ТЕОРЕТИЧЕСКОЕ ОБОСНОВАНИЕ СТОЙКОСТИ НЕРАЗЛИЧИМОЙ ОБФУСКАЦИИ

Козачок А.В.[1], Бочков М.В.[2], Лай М.Т.[3]

Целью проводимого исследования является построение комплекса моделей и алгоритмов процесса контролируемого разграничения доступа к файлам документальных форматов, позволяющего осуществить защиту от несанкционированного доступа к информации за счет применения неразличимой обфускации программного кода. Предлагается теоретическое обоснование стойкости аналитической модели защиты файлов документальных форматов от несанкционированного доступа.

В статье рассмотрены различные подходы к определению стойкости неразличимой обфускации, предложен алгоритм декомпозиции элементов и описан подход к моделированию неразличимого обфускатора в виде модели со случайным оракулом. Одним из возможных вариантов доказательства стойкости криптографических преобразований является применение модели случайного оракула, под оракулом понимается вспомогательный алгоритм, который выполняется за один шаг работы основного алгоритма.

Совокупность сформулированных и доказанных утверждений позволила теоретически обосновать возможность построения обфускатора, стойкого в модели виртуального «черного ящика». Это означает, что не существует алгоритма распознавания обфускации более эффективного, чем обычное предположение, сделанное на основе анализа входов и выходов обфусцированной программы.

***Ключевые слова**: обфускация, случайный оракул, декомпозиция, стойкость, виртуальный черный ящик.*

Введение

В последние годы при проведении научных исследований особое внимание уделяется вопросам обеспечения безопасности инфокоммуникационных систем. Главным образом, эти исследования касаются формальной верификации свойств безопасности. Основной целью при этом является разработка формальной математической модели свойств безопасности системы, а также верификация этой модели с помощью математических доказательств.

Целью проводимого исследования является построение комплекса моделей и алгоритмов процесса контролируемого разграничения доступа к файлам документальных форматов, позволяющего осуществить защиту от несанкционированного доступа к информации за счет применения неразличимой обфускации программного кода [1, 2].

Обфускацией программы называется всякое ее преобразование, которое сохраняет вычисляемую программой функцию (эквивалентное преобразование), но при этом придает программе такую форму, что извлечение из текста программы (программного кода) ключевой информации об алгоритмах и структурах данных, реализованных в этой программе, становится трудоемкой задачей [3].

Исходя из вышеизложенного, в статье предлагается теоретическое обоснование стойкости аналитической модели защиты файлов документальных форматов от несанкционированного доступа, отличающейся применением процедуры неразличимой обфускации программного кода [4].

1. Понятие стойкости неразличимой обфускации

В 2001 году впервые строгое математическое определение стойкости обфускации программного кода была предложено Бараком [5]: обфускация считается стойкой, если всякий противник может извлечь из текста обфусцированной программы за разумное (полиномиальное относительно размера входных данных) время не больше информации, чем можно было бы получить, проводя тестовые испытания этой программы как «черного ящика». Рассмотрим основные понятия стойкости обфускации [5] – стойкость на основе понятия неразличимости (iO) и стойкость обфускации в модели виртуального «черного ящика» (VBB).

Определение 1. (Обфускация в модели виртуального «черного ящика»). Пусть $\{C_\ell\}_{\ell\in N}$ – класс булевых схем, тогда обфускатором в модели виртуального «черного ящика» для класса схем $\{C_\ell\}_{\ell\in N}$ называется такая (полиномиальная

1 Козачок Александр Васильевич, кандидат технических наук, Академия ФСО России, г. Орёл, alex.totrin@gmail.com
2 Бочков Максим Вадимович, доктор технических наук, профессор, ЧОУ ДПО «ЦПР», г. Санкт-Петербург, mvboch@yandex.ru
3 Лай Минь Туан, Академия ФСО России, г. Орёл, lmtuan.1989@gmail.com

вероятностная машина Тьюринга) PPT O, которая удовлетворяет следующим требованиям:

– функциональная эквивалентность. Для любой схемы $C \in \{C_\ell\}_{\ell \in N}$ и любого входа x существует пренебрежимо малая функция μ, такая что:

$$\Pr[(O(C))(x) \neq C(x)] \leq \mu(|C|);$$

– полиномиальные издержки. Существуют полином $poly(\cdot)$ такой, что для любой схемы $C \in \{C_\ell\}_{\ell \in N}$:

$$|O(C)| \leq poly(|C|);$$

– свойство виртуального «черного ящика»: для любой PPT A (противника) существуют такая PPT Sim (симулятор) и пренебрежимо малая функция μ такая, что для любой схемы $C \in \{C_\ell\}_{\ell \in N}$:

$$|Pr[A(O(C)) = 1] - Pr[Sim^C(1^{|C|}) = 1]| \leq \mu(|C|).$$

В работе Барака также было доказано утверждение о невозможности построения идеального обфускатора для всех классов булевых функций, но данное утверждение не исключает возможности построения обфускатора, стойкого в модели виртуального «черного ящика». В 2013 году была предложена новая конструкция [6], так называемая «обфускация неразличимости» (indistinguishable obfuscation-*iO*).

Определение 2. Обфускатором неразличимости O называется однородная полиномиальная вероятностная машина Тьюринга для булевых схем класса $\{C_\lambda\}$, где λ – параметр безопасности, если обеспечивается выполнение следующих требований [6]:

– функциональная эквивалентность. Существует пренебрежимо малая функция $\mu(\lambda)$ такая, что для всех $\lambda \in$ и всякой схемы $C \in C_\lambda$:

$$\forall x: \Pr[iO(C(x)) = C(x)] \geq 1 - \mu(\lambda);$$

– свойство виртуального «черного ящика» (стойкость). Для любой PPT $Dist$ (распознавателя), существует пренебрежимо малая функция μ такая, что для всех $\lambda \in$ и любой пары эквивалентных схем $C_1, C_2 \in C_\lambda$, имеющих одинаковый размер, распределения вероятностей случайных величин $O(C_1)$ и $O(C_2)$ вычислительно неразличимы, то есть выполняется соотношение:

$$\forall x: |Pr[D(iO(\lambda, C_1(x))) = 1] - Pr[D(iO(\lambda, C_2(x))) = 1]| \leq \mu(\lambda).$$

Это означает, что не существует алгоритма распознавания обфускации более эффективного, чем обычное предположение, сделанное на основе анализа входов и выходов обфусцированной программы (функции или обфусцированного набора инструкций).

