Современное состояние исследований в области обфускации программ: определения стойкости обфускации

[1] *Н.П. Варновский*
[2] *В.А.Захаров <zakh@cs.msu.su>*
[3] *Н.Н. Кузюрин <nnkuz@ispras.ru>*
[3] *А.В. Шокуров <shok@ispras.ru>*
[1] *Институт проблем информационной безопасности, 119192, г. Москва, Мичуринский проспект, 1, офис 10*
[2] *Факультет ВМК, МГУ имени М.В. Ломоносова, 119991 ГСП-1 Москва, Ленинские горы, 2-й учебный корпус*
[3] *ИСП РАН, 109004, Россия, г. Москва, ул. А. Солженицына, дом 25*

Аннотация. Обфускацией программ называется такое эквивалентное преобразование программ, которое придает программе форму, затрудняющую понимание алгоритмов и структур данных, реализуемых программой, и препятствующую извлечению из текста программы определенной секретной информации, содержащейся в ней. Поскольку обфускация программ может найти широкое применение при решении многих задач криптографии и компьютерной безопасности, задаче оценки стойкости обфускации придается очень большое значение, начиная с самых первых работ в этой области. В этой статье приводится обзор различных определений стойкости обфускации программ и результатов, устанавливающих возможность или невозможность построения стойкой обфускации программ в тех или иных криптографических предположениях.

Ключевые слова: программа, обфускация, стойкость обфускации, сложность, машина Тьюринга, модель черного ящика

1. Введение

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. Обфускация программ в противоположность реорганизации (рефакторингу) преследует цель затруднить понимание программ и воспрепятствовать целенаправленной их модификации. Поэтому задачу обфускации программ можно счесть одной из задач системного

программирования, подобной другим задачам преобразования программ – трансляции, оптимизации, реорганизации, распараллеливания. С другой стороны, обфускацию можно также рассматривать как особую разновидность шифрования программ. В отличие от традиционных видов шифрования обфускация не предполагает построения эффективных алгоритмов расшифрования, т.е. восстановления исходного текста программы, но зато требует сохранения смысла зашифрованного сообщения – функции, вычисляемой обфускируемой программой. Поэтому задача обфускации программ может быть также отнесена к области криптографии и криптоанализа. Именно двойственность этой задачи и объясняет тот факт, что ее исследование вот уже более 15 лет проводится по двум направлениям – со стороны системного программирования и со стороны криптографии, – которые очень мало взаимодействуют друг с другом. Цель настоящей статьи – ознакомить специалистов в области системного программирования с теми результатами изучения задачи обфускции программ, которые были достигнуты к настоящему времени в области математических основ криптографии. Эта статья открывает цикл работ, посвященных вопросам обфускации программ, рассматриваемых с точки зрения математики. При изучении математической проблемы обфускации программ начинать нужно с определения стойкости обфускации. Требования стойкости существенно зависят от тех приложений, в которых используется обфускация. И поэтому в первой статье этого цикла мы рассмотрим и проанализируем определения понятия стойкости обфускации программ. В последующей работе планируется провести исследование тех задач системного программирования и криптографии, для решения которых можно было бы использовать обфускацию программ, и оценить, в какой мере современные методы обфускации приблизились к их решению. И, наконец, в последней статье цикла мы расскажем о новейших достижениях в решении задачи построения стойких обфускаторов программ.

Вероятно, задача обфускации была впервые упомянута (без явного употребления термина «обфускация») в 1976 году в основополагающей работе Диффи и Хеллмана [1]. Желая проиллюстрировать концепцию шифрования с открытым ключом, они предложили следующую простую схему ее реализации. Выбирается произвольная криптосистема с секретным ключом, в процедуру шифрования вставляется секретный ключ, и затем инициализированная этим ключом программа шифрования запутывается так, чтобы извлечение из ее текста секретного ключа было очень трудной задачей. Таким образом, модифицированная процедура шифрования становится открытым ключом новой криптосистемы. Запутывание процедуры шифрования с целью предотвращения извлечения из ее текста секретного ключа является одним из возможных применений обфускации программ для решения некоторых задач криптографии и компьютерной безопасности.

В явном виде понятие обфускации программ было введено в 1997 году в работе Коллберга, Томборсона и Лоу [2]. Авторы этой работы рассматривали обфускацию программ, в первую очередь, как средство защиты прав интеллектуальной собственности на алгоритмы, которые реализуются в программах с открытым кодом. В работе [2] были предложены простейшие виды обфускирующих преобразований программ, проведена их систематическая классификация и прослежена взаимосвязь задачи обфускации программ с некоторыми известными задачами системного программирования.

Если вкратце подытожить результаты исследования проблемы обфускации программ за прошедшие 18 лет, то можно сделать следующие выводы.

1. Спектр задач, для решения которых можно было бы использовать алгоритмы программной обфускации, весьма обширен, и цели применения обфускации могут быть противоположны. Обфускацию можно использовать как для защиты программ от вирусных атак [3], так и для маскировки компьютерных вирусов [4,5]. При обфускации программ для нужд криптографии целью маскировки является сокрытие данных (секретного ключа), но не алгоритмов. Но когда методы обфускации применяются для обеспечения компьютерной безопасности, то целью маскировки является сокрытие алгоритмов, но не обрабатываемых данных. Таким образом, “проблема обфускации программ” включает в себя целое семейство задач маскировки программ, для каждой из которых вводятся специальные требования стойкости обфускации.
2. Имеется большой разрыв между теоретическими требованиями стойкости обфускации программ и применяемыми на практике методами и средствами решения этой задачи. Известно немало работ, в которых предлагаются различные практические методы обфускации программ; подробное описание многих из них представлено в монографии [6]. Некоторые из этих методов были реализованы в коммерческих программных продуктах (см., например, [7-12]). Однако влияние фундаментальных теоретических результатов (см. [13-16]) на эту ветвь развития программного обеспечения минимально: требования безопасности, исследуемые в контексте криптографических приложений, либо являются слишком сильными, либо неадекватны тем задачам защиты программного обеспечения, которые возникают на практике.
3. Между положительными и отрицательными результатами решения задачи обфускации образовался большой разрыв. В статье [13] доказано, что для некоторых строго формализованных определений стойкости обфускации (стойкость в модели виртуального «черного ящика») существуют такие семейства эффективно вычислимых функций, которые не допускают стойкой обфускации. Стойкую обфускацию удалось построить для существенно более слабых требований стойкости и лишь для очень простых функций - точечных функций и близких к ним семейств функций [17-21]. Несмотря на то, что положительные результаты были обобщены в статье [22-24] для более широких классов функций, вопрос о (не)возможности эффективной обфускации для общих криптографических протоколов или же для любого значимого класса программ (например, для конечных автоматов) при стандартных криптографических предположениях остается открытым.

Приведенные выводы показывают, что проблема обфускации программ – это очень сложная и многогранная задача, для которой вряд ли удастся найти единый универсальный метод решения. Мы надеемся, что дальнейший прогресс в исследовании этой проблемы позволит сформировать математические основы для создания широкого многообразия формальных концепций и методов обфускации программ в контексте различных приложений. Создание такого математического аппарата следует начать с разработки различных определений стойкости обфускации и исследования взаимосвязи между предложенными определениями и подходящими понятиями и моделями дискретной математики, математической криптографии, теории сложности вычислений. Это поможет нам открыть наиболее важные свойства для всех типов обфускации программ. Располагая спектром различных определений стойкости обфускации, исследователям будет легче понять, какие требования безопасности обеспечивают те или иные обфускирующие преобразования и оценить, насколько эти преобразования удовлетворяют заявленным целям. Многообразие новых формальных определений стойкости может прояснить решение поставленных задач. Наконец, введение новых формальных требований стойкости обфускации программ откроет новые возможности адаптации формальных методов теоретической информатики к решению задач защиты программ.

Статьи [1] и [2] положили начало двум направлениям исследований задачи обфускации программ, а именно, обфускации программ как средства решения некоторых криптографических задач и обфускации программ как одного из методов обеспечения компьютерной безопасности. Мы уделим главное внимание первому направлению исследований. Однако для полноты картины в следующем разделе статьи мы коротко опишем основные достижения в решении задачи обфускации программ и некоторых смежных с ней задач в области системного программирования. Далее, в разделе 3 мы рассмотрим наиболее известные формальные определения стойкости обфускации программ и взаимосвязь между этими определениями.

2. Обфускация программ с позиции системного программирования

Программная обфускация могла бы быть очень полезной для решения многих задач системного программирования и компьютерной безопасности. Уже в

ранних работах, посвященных обфускации программ, было показано, что обфускирующие преобразования могут быть использованы для защиты интеллектуальной собственности в качестве средства, препятствующего восстановлению исходных алгоритмов на основе открытого программного кода [2,25-27] и удалению из программ водяных знаков (watermarkings) и «отпечатков пальцев» (fingerprintings) [6,8,28-31], для защиты программного обеспечения от атак со стороны вредоносных программ (компьютерных вирусов) и обеспечения безопасности мобильных агентов в информационных сетях [32-36], для проведения безопасного поиска в потоках данных [37], защиты баз данных [38], защиты проектных решений при проектировании микроэлектронных схем [39,40]. Оборотной стороной полезных достоинств обфускации является возможность ее использования для затруднения обнаружения вредоносных программ [4,5,41], а также создания уязвимостей в системах защиты компьютеров [42].

Впервые простейшие приемы обфускации программ были перечислены и систематизированы в работах [2,43]. Авторы этих работ пришли к ряду выводов, которые в дальнейшем определили несколько основных линий исследования задачи обфускации программ методами и средствами системного программирования и теории анализа программ.

