

В.В. Ерохин
Х.М. Бахадиров

**ВЕРИФИКАЦИЯ ИНФОРМАЦИИ
И ЗАЩИТА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ
В ИНФОРМАЦИОННО-ТЕЛЕКОММУНИКАЦИОННЫХ
СИСТЕМАХ БАНКА**

Монография



Москва 2021

УДК 004.056.5:004.627

ББК 32.972.53

Е 78

Авторы:

доктор технических наук, доцент, профессор кафедры математических методов и бизнес-информатики Московского государственного института международных отношений (университета) МИД России

Ерохин Виктор Викторович;

магистрант направления «Бизнес-информатика»

Московского государственного института международных отношений
(университета) МИД России

Бахадиров Хикмет Муратович

Научный редактор:

доктор технических наук

Лагерев Александр Валерьевич

Рецензенты:

доктор технических наук

Гольдфарб Вениамин Иосифович;

доктор технических наук

Лагерев Игорь Александрович

Ерохин В.В., Бахадиров Х.М.

Е 78

Верификация информации и защита программного обеспечения в информационно-телекоммуникационных системах банка: Монография.– М.: Издательство «Спутник +», 2021. – 181 с.

ISBN 978-5-9973-5919-5

Излагаются основные аспекты и характеристики технологии защиты информации в банковских телекоммуникационных системах, а также подходы к анализу информационной безопасности банковских систем. Решаются задачи обеспечения информационной верификации информации в компьютерных и экономических системах банка.

Монография предназначена для специалистов в области защиты компьютерных систем и студентов по направлениям подготовки 09.04.01 – «Информатика и вычислительная техника», 09.04.02 – «Информационные системы и технологии», 38.04.05 – «Бизнес-информатика».

Ил. 49. Табл. 19. Библиогр. – 74 назв.

УДК 004.056.5:004.627

ББК 32.972.53

Отпечатано с готового оригинал-макета.

ISBN 978-5-9973-5919-5

© Ерохин В.В., Бахадиров Х.М., 2021

ВВЕДЕНИЕ

В предлагаемой монографии излагаются основные принципы и положения формирования общих знаний в области безопасности экономических систем, в частности, коммерческих банков.

Современные коммерческие банки всё чаще сталкиваются с проблемой нарушения параметров защиты информационного и программного обеспечения банка, связанной с несанкционированными действиями легальных пользователей – сотрудников банка. Ведущие коммерческие банки страны ставят задачи усовершенствования механизмов и технологий контроля работы с информационным и программным обеспечением. В настоящее время банки используют интегрированные информационные среды, технологии классов CALS, ERP, MRP, SCADA, CAD/CAM/CAE, применение которых предполагает наличие высокого уровня защиты информационного и программного обеспечения. Существует ряд систем предотвращения несанкционированного использования информационного и программного обеспечения, но ни одна из них не обеспечивает комплексной защиты ресурсов банка. Необходимость обеспечения комплексной защиты информационных и программных ресурсов требует разработки единой, интегрированной в основную информационную среду банка, системы, основной задачей которой является предотвращение максимального количества видов «инсайдерских» атак. Сложность структуры информационного и программного обеспечения коммерческого банка определяет сложность структуры разрабатываемой автоматизированной системы разграничения доступа к информационному и программному обеспечению (АСРДкИПО).

Большинство вычислительных систем, используемых для технических, инженерных, коммерческих, научных целей имеют сложную архитектуру. Будучи состоящими из множества взаимодействующих компонентов, современные вычислительные системы подвержены сбоям, причем сбои могут происходить как вследствие отказа оборудования, так и вследствие некорректной работы программного обеспечения. В этой связи, все большее значение и актуальность приобретают математические методы оценки влияния сбоев на работу компьютерных систем. Одним из подходов к рассматриваемым проблемам является вероятностный.

Проблема анализа качества аппаратного и программного обеспечения (ПО) становится сегодня все более острой, особенно по мере расширения использования в информационных банковских технологиях при разработке ПО.

Экспоненциальный рост сложности аппаратного и программного обеспечения вычислительных процессов порождает повышенные требования к бездефектному проектированию. Известны примеры, как дорого обходятся ошибки, допущенные на различных этапах проектирования, поэтому все современные ИС обязательно снабжаются методологическими, программными и инструментальными средствами анализа разрабатываемого изделия на всех этапах автоматизированного проектирования. Не менее актуальными являются проблемы, связанные с обеспечением проектирования надежного ПО. Большой вклад в становление и развитие методов решения данной проблемы внесли отечественные ученые Пархоменко П.П., Липаев В.В., Согомонян Е.С., Майоров С.А., Немолочнов О.Ф., Рябов Г.Г., Селютин В.А., Курейчик В.М. и многие другие.

В настоящее время сложилось несколько подходов к проверке правильности программного обеспечения (ПО): тестирование, теоретико-доказательный подход и верификация моделей программ.

Наиболее распространённым методом является тестирование. При тестировании на вход программе подаются заранее подготовленные тестовые данные и проверяется соответствие результата, выданного программой, ожидаемому. Важным свойством, используемом при тестировании, является свойство детерминированности алгоритма. В случае распределённого ПО, использующего несколько процессоров и выполняющегося на нескольких ЭВМ, поведение ПО становится существенно неоднозначным. В этом случае для проверки поведения ПО в ходе вычисления могут использоваться средства трассировки.

Методы верификации направлены на доказательство утверждений о свойствах поведения ПО или результатов её работы. Этот подход к проверке правильности ПО был впервые предложен в работах А.А. Ляпунова, Т. Хоара, Э. Дейкстры, Пратта и Флойда.

Методы верификации ПО, использующие теоретико-доказательный подход, основываются на средствах автоматического доказательства теорем (САДТ, theorem prover, «прувер»). Описание ПО представляется в виде множества логических утверждений. Спецификация ПО также представляется в виде утверждения. САДТ

строит доказательство выполнимости спецификации, исходя из утверждений, описывающих проверяемое ПО.

При использовании САДТ в процессе построения доказательства может потребоваться участие эксперта. Одной из наиболее трудных задач, возникающих при верификации ПО с помощью САДТ, является задача порождения инварианта цикла. Инвариант цикла требуется при проверке выводимости постусловия функции из предусловия.

Верификация ПО с использованием логических утверждений, построенных непосредственно на основании исходных кодов ПО, увеличивает сложность процесса доказательства. Некоторые фрагменты кода могут быть несущественными для проверки выделенных свойств ПО. Свойства ПО могут быть проверены на модели ПО, более простой, чем исходное ПО. Группа методов, основанных на верификации моделей (Model Checking, MC), использует компромиссный подход между полноценной верификацией ПО и тестированием, использующим формальную проверку свойств.

В области ЭВМ исследуются классы ИС, параметризованных по числу взаимодействующих компонентов:

- распределённые алгоритмы: волновые алгоритмы, распределения ресурсов, взаимного исключения доступа к критической секции, избрания лидера (leader election), обнаружения завершения (termination detection), согласованного принятия транзакций (distributed commit);
- сетевые протоколы: кольцо с маркером, протоколы маршрутизации, протоколы обеспечения качества сервиса, широковещательные протоколы;
- аппаратные схемы: аппаратные схемы управления доступом к шине при различном числе клиентских устройств, протоколы обеспечения когерентности кэшей.

При построении модели распределённой ИС, которая может потенциально содержать любое количество однотипных процессов, ограничиваются моделью с небольшим, фиксированным числом процессов. Предполагается, что свойства, проверенные на модели с фиксированным числом процессов, автоматически масштабируются для моделей с большим числом процессов. Однако этот приём не является вполне корректным: известны примеры, когда спецификация была верна на модели с одним числом процессов и нарушалась при изменении числа процессов.

Это означает, что для верификации параметризованных моделей ПО недостаточно проверить спецификацию на нескольких моделях с фиксированным числом процессов: необходимо обоснование масштабируемости проверенной спецификации на остальные модели. Так как число процессов в моделях не ограничено, то спецификация должна быть проверена на бесконечном семействе моделей с конечным числом состояний.

Наиболее перспективными методами, настраиваемыми на различные виды топологии параметризованных систем, являются методы, основанные на поиске инвариантов, и символьные методы.

Символьные методы верификации параметризованных систем могут потребовать участия эксперта для переработки модели и указания дополнительных подсказок.

Помимо верификации моделей с произвольным числом однотипных процессов алгоритмы решения задачи РМС позволяют проводить редукцию моделей с большим числом однотипных процессов к моделям с малым числом процессов. Решение задачи РМС имеет практический смысл даже тогда, когда рассматриваются модели с фиксированным, но очень большим числом процессов. В этом случае задача, требующая недоступных на практике вычислительных ресурсов, может быть сведена к практически решаемой задаче.

Задача верификации параметризованных моделей распределённых ИС актуальна. Однако при разработке таких методов и средств необходимо сохранить основное преимущество метода МС – свойство автоматической верификации модели. Необходимо, чтобы для верификации параметризованной модели не требовалась существенная перестройка модели, использованной при верификации средствами МС, или участие эксперта.

В четвертой главе использовался формальный анализ для разработки и проверки платежных протоколов, но, несмотря на развитие семантики и выразительности, в литературе мало внимания уделялось аспектам автоматизации систем подтверждения платежей в банковской системе. Это исследование совершенствует платформу автоматизированного анализа для платежных протоколов.

Усовершенствована комплексная и автономная система подтверждения, которая позволяет в платежных системах повысить конфиденциальность, целостность, аутентификацию, актуальность, подтверждение и неотказуемость. Моделируются общие примитивы безопасности, такие как шифрование, дешифрование, цифровые подпи-

си, дайджесты сообщений, коды аутентификации сообщений и сертификаты X.509.

Представлен автоматизированный анализ реальных протоколов платежей по банковским картам, а также протоколов бесконтактных мобильных платежей. В некоторых протоколах обнаружены недостатки безопасности; рассматриваются их причины и последствия.

По тематике настоящей монографии имеется достаточно обширная литература, однако в них недостаточно изложены вопросы практического применения основных правил функционирования различных технологий безопасности информации в коммерческих банках. Это дает нам право предложить данную монографию в качестве ключевого систематизированного материала для более полного и достаточного изучения экономической безопасности в области технологий обработки информации коммерческим банком.

1. АНАЛИЗ АВТОМАТИЗИРОВАННЫХ МЕТОДОВ ПОСТРОЕНИЯ УПРАВЛЕНИЯ ДОСТУПОМ К ИНФОРМАЦИОННОМУ И ПРОГРАММНОМУ ОБЕСПЕЧЕНИЮ БАНКА

На сегодняшний день существует множество систем защиты информационного и программного обеспечения от внутренних угроз. Одной из наиболее популярных технологий является технология IPC (Information Protection and Control). Основные функции, выполняемые технологией IPC:

- предотвращение различных видов утечек информационных ресурсов;
- защита информационных и программных ресурсов от внутренних угроз;
- предупреждение промышленного шпионажа.

К информационным ресурсам банка относятся:

- данные банка о разработках, проектах;
- персональные данные сотрудников банка.

К данным банка о проектах можно отнести:

- технические задания;
- программные коды разработок;
- проектно-конструкторскую документацию;
- сопровождающую документацию;
- экономические характеристики проектов.

К персональным данным сотрудников банка относятся:

- номера паспортов;
- номера документов, заверяющих личность;
- номера страховок;
- номера кредитных и дебетовых банковских карт.

Разрабатываемая автоматизированная система разграничения доступа к информационному и программному обеспечению (далее АСРДкИПО) имеет сложную структуру, так как должна решать не только проблему утечки информационных ресурсов за пределы программного обеспечения банка. Основными объектами защиты, проектируемой АСРДкИПО являются:

- информационные ресурсы банка, потеря или несанкционированное изменение которых влечёт за собой ущерб, потерю репутации банка;

- процессы обработки информационных и программных ресурсов. Примером процессов обработки могут служить хранение, исполнение (запуск), передача, отображение, изменение, вычислительная обработка, утилизация;

- программное обеспечение банка;

- информационная архитектура, представляющая собой структурированную совокупность автоматизированных систем и технологий банка.

Информационная среда банка является распределенной структурой, объединяющей информационные подсистемы банка в единую интегрированную автоматизированную компьютерную систему банка, осуществляющую обработку, хранение и преобразование данных. К основным особенностям информационной среды банка относятся:

- хранение информационных ресурсов в единых хранилищах (северах, банках данных, «серверных фермах» и т.д.);

- многофункциональность и широкий круг применения программного и информационного обеспечения банка;

- возрастание ответственности принимаемых решений, возложенной на программное обеспечение банка;

- функциональное разнообразие выполняемых задач программного обеспечения банка;

- высокий показатель нагрузки внутреннего информационного трафика (особенно это касается файлов банковской документации, графических файлов);

- применение особых технологий обработки графических файлов;

- недопустимость возникновения перебоев работы ПО банка, особенно при применении систем реального времени;

- многообразие пользователей ПО банка – сотрудников банка;

- объединение большого количества информационного, программного и аппаратного обеспечения в единую интегрированную информационную систему.

С учётом перечисленного ужесточаются требования к соблюдению норм и правил работы с информационным и программным обеспечением коммерческого банка. К субъектам защиты АСРДкИПО относятся:

- юридические и физические лица, задействованные в обеспечении выполнения банком своих функций (разработчики, консультан-

ты, сторонние организации, привлекаемые для оказания услуг, обслуживающий персонал и т.д.);

- сотрудники структурных подразделений банка;
- подразделения банка;
- руководство банка;
- банк в целом;
- представители третьей стороны.

Перечисленные субъекты заинтересованы в обеспечении:

- целостности информационных ресурсов ПО банка;
- полезности информационных ресурсов ПО банка;
- достоверности информационных ресурсов ПО банка;
- конфиденциальности информационных ресурсов ПО банка;
- доступности информационных ресурсов ПО банка;
- защиты информационных ресурсов ПО банка от несанкционированного копирования, распространения, тиражирования;

- защиты программных ресурсов банка от некомпетентного и некорректного использования (как специально, так и по неосторожности);

- разграничения ответственности за нарушения и отклонения от правил работы с информационными и программными ресурсами банка;

- возможности мониторинга работы с информационными и программными ресурсами банка, определения причин инцидентов нарушения их защиты.

Основной целью проектируемой АСРДкИПО является защита перечисленных объектов от возможного нанесения им физического, материального, морального или иного ущерба. Указанная цель должна достигаться посредством обеспечения и постоянного поддержания следующих свойств информационных ресурсов банка:

- конфиденциальность – сохранение в секрете информационных ресурсов, находящихся внутри ПО банка;

- целостность – сохранение свойств и полноты информационных ресурсов, находящихся внутри ПО банка;

- доступность информационных ресурсов для легальных пользователей – нахождение ПО банка в таком состоянии, при котором санкционированные пользователи (сотрудники банка) могли обращаться и работать с запрашиваемыми информационным и программными ресурсами банка.

К качеству информационных ресурсов, предоставляемых сотрудникам банка, предъявляются определенные требования, прежде всего требования о том, чтобы они были полезными. Для того чтобы информационные ресурсы банка обладали перечисленными свойствами, необходима разработка методики защиты информационных и программных ресурсов банка. Методика защиты информационных и программных ресурсов банка должна включать в себя следующие аспекты:

- применение принципа предоставления сотруднику минимального количества разрешенных программ, специального программного обеспечения для работы с журналами безопасности (для сотрудников отдела информационной безопасности банка), оперативного контроля состояния автоматизированных рабочих мест сотрудников;
- обеспечение комплексного использования средств защиты информационных и программных ресурсов банка;
 - централизованное управление потоками информации в банке;
 - централизованное управление учётными записями сотрудников банка, зарегистрированных в ПО банка, их правами доступа к информационным массивам;
 - комплексное использование средств защиты от утечки информационных ресурсов по техническим каналам.

Для обеспечения доходности, ликвидности, конкурентоспособности, сохранения репутации банка информационные ресурсы ПО банка должны обладать всеми перечисленными свойствами. В соответствии с изложенными аспектами, в АСРДкИПО предполагается использовать средства защиты информационного и программного обеспечения банка:

- централизованное управление параметрами операционных систем, установленных на рабочих станциях сотрудников банка;
- централизованное управление учётными записями сотрудников банка – пользователей ПО банка;
- применение современных методов идентификации, аутентификации и авторизации сотрудников банка при работе с информационным и программным обеспечением банка;
- применение методов шифрования;
- использование только зарегистрированных, прошедших контроль на наличие закладок и вирусов, устройств и внешних носителей;

- использование лицензированного, прошедшего специальный контроль программного и аппаратного обеспечения;
- использование только сертифицированных средств защиты информации;
- применение методики односторонней передачи информации от удаленных подсистем, между контурами различных грифов секретности, а также с автоматизированных рабочих мест ввода данных, при обращении к базе данных инцидентов нарушения защиты информационного и программного обеспечения банка;
- защита от несанкционированного физического доступа к серверам, хранилищам информации, «серверным фермам»;
- проведение специальных проверок технических средств банка;
- проведение специальных проверок аппаратного обеспечения банка;
- проведение специальных проверок программного обеспечения банка;
- проведение специальных исследований технических средств банка;
- проведение специальных исследований помещений;
- проведение специальных исследований программного обеспечения банка;
- проведение специальных исследований аппаратного обеспечения банка;
- применение механизма сигнализации – события несанкционированного доступа на автоматизированных рабочих местах сотрудников – отслеживание и оповещение сотрудников отдела информационной безопасности банка. Также должна существовать возможность информирования не только сотрудников отдела безопасности банка, но и непосредственно руководства. Важно отметить, что события несанкционированного доступа также заносятся в базу данных инцидентов нарушения правил работы с информационным и программным обеспечением банка;
- применение методики «замкнутой программной среды» – для каждого сотрудника банка – пользователя ПО банка – формируется определенный перечень программ, разрешенных только ему для запуска. Перечень может быть задан как индивидуально для каждого пользователя, так и определен на уровне групп пользователей. Применение данного режима необходимо для исключения попыток заражения вирусами, «червями», для предотвращения промышленного

шпионажа, а также применения нелицензированного программного обеспечения.

Информация в коммерческом банке может существовать в различных формах:

- в бумажной форме;
- в электронном виде;
- устно.

В электронном виде информация может передаваться как в локально-вычислительной сети (ЛВС), так и в глобальной вычислительной сети (ГВС). Безотносительно формы выражения информационных ресурсов, средств их распространения или хранения они должны всегда быть защищены. В банке большая часть информационных ресурсов хранится в электронном виде. Это, в первую очередь, связано с повсеместным внедрением:

- электронного документооборота;
- интеграционных средств поддержки;
- единых сред обработки большого количества информации;
- технологии непрерывной информационной поддержки поставок и жизненного цикла изделий CALS;
- технологии планирования ресурсов банка ERP;
- технологии управления и сбора данных систем реального времени SCADA;
- технологии проектирования CAD/CAM/CAE;
- применение дополнительных вспомогательных программных модулей и комплексов.

При проектировании специального программного обеспечения (ПО), используемого сотрудниками банка при выполнении своих должностных обязанностей, должны учитываться требования методики защиты ПО банка. Примером такого специального ПО являются среди программирования:

- Oracle JDeveloper;
- Microsoft Visual Studio;
- Embarcadero RAD Studio;

и среди проектирования:

- AutoCAD;
- ProEngineer;
- T-Flex;

редакторы-верификаторы G-кода:

- NCManager;

- CIMCO Edit;
- ctech VERICUT;

текстовые редакторы:

- Microsoft Word;
- TEX.