Неразличимость обфусцированных программ означает, что если две эквивалентные программы (одинакового размера) P_1, P_2, такие, что $\forall x: P_1(x) = P_2(x)$, а O – это обфускатор неразличимости, который принимает на вход программу $P(x)$ и на выходе генерирует новую программу $O(P(x))$, то $iO(P_1(x))$ и $iO(P_2(x))$ будут вычислительно неразличимы:

$$\exists P_1(x), P_2(x). \forall x: P_1(x) = P_2(x) \rightarrow \rightarrow iO(P_1(x)) \approx iO(P_2(x)).$$

На основе данной конструкции можно построить доказательство о том, что возможно построение неразличимого обфускатора для всех классов булевых функций, стойкого в модели виртуального «черного ящика». Рассмотрим основу построения обфускатора неразличимости [6, 7]. Обфускатор O принимает в качестве входных данных программу f, представленную в виде булевой функции, и на выходе генерирует обфусцированную программу $O(f)$. Формально, обфускатор неразличимости O можно считать компилятором, принимающим на вход булеву функцию $f(x)$ и генерирующим на выходе обфусцированную программу $f'(x) = O(f(x))$. При этом должно выполняться следующее условие: $\forall x: f'(x) = f(x)$. Определение неразличимой обфускации предъявляет следующее требование по стойкости: для любой PPT A (противника) существует вычислительно неограниченный симулятор Sim, такой, что для каждой схемы C, за полиномиальное время распознаватель $Dist$ не может различить результат, вычисленный противником A (A принимает обфусцированную программу $O(C)$ в качестве входа), от результата, вычисленного симулятором Sim, имеющим лишь оракульный доступ к C, при этом Sim может сделать неограниченное количество запросов к C. Определение стойкости обфускации в модели виртуального «черного ящика» требует, чтобы для любой PPT A (противника) существовал полиномиальный симулятор Sim, такой, что для каждой схемы C, за полиномиальное время распознаватель $Dist$ не может различить результат, вычисленный противником A (A принимает обфусцированную программу $O(C)$ в качестве входа), от результата, вычисленного симулятором Sim, имеющим лишь оракульный доступ к C, при этом Sim может сделать полиномиальное количество запросов к C.

2. Доказательство стойкости неразличимой обфускации в модели виртуального «черного ящика»

Для описания доказательства стойкости разработанной модели относительно большого класса алгебраических атак в обобщенной модели полилинейного отображения необходимо ввести ряд обозначений:

U – множество универсума;

$S \subseteq U$ – совокупность множеств индексов, являющихся подмножествами универсума U;

Z_p – кольцо вычетов по модулю простого числа p;

λ – параметр безопасности;

p – открытые параметры;

sk – секретные параметры;

n – длина ветвящейся программы;

$inp(i)$ – функция выбора позиции матриц для текущего входного бита;

l – длина входного вектора;

w – ширина ветвящейся программы.

Для любого элемента $x \in Z_p$ обозначим $[x]_S$ как разрешенное кодирование x относительно множества S [4]. Система дифференциального кодирования, реализующая полилинейное отображение [8, 9] реализует следующие операции:

– генерация параметров:

$$MM.Setup(U, 1^\lambda) \to (pp, sk);$$

– кодирование:

$$MM.Encode(sp, x, S) \to [x]_S;$$

– сложение:

$$MM.Add([x]_S, [y]_S) \to [x+y]_S;$$

– умножение:

$$MM.Mult([x]_{S_1}, [y]_{S_2}) \to$$

$$\to \begin{cases} [xy]_{S_1 \cup S_2}, \text{если } S_1 \cap S_2 = \varnothing, S_1 \cup S_2 \subseteq U, \\ \perp; \end{cases}$$

– проверка кодирования нуля:

$$MM.ZeroTest([x]_S) \to$$

$$\to \begin{cases} 0, \text{если } S = U, x = 0 \in Z_p, \\ 1. \end{cases}$$

Предположим, что противник имеет доступ к некоторому оракулу $\Re$. Одним из возможных вариантов доказательства стойкости криптографических преобразований является применение модели случайного оракула [10], под оракулом понимается вспомогательный алгоритм, который выполняется за один шаг работы основного алгоритма. Другими словами, при оценке трудоемкости атак противника время работы оракула не учитывается, учитывается только число обращений к нему. Входные данные, которые передаются оракулу, называются запросом, выходные — ответом. При каждом новом запросе значение функции на данном аргументе выбирается случайным образом. При этом оракул запоминает пару (аргумент, значение) и при повторном запросе для этого аргумента, независимо от того, кто из участников его выдал, будет возвращено все то же запомненное значение. Противник может задавать оракулу любые запросы, получать и анализировать ответы. В обобщенной модели полилинейных отображений противник может выполнять любые операции. При этом случайный оракул $\Re$ возвращает так называемый обработчик h, который однозначно сопоставляется переданным параметрам операции. Оракул $\Re$ запоминает входные данные, генерируемое значение и инициализирует внутреннюю таблицу T (для хранения обработчиков h).

В идеализированной модели полилинейных отображений [11] под элементом e будем понимать пару (α, S). При этом $\alpha(e)$ – значение элемента ($\alpha \in R$), $S(e)$ – индекс элемента ($S \subseteq U$). При выполнении операций $\circ \in \{+, -, \cdot\}$ над элементами e_1, e_2, которые содержат формальные переменные, результат операции $e_1 \circ e_2$ представляет собой формальное арифметическое выражение $\alpha(e_1) \circ \alpha(e_2)$ (при этом считаем, что индексы элементов удовлетворяют требованиям осуществления операции). Формальные выражения представляются в виде булевых схем, это позволяет гарантировать, что размер представления остается полиномиальным. Элемент e является базовым, если его значение является выражением, которое не содержит оператор (формальная переменная, которая не является вершиной графа). Элемент e' является подэлементом элемента e, если e может быть выведен из e' с помощью последовательности операций.

Оракул $\Re$ отвечает при каждом запросе следующим образом:

– при запросе $Encode(x, S)$ оракул $\Re$ осуществляет поиск ячейки в таблице T, соответствующей паре $e = (x, S)$. Если такой ячейки в таблице нет, то оракул $\Re$ генерирует обработчик h и добавляет ячейку $h \mapsto e$ в таблицу T, и возвращает h.

– при запросе $Add(h_1,h_2)$, где $h_1 \mapsto e_1$, $h_2 \mapsto e_2$, $S_1 = S_2 = S \subseteq U$, оракул $\Re$ осуществляет поиск ячейки в таблице T, соответствующей обработчикам h_1, h_2. Если такой ячейки в таблице нет, то оракул $\Re$ генерирует обработчик h и добавляет ячейку $h \mapsto (x_1 + x_2, S)$ в таблицу T, и возвращает h.