- Целью обфускации программ является противодействие алгоритмам статического и динамического анализа программ. Поэтому качество обфускации можно оценивать экспериментально в зависимости от того, насколько результативными оказывается применение программно-инструментальных средств анализа программ к программам, подвергшимся обфускации. Таким образом, разработку средств обфускации программ целесообразно проводить в тесной взаимосвязи с совершенствованием средств деобфускации программ по принципу состязания щита и меча. Развитие этого состязания можно проследить в серии работ [44-60] и нашло воплощение в ежегодно проводимом конкурсе обфускированных программ.
- Для предварительной теоретической оценки качества обфускации можно использовать метрические характеристики, описывающие сложность устройства программ: обфускация программы должна приводить к значительному увеличению этих характеристик. Анализ качества обфускирующих преобразований на основании метрик сложности программ проводился в работах [61-65].
- Поскольку целью обфускации является противодействие алгоритмам статического анализа, то стойкость обфускирующих преобразований можно оценивать также относительно сложности тех моделей программ, которые используются в алгоритмах статического анализа, направленных на выявление и устранение последствий маскировки программного кода: обфускация считается тем более стойкой, чем более сложные модели программ необходимы для проведения деобфускирующего статического анализа программ. Исследования стойкости обфускирующих преобразований относительно методов статического анализа было проведено в работах [66-73]. Результаты этих исследований, в частности привели к созданию специализированных алгоритмов для обнаружения полиморфных компьютерных вирусов, использующих процедуры обфускации в процессе репликации [74-76].
- В статье [2] было отмечено, что ключевым элементом многих приемов обфускации программ являются т.н. непроницаемые предикаты (opaque predicates) – предикаты, постусловия которых трудно вычислить посредством алгоритмов статического анализа. В нескольких работах [29,31,77-79] было проведено изучение способов построения и оценки стойкости непроницаемых предикатов. Авторы статьи [2] также обратили внимание на то, что для повышения стойкости обфускирующих преобразований целесообразно использовать вычислительно трудные комбинаторные задачи, конструируя на их основе непроницаемые предикаты таким образом, чтобы раскрытие поведения такого предиката было бы равносильно решению указанной задачи. Этот прием был использован в работах [25,80-83] для демонстрации того, что в некоторых случаях задача деобфускации программ может оказаться вычислительно трудной.
- Задачу обфускации программ можно решать и в такой постановке, когда лишь некоторая часть программы доступна противнику для анализа и модификации, в то время как все остальные компоненты программы выполняются на защищенном вычислительном устройстве. Впервые в такой постановке задача защиты программ была рассмотрена в статье [84]; в этой статье рассматривалась модель вычислений, состоящая из защищенного устройства памяти и общедоступного процессора. Успешное решение этой задачи привело к разработке некоторых способов обфускации программ, опирающихся на аппаратную поддержку [85,86].

Результаты проведенных теоретических и экспериментальных исследований задачи обфускации программ воплотились в нескольких десятках программно-инструментальных средств защиты программ путем применения обфускирующих преобразований того или иного вида. Большая часть предложенных подходов представляют собой эвристики (некоторые из которых весьма изощренны), предназначенные для усложнения программ анализа алгоритмов. Общим их недостатком является отсутствие обоснования гарантированной стойкости. Некоторые из этих средств обфускации довольно успешно противостоят автоматическим инструментам статического анализа и декомпиляции программных кодов. Однако в случае применения методов динамического анализа программ и привлечения квалифицированных экспертов в области системного программирования стойкости существующих

средств обфускации программ оказывается уже недостаточно. Таким образом, разработанные к настоящему времени практические методы и алгоритмы обфускации программ способны затруднить (порой, весьма значительно) понимание и модификацию программ, но не могут рассматриваться в качестве средств защиты секретной информации, содержащейся в программном коде, подобных системам шифрования.

3. Обфускация программ с позиции математической криптографии

Обфускация программ имеет важные приложения в области криптографии. Уже в ранних работах [1,13] было отмечено, что обфускация программ позволяет преобразовывать криптосистемы с секретным ключом в криптосистемы с открытым ключом: в качестве открытого ключа выступает обфускированная процедура шифрования с вставленным в нее секретным ключом. При помощи обфускации программ можно также конструировать гомоморфные системы шифрования [13], функциональные системы шифрования [87], доверенные схемы перешифрования [23] и электронно-цифровой подписи [88], избавляться от модели случайного оракула при доказательстве стойкости криптографических протоколов [2], осуществлять случайную перетасовку зашифрованных сообщений в схемах тайного голосования [89], создавать схемы дезавуируемого (двусмысленного) шифрования [90,91] и односторонние функции с секретом [91]. Более подробно об этих приложениях обфускации программ в следующей статье нашего цикла. Однако для того, чтобы каждое из перечисленных приложений обладало определенной криптографической стойкостью, используемая для его построения обфускация программ также должна удовлетворять некоторым требованиям стойкости. Поэтому при исследовании проблемы обфускации программ с позиции математической криптографии требование стойкости выдвигается на первый план. В этом разделе статьи мы рассмотрим различные определения стойкости обфускации, известные в современной математической литературе и приведем некоторые основные результаты, обосновывающие возможность или невозможность построения стойких обфускирующих преобразований.

3.1. Обфускация программ в модели виртуального «черного ящика»

Впервые строгое определение обфускации программ было сформулировано в статье [13], авторы которой опирались на результаты более ранней статьи [92]. К обфускации программ предъявляются три главных требования – сохранение функциональности программы, незначительное изменение ее размера и быстродействия и требование стойкости. Обфускация программ считается стойкой в модели виртуального «черного ящика», если противник, имеющий неограниченный доступ к тексту обфускированной программы, может извлечь

из этого текста только ту информацию об исходной программе, которую можно было бы получить, проводя одни лишь тестовые эксперименты с программой без доступа к ее тексту. Для строгого математического определения обфускации программ в модели виртуального «черного ящика» нужно ввести ряд вспомогательных понятий.

В литературе по теории сложности и математической криптографии используются две общепринятые формализации понятия «вычислительная программа». Согласно одному из них программы представляются в виде машин Тьюринга, а согласно второму программы представляются в виде схем из функциональных элементов (логических схем). В тех случаях, когда сложность вычислений и размеры программ оцениваются с точностью до полиномиальных преобразований, эти два определения равноправны (см. [93]). Для единообразия мы ограничимся рассмотрением представления программ в виде машин Тьюринга. В криптографии в качестве модели противника чаще всего используются вероятностные алгоритмы, подразумевающие возможность проведения вычислений с использованием случайных величин (датчиков случайных чисел, случайных двоичных строк и т.п.). При этом вычисления противника должны завершаться за время, полиномиально зависящее от размера тех данных, которыми он располагает. Таким образом, в качестве модели противника также используются машины Тьюринга, но при этом вероятностные и ограниченные полиномиальным временем. Для обозначения этого класса машин используется аббревиатура PPT. В теории сложности вычислений наряду с обычными и вероятностными машинами Тьюринга также используются машины Тьюринга с оракулом. В качестве оракула могут выступать любые функции или предикаты, которые рассматриваются как вспомогательные средства вычисления. ТМ с оракулом F может обратиться к оракулу для вычисления значения функции $F(y)$ для произвольной строки y, записанной на специальной ленте оракула. Это значение вычисляется за один шаг независимо от сложности функции F.

Для машины Тьюринга (ТМ) π и двоичной строки x, используемой в качестве входных данных для ТМ, условимся использовать запись $|\pi|$ для обозначения размера ТМ π, запись $\pi(x)$ для обозначения результата вычисления ТМ π на входных данных x, и запись $time(\pi(x))$ для обозначения времени (числа шагов) вычисления ТМ π на входных данных x. ТМ π_1 и π_2 считаются эквивалентными, если $\pi_1(x) = \pi_2(x)$ для любых входных данных x. Функция $\nu : N \to [0,1]$ считается пренебрежимо малой, если она убывает быстрее, чем любая функция, обратная многочлену, т.е. для любого $k \in N$ существует такое $n_0 \in N$, что для всех $n > n_0$

выполняется неравенство $\nu(n) < 1/n^k$. Для обозначения произвольного полинома и произвольной пренебрежимо малой функции традиционно используются обозначения $poly(\cdot)$ и $neg(\cdot)$ соответственно.

Определение 1 (модель виртуального «черного ящика») [13]

Обфускатором в модели виртуального «черного ящика» для семейства ТМ M называется такая РРТ O, которая удовлетворяет следующим трем условиям:

1. **Функциональная эквивалентность.** Для любой ТМ π, $\pi \in M$, в результате применения обфускатора O к π строится ТМ, эквивалентная исходной машине π.
2. **Полиномиальные издержки.** Для любой ТМ π, $\pi \in M$, размер и быстродействие любой ТМ $O(\pi)$ отличается от размера и быстродействия ТМ π не более чем полиномиально, т.е. $|O(\pi)| = poly(|\pi|)$ и $time(O(\pi(x))) = poly(time(\pi(x)))$.
3. **Свойство виртуального «черного ящика».** Для любой РРТ A (противника) существует такая РРТ S (симулятор), для которой соотношение

$$|\Pr[A(O(\pi)) = 1] - \Pr[S^{\pi}(1^{|\pi|}) = 1]| \le neg(|\pi|)$$

выполняется для любой ТМ π, $\pi \in M$, причем первая вероятность вычисляется для случайных величин, используемых обфускатором O и противником A, а вторая вероятность вычисляется для случайных величин, используемых симулятором S.

Один из основных результатов работы [13] таков.

Теорема 1. [13]. Существуют такие семейства ТМ, для которых нельзя построить обфускатор в модели виртуального «черного ящика».

Идея доказательства этой теоремы проста. Рассмотрим пару ТМ $\pi_{a,b}$ и $\pi_{c,d}$. Первая из них для каждой входной строки x вычисляет на выходе строку b, если $x = 1$, и выдает на выходе нулевую строку в противном случае. Вторая из этих ТМ рассматривает входную строку x как описание некоторой программы и применяет эту программу к строке c и выдает на выходе 1, если это вычисление завершается за полиномиальное время (относительно длины x) с результатом d; в противном случае ТМ $\pi_{c,d}$ выдает на выходе 0. Из этих двух ТМ можно сформировать ТМ $\pi_{a,b,c,d}(x, y)$, которая в случае $y = 0$ проводит вычисление так же, как

ТМ $\pi_{a,b}(x)$, а в противном случае проводит вычисление так же, как ТМ $\pi_{c,d}(x)$. Целью (угрозой) противника является выяснение для заданной ТМ $\pi_{a,b,c,d}(x, y)$, верно ли, что $a = c$ и $b = d$. Этой цели легко достигает детерминированная ТМ A, которая, получив на входе обфускированную программу $\pi' = O(\pi_{a,b,c,d})$, подставляет в эту программу в качестве второй компоненты входных данных 0 и 1, а затем вычисляет значение $\pi'(\pi'(\cdot,0),1)$. Очевидно, что равенства $a = c$ и $b = d$ выполняются тогда и только тогда, когда указанное значение, вычисленное противником A, равно 1. В то же время всякий симулятор S, завершающий работу за полиномиальное время и имеющий лишь оракульный доступ к программе $\pi_{a,b,c,d}$, может правильно проверить выполнимость условий $a = c$ и $b = d$ лишь за счет случайного угадывания подходящих запросов к оракулу, т.е. с пренебрежимо малой вероятностью.