Определение требований согласно методике обеспечения защиты информационного и программного обеспечения банка включает в себя:

- оценку рисков безопасности функционирования ПО банка. В рамках оценки рисков необходимо произведение подсчёта возможных угроз нарушения защиты ПО банка, планирование комплекса мероприятий по их предотвращению;
- приведение к соответствию требованиям нормативных документов ключевых параметров АСРДкИПО;
- формирование специального набора норм, законов, целей и требований, разработанных банком в отношении работы с информационными и программными ресурсами банка.

Решения о расходах на мероприятия по защите информационного и программного обеспечения банка должны приниматься, исходя и из возможного ущерба, нанесенного банку в результате нарушений параметров защиты систем. Оценки рисков могут проводиться как для ПО банка в целом, так и для отдельных составляющих модулей. Оценка риска – это систематический анализ следующих вероятностей:

- возникновения угрозы;
- возникновения инцидента нарушения правил работы с информационными и программными ресурсами банка – реализации угрозы;
- возникновение последствий инцидента.

Оценка рисков должна производиться с использованием принципов системности и комплексности: для банка должны быть выявлены возможные источники угроз, предоставлены подробные отчёты о проведенном анализе. В ходе анализа необходимо учитывать:

- технический прогресс;
- возможные изменения направления деятельности банка;
- возможные изменения приоритетов деятельности банка;
- появление новых видов уязвимостей ПО банка;
- появление новых видов угроз.

После анализа рисков необходима разработка комплекса мероприятий для обеспечения требуемого уровня защиты. При выборе

мероприятий следует придерживаться принципа, что стоимость их проведения не должна превышать ценности защищаемых ресурсов ПО банка. Ценность информационных ресурсов составляют их актуальность и полезность. Ценность программных ресурсов составляет их функциональность и полезность. Также следует принимать во внимание факторы, которые не могут быть представлены в денежном выражении, например, потерю репутации. Ключевыми мерами контроля с точки зрения законодательства Российской Федерации являются:

- обеспечение конфиденциальности персональных данных сотрудников банка;
- защита учётных данных банка;
- защита прав на интеллектуальную собственность.

Типами доступа, которые следует рассматривать согласно ГОСТ Р ИСО/МЭК 17799-2005, являются:

- физический – к помещениям, компьютерным комнатам, серверным;
- логический – к базам данных, информационным системам банка.

В соответствии с системой классификации информационных ресурсов банка, принятой в банке, необходимо проведение процедур их маркировки. Процедуры маркировки должны осуществляться для информационных ресурсов как в электронной, так и в бумажной форме. В рамках работы рассматривается маркировка информационных ресурсов в электронном виде. При проведении маркировки следует учитывать процессы обработки информационных ресурсов (хранение, копирование, утилизация, передача и т.д.). Маркирование информационных ресурсов также необходимо для последующего проведения процедуры авторизации сотрудников банка.

Технология Information Protection and Control (IPC) является технологией защиты информационных ресурсов от внутренних угроз. Решения класса IPC предназначены для защиты информационных ресурсов от внутренних угроз, предотвращения различных видов утечек информационных ресурсов, промышленного шпионажа и разведки. Задача технологии IPC – предотвращение передачи информационных ресурсов за пределы периметра основной информационной системы. Технология IPC соединяет в себе три основных принципа:

- мониторинг технических каналов связи с помощью технологии Data Loss Prevention (DLP);

- контроль доступа к сети, приложениям и данным;
- шифрование носителей информации.

В ходе работы систем ИБ широко используется детектирование информационного контента. Детектирование информационного контента может происходить с помощью следующих методов:

- метод ручного детектирования («Карантин»);
- метод «цифровых отпечатков»;
- метод сигнатур;
- метод, основанных на проставлении меток;
- метод лингвистического анализа;
- метод, основанный на «регулярных выражениях».

Метод ручного детектирования («Карантин») основан на применении ручной проверки. При обращении сотрудника к файлам, содержащим конфиденциальные данные, производится информирования сотрудников отдела информационной безопасности банка, которые принимают решение о дальнейших действиях: предоставлении сотруднику запрашиваемых данных или отказе в доступе.

Достоинством метода является его эффективность. Недостатком является тот факт, что данный метод практически не реализуем в крупных банках, т.к. требует привлечения больших человеческих ресурсов (для формирования штата сотрудников отдела безопасности). Метод «карантин» можно применять с другими методами для анализа данных выбранных «подозрительных» сотрудников. При использовании метода «цифровых отпечатков» (англ. «Digital Fingerprints») формируется база данных образцов (паттернов, шаблонов) конфиденциальных файлов. В начале работы создается файл-шаблон и передается DLP-системе, далее формируется отпечаток и передается в специальную базу данных так называемых «паттернов». Затем в правилах фильтрации контента файлов настраивается процентное соответствие шаблону из базы. Достоинством метода является тот факт, что система фильтрации информационного трафика, основанная на данном методе, способна работать с информацией любого формата. Так же достоинствами системы на основе технологии «цифровых отпечатков» являются:

- прозрачность работы алгоритма для сотрудников отдела информационной безопасности;
- высокая степень детектирования инцидентов нарушения правил работы с информационным обеспечением;
- простота добавления новых файлов-шаблонов (паттернов).

Недостатками метода являются:

- чувствительность к изменениям файлов-шаблонов;
- требование обеспечения дополнительной защиты базы данных паттернов (особое расположение серверов, формирование правил и настройка допуска сотрудников к базе);
- снижение эффективности детектирования при переполнении базы данных паттернов, что на практике происходит довольно часто;
- высокая степень влияния на производительность основной информационной системы банка;
- низкая эффективность метода при работе с графическими файлами.

В крупных банках данную технологию трудно реализовать, так как база данных файлов-шаблонов увеличивается слишком быстро, что повышает нагрузку на сервера DLP-системы, кроме того постоянно приходится сталкиваться с проблемой заполнения базы данных, распределения нагрузки. Метод сигнатур представляет собой поиск в информационном потоке определенных символьных последовательностей, так называемых «стоп-выражений» запрещенных последовательностей символов. Метод относится к «детерминистским», т.к. алгоритм метода настроен на поиск 100 % совпадения «стоп-выражения». Большинство сигнатурных систем предназначены для поиска нескольких слов и частоту их встречаемости. Достоинства метода сигнатур:

- очевидность принципа работы;
- простота реализации и настройки.

Недостатками метода являются:

- чувствительность к специальной замене некоторых символов (например, на похожие символы из другого алфавита);
- неэффективность метода при работе с файлами программного кода;
- неприменимость метода к графическим файлам;
- зависимость функционирования от языка (как естественного, так искусственного).

DLP-системы, основанные на методе сигнатур хорошо «приспособлены» для западного рынка, но практически не пригодны в России. Причина – русский язык несигнатурен: в нём много приставок, суффиксов и окончаний. Метод «меток» заключается в расстановке специальных «меток» внутри файлов конфиденциальной информации. Достоинства метода:

- высокая эффективность детектирования;
- точность предоставляемой информации;
- высокое быстродействие.

Недостатки метода:

- сложность реализации - данный метод требует изменение структуры всей основной информационной системы банка;
- требование выполнения дополнительных настроек при формировании нового файла конфиденциальной информации.

Лингвистические методы являются самыми распространенными в применении в DLP-системах, т.к. предоставляют гибкий аппарат детектирования информационного контента. Лингвистический анализ текста включает в себя несколько методов детектирования:

- синтаксический анализ;
- семантический анализ;
- морфологический анализ и т.д.

Применение перечисленных методов анализа увеличивает эффективность DLP-системы. Недостатки лингвистических методов:

- чувствительность к переформированию предложений;
- неэффективность метода к файлам программного кода;
- неприменимость метода для графических файлов;
- зависимость от языка информационного контента.

Метод «регулярных выражений» (метод «текстовых идентификаторов») применяется в DLP-системах недавно. Регулярные выражения позволяют находить совпадения по типу данных (их форме представления). Основное отличие от метода «сигнатур» является особенность поиска: не по значению, а по типу данных. Таким образом, производится поиск целого блока идентичной информации. Подобный метод эффективен для поиска:

- обозначений изделий;
- дат;
- адресов (MAC, IP, интернет-сервисов);
- номеров портов;
- номеров кредитных карт;
- номеров счетов;
- номеров паспортов;
- классификаторов и т.д.

Достоинства метода:

- высокая эффективность;
- высокий уровень быстродействия;

- способность детектировать специфичный для каждого банка тип контента;

- применимость метода для работы с графическими файлами.

Часто метод «регулярных выражений» применяется в качестве дополнительного алгоритма в DLP-системе, но не в качестве основного. Следует отметить, что вышеперечисленные методы, применяемые в современных системах DLP, работают с открытыми данными. Если злоумышленник в лице сотрудника банка, превышающего должностные обязанности и нарушающего правила работы с информационными ресурсами банка, попытается передать конфиденциальную информацию в зашифрованном виде, современные DLP-системы не смогут предотвратить утечку. В проектируемой АСРДкИПО предусмотрено решение данной проблемы путём добавления специального модуля, в ядро которого включены криptoаналитические меры.

Следует уделить особое внимание модулю мониторинга информационного трафика банка (как внутреннего, так и внешнего) разрабатываемой АСРДкИПО, а именно, работе с зашифрованными данными. Также следует отметить, что большую часть информационного контента банка составляют графические файлы. Большинство методов детектирования не предназначены для работы с такими файлами. Злоумышленник может воспользоваться данным недостатком и передать конфиденциальные информационные ресурсы, «зашив» их в графические файлы с помощью методов стеганографии, кроме того, сами графические файлы представляют собой конфиденциальные информационные ресурсы.

Важно предусмотреть мероприятия по управлению защитой информационного и программного обеспечения банка, необходимые для управления доступом к средствам обработки информационных ресурсов третьей стороной. Все требования безопасности, связанные с доступом третьей стороны или мероприятиями по управлению защитой ПО банка, следует отражать в контракте с третьей стороной. Например, если существует специальная потребность в обеспечении конфиденциальности информационных ресурсов, следует заключить соглашение о их неразглашении. Недопустимо предоставлять третьей стороне доступ к информационным ресурсам и средствам их обработки до тех пор, пока не установлены соответствующие мероприятия по управлению защитой и не подписан контракт, определяющий условия предоставления информационных и программных ресурсов

банка. Распределение уровней доступа, предоставляемых сотрудникам банка, лицам третьей стороны, представлен на рис. 1. Согласно рис. 1, субъектам информационных отношений может быть присвоен уровень доступа к объектам информационных отношений. Управление присвоением соответствующего уровня доступа обеспечивается разрабатываемой АСРДкИПО.



Рис. 1. Распределение уровней доступа для сотрудников банка и представителей «третьей стороны»

Для обеспечения должного уровня защиты информационного и программного обеспечения банка необходимо проведение надлежащего контроля запросов сотрудников банка к соответствующим ресурсам. Одной из важных задач АСРДкИПО является контроль запросов сотрудников банка к информационным и программным ресурсам банка. Так как запросы пользователей могут быть адресованы как в Интернет, так и в сеть серверов данных банка с одного и того же автоматизированного рабочего места (АРМ), может произойти утечка информационных ресурсов. С целью предотвращения утечки

секретных данных в разрабатываемой автоматизированной системе должен быть модуль контроля запросов сотрудников банка к информационным ресурсам (МКЗСПкИР). МКЗСПкИР должен производить мониторинг запросов пользователей к информационным ресурсам банка и отправлять соответствующие отчёты в модули аудита, сотрудникам отдела информационной безопасности банка, руководству в случае необходимости. МКЗСПкИР в качестве основных механизмов должен использовать, в первую очередь, методы детектирования информационного контента. Доступ к информационным и программным ресурсам в банке должен быть регламентирован. По установленному регламенту все информационные массивы банка разделяются на группы (грифы) секретности с помощью специальных меток. Каждой такой группе соответствует список лиц, допущенных к работе с ней.

Цель системы класса DLP – предотвращение утечки стратегически важных информационных ресурсов банка за пределы основной информационной системы – определяет ее архитектуру. Архитектура DLP-системы представлена на рис. 2.

DLP –система включает в себя подсистему контроля и мониторинга информационного контента и подсистему реагирования на инциденты нарушения параметров защиты ПО банка. Подсистема контроля и мониторинга информационного контента включает в себя контроль доступа к данным, к сети, к приложениям, шифрование информации, детектирование информационного контента.

Подсистема реагирования на инциденты включает настройку правил и условий срабатывания сигнала-оповещения (alarm) и самого модуля оповещение сотрудников службы безопасности и руководства банка.

Настройка подсистемы мониторинга информационного контента осуществляется сотрудниками службы информационной безопасности банка. Настройка подсистемы реагирования на инциденты производится сотрудниками службы безопасности банка с непосредственным контролем руководящего состава банка. Важно отметить, что в условиях совершенствования методов осуществления угроз снижения уровня защиты основной информационной системы, в качестве которой в работе рассматривается ПО банка, необходимо учитывать:

- возможность совершенствования DLP-системы;
- возможность модификации основных составляющих;
- возможность добавление компонентов.



Рис. 2. Архитектура DLP-системы

Инфраструктура DLP-системы представлена на рис. 3.

Условные обозначения:

- АРМ П – автоматизированное рабочее место пользователя (сотрудника банка);
- АРМ А – автоматизированное рабочее место (АРМ) администратора (сотрудника отдела безопасности банка);
- АРМ Р – АРМ руководящего состава банка;
- синяя стрелка – запрос к данным;
- красная стрелка – осуществление контроля;
- зелёная стрелка – оповещение;
- зелёная пунктирная стрелка – «скрытое» (прозрачное для всех сотрудников банка) оповещение. Передача информации происходит от одного модуля (программного или аппаратного обеспечения) к другому автоматически (без участия сотрудников банка).

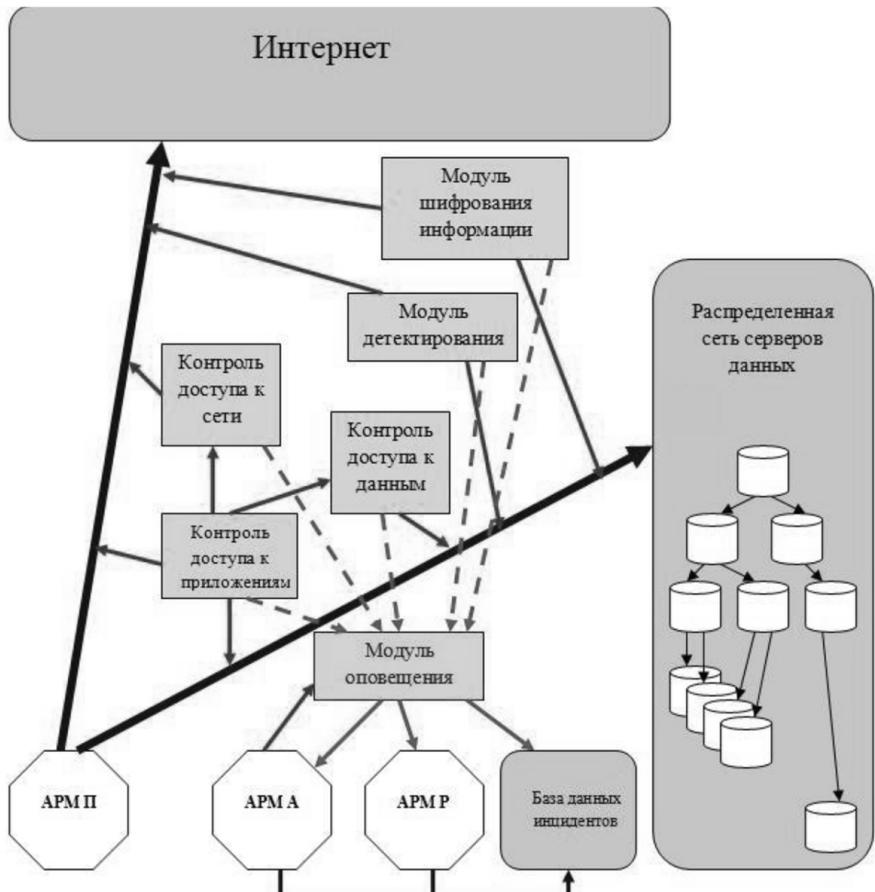


Рис. 3. Инфраструктура DLP-системы

Согласно рис. 3, пользователь (сотрудник банка) может обращаться к информационным ресурсам банка, физически расположенной во внутренней распределённой сети банка, а также обращаться за необходимой информацией в Интернет. Кроме того, сотрудник может обращаться к приложениям и сервисам (программному обеспечению) банка и ГВС. Таким образом, формируются два основных информационных потока запросов, над которыми необходимо осуществлять контроль со стороны АСРДкИПО.

На рис. 4 представлены взаимодействия основных структур DLP-системы. В состав DLP-системы входят три основных компонента:

- рабочие станции сотрудников банка (автоматизированные места сотрудников банка, администраторов и руководящего состава банка);

- контролирующие структуры (модули);

- данные (Интернет, файлы локальной сети банка, база данных инцидентов нарушений правил работы с информационным и программным обеспечением банка). Сотрудники банка производят обращение к данным (в Интернет, в локальной сети банка, к базе данных инцидентов безопасности банка).

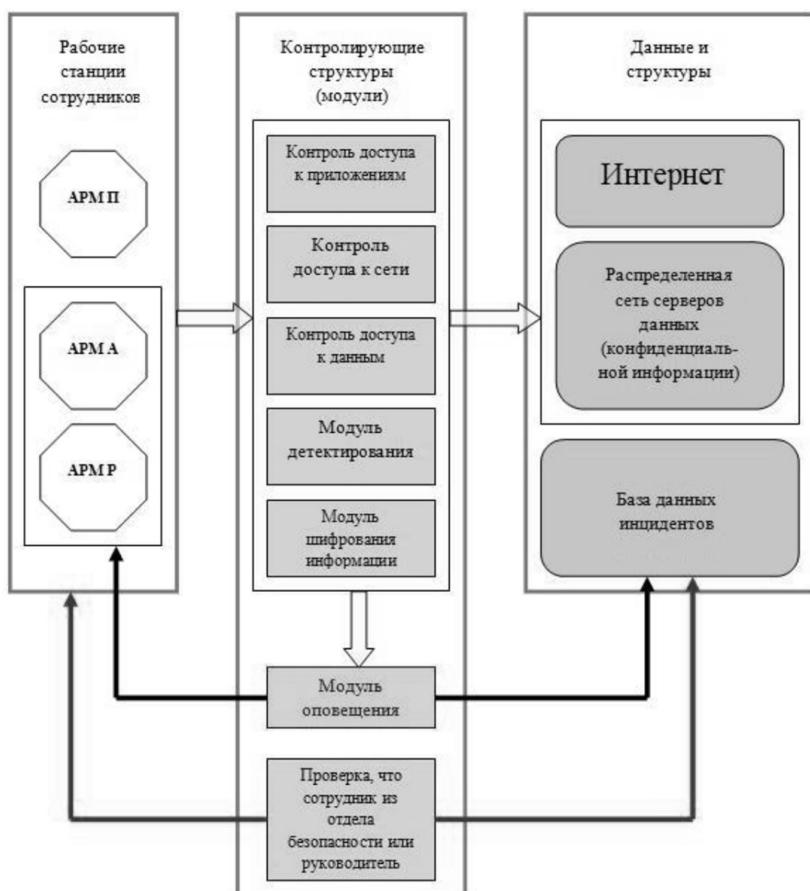


Рис. 4. Взаимодействие основных структур DLP-системы

Условные обозначения к рис. 4:

- фигурная стрелка – запрос данных, обращение к модулю оповещения;

- чёрная стрелка – действия модуля оповещения (информирование сотрудников отдела безопасности и руководства, запись в базу данных инцидентов нарушения правил работы с информационным и программным обеспечением банка);

- красная стрелка – обращение сотрудников отдела информационной безопасности и руководства к базе данных инцидентов банка и получение ответа. Затем сотрудник получает доступ к запрашиваемым информационным ресурсам.