– при запросе $Mult(h_1,h_2)$ где $h_1 \mapsto e_1$, $h_2 \mapsto e_2$, $S_1 \cap S_2 = \varnothing$, $S_1 \bigcup S_2 \subseteq U$, оракул $\Re$ читает из памяти и находит ячейку в таблице T, соответствующую обработчикам h_1, h_2. Если такой ячейки в таблице нет, то оракул $\Re$ генерирует обработчик h и добавляет ячейку $h \mapsto (x_1 x_2, S_1 \bigcup S_2)$ в таблицу T, и возвращает h.

– при запросе $ZeroTest(h)$, где $h \mapsto (x,U)$, оракул $\Re$ возвращает ноль, если $x \equiv 0 (mod\ p)$ и единицу – в других случаях.

При этом будем считать, что противник или распознаватель $Adv(A)$ выполняет вероятностный алгоритм A, имеющий доступ к случайному оракулу $\Re$. Работа алгоритма заканчивается формированием искомого запроса $ZeroTest(h)$, такого что при обращении к оракулу $\Re$ с этим запросом он возвращает ноль. Пусть $P[Adv(A)]$ – вероятность того, что будет получен ответ – ноль. При обосновании стойкости требуется доказать, что для любого противника $Adv(A)$ с полиномиальными ограничениями на его ресурсы, вероятность $P[Adv(A)]$ пренебрежимо мала.

Для доказательства стойкости обфускатора O в модели виртуального «черного ящика» необходимо построить вероятностный алгоритм B (будем обозначать его симулятором Sim), которому на вход даются параметры булевой схемы $1^{|C|}$, описание алгоритма работы противника A и оракульный доступ к схеме C. Симулятор Sim начинает работу с эмуляции алгоритма обфускации O. Следует отметить, что O преобразует схему C в ветвящуюся программу B. Однако, поскольку симулятор Sim, имеет лишь оракульный доступ к схеме C, он не может вычислить матрицы $B_{i,b}$. Следовательно симулятор Sim не может моделировать список исходных элементов оракулу $\Re$. Вместо этого симулятор Sim инициализирует $\Re$ формальными переменными.

Для моделирования обфускации O симулятор Sim должен эмулировать оракул $\Re$ к которому обращается O. Симулятор Sim может эффективно эмулировать все операции оракула $\Re$ кроме проверки кодирования нуля. Проблема моделирования операции $ZeroTest$ заключается в том, что симулятор Sim не может проводить тестирование, если значение формального выражения равно 0. Однако важно отметить, что обфускатор O не осуществляет запросов к оракулу $\Re$ для выполнения операции $ZeroTest$ (данные запросы выполняются лишь на этапе вычисления обфусцированной функции).

Результатом работы симулятора Sim по моделированию O является обфусцированная схема $\tilde{O}(C)$. Затем симулятор Sim переходит к моделированию действий противника $Adv(A)$. Когда $Adv(A)$ делает запрос к оракулу, который не является вызовом операции $ZeroTest$, симулятор Sim моделирует ответ оракула. Работа оракула $\Re$ с обфускатором и противником осуществляется на основе одних и тех же таблиц обработки.

Имея обработчик h некоторого элемента e, симулятор Sim проверяет, является ли значение элемента равным нулю по двум составляющим. В первой части симулятор Sim декомпозирует элемент в виде суммы полиномиального числа более простых элементов, которые будем называть одновходными элементами. Каждый одновходный элемент имеет значение, которое зависит от подмножества формальных переменных, которые соответствуют конкретному входу ветвящейся программы. Основная сложность на первом этапе заключается в доказательстве, того что число одновходных элементов разложения является полиномиальным.

Во второй части, симулятор Sim моделирует значение каждого одновходного элемента отдельно. Основная цель данного этапа заключается в том, чтобы показать, что значение одновходного элемента для входа x может быть рассчитано только, когда задана схема $C(x)$. Для этого будем использовать теорему Килиана для рандомизации матричных ветвящихся программ [12]. Оракульный доступ не предусматривает возможность проверки всех одновходных элементов одновременно. При этом корректным является расчет каждого одновходного элемента по отдельности. Поэтому будем считать, что значение элемента равно нулю в случае, если каждый одновходный элемент в разложении равен нулю.

2.1. Декомпозиция элементов на одновходные элементы

Итак, докажем, что каждый элемент e может быть разложен на полиномиальное число одновходных элементов. Для каждого элемента e зададим следующее описание $prof(e) \in \{0,1,*\}^{\ell} \bigcup \{\perp\}$. Если представить

e как промежуточный элемент вычисления ветвящейся программы для некоторого входного x длиной l, то $prof(e)$ представляет собой частичную информацию, которая может быть получена для x, основанная на формальных переменных, которые присутствуют в e. Формально, для каждого элемента e и каждого $j \in [\ell]$, имеем, что бит j описания e равен значению $b \in \{0,1\}$, если e имеет базовый подэлемент e' такой, что $S(e') = S(i,b)$ и $j = inp(i)$.

Для $\forall j \in [\ell]$ и $b \in \{0,1\}$, считаем, что:

– $prof(e)_j = b$;

– $prof(e)_j = *$, если $prof(e)_j$ не равен 0 или 1;

– $prof(e) = \perp$, если существует значение $j \in [\ell]$, такое что $prof(e)_j$ может быть равен 0 и 1 одновременно. В этом случае e не является одновходным элементом и такое описание недопустимо. Следовательно e является одновходным элементом, если $prof(e) \neq \perp$. Если существует $prof(e) \in \{0,1\}^{\ell}$, то описание $prof(e)$ является полным.

Далее опишем алгоритм D, используемый симулятором Sim для разложения элементов e на одновходные элементы s:

$$e = \sum_{s \in D(e)} s,$$

где $D(e)$ – совокупность одновходных элементов, $s \in D(e), S(s) = D(e)$.