Авторы работы [13] смогли построить *наследственно необфускируемое семейство функций,* т.е. такое множество функций, что всякая программа вычисления любой из функций этого множества не допускает обфускации в модели виртуального «черного ящика». Интересно отметить, что в класс наследственно необфускируемых семейств функций включаются некоторые семейства псевдослучайных функций, функций шифрования с секретным ключом, схемы электронно-цифровой подписи. Кроме того, в статье [13] было показано, что обфускация в модели виртуального «черного ящика» невозможна для некоторых семейств программ, представимых логическими схемами из класса TC_0. Схемы этого класса строятся из пороговых элементов, и глубина схем ограничена некоторой константой.

В целом, результаты статьи [13] показывают, что задача обфускации программ не имеет простого решения: существуют такие семейства программ, вычисляющие сравнительно простые функции, для которых нельзя построить стойкую обфускацию в модели виртуального «черного ящика». В связи с этим дальнейшие исследования проблемы обфускации были сосредоточены на получении ответов на следующие вопросы. Существуют ли другие определения стойкости обфускации, которые являются менее требовательными, нежели стойкость в модели виртуального «черного ящика», но при этом позволяют находить для обфускации подходящие криптографические приложения? Для каких классов программ можно построить обфускаторы, удовлетворяющие тем или иным разумным требованиям стойкости? Каковы необходимые и достаточные условия необфускируемости программ?

3.2. Вариации модели виртуального «черного ящика»

Уже из самого определения обфускации в модели виртуального «черного ящика» видно, что оно допускает некоторые вариации за счет изменения ограничений, налагаемых на противника A и симулятор S.

Например, можно предполагать, что вычислительные возможности противника неограниченны. Тогда удается полностью описать класс программ, допускающих обфускацию. Семейство программ M называется эффективно выведываемой (learnable) [94], если существует такая PPT L, что для любой программы $\pi, \pi \in M,$ TM L^{π} выдает на выходе программу π', эквивалентную программе π. В статье [20] было установлено, что семейство программ допускает обфускацию в модели виртуального «черного ящика» с неограниченным противником в том и только том случае, когда это семейство программ эффективно выведываемо.

Можно рассмотреть такой вариант определения 1, в котором вычислительные возможности симулятора неограничены, но при этом число обращений симулятора к оракулу ограничено величиной, полиномиально зависящей от размера обфускируемой программы π. Такая разновидность обфускации была введена в статье [95]; будем называть ее обфускацией со слабо ограниченным симулятором. В этой же работе было показано, что требования стойкости обфускации в модели со слабо ограниченным симулятором существенно слабее требования стойкости обфускации в модели виртуального «черного ящика».

Теорема 2. [95]. Если существуют односторонние перестановки, то существует такое семейство TM, которое допускает обфускацию в модели со слабо ограниченным симулятором, но не допускает обфускации в модели виртуального «черного ящика».

Доказательства этой теоремы основывается на той же идее, что и доказательство теоремы 1.

Авторам статьи [95] удалось также показать существование таких семейств TM, для которых невозможно построить обфускации в модели со слабо ограниченным симулятором.

Возможности симулятора можно усилить и другим способом. Поскольку противник, имеющий доступ к обфускированной программе, может наблюдать не только пары вход-выход, но отслеживать трассы вычислений, целесообразно проверить, нельзя ли добиться того, чтобы в результате преобразования обфускации программы, ее вычислительные трассы составляли бы единственную полезную информацию, которую может извлечь противник проводя эксперименты с программой.

Для достижения этой цели модифицируем определение обфускации в модели виртуального «черного ящика». Обфускация нового вида, введенная и

исследованная в статье [14], была названа обфускацией в модели виртуального «серого ящика». Модель виртуального «серого ящика» отличается от модели виртуального «черного ящика» в двух аспектах.

Во-первых, оракул симулятора в ответ на запрос x выдает не только результат вычисления обфускируемой программы $\pi(x)$, но также и трассу вычисления программы π для входных данных x. Таким образом, здесь важно знать, какая именно из эквивалентных программ выбрана в качестве оракула. Мы требуем, чтобы такой программой была исходная программа π. Это означает, что обфускатор не обязан скрывать ни одно из тех свойств программы π, которые можно эффективно установить на основе трасс вычисления TM π.

Во-вторых, вместо завершающихся программ мы рассматриваем реагирующие программы. В отличие от завершающихся программ, предназначенных для вычисления отношений между входными и выходными данными, реагирующие программы осуществляют преобразования потоков событий-запросов, поступающих на вход программы, в поток откликов-реакций на эти события, вырабатываемых на выходе программы. Таким образом, реагирующая программа вычисляет функцию, отображающую бесконечную последовательностей входных данных $x_1, x_2, \ldots, x_n, \ldots$ (последовательность запросов) в бесконечную последовательность выходов $y_1, y_2, \ldots, y_n, \ldots$ (последовательность откликов), так что каждый выход y_n зависит только от входов $x_1, x_2, \ldots, x_n$. Примерами реагирующих программ могут служить сетевые протоколы (в том числе криптографические), встроенные системы, операционные системы и т.п. Реагирующую программу можно определить формально как обычную TM, которая использует входную и выходную ленты, а также некоторое количество вспомогательных лент так, что чтение очередного входного слова x_{n+1} не осуществляется до тех пор, пока не будет полностью завершена запись выходного слова y_n. Для обозначения таких машин будем использовать аббревиатуру RTM, чтобы отличить их от обычных TM.

Чтобы отличать оракул из определения обфускатора в модели виртуального «черного ящика» (Определение 1) от оракула, используемого в новом определении, условимся обозначать последний записью $Tr(\pi)$. В ответ на каждый очередной запрос x_n оракул $Tr(\pi)$ выдает пару $\langle y_n, tr(x_n) \rangle$, где y_n - результат работы машины π на входе x_n (отметим, что y_n зависит не только от x_n, но и на всех предыдущих входов, которые ранее

использовались в качестве запросов к оракулу) и $tr(x_n)$ - трасса выполнения π на этом входе.

Определение 2 (модель виртуального «серого ящика») [14]

Обфускатором в модели виртуального «серого ящика» для семейства RTM M называется такая PPT O, которая удовлетворяет условиям функциональной эквивалентности и полиномиальных издержек, а также следующему требованию стойкости:

Свойство виртуального «серого ящика». Для любой PPT A (противника) существует такая PPT S (симулятор), для которой соотношение

$$|\Pr[A(O(\pi)) = 1] - \Pr[S^{Tr(\pi)}(1^{|\pi|}) = 1]| \leq neg(|\pi|)$$

выполняется для любой RTM $\pi, \pi \in M$, причем первая вероятность вычисляется для случайных величин, используемых обфускатором O и противником A, а вторая вероятность вычисляется для случайных величин, используемых симулятором S.

Теорема 3. [14]. Если существуют односторонние функции, то существуют такие семейства RTM, обфускация которых в модели виртуального «серого ящика» невозможна.

Еще один вариант определения обфускации, близкий к модели виртуального «черного ящика», был предложен в статье [96]. На практике противник может проводить анализ обфускированной программы, располагая некоторыми дополнительными данными, которые могут иметь или не иметь отношение к этой программе. Например, противник может располагать наряду с обфускированной программой π также и демонстрационной упрощенной версией этой программы π'. Тогда модель виртуального «черного ящика» может быть соответствующим образом модифицирована.

Определение 3 (модель виртуального «черного ящика» с дополнительным входом) [96]

Обфускатором в модели виртуального «черного ящика» с дополнительным входом для семейства ТМ M называется PPT O, удовлетворяющая условиям функциональной эквивалентности и полиномиальных издержек, а также следующему условию:

Свойство виртуального «черного ящика» с дополнительным входом. Для любой PPT A (противника) существует такая PPT S (симулятор), для которой соотношение

$$|\Pr[A(O(\pi), z) = 1] - \Pr[S^{Tr(\pi)}(1^{|\pi|}, z) = 1]| \leq neg(|\pi|)$$

выполняется для любой ТМ $\pi, \pi \in M$, и двоичной строки (дополнительного входа) z, причем первая вероятность вычисляется для случайных величин, используемых обфускатором O и противником A, а вторая вероятность вычисляется для случайных величин, используемых симулятором S.

В этом определении допускается возможность того, что дополнительные входные данные z могут зависеть или не зависеть от обфускируемой программы. Поэтому, есть два варианта обфускации с дополнительным входом, и для обоих была установлена их взаимосвязь с введенными выше разновидностями обфускации.

Теорема 4. [95,96]. Если для семейства ТМ M существуют обфускатор в модели виртуального «черного ящика», то для этого семейства ТМ существует обфускатор с дополнительным входом, не зависящим от обфускируемой программы. Если для семейства ТМ M существуют обфускатор в модели виртуального «черного ящика» со слабо ограниченным симулятором, то для этого семейства ТМ существует обфускатор с дополнительным входом, зависящим от обфускируемой программы.

В статье [96] был также описан класс логических схем – схемы со сверхполиномиальной энтропией, вычисляющие функции с NP-полным фильтром, – для которых невозможно построить обфускаторов с зависимым или независимым дополнительным входом. Авторы упомянутой статьи, в частности, показали, что класс схем со сверхполиномиальной энтропией включает в себя всякое семейство логических схем, вычисляющих псевдослучайные функции, а также всякое семейство логических схем, реализующих криптосистемы, в которых используются псевдослучайные функции.

Полученные результаты показывают, что простые ослабления требования стойкости обфускации в модели виртуального «черного ящика» не влекут за собой существенного упрощения задачи обфускации программ. Кроме того, выяснилось, что для многих криптографических функций никакая программа их вычисления не может быть полностью обфускирована. Поэтому целесообразно рассмотреть другие виды обфускации программ, менее требовательные, нежели обфускации в модели виртуального «черного ящика».

3.3. Обфускация алгоритмов

Наиболее очевидное применение обфускации для защиты программных продуктов предлагает следующий сценарий. Предположим, что некто изобрел быстрый алгоритм решения сложной и практически важной задачи. Целью обфускации в этом случае, является такое преобразование программы, реализующий этот алгоритм, которое не позволяет извлечь его с легкостью из обфускированной программы. Обфускация в модели виртуального «черного

ящика» здесь непригодна: вряд ли найдется клиент, готовый купить «черный ящик», как часть программного обеспечения. Напротив, любой программный продукт на рынке должен иметь руководство пользователя, с указанием свойств функциональности. Если известна функция, вычисляемая программой, то цель обфускации в этом случае состоит в том, чтобы воспрепятствовать пониманию оригинальных особенностей алгоритма, на основе которого простроена данная программа. Такая разновидность обфускации программ более соответствует целям защиты алгоритмов, нежели тотальная обфускация в модели виртуального «черного ящика». Самый простой способ формализации понятия обфускации алгоритмов состоит в том, чтобы предоставить противника в качестве дополнительной информации некоторую (произвольную) программу π_0, вычисляющую ту же самую функцию, что и обфускируемая программа π. Таким образом, мы приходим к нескольким определениям обфускатора алгоритмов, которые были предложены в работах [13,14,16].

