УДК 519.7
ББК 22.17

ПРОТИВОДЕЙСТВИЕ СГОВОРУ В ДИСКРЕТНЫХ ДИНАМИЧЕСКИХ МОДЕЛЯХ КОМПЬЮТЕРНЫХ СЕТЕЙ[1]

Горбатенко Д. Е.[2], **Семенов А. А.**[3]
(*Институт динамики систем и теории управления им. В.М. Матросова, Иркутск*)

В статье представлен новый подход к противодействию ситуациям сговора в компьютерных сетях. Приведен алгоритм выделения т.н. «графов сговора» в рамках известной модели «Take-Grant» (TG). На основании графа сговора строится дискретная динамическая система (ДДС). Получение доступа к данным целевой вершины рассматривается как результат процесса в рамках введенной ДДС. Этот процесс можно блокировать за счет деактивации некоторых вершин в графе сговора. Ставится задача построения деактивирующего множества наименьшей мощности, которая сводится к задаче о булевой выполнимости (SAT) и решается с использованием современных SAT решателей.

Ключевые слова: дискретные динамические системы, модель Take-Grant, граф сговора, SAT.

1. Введение

В связи с интенсивным развитием информационных сетей вопросы обеспечения различных аспектов безопасности в таких сетях приобретают особую актуальность. В данном контексте весьма ценными представляются любые формальные методы

[1] *Работа выполнена при финансовой поддержке Российского научного фонда, проект № 16-11-10046.*

[2] *Горбатенко Дмитрий Евгеньевич, аспирант gorbadima@yandex.ru.*

[3] *Семенов Александр Анатольевич, кандидат технических наук, доцент biclop.rambler@yandex.ru.*

описания, моделирования и анализа действий злоумышленников, атакующих рассматриваемую сеть. При работе с соответствующими моделями могут возникать задачи комбинаторной природы, для эффективного решения которых «вычислительного потенциала» самой рассматриваемой модели может оказаться недостаточно. Более конкретно, мы подразумеваем ситуации, когда наблюдение за поведением сети при помощи, скажем, компьютерной симуляции, не позволяет получить в достаточной степени полную картину о некоторых принципиально возможных в рамках данной сети процессах.

Простой пример такого типа – возникновение несанкционированных прав доступа вследствие неумышленных или умышленных действий некоторых участников сети. К неумышленным действиям можно отнести программно-аппаратные сбои, тогда как к умышленным – сговор между недобросовестными пользователями. Мэтт Бишоп в [12] построил эффективные алгоритмы выявления ситуаций сговора в рамках известной модели распространения прав доступа Take-Grant. Однако нам ничего не известно о работах, в которых рассматривались бы задачи блокирования сговора. При детализации постановок такого рода задач становится ясна их комбинаторная природа. И если для выявления множеств объектов, которые в принципе могут вступить в сговор, может быть использовано имитационное моделирование (о чем М. Бишоп, кстати, не упоминает), то для блокирования ситуаций сговора обычного «проигрывания модели во времени», вообще говоря, недостаточно. Как мы увидим далее, ключевое свойство объектов в сети, способствующее возникновению сговора, это свойство быть активными, то есть выступать в роли проводников информации. Соответственно, противодействовать сговору можно деактиваций некоторых объектов. Деактивация без ограничений, однако, может привести к потере важных функций сети. В контексте сказанного возникают задачи оптимизационного характера с дополнительными условиями: для противодействия сговору требуется деактивировать как можно меньше объектов, притом что деактивировать некоторые объекты в принципе

недопустимо. Такого рода задачи имеют комбинаторную природу (требуется перебирать большое число различных кандидатов в решения). Конечно же, в подобных ситуациях применение имитационного моделирования, вообще говоря, не гарантирует приемлемый результат.

С нашей точки зрения наиболее естественный подход к описанным проблемам состоит в их сведении к комбинаторным задачам, для решения которых существуют алгоритмы с обоснованной аргументацией их практической эффективности. Основная цель настоящей статьи состоит в демонстрации возможности теоретической и практической реализации данного тезиса. В качестве основы для формального описания понятий и процессов, напрямую относящихся к компьютерной безопасности, была взята известная модель Take-Grant (TG) [18], [19], [1], [2], [12]. В роли комбинаторной задачи с хорошо развитой алгоритмикой использовалась задача о булевой выполнимости (Boolean Satisfiability Problem), сокращенно обозначаемая SAT [11]. Наконец, для исследования механизмов противодействия ситуациям, приводящим к сговору в сети, применялся подход, близкий к подходу, использованному ранее для анализа феномена конформного коллективного поведения [8], [17].

Приведем краткое описание структуры статьи. Второй раздел содержит необходимые в дальнейшем понятия, касающиеся дискретных динамических систем (ДДС), которые порождаются сетями. Здесь же приведены основные определения, относящиеся к задаче о булевой выполнимости (SAT), а также схематично описана техника сведения к SAT задач поиска циклов в ДДС. В третьем разделе приведено краткое описание известной модели Take-Grant (TG) передачи прав доступа в компьютерных сетях, а также результатов М.Бишопа по анализу ситуаций сговора в рамках TG-модели. Четвертый раздел является основным. В нем вводится понятие графа сговора и описывается эффективный (линейный по сложности) алгоритм выделения графа сговора из графа TG-модели. Данный алгоритм является модификацией алгоритма Бишопа. Затем ставится задача устранения сговора за

счет деактивации как можно меньшего числа активных вершин в графе сговора (задача построения наименьшего по мощности множества, устраняющего сговор). Рассматриваются дискретные динамические системы, порождаемые графами сговора и их модификациями, в которых некоторые изначально активные вершины деактивированы. Установлена прямая связь между тем, что множество деактивированных вершин устраняет сговор, и видом неподвижной точки соответствующей ДДС. Затем задача поиска наименьшего по мощности множества, устраняющего сговор, сводится к SAT с использованием разработанных ранее техник пропозиционального кодирования ДДС. Пятый раздел содержит результаты вычислительных экспериментов.

2. Теоретические основы.

2.1. СЕТИ И ПОРОЖДАЕМЫЕ ИМИ ДИСКРЕТНЫЕ ДИНАМИЧЕСКИЕ СИСТЕМЫ.