Алгоритм разложения D определяется следующим образом:

– если элемент e является базовым, то выходом алгоритма D будет множество с единственным элементом $\{e\}$. Пусть $S(e) = S(i,b_1,b_2)$. Тогда $prof(e)_{inp_1(i)} = b_1$, $prof(e)_{inp_2(i)} = b_2$ и $prof(e)_j = *$ для $\forall j \in [\ell], j \neq inp_1(i), j \neq inp_2(i)$;

– если элемент e выполняет операцию сложения $e_1 + e_2$, то D вычисляется рекурсивно $L_1 = D(e_1)$ $L_2 = D(e_2)$ и выдает $L = L_1 \bigcup L_2$. Если существуют элементы $s_1, s_2 \in L$ с одинаковым описанием, D заменяет два элемента одним элементом $s = s_1 + s_2$ и $prof(s) = prof(s_1)$. Этот процесс повторяется итеративно до тех пор, пока все описания в L не будут различными;

– если элемент e выполняет операцию умножения $e_1 \cdot e_2$, то D вычисляется рекурсивно $L_1 = D(e_1)$ $L_2 = D(e_2)$. Для каждого $s_1 \in L_1$ и $s_2 \in L_2$ D добавляет выражение $s = s_1 \cdot s_2$ к выходному множеству L и вычисляет $prof(s) = prof(s_1) \cdot prof(s_2)$. Затем D устраняет повторяющиеся описания из L, как было описано ранее.

На основе предложенного выше алгоритма произведем доказательство ряда утверждений.

Утверждение 1. Если $prof(e)_j = b$, то существует базовый подэлемент e' элемента e, такой что $S(e') \bigcap U_j = S^j_{i,b}$ для некоторого $i \in ind(j)$.

Доказательство утверждения 1. Если $prof(e)_j = b$, то (по определению) один из базовых подэлементов e является элементом e' таким, что $S(e') = S(i,b)$ и $j = inp(i)$. При этом: $S(e') = S(i,b_1,b_2) = S^{inp_1(i)}_{i,b_1} \bigcup S^{inp_2(i)}_{i,b_2}$. Если $j = inp_1(i)$ и $b_1 = b$, то $S(e') \bigcap U_j = S^{inp_1(i)}_{i,b_1} = S^j_{i,b}$. Аналогичным образом, если $j = inp_2(i)$ и $b_2 = b$, тогда $S(e') \bigcap U_j = S^{inp_2(i)}_{i,b_1} = S^j_{i,b}$.

Утверждение 2. Если e' является подэлементом e и $C' \subseteq S^j$ является точным покрытием $S(e') \bigcap U_j$, то существует точное покрытие $C \subseteq S^j$ $S(e) \bigcap U_j$ такое, что $C' \subseteq C$.

Доказательство утверждения 2. Докажем утверждение по методу индукции. Если e имеет вид $e_1 + e_2$ и $C_1 \subseteq S^j$ является точным покрытием $S(e_1) \bigcap U_j$, то C_1 также является точным покрытием $S(e) \bigcap U_j$ поскольку $S(e) = S(e_1)$. Аналогичным образом, если e является элементом вида $e_1 \cdot e_2$ и $C_1, C_2 \subseteq S^j$ являются точными покрытиями $S(e_1) \bigcap U_j$ и $S(e_2) \bigcap U_j$, соответственно. Тогда поскольку $S(e_1) \bigcap S(e_2) = \varnothing$ и $S(e_1) = S(e_1) \bigcup S(e_2)$, то $C_1 \bigcup C_2$ является точным покрытием $S(e) \bigcap U_j$.

Утверждение 3. Если $U \subseteq S(e)$, то все элементы в $D(e)$ являются одновходными элементами. То есть для $\forall s \in D(e)$ имеем $prof(s) \neq \perp$.

Доказательство утверждения 3. Докажем утверждение от противного. Пусть e^{bad} – первый подэлемент e такой, что $D(e^{bad})$ содержит элементы с некорректным описанием. То есть $D(e^{bad})$ содержит элемент с некорректным описанием, но разложение всех подэлементов e^{bad} содержит только элементы с корректным описанием.

Следует отметить, что e^{bad} не может быть базовым элементом, так как его описание является корректным, и $D(e^{bad})$ является множеством с одним элементом $\{e^{bad}\}$. Также e^{bad} не может иметь вид выражения $e_1 + e_2$, так как в этом случае, описание любого элемента в $D(e^{bad})$ будет также в $D(e_1)$ или $D(e_2)$, что противоречит предположению о e^{bad}. Таким образом, e^{bad} должен иметь вид $e_1 \cdot e_2$. Тогда по определению должны существовать $s_1 \in D(e_1)$ и $s_2 \in D(e_2)$, такие, что $prof(s_1) \neq \perp$ и $prof(s_2) \neq \perp$, но $prof(s_1 \cdot s_2) = \perp$. Таким образом, существует $j \in [\ell]$ такой, что $prof(s_1)_j = 0$ и $prof(s_2)_j = 1$. Из утверждений 1 и 2, следует что существует точное покрытие $S(s_1) \bigcap U_j$, которое содержит множество $S^j_{i,0}$ для некоторого $i \in ind(j)$, и существует точное покрытие $S(s_2) \bigcap U_j$, которое содержит множество $S^j_{i',1}$ для некоторого $i' \in ind(j)$. Так как $S(s_1) = S(e_1)$, $S(s_1) = S(e_2)$ и e^{bad} имеет вид $e_1 \cdot e_2$, существует точное покрытие $S(e) \bigcap U_{j'}$, которое содержит $S^j_{i,0}$ и $S^j_{i',1}$. Поскольку e^{bad} является подэлементом e, то как следует из утверждения 2, существует точное покрытие $S(e) \bigcap U_j$, которое содержит как $S^j_{i,0}$, так и $S^j_{i',1}$. Однако, поскольку $U_j \subseteq U \subseteq S(e)$ имеем, что $S(e) \bigcap U_j = U_j$. Это означает, что существует точное покрытие U_j, которое содержит как $S^j_{i,0}$, так и $S^j_{i',1}$. Пришли к противоречию относительно определения системы разделенных множеств S_j [4].

Утверждение 4. Алгоритм D работает за полиномиальное время, и количество элементов в разложении $D(e)$ также является полиномиальным.

Доказательство утверждения 4. Покажем, что время работы алгоритма D при разложении элемента e полиномиально зависит от мощности множества $|D(e)|$. $|D(e)|$ – полиномиально, относительно утверждения, что $\forall s \in D(e)$ существует одновходный подэлемент e' элемента e, такой что $prof(s) = prof(e')$. Поскольку описания в $D(e)$ являются различными, $|D(e)|$ ограничено числом подэлементов элемента e и, следовательно, $|D(e)|$ является полиномиальным.

Зафиксируем некоторое $s \in D(e)$. Пусть e_0 – первый подэлемент элемента e такой, что $D(e_0)$ содержит элемент с описанием $prof(s)$. А именно, предположим, что $D(e_0)$ содержит элемент с описанием $prof(s)$, но разложение любого подэлемента e_0 не содержит элемент с описанием $prof(s)$.