В табл. 1 приведена информация о DLP-системах от основных поставщиков IPC-продуктов на российском рынке:

- Symantec Data Loss Prevention;
- InfoWatch Traffic Monitor;
- McAfee Host Data Loss Prevention;
- Websense Data Security Suite.

Изложенные в предыдущем пункте характеристики информационных систем класса DLP можно привести к восьми основным (рис. 5):

- гибкость;
- способность к адаптации;
- способность к интеграции;
- производительность детектирования ошибок I рода;
- производительность детектирования ошибок II рода;
- масштабируемость;
- скорость анализа сетевого трафика;
- прозрачность работы.

Из рис. 5 видно, что лучшей системой класса DLP по перечисленным критериям является Symantec DLP. Существует только один критерий, по которому она уступает отечественной разработке InfoWatch Traffic Monitor – способность к интеграции. Это, прежде всего, связано с особенностями предпочтения использования программных продуктов почты и т.д.) на российском рынке.

Таблица 1

Сравнение основных DLP-систем

| Критерий | Поставщики DLP-систем | | | |
|--|-------------------------------------|-------------------------------------|--|-------------------------------------|
| | <i>Simantec DLP</i> | <i>InfoWatch Traffic Monitor</i> | <i>McAfee Host Data Loss Prevention</i> | <i>Websense DSS</i> |
| 1 Наличие модуля сетевого уровня | Да | Да | Да | Да |
| 2 Наличие модуля уровня хоста | Да | Да | Да | Да |
| 3 Метод классификации защищаемых информационных ресурсов | Метод «сигнатур» | Лингвистический анализ | Метод «сигнатур», метод «регулярных выражений» | Не указан |
| 4 Метод анализа содержимого документа | Метод «сигнатур» | Лингвистический анализ | Метод «сигнатур», метод «регулярных выражений» | Не указан |
| 5 Временная характеристика метода анализа информационного контента (как система реального времени) | Система «мягкого» реального времени | Система «мягкого» реального времени | Система «мягкого» реального времени | Система «мягкого» реального времени |
| 6 Просмотр произошедших передач данных (наличие модуля контроля утечек контента в рамках ретроспективного анализа) | Да | Да | Да | Да |

| | | | | | |
|----|--|---|---|---|---|
| 7 | Гибкость DLP-системы, способность к адаптации и интеграции | Высокая степень гибкости, способности к интеграции и «апгрейду» | Средняя степень гибкости, способности к интеграции и «апгрейду» | Высокая степень гибкости, способности к интеграции и «апгрейду» | Высокая степень гибкости, способности к интеграции и «апгрейду» |
| 8 | Наличие модуля подотчётности | Да | Да | Да | Да |
| 9 | Наличие модуля поиска по заданным критериям о расположении, времени модификации и перемещении информационных ресурсов, а также их инвентаризации и категоризации | Да | Да | Да | Да |
| 10 | Наличие модуля контроля использования данных в пределах предприятия (автоматизированный контроль перемещения информационных ресурсов) | Да | Нет | Да | Да |
| 11 | Наличие модуля задания правил использования информационных ресурсов сотрудниками предприятия | Да | Да | Да | Да |

| | | | | | |
|----|--|---|--|--|-----|
| 12 | Наличие центрального управления политикой использования данных в электронной форме | Да | Нет Модули настраиваются отдельно | Да | Да |
| 13 | Наличие модуля получения данных о корреляции работы системы контроля защиты и бизнес-процессов. (Система безопасности сама является надстройкой к основной информационной системе предприятия и может влиять на её функционирование, особенно, на производительность, отчётов о работе системы | Нет | Нет | Нет | Нет |
| 14 | Поддержка режима блокирования трафика | Да | Да | Да | Да |
| 15 | Поддержка режима мониторинга трафика | Да | Да | Да | Да |
| 16 | Сохранение всей переписки сотрудников | Да (официально: в зависимости от принятой методики защиты) | Да Возможна настройка прав просмотра сохраненной переписки офицерами безопасности | Да (официально: в зависимости от принятой руководством методики защиты) | Да |

| | | | | | |
|----|--|----|-----|----|----|
| 17 | Контроль за манипуляцией информационного контента (корпоративная почта, ftp- соединения, передача мгновенных сообщений, передача документов на принтер, использование внешних устройств памяти: USB, CD, DVD, мобильные устройства, локальные соединения Bluetooth, WiFi и т.д.) | Да | Да | Да | Да |
| 18 | Наличие автоматизированного аудита мест расположения информационных ресурсов | Да | Да | Да | Да |
| 19 | Наличие модуля отслеживания общего уровня риска (на основе отчёта по инцидентам) | Да | Нет | Да | Да |
| 20 | Наличие модуля контроля утечек информационных ресурсов в режиме немедленного реагирования (как система реального времени) | Да | Да | Да | Да |

| | | | | | |
|----|--|---------|---------|---------|---------|
| 21 | Соответствие требованиям стандарта PCI DSS | Да | Да | Да | Да |
| 22 | Производительность системы в рамках критерия ошибок первого и второго рода (ошибка первого рода – когда система допускает несанкционированный субъект к запрашиваемому объекту, ошибка второго рода – когда система не допускает санкционированный субъект к запрашиваемому объекту) | Высокая | Высокая | Средняя | Высокая |
| 23 | Наличие модуля (в виде клиентской части) контроля обработки информации на рабочем месте сотрудника | Да | Да | Да | Да |
| 24 | Наличие модуля сканирования сети предприятия на предмет неупорядоченного хранения сведений конфиденциального характера | Да | Нет | Да | Да |

| | | | | | |
|----|--|---|--|---|---|
| 25 | Модуль сравнения документа с первоисточником | Да | Да | Да | Да |
| 26 | Анализ совершенно новой информации (новые программные комплексы). Дополнительный контроль за разработчиками, программистами и т.д. | Нет | Нет | Нет | Нет |
| 27 | Функционирование системы в виде агентской программы на рабочих станциях сотрудников и серверах управления | Да | Да | Да | Да |
| 28 | Расположение вычислительной части системы контроля защиты | Отдельно от основной АСУТП, АСУП, АСТПП предприятия | Отдельно от основной АСУТП, АСУП, АСТПП предприятия | Отдельно от основной АСУТП, АСУП, АСТПП предприятия | Отдельно от основной АСУТП, АСУП, АСТПП предприятия |
| 29 | Способность системы к обучению | Нет | Да (если рассматривать накопление баз контентной фильтрации) | Нет | Нет |
| 30 | Период обучения системы | – | В зависимости от решения руководства: в течение функционирования системы | – | – |

| | | | | | |
|----|--|------|------|------|------|
| 31 | Наличие модуля защиты документов на уровне фрагментов | Да | Да | Да | Да |
| 32 | Наличие модуля контроля файлов формата pdf и других графических форматов | Да | Да | Да | Да |
| 33 | Наличие функции контроля операции PrintScreen для определённых приложений | Нет | Нет | Да | Нет |
| 34 | Наличие модуля контроля запуска приложений на рабочих станциях сотрудников | Да | Нет | Да | Нет |
| 35 | Требование наличия сотрудника, сопровождающего систему управления защитой | Да | Да | Да | Да |
| 36 | Соотношение «офицер безопасности» - количество сотрудников | 1:50 | 1:20 | 1:50 | 1:50 |
| 37 | Возможность построения иерархии серверов системы контроля защиты | Нет | Нет | Нет | Нет |

| | | | | | |
|----|---|-------------------------------|--|--|-------------------|
| 38 | Прозрачность работы системы контроля защиты | Высокий уровень | Высокий уровень | Высокий уровень | Высокий уровень |
| 39 | Форма определения степени важности информационного контента | Использование метода шаблонов | Лингвистический анализ | Метод «сигнатур», метод «регулярных выражений» | Не указана |
| 40 | Основной математический метод | Информация скрыта | методы иерархического списка категорий | Информация скрыта | Информация скрыта |

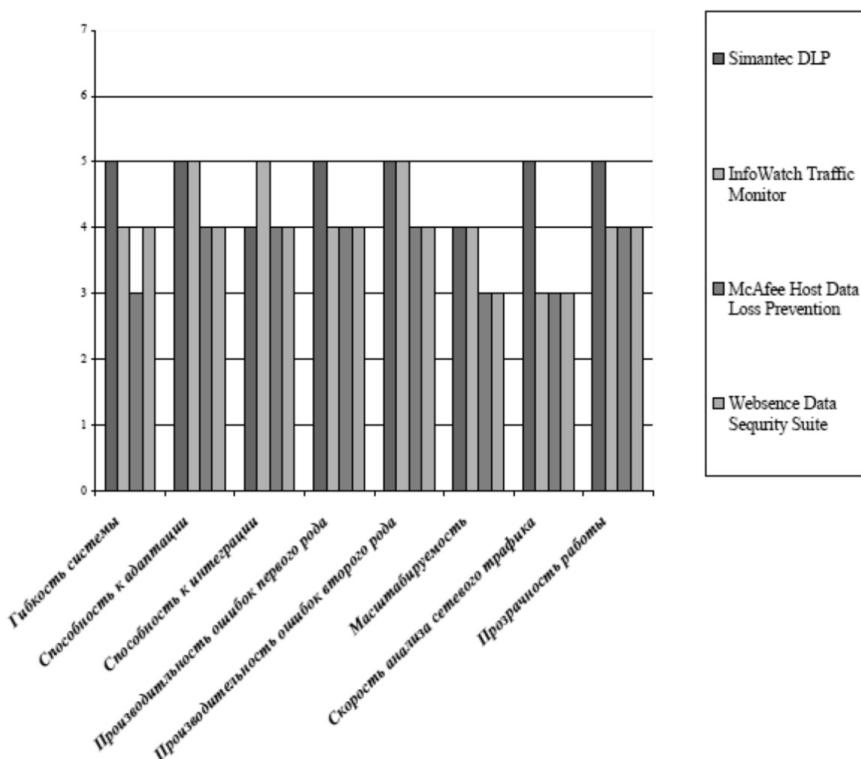


Рис. 5. Сравнительная диаграмма основных характеристик систем класса DLP

Надежность и точность идентификации стратегически важных информационных ресурсов в информационных потоках банка с по-

мощью технологии лингвистического анализа зависят от принципа формирования БКФ, на основе которой осуществляется анализ. Поэтому важно создать базу, которая обеспечит надежные результаты фильтрации информационного контента по категориям. Основным методом лингвистического анализа с помощью БКФ является поиск в анализируемом фрагменте информации слов и словосочетаний, описывающих стратегически важные данные и структурированных по категориям групп слов.

1.1. Управления доступом к информационному и программному обеспечению в коммерческом банке

Управление доступом (access control) – это процесс проверки запросов на доступ к сервису с целью определить разрешить или запретить доступ. Большая часть современных систем использует модель управления доступом, предложенную Лампсоном в 1974 г (рис. 6).

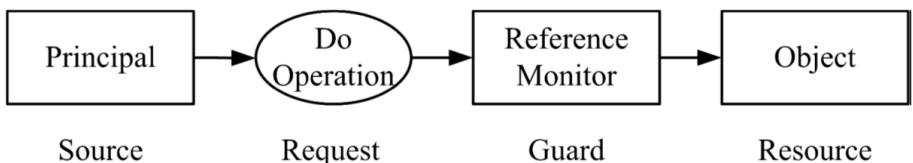


Рис. 6. Модель управления доступом (Лампсон, 1974)

Модель Лампсона включает следующие элементы:

- Principal – автор запроса (принципал, иногда называют субъектом).
- Object – ресурс, информационный, сетевой или вычислительный.
- Request – запрос на выполнение операции с объектом.
- Reference Monitor – диспетчер доступа, проверяющий все запросы к объекту и принимающий решение о разрешении или запрещении доступа.

Диспетчер доступа для принятия решения проводит три процедуры защиты информации. Определение источника запроса называется идентификацией, подтверждение подлинности источника назы-

вается аутентификацией, анализ правил разграничения доступа называется авторизацией.

Наиболее полную интероперабельность интегрируемых систем обеспечивает семантическая интеграция. Интеграционные решения, использующие синтаксический и структурный подход, являются частными, рассчитанными на определенные системы. Широкое применение таких решений затруднено.

В настоящее время наиболее популярной технологией семантической интеграции являются онтологии. Онтологии определяют общий словарь предметной области, который может совместно использоваться людьми или информационными системами. Для разработки онтологий существует широкий набор инструментальных средств: язык описания онтологий OWL (Web Ontology Language), являющийся стандартом W3C, среды редактирования онтологий Protégé, Ontolingua, Chimaera, готовые и доступные к использованию онтологии, описывающие различные предметные области.

Выявлены аналоги трех типов: системы централизованного управления, системы федеративной идентификации, системы интегрированного управления. Оценка наиболее близких аналогов приведена в табл. 2.

Таблица 2

Оценка существующих систем интеграции управления доступом

| Название | Интеропера- бель- ность | Функцио- нальные возможно- сти | Методы управле- ния до- ступом | Схема рас- простране- ния | Область приме- нения | Всего |
|---|-------------------------------|---|---|---------------------------------|-------------------------|-------|
| Системы централизованного управления | | | | | | |
| AAA | 1 | 6 | 1 | 10 | 5 | 23 |
| LDAP | 1 | 6 | 1 | 10 | 5 | 23 |
| KERBEROS | 1 | 6 | 1 | 10 | 5 | 23 |
| A-Select | 1 | 6 | 1 | 10 | 5 | 23 |
| SAML | 1 | 7 | 1 | 10 | 5 | 24 |
| Системы федеративной идентификации | | | | | | |
| Liberty | 4 | 7 | 7 | 8 | 5 | 31 |
| WS-Federation | 3 | 7 | 7 | 5 | 5 | 27 |
| OpenID | 2 | 5 | 5 | 8 | 3 | 23 |
| Windows CardSpace | 2 | 5 | 6 | 5 | 3 | 21 |

| Системы интегрированного управления | | | | | | |
|--|----------|-----------|-----------|-----------|-----------|-----------|
| IBM Tivoli Identity Manager | 5 | 10 | 10 | 5 | 10 | 40 |
| Sun Identity Manager | 5 | 10 | 10 | 10 | 10 | 45 |
| Oracle Identity Manager | 5 | 10 | 10 | 5 | 10 | 40 |
| Novell Identity Manager | 5 | 9 | 8 | 5 | 10 | 37 |
| Microsoft Identity Intergration Server | 5 | 9 | 8 | 5 | 10 | 37 |

Оценка наиболее близких онтологий с точки зрения возможности применения для интеграции управления доступом к сервисам разных типов приведена в табл. 3.

Таблица 3

Оценка онтологий в предметной области управления доступом

| Онтология | Широта применения | Ориентация на информационные системы | Практическое использование | Всего |
|---|-------------------|--------------------------------------|----------------------------|-----------|
| Онтологии для управления доступом к конкретным информационным системам | | | | |
| HL7 | 1 | 10 | 1 | 12 |
| LSDIS | 1 | 10 | 1 | 12 |
| Access-eGov | 1 | 10 | 1 | 12 |
| Онтологии для методов управления доступом к сервисам | | | | |
| Access Control Lists ontology | 3 | 10 | 3 | 16 |
| Role base access control ontology | 3 | 10 | 3 | 16 |
| Attribute base access control ontology | 3 | 10 | 2 | 15 |
| Rule base access control ontology | 3 | 10 | 2 | 15 |
| Context aware access control ontology | 2 | 10 | 1 | 13 |
| Онтологии верхнего уровня | | | | |
| Cys | 10 | 1 | 10 | 21 |
| SUMO | 10 | 2 | 9 | 21 |
| GOL | 8 | 2 | 6 | 16 |
| BWW | 6 | 10 | 8 | 24 |
| DOLCE | 7 | 8 | 8 | 23 |

Схема прототипа и существующее решение в «Газпромбанке» приведена на рис. 7.

Основным недостатком прототипа является низкая интероперабельность, вызванная использованием структурной интеграции. Для увеличения интероперабельности предлагается использовать семантическую интеграцию: разработать семантическую модель в виде онтологий, ввести блок согласования семантики в адаптеры сервисов,

модифицировать схему данных и блоки хранения правил разграничения доступа, ПО управления доступом и консоль управления доступом.

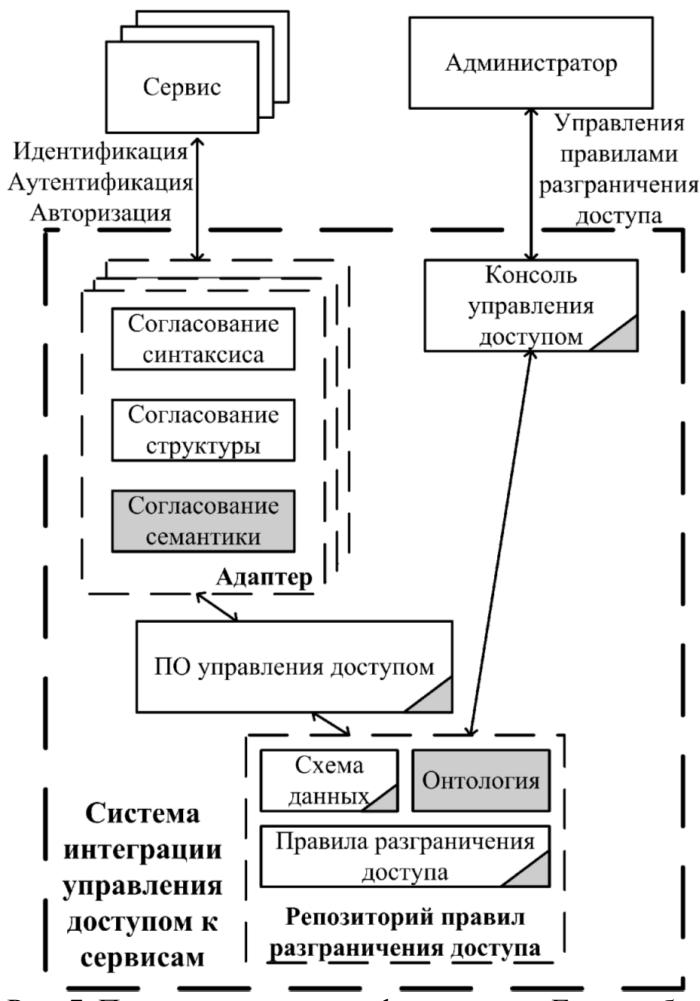


Рис. 7. Процесс доступа к информации в «Газпромбанке»

Используется методика интеграции управления доступом к сервисам разных типов на основе семантического подхода. Методика заключается в выполнении следующих шагов:

1. Задать множество сервисов S , которые подлежат интеграции.