Везде далее сеть – это произвольный граф (как правило, ориентированный и размеченный), вершины которого интерпретируют участников некоторого коллектива, часто называемых агентами, а ребра или дуги интерпретируют бинарные отношения на множестве агентов (например, «друзья», «влияние», «доступ» и т.п.). Наша ближайшая цель – дать содержательное определение динамических процессов в сети, при помощи которых в дальнейшем можно будет представлять распространение прав доступа. С этой целью мы свяжем с рассматриваемой сетью дискретную динамическую систему (ДДС) автоматного типа [4]. Такие системы могут определяться различными способами. Мы начнем с описания класса ДДС, нашедших широкое применение при изучении коллективного поведения.

Итак, пусть $G = (V, A)$ – простой ориентированный граф, в котором V – множество вершин, A – множество дуг. Везде далее граф G называем сетью. Произвольная дуга $(w, v) \in A$ будет интерпретировать отношение «агент w влияет на агента v». Для произвольной вершины $v \in V$ определим ее окрестность следу-

ющим образом:

$$V_v = \{w \in V, w \neq v | (w, v) \in A\}.$$

Можно сказать, что окрестность v образована теми вершинами сети, которые напрямую влияют на данную вершину. Под этим влиянием вершина v может менять свое поведение, например, переходить в активное или неактивное состояние. Для отображения такого рода процессов будем связывать с произвольной вершиной $v \in V$ специальную функцию, значениями которой являются элементы некоторого конечного множества Ω_v. Будем рассматривать нашу сеть в дискретные моменты времени $t \in \{0, 1, 2, \dots\}$, полагая, что в каждый конкретный момент t вершине v сопоставлен некоторый $\omega_v(t) \in \Omega_v$ (вообще говоря, некоторый символ – например, 0 или 1).

Пусть $v \in V$ – произвольная вершина сети и $V_v = \{v_1', \dots, v_l'\}$ – окрестность v. Определим правило f_v, в соответствии с которым в момент времени $t+1$ элементам $\omega_{v_1'}(t), \dots, \omega_{v_l'}(t)$ сопоставляется элемент $\omega_v(t+1)$:

(1) $$\omega_v(t+1) = f_v(\omega_{v_1'}(t), \dots, \omega_{v_l'}(t)).$$

Определение 1. *Функцию* $f_v : \Omega_{v_1'} \times \dots \times \Omega_{v_l'} \to \Omega_v$ *вида* (1) *будем называть весовой функцией вершины* v. *Явное задание всех весовых функций вершин сети задает функцию*

(2) $$F_G : \Omega \to \Omega,$$

где $\Omega = \Omega_{v_1} \times \dots \times \Omega_{v_n}$. *Значения* F_G, *рассматриваемые в моменты времени* $t \in \{0, 1, 2, \dots\}$, *будем называть состояниями сети. Произвольное состояние сети в момент* t *будем обозначать через* $W_G(t)$. *Состояние* $W_G(0)$ *называется начальным.*

Если для каждой $v \in V$ имеет место $\Omega_v = \{0, 1\}$, то рассматриваемая сеть называется синхронной булевой сетью (Synchronous Boolean Network, SBN) или сетью Кауффмана. Таким образом, в SBN весовые функции – это булевы функции, для задания которых можно использовать таблицы истинности или формулы.

В оригинальной статье Стюарта Кауффмана [16] рассматривались SBN, которые использовались в роли простых моделей

динамических процессов в генных сетях. Однако природа коллективов, к которым могут применяться ключевые понятия и идеи этой работы, существенно разнообразней. Следуя [16], мы будем представлять функцию (2) в виде специального графа Γ, называемого графом переходов между состояниями или кратко графом состояний (State Transition Graph, STG). Вершины Γ интерпретируют состояния сети, то есть n-кортежи из Ω. Дуги Γ интерпретируют переходы между состояниями сети в моменты t и $t+1$. В некоторых работах Γ называют пространством фазовых переходов, подразумевая аналогию с непрерывными динамическими системами. Часто говорят, что граф Γ представляет поведение дискретной динамической системы (ДДС), заданной сетью G.

В случае ДДС граф Γ всегда содержит конечное число вершин. Так, если G – SBN, то граф Γ содержит 2^n вершин, каждой из которых соответствует двоичное слово длины n, представляющее собой набор значений весовых функций всех вершин сети.

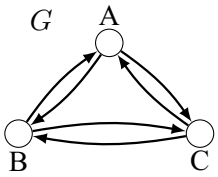
Определение 2. Ввиду конечности числа состояний ДДС, для любого начального состояния $W_G(0)$ обязательно найдутся такие натуральные числа L и M, $0 \leqslant L < M$, что в последовательности состояний

$$W_G(0), W_G(1), \ldots, W_G(M)$$

имеет место: $W_G(L) = W_G(M)$. Обозначим через L_0,M_0 наименьшие L и M, обладающие таким свойством. Последовательность $W_G(L_0), \ldots, W_G(M_0)$ будем называть циклом (в некоторых работах используется термин «аттрактор») длины $M_0 - L_0$. Цикл длины 1 называется стационарным состоянием или неподвижной точкой функции F_G.

На следующем рисунке приведен пример SBN, иллюстрирующий введенные выше понятия. Приведено два способа задания весовых функций вершин рассматриваемой сети – таблицами и формулами.

Поскольку в общем случае имеет место экспоненциальная зависимость числа вершин в графе Γ от числа вершин в сети G, то задачи поиска стационарных состояний и циклов отображений



Γ

(000) → (011) → (110)

(001) → (111) → (101) → (100) → (010)

$f_B(t)$	$f_C(t)$	$f_A(t+1)$
0	0	0
0	1	1
1	0	0
1	1	1

$f_A(t)$	$f_C(t)$	$f_B(t+1)$
0	0	1
0	1	1
1	0	1
1	1	0

$f_A(t)$	$f_B(t)$	$f_C(t+1)$
0	0	1
0	1	0
1	0	0
1	1	1

$$f_A(t+1) = (f_B(t) \vee f_C(t)) \wedge (\neg f_B(t) \vee f_C(t)))$$
$$f_B(t+1) = \neg(f_A(t) \wedge f_C(t))$$
$$f_C(t+1) = (f_A(t) \equiv f_B(t))$$

вида (2) относятся к комбинаторным. При исследовании конкретных сетей для решения данных задач можно использовать различные эффективные на практике комбинаторные алгоритмы. В ряде работ для этих целей использовались алгоритмы решения проблемы булевой выполнимости (SAT) [14], [3], [8], [15], [17]. В следующем разделе мы кратко опишем техники сведения к SAT задач поиска циклов отображений вида (2).