Если e_0 является базовым, то e_0 также является одновходным элементом, так как $prof(e_0) = prof(s)$. Следует отметить, что e_0 не может иметь вид $e_1 + e_2$, поскольку описание любого элемента $D(e_0)$ появляется также в $D(e_1)$ или $D(e_2)$, что противоречит предположению о e_0. Поэтому, предположим, что e_0 не является базовым, тогда он должен быть иметь вид $e_1 \cdot e_2$. Из этого следует, что в этом случае e_0 является одновходным подэлементом элемента e (т.е., что $|D(e)| = 1$) и $prof(s) = prof(e_0)$.

Предположим от противного, что $|D(e)| > 1$. По определению D, и предположению, что $e_0 = e_1.e_2$, $|D(e_1)| = 1$ или $|D(e_2)| = 1$. Пусть $|D(e_1)| > 1$ и элемент $s_0 \in D(e_0)$ такой, что $prof(s) = prof(s_0)$, и пусть $s_1 \in D(e_1)$ и $s_2 \in D(e_2)$ – элементы, такие, что $s_0 = s_1 \cdot s_2$. Тогда из выражения $|D(e_1)| > 1$ следует, что существует $s'_1 \in D(e_1)$ такой, что $s'_1 \neq s_1$ и $S(s'_1) = S(s_1) = S(e_1)$. Так как все описания в $D(e)$ различны, значит существует некоторый $j \in [\ell]$ такой, что $prof(s_1)_j \neq prof(s'_1)$. Предположим, что $prof(s_1)_j = 1$ и $prof(s'_1)_j \in \{0, *\}$.

Если $prof(s'_1)_j = *$, то для любого подэлемента e' элемента s'_1 имеем $S(e') \bigcap U_j = \varnothing$, а следовательно, $S(s'_1) \bigcap U_j = \varnothing$. С другой стороны, так как $prof(s_1)_j = 1$, то по утверждению 1 существует подэлемент e' элемента s'_1 такой, что $S(e') \bigcap U_j = \varnothing$ и $S(s_1) \bigcap U_j = \varnothing$, что противоречит факту $S(s'_1) = S(s_1)$.

Если $prof(s'_1)_j = 0$ и $prof(s_1)_j = 1$, то из утверждений 1 и 2, следует, что существует точное покрытие $S(s_1) \bigcap U_j$, которое содержит множество $S^j_{i,1}$ для некоторого $i \in ind(j)$, и существует точное покрытие $S(s'_1) \bigcap U_j$, которое содержит множество $S^j_{i',1}$ для некоторого

$i' \in ind(j)$. Из $S(s_1') = S(s_1)$ следует, что $S(s_1') \bigcap U_j = S(s_1) \bigcap U_j$, которое относительно определения системы разделенных множеств [2] означает, что $S(s_1) \bigcap U_j = U_j$.

Исходя из вышесказанного следует, что если $U_j \subseteq S(s_1)$ то описание $prof(s_1)$ является полным. Если $prof(s_1)$ является полным, то умножение его с другим элементом не может изменить его описание (не нарушив его корректность) и, следовательно, $prof(s_0) = prof(s_1 \cdot s_2) = prof(s_1)$ что противоречит нашему предположению о e_0.

Утверждение 5. Пусть s – одновходный элемент. Если $U_j \subseteq S(s)$, то:

– описание $prof(s)$ является полным;

– $\forall i \in ind(j)$ существует базовый подэлемент e_i элемента s такой, что $S(e_i) = S(i, b_1^i, b_2^i)$ для $b_1^i = prof(s)_{inp_1(i)}$ и $b_2^i = prof(s)_{inp_2(i)}$;

– если e является базовым подэлементом элемента s и $S(e) = S(i, b_1, b_2)$, то $(b_1, b_2) = (b_1^i, b_2^i)$.

Доказательство утверждения 5. Тот факт, что s – одновходный элемент, означает, то $prof(s) \neq \perp$. Кроме того, $prof(s)_j \neq *$, любой подэлемент e' элемента s удовлетворяет $S(e') \bigcap U_j = \varnothing$, а следовательно, и $S(s) \bigcap U_j = \varnothing$, что противоречит предположению, что $U_j \subseteq S(s)$.

По определению $prof(s)$, вместе с предположением о том, что $prof(s) \neq \perp$, если существует подэлемент e' элемента s такой, что $S(e') \bigcap U_j = S^j_{i',b}$ для некоторого $i' \in ind(j)$, то $b = prof(s)_j$. Таким образом, тот факт, что $U_j \subseteq S(s)$ означает, что $\forall i \in ind(j)$ существует базовый подэлемент e_i элемента s такой, что $S(e_i) \bigcap U_j = S^j_{i',b}$ для $b = prof(s)_j$. В частности, поскольку e_i является базовым, он должен удовлетворять условию $S(e_i) = S(i, b_1, b_2)$ для $(b_1, b_2) \in \{0,1\}$. Это означает, что $(b_1, b_2) = (prof(s)_{inp_1(i)}, prof(s)_{inp_2(i)})$. Это доказывает второе условие утверждения. Также из $prof(s) \neq \perp$ следует, что для любого базового подэлемента e элемента s, если $S(e_i) = S(i, b_1, b_2)$, то $(b_1, b_2) = (prof(s)_{inp_1(i)}, prof(s)_{inp_2(i)})$, что доказывает третье условие утверждения.

Для доказательства первого условия следует зафиксировать значение $j' \in [\ell]$. По предположению о структуре ветвящейся программы [4] существует значение $i \in [n]$ такое, что $\{inp_1(i), inp_2(i)\} = \{j, j'\}$. Предположим, что $\{inp_1(i), inp_2(i)\} = \{j, j'\}$. При этом $inp_1(i) = j$ означает, что $i \in ind(j)$, и существует подэлемент e_i элемента s такой, что $S(e_i) = S(i, b_1, b_2)$. Так как $inp_2(i) = j'$, $prof(s)_{j'} \neq *$. Это верно для любого $j' \in [\ell]$, и, следовательно, описание $prof(s)$ является полным.

На основе совокупности доказанных утверждений и алгоритма декомпозиции D предлагается подход к моделированию запросов $ZeroTest$. Для этого воспользуемся методом моделирования Килиана для рандомизированной ветвящейся программы [12].