Определение 4 (обфускация алгоритмов) [14]

Обфускатором алгоритмов для семейства ТМ M называется такая РРТ O, которая удовлетворяет условиям функциональной эквивалентности и полиномиальных издержек, а также следующему требованию стойкости:

Свойство сокрытия алгоритма. Для любой РРТ A (противника) существует такая РРТ S (симулятор), для которой соотношение

$$|\Pr[A(O(\pi),\pi_0)=1]-\Pr[S(1^{|\pi|},\pi_0)=1]|\le neg(|\pi|)$$

выполняется для любой пары эквивалентных ТМ π, π_0 из класса M, удовлетворяющей условию $|\pi_0| = poly(|\pi|)$, причем первая вероятность вычисляется для случайных величин, используемых обфускатором O и противником A, а вторая вероятность вычисляется для случайных величин, используемых симулятором S.

Помимо указанного выше определения обфускатора алгоритмов в статьях [13,16] для сокрытия алгоритмов были предложены два сходных альтернативных определения обфускации, основанные на концепции вычислительно неотличимых распределений случайных величин – неотличимая обфускация и наилучшая обфускация программ.

Определение 5 (неотличимая обфускация) [13]

Неотличимым обфускатором для семейства ТМ M называется такая РРТ O, которая удовлетворяет условиям функциональной эквивалентности и полиномиальных издержек, а также следующему условию:

Свойство эффективной неотличимости программ. Для любой пары эквивалентных ТМ π_1, π_2 из класса M, имеющих одинаковый размер, распределения вероятностей случайных величин $O(\pi_1)$ и $O(\pi_2)$ вычислительно неотличимы, т.е. для любой РРТ D справедливо соотношение $|\Pr[D(O(\pi_1))=1]-\Pr[D(O(\pi_2))=1]|\le neg(|\pi|)$

Нетрудно видеть, что свойство неотличимости программ следует из свойства виртуального «черного ящика», однако тот прием, который применялся в теореме 1 для доказательства невозможности построения обфускатора в модели виртуального «черного ящика», уже непригоден в случае неотличимой обфускации. Недостаток определения 5 состоит в том, что оно не дает интуитивно понятных гарантий того, что обфускированная программа действительно маскирует реализуемый алгоритм. Поэтому в статье [16] было предложено альтернативное определение, в основу которого также положена идея о неотличимости обфускированных программ.

Определение 6 (наилучшая обфускация) [16]

Наилучшим обфускатором для семейства ТМ M называется такая РРТ O, которая удовлетворяет условиям функциональной эквивалентности и полиномиальных издержек, а также следующему условию:

Свойство ограниченной эффективной выведываемости. Для любой РРТ L (выведыватель) существует такая РРТ S (симулятор), что для любой пары эквивалентных ТМ π_1, π_2 из класса M, имеющих одинаковый размер, распределения вероятностей случайных величин $L(O(\pi_1))$ и $S(\pi_2)$ вычислительно неотличимы.

Свойство ограниченной выведываемости подразумевает, что противник может извлечь из текста обфускированной программы лишь такую полезную информацию, которую можно было бы эффективно вычислить, имея текст любой эквивалентной программы.

В определениях 5 и 6 условия вычислительной неотличимости распределений вероятностей можно усилить, потребовав, чтобы рассматриваемые распределения вероятностей совпадали (абсолютная неотличимость) или чтобы статистическое расстояние между ними не превосходило некоторой заранее заданной константы (статистическая неотличимость). И хотя взаимосвязь определений обфускации 1, 5 и 6 неочевидна, справедлива следующая

Теорема 5. [16,95]. Следующие три вида обфускации программ равносильны:

1. неотличимая обфускация,
2. наилучшая обфускация,

3. обфускация в модели виртуального «черного ящика» с симулятором, имеющим неограниченные вычислительные возможности.

Из этой теоремы видно, что обфускация всякого класса программ в модели виртуального «черного ящика» влечет за собой и наилучшую обфускацию того же класса программ. Поэтому требование эффективности симулятора S в определении 6 является избыточным. Кроме того, обфускация алгоритмов для любого класса программ влечет наилучшую обфускацию того же класса программ. Однако вопрос о том, верно ли обратное включение, остается открытым. Открыт и вопрос о существовании семейств программ, для которых не существует наилучшей обфускации. Однако в статье [16] была доказана следующая теорема

Теорема 6. [16]. Если семейство программ, представленных в виде 3-КНФ, имеет наилучшую (в смысле статистической неотличимости) обфускацию, то коллапс полиномиальной иерархии классов сложности происходит на втором уровне.

В настоящее время вопрос об устройстве полиномиальной иерархии остается открытым, хотя большинство специалистов в области теории вычислений придерживаются мнения о том, что она имеет бесконечно много уровней.

Важным частным случаем обфускации алгоритмов является обфускация констант. Предположим, что имеется параметризованное семейство ТМ $M(C) = \{\pi(c) : c \in C\}$, которые отличаются друг от друга только значением одного секретного параметра, используемого в качестве константы. Таким параметром может быть, например, секретный ключ в процедуре шифрования или электронно-цифровой подписи. Алгоритм, реализуемый программами семейства $M(C)$, не составляет секрета и не нуждается в маскировке.

В публикациях, посвященных проблеме обфускации программ, предложено несколько определений параметризованных программ. Одно из них является простой адаптацией определения обфускатора в модели виртуального «черного ящика».

Определение 7 (обфускация констант в модели виртуального «черного ящика») [14,21]

Обфускатором констант для семейства ТМ $\mathrm{M}(\mathrm{C})$ называется такая ППТ O, которая удовлетворяет условиям функциональной эквивалентности и полиномиальных издержек, а также следующему требованию стойкости:

Свойство сокрытия констант. Для любой ППТ A (противника) существует такая ППТ S (симулятор), для которой соотношение

$$|\Pr[A(O(\pi(c_0)), \pi(c)) = 1] - \Pr[S^{\pi(c_0)}(1^{|\pi(c_0)|}, \pi(c)) = 1]| \le neg(|\pi(c_0)|)$$

выполняется для любой пары ТМ $\pi(c), \pi(c_0)$ из класса $\mathrm{M}(\mathrm{C})$,

удовлетворяющей условию $|\pi(c_0)| = poly(|\pi(c)|)$, причем первая вероятность вычисляется для случайных величин, используемых обфускатором O и противником A, а вторая вероятность вычисляется для случайных величин, используемых симулятором S.

Пример программ, для которых невозможно построить обфускацию в модели виртуального «черного ящика», свидетельствует также и том, что для некоторых семейств параметризованных программ невозможно провести обфускацию констант.

Другое определение является переложением определения неотличимого обфускатор. Пусть на множестве констант C задано некоторое распределение вероятностей.

Определение 8 (неотличимая обфускация констант) [21]

Обфускатором констант для семейства ТМ $M(C)$ называется такая пара ППТ (G, O), которая удовлетворяет следующим трем условиям:

1. **Функциональная эквивалентность.** Для любой константы c, $c \in C$, выходом $G(c)$ является константа d, $d \in C$, а выходом $O(G(c), c)$ является ТМ π', такие, что ТМ $\pi(c)$ эквивалентна ТМ $\pi'(d)$.
2. **Полиномиальные издержки.** Для любой константы c, $c \in C$, размер любой константы $G(c)$ отличается от размера константы c не более чем полиномиально, и размер и время выполнения любой ТМ $O(\pi)$ отличается от размера ТМ π не более чем полиномиально.
3. **Свойство неотличимости констант.** Для любой ППТ D (распознавателя) существует такая ППТ S (симулятор), для которой соотношение

$$|\Pr[D^{\pi(c)}(1^{|c|}, G(c)) = 1] - \Pr[D^{\pi(c)}(1^{|c|}, S^{\pi(c)}(1^{|c|})) = 1]| \le neg(|c|)$$

где константа c выбирается согласно указанному выше распределению вероятностей, и обе вероятности, фигурирующие в левой части соотношения вычисляется для случайных величин, используемых обфускатором O, распознавателем D и симулятором S.

Здесь обфускации подвергается сама константа, а программа, использующая ее, лишь модифицируется, чтобы удовлетворять условию функциональной эквивалентности. Это определение обфускации констант специально введено для использования его в решении задач преобразования криптосистем с секретным ключом в криптосистемы с открытым ключом. Однако для определения 8 существуют программы, которые нельзя обфускировать.

Теорема 7. [21]. Семейство программ $M(C)$, реализующих псевдослучайную функцию невозможно обфускировать в соответствие с определением 8.

3.4. Обфускация предикатов

Спектр формальных определений обфускации программ замыкает обфускация предикатов. Она предназначена для сокрытия какого-либо выделенного свойства программ. Предположим, что на множестве ТМ M задан предикат P.

Определение 9 (обфускация предикатов) [14]

Обфускатором предиката P для семейства ТМ M называется такая PPT O, которая удовлетворяет условиям функциональной эквивалентности и полиномиальных издержек, а также следующему требованию стойкости:

Свойство сокрытия предиката. Для любой PPT A (противника) существует такая PPT S (симулятор), для которой соотношение

$$|\Pr[A(O(\pi)) = P(\pi)] - \Pr[S^{\pi}(1^{|\pi(c_0)|}) = P(\pi)]| \leq neg(|\pi|)$$

выполняется для любой ТМ π из класса M, причем первая вероятность вычисляется для случайных величин, используемых обфускатором O и противником A, а вторая вероятность вычисляется для случайных величин, используемых симулятором S.

Сопоставив определения 1 и 8, можно заметить, что обфускация программ в модели виртуального «черного ящика» должна скрывать все возможные предикаты на заданном семействе программ, в то время как определение 8 допускает построение различных обфускаторов для разных предикатов. Существование необфускируемых предикатов для некоторых семейств программ следует из теоремы 1.