2.2. ЭФФЕКТИВНЫЕ СВОДИМОСТИ К SAT ЗАДАЧ ПОИСКА ЦИКЛОВ ДИСКРЕТНЫХ ФУНКЦИЙ, ЗАДАВАЕМЫХ СЕТЯМИ.

Везде далее функции, преобразующие двоичные слова в двоичные слова, будем называть дискретными функциями. Далее нас будут интересовать дискретные функции типа (2). Как мы увидим, задачи поиска циклов и неподвижных точек функции F_G удобно связывать с задачей обращения данной функции, которая заключается в нахождении по известному образу F_G произвольного его прообраза.

Пусть $\{0,1\}^n$– множество всех двоичных последовательно-

стей длины $n (n \in N)$, а $\{0,1\}^*$ – множество всех двоичных последовательностей произвольной конечной длины.

Рассмотрим произвольную всюду определенную (тотальную) дискретную функцию $f : \{0,1\}^* \to \{0,1\}^*$, заданную некоторым алгоритмом $A(f)$. Алгоритм $A(f)$ задает счетное семейство тотальных функций вида $f_n : \{0,1\}^n \to \{0,1\}^*$. Везде далее мы будем рассматривать только такие ситуации, когда сложность $A(f)$ ограничена полиномом от n.

Задача обращения произвольной функции

$$f_n : \{0,1\}^n \to \{0,1\}^m \tag{3}$$

состоит в том, чтобы по известному $\gamma \in Range f_n$ найти такой $\alpha \in \{0,1\}^n$, что $f_n(\alpha) = \gamma$.

Хорошо известно, что для рассматриваемого класса функций описанная задача обращения эффективно (за полиномиальное от n время) сводится поиску выполняющего набора выполнимой конъюнктивной нормальной формы (КНФ). По сути, данный факт является прямым следствием известной теоремы Кука-Левина [13], [5].

Задача SAT (Boolean Satisfiability Problem) заключается в ответе на вопрос, выполнима ли произвольная КНФ C. SAT в распознавательной формулировке – классическая NP-полная задача. Тем не менее, в последние 20 лет наблюдается выдающийся прогресс в разработке вычислительных алгоритмов решения SAT, применимых к различным тестам индустриального характера. В настоящее время лучшие алгоритмы решения SAT справляются с КНФ, содержащими десятки тысяч переменных и сотни тысяч дизъюнктов. Такие алгоритмы используются в символьной верификации, биоинформатике, комбинаторике и криптоанализе. Опишем кратко суть процедур перехода от задачи обращения дискретной функции к SAT.

Итак, рассматриваем задачу обращения произвольной функции вида (3). Заметим, что произвольную такую функцию мы можем рассматривать как набор из m булевых функций арности n. Соответственно, функцию (3) мы можем задать при помощи схемы из функциональных элементов S_{f_n} над любым пол-

ным базисом [6]. У схемы S_{f_n} будет n входов и m выходов. Существует специальная техника, называемая преобразованиями Цейтина [7], которая произвольной схеме S_{f_n} ставит в соответствие CNF C_{f_n} над множеством булевых переменных U, $X \subset U$. Здесь $X = \{x_1, \ldots, x_n\}$ – переменные, приписанные входам схемы S_{f_n}, а $U \backslash X$ – переменные, приписанные всем гейтам S_{f_n}. Сложность (time complexity) процедуры построения C_{f_n} по S_{f_n} линейна от общего числа узлов схемы S_{f_n}. В U выделим подмножество Y переменных, приписанных выходам S_{f_n}. Пусть $\gamma \in \{0,1\}^m$ – произвольный набор значений переменных из Y. Означим в C_{f_n} все переменные из Y их значениями из γ и проделаем все возможные элементарные преобразования над полученной формулой. Обозначим результирующую CNF через $C_{f_n}(\gamma)$. Из свойств преобразований Цейтина следует, что если $\gamma \in Range f_n$, то $C_{f_n}(\gamma)$ выполнима, и из любого выполняющего ее набора можно выделить такой набор α значений переменных из X, что $f_n(\alpha) = \gamma$. Тем самым задача обращения произвольной функции вида (3) сведена к SAT.

В соответствии с приведенной схемой можно эффективно преобразовать задачу поиска цикла произвольного отображения вида (2) к SAT. Проиллюстрируем это на примере SBN. Пусть $G = (V, A)$ – SBN на n вершинах, весовые функции которых заданы булевыми формулами. Легко видеть, что функция (2) в данном случае – это дискретная функция вида

$$F_G : \{0,1\}^n \to \{0,1\}^n, \tag{4}$$

которую можно задать простым алгоритмом. Используя описанную выше технику, построим CNF C_{F_G}. Пусть U – множество булевых переменных, фигурирующих в C_{f_n}, и пусть $X = \{x_1, \ldots, x_n\}$ и $Y = \{y_1, \ldots, y_n\}$ – переменные, приписанные входам/выходам схемы S_{F_G}. Рассмотрим следующую CNF:

$$C_{F_G} \wedge (x_1 \equiv y_1) \wedge \cdots \wedge (x_n \equiv y_n), \tag{5}$$

где через $\equiv$ обозначена логическая эквивалентность. В свете сказанного выше несложно видеть, что CNF (5) выполнима тогда и только тогда, когда отображение F_G имеет неподвижные точки. Если (5) выполнима и μ – выполняющий её набор, то из него

можно выделить набор α переменных из X, который определяет стационарное состояние в STG $\Gamma(F_G)$. Подобным же образом к SAT может быть сведена и задача поиска циклов произвольной длины в графах состояний отображений типа (2). В случае, когда G – это SBN на n вершинах, для этой цели можно рассмотреть функцию F_G^t, являющуюся t-кратной суперпозицией функции вида (4), где t – длина искомого цикла. Соответствующая схема $S_{F_G^t}$ – это, по сути, цепочка из t схем вида S_{F_G}, в которой входы каждой последующей схемы соединены с выходами предшествующей. К КНФ $C_{F_G^t}$ по аналогии с (5) потребует приписать условие на совпадение состояний на входе и выходе F_G^t.

Описанный здесь алгоритм перехода от задачи поиска циклов в STG отображений типа (2) к SAT относится к классу процедур, называемых пропозициональными кодированиями. В работах [8], [17] более детально описаны процедуры пропозиционального кодирования ряда задач активационной динамики на сетях Кауффмана.

3. Модели компьютерной безопасности семейства Take-Grant и их основные свойства.