Теорема Килиана. Существует эффективный алгоритм моделирования S_{BP}, такой что $\forall x \in \{0,1\}^\ell$ выполняется требование:

$$\{R_0, R_n, \{\widetilde{B}_{i,b_1^i,b_2^i} : i \in [n], b_1^i = x_{inp_1(i)}, b_2^i = x_{inp_2(i)}\}\} \approx S_{BP}(1^n, BP(x)).$$

2.2. Моделирование процедуры проверки кодирования нуля

Пусть e – элемент, относительно которого будут осуществляться запросы $ZeroTest$, p_e – полином, описывающий значение e как функцию от формальных переменных, вычисляемый по схеме $\alpha(e)$. Для запроса $ZeroTest$, содержащего элемент e и $S(e) = U \bigcup U_s \bigcup U_t$ симулятор Sim обрабатывает запрос следующим образом:

1. Симулятор Sim начинает работу с разложения элемента e на одновходные элементы s так, чтобы $e = \sum_{s \in D(e)} s$. Обозначим $D(e) = \{s_1, \ldots, s_m\}$ как совокупность одновходных элементов. Симулятор Sim на выходе выдает значение 0, если $e = 0$, или 1 – в других случаях.

2. Для всех $s \in D(e)$ выполняются следующие этапы:

– симулятор Sim отправляет запросы оракулу C, моделирующему схему, выбирает x, соответствующий $prof(s)$, для вычисления $C(x)$;

– симулятор Sim запускает симулятор S_{BP}, на вход которого подается $C(x)$, и получает следующие значения:

$$R_0, R_n, \{\widetilde{B}_{i,b_1^i,b_2^i} : i \in [n], b_1^i = x_{inp_1(i)}, b_2^i = x_{inp_2(i)}\};$$

– симулятор Sim производит выборку равномерно распределенных случайных величин V_s^{Sim}:

$$s, t, \{\alpha_{i,b_1^i,b_2^i} : i \in [n], b_1^i = x_{inp_1(i)}, b_2^i = x_{inp_2(i)}\};$$

– симулятор Sim на выходе выдает значение 1, если $p_s(V_s^{Sim}) \neq 0$, где p_s – полином, вычисляемый по схеме $\alpha(s)$ по V_s^{Sim}.

3. Если для всех $s \in D(e)$ выполняется равенство $p_s(V_s^{Sim}) = 0$, то симулятор Sim на выходе выдает 0.

Идея моделирования заключается в том, что для каждого одновходного элемента моделируется процедура $ZeroTest$ отдельно. Чтобы доказать адекватность предложенной модели, необходимо показать, что значение элемента становится равным нулю в результате взаимоисключения двух или более ненулевых одновходных элементов с пренебрежимо малой вероятностью. Поэтому значение элемента становится равным нулю с вероятностью близкой к единице в случае, если значения всех одновходных элементов в разложении равны нулю.

Докажем, что моделирование $ZeroTest$ запросов является статистически близким к распределению в реальном обфускаторе. Формально, пусть V_C^{real} – случайная величина, представляющая значения исходных элементов, которые обфускатор $O(C)$ дает оракулу $\Re$ во время инициализации (считаем, что V_C^{real} – значения переменных полинома p_e). Требуется, чтобы для любого элемента e, такого что $S(e) = U \bigcup B_s \bigcup B_t$, вероятность того, что Sim выдает значение элемента равным нулю пренебрежимо близка к вероятности того, что $p_e(V_C^{real}) = 0$. Из этого, в свою очередь, будет следовать адекватность моделирования в целом, так как противник может осуществить полиномиалное число запросов процедуры $ZeroTest$.

Утверждение 6. Для любого элемента e, такого что $U \subseteq S(e)$, полином p_e, вычисляемый по схеме $\alpha(e)$, можно представить в виде:

$$p_e = \sum_{s \in D(e)} p_s = \sum_{s \in D(e)} q_{prof(s)} \cdot \widetilde{\alpha}_{prof(s)}.$$

При этом для любого одновходного элемента $s \in D(e)$ выполняются следующие условия:

– $\widetilde{\alpha}_{prof(s)} = \prod_{i=1}^{n} \alpha_{i,b_1^i,b_2^i}$,

где $(b_1^i, b_2^i) = (prof(s)_{inp_i(i)}, prof(s)_{inp_2(i)})$;

– $q_{prof(s)}$ является полиномом от переменных $\widetilde{s}, \widetilde{t}$ и элементов матриц $\widetilde{B}_{i,b_1^i,b_2^i}$, где степень каждой переменной равна единице.

Доказательство утверждения 6. По свойству декомпозиционного алгоритма D имеем, что $p_e = \sum_{s \in D(e)} p_s$. Пусть p_s – полином, вычисляемый по арифметической схеме $\alpha(s)$, где элемент $s \in D(e)$. Предположим, что одновходный элемент s можно представить в виде суммы одночленов (возможно экспоненциальной), а именно $s = \sum_k s_k$. При этом, s_k удовлетворяет следующим свойствам:

– $\forall s_k : S(s_k) = S(s)$ и, следовательно, $U_j \subseteq S(s_k)\ \forall j \in [l]$;

– $\alpha(s)$ содержит только операции умножения;

– базовые подэлементы каждого s_k являются подмножеством базовых подэлементов s.

По утверждению 5, описание $prof(s_k)$ является полным, и, поскольку любой базовой подэлемент s_k также является базовым подэлементом s, то $prof(s_k) = prof(s)$. Кроме того, для $\forall i \in [l]$, существует базовый подэлемент e' элемента s_k такой, что $S(e') = S(i, b_1^i, b_2^i)$ для $b_1^i = prof(s_k)_{inp_1(i)}, b_2^i = prof(s_k)_{inp_2(i)}$. Второе свойство s_k означает, что s_k имеет только один базовый подэлемент с множеством индексов $S(i, b_1^i, b_2^i)$ для $\forall i \in [l]$. Только базовые элементы, известные противнику как часть обфускации с множеством индексов $S(i, b_1^i, b_2^i)$ являются α_{i,b_1^i,b_2^i} и $\alpha_{i,b_1^i,b_2^i} \cdot \widetilde{B}_{i,b_1^i,b_2^i}$. Из выше сказанного справедливо, что для любого s_k полином p_s может представить в виде выражения $q_{prof(s)} \cdot \widetilde{\alpha}_{prof(s)}$.

Утверждение 7. Для любого одновходного элемента s такого, что $U \subseteq S(s)$ значения переменных V_s^{Sim}, сгенерированные симулятором Sim, и значения, присвоенные V_C^{real}, имеют одинаковые распределения.

