 \to D_2$, что выполнено условие

$$\forall a, b \in D_1 \big(a \neq b \Rightarrow g(a, \varphi(a)) \neq g(b, \varphi(b))\big). \tag{2}$$

К сожалению, тест не конструктивен, так как требует проверки существования нужного отображения. Если отображение, удовлетворяющее условию (2), удалось найти, то функция восстановления $f : D_g \to D_1$ строится так:

1. Для всех $a \in D_1$ положить $f(g(a, \varphi(a))) = a$.
2. Для каждого $y \in D_g$, такого, что значение $f(y)$ не определено на шаге 1, выбрать в качестве $f(y)$ произвольное значение из D_1.

Функциональность отношения f следует из того, что, в силу условия (2), все значения $g(a, \varphi(a))$ для $a \in D_1$ попарно различны.

Алгоритм 2 генерации обратимой типа $\forall\exists$ функции $g : D_1 \times D_2 \to D$:

1. Положить $C = D$.
2. Для всех $a \in D_1$:
   2.1) выбрать случайные значения $b \in D_2$ и $c \in C$;
   2.2) положить $g(a, b) = c$;
   2.3) $C := C \setminus \{c\}$;
   2.4) для каждого $x_2 \in D_2 \setminus \{b\}$ выбрать в качестве $g(a, x_2)$ произвольное значение из D.

Корректность алгоритма 2: будем параллельно с функцией g строить отображение $\varphi : D_1 \to D_2$, полагая в шаге 2.1 $\varphi(a) = b$. Тогда для этих g и φ выполнено условие (2), поскольку шаг 2.3 обеспечивает попарную различность значений $g(a, b)$.

Полнота алгоритма 2: пусть для функции g и отображения φ выполнено условие (2). Тогда именно эта функция будет построена алгоритмом 2 при выборе значений $b = \varphi(a)$ и $c = g(a, b)$ в шаге 2.1 и значений $g(a, x_2)$ в качестве соответствующих «произвольных» в шаге 2.4.

3. Обратимость типа $\exists\forall$

Условие обратимости типа $\exists\forall$ функции $g(x_1, x_2)$ по переменной x_2:

$$\exists f \exists x_1 \forall x_2 \big(f(g(x_1, x_2)) = x_2\big).$$

Утверждение 4 (тест обратимости типа $\exists\forall$). Функция $g : D_1 \times D_2 \to D_g$ обратима типа $\exists\forall$ по переменной x_2, если и только если существует такое $a \in D_1$, что

$$|G_a| = |D_2|. \tag{3}$$

Функция восстановления $f : D_g \to D_2$ строится так:

1. Для $a \in D_1$, удовлетворяющего условию (3), и каждого $x_2 \in D_2$ положить $f(g(a, x_2)) = x_2$.
2. Для каждого $y \in D_g$, такого, что значение $f(y)$ не определено на шаге 1, выбрать в качестве $f(y)$ произвольное значение из D_2.

Функциональность отношения f следует из того, что, ввиду условия (3), значения $g(a, x_2)$, $x_2 \in D_2$, попарно различны.

Алгоритм 3 генерации обратимой типа $\exists\forall$ функции $g : D_1 \times D_2 \to D$:

1. Выбрать случайное значение $a \in D_1$.
2. Положить $C = D$.
3. Для всех $b \in D_2$:
   - 3.1) выбрать случайное значение $c \in C$;
   - 3.2) положить $g(a, b) = c$;
   - 3.3) $C := C \setminus \{c\}$.
4. Для всех $x_1 \in D_1 \setminus \{a\}$:
   - 4.1) для каждого $x_2 \in D_2$ выбрать в качестве $g(x_1, x_2)$ произвольное значение из D.

Корректность алгоритма 3: шаг 3 обеспечивают выполнение условия (3) для значения a, выбранного на шаге 1. Полнота доказывается так же, как для алгоритма 2.