Базовая модель Take-Grant (TG) [18], [19] – это формальная модель, интерпретирующая процессы передачи прав доступа в компьютерной сети. Сеть представляется в рамках TG-модели в виде простого ориентированного размеченного графа $G = (O, A)$.

O – множество всех вершин графа, интерпретирующих объекты. В множестве O выделяют специальное множество S, т.н. «активных объектов», называемых также субъектами – субъекты могут совершать некоторые действия в сети, например, брать или передавать другим объектам/субъектам какие-либо права доступа. Субъекты обозначаются в графе как ●. Объекты, не являющиеся субъектами (то есть неактивные), обозначаются как ○.

$R = \{r_1, r_2 \dots r_n\} \cup \{t, g, r, w\}$ – множество видов прав доступа. Где:

- t – право брать права доступа;
- g – право давать (делегировать) права доступа;
- r – право на чтение;
- w – право на запись;

A – множество дуг графа: $A \subseteq O \times O \times R$. Таким образом, дуги интерпретируют в TG права доступа между объектами.

Изменениям состояний системы соответствуют изменения графа. В классической TG-модели выделено два типа базовых правил, определяющих передачу прав доступа между объектами: $take(P, x, y, z)$ и $grant(P, x, y, z)$. Через x, y, z здесь обозначены объекты, причем x – обязательно субъект. Через P обозначено множество прав доступа, которые в случае правила «$take$» x берет у y и передает z; в случае правила «$grant$» x передает свои права доступа P к объекту z некоторому объекту y (в соответствии с отношением $x\ grant\ y$). Базовые правила $take$ и $grant$ называются де-юре правилами.

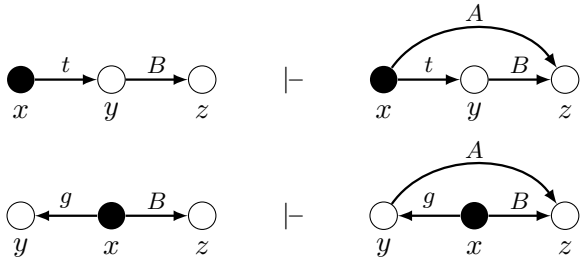


Рис. 1. Иллюстрация работы правил «take» и «grant» в базовой TG-модели.

На рисунке 1 показано, что в некоторый момент времени субъект x, имеющий право брать у объекта y некоторые права доступа на объект z, получает часть прав $(A \subseteq B)$ на объект z. Как результат в графе доступов возникает новая дуга (x, z) помеченная A. Так работает правило $take$. Работа правила $grant$ продемонстрирована на втором рисунке в аналогичном стиле.

Также в классической TG-модели присутствуют т.н. «де-факто правила», в том числе и правила, позволяющие создавать или удалять новые объекты. В расширенной TG-модели отдельно рассматриваются несколько правил, регламентирующих работу с информационными потоками – соответствующие дуги интерпретируют отношения типа $x\ read\ y$ (x читает данные y) и $x\ write\ y$ (x модифицирует данные y). Весьма детально базовая модель TG и ее расширения рассмотрены в [1], [2]. Для варианта Take-Grant без возможности создания и удаления объектов в [18], [19] были предложены эффективные (линейные от числа вершин в графе G) алгоритмы проверки ряда свойств, определяющих безопасность системы.

Основные интересующие нас вопросы будут касаться ситуаций получения в рамках модели TG прав права вида $p\ read\ q$ в результате вступления в сговор некоторых объектов сети. Соответствующие проблемы были поставлены и изучены М. Бишопом в работе [12]. Сговор между участниками сети в [12] явно не определяется, однако по смыслу это ситуация, в которой сотрудничество ряда объектов/субъектов может привести к появлению права доступа на некоторый критический объект в сети. Важный факт заключается в том, что сговор может быть успешным лишь тогда, когда среди объектов, вступающих в сговор, имеется необходимое число активных объектов, в отношении которых М. Бишоп использует термин «$actors$». Именно такие объекты выполняют роль «проводников» информации, обеспечивая передачу прав по сети. Основные достижения работы [12] – это эффективные (полиномиальные от числа вершин в графе) процедуры проверки возможности похищения прав в результате сговора. Также в [12] устанавливается явная связь между числом активных объектов $(actors)$ в специальном графе действий $(acting\ graph)$ и успешностью сговора. Проиллюстрируем сказанное на следующем примере из [12].

В верхней части рисунка 2 изображен фрагмент графа доступа компьютерной сети, рассматриваемой в рамках модели Take-Grant. Вершины графа помечены латинскими буквами. В данном

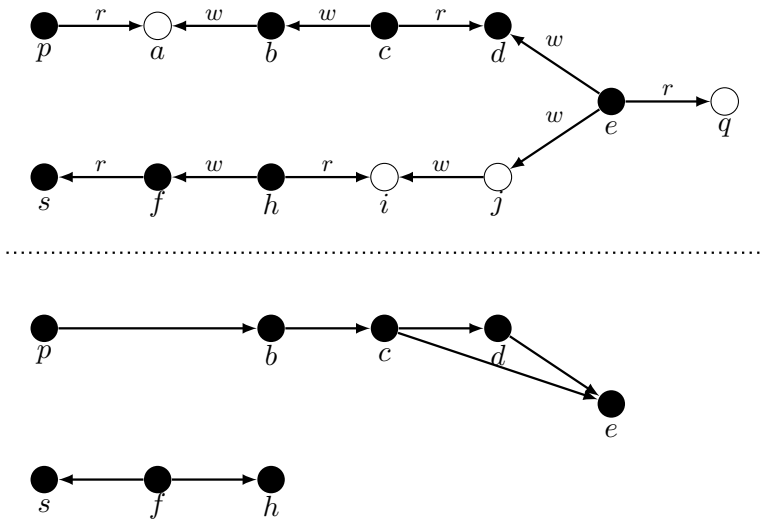


Рис. 2. Фрагмет сети Take-Grant и пример графа действий из работы [12]

примере по сети распространяются только два права: $read(r)$ и $write(w)$. Активные объекты $(actors)$ обозначены сплошными точками, неактивные – пустыми. В нижней части рисунка 2 изображен соответствующий граф действий $(acting\ graph)$. Произвольная дуга (v_1, v_2) в графе действий означает получение права $read$ объекта v_1 на данные объекта v_2. Можно заметить, что вершина p получает право на вершину b, поскольку b пишет в a (которая сама по себе неактивна), а p читает из a. Аналогично, c читает из d и затем пишет в b, соответственно, у p возникает право читать данные d. И так далее. Как итог, p может читать данные объекта q. Описанная ситуация дает пример сговора по похищению права на доступ к данным вершины q: если вершины из множества $\{p, b, c, d, e\}$ действуют так, как только что было сказано, то p получает доступ к данным на q. С другой стороны, вершина s не может получить доступ к данным q. И, что очень важно, этому препятствуют две неактивных вершины i, j. Несмотря на право j писать в i, вершина j этого не делает, поскольку неактив-

на (здесь принципиальное отличие от предыдущего случая, где от вершины a не требовалось никаких действий). Соответственно, информация от q не проходит дальше вершины j.