2. Задать множество моделей M , используемых для управления доступом к сервисам из множества S .
 3. Задать множество репозиториев правил разграничения доступа R , которые подлежат интеграции.
 4. Задать множество протоколов управления доступом P , которые используются сервисами из множества S .
 5. Разработать унифицированную модель управления доступом M_0 . Модель включает семантическое описание базовых понятий и операций систем управления доступом (представленное в виде онтологии) и синтаксис записи правил разграничения доступа с использованием данной семантики.
 6. Определить отображение моделей управления доступом $m_i \in M$ в унифицированную модель M_0 : $F: m_i \rightarrow M_0 | m_i \in M$.
 7. Создать консолидированный репозиторий правил разграничения доступа R_0 , использующий унифицированную модель M_0 .
 8. Разработать адаптеры AR , обеспечивающие перенос данных из репозиториев $r_i \in R$ в консолидированный репозиторий R_0 .
 9. Создать программное обеспечение управления доступом, реализующее процедуры защиты информации с использованием правил разграничения доступа в консолидированном репозитории R_0 .
 10. Разработать адаптеры AS протоколов управления доступом $p_i \in P$, обеспечивающие взаимодействие сервисов $s_i \in S$ с ПО управления доступом, для выполнения процедур защиты информации с использованием протокола $p_i \in P$.
 11. Настроить сервисы $s_i \in S$ для взаимодействия с ПО управления доступом с помощью адаптеров $a_i \in AS$ по протоколам $p_i \in P$.
 12. Разработать консоль управления правилами разграничения доступа в консолидированном репозитории R_0 .
- Структурная схема системы семантической интеграции управления доступом к сервисам разных типов, получаемой с помощью предлагаемой методики, показана на рис. 8.
- Особенность предлагаемого метода интеграции управления доступом к сервисам разных типов заключается в применении семантического подхода к интеграции, обеспечивающее большую интероперабельность, по сравнению с используемыми в настоящее время синтаксическим и структурным подходами. Построенная с применением данного метода система обеспечит интеграцию более широкого круга сервисов и использование большего количества методов управления доступом.



Рис. 8. Структурная схема системы семантической интеграции управления доступом к сервисам

Логическая архитектура комплекса программ показана на рис. 9 и состоит из трех уровней:

- Уровень технических служб отвечает за взаимодействие с репозиториями правил разграничения доступа $r_i \in R$, обеспечивает создание консолидированного репозитория правил разграничения доступа R_0 .
- Уровень приложений реализует прикладную логику управления доступом: процедуры защиты информации, управление сессиями и т.д.
- Уровень представления обеспечивает взаимодействие комплекса с внешним миром: сервисами, администраторами, пользователями и внешними информационными системами.



Рис. 9. Архитектура комплекса программ по управлению доступом к сервисам

Консолидированный репозиторий правил разграничения доступа построен на основе LDAP-каталога Sun Java System Directory Server. АдAPTERЫ протоколов идентификации реализованы с использованием ПО FreeRADIUS, OpenSSO, Samba. Система управления построена на основе Sun Java System Delegated Administrator, функциональность которого расширена для возможности управления доступом к сервисам разных типов. Система управления предоставляет два типа интерфейса: Web и командную строку. Распределение полномочий между администраторами реализовано с помощью делегирования на двух уровнях: администратор с полным доступом и администратор организации. Комплекс работает под управлением ОС Solaris. Надежность работы комплекса обеспечивается с помощью структурного резервирования.

Удобство использования сервисов повышается за счет единого идентификатора и пароля. Количество необходимых для работы идентификаторов сократилось с 10 (соответствует количеству сервисов, подключенных к системе управления доступом) до одного. Со-

кращение количества идентификаторов также привело к уменьшению количества запросов в службу поддержки по восстановлению забытых паролей и консультаций по порядку доступа к сервисам на 60 %.

Проведен сравнительный анализ временных характеристик процесса управления доступом к сервисам с использование системы интегрированного управления и без нее. Результаты сравнительного анализа представлены в табл. 4.

Таблица 4

Результаты сравнительного анализа временных затрат в ходе управления доступом к сервисам

| Операция | С помощью традиционных систем управления доступом, мин. | С помощью системы интегрированного управления, мин. |
|--------------------------|---|---|
| Создание учетной записи | 15-30 | 3-5 |
| Изменение учетной записи | 5-10 | 2-3 |
| Удаление учетной записи | 15-30 | 3-5 |
| Назначение прав доступа | 10-15 | 4-10 |
| Изменение прав доступа | 5-10 | 1-2 |

Время, сэкономленное при управлении доступом с использованием системы интегрированного управления, может составлять до 80% от обычного времени работы с применением существующих систем управления доступом за счет сокращения количества ручных операций.

Интеграция управления доступом к сервисам разных типов позволяет снизить требования к администраторам за счет предоставления единой интуитивной понятной системы управления с Web-интерфейсом.

1.2. Анализ методов проверки корректности работы программного обеспечения

Если корректность модели можно гарантировать с помощью ограничений на методы построения модели в ПО, тогда адекватность может быть проверена только после того, как проведена верификация модели. Для построения корректной модели могут использоваться

методы автоматического порождения моделей (Model Extraction) по заданному алгоритму, а автоматическое построение адекватной модели достигается за счёт комбинирования методов булевой абстракции с итеративной верификацией моделей. Такой подход к построению моделей получил название Counter Example Guided Abstraction Refinement. В случае неудачной проверки построенной модели проводится уточнение модели с помощью САДТ. В данном случае комбинируются средства МС и САДТ. Гарантия полностью автоматической проверки моделей исчезает, т.к. не во всех случаях САДТ способны выдать доказательство запрашиваемого утверждения.

Эффект «комбинаторного взрыва» связан с экспоненциальным ростом пространства состояний при линейном росте числа взаимодействующих процессов. Последствиями экспоненциального роста пространства состояний становятся высокие требования к объему памяти, используемой верификатором моделей, и долгое время проверки модели. Для сжатия проверяемого множества состояний используются символьные способы представления моделей, например, двоичные разрешающие диаграммы (Binary Decision Diagrams, BDD). В символьном представлении модели выражаются с помощью BDD, которые реализуют функции алгебры логики. При поиске в пространстве состояний вместо перебора состояний используются операции над функциями.

Алгоритмы верификации моделей с помощью BDD могут быть значительно эффективнее явного перебора состояний в том случае, когда BDD системы переходов модели и промежуточных результатов остаются компактными. Для моделей аппаратных схем представление модели в виде BDD даёт хороший коэффициент сжатия по отношению к явному представлению множеств. Модели распределённого ПО сжимаются несколько хуже, т.к. в распределённом ПО используется асинхронное взаимодействие.

Для борьбы с «комбинаторным взрывом» при верификации моделей ПО используются методы редукции частичных порядков. В этих методах проверяется достаточное подмножество множества всех трасс. Такое подмножество вычисляется на основании зависимостей между переходами процессов модели. Типы проверяемых зависимостей выявляются эмпирически. Оптимальный выбор правил редукции может существенно ускорить алгоритм верификации модели.

Третьей проблемой применения группы методов МС является требование конечности числа состояний проверяемой модели. Алго-

ритмы МС гарантированно завершаются и эффективно работают именно на моделях с конечным числом состояний. При попытке обобщения задачи верификации методами МС с целью проверки систем с бесконечными множествами состояний возникает два направления: верификация моделей с бесконечным числом состояний и верификация параметризованных моделей.

При верификации моделей с бесконечным числом состояний (Infinite State-Space Model Checking) в описании модели могут присутствовать переменные с неограниченными типами данных, а порождаемая по описанию система переходов обладает бесконечным множеством состояний. Для решения этой задачи используются комбинации символьных методов верификации моделей.

Задача верификации параметризованных моделей (Parameterized Model Checking, PMC) возникает при рассмотрении моделей распределённых систем, в которых число однотипных взаимодействующих процессов зависит от начальной конфигурации системы и может быть сколь угодно большим. Увеличение числа вычислителей не всегда приводит к линейному росту производительности распределённой ИС. Аналогично, выводы о выполнимости спецификаций, полученные для конфигурации с одним числом процессов, нельзя переносить на конфигурации с другим числом процессов, не приводя дополнительной аргументации. Решение задачи PMC позволяет убедиться в выполнимости проверяемых спецификаций при масштабировании распределённого ПО, т.е. при росте числа взаимодействующих процессов.

При фиксированном числе процессов может быть построена модель распределённой ИС с конечным числом состояний. Однако число взаимодействующих процессов неограниченно, и множество моделей с различным числом процессов бесконечно. Проверка произвольных моделей с фиксированным числом процессов не даёт гарантии выполнимости проверяемого свойства при увеличении числа взаимодействующих процессов, а все модели проверить невозможно.

В задаче PMC рассматриваются распределённые ИС взаимодействующих процессов. В этих ИС присутствует фиксированное число статических процессов и неограниченное число однотипных процессов. Так как число однотипных процессов не ограничено, то по распределённой ИС может быть построено бесконечное семейство моделей. Каждая модель соответствует конфигурации ИС с фиксированным числом процессов. Для проверки свойств распределённой ИС

необходимо проверить спецификацию на всех моделях бесконечного семейства.

Методы решения задачи РМСР развивались в следующих направлениях:

- аналитические методы редукции;
- методы абстракции;
- символьные методы;
- методы, основанные на поиске инварианта.

В аналитических методах редукции указывается способ отображения моделей семейства параметризованных моделей на одну модель этого семейства (*cut-off model*). В этом случае точно известны ограничения, которым должны удовлетворять модели семейства.

Несомненными достоинствами аналитических методов редукции являются: точное описание области применимости методов, алгоритмизация и завершаемость. С помощью средств МС достаточно проверить все модели M_1, \dots, M_N до порогового значения N , где N известно заранее.

Среди недостатков подхода можно перечислить следующие. Во-первых, аналитические методы редукции применимы только к двум видам ИТС: асинхронным ИТС с кольцевой топологией, процессы которых обмениваются маркером, а также асинхронным широковещательным ИТС с переходами специального вида. Во-вторых, теоретические оценки порогового значения для некоторых методов зависят от числа состояний процессов, поэтому число состояний пороговой модели может быть настолько большим, что в результате проверка пороговой модели будет практически невозможной.

Для методов абстракции также указывается способ отображения параметризованных моделей на одну модель, но эта модель не принадлежит самому параметризованному семейству, а моделирует поведение любого числа однотипных процессов. Из способа отображения следуют структурные ограничения на параметризованные семейства моделей, которые могут быть верифицированы таким способом.

Основное достоинство методов абстракции заключается в том, что после нахождения вида абстрактной модели и отображения моделей семейства \mathcal{F} на абстрактную модель, проверка свойств осуществляется автоматически для любой параметризованной модели (за исключением метода Лесена и Сайди, где булева абстракция строится с помощью САДТ). Методы верификации параметризованных моделей с помощью абстракции опираются на постоянно развивающийся ап-

парат построения булевых абстракций и абстрактной интерпретации. Поэтому область применимости методов абстракции к задаче РМС постепенно расширяется.

Недостатки. Даже в тех случаях, когда экспертам удаётся указать вид абстрактной модели A и способ абстракции α для класса моделей, область применимости методов ограничивается: ИТС, процессы которых обмениваются маркером; широковещательными ИТС; ИТС с топологией «звезда». В остальных случаях методы абстракции требуют подсказок от эксперта. Например, требуется явное указание существенных переменных и предикатов в ИТС переходов. Когда абстракция построена, спецификация может не выполняться на абстрактной модели. Из чего не следует, что спецификация нарушается и на моделях параметризованного семейства \mathcal{F} . В таком случае требуется участие эксперта для уточнения абстракции, однако уточнение возможно не для всех указанных методов.

Символьные методы используют модификации символьного подхода к решению задачи МС. Для описания параметризованного семейства моделей используются системы переходов, в которых переходы задаются неявно, с помощью символьных описаний. Например, множества переходов могут представляться регулярными выражениями или логическими формулами с кванторами существования и всеобщности. Для заданной системы переходов ищется множество состояний, достижимых из начальных состояний. В случае символьных методов, как правило, нет гарантии завершаемости.

Достиныства. Символьные методы могут применяться для проверки свойств информационной безопасности (ИБ) параметризованных моделей асинхронных распределённых ИТС со следующими топологиями: кольцевая, древовидная, полно связная. Положительным свойством методов является использование известных в математической логике языков и готовых средств вывода для этих языков. Выразительные возможности применяемых языков позволяют описывать протоколы, параметризованные не только по числу процессов, но и использующие бесконечные типы данных, например, целочисленные счётчики.

Недостатки. Так как символьные методы основаны на представлении ИТС переходов параметризованного семейства \mathcal{F} в целом с помощью более выразительных языков, нежели LTS, размеченных пропозициональными переменными, завершаемость методов, в общем случае не гарантируется. В некоторых случаях завершаемость

может быть достигнута модификацией преобразователя T , например, с помощью применения техник добавления дополнительных переходов (акселерация, acceleration) и поиска аппроксимаций (расширение, widening). Однако в этом случае требуется участие эксперта с целью изменения преобразователя. Другим недостатком является требование повторного описания модели в параметризованном виде на языках, сильно отличающихся от языков описания моделей, используемых в средствах МС. После того, как модель распределённой ИТС построена и верифицирована для фиксированного числа процессов с помощью стандартного средства МС, желательно использовать выбранный уровень абстракции и перевести модель в параметризованный вид с минимальным вмешательством эксперта.

В методах, основанных на поиске инварианта, ищется или конструируется процесс-инвариант, моделирующий поведение нескольких процессов ИС. Инвариант должен обладать следующим свойством: ИС, состоящая из инварианта с дополнительными однотипными процессами, также моделируется инвариантом. С помощью индукции показывается, что ИС со сколь угодно большим числом однотипных процессов моделируется системой, состоящей из фиксированного числа процессов и инварианта. Основное отличие от методов абстракции состоит в том, что не требуется явное отображение каждой модели параметризованного семейства на модель-инвариант.

Достоинства. В методах инвариантов используется способ индуктивного доказательства и довольно простой критерий проверки инварианта. Поэтому методы могут быть автоматизированы. Поиск инварианта среди моделей семейства \mathcal{F} с малым числом процессов соответствует интуитивному представлению инженеров, разрабатывающих модель, о поведении модели. С помощью методов инвариантов могут проверяться параметризованные модели с нетривиальной топологией, представляющей сетевыми грамматиками, и различными прототипами процессов. Существенным достоинством является и то, что после нахождения инварианта (или нескольких инвариантов в случае сетевой грамматики), для проверки модели, содержащей инвариант, достаточно применить развитые средства метода верификации моделей.

Недостатки. В тех случаях, когда не удается найти инвариант среди моделей с малым числом процессов, необходимо дополнительно применять методы абстракции, требующие знаний эксперта. Так как автоматический поиск инварианта проводится среди моделей се-

мейства, в общем случае для процедуры поиска нет гарантии завершаемости. Однако, на практике инварианты необходимо искать среди моделей с довольно небольшим числом процессов, так как размер множеств состояний в моделях параметризованного семейства растёт экспоненциально с числом процессов.

На основе описания достоинств и недостатков представленных методов решения задачи РМСР, большинство аналитических методов редукции и методов абстракции обладает свойствами алгоритмизации и завершаемости.

Однако область применимости обеих групп методов невелика: асинхронные модели, в которых процессы обмениваются маркером; асинхронные модели с широковещательной передачей сообщений; асинхронные модели с топологией «звезда». При необходимости проверки моделей с другой топологией требуется разработка новых методов, возможно, аналогичных уже известным.

Сравним символьные методы и методы инвариантов. Символьные методы и методы инвариантов применяются к параметризованным моделям с различной топологией: кольцевой, линейной, древовидной. Символьные методы применяются, в основном, для проверки свойств безопасности, в то время как многие методы инвариантов применимы и для проверки свойств живучести.

Обе группы методов используют полуразрешающие процедуры, поэтому, в общем случае, не завершаются.

Для применения символьных методов описание параметризованной модели требуется представить на специальном языке метода, обычно довольно сильно отличающегося от языков описания моделей средств МС. Методы инвариантов используют описания процессов в виде LTS, размеченных пропозициональными переменными, и совместимы с языками описания моделей средств МС. Методы инвариантов используют для проверки спецификации и поиска контрпримеров существующие средства МС, символьные же методы требуют реализации специальных алгоритмов.

Учитывая вышесказанное, в настоящей работе в качестве базового метода выбран метод инвариантов, а процедура порождения моделей параметризованного семейства и поиска инвариантов основана на процедуре, используемой в методах сетевых инвариантов.

Метод сетевых инвариантов с использованием квазиблочной и блочной симуляций. Идея метода заключается в поиске моделей-инвариантов среди LTS, соответствующих деревьям вывода, с ис-

пользованием отношения блочной симуляции. Если такие модели-инварианты найдены, задача РМС может быть сведена к проверке конечного числа моделей методом МС.

При поиске инварианта сначала ищутся инварианты для тех нетерминалов A , из которых выводятся лишь терминалы или сам нетерминал A . После того, как инварианты для таких нетерминалов найдены, осуществляется поиск инвариантов для нетерминалов, использующих в правилах вывода другие нетерминалы.

Так как поиск осуществляется только среди LTS, выводимых из нетерминалов, перебор осуществляется полностью автоматически. Однако, не для каждого параметризованного семейства процедура перебора завершается. Алгоритм построения блочной симуляции является ключевым для поиска инварианта. Из этого следует, что достаточно ограничиться построением полублочной симуляции. Таким образом, алгоритм построения полублочной симуляции является ключевым в методе поиска инвариантов.

Алгоритм построения полублочной симуляции. В начале работы алгоритма пары начальных состояний первой и второй модели заносятся в множество неподтверждённых пар. На каждой итерации алгоритма проверяется гипотеза о том, что все неподтверждённые пары удовлетворяют определению полублочной симуляции. Если в ходе проверки обнаруживается, что некоторая пара не удовлетворяет этому определению, она заносится в множество опровергнутых пар. Проверка прекращается, как только множество неподтверждённых и опровергнутых пар перестаёт расширяться.

При анализе ПО выделяют два основных метода контроля и анализа программ: статический и динамический. Статические методы основаны на исследовании тех или иных свойств ПО без его выполнения с использованием различных инструментальных сред. Наиболее широкое распространение имеют следующие методы статического контроля:

- синтаксический контроль;
- контроль структуированности ПО;
- контроль правдоподобия программ;
- верификация ПО.

Методы верификации основаны на доказательстве корректности или некорректности ПО, соответствии разработанного ПО его спецификациям.

Верификация ПО весьма сложна, но практически гарантирует правильность полученного ПО относительно сформулированных для него спецификаций. К недостаткам методов верификации можно отнести то, что формирование метаописания ПО весьма сложно. Кроме того, верификация практически не применяется на поздних этапах жизненного цикла ПО.

Существует два основных вида тестирования ПО:

- функциональное тестирование;
- структурное тестирование.

Функциональное тестирование рассматривает программу как «черный ящик». Проверяется соответствие программы ее внешней спецификации. Отмечено, что исчерпывающее функциональное тестирование невозможно, а для программных продуктов большого размера применение его нецелесообразно. Структурное тестирование основано на знании внутренней структуры программы и предполагает обход всех ветвей по графу управления ПО.

2. АВТОМАТИЗАЦИЯ ПРОЦЕССА УПРАВЛЕНИЯ ДОСТУПОМ К ИНФОРМАЦИОННОМУ И ПРОГРАММНОМУ ОБЕСПЕЧЕНИЮ БАНКА

2.1. Автоматизация разграничения доступа к информационному и программному обеспечению банка

Для создания структурной модели управлением банком необходимо выявление взаимосвязей структурных подразделений банка с точки зрения управления защитой информационного и программного обеспечения банка. На рис. 10 представлена структура банка с точки зрения управления защитой информационного и программного обеспечения банка. Объектом защиты являются данные о деятельности банка, составляющие совершенно секретную информацию, коммерческую тайну, а также другую конфиденциальную информацию.