Доказательство утверждения 7. По теореме Килиана, распределения следующих величин, генерируемых симулятором Sim и O, равны:

$$R_0, R_n, \{\tilde{B}_{i,b_1^i,b_2^i} : i \in [n], b_1^i = prof(s)_{inp_1(i)}, b_2^i = prof(s)_{inp_2(i)}\}.$$

Кроме того, следующие величины генерируются случайным образом симулятором Sim и обфускатором O:

$$s, t, \{\alpha_{i,b_1^i,b_2^i} : i \in [n], b_1^i = prof(s)_{inp_1(i)}, b_2^i = prof(s)_{inp_2(i)}\}.$$

Данное утверждение следует из того, что значения V_s^{Sim} генерируются симулятором Sim, а значения V_C^{real} – обфускатором, на основе указанных выше исходных данных по одному и тому же алгоритму.

Перейдем к доказательству адекватности моделирования процедуры $ZeroTest$. Пусть на вход $ZeroTest$ подается элемент e такой, что $S(e) = U \bigcup B_s \bigcup B_t$. Считаем, что $p_e(V_C^{real}) \equiv 0$, если значения всех одновходных элементов разложения равны нулю.

Для случая, когда $p_e(V_C^{real}) \equiv 0$ и, поскольку предельное распределение значений α_{i,b_1^i,b_2^i} в V_C^{real} является равномерным, из структуры p_e (по утверждению 6) и на основании леммы Шварца-Зиппеля [13], для каждого $s \in D(e)$ имеем, что $q_{prof(s)}(V_C^{real}) \equiv 0$, и, следовательно, $p_s(V_C^{real}) \equiv 0$. По утверждению 7, так как $p_s(V_s^{Sim}) \equiv 0$ для всех $s \in D(e)$, следовательно, Sim ответит, что e равно нулю с вероятностью единица. Для случая, когда $p_e(V_C^{real}) \neq 0$ будем использовать следующее утверждение.

Утверждение 8. Для любого элемента e такого, что p_e является полиномом степени $poly(n)$, если $p_e(V_C^{real}) \neq 0$, то:

$$\Pr_{V_C^{Real}}[p_e(V_C^{real}) = 0] = \mu(n),$$

где $\mu(n)$ – пренебрежимо малая функция.

Доказательство утверждения 8. Данное утверждение следует напрямую из леммы Шварца-Зиппеля, если распределение случайных переменных V_C^{real} является равномерным. Для доказательства необходимо преобразовать полином p_e в p'_e, распределение V_C^{real} в $\tilde{V}_C^{real}$ так, чтобы:

– $\Pr_{V_C^{real}}[p_e(V_C^{real}) = 0] = \Pr_{\tilde{V}_C^{real}}[p'_e(\tilde{V}_C^{real}) = 0$

– степень полинома p'_e: $deg(p'_e) = poly(n)$;

– распределение, соответствующее V_C^{real} может быть рассчитано полиномиально по значениям, распределенным равномерно над кольцом Z_p.

Для преобразования полинома p_e в p'_e, матрицы R_i^{-1} в p_e заменяются союзными матрицами $adj(R_i)\prod_{i \neq j} det(R_j)$, где $adj(R_i) = R_i^{-1} det(R_i)$. Аналогично, для преобразования распределения V_C^{real} в $\tilde{V}_C^{real}$ – распределение значений R_i^{-1} заменяется распределением значений $adj(R_i)\prod_{i \neq j} det(R_j)$.

Поскольку матрица R_i^{-1} является невырожденной и по утверждению 6 степень элементов полинома p_e в R_i^{-1} равна единице, то имеем:

$$\Pr_{V_C^{real}}[p_e(V_C^{real}) = 0] = \Pr_{V_C^{real}}[p_e(V_C^{real}) \cdot \prod_{i \in [n]} det(R_i) = 0] = \Pr_{V_C^{real}}[p'_e(\tilde{V}_C^{real}) = 0].$$

Данное выражение доказывает, что первое свойство утверждения удовлетворяется. Поскольку степень $deg(\prod_{i \in [n]} det(R_i)) \leq n \cdot w$ и, следовательно, $deg(p'_e) \leq n \cdot w \cdot deg(p_e)$. Из этого следует, что $deg(p'_e) = poly(n)$ (второе свойство утверждения). Так как элементы $adj(R_i)$ могут быть представлены в виде полиномов степени w и значений R_i, имеем, что все значения $\tilde{V}_C^{real}$ распределены равномерно над кольцом Z_p или могут быть представлены в виде полиномов степени не более w по значениям,

которые равномерно распределены над кольцом Z_p. Поскольку $deg(p'_e) = poly(n)$, по лемме Шварца-Зиппеля:

$$\Pr_{V_C^{real}}[p_e(V_C^{real}) = 0] =$$

$$= \Pr_{V_C^{real}}[p'_e(V_C^{real}) = 0] = \mu(n).$$

Возвращаясь к случаю, когда $p_e(V_C^{real}) \neq 0$, что следует из структуры p_e, заданной утверждением 6, существует $s \in D(e)$, такой что $p_s(V_C^{real}) \neq 0$. В силу утверждения 8, $Pr[p_s(V_C^{real}) \neq 0] = 1 - \mu(n)$. По утверждению 7, $p_s(V_S^{Sim}) \neq 0$ с вероятностью $Pr[p_s(V_S^{sim}) \neq 0] = 1 - \mu(n)$.

Заключение

Таким образом, совокупность доказанных утверждений, позволяет сделать вывод о том, что предложенная аналитическая модель защиты файлов документальных форматов от несанкционированного доступа, отличающаяся применением процедуры неразличимой обфускации программного кода, является стойкой в модели виртуального «черного ящика». Данный факт, в свою очередь, позволяет утверждать, что не существует алгоритма распознавания обфускации более эффективного, чем обычное предположение, сделанное на основе анализа входов и выходов обфусцированной программы. То есть задача пропозициональной выполнимости обфусцированной функции, которую решает противник, сводится к задаче тотального перебора 2^l вариантов входных данных.

Обоснованность и достоверность полученных результатов подтверждается аналитически, применением апробированного математического аппарата. Направлением дальнейших исследований является практическая реализация и экспериментальная проверка качества разработанных моделей.