4. Противодействие сговору в ДДС на базе модели Take-Grant (качественный анализ).

Еще раз отметим, что некоторые задачи анализа свойств безопасности сетей в рамках TG-модели, рассмотренные в работах [18], [19] и [12], решаются за полиномиальное время. Тем не менее, несложно описать задачи, естественным образом связанные с рассмотренными, но при этом являющиеся комбинаторными по своей природе. Далее мы концентрируемся на изучении задач такого типа. Следуя [12], будем рассматривать распространение по сети только прав $read$ и $write$. Дополнительно отметим, что (как и в работах [18], [19], [12]) во всех рассматриваемых далее сетях запрещено создавать новые и удалять имеющиеся объекты.

Сначала обратимся к задаче выявления множеств объектов, сговор между которыми приводит к появлению права $read$ у объекта p на объект q. Предварительно уточним ряд понятий.

Определение 3. *Пусть $G(O, A)$ – граф модели Take-Grant до начала каких-либо действий по распространению прав. Все дуги из A будем называть базовыми. Создание нового права между двумя объектами интерпретируется в Take-Grant новой дугой, не входящей в A. Будем называть такие дуги приобретенными.*

Теперь предположим, что в результате распространения прав в рассматриваемой TG-модели возникло право $p\ read\ q$. В работе [12] был предложен эффективный (на самом деле линейный от числа вершин в исходном графе) алгоритм построения графа действий (acting graph) G'. Все вершины, входящие в G', – это активные объекты, принимающие участие в передаче прав от q к p. Для дальнейших целей нам более удобен граф, который содержит вообще все объекты (как активные, так и неактивные), при участии которых возникает отношение $p\ read\ q$.

Определение 4. *Граф $G_{pq} = (O_{pq}, A_{pq})$, который содержит*

все вершины и базовые дуги G, участвующие в передаче прав от q к p, будем называть графом сговора. Произвольное множество O′,O′ ⊆ O_{pq}, объекты которого, взаимодействуя в рамках правил модели TG, обеспечивают передачу прав от q к p, будем называть множество сговора. Множество O_{pq} будем называть полным множеством сговора.

Еще раз подчеркнем, что множество O_{pq} может содержать неактивные объекты (в рассмотренном выше примере такой объект представлен вершиной a). Анализируя алгоритм, приведенный в [12], несложно его модифицировать с целью построения графа G_{pq}. Соответствующий алгоритм неформально описан ниже.

Алгоритм 1 (Выделение графа сговора G_{pq})**.**

1. Проследить развитие сети в соответствии с правилами Take-Grant, запоминая для каждой приобретенной дуги те дуги (как базовые, так и приобретенные), благодаря которым она возникла (такие дуги называем предками);

2. в конечном состоянии сети (после которого не возникает новых дуг) убедиться, что между вершинами p, q существует отношение $p\ read\ q$, которое отсутствовало в начальном состоянии;

3. рекурсивно обойти предков приобретенной дуги (p, q);

4. те вершины графа G и его базовые дуги, которые были пройдены в процессе выполнения шага 3, образуют граф сговора $G_{pq} = (O_{pq}, A_{pq})$.

Нетрудно убедиться, что для конкретных заданных $p, q \in O$ сложность описанного алгоритма построения G_{pq} линейна от числа объектов (вершин) в рассматриваемой сети.

Определение 5. Пусть $G_{pq} = (O_{pq}, A_{pq})$ – граф сговора по передаче прав от q к p. Пусть O′,O′ ⊆ O_{pq}, произвольное множество сговора. В O′ исключим из рассмотрения произвольную вершину v, обозначив полученное множество через O″. Если для любой вершины v ∈ O′ соответствующее множество O″ не является множеством сговора по передаче прав от q к p, то множество O′ назовем каналом сговора в графе G_{pq}.

Грубо говоря, канал – это один конкретный пример реали-

зации сговора. Множество различных каналов, обеспечивающих отношение $p\ read\ q$ в графе G_{pq}, может быть очень большим. Наша ближайшая задача – научиться блокировать такие каналы. Как уже было сказано, передача прав в сети возможна только благодаря активным объектам. Соответственно, естественный способ блокирования каналов сговора – это придание некоторым активным объектам неактивного статуса (деактивация). Заметим, однако, что на практике далеко не каждый объект может быть деактивирован.

Определение 6. *Пусть $G_{pq} = (O_{pq}, A_{pq})$ – граф сговора по передаче прав от q к p. Деактивируем один или несколько объектов из O_{pq}. Любое подмножество в O_{pq}, которое до деактивации было каналом сговора, а после деактивации перестало им быть, назовем фиктивным каналом. Множество вершин $D \subseteq O_{pq}$, деактивация которых делает фиктивными все каналы сговора в G_{pq}, будем называть множеством, устраняющим сговор по передаче прав от q к p.*

Далее мы будем рассматривать задачу построения множества D наименьшей возможной мощности, а также задачу построения множеств типа D при дополнительных запретах на деактивацию некоторых вершин. Заметим, что сформулированные задачи интуитивно являются комбинаторными – так например, задача поиска деактивирующего множества наименьшей мощности похожа по постановке на известную NP-трудную задачу поиска наименьшего независимого множества. В настоящей статье мы, однако, не ставим вопрос выяснения структурной сложности изучаемых задач, а предлагаем вычислительные алгоритмы, которые могут быть использованы для их решения.

В данном контексте важно отметить следующий нюанс: деактивация некоторого активного объекта в канале, обеспечивающем право $p\ read\ q$, может не дать фиктивный канал. Соответствующий пример приведен на рисунке 2 – в канале (p, a, b, c, d, e, q) вершина a неактивна, но данный канал не является фиктивным. Таким образом, на первый взгляд непонятно, как можно было бы свести рассматриваемую задачу к «традици-

онным» комбинаторным задачам на графах (типа поиска максимальных клик или независимых множеств).