К совершенно секретным информационным и программным ресурсам банка относятся проекты, разработки банка, такие как проектно-конструкторская документация, программно-аппаратные продукты, сопровождающая документация к ним. К коммерческой тайне относится информация о материальных и финансовых ресурсах, вложенных в производство конечного продукта банка.

К другой конфиденциальной информации можно отнести идентифицирующую информацию сотрудников.

Согласно рис. 10, в банке существуют, как минимум, три уровня доступа к информационным и программным ресурсам:

- «красный» уровень, к которому относится доступ непосредственно к стратегически важным информационным и программным ресурсам банка, информации о разработках, проектах, сопровождающей документации к ним;

- «жёлтый» уровень – доступ к информационным и программным ресурсам банка, связанным с разработками, проектами, но не имеющим непосредственного отношения к совершенно секретным;

- «зелёный» уровень – отсутствие доступа к информационным и программным ресурсам банка, связанным с разработками и проектами банка.

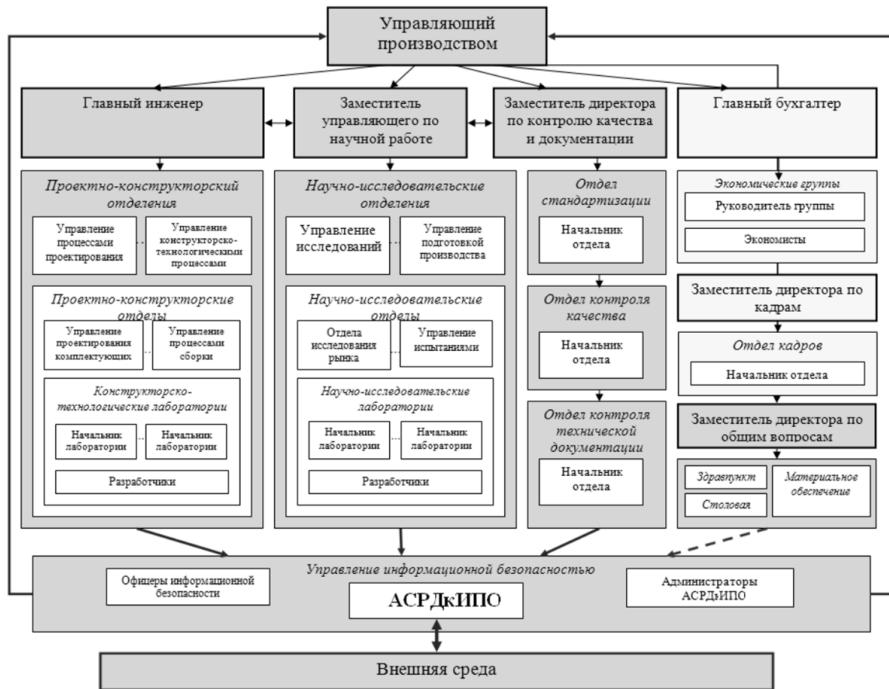


Рис. 10. Структура коммерческого банка в аспекте управления защитой информационного и программного обеспечения банка

К «красному» уровню доступа относятся структурные подразделения банка, которые имеют отношение к разработке проектов и их сопровождению. К сопровождению относятся: тестирование, проверка качества продуктов банка, проверка технической документации, проверка соблюдения установленных норм защиты информационного и программного обеспечения банка.

К «желтому» уровню доступа относятся структурные подразделения банка, которые производят обработку информации о проектах и разработках банка, не содержащей совершенно секретных данных. К ним относятся структуры, проводящие экономический расчёт, отдел кадров, отдел закупок. Экономическим группам, отделу снабжения нет необходимости знать детали разработок и проектов. Данная информация является для них избыточной. То же касается отдела кадров банка: сотрудники отдела кадров имеют доступ только к персональным данным других сотрудников банка.

К «зеленому» уровню доступа относятся подразделения банка, которым нет необходимости в ознакомлении со стратегически важными информационными и программными ресурсами банка. К ним относятся подразделения сторонних служб, например, здравпункт и столовая. Основной целью разрабатываемой АСРДкИПО является обеспечение должного уровня защиты информационного и программного обеспечения банка согласно требованиям ранжирования доступа, представленного на рис. 10. Данные об уровне доступа сотрудников структурных подразделений банка к информационным ресурсам банка представлен в табл. 5. Согласно представленной структурной модели управления банка с учётом требований защиты информационного и программного обеспечения банка, а также представленных в таблице уровней доступа подразделений банка к информации о проектах, требуется создание эффективного решения в виде отдельного модуля АСРДкИПО.

АСРДкИПО при работе с «красным» уровнем доступа должна отслеживать внутренние информационные потоки банка. Поддержка журнала аудита важна с целью детектирования инцидентов нарушений правил работы с информационным и программным обеспечением банка и дальнейшего их расследования.

Что касается внешних информационных потоков на «красном» уровне доступа, то лучшим вариантом является изолирование рабочих мест сотрудников, относящихся к «красному» уровню доступа, от ГВС (Интернет). Если полное изолирование невозможно, то необходимо обеспечить фильтрацию информационного потока в виде агента подсистемы класса DLP – Endpoint Solution. Также это касается переносных носителей данных и принтеров: лучшим вариантом является полный запрет на распечатку или хранение на съёмных носителях стратегически важной информации – организация хранения информационных ресурсов только в контуре ЛВС банка. Если это решение невозможно, то необходимо также применять подсистему класса DLP – Endpoint Solution. На рис. 10 видно, что также к «красному» уровню относятся отделы стандартизации, контроля качества и контроля технической документации. Перечисленные отделы также имеют доступ к стратегически важной информации о разрабатываемом проекте, поэтому им необходимо уделить особое внимание.

Таблица 5
Уровень доступа структурных подразделений банка к стратегически важным информационным ресурсам банка

| Должности руководящего состава и структурные подразделения предприятия | | | | | | | |
|--|-----------------|-------------------|---|--|-----------------------------------|--|----------------------|
| Генеральный директор, его заместители, кроме заместителя по общим вопросам | Главный инженер | Главный бухгалтер | Заместитель директора по общим вопросам | Проектно-изыскательские, испытательные, экспериментальные, конструкторские отделы, непосредственно занятыми разработкой текущего проекта | Отдел информационной безопасности | Отдел стандартации и контроля качества, технической документации | Экологические группы |
| Назначение | Полный доступ | Полный доступ | Нет доступа | Полный доступ | Полный доступ | Полный доступ | Нет доступа |
| Детальное описание модулей, их характеристики | Полный доступ | Полный доступ | Нет доступа | Полный доступ | Полный доступ | Полный доступ | Нет доступа |
| Экологические характеристики (название материалов, дополнительные затраты на приобретение экологического оборудования и амортизацию) | Полный доступ | Полный доступ | Нет доступа | Частичный доступ | Частичный доступ | Полный доступ | Нет доступа |
| Группы разработчиков, замечательных в создании и поддержке проекта | Полный доступ | Полный доступ | Частичный доступ | Полный доступ | Полный доступ | Полный доступ | Част. доступ |

Сопровождающая документация представлена в электронном или бумажном виде, и она также является стратегически важным информационным ресурсом, поэтому при работе с ней необходимо подключать все три составляющие АСРДкИПО: подсистему класса DLP – Endpoint Solution; подсистему слежения за внутренними потоками информации; подсистему разграничения доступа сотрудников к информационным и программным ресурсам банка. Основная задача отдела информационной безопасности банка – отслеживание, предотвращение и ведение расследований инцидентов нарушения правил работы с информационным и программным обеспечением банка. Также в полномочия сотрудников отдела входит проверка поступающего в банк аппаратного и программного обеспечения на наличие вирусов, программных и аппаратных закладок, подслушивающих устройств и т.д. Отдел информационной безопасности также несет ответственность за поддержку функционирования АСРДкИПО, поэтому его сотрудники должны проходить особую проверку (на полиграфе). «Who guards the guardians?» «Кто следит за следящими?» философский вопрос, который до сих пор не решён, поэтому подбор персонала в отдел информационной безопасности банка должен осуществлять особо строго.

Информационные ресурсы, доступные сотрудникам отдела кадров, является, несомненно, конфиденциальными, но не обязательно критичными для банка. Злоумышленник, перехвативший данные из отдела кадров, не будет осведомлён о разработках банка, более того, ему не будет доступен список сотрудников, задействованных в тех или иных текущих проектах, поэтому уровень доступа для сотрудников отдела кадров – «жёлтый». К информации о разработках сотрудники отдела кадров вовсе не имеют доступа, поэтому необходимости в очень строгом уровне контроля нет. Аналогична ситуация с экономическими службами банка – расчёт заработной платы и расходно-доходной части также производится изолированно. Сотрудники экономических отделов не имеют доступа к стратегически важным информационным ресурсам банка – к информации о разработках и проектах. Злоумышленник, перехвативший данные из экономического отдела максимум, что может узнать – это размер заработной платы сотрудника и кодировку проекта, в котором сотрудник задействован, поэтому все экономические службы банка имеют «жёлтый» уровень доступа. Сотрудники отделов снабжения и закупок осведомлены о материальных ресурсах, требуемых для основного производства, но

им присвоен «жёлтый» уровень доступа. Сотрудники общего отдела не имеют доступа ни к каким стратегически важным информационным ресурсам банка, поэтому они относятся к «зелёному» уровню доступа. Таким образом, три уровня доступа охватывают всех сотрудников банка. Основной задачей разрабатываемой АСРДкИПО является поддержание описанной выше методики управления защищённой информационного и программного обеспечения банка. На рис. 11 отображены части информации о проекте, к каждой из которых определен свой уровень доступа.

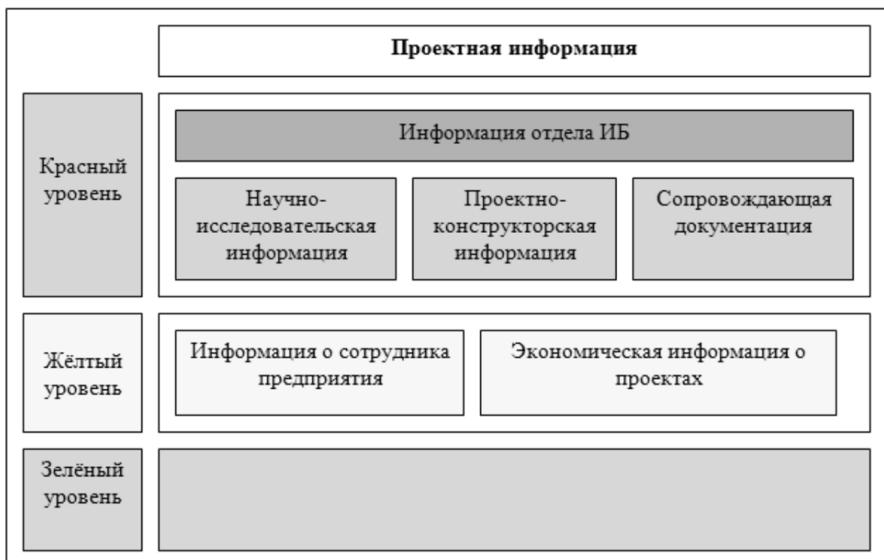


Рис. 11. Области доступа к информации о проекте

Для осуществления организации допуска сотрудника к той или иной части информации о проекте необходим дополнительный модуль – менеджер доступа. Основной задачей менеджера доступа является проведение аутентификации и авторизации сотрудника. Для работы менеджера потребуется база данных сотрудников банка. В базе данных должны содержаться имена сотрудников и их аутентифицирующая информация, в качестве которой могут выступать пароль, информация о материальном носителе, информация о биометрических характеристиках сотрудника; также в базе данных необходимо

наличие информации о метках доступа объектов информационных отношений – информационном и программном обеспечении банка.

На рис. 12 представлен алгоритм работы сотрудника банка с АСРДкИПО.

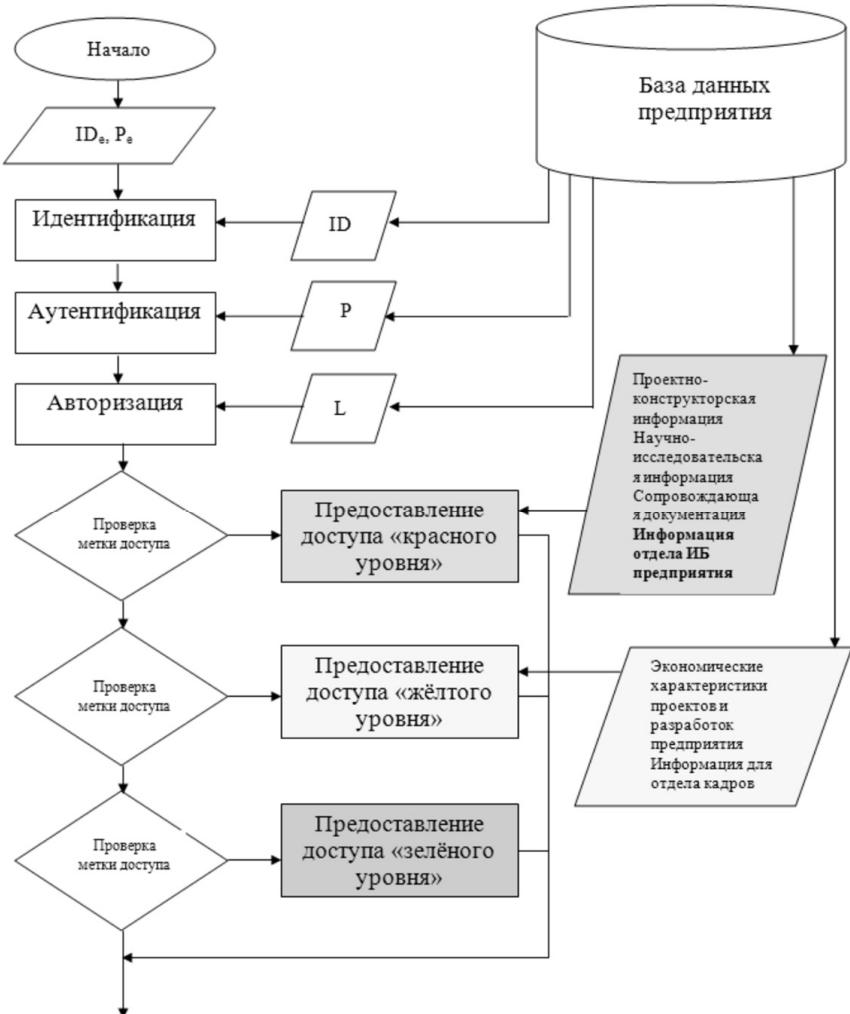


Рис. 12. Алгоритм работы сотрудника банка с АСРДкИПО

Перед началом работы сотрудника с ПО банка, управление передаётся АСРДкИПО. Сотрудник предоставляет идентифицирующую

информацию (IDe) и аутентификатор (Pe), в качестве которого могут выступать:

- пароль;
- материальный носитель;
- биометрические данные сотрудника.

Далее происходят процессы:

- идентификации;
- аутентификации;
- авторизации сотрудника.

Все процедуры проходят последовательно. При неудачном прохождении любой из них управление передаётся в начало алгоритма. После удачно пройденных последовательных процедур идентификации, аутентификации и авторизации сотруднику предоставляется метка доступа L. Метка доступа содержит данные об уровне доступа сотрудника к информационным массивам банка. Соответственно представленной структурной модели автоматизированного управления банком, метка доступа может определять «красный уровень», «жёлтый уровень» и «зелёный уровень» доступа.

Для реализации методики защиты информационного и программного обеспечения банка необходимо использования базы данных, которая содержит информацию как об объектах, так и о субъектах защиты.

На рис. 15 представлена схема таблиц базы данных банка.

В базе данных банка находятся шесть основных таблиц:

- таблица «Сотрудники»;
- таблица «Проекты»;
- таблица «Сопровождающая документация»;
- таблица «Экономические характеристики»;
- таблица «Отдела кадров предприятия»;
- таблица «Информационная безопасность».

Поля таблицы базы данных перечислены в табл. 6.

Таблица 6

Таблица базы данных «Сотрудники»

| ID | P | Descr | accessLevel |
|----|---|-------|-------------|
| | | | |

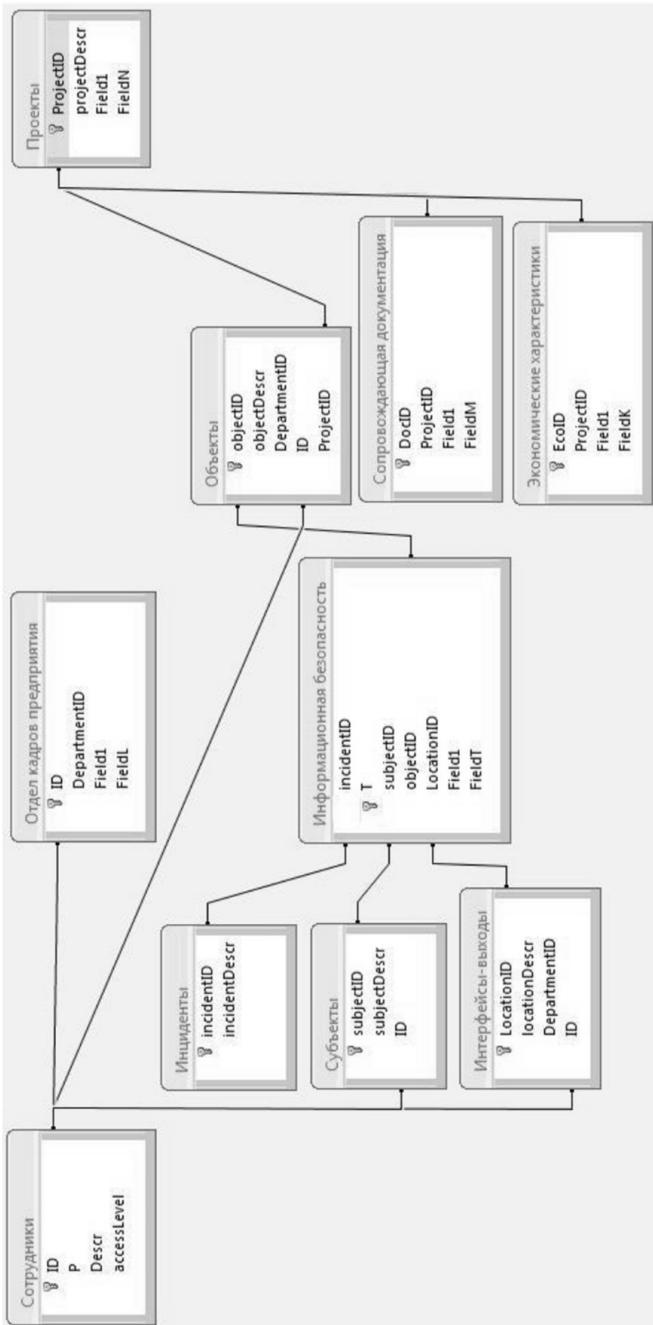


Рис. 13. Схема таблиц базы данных коммерческого банка

В табл. 6 имеются следующие поля:

- ID – идентифицирующая информация сотрудника (ключевое поле);
- P – аутентифицирующая информация сотрудника, в качестве которой могут выступать пароль, материальный носитель, биометрические данные сотрудника;
- Descr – описание сотрудника – поле, в котором могут храниться данные о сотруднике (ФИО, год рождения, подразделение банка, к которому относится сотрудник);
- Level – информация об уровне доступа сотрудника («красный», «жёлтый» или «зелёный»).