Пусть все множества D_1, D_2, D конечны и $D_2 = \{b_1, \ldots, b_m\}$. Для подсчёта количества обратимых типа $\exists\forall$ функций вычислим количество необратимых. Условие необратимости (отрицание теста) можно записать так:

$$\forall a \in D_1 \, (|G_a| < |D_2|). \tag{4}$$

Для $a \in D_1$ рассмотрим вектор $\big(g(a, b_1), \ldots, g(a, b_m)\big)$; существует всего $|D|^{|D_2|}$ различных таких векторов, из них $\binom{|D|}{|D_2|}|D_2|!$ состоят из попарно различных значений. Таким образом, количество необратимых функций (удовлетворяющих условию (4)) равно

$$C_{\text{необр}} = \left(|D|^{|D_2|} - \binom{|D|}{|D_2|}|D_2|!\right)^{|D_1|}.$$

Количество обратимых типа $\exists\forall$ функций равно

$$|D|^{|D_1|\cdot|D_2|} - C_{\text{необр}};$$

в частности, для $|D| = |D_1| = |D_2| = m$ получаем $m^{m^2} - (m^m - m!)^m$.

ЛИТЕРАТУРА

1. *Agibalov G. P.* Cryptanalytical finite automaton invertibility with finite delay // Прикладная дискретная математика. 2019. № 46. С. 27–37.
2. *Agibalov G. P.* Problems in theory of cryptanalytical invertibility of finite automata // Прикладная дискретная математика. 2020. № 50. С. 62–71.
3. *Agibalov G. P.* Cryptanalytic concept of finite automaton invertibility with finite delay // Прикладная дискретная математика. 2019. № 44. С. 34–42.

УДК 004.056 DOI 10.17223/2226308X/14/15

ЭКСПЕРИМЕНТАЛЬНОЕ ИССЛЕДОВАНИЕ ХАРАКТЕРИСТИК ОДНОГО СПОСОБА КОНТРОЛЯ ЦЕЛОСТНОСТИ ПРИ ХРАНЕНИИ ДАННЫХ БОЛЬШОГО ОБЪЁМА

Д. А. Бобровский, Д. И. Задорожный, А. М. Коренева, Т. Р. Набиев, В. М. Фомичёв

Описан способ встраивания высокопроизводительного алгоритма генерации кода контроля целостности, представленного авторами на РусКрипто'2020, в функцию хэширования, определенную в ГОСТ 34.11-2018 (256 бит). Полученные ранее результаты существенно улучшены. Проведены экспериментальные исследования производительности и криптографических свойств нового алгоритма. Установлено, что предложенный алгоритм производительнее известных криптографических функций хэширования, близок по производительности к CRC32, а по криптографическим свойствам значительно его превосходит.

Ключевые слова: *аддитивные генераторы, контроль целостности, матрично-графовый подход, перемешивающие свойства, регистры сдвига, AG-S, AG-S-Стрибог, SMHasher.*

Введение

Обеспечение целостности хранимых данных относится к основным задачам защиты информации. В настоящее время применяются различные алгоритмы и подходы: хэш-функции (MD, SHA, ГОСТ 34.11-2018 и др.), хэш-функции с ключом (HMAC), блочные шифры в режиме выработки имитовставки (CMAC). При контроле целостности (КЦ) применяются также некриптографические методы с использованием кодов, обнаруживающих и/или исправляющих ошибки (коды Хэмминга, циклические коды (CRC) и др.).

При динамическом контроле больших объёмов данных, КЦ исполняющей среды функционирования, а также при проведении оперативного аудита целевых систем возникает проблема вычислений с высокой ресурсоёмкостью. Непосредственное использование для решения данной проблемы известных подходов и алгоритмов затруднительно в силу имеющихся у них недостатков: высокой ресурсоёмкости, слабых криптографических характеристик и пр. В работе предложено альтернативное решение на основе комбинации высокопроизводительного алгоритма (например, CRC32) и хэш-функции, соответствующей современным требованиям к криптографической стойкости (например, ГОСТ 34.11-2018). Удачное решение подразумевает компромисс между скоростью