Далее для решения задач блокирования каналов сговора в сети обратимся к дискретным динамическим системам и описанным выше техникам их пропозиционального кодирования. Мы начнем с того, что свяжем с рассматриваемым вариантом TG-модели дискретную динамическую систему. Более точно, нас будет интересовать ДДС, порождаемая графом сговора. Ниже приведено описание этой ДДС.

Итак, пусть $G = (O, A)$ – исходная сеть, рассматриваемая в рамках модели Take-Grant, описывающей распространение прав $read$ и $write$. Пусть $p, q \in O$ – фиксированные вершины сети и $G_{pq} = (O_{pq}, A_{pq})$ – эффективно выделенный из исходной сети граф сговора по передаче прав от q к p. Будем рассматривать распространение информации по графу G_{pq} как дискретный динамический процесс, шаги которого осуществляются в моменты времени $t \in \{0, 1, \dots\}$.

Выделим в множестве $O_{pq} \times O_{pq}$ подмножество A^0_{pq}, образованное всеми теми парами вида (u, v), в которых u не имеет права $read$ на v в графе G_{pq}. Каждому моменту $t \in \{0, 1, \dots\}$ поставим в соответствие граф G^t_{pq}. Граф G^0_{pq} совпадает с графом G_{pq}. При переходе от $t = 0$ к $t = 1$ для любой пары вершин $(u, v) \in A^0_{pq}$ проверяется возможность возникновение права $u\ read\ v$. Если такое право возникает, оно интерпретируется новой дугой (u, v). Множество пар из A^0_{pq}, для которых отношение «$u\ read\ v$» не возникло, обозначается через A^1_{pq}. При переходе от $t = 1$ к $t = 2$ аналогичным образом проверяются все пары из A^1_{pq}. И так далее. Ввиду определенных выше правил, через конечное число M моментов времени данный процесс перестанет порождать новые дуги. Рассмотрим граф G^M_{pq}. Все дуги в G^M_{pq}, отличные от дуг из G_{pq}, порожденные описанным выше процессом, будем называть пунктирными. Очевидно, что каждой пунктирной дуге соответствует некоторая приобретенная дуга в смысле определения 3. Удалим из G^M_{pq} все дуги графа G_{pq}. Обозначим получившийся граф через $G^*_{pq} = (O_{pq}, A^*)$. Множество A^*, таким образом,

включает только пунктирные дуги.

Пусть $A^* = \{a_1, \ldots, a_K\}$, причем $a_K = (p, q)$ (очевидно, что пунктирная дуга a_K заведомо существует, поскольку G_{pq} – граф сговора по передаче прав от q к p). В каждый момент времени $t \in \{0, 1, \ldots, M\}$ произвольной дуге $a \in A^*$ поставим в соответствие вес $w_a(t) \in \{0, 1\}$. Факт $w_a(t) = 0$ интерпретирует отсутствие в момент t права $read$ между соответствующими объектами, а $w_a(t) = 1$ интерпретирует наличие данного права. Вектор $W(t) = (w_{a_1}(t), \ldots, w_{a_K}(t))$ – это состояние нашей ДДС в момент t. Начальное состояние соответствует состоянию рассматриваемой TG-модели до начала каких-либо действий по распространению прав. В этот момент всем пунктирным дугам приписан вес 0. В моменты времени $t \in \{1, 2, \ldots, M\}$ веса пунктирных дуг пересчитываются фактически в соответствии с правилами выделения графа сговора приведенным выше алгоритмом (Алгоритм 1). Иными словами, для каждого момента в отношении конкретной пунктирной дуги проверяется, возникла ли в рамках TG-модели соответствующая приобретенная дуга; в случае ответа «да» рассматриваемая пунктирная дуга получает вес 1, в противном случае она получает вес 0. Как итог имеем ДДС, которую будем обозначать через $\Delta(G^*_{pq})$. Ввиду всего сказанного выше несложно заметить, что данная ДДС не имеет циклов длины > 1 и имеет единственную неподвижную точку – состояние $W(M)$.

Рассмотрим граф $G^*_{pq} = (O_{pq}, A^*)$ и рассмотрим произвольное множество $D, D \subseteq O_{pq}$. Промаркируем в O_{pq} все вершины, входящие в D, как неактивные. Полученное множество обозначим через $\widetilde{O}_{pq}$. Рассмотрим граф $\widetilde{G}_{pq} = (\widetilde{O}_{pq}, A^*)$. Построим на основе графа $\widetilde{G}_{pq}$ ДДС, действующую в точности по тем же правилам, по которым действует ДДС $\Delta(G^*_{pq})$. Обозначим данную ДДС через $\Delta(\widetilde{G}_{pq})$. Установим справедливость следующего факта.

Теорема 1. *Множество вершин $D \subseteq O_{pq}$ устраняет сговор по передаче прав от q к p тогда и только тогда, когда в неподвижной точке системы $\Delta(\widetilde{G}_{pq})$ вес дуги a_K равен 0.*

Доказательство. Докажем необходимость. Пусть D – множество, устраняющее сговор по передаче прав от q к p. Это, согласно определению 4, означает, что деактивация вершин из D делает фиктивными все каналы в G_{pq}, обеспечивающие передачу прав от q к p. Если предположить, что в неподвижной точке системы $\Delta(\widetilde{G}_{pq})$ дуга e_K имеет вес 1, то это, согласно правилам построения $\Delta(\widetilde{G}_{pq})$, будет означать, что в процессе перехода данной системы от начального состояния к неподвижной точке у объекта p возникло право $read$ на объект q. Следовательно, некоторый канал в графе сговора после деактивации вершин из D не является фиктивным. Полученное противоречие доказывает необходимость. Наоборот, пусть в неподвижной точке системы $\Delta(\widetilde{G}_{pq})$ дуга e_K имеет вес 0. Предположим, что при этом существует канал сговора в G_{pq}, не являющийся фиктивным. Но тогда в процессе входа системы $\Delta(\widetilde{G}_{pq})$ в неподвижную точку обязательно должно возникнуть право $p\ read\ q$, соответственно, дуга e_K в этом случае получила бы вес 1. Данное противоречие доказывает достаточность. Теорема 1 доказана.