Поля таблицы базы «Проекты» данных перечислены в табл. 7.

Таблица 7

Таблица базы данных «Проекты»

| ProjectID | projectDescr | Field1 | ... | FieldN |
|-----------|--------------|--------|-----|--------|
| | | | | |

Как представлено в табл. 7, таблица базы данных «Проекты» имеет поля:

- ProjectID – идентификатор проекта (ключевое поле);
- Descr – описание проекта;

- Field1,..., FieldN – поля подробной информации о проекте. В качестве подробной информации о проекте могут выступать сроки сдачи проекта, дополнительные характеристики, входные данные, ограничения и т.д.

Поля таблицы базы данных «Сопровождающая документация» перечислены в табл. 8.

Таблица 8

Таблица базы данных «Сопровождающая документация»

| DocID | ProjectID | Field1 | ... | FieldM |
|-------|-----------|--------|-----|--------|
| | | | | |

Таблица базы данных «Сопровождающая документация» имеет поля:

- DocID – идентификатор документа (ключевое поле);

- ProjectID – идентификатор проекта, к которому относится документ;

- Field1,..., FieldM – поля подробной информации о документе.

В качестве подробной информации о документе могут выступать сроки сдачи документа, грифы секретности, дополнительные ограничения и т.д.

Поля таблицы базы данных «Экономические характеристики» перечислены в табл. 9.

Таблица 9

Таблица базы данных «Экономические характеристики»

| EcoID | ProjectID | Field1 | ... | FieldK |
|-------|-----------|--------|-----|--------|
| | | | | |

Таблица базы данных «Экономические характеристики» имеет поля:

- EcoID – идентификатор экономического показателя (ключевое поле);

- ProjectID – идентификатор соответствующего проекта;

- Field1,..., FieldK – поля подробной информации экономического показателя. В качестве подробной информации могут выступать стоимость, условия финансирования, номер контракта, договора и т.д.

Поля таблицы базы данных «Отдел кадров предприятия» перечислены в табл. 10.

Таблица 10

Таблица базы данных «Отдел кадров предприятия»

| ID | DepartmentID | Field1 | ... | FieldL |
|----|--------------|--------|-----|--------|
| | | | | |

Таблица базы данных «Экономические характеристики» имеет поля:

- ID – идентифицирующая информация сотрудника (ключевое поле);

- DepartmentID – идентификатор подразделения банка;

- Field1,..., FieldL – поля дополнительной информации о сотруднике банка. В качестве дополнительной информации могут высту-

пать: фамилия, имя, отчество, должность, семейное положение, дата принятия на работу, другие персональные данные сотрудника.

Поля таблицы базы данных «Информационная безопасность» перечислены в табл. 11.

Таблица 11

Таблица базы данных «Информационная безопасность»

| T | incidentID | subjectID | objectID | LocationID | Field1 | ... | FieldT |
|---|------------|-----------|----------|------------|--------|-----|--------|
| | | | | | | | |

Таблица базы данных «Информационная безопасность» имеет следующие поля:

- T – время инцидента (ключевое поле);
- incidentID – идентификатор инцидента нарушения правил работы с информационным и программным обеспечением банка;
- subjectID – идентификатор субъекта, породившего инцидент, в качестве которого может выступать как сотрудник организации, так и сама система (ее процесс);
- objectID – идентификатор объекта инцидента – части стратегически важного информационного пространства банка;
- LocationID – идентификатор места инцидента, в качестве которого могут выступать любые «интерфейсы-выходы» из ПО банка (порт связи с Интернет, USB-порт, принтер и т.д.);
- Field1,..., FieldT – поля дополнительной информации об инциденте нарушения правил работы с информационным и программным обеспечением банка. В качестве дополнительной информации могут выступать: коды ошибок и исключений (детализация происшествия) – информация для сотрудника отдела информационной безопасности банка, идентификатор «подозрительного» пакета передачи данных и т.д.

Вспомогательные таблицы базы данных. Для корректной работы АСРДкИПО с базой данной сотрудников банка необходимо наличие вспомогательных таблиц.

- таблица «Инциденты»;
- таблица «Субъекты»;
- таблица «Объекты»;
- таблица «Интерфейсы-выходы».

Поля таблицы базы данных «Инциденты» перечислены в табл. 12.

Таблица 12

Таблица базы данных «Инциденты»

| incidentID | incidentDescr |
|------------|---------------|
| | |

Таблица базы данных «Инциденты» имеет следующие поля:

- incidentID – идентификатор инцидента нарушения правил работы с информационным и программным обеспечением банка (ключевое поле);

- incidentDescr – описание инцидента.

Поля таблицы базы данных «Субъекты» перечислены в табл. 13.

Таблица 13

Таблица базы данных «Субъекты»

| subjectID | subjectDescr | ID |
|-----------|--------------|----|
| | | |

Таблица базы данных «Субъекты» имеет следующие поля:

- subjectID – идентификатор субъекта информационных отношений (ключевое поле);

- subjectDescr – описание субъекта;

- ID – идентификатор сотрудника, соответствующего идентификатору субъекта информационных отношений.

Поля таблицы базы данных «Объекты» перечислены в табл. 14.

Таблица 14

Таблица базы данных «Объекты»

| objectID | objectDescr | DepartmentID | ID |
|----------|-------------|--------------|----|
| | | | |

Таблица базы данных «Объекты» имеет следующие поля:

- objectID – идентификатор объекта информационных отношений (ключевое поле);

- objectDescr – описание объекта информационных отношений;

- DepartmentID – идентификатор подразделения банка, к которому относится соответствующий объект информационных отношений;

- ID – идентификатор сотрудника, несущего ответственность за соответствующий объект информационных отношений.

Поля таблицы базы данных «Интерфейсы-выходы» перечислены в табл. 15.

Таблица 15

Таблица базы данных «Интерфейсы-выходы»

| LocationID | locationDescr | DepartmentID | ID |
|------------|---------------|--------------|----|
| | | | |

Таблица базы данных «Интерфейсы-выходы» имеет следующие поля:

- LocationID – идентификатор потенциального места инцидента нарушения правил работы с информационным и программным обеспечением банка – «интерфейса-выхода» из информационной системы банка (ключевое поле);

- locationDescr – описание места инцидента;

- DepartmentID – идентификатор подразделения банка, к которому относится соответствующий «интерфейс-выход» из ПО банка.

- ID – идентификатор сотрудника, несущего ответственность за соответствующий «интерфейс-выход».

Для достижения требований принятой методики защиты информационного и программного обеспечения банка необходимо разграничить доступ к таблицам базы данных.

Сотрудники разных структурных подразделений банка обращаются к одному и тому же информационному ресурсу – базе данных сотрудников банка. Соответственно, к таблицам: «Информационная безопасность», «Инциденты», «Субъекты», «Интерфейсы-входы», «Объекты» имеют доступ только сотрудники отдела информационной безопасности. Сотрудники отдела кадров имеют доступ к таблице «Отдел кадров предприятия». Экономические группы (планово-финансовый отдел, бухгалтерия, отдел труда и т.д.) имеют доступ к таблице «Экономические характеристики». К таблице «Проекты» полного доступа не имеет никто. Разграничение доступа производится относительно строк таблицы, а не таблицы целиком. Также организуется доступ к таблице «Сопровождающая документация». Это связано с такой характеристикой информации, как полезность.

Сотрудник банка, принадлежащий к одному из отделов разработки, имеет доступ только к тем строкам (кортежам таблицы), которые относятся к проекту, над которым он работает в данный момент времени. Причём, эффективность работы сотрудника увеличится, если ему будет предоставлена максимально большая часть информации о проекте. Это связано с тем, что зачастую за надуманной секретностью разработки прячется корыстное утаивание от разработчика важной для него информации, в том числе информации о технических заданиях, о требованиях заказчика, о планах и сроках сдачи и т.д. – той информации, которая является важной для разработчика. Поэтому в рамках предлагаемой методики защиты информационного и программного обеспечения банка предполагается следующее утверждение: если конкретный разработчик (сотрудник одного из проектно-конструкторских или научно-исследовательских подразделений банка) имеет доступ к проекту, то этот доступ определяет максимальный уровень осведомленности об информации о этом проекте. Данное утверждение не противоречит принципу полезности информационных и программных ресурсов банка, описанном ранее. Для более высокого уровня защиты информационного и программного обеспечения банка доступ к таблице «Проекты» сотрудник получает не напрямую, а через таблицу «Объекты» (связь по идентификатору сотрудника «ID»). Аналогично, сотрудник экономической группы получает информацию о сотруднике банка (например, когда производит расчёт заработной платы) не напрямую, а через таблицы «Проекты», «Объекты», «Сотрудники». Полного доступа к таблице «Сотрудники» не имеет никто из физических лиц, в том числе и доступа по строкам. Обращение к этой таблице также происходит во время процедур идентификации, аутентификации и авторизации пользователей. Соответственно, это обращение формирует сама АСРДкИПО. Также можно организовать дополнительную защиту за счёт использования «двойной аутентификации». Сотрудник банка проходит процедуру во время входа в систему, а также во время обращения к данным. На рис. 14 отображено соответствие таблиц базы данных банка и структурным подразделениям. Примеры окон программы работы с базой данных банка приведены в приложении.

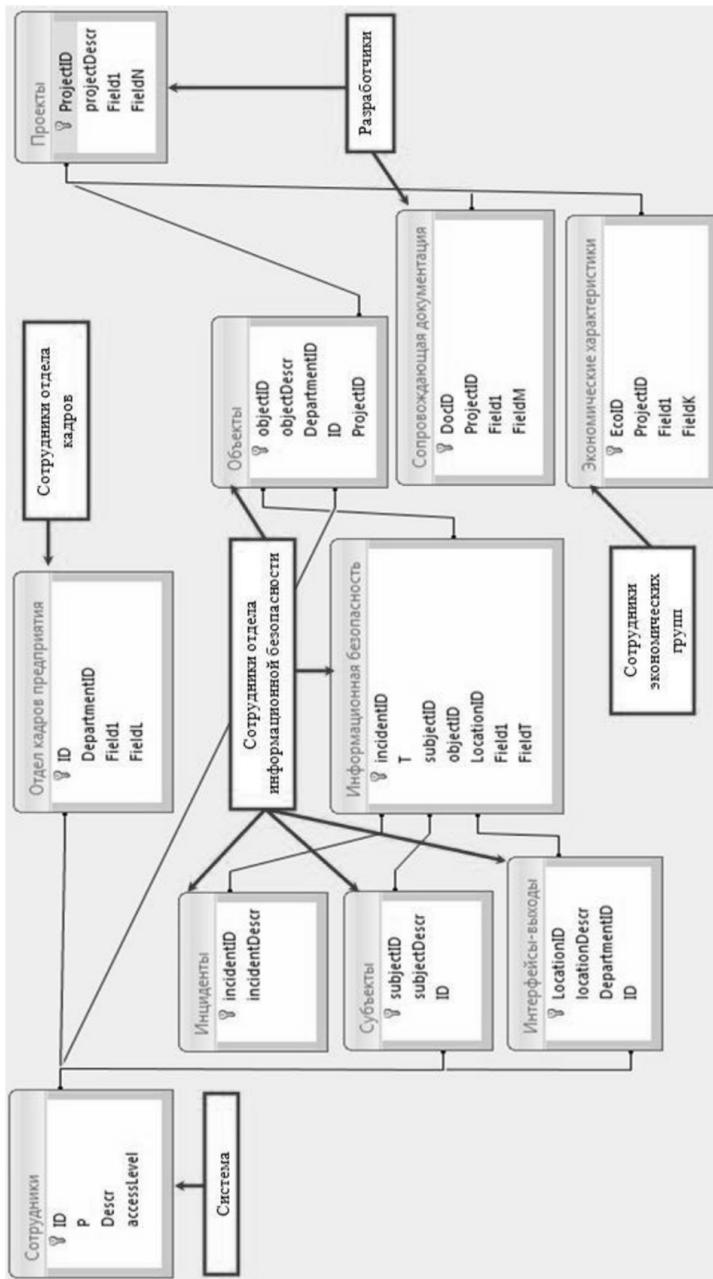


Рис. 14. Соответствие таблиц базы данных банка и доступа к ним сотрудникам структурных подразделений

Для фиксации инцидентов, а также для проведения расследований при разработке АСРДкИПО необходимо предусмотреть наличие журнала аудита. Журнал аудита представляет собой таблицу базы данных, а также копию данной таблицы в отдельном файле (чаще всего лог-файле). Файл-лог может иметь несколько копий в системе. Также необходимо предусмотреть тот факт, что журнал аудита должен быть доступен только для просмотра и только сотрудниками отдела информационной безопасности банка. «Дозапись» строк (кортежей) должна производиться только самой АСРДкИПО. Согласно представленным структурной модели автоматизированного управления банком на основе принципов ограничения доступа к стратегически важным информационным, программным ресурсам, и схеме базы данных, журнал аудита относится к таблице «Информационная безопасность». Полями таблицы являются:

- Идентификационный номер инцидента нарушения правил работы с информационным и программным обеспечением банка – является уникальным, поэтому является ключевым полем. Тип данных – счётчик;

- Время инцидента. Тип данных - время;

- Идентификатор субъекта, инициализировавшего инцидент (идентификатор сотрудника). Тип данных - GUID;

- Идентификатор объекта, к которому относится инцидент (файл конфиденциальной информации). Тип данных - GUID;

- Идентификатор места инцидента – физическое расположение инцидента (USB-порт, принтер, терминал сотрудника, терминал ГВС и т.д.). Тип данных - GUID;

- Дополнительные поля – поля, в которых располагается специфическая информация об инциденте. Как правило, данное поле имеет тип данных «строка», так как чаще всего специфической информацией об инциденте являются коды ошибок, инициализированные исключения.

Необходимо предусмотреть функцию просмотра таблицы «Информационная безопасность» с возможностью фильтрации по:

- времени;
- субъекту;
- объекту;
- месту происхождения инцидента.

Как видно из рис. 14 данная таблица связана с таблицами:

- «Инциденты», в которой хранится подробная информация об инциденте – описание инцидента и т.д.;
- «Субъекты», в которой хранятся данные о сотрудниках банка;
- «Объекты», в которой хранится подробная информация о файлах стратегически важных информационных ресурсах банка;
- «Интерфейсы-входы», в которой перечислены все возможные каналы утечки информационных ресурсов.

Перечисленные таблицы также доступны только для просмотра и записи только сотрудниками отдела информационной безопасности банка. Как правило, данные таблицы формируются при проектировании АСРДкИПО сотрудниками отдела информационной безопасности с занесение в них информации о файлах стратегически важных информационных и программных ресурсов банка, о сотрудниках, о физических расположения портов. В таблицы заносятся изменения в процессе работы АСРДкИПО при следующих ситуациях:

- изменение списка сотрудников;
- изменение списка файлов стратегически важных информационных ресурсах;
- изменение списка портов, физических устройствах, терминалов входа в ГВС (Интернет) и т.д.;
- изменение топологии ЛВС банка.

С целью повышения отказоустойчивости АСРДкИПО необходимо предусмотреть обязательное резервное копирование информации о нарушениях – сохранение таблицы инцидентов в отдельном хранилище «Архив инцидентов нарушения правил работы с информационным и программным обеспечением банка». Работу с «архивом инцидентов» необходимо организовать таким образом, чтобы запись в архив (изменение содержимого архива) могла производить только АСРДкИПО (без участия человека) – т.е. в автоматическом режиме. Чтение информации из архива должно быть доступно всем сотрудникам отдела информационной безопасности банка. Процесс работы с архивом инцидентов нарушений правил работы с информационным и программным обеспечением банка представлен на рис. 15.

Условные обозначения на рис. 15:

- зеленая стрелка – обращение по чтению;
- красная стрелка – обращение по записи.



Рис. 15. Процесс работы с архивом инцидентов

Подразделение информационной безопасности банка относится к «красному» уровню доступа, но ответственность за обеспечение функционирования АСРДкИПО ложится именно на сотрудников этого подразделения, поэтому данное подразделение можно классифицировать как подразделение высшего уровня доступа. Сотрудники подразделения информационной безопасности банка – офицеры безопасности – наделены высшим уровнем доступа, поэтому при зачислении в штат банка им необходимо пройти особую проверку, в том числе с использованием полиграфа. Для непосредственной организации и эффективного функционирования АСРДкИПО как системы обеспечения защиты информационного и программного обеспечения банка, исключающей возможные конфликты интересов, в банке необходимо наличие единого подразделения. На данное подразделение необходимо возложить следующие задачи:

- проведение в жизнь принятой методики обеспечения защиты информационного и программного обеспечения банка;
- анализ текущего состояния обеспечения защиты;
- организация мероприятий и координация работ всех подразделений банка по комплексной защите составляющих ПО банка;
- контроль и оценка эффективности принятых мер и применяемых средств защиты;

Основные функции подразделения обеспечения защиты составляющих ПО банка заключается в следующем:

- формирование требований к системам защиты (АСРДкИПО) в процессе создания и дальнейшего развития существующих компонентов банка;
- участие в разработке, сопровождению АСРДкИПО, испытаниях и приемке в эксплуатацию;
- обеспечение корректного функционирования АСРДкИПО;
- постоянный контроль и мониторинг основных программно-аппаратных модулей АСРДкИПО на наличие вредоносного программного и аппаратного обеспечения;
- проверка информационных файлов, генерируемых АСРДкИПО в ходе функционирования системы (так называемы log-файлов), в которых содержатся данные об информационных потоках и о работе основных пользователей ПО банка со стратегически важными информационными и программными ресурсами;
- реагирование на инциденты нарушения правил работы с информационным и программным обеспечением банка согласно соответствующим инструкциям и нормативным документам;
- проведение расследований инцидентов на основании полученных о АСРДкИПО файлов-логов;
- генерация и распределение между пользователями необходимых атрибутов доступа к информационным и программным ресурсам банка;
- наблюдение за функционированием АСРДкИПО и ее элементов;
- постоянная проверка надёжности функционирования АСРДкИПО;
- разработка мер нейтрализации моделей возможных атак;
- обучение пользователей и обслуживающего персонала правилам работы с информационным и программным обеспечением банка (в том числе особым правилам по работе с информационными ресурсами особой важности);
- оказание методической помощи сотрудникам банка в вопросах обеспечения параметров защиты;
- контроль за соблюдением пользователями (сотрудника банка) и обслуживающим персоналом установленных правил обращения с информационными и программными ресурсами банка (в том числе со стратегически важными);

- организация по указанию руководства служебного расследования по фактам нарушения правил обращения с информационными и программными ресурсами (в том числе со стратегически важными), оборудованием банка;

- принятие мер при попытках несанкционированного доступа к информационным и программным ресурсам и компонентам ПО банка или при нарушениях правил функционирования системы защиты (АСРДкИПО);

- сбор, накопление, систематизация и обработка информации по вопросам обеспечения защиты информационных и программных ресурсов банка.