Литература:

1. Козачок А. В., Туан Л. М. Обоснование возможности применения неразличимой обфускации для защиты исполняемых файлов. В сборнике: Перспективные информационные технологии (ПИТ 2015) труды Международной научно-технической конференции. СГАУ. Самара, 2015. С. 269-272.
2. Козачок А. В., Туан Л. М. Комплекс алгоритмов контролируемого разграничения доступа к данным, обеспечивающий защиту от несанкционированного доступа // Системы управления и информационные технологии. 2015. Т. 61. № 3. С. 58-61.
3. Варновский Н. П., Захаров В. А., Кузюрин Н. Н. Математические проблемы обфускации. Математика и безопасность информационных технологий. Материалы конференции в МГУ 28–29 октября 2004 г., 2005, С. 65–91.
4. Козачок, А. В., Аналитическая модель защиты файлов документальных форматов от несанкционированного доступа / А. В. Козачок, М. В. Бочков, Р. Р. Фаткиева, Л. М. Туан. // Труды СПИИРАН. 2015. № 6. С. 228-252.
5. B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. P. Vadhan, K. Yang. On the (im)possible obfuscating programs. In Advances in Cryptology – CRYPTO, 2001, pp. 1–18.
6. S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits». In FOCS, 2013, pp. 40–49.
7. P. Ananth, D. Gupta, Y. Ishai, A. Sahai. Optimizing obfuscation: Avoiding barrington's theorem. In Proceeding of the 2014 ACM SIGSAC, 2014, pp. 646–658.
8. S. Garg, C. Gentry, S. Halevi. Candidate multilinear maps from ideal lattices. In Advances in Cryptology – EUROCRYPT, 2013, pp. 1–17.
9. J. S. Coron, T. Lepoint, M. Tibouchi. Practical multilinear maps over the integers. In Advances in Cryptology – CRYPTO, 2013, pp. 476–493.
10. B. Lynn, M. Prabhakaran, A. Sahai. Positive results and techniques for obfuscation // Lecture Notes in Computer Science, v. 3027, 2004, pp. 20-39.
11. B. Barak, S. Garg, Y. T. Kalai, O. Paneth, A. Sahai. Protecting obfuscation against algebraic attacks. In Advances in Cryptology – EUROCRYPT, 2014, pp. 221–238.
12. J. Kilian. Founding cryptography on oblivious transfer. In 20th Annual ACM Symposium on Theory of Computing. 1988, pp. 20–31.
13. J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. Journal of the ACM, 1980, vol. 27, pp. 701–717.

Рецензент: Марков Алексей Сергеевич, доктор технических наук, профессор МГТУ им.Н.Э.Баумана, a.markov@bmstu.ru

 УДК 003.26 + 004.056.53

INDISTINGUISHABLE OBFUSCATION SECURITY THEORETICAL PROOF

Kozachok A.V.[4], Bochkov M.V.[5], Lai M.T.[6]

The purpose of the research is to construct complex of models and algorithms to control data access restriction to the documentary file formats based on indistinguishable program code obfuscation. The theoretical security proof of analytical model for protecting documentary file formats from unauthorized access is proposed.

Different approaches to the indistinguishable obfuscation security definition, decomposition algorithm and the approach to modeling indistinguishable obfuscator under the random oracle model are represented. Using random oracle model is one of the main directions to prove cryptographic protocol security. Oracle is auxiliary algorithm that performs in a single step of main algorithm.

The set of proposed and proved statements allowed constructing secure obfuscator satisfying virtual black box property. This fact means that no algorithm can distinguish obfuscated program more efficiently then hypothesis made by viewing inputs and outputs of obfuscated program.

Keywords: obfuscation, random oracle, decomposition, security, virtual black box

References:

1. Kozachok A. V., Tuan L. M. Obosnovanie vozmozhnosti primeneniya nerazlichimoy obfuskatsii dlya zashchity ispolnyaemykh faylov, V sbornike: Perspektivnye informatsionnye tekhnologii (PIT 2015) trudy Mezhdunarodnoy nauchno-tekhnicheskoy konferentsii. SGAU. Samara, 2015, pp. 269-272.
2. Kozachok A. V., Tuan L. M. Kompleks algoritmov kontroliruemogo razgranicheniya dostupa k dannym, obespechivayushchiy zashchitu ot nesanktsionirovannogo dostupa, Sistemy upravleniya i informatsionnye tekhnologii. 2015. T. 61. No 3, pp. 58-61.
3. Varnovskiy N. P., Zakharov V. A., Kuzyurin N. N. Matematicheskie problemy obfuskatsii, Matematika i bezopasnost' informatsionnykh tekhnologiy. Materialy konferentsii v MGU 28–29 oktyabrya 2004 g., 2005, pp. 65–91.
4. Kozachok, A. V., Analiticheskaya model' zashchity faylov dokumental'nykh formatov ot nesanktsionirovannogo dostupa / A. V. Kozachok, M. V. Bochkov, R. R. Fatkieva, L. M. Tuan., Trudy SPIIRAN. 2015. No 6, pp. 228-252.
5. B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. P. Vadhan, K. Yang. On the (im)possible obfuscating programs. In Advances in Cryptology – CRYPTO, 2001, pp. 1–18.
6. S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits». In FOCS, 2013, pp. 40–49.
7. P. Ananth, D. Gupta, Y. Ishai, A. Sahai. Optimizing obfuscation: Avoiding barrington's theorem. In Proceeding of the 2014 ACM SIGSAC, 2014, pp. 646–658.
8. S. Garg, C. Gentry, S. Halevi. Candidate multilinear maps from ideal lattices. In Advances in Cryptology – EUROCRYPT, 2013, pp. 1–17.
9. J. S. Coron, T. Lepoint, M. Tibouchi. Practical multilinear maps over the integers. In Advances in Cryptology – CRYPTO, 2013, pp. 476–493.
10. B. Lynn, M. Prabhakaran, A. Sahai. Positive results and techniques for obfuscation // Lecture Notes in Computer Science, v. 3027, 2004, pp. 20-39.
11. B. Barak, S. Garg, Y. T. Kalai, O. Paneth, A. Sahai. Protecting obfuscation against algebraic attacks. In Advances in Cryptology – EUROCRYPT, 2014, pp. 221–238.
12. J. Kilian. Founding cryptography on oblivious transfer. In 20th Annual ACM Symposium on Theory of Computing. 1988, pp. 20–31.
13. J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. Journal of the ACM, 1980, vol. 27, pp. 701–717.

4 Aleksandr Kozachok, Ph.D., The Academy of the Federal Guard Service, Orel, alex.totrin@gmail.com
5 Maxim Bochkov, Dr.Sc., Professor, Business Risk Educational Center, Saint-Petersburg, mvboch@yandex.ru
6 Minh Lai, The Academy of the Federal Guard Service, Orel, lmtuan.1989@gmail.com