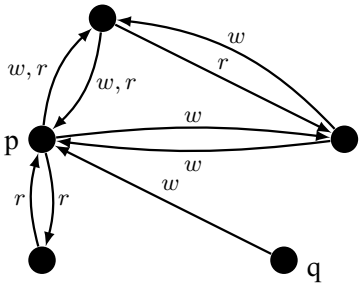


Рис. 3. Пример графа сговора

5. Результаты вычислительных экспериментов

Опишем кратко вычислительные алгоритмы, которые позволяют строить устраняющие сговор множества наименьшей мощности. В основе разработанных алгоритмов лежат те же идеи, ко-

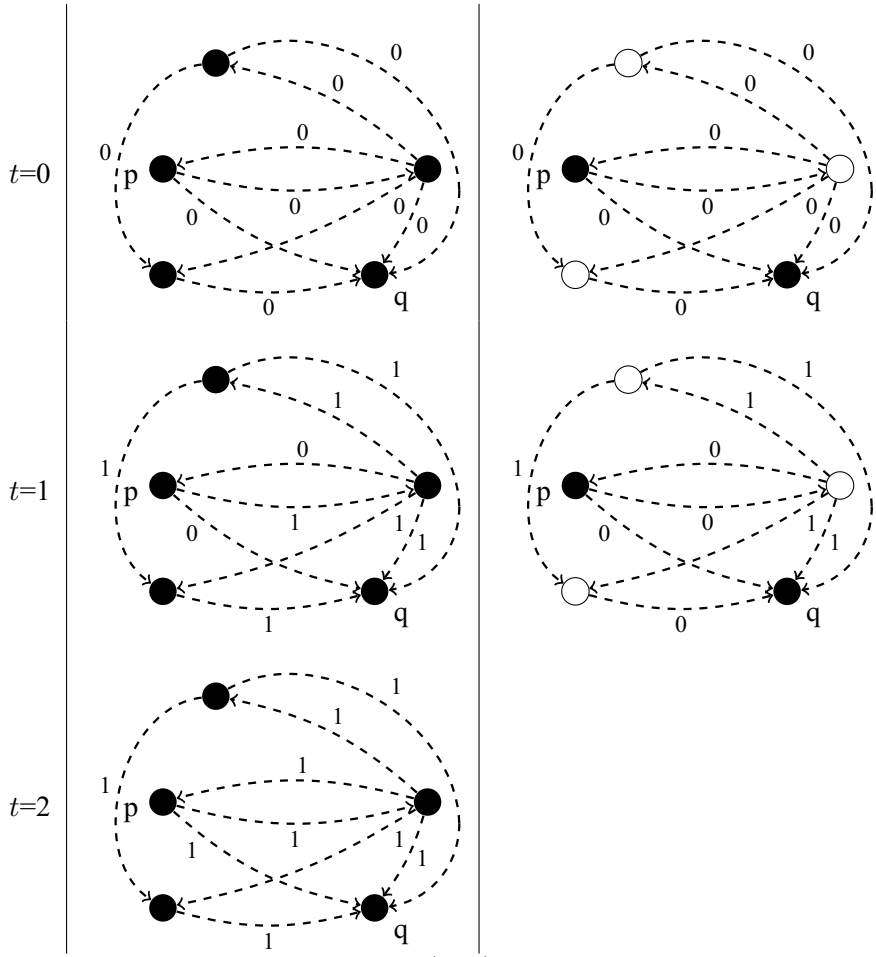


Рис. 4. Слева изображена ДДС $\Delta(G^*_{pq})$, построенная на основе графа сговора, приведенного на рисунке 3. Справа ДДС $\Delta(\widetilde{G}_{pq})$. В первом случае ДДС входит в неподвижную точку в момент $t = 2$, во втором случае в момент $t = 1$.

торые были использованы в статье [17]. Более точно, рассматривается конкретный граф сговора G_{pq}. Относительно каждой вершины из O_{pq} за исключением специально выделенных (вершина q, а также вершины, находящиеся слишком близко к q в смысле расстояния в графе) делается предположение о ее потенциально возможном попадании в D. Данный факт кодируется при помощи специально вводимых булевых переменных (значение 1 такой переменной соответствует тому факту, что соответствующая вершина попадает в D, значение 0 означает, что рассматриваемая вершина не попадает в D). Затем мы кодируем процесс входа системы $\Delta(\widetilde{G}_{pq})$ в неподвижную точку. Из сказанного выше следует, что для этой цели достаточно закодировать последовательность переходов $\Delta(\widetilde{G}_{pq})$ между состояниями $W(0), \ldots, W(M)$. Затем мы добавляем условие $w_{e_K}(M) = 0$, а также условие, что число точек из O_{pq}, оказавшихся в D, не превосходит некоторого числа d. Также мы используем дополнительные ограничения на возможность вершин из O_{pq} попадать в D. В частности, мы запрещаем попадать в D вершинам, находящимся от q (целевой вершины) на расстоянии, меньшем заданного значения (в смысл стандартного расстояния в графе). Для кодирования последнего условия так же, как и в [17], используются известные техники работы с мощностными ограничениями (cardinality constraints) [9].

В тестовых экспериментах в роли исходных графов модели Take-Grant использовались случайные графы на 200 вершинах, сгенерированные в соответствии с моделью Барабаши-Альберт [10]. На исходный граф модели Take-Grant накладывалось ограничение, что число активных вершин (субъектов) в нем не должно превышать 40. В каждом таком графе строился граф сговора G_{pq} (вершины p и q выбирались случайным образом). Всего в соответствии с описанной схемой было сгенерировано 700 графов модели Take-Grant, для которых число вершин в графе G_{pq} варьировалось от 30 до 50. Для каждого такого G_{pq} ставилась задача построения множества D (устраняющего сговор по передаче прав от q к p) наименьшей мощности. При этом мы накладывали запрет на деактивацию любой вершины в G_{pq}, из которой су-

ществует путь в q по дугам G_{pq} длины $\leqslant 6$. Поставленная так задача сводилась к SAT с использованием описанной выше техники. Средний размер (по 700 задачам) построенных КНФ составил 126 мегабайт. Для решения SAT задач использовался SAT-решатель minisat2.2 [20]. Все SAT-задачи были решены. Среднее время решения составило 0,11 секунды. Использовалась вычислительная платформа следующей конфигурации: процессор Core i7-2670QM, 8Gb RAM. Примерно в 50 случаях наименьшее множество, устраняющее сговор, имело размер от 3 до 5 вершин. В большинстве случаев находилось D мощности менее чем из 3 вершин. Примерно в 100 тестах SAT-решатель выдал UNSAT. Это означает, что в соответствующих случаях при имеющихся ограничениях (на расстояние от q до вершин, которые допустимо деактивировать) устранить сговор невозможно.