Организационно-правовой статус подразделения обеспечения защиты банка должен определяться следующим образом:

- численность подразделения должна быть достаточной для выполнения всех перечисленных выше функций;

- сотрудники, занимающиеся обеспечением защиты информационного и программного обеспечения банка не должны иметь других обязанностей, связанных с обеспечением функционирования технических компонентов ПО банка;

- сотрудники подразделения обеспечения защиты должны иметь право доступа во все помещения, где установлены технические средства банка, и право прекращать обработку информационных ресурсов при наличии непосредственной угрозы для них;

- руководителю подразделения должно быть предоставлено право запрещать включение новых компонентов ПО банка в число действующих, если они не отвечают требованиям защиты и это может привести к серьезным последствиям в случае реализации значимых угроз нарушения параметров защиты;

- подразделению обеспечения контроля защиты ПО банка должны обеспечиваться все условия, необходимые для выполнения своих функций.

Основные компоненты АСРДкИПО представлены на рис. 16.

Основная задача, решаемые данной подсистемой разграничения доступа сотрудников банка к информационным и программным ресурсам банка – обеспечение соблюдения разграничения доступа за счёт применения организационных, правовых, технических (аппаратно-программные), физических мер защиты. Основные модули АСРДкИПО представлены на рис. 17.

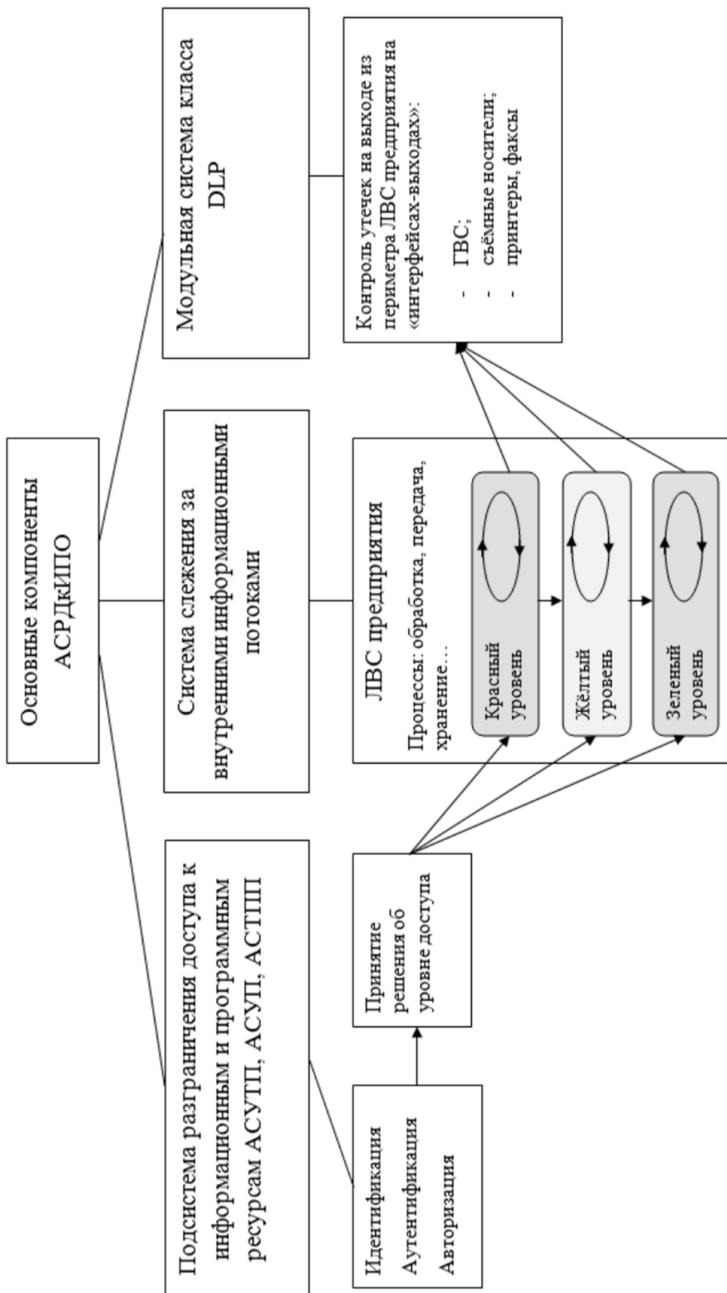


Рис. 16. Основные компоненты АСРДкИПО



Рис. 17. Основные модули АСРДкИПО

При разработке АСРДкИПО особое внимание уделялось применению аппаратно-программных мер защиты: применение алгоритма разграничения доступа, включающего процессы идентификации, аутентификации, авторизации.

Организационные меры защиты – это меры, регламентирующие процессы функционирования системы обработки данных, использование ее ресурсов, деятельность персонала, а также порядок взаимодействия пользователей с системой таким образом, чтобы в наибольшей степени затруднить или исключить возможность реализации угроз безопасности циркулирующей в системе информации.

К правовым мерам защиты относятся действующие в стране законы, указы и другие нормативные акты, регламентирующие правила обращения с информационными ресурсами и ответственность за их нарушения, препятствующие тем самым неправомерному их использованию и являющиеся сдерживающим фактором для потенциальных нарушений. Технические (аппаратно-программные) средства защиты – различные электронные устройства и специальные программы, которые выполняют (самостоятельно или в комплексе с другими средствами) функции защиты информационных и программных ресурсов банка (идентификацию и аутентификацию пользователей, разграничения доступа к ресурсам, регистрацию событий, криптографическое закрытие информации и т.д.). К физическим мерам защиты относят

разного рода механические, электронные или электронно-механические устройства и сооружения, специально предназначенные для создания физических препятствий на возможных путях проникновения и доступа потенциальных нарушителей к защищаемому информационному и программному обеспечению и другим ресурсам ПО банка, а также технические средства визуального наблюдения, связи охранной сигнализации. Система слежения за внутренними информационными потоками в банке позволяет отслеживать весь жизненный цикл стратегически важных информационных ресурсов банка – кто создал, изменил, где он (ресурс) хранится, кто из сотрудников его копировал, кому передавал по внутренней электронной почте и т.д.

Основной функцией модульной подсистемы класса DLP является отслеживание утечек информационных ресурсов банка через сеть, съёмные носители данных, принтеры и т.д. – т.е. детектирование угроз нарушения параметров защиты информационного и программного обеспечения банка происходит только «в месте» соприкосновения ПО банка с внешней средой – с ГВС, а также на всех «интерфейсах-выходах»: съёмные носители данных, факсы, принтеры. За мониторинг внутренних потоков конфиденциальной информации в банке подсистема класса DLP ответственности не несёт. Одной из задач модульной подсистемы класса DLP является контроль работы пользователей ПО банка в глобальной сети Интернет. Целью защиты стратегически важных информационных и программных ресурсов является предотвращение или минимизация наносимого ущерба (прямого или косвенного) субъектам информационных отношений посредством нежелательного воздействия на компоненты ПО банка, а также разглашения (утечки), искажения (модификации), утраты (снижения степени доступности) или незаконного тиражирования информационных ресурсов, некомпетентного и некорректного использования программных ресурсов. Таким образом, применение только одной из перечисленных технологий не способно решить поставленных в работе задач обеспечения защиты информационного и программного обеспечения банка. Для обеспечения требуемого уровня защиты необходимо применение всех трёх подсистем в едином комплексе.

2.2. Обеспечение надёжности и доступности компьютерных систем банка

Под надежностью понимается вероятность события, состоящего в том, что система выполняет требуемую задачу на временном интервале $[0, t]$. Доступность есть вероятность события, состоящего в том, что система выполняет требуемую задачу в определенный момент времени. В монографии рассмотрим модель без восстановления рабочей станции после сбоя и модель с восстановлением. Рассмотрим сначала случай без восстановления рабочей станции.

Пусть (i, j) есть состояние, в котором находится система. Индекс i обозначает количество работоспособных рабочих станций (принимает значения 2, 1, 0), индекс j принимает значения 1-сервер работает, 0-сервер находится в состоянии сбоя. Считаем, что система находится в работоспособном состоянии, если сервер и хотя бы одна из рабочих станций работоспособны. Множество возможных состояний системы есть $A = \{(0, 1), (1, 1), (1, 0), (2, 0), (2, 1)\}$. Пусть $X_t(w)$, $t \in \mathbb{R}_+$ есть случайный процесс, описывающий состояние работы системы, принимающий значения из множества A . Пусть $\pi_i(t) = P(w: X_t(w)=i)$, $i \in \{(2, 1), (1, 1), (0, 1), (2, 0), (1, 0)\}$. Предположим, что случайный процесс $X_t(w)$ есть марковская цепь со следующей диаграммой, рис. 18.

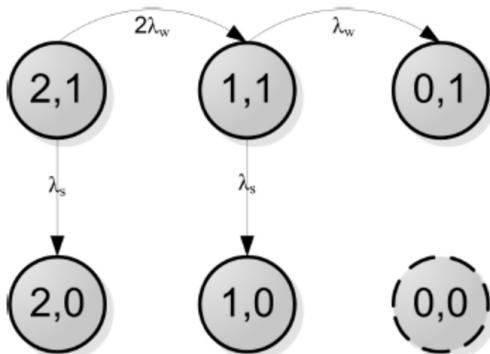


Рис. 18. Диаграмма марковской цепи

Величина надежности $R(t)$ для данной системы имеет вид $R(t) = \pi_{(2,1)}(t) + \pi_{(1,1)}(t) = 2e^{-(\lambda_w+\lambda_s)t} - e^{-(2\lambda_w+\lambda_s)t}$.

Диаграмма марковской цепи для системы с восстановлением представлена на рис. 19.

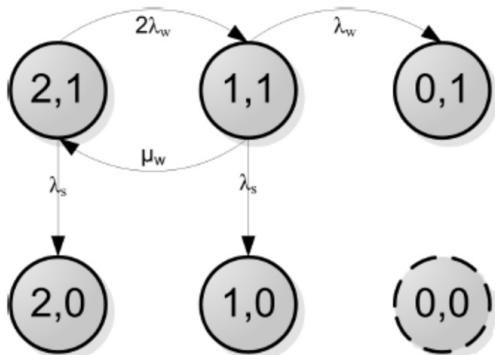


Рис. 19. Диаграмма марковской цепи для системы с восстановлением

$$R(t) = \left(\frac{\mu_w}{k_1 + 2\lambda_w + \lambda_s} + 1 \right) C_1 e^{k_1 t} + \left(\frac{\mu_w}{k_2 + 2\lambda_w + \lambda_s} + 1 \right) C_2 e^{k_2 t};$$

$$\begin{cases} C_1 = \frac{(k_1 + 2\lambda_w + \lambda_s)(k_2 + 2\lambda_w + \lambda_s)}{\mu_w(k_2 - k_1)}; \\ C_2 = -C_1; \end{cases}$$

$$k_{1,2} = \frac{-(3\lambda_w + 2\lambda_s + \mu_w) \pm \sqrt{\lambda_w^2 + \mu_w^2 + 6\lambda_w\mu_w}}{2}.$$

$$\lim_{\mu_w \rightarrow \infty} R(t) = e^{-\lambda_s t}.$$

Условие $\mu_w \Rightarrow +\infty$ подразумевает, что в случае сбоя одной из рабочих станций, осуществляется мгновенный переход в «идеальное» состояние (2,1). В этом случае показатель надежности $R(t)$ достигает максимума и его предельное значение зависит лишь от параметра λ_s -среднего времени сбоя сервера.

Далее исследуем метод обновления работы программы. Предполагаем, что с самого начала работы компьютерная система, на которой исполняется программа, находится в некотором идеальном состоянии. Вероятность сбоев в этом состоянии близка к нулю. Идеальное состояние обозначим через «0». Предполагаем, что пребывание системы в состоянии «0» не бесконечно и через некоторое время система переходит в состояние «P», в котором вероятность сбоя уже от-

лична от нуля. Происходит этот переход по причине того, что процесс функционирования системы со временем деградирует и начинает происходить постепенная утечка ресурсов компьютерной системы. Из состояния «Р» система через некоторое время переходит в состояние «F», что соответствует сбою в работе системы. Данный процесс можно формализовать с помощью следующей математической модели. Пусть $X_t(w), t \in R_+$ есть случайный процесс со значениями во множестве $\{0, P, F\}$. Предполагаем, что $X_t(w), t \in R_+$ есть марковская цепь, диаграмма которой представлена на рис. 20.

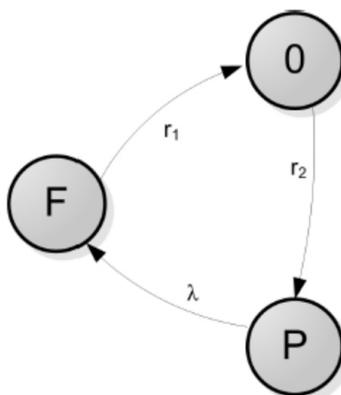


Рис. 20. Диаграмма марковской цепи $X_t(w), t \in R_+$

Пусть также процесс $X_t(w)$ однородный по времени и обладает марковским свойством. Обозначим

$$\pi_i(t) = P(X_t(\omega) = i), i \in \{0, P, F\}$$

Тогда справедлива система дифференциальных уравнений Колмогорова:

$$\begin{cases} \pi'_0(t) = -r_2\pi_0(t) + r_1\pi_F(t) \\ \pi'_P(t) = -\lambda\pi_P(t) + r_2\pi_0(t) \\ \pi'_F(t) = -r_1\pi_F(t) + \lambda\pi_P(t). \end{cases}$$

В силу эргодической теоремы

$$\exists \pi_i = \lim_{t \rightarrow \infty} \pi_i(t), i \in \{0, P, F\}.$$

Таким образом, имеем следующую систему уравнения для предельных вероятностей:

$$\begin{cases} -r_2\pi_0 + r_1\pi_F = 0 \\ -\lambda\pi_P + r_2\pi_0 = 0 \\ -r_1\pi_F + \lambda\pi_P = 0 \\ \pi_0 + \pi_P + \pi_F = 1. \end{cases}$$

Отсюда

$$\pi_F = \frac{1}{1 + \frac{r_1}{\lambda} + \frac{r_1}{r_2}}.$$

Величина π_F есть предельная вероятность пребывания системы в состоянии сбоя. С ее помощью можно оценить надежность работы компьютерной системы. Если величина π_F велика, то это говорит о низкой надежности системы. Величина доступности системы определяется как $\pi_A = 1 - \pi_F$. Далее рассмотрим модель, в которой система из состояния «P» может также переходить в состояние «R», в котором осуществляется обновление работы системы. Пусть диаграмма переходов системы имеет вид, представленный на рис. 21.

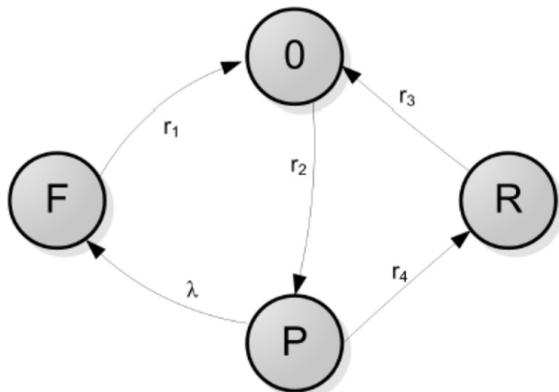


Рис. 21. Диаграмма переходов компьютерной системы

Считаем, что система во время пребывания в состоянии «R» не функционирует. Тогда выражение для предельной вероятности сбоя системы примет вид

$$\pi_F^{rejuv} = \left(\frac{\lambda}{r_1} + \frac{r_4}{r_3} \right) \cdot \frac{1}{1 + \frac{\lambda}{r_1} + \frac{r_4}{r_3} + \frac{\lambda + r_4}{r_2}}.$$

Величина доступности системы определяется как $\pi_A^{rejuv} = 1 - \pi_F^{rejuv}$.

1. Если $r_3 > r_1 \left(1 + \frac{r_2}{\lambda}\right)$, то π_F^{rejuv} убывает по параметру r_4 .

2. Если $r_3 < r_1 \left(1 + \frac{r_2}{\lambda}\right)$, то π_F^{rejuv} возрастает по параметру r_4 .

Последние выражения указывают достаточное условие, при котором метод обновления увеличивает величину предельной доступности системы.

Моделирование работы банкомата с применением ограничений доступа к программному обеспечению банка.

Модель общения банкомата с сервером банка построена на основе транзакций: в ходе взаимодействия с пользователем устройство банкомата накапливает вводимую информацию в специальном внутреннем списке, отправляя серверу по требованию совершения операции полный набор информации, описывающий транзакцию. Так, в начале работы пользователь вводит номер карты. После этого банкомат запрашивает у него PIN-код. Как номер карты, так и введенный PIN-код запоминаются во внутреннем списке. Банкомат запрашивает у сервера авторизацию для карты. В случае неверного ввода PIN-кода, карта возвращается.

После успешной проверки PIN-кода клиенту становится доступно основное меню, в котором он может выбрать одно из действий:

- забрать карту;
- посмотреть остаток счета;
- снять деньги со счета.

При выборе первого пункта пользователю возвращается карта, и работа с банкоматом завершается. При выборе последнего пункта пользователю необходимо дополнительно выбрать одну из предопределенных сумм для снятия или ввести сумму с клавиатуры. После этого, как и в случае выбора второго варианта, автомат отправляет серверу накопленные данные. При получении ответа от сервера клиенту отображается информация о результате операции. После этого банкомат вновь отображает главное меню, или клиент забирает карту.

Управление осуществляется системой взаимодействующих автоматов. Формальное описание, содержащее в себе объекты управления, источники событий и систему автоматов, приведено ниже. На рис. 22 приведена схема связей автоматов.

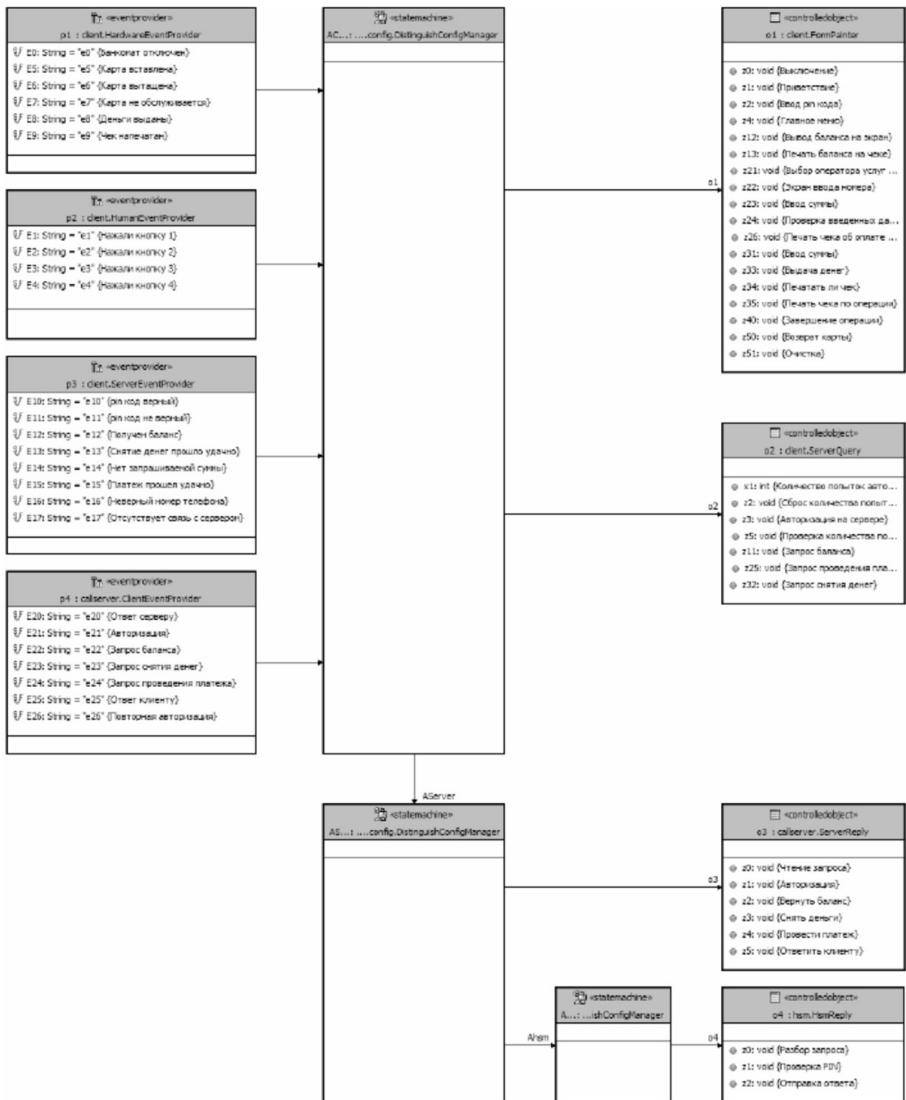


Рис. 22. Схема связей автоматов с поставщиками событий и объектами управления

На рис. 23 представлен график переходов автомата, управляющего банкоматом.