4. Заключение

Как видно из приведенной в этом разделе онтологии определений обфускации программ, обфускаторы могут быть условно разделены на два класса – «сильные» обфускаторы (определения 1-3), призванные маскировать любые невыведываемые свойства программ, и «слабые» обфускаторы (определения 4-8), предназначенные для сокрытия лишь отдельных свойств программ. Для каждого определения обфускации доказано существование таких представительных классов функций (включающих функции, имеющие криптографические приложения), которые не могут быть реализованы обфускируемыми программами. Математическая техника, развитая в работах [13,14,16,20-22,96] позволяет, по-видимому, построить примеры такого рода

для любого «разумного» определения обфускации. Гораздо более трудно доказать, что некоторые практически значимые программы и функции могут быть обфускированы. Обзор наиболее интересных результатов в этом направлении исследований приведен в разделе 5. Но прежде чем обратиться к эти результатам, в какой мере обфускация программ может быть полезна для решения тех или иных задач криптографии и компьютерной безопасности.

Список литературы

[1]. Diffie W., Hellman M. New directions in cryptography // *IEEE Transactions on Information Theory*, IT-22(6), 1976, p.644-654.

[2]. Collberg C., Thomborson C., Low D. A Taxonomy of Obfuscating Transformations // Technical Report, N 148, Univ. of Auckland, 1997.

[3]. Cohen F. Operating system protection through program evolution // Computers and Security, v. 12, N 6, 1993, p. 565-584.

[4]. Chess D., White S. An undetectable computer virus // *Proceedings of the 2000 Virus Bulletin Conference*, 2000.

[5]. Szor P., Ferrie P. Hunting for metamorphic // *Proceedings of the 2001 Virus Bulletin Conference*, 2001, p.123-144.

[6]. Collberg C., J. Nagra. Surreptitious Software: Obfuscation, Watermarking, and Tamperproofing for Program Protection. Addison-Wesley Professional, 2009.

[7]. Aucsmith D. Tamper resistant software: an implementation // *Information Hiding Conference*, Lecture Notes in Computer Science, v. 1174, 1996, p. 317-333.

[8]. Scud T.T. ObjObf - x86/Linux ELF relocateable object obfuscator, 2003. http://packetstormsecurity.org/files/31524/objobf-0.5.0.tar. bz2.

[9]. Solutions P. DashO - the premier Java obfuscator and efficiency enhancing tool. http://www.preemptive.com/products/dasho/.

[10]. Solutions P. Dotfuscator - the premier .NET obfuscator and efficiency enhancing tool. http://www.preemptive.com/products/dotfuscator/.

[11]. Z. KlassMaster. The second generation Java obfuscator. http://www.zelix.com/.

[12]. Ge J., Chaudhuri S., Tyagi A. Control Flow Based Obfuscation // *Proceedings of the Digital Rights Management Workshop*. Alexandria, VA, USA, 2005, p. 83-92

[13]. Barak B., Goldreich O., Impagliazzo R., Rudich S., Sahai A., Vadhan S., Ke Yang. On the (im)possibility of obfuscating programs // *Advances in Cryptology - CRYPTO'01*, Lecture Notes in Computer Science, v. 2139, 2001, p. 1-18 (см. также Journal of the ACM 2012).

[14]. Varnovsky N.P. A note on the concept of obfuscation // Proceedings of Institute for System Programming, Moscow, N 6, 2004, p. 127-137.

[15]. Kuzurin N.N., Shokurov A.V., Varnovsky N.P., Zakharov V.A. On the concept of software obfuscation in computer security // *Information Security Conference*, Lecture Notes in Computer Science, v. 4779, 2007, p. 281-298.

[16]. Goldwasser S., Rothblum G.N. On best possible obfuscation // *Theory of Cryptography Conference*, Lecture Notes in Computer Science, v. 4392, 2007, p. 194-213.

[17]. Canetti R. Towards realizing random oracles: hash functions that hide all partial information // *Advances in Cryptology --- CRYPTO'97*, Lecture Notes in Computer Science, v. 1294, 1997, p. 455-469.

[18]. Varnovsky N.P., Zakharov V.A. On the possibility of provably secure obfuscating programs // *Conference ``Perspectives of System Informatics''*, Lecture Notes in Computer Science, v. 2890, 2004, p. 91-102.
[19]. Lynn B., Prabhakaran M., Sahai A. Positive results and techniques for obfuscation // *Advances in Cryptology - EUROCRYPT 2004*, Lecture Notes in Computer Science, v. 3027, 2004, p. 20-39.
[20]. Wee H. On obfuscating point functions // *Proceedings of the 37-th Symposium on Theory of Computing*, 2005, p. 523-532.
[21]. Hofheinz D., Malone-Lee J., Stam M. Obfuscation for cryptographic purpose // *Theory of Cryptography Conference*, Lecture Notes in Computer Science, v. 4392, p. 214-232.
[22]. Canetti R., Dakdouk R. R. Obfuscating point functions with multibit output // *Advances in Cryptology – EUROCRYPT 2008*, Lecture Notes in Computer Science, 2008, v. 4965, p. 489–508.
[23]. Hohenberger S., Rothblum G.N., Shelat A., Vaikuntanathan V. Securely obfuscating re-encryption // *Proceedings of the 4-th Conference on Theory of Cryptography*, 2007, p. 233-252
[24]. Canetti R., Rothblum G.N., Varia M. Obfuscation of hyperplane membership // *Proceedings of the 7-th Conference on Theory of Cryptography*, 2010, p. 72-89.
[25]. Collberg C., Thomborson C., Low D. Manufacturing cheap, resilient and stealthy opaque constructs // *Proceedings of the Symposium on Principles of Programming Languages*, 1998, p. 184-196.
[26]. de Oor A., van der Oord L. Stealthy obfuscation techniques: misleading pirates // Technical Report of Department of Computer Science University of Twente Enschede, The Netherlands, 2003.
[27]. Naumovich G, Memon N. Preventing piracy, reverse engineering,and tampering // *IEEE Computer*, 2003, v. 36, N 7, p. 64-71.
[28]. Collberg C, Thomborson C., Watermarking, Tamper-Proofing, and Obfuscation - Tools for Software Protection // IEEE Transactions on Software Engineering, v. 28, N 6, 2002.
[29]. Arboit G. A method for watermarking Java programs via opaque predicates//Proceedings of the International Conference on Electronic Commerce Research (ICECR-5). Montreal, Canada, 2002: 1-8.
[30]. Zhu W., Thomborson C., Wang F.-Y. A survey of software watermarking // Lecture Notes in Computer Science, v.3495, 2005, p. 454-458.
[31]. Myles G, Collberg C. Software watermarking via opaque predicates: Implementation, analysis, and attacks. Electronic Commerce Research, 2006, 6(2): 155-171.
[32]. Sander T., Tchudin C.F. Protecting mobile agents against malicious hosts // *Mobile Agents and Security*, Lecture Notes in Computer Science, 1997, p. 44-60.
[33]. Hohl F. Time limited blackbox security: protecting mobile agents from malicious hosts // *Mobile Agents and Security*, Lecture Notes in Computer Science, v. 1419, 1998, p. 92-113.
[34]. D'Anna L., Matt B., Reisse A., Van Vleck T. , Schwab S., LeBlanc P. Self-Protecting Mobile Agents Obfuscation Report // Technical Report N 03-015, Network Associates Laboratories, June 2003.
[35]. Wu J., Zhang Y., Wang X. et al. A scheme for protecting mobile agents based on combining obfuscated control flow and time checking technology // *Proceedings of the Conference on Computational Intelligence and Security*. Harbin, Heilongjiang, China, 2007, p. 912-916

[36]. Roeder T., Schneider F.B. Proactive Obfuscation // *ACM Transactions on Computer Systems*, v. 28, N 2, 2010.
[37]. Ostrovsky R., Skeith W.E. Private searching on streaming data //*Advances in Cryptology - CRYPTO-2005*, Lecture Notes in Computer Science, v. 3621, 2005, p. 223-240.
[38]. Narayanan A., Shmatikov V. Obfuscated databases and group privacy // *Proceedings of the 12th ACM Conference on Computer and Communications Security*, 2005, p. 102-111.
[39]. Иванников В.П., Варновский Н.П., Захаров В.А., Кузюрин Н.Н., Шокуров А.В., Кононов А.Н., Калинин А.В. Методы информационной защиты проектных решений при изготовлении микроэлектронных схем // Известия Таганрогского радиотехнического университета, 2005, т. 4, с. 112-119.
[40]. Варновский Н.П., Захаров В.А., Кузюрин Н.Н., Чернов А.В., Шокуров А.В. Задачи и методы обеспечения информационной безопасности при производстве микроэлектронных схем // Труды Института системного программирования РАН, 2006, т. 11. с. 29-61.
[41]. Borello J.M., Me L. Code obfuscation technique for metamorphic viruses // *Journal of Computer Virology*, 2008, v. 4, p, 211-220.
[42]. Bhatkar S., Du Varney D.C., Sekar R. Efficient techniques for comprehensive protection from memory error exploits // *USENIX Security*, 2005.
[43]. Wroblewski G. General method of program code obfuscation // Draft, 2002, 84 p.
[44]. Linn C., Debray S. Obfuscation of executable code to improve resistance to static disassembly // *Proceedings of the 10-th ACM Conference on Computer and Communication Security*, 2003, p. 290-299.
[45]. Sosonkin M, Naumovich G, Memon N. Obfuscation of design intent in object-oriented applications // *Proceedings of the Digital Rights Management Workshop*. Washington, DC, USA, 2003, p. 142-153.
[46]. Collberg C., Myles G., Huntwort A. Sandmark – a tool for software protection research // IEEE Security and Privacy, 2003, v. 1, N 4, p. 40-49.
[47]. Heffner K., Collberg C. The obfuscation executive // *Information Security Conference*, Lecture Notes in Computer Science, 2004, v. 3225.
[48]. Chan J. T., Yang W. Advanced obfuscation techniques for Java bytecode. Journal of Systems and Software, 2004, v. 71, N 1-2, p. 1-10.
[49]. Cimato S., De S. A., Petrillo U. F. Overcoming the obfuscation of Java programs by identifier renaming. Journal of Systems and Software, 2005, v. 78, N 1, p. 60-72.
[50]. Madou M., Anckaert B., de Sutter B., de Bosschere K. Hybrid static-dynamic attacks against software protection mechanisms // *Proceedings of the 5th ACM workshop on Digital rights management*, 2005, p. 75-82.
[51]. Udupa S. K., Debray S. K., Madou M. Deobfuscation: Reverse engineering obfuscated code // *Proceedings of the 12-th Working Conference on Reverse Engineering*. Pittsburgh, PA, USA, 2005, p. 45-54.
[52]. Ge J., Chaudhuri S., Tyagi A. Control Flow Based Obfuscation // *Proceedings of the Digital Rights Management Workshop*. Alexandria, VA, USA, 2005, p. 83-92.
[53]. Chen K., Chen J. B. On instrumenting obfuscated java bytecode with aspects // *Proceedings of the 2006 International Workshop on Software Engineering for Secure Systems*. Shanghai, China, 2006, p. 19-26.
[54]. Madou M., Anckaert B., de Sutter B., de Bosschere K., Cappaert J., Preenel B. On the effectiveness of source code transformations for binary obfuscation // *Proceedings of the International Conference on Software Engineering Research and Practice*, 2006, p.527-533.