6. Заключение

В статье представлен подход к анализу и устранению ситуаций сговора, возникающих в рамках известной модели Take-Grant передачи прав доступа в компьютерных сетях. В основе подхода лежит эффективный алгоритм выделения в графе TG-модели т.н. «графа сговора». Данный алгоритм является модификацией известного алгоритма построения графа действий, предложенного М. Бишопом в 1996 году. Деактивируя в графе сговора некоторые вершины, можно полностью устранить ситуацию сговора по передаче прав между двумя выделенными вершинами. Соответствующее множество деактивируемых вершин называется множеством, устраняющим сговор. Мы рассматриваем задачу построения наименьшего по мощности множества, устраняющего сговор. Данная задача является комбинаторной по своей природе, и для ее вычислительного решения мы сводим ее к задаче о булевой выполнимости (SAT). Для сведения мы строим на основе графа сговора дискретную динамическую систему и используем известные техники пропозиционального кодирования таких систем. Описанный в статье подход к устранению ситуаций сговора в компьютерных сетях был программно реализован

и протестирован на случайных графах TG-модели размерностью 200 вершин. Задачи построения устраняющих сговор множеств наименьшей мощности удавалось успешно решать на обычном персональном компьютере при помощи SAT решателя minisat2.2.

Литература

1. ДЕВЯНИН П.Н. *Модели безопасности компьютерных систем*: учеб. пособие. – 2005.
2. ДЕВЯНИН П.Н. *Модели безопасности компьютерных систем. Управление доступом и информационными потоками*: учеб. пособие. – 2011.
3. ЕВДОКИМОВ А.А., КОЧЕМАЗОВ С.Е., СЕМЕНОВ А.А. *Применение символьных вычислений к исследованию дискретных моделей некоторых классов генных сетей.*: Вычислительные технологии. – 2011. – Т. 16, №1. – С. 30–47.
4. *Системная компьютерная биология*: [под ред. Н.А. Колчанова]. – Новосибирск: Изд-во Сибирского отделения Российской академии наук, 2008. – 769 с.
5. ЛЕВИН Л.А. *Универсальные задачи перебора*: Пробл. передачи информ. –1973. – Т. 9, №. 3. – Р. 265-–266.
6. НИГМАТУЛЛИН Р.Г. *Сложность булевых функций*. – М.: Наука. Главная редакция физико-математической литературы, 1991. – 240 с.
7. ЦЕЙТИН Г.С. *О сложности вывода в исчислении высказываний*: Записки научных семинаров ЛОМИ АН СССР. – 1968. – Т.8. – С. 234–259.
8. СЕМЁНОВ А. А., КОЧЕМАЗОВ С. Е. *О дискретно-автоматных моделях конформного поведения*: Управление большими системами. Выпуск 46. М.: ИПУ РАН – 2013. – С. 266–292.
9. ASIN R., NIEUWENHUIS R., OLIVERAS A., RODRIGUEZ-CARBONELL E. *Cardinality networks: a theoretical and empirical study*: Constraints. – 2011. – V. 16, No. 2. – P. 195-–221.

10. BARABASI A.L., ALBERT R.: Emergence of scaling in random networks. Science. – 1999. – V. 286. P. 509-–512.
11. BIERE A., HEULE M., VAN MAAREN H., WALSH T. *Handbook of Satisfiability*: Frontiers in Artificial Intelligence and Applications. – 2009. – T. 185
12. BISHOP M. *Conspiracy and Information Flow in the Take-Grant Protection Model*: Journal of Computer Security – 1996 – V. 4, No. 4. – P. 331–359.
13. COOK S. *The complexity of theorem-proving procedures*: STOC ’71 Proceedings of the third annual ACM symposium on Theory of computing. – 1971. – P. 151–158.
14. DUBROVA E., TESLENKO M. *A SAT-based algorithm for finding attractors in synchronous boolean networks*: IEEE/ACM Trans Comput Biology Bioinform. – 2011. – V. 8. – P. 1393–1399.
15. EVDOKIMOV A.A., KOCHEMAZOV S.E., OTPUSHCHENNIKOV I.V., SEMENOV A.A. *Study of discrete automaton models of gene networks of non-regular structure using symbolic calculations*: Journal of Applied and Industrial Mathematics. – 2014. V. 8, No. 3. – P. 307–316.
16. KAUFFMAN S. *Metabolic stability and epigenesis in randomly constructed genetic nets*: Journal of Theoretical Biology. – 1969. – V. 22, – P. 437–467.
17. KOCHEMAZOV S., SEMENOV A. *Using Synchronous Boolean Networks to Model Several Phenomena of Collective Behavior.*: PLOS ONE. – 2014. – V. 9, No. 12, e115156. – P. 1–28.
18. JONES A., LIPTON R., SNYDER L. *A Linear Time Algorithm for Deciding Security*: Proc. 17th Annual Symp. on the Foundations of Computer Science – 1976. – P. 33–41.
19. LIPTON R., SNYDER L. *A Linear Time Algorithm for Deciding Subject Security*: J. ACM. – 1977. – V. 24, No. 3. – P. 455–464.
20. *The MiniSat Page* [Электронный ресурс]. – http://minisat.se.

COUNTERACTION TO CONSPIRACY IN DISCRETE DYNAMICAL MODELS OF COMPUTERS NETWORK

Dmitriy Gorbatenko, Matrosov Institute for System Dynamics and Control Theory, Irkutsk, postgraduate student (gorbadima@yandex.ru).
Alexander Semenov, Matrosov Institute for System Dynamics and Control Theory, Irkutsk, Cand.Sc., assistant professor (biclop.rambler@yandex.ru).

Abstract: In this paper we present a new approach to counteracting conspiracy in computer networks. The algorithm is proposed to extract the so-called "conspiracy graphs" within the context of the known "Take-Grant" model (TG). Based on a conspiracy graph a discrete dynamical system (DDS) of automaton type is constructed. The situation, when the data of a target vertex is compromised, is considered as a result of a process which takes place inside an introduced DDS. This process can be blocked by deactivating some vertices in a conspiracy graph. The problem of constructing the deactivating set of the smallest possible size is considered. It is reduced to Boolean satisfiability problem (SAT) and solved using state-of-the-art SAT solver.

Keywords: discrete dynamical systems, Take-Grant model, conspiracy graph, SAT.

Статья представлена к публикации членом редакционной коллегии . . .

Поступила в редакцию . . .
Дата опубликования . . .