[55]. Madou M., Anckaert B., Moseley P., Debray S., de Sutter B., de Bosschere K. Software protection through dynamic code mutation // *Proceedings of the 6-th international conference on Information Security Applications*, 2006, p. 194-206.
[56]. Drape S, Majumdar A, Thomborson C. Slicing aided design of obfuscating transforms // *Proceedings of the International Computing and Information Systems Conference (ICIS 2007)*. Melbourne, Australia, 2007, p. 1019-1024.
[57]. Majumdar A., Drape S., Thomborson C. Slicing obfuscations: Design, correctness, and evaluation // *Proceedings of the 2007 ACM Workshop on Digital Rights*. Alexandria, VA, USA, 2007, p. 70-81.
[58]. Batchelder M., Hendren L. Obfuscating Java: The most pain for the least gain // *Proceedings of the Compiler Construction*. Braga, Portugal, 2007, p. 96-110.
[59]. Ceccato M., Di. P. M., Nagra J. et al. Towards experimental evaluation of code obfuscation techniques // *Proceedings of the 4th ACM Workshop on Quality of Protection*. Alexandria, VA, USA, 2008, p. 39-46.
[60]. Darwish S.M., Guirguis S.K., Zalat M.S. Stealthy code obfuscation technique for software security // *Proceedings of the International Conference on Computer Engineering and Systems*, 2010, p. 93-99.
[61]. А.В. Чернов. Об одном методе маскировки программ // *Труды Института системного программирования РАН*, 2003, т. 4. с. 85-119.
[62]. Majumdar A., Drape S., Thomborson C. et al. Metrics-based evaluation of slicing obfuscations // *Proceedings of the 3rd International Symposium on Information Assurance and Security*. Manchester, United Kingdom, 2007, p. 472-477.
[63]. Naeem N. A., Batchelder M., Hendren L. Metrics for Measuring the Effectiveness of Decompilers and Obfuscators // *Proceedings of the 15th IEEE International Conference on Program*. Banff, Alberta, Canada, 2007, p. 253-258.
[64]. Anckaert B., Madou M., De S. B. et al. Program obfuscation: A quantitative approach // *Proceedings of the 2007 ACM Workshop on Quality of Protection*. Alexandria, VA, USA, 2007, p. 15-20.
[65]. Tsai H. Y., Huang Y. L., Wagner D. A graph approach to quantitative analysis of control-flow obfuscating // *IEEE Transactions on Information Forensics and Security*, 2009, v. 4, N 2, p. 257-267.
[66]. Cousot P., Cousot R. An abstract interpretation-based framework for software watermarking // Proceedings of 31st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, 2004, p. 173-185.
[67]. Zakharov V.A. Ivanov K.S. Program obfuscation as obstruction of program static analysis // *Труды Института системного программирования РАН*, 2004, т. 6. с. 141-161.
[68]. Захаров В.А., Иванов К.С. О противодействии некоторым алгоритмам статического анализа программ // *Труды конференции 'Математика и безопасность информационных технологий' (МаБИТ-03)*, 2003, с. 282-286.
[69]. Dalla Preda M., Giacobazzi R. Semantic-based code obfuscation by abstract interpretation // *International Colloquium on Automata, Language and Programming*, Lecture Notes in Computer Science, v. 3580, 2005, p.1325-1336.
[70]. Захаров В.А., Иванов К.С. О моделях программ в связи с задачей противодействия алгоритмам статического анализа программ // *Труды Института системного программирования РАН*, 2006, т. 11.
[71]. Варновский Н.П., Захаров В.А., Кузюрин Н.Н., Подловченко Р.И., Шокуров А.В., Щербина В.Л. О применении методов деобфускации программ для обнаружения

сложных компьютерных вирусов // *Известия Таганрогского радиотехнического университета*, 2006, т. 6, с. 18-27.
[72]. Kuzurin N.N., Podlovchenko R.I., Scherbina V.L., Zakharov V.A. Using algebraic models of programs for detecting metamorphic malwares // *Труды Института системного программирования РАН*, 2007, т. 12, с. 77-94.
[73]. Della Preda M., Giacobazzi G. Semantic-based code obfuscation by abstract interpretation // Journal of Computer Security, 2009, v. 17, N 6, p. 855-908.
[74]. Christodorescu M., Jha S. Static analysis of executables to detect malicious patterns // *Proceedings of the 12-th Security Symposium*, 2003, p. 169-186.
[75]. Della Preda M., Christodorescu M., Jha S., Debray S. A semantic-based approach to malware detection // Proceedings of the *34th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, 2007, p. 377-388.
[76]. Della Preda M., Giacobazzi G., Debray S., Coogan K., Townsend G. Modelling Metamorphism by Abstract Interpretation // *Proceedings of the 17th International Static Analysis Symposium (SAS'10)*. Lecture Notes in Computer Science, 2010, v. 6337, p. 218-235.
[77]. Majumdar A, Thomborson C. On the use of opaque predicates in mobile agent code obfuscation // *Proceedings of the ISI 2005*. Altanta, GA, USA, 2005, p. 648-649.
[78]. Majumdar A., Thomborson C. Manufacturing opaque predicates in distributed systems for code obfuscation // *Proceedings of the 4th International Conference on Information Security*. Hobart, Tasmania, Australia, 2006, p. 187-196.
[79]. Della Preda M., Giacobazzi G., Madou M., de Bosschere K. Opaque predicate detection by abstract interpretation // *11th International Conference on Algebriac Methodology and Software Technology*. Lecture Notes in Computer Science, v 4019, 2006, p. 81-95.
[80]. Wang C., Davidson J., Hill J., Knight J. Protection of software-based survivability mechanisms // Proceedings of the International Conference of Dependable Systems and Networks, 2001.
[81]. Chow S., Gu Y., Johnson H., Zakharov V. An approach to obfuscation of control-flow of sequential programs // *Information Security Conference*, Lecture Notes in Computer Science, v. 2000, 2001, p. 144-155
[82]. Ogiso T., Sakabe Y., Soshi M. Miyaji A. Software obfuscation on a theoretical basis and its implementation // IEEE Transactions Fundamentals, E86-A(1), 2003.
[83]. Варновский Н.П., Захаров В.А., Кузюрин Н.Н., Шокуров А.В. О перспективах решения задачи обфускации компьютерных программ // Труды конференции 'Математика и безопасность информационных технологий' (МаБИТ-03), 2003, с. 344-351.
[84]. Ostrovsky R. Efficient computation on oblivious RAMs // *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, 1990, p. 514-523.
[85]. Zhuang X., Zhang T., Lee H.-H. S., Pande S. Hardware assisted control flow obfuscation for embedded processes // Proceedings of the 2004 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems, 2004, p. 292-302.
[86]. Bhatkar S., du Varney D.C., Sekar R. Address obfuscation: an efficient approach to combat a broad range of memory error exploits // Proceedings of the 12th conference on USENIX Security Symposium, 2003, v. 8.
[87]. Garg S., Gentry C., Halevi S., Raykova M., Sahai A., Waters B. Candidate Indistinguishability Obfuscation and Functional Encryption for all circuits // IACR Cryptology ePrint Archive 2013, 451 (2013).
[88]. Hada S. Secure obfuscation for encrypted signatures // *Advances in Cryptology - EUROCRYPT 2010*, Lecture Notes in Computer Science, v. 6110, 2010, p. 92-112.

[89]. Adida B. , Wikström D. How to shuffle in piblic // *Proceedings of the 4th Conference on Theory of Cryptography*, Lecture Notes in Computer Science, 2007, v. 4392, p. 555-574.
[90]. Canetti R., Dwork C., Naor M., Ostrovsky R. Deniable encryption // *Advances in Cryptology- CRYPTO 97*, Lecture Notes in Computer Science, v. 1294, 1997, p. 90-104.
[91]. Sahai A., Waters B. How to Use Indistinguishability Obfuscation: Deniable Encryption, and More // CRYPTO ePrint 2013.
[92]. Hada S. Zero-knowledge and code obfuscation // *Advances in Cryptology- ASIACRYPT 2000*, Lecture Notes in Computer Science, v. 1976, 2000, p. 443-457.
[93]. Savage J. Models of Computation: Exploring the Power of Computing. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1997, 672 p.
[94]. Valiant L. A theory of learnable // Communications of the ACM, 1984, v. 27, N 11, p. 1134-1142.
[95]. Bitansky N., Canetti R. On obfuscation with strong simulators // *Advances in Cryptology- CRYPTO 2010*, Lecture Notes in Computer Science, v. 6223, 2010, p. 520-537.
[96]. Goldwasser S., Kalai T.Y. On the impossibility of obfuscation with auxiliary input // *Proceedings of the 46-th IEEE Symposium on Foundations of Computer Science*, 2005, p. 553-562.

The current state of art in program obfuscations:definitions of obfuscation security

[1] N.P. Varnovsky
[2] V.A.Zakharov <zakh@cs.msu.su>
[3] N.N. Kuzurin <nnkuz@ispras.ru>
[3] V.A. Shokurov <shok@ispras.ru>
[1] Information Security Institute, Office 10, 1, Michurinskiy prospect, Moscow, 119192, Russian Federation
[2] Lomonosov Moscow State University, Faculty CMC, 2nd Education Building, GSP-1, Leninskie Gory, Moscow
[3] ISP RAS, 25 Alexander Solzhenitsyn Str., Moscow, 109004, Russian Federation

Annotation. Program obfuscation is a semantic-preserving transformation aimed at bringing a program into such a form, which impedes the understanding of its algorithm and data structures or prevents extracting of some valuable information from the text of a program. Since obfuscation could find wide use in computer security, information hiding and cryptography, security requirements to program obfuscators became a major focus of interests for pioneers of theory of software obfuscation. In this paper we give a survey of various definitions of obfuscation security and main results that establish possibility or impossibility of secure program obfuscation under certain cryptographic assumptions. We begin with a short retrospective survey on the origin and development of program obfuscation concept in software engineering and mathematical cryptography. In the introduction we also point out on the main difficulties in the development of practical and secure obfuscation techniques. In the next section we discuss the main line of research in the application of program obfuscation to the solution of various problems in system programming and software security. Finally, in section 3 we present and discuss a compendium of formal definitions of the program obfuscation concept developed so far in mathematical cryptography - black-box obfuscation, gray-box obfuscation, the best possible obfuscation, obfuscation with auxiliary inputs, etc.. We also make a comparative analysis of these definitions and present the main results on the (im)possibility of secure program obfuscation w.r.t. every such definition.

Keywords: program, obfuscation, security, complexity, black box model, Turing machine

References

[1]. Diffie W., Hellman M. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6), 1976, p.644-654.

[2]. Collberg C., Thomborson C., Low D. A Taxonomy of Obfuscating Transformations. Technical Report, N 148, Univ. of Auckland, 1997.
[3]. Cohen F. Operating system protection through program evolution. *Computers and Security*, v. 12, N 6, 1993, p. 565-584.
[4]. Chess D., White S. An undetectable computer virus. *Proceedings of the 2000 Virus Bulletin Conference*, 2000.
[5]. Szor P., Ferrie P. Hunting for metamorphic. *Proceedings of the 2001 Virus Bulletin Conference*, 2001, p.123-144.
[6]. Collberg C., J. Nagra. Surreptitious Software: Obfuscation, Watermarking, and Tamperproofing for Program Protection. Addison-Wesley Professional, 2009.
[7]. Aucsmith D. Tamper resistant software: an implementation. *Information Hiding Conference*, Lecture Notes in Computer Science, v. 1174, 1996, p. 317-333.
[8]. Scud T.T. ObjObf - x86/Linux ELF relocateable object obfuscator, 2003. http://packetstormsecurity.org/files/31524/objobf-0.5.0.tar. bz2.
[9]. Solutions P. DashO - the premier Java obfuscator and efficiency enhancing tool. http://www.preemptive.com/products/dasho/.
[10]. Solutions P. Dotfuscator - the premier .NET obfuscator and efficiency enhancing tool. http://www.preemptive.com/products/dotfuscator/.
[11]. Z. KlassMaster. The second generation Java obfuscator. http://www.zelix.com/.
[12]. Ge J., Chaudhuri S., Tyagi A. Control Flow Based Obfuscation. *Proceedings of the Digital Rights Management Workshop*. Alexandria, VA, USA, 2005, p. 83-92
[13]. Barak B., Goldreich O., Impagliazzo R., Rudich S., Sahai A., Vadhan S., Ke Yang. On the (im)possibility of obfuscating programs. *Advances in Cryptology - CRYPTO'01*, Lecture Notes in Computer Science, v. 2139, 2001, p. 1-18 (see also *Journal of the ACM* 2012).
[14]. Varnovsky N.P. A note on the concept of obfuscation. *Trudy ISP RAN* [The Proceedings of ISP RAS], vol. 6, 2004, p. 127-137.
[15]. Kuzurin N.N., Shokurov A.V., Varnovsky N.P., Zakharov V.A. On the concept of software obfuscation in computer security. *Information Security Conference*, Lecture Notes in Computer Science, v. 4779, 2007, p. 281-298.
[16]. Goldwasser S., Rothblum G.N. On best possible obfuscation. *Theory of Cryptography Conference*, Lecture Notes in Computer Science, v. 4392, 2007, p. 194-213.
[17]. Canetti R. Towards realizing random oracles: hash functions that hide all partial information. *Advances in Cryptology - CRYPTO'97*, Lecture Notes in Computer Science, v. 1294, 1997, p. 455-469.
[18]. Varnovsky N.P., Zakharov V.A. On the possibility of provably secure obfuscating programs. *Conference ``Perspectives of System Informatics''*, Lecture Notes in Computer Science, v. 2890, 2004, p. 91-102.
[19]. Lynn B., Prabhakaran M., Sahai A. Positive results and techniques for obfuscation. *Advances in Cryptology - EUROCRYPT 2004*, Lecture Notes in Computer Science, v. 3027, 2004, p. 20-39.
[20]. Wee H. On obfuscating point functions. *Proceedings of the 37-th Symposium on Theory of Computing*, 2005, p. 523-532.
[21]. Hofheinz D., Malone-Lee J., Stam M. Obfuscation for cryptographic purpose. *Theory of Cryptography Conference*, Lecture Notes in Computer Science, v. 4392, p. 214-232.
[22]. Canetti R., Dakdouk R. R. Obfuscating point functions with multibit output. *Advances in Cryptology – EUROCRYPT 2008*, Lecture Notes in Computer Science, 2008, v. 4965, p. 489–508.

[23]. Hohenberger S., Rothblum G.N., Shelat A., Vaikuntanathan V. Securely obfuscating re-encryption. *Proceedings of the 4-th Conference on Theory of Cryptography*, 2007, p. 233-252
[24]. Canetti R., Rothblum G.N., Varia M. Obfuscation of hyperplane membership. *Proceedings of the 7-th Conference on Theory of Cryptography*, 2010, p. 72-89.
[25]. Collberg C., Thomborson C., Low D. Manufacturing cheap, resilient and stealthy opaque constructs. *Proceedings of the Symposium on Principles of Programming Languages*, 1998, p. 184-196.
[26]. de Oor A., van der Oord L. Stealthy obfuscation techniques: misleading pirates. Technical Report of Department of Computer Science University of Twente Enschede, Netherlands, 2003.
[27]. Naumovich G, Memon N. Preventing piracy, reverse engineering, and tampering. *IEEE Computer*, 2003, v. 36, N 7, p. 64-71.
[28]. Collberg C, Thomborson C., Watermarking, Tamper-Proofing, and Obfuscation - Tools for Software Protection. *IEEE Transactions on Software Engineering*, v. 28, N 6, 2002.
[29]. Arboit G. A method for watermarking Java programs via opaque predicates. *Proceedings of the International Conference on Electronic Commerce Research*. Montreal, Canada, 2002: 1-8.
[30]. Zhu W., Thomborson C., Wang F.-Y. A survey of software watermarking. Lecture Notes in Computer Science, v.3495, 2005, p. 454-458.
[31]. Myles G, Collberg C. Software watermarking via opaque predicates: Implementation, analysis, and attacks. *Electronic Commerce Research*, 2006, v. 6, N 2, p. 155-171.
[32]. Sander T., Tchudin C.F. Protecting mobile agents against malicious hosts. *Mobile Agents and Security*, Lecture Notes in Computer Science, 1997, p. 44-60.
[33]. Hohl F. Time limited blackbox security: protecting mobile agents from malicious hosts. *Mobile Agents and Security*, Lecture Notes in Computer Science, v. 1419, 1998, p. 92-113.
[34]. D'Anna L., Matt B., Reisse A., Van Vleck T. , Schwab S., LeBlanc P. Self-Protecting Mobile Agents Obfuscation Report. Tech. Rep. N 03-015, Network Associates Laboratories, June 2003.
[35]. Wu J., Zhang Y., Wang X. et al. A scheme for protecting mobile agents based on combining obfuscated control flow and time checking technology. *Proceedings of the Conference on Computational Intelligence and Security*. Harbin, Heilongjiang, China, 2007, p. 912-916
[36]. Roeder T., Schneider F.B. Proactive Obfuscation. *ACM Transactions on Computer Systems*, v. 28, N 2, 2010.
[37]. Ostrovsky R., Skeith W.E. Private searching on streaming data. *Advances in Cryptology - CRYPTO-2005*, Lecture Notes in Computer Science, v. 3621, 2005, p. 223-240.
[38]. Narayanan A., Shmatikov V. Obfuscated databases and group privacy. *Proceedings of the 12th ACM Conference on Computer and Communications Security*, 2005, p. 102-111.
[39]. Ivannikiov V.P., Varnovsky N.P., Zakharov V.A., Kuzurin N.P., Shokurov A.V., Kononov A.N., Kalinin A.V. Metody informazionnoy zaschity proektnyh resheniy pri izgotovlenii microelectronnyh shem [Information security techniques in the development of microelectronic circuits] *Izvestiya Taganrogskogo radiotehnicheskogo universiteta* [Bulletin of Taganrog Radiotechnical University], 2005, v. 4, p. 112-119.
[40]. Varnovsky N.P., Zakharov V.A., Kuzurin N.P., Chernov A.V., Shokurov A.V. Zadaschi I metody obespecheniya informazionnoy bezopasnosti pri proizvodstve microelectronnyh shem [Information security problems and techniques in the

development of microelectronic circuits], *Trudy ISP RAN* [The Proceedings of ISP RAS], 2006, т. 1, с. 29-61.
[41]. Borello J.M., Me L. Code obfuscation technique for metamorphic viruses. *Journal of Computer Virology*, 2008, v. 4, p, 211-220.
[42]. Bhatkar S., Du Varney D.C., Sekar R. Efficient techniques for comprehensive protection from memory error exploits. *USENIX Security*, 2005.
[43]. Wroblewski G. General method of program code obfuscation. Draft, 2002, 84 p.
[44]. Linn C., Debray S. Obfuscation of executable code to improve resistance to static disassembly. *Proceedings of the 10-th ACM Conference on Computer and Communication Security*, 2003, p. 290-299.
[45]. Sosonkin M, Naumovich G, Memon N. Obfuscation of design intent in object-oriented applications. *Proceedings of the Digital Rights Management Workshop*. Washington, DC, USA, 2003, p. 142-153.
[46]. Collberg C., Myles G., Huntwort A. Sandmark – a tool for software protection research. *IEEE Security and Privacy*, 2003, v. 1, N 4, p. 40-49.
[47]. Heffner K., Collberg C. The obfuscation executive. *Information Security Conference*, Lecture Notes in Computer Science, 2004, v. 3225.
[48]. Chan J. T., Yang W. Advanced obfuscation techniques for Java bytecode. *Journal of Systems and Software*, 2004, v. 71, N 1-2, p. 1-10.
[49]. Cimato S., De S. A., Petrillo U. F. Overcoming the obfuscation of Java programs by identifier renaming. *Journal of Systems and Software*, 2005, v. 78, N 1, p. 60-72.
[50]. Madou M., Anckaert B., de Sutter B., de Bosschere K. Hybrid static-dynamic attacks against software protection mechanisms. *Proceedings of the 5th ACM workshop on Digital rights management*, 2005, p. 75-82.
[51]. Udupa S. K., Debray S. K., Madou M. Deobfuscation: Reverse engineering obfuscated code. *Proceedings of the 12-th Working Conference on Reverse Engineering*. Pittsburgh, PA, USA, 2005, p. 45-54.
[52]. Ge J., Chaudhuri S., Tyagi A. Control Flow Based Obfuscation. *Proceedings of the Digital Rights Management Workshop*. Alexandria, VA, USA, 2005, p. 83-92.
[53]. Chen K., Chen J. B. On instrumenting obfuscated java bytecode with aspects. *Proceedings of the 2006 International Workshop on Software Engineering for Secure Systems*. Shanghai, China, 2006, p. 19-26.
[54]. Madou M., Anckaert B., de Sutter B., de Bosschere K., Cappaert J., Preenel B. On the effectiveness of source code transformations for binary obfuscation. *Proceedings of the International Conference on Software Engineering Research and Practice*, 2006, p.527-533.
[55]. Madou M., Anckaert B., Moseley P., Debray S., de Sutter B., de Bosschere K. Software protection through dynamic code mutation. *Proceedings of the 6-th international conference on Information Security Applications*, 2006, p. 194-206.
[56]. Drape S, Majumdar A, Thomborson C. Slicing aided design of obfuscating transforms. *Proceedings of the International Computing and Information Systems Conference (ICIS 2007)*. Melbourne, Australia, 2007, p. 1019-1024.
[57]. Majumdar A., Drape S., Thomborson C. Slicing obfuscations: Design, correctness, and evaluation. *Proceedings of the 2007 ACM Workshop on Digital Rights*. Alexandria, VA, USA, 2007, p. 70-81.
[58]. Batchelder M., Hendren L. Obfuscating Java: The most pain for the least gain. *Proceedings of the Compiler Construction*. Braga, Portugal, 2007, p. 96-110.

[59]. Ceccato M., Di. P. M., Nagra J. et al. Towards experimental evaluation of code obfuscation techniques. *Proceedings of the 4th ACM Workshop on Quality of Protection*., 2008, p. 39-46.
[60]. Darwish S.M., Guirguis S.K., Zalat M.S. Stealthy code obfuscation technique for software security. *Proceedings of the International Conference on Computer Engineering and Systems*, 2010, p. 93-99.
[61]. Chernov A.V. Ob odnom metode maskirovki program [On one program obfuscation techniques]. *Trudy ISP RAN* [The Proceedings of ISP RAS], 2003, v. 4, p. 85-119.
[62]. Majumdar A., Drape S., Thomborson C. et al. Metrics-based evaluation of slicing obfuscations. *Proceedings of the 3rd International Symposium on Information Assurance and Security*. Manchester, United Kingdom, 2007, p. 472-477.
[63]. Naeem N. A., Batchelder M., Hendren L. Metrics for Measuring the Effectiveness of Decompilers and Obfuscators. *Proceedings of the 15th IEEE International Conference on Program*. Banff, Alberta, Canada, 2007, p. 253-258.
[64]. Anckaert B., Madou M., De S. B. et al. Program obfuscation: A quantitative approach. *Proceedings of the 2007 ACM Workshop on Quality of Protection*. 2007, p. 15-20.
[65]. Tsai H. Y., Huang Y. L., Wagner D. A graph approach to quantitative analysis of control-flow obfuscating. *IEEE Trans. on Information Forensics and Security*, 2009, v. 4, N 2, p. 257-267.
[66]. Cousot P., Cousot R. An abstract interpretation-based framework for software watermarking. *Proceedings of 31st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, 2004, p. 173-185.
[67]. Zakharov V.A. Ivanov K.S. Program obfuscation as obstruction of program static analysis. *Trudy ISP RAN* [The Proceedings of ISP RAS], 2004, v. 6. p. 141-161.
[68]. Zakharov V.A. Ivanov K.S. O protivodeystvii nekotorym algorytmam staticheskogo analiza program [On the hindering some program static analysis algorithms]. *Trudy konferencii "Matematika i bezopasnost' informazionnyh tehnologiy" (MaBIT-03)* [Proceedings of the Conference "Mathematics and security of information technologies"], 2003, с. 282-286.
[69]. Dalla Preda M., Giacobazzi R. Semantic-based code obfuscation by abstract interpretation. *International Colloquium on Automata, Language and Programming*, Lecture Notes in Computer Science, v. 3580, 2005, p.1325-1336.
[70]. Zakharov V.A. Ivanov K.S. O modelyah program v svyazi s zadachey protivideystviya algoritmam ststicheskogo analiza [On the program models related with the proble of hindering program static analysis algorithms]. *Trudy ISP RAN* [The Proceedings of ISP RAS], 2006, т. 11.
[71]. Varnovsky N.P., Zakharov V.A., Kuzurin N.P., Podlovchenko R.I., Shokurov A.V., Shcherbina V.L. O primenenii metodov deobfuscazii program dlya obnaruzheniya slojnyh komputernyh virusov [On the application of program deobfuscation techniques for detecting non-trivial computer viruse]. *Izvestiya Taganrogskogo radiotehnicheskogo universiteta* [Bulletin of Taganrog Radiotechnical University], 2006, т. 6, с. 18-27.
[72]. Kuzurin N.N., Podlovchenko R.I., Scherbina V.L., Zakharov V.A. Using algebraic models of programs for detecting metamorphic malwares. *Trudy ISP RAN* [The Proceedings of ISP RAS], 2007, v. 12, p. 77-94.
[73]. Della Preda M., Giacobazzi G. Semantic-based code obfuscation by abstract interpretation. *Journal of Computer Security*, 2009, v. 17, N 6, p. 855-908.
[74]. Christodorescu M., Jha S. Static analysis of executables to detect malicious patterns. *Proceedings of the 12-th Security Symposium*, 2003, p. 169-186.

[75]. Della Preda M., Christodorescu M., Jha S., Debray S. A semantic-based approach to malware detection. *Proceedings of the 34th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, 2007, p. 377-388.
[76]. Della Preda M., Giacobazzi G., Debray S., Coogan K., Townsend G. Modelling Metamorphism by Abstract Interpretation. *Proceedings of the 17th International Static Analysis Symposium (SAS'10)*. Lecture Notes in Computer Science, 2010, v. 6337, p. 218-235.
[77]. Majumdar A, Thomborson C. On the use of opaque predicates in mobile agent code obfuscation. *Proceedings of the ISI 2005*. Altanta, GA, USA, 2005, p. 648-649.
[78]. Majumdar A., Thomborson C. Manufacturing opaque predicates in distributed systems for code obfuscation. *Proceedings of the 4th International Conference on Information Security*. Hobart, Tasmania, Australia, 2006, p. 187-196.
[79]. Della Preda M., Giacobazzi G., Madou M., de Bosschere K. Opaque predicate detection by abstract interpretation. *11th International Conference on Algebriac Methodology and Software Technology.* Lecture Notes in Computer Science, v 4019, 2006, p. 81-95.
[80]. Wang C., Davidson J., Hill J., Knight J. Protection of software-based survivability mechanisms. *Proceedings of the International Conference of Dependable Systems and Networks*, 2001.
[81]. Chow S., Gu Y., Johnson H., Zakharov V. An approach to obfuscation of control-flow of sequential programs. *Information Security Conference*, Lecture Notes in Computer Science, v. 2000, 2001, p. 144-155
[82]. Ogiso T., Sakabe Y., Soshi M. Miyaji A. Software obfuscation on a theoretical basis and its implementation. I*EEE Transactions Fundamentals*, E86-A(1), 2003.
[83]. Varnovsky N.P., Zakharov V.A., Kuzurin N.P., Podlovchenko R.I., Shokurov A.V. O perspektivah resheniya zadach obfuscacii komputernyh program [On the prospects of the solution of the obfuscation problems for computer programs] *Trudy konferencii "Matematika i bezopasnost' informazionnyh tehnologiy" (MaBIT-03)* [Proceedings of the Conference "Mathematics and security of information technologies"], 2003, c. 344-351.
[84]. Ostrovsky R. Efficient computation on oblivious RAMs. *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, 1990, p. 514-523.
[85]. Zhuang X., Zhang T., Lee H.-H. S., Pande S. Hardware assisted control flow obfuscation for embedded processes. *Proceedings of the 2004 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems*, 2004, p. 292-302.
[86]. Bhatkar S., du Varney D.C., Sekar R. Address obfuscation: an efficient approach to combat a broad range of memory error exploits. *Proceedings of the 12th conference on USENIX Security Symposium*, 2003, v. 8.
[87]. Garg S., Gentry C., Halevi S., Raykova M., Sahai A., Waters B. Candidate Indistinguishability Obfuscation and Functional Encryption for all circuits. *IACR Cryptology ePrint Archive 2013*, 451 (2013).
[88]. Hada S. Secure obfuscation for encrypted signatures. *Advances in Cryptology - EUROCRYPT 2010*, Lecture Notes in Computer Science, v. 6110, 2010, p. 92-112.
[89]. Adida B. , Wikström D. How to shuffle in piblic. *Proceedings of the 4th Conference on Theory of Cryptography*, Lecture Notes in Computer Science, 2007, v. 4392, p. 555-574.
[90]. Canetti R., Dwork C., Naor M., Ostrovsky R. Deniable encryption. *Advances in Cryptology- CRYPTO 97*, Lecture Notes in Computer Science, v. 1294, 1997, p. 90-104.
[91]. Sahai A., Waters B. How to Use Indistinguishability Obfuscation: Deniable Encryption, and More. *CRYPTO ePrint* 2013.

[92]. Hada S. Zero-knowledge and code obfuscation. *Advances in Cryptology- ASIACRYPT 2000*, Lecture Notes in Computer Science, v. 1976, 2000, p. 443-457.
[93]. Savage J. Models of Computation: Exploring the Power of Computing. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1997, 672 p.
[94]. Valiant L. A theory of learnable. *Communications of the ACM*, 1984, v. 27, N 11, p. 1134-1142.
[95]. Bitansky N., Canetti R. On obfuscation with strong simulators. *Advances in Cryptology- CRYPTO 2010*, Lecture Notes in Computer Science, v. 6223, 2010, p. 520-537.
[96]. Goldwasser S., Kalai T.Y. On the impossibility of obfuscation with auxiliary input. *Proceedings of the 46-th IEEE Symposium on Foundations of Computer Science*, 2005, p. 553-562.