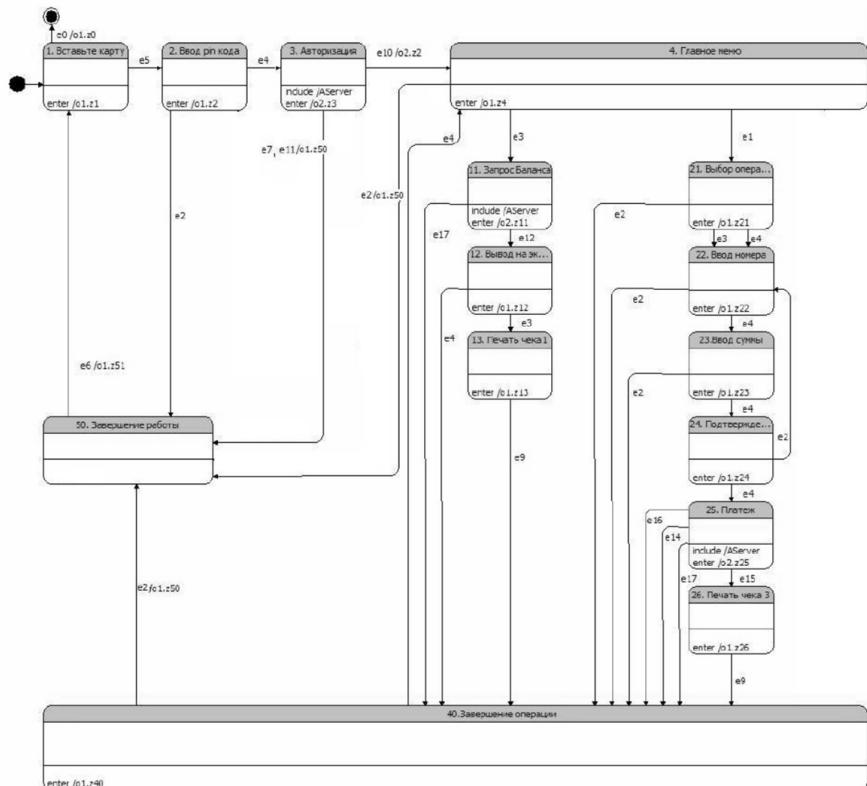


Рис. 23. Граф переходов автомата, отвечающего за работу банкомата

Рассмотрим ситуацию, когда к созданной программе изменяются требования. Пусть сценарий работы с банкоматом изменился следующим образом: «В случае неверного ввода PIN-кода он запрашивается повторно. В случае неверного ввода PIN-кода три раза подряд работа с банкоматом принудительно завершается».

Граф переходов автомата, отвечающего за работу банкомата, достаточно громоздок. Упростим его, выделив два вызываемых автомата: один для режима просмотра остатка счета, второй для режима снятия денег со счета. Граф переходов автомата после применения таких рефакторингов представлен на рис. 24.

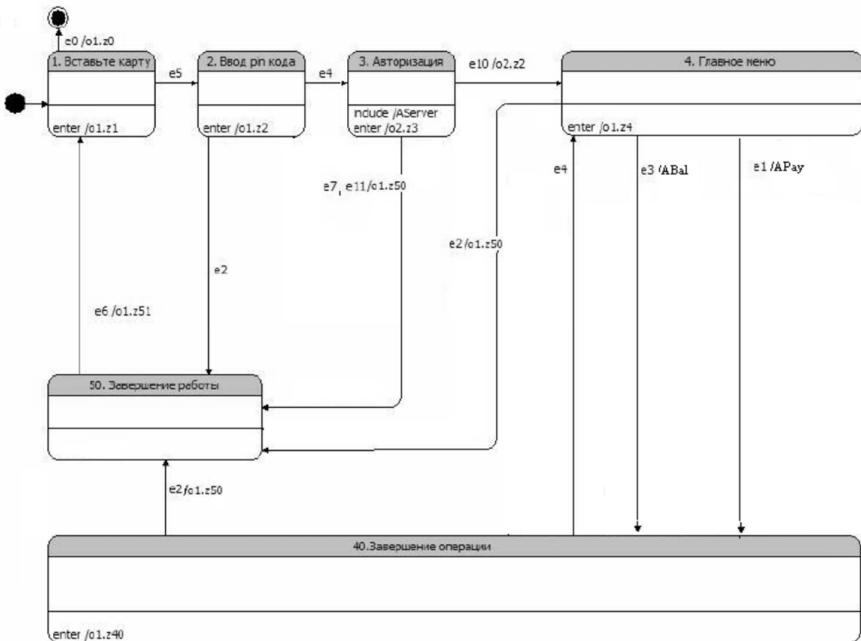


Рис. 24. Граф переходов автомата, отвечающего за работу банкомата, после выделения вызываемых автоматов

Граф переходов стал существенно проще для восприятия. В новом графе существует фрагмент, который нуждается в изменении: из состояния 3 («Авторизация») по событию $e11$ («pin код неверный») совершается действие $o1.z50$ («Возврат карты») и переход в состояние 50 («Завершение работы»). Необходимо, чтобы по событию $e11$ карта не возвращалась, а производился выбор действия в зависимости от числа неверных попыток ввода PIN-кода. Получается, что следует некоторым образом разделить связь события $e11$ и выходного воздействия $o1.z50$. Самый простой способ сделать это – осуществить перенос действий с переходов, ведущих в состояние 50, внутрь этого состояния.

Граф переходов автомата, получающийся после применения такого рефакторинга, представлен на рис. 25.

По событию $e11$ должно осуществляться ветвление в зависимости от числа попыток. Поэтому добавим в граф переходов состояние 5 («Неверный PIN-код»), в котором будет это ветвление осуществляться. Теперь установим конечным состоянием перехода по собы-

тию e_{11} добавленное состояние. Полученный граф переходов представлен на рис. 26.

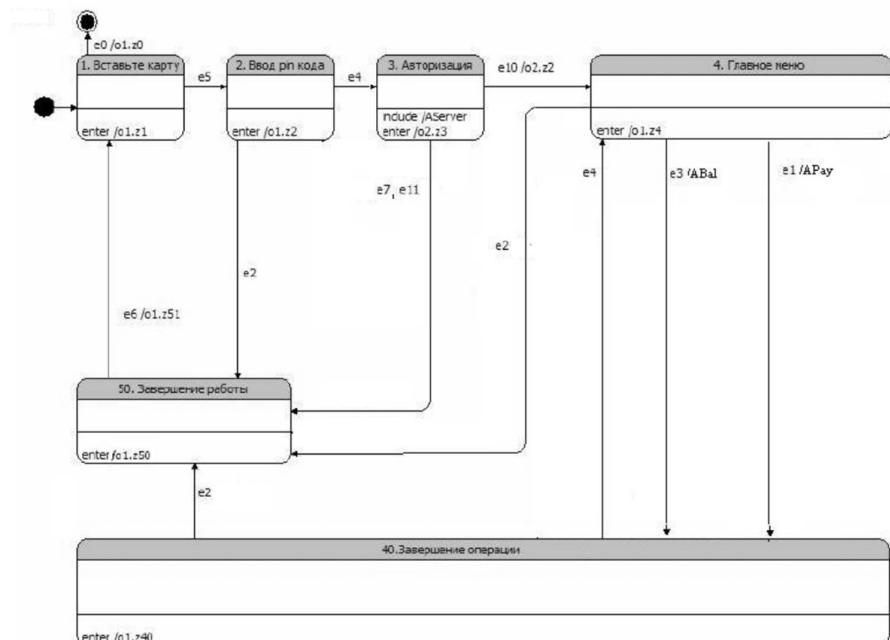


Рис. 25. Граф переходов автомата, отвечающего за работу банкомата, после переноса воздействия внутрь состояния

При входе в состояние 5 выполняется выходное воздействие $o2.z5$ («Проверка количества попыток»). Воздействие $o2.z5$ активизирует одно из двух событий: e_{26} («Повторная авторизация») или e_7 («Карта заблокирована»). В первом случае, автомат должен перейти в состояние 2, во втором – в состояние 50. Соответствующие изменения графа переходов легко осуществить (рис. 27).

Полученный график переходов удовлетворяет всем поставленным требованиям. Стоит отметить, что при модификации программы применялись только рефакторинги и базовые изменения, что позволило избежать анализа графа переходов после каждого проведенного изменения.

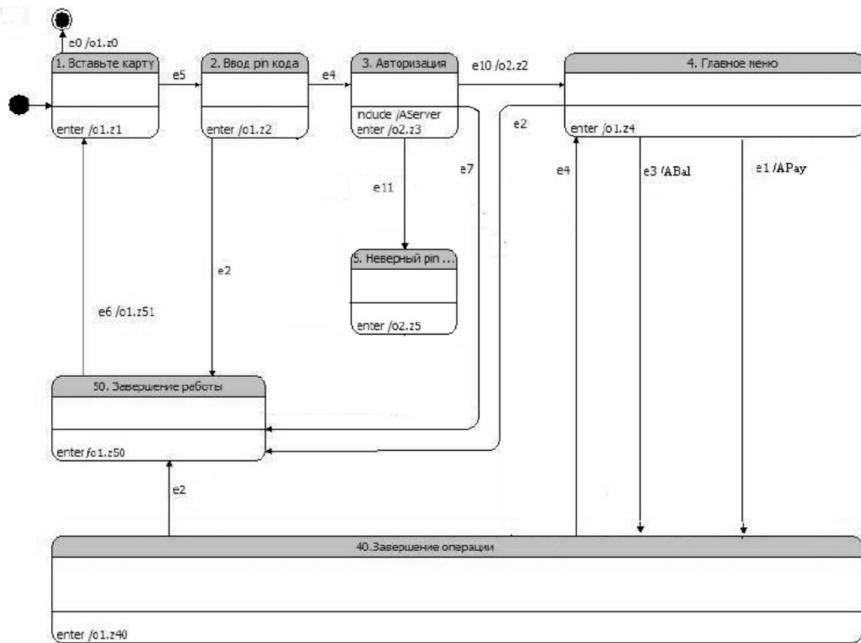


Рис. 26. Граф переходов автомата, отвечающего за работу банкомата, после добавления нового состояния

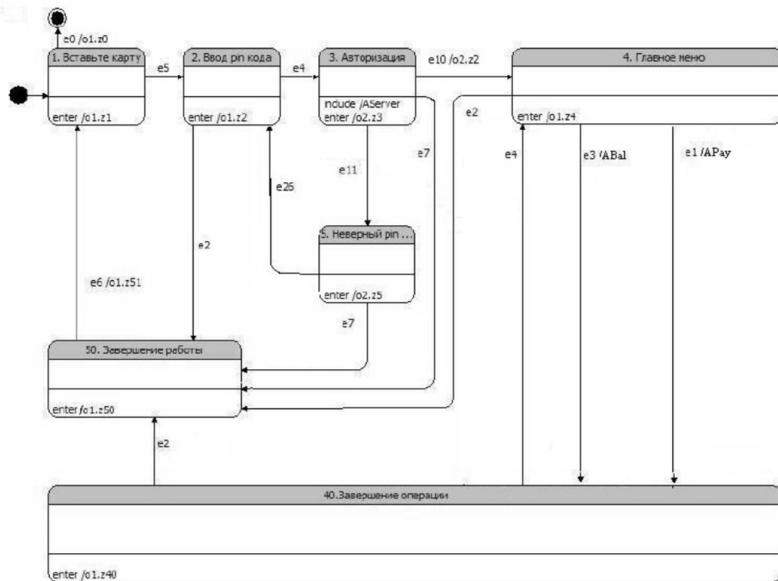


Рис. 27. Граф переходов автомата, отвечающего за работу банкомата

3. АНАЛИЗ ДОСТОВЕРНОСТИ И ВЕРИФИКАЦИИ ИНФОРМАЦИИ В ИНФОРМАЦИОННО- ТЕЛЕКОММУНИКАЦИОННЫХ СИСТЕМАХ БАНКА

3.1. Алгоритм анализа достоверности и верификации информации

Алгоритм построения отношения полублочной симуляции между двумя моделями при анализе достоверности и верификации информации в информационно-телекоммуникационных системах КБ. В разделе приведена упрощённая версия алгоритма. Различные оптимизации, нацеленные на повышение эффективности, указаны в приложении А.

При описании алгоритма используются синтаксис и семантика языка C++, дополненные следующим синтаксисом, типами и вспомогательными ключевыми словами:

- Тип $\text{tuple } < T_1, \dots, T_n >$ описывает кортеж над типами T_1, \dots, T_n . Кортеж может записываться следующим образом: (a_1, \dots, a_n) , где a_i – значение или переменная типа T_i . Кортежи поддерживают операции присваивания и сравнения.
- Тип $\text{set } < T >$ описывает множество из элементов типа T . Тип set соответствует типу `std::set` стандартной библиотеки языка C++. Для краткой записи пользуемся математическими обозначениями пустого множества (\emptyset) и операций: $\in, / \in, \cup, \cap, \setminus$.
- Тип $\text{vector } < T >$ описывает массив из элементов типа T . Тип vector соответствует типу `std::vector` стандартной библиотеки языка C++. Для типа $\text{vector } < T >$ используются стандартные функции: `push_back`, `pop_back`, `back`, `operator[]`, `size`.
- Тип $\text{list } < T >$ описывает список из элементов типа T . Тип list соответствует типу `std::list` стандартной библиотеки языка C++. Для типа $\text{list } < T >$ используются стандартные функции: `push_back`, `pop_back`, `back`, `size`.
- Цикл по множеству: `foreach x in X`. Перебирает элементы множества X в произвольном порядке, на каждой итерации текущий элемент множества X сохраняется в переменной x . Оценка сложности

операции зависит от структуры данных, используемой для представления множества.

- Выбор произвольного элемента из множества: $x = \text{any from } X$.

Записывает в переменную x произвольный элемент непустого множества X . В случае пустого множества X выполнение оператора приводит к ошибочной ситуации. Оценка сложности операции зависит от структуры данных, используемой для представления множества.

Используемые типы:

- тип State_1 описывает состояния LTS M_1 ,
- тип State_2 описывает состояния LTS M_2 ,
- тип Action_1 описывает действия LTS M_1 ,
- тип Action_2 описывает действия LTS M_2 .

Для построения полублочной симуляции воспользуемся подходом, основанным на итеративном вычислении неподвижной точки. Для того, чтобы избежать построения первого приближения отношения, которое на практике требует хранения большого количества состояний, инициализировали множество H «ленивым» способом. Строящееся множество H делится на два непересекающихся подмножества: опровергнутые пары (*negatives*) и неподтверждённые пары (*positives*).

В начале работы алгоритма пары начальных состояний первой и второй модели заносятся в множество неподтверждённых пар. На каждой итерации алгоритма проверяется гипотеза о том, что все неподтверждённые пары удовлетворяют определению полублочной симуляции. Если в ходе проверки обнаруживается, что некоторая пара не удовлетворяет этому определению, тогда она заносится в множество опровергнутых пар. Проверка прекращается, как только множества неподтверждённых и опровергнутых пар перестаёт расширяться.

Листинг. Краткое описание алгоритма построения полублочной симуляции.


```

23 // Проверка пары в множествах неподтверждённых и опровергнутых пар.
24 // Если пары нет в обоих множествах, пара добавляется в множество
25 // неподтверждённых. В случае, если пара находится в множестве
26 // опровергнутых, возвращается ложь. Иначе возвращается истина.
27 bool lazyHasPositive(State1 s1, State2 s2);
28 // Найдёт все финальные переходы блоков из состояния s1,
29 // т.е. такие пары  $(a, t) \in A_1 \setminus \{\theta\} \times S_1$ , что
30 // найдётся путь  $s_1 \xrightarrow{\theta^*} s' \xrightarrow{a} t$ .
31 set<tuple<Action1, State1>> buildFront1(State1 s1);
32 // Найдёт все переходы  $u \xrightarrow{a} v$ ,  $a \neq \theta$ , достигимые из s2,
33 // т.е. такие, что  $s_2 \xrightarrow{\theta^*} u$ .
34 set<tuple<State2, Action2, State2>> buildFront2(State2 s2);
35 // Проверка определения полублоочной симулляции для
36 // заданной пары относительно множеств P, Pold и N.
37 bool satisfiesDef(State1 s1, State2 s2) {
38     set<tuple<Action1, State1>> f1 = buildFront1(s1);
39     set<tuple<State2, Action2, State2>> f2 = buildFront2(s2);
40     foreach (u1, a2, u2) in f2 {
41         foreach (a1, t1) in f1 such that a1 == a2
42             if (lazyHasPositive(t1, u2)
43                 && (u1 == s2 || lazyHasPositive(s1, u1))
44                 && (s1  $\notin$  PreCycle || u1  $\in$  InCycle))
45                 f1 = f1 \ {(a1, t1)};
46     }
47     return f1 ==  $\emptyset$ ;
48 }
49 // Итеративное построение отношения полублоочной симулляции.
```

```

50 // До тех пор, пока множество  $P$  изменяется по сравнению с
51 // множеством  $P_{old}$ , для каждой пары состояний из множества
52 //  $P_{old}$  проводится шаг проверки. Если пара удовлетворяет
53 // определению, она добавляется в множество  $P$ , иначе –
54 // добавляется в множество  $N$ .
55 void build() {
56     init();
57     modified = true;
58     while (i == 1 || modified) {
59         modified = false;  $P_{old} = P$ ;  $P = \emptyset$ ;
60         while ( $P_{old} \neq \emptyset$ ) {
61             ( $s_1, s_2$ ) = any from  $P_{old}$ ;
62              $P_{old} = P_{old} \setminus \{(s_1, s_2)\}$ ;
63             if (satisfiesDef( $s_1, s_2$ )) {
64                 insertPositive( $s_1, s_2$ );
65             } else {
66                 insertNegative( $s_1, s_2$ );
67                 modified = true;
68             }
69         }
70     }
71 }
```

Функция `build` строит отношение полублочной симуляции описаным выше способом.

Функция `satisfiesDef` проверяет определение полублочной симуляции для одной пары состояний, используя множества P_{old} и N .

Функция `satisfiesDef` использует функции `lazyHasPositive`, `buildFront1` и `buildFront2`.

Функция `lazyHasPositive` проверяет наличие пары в множествах $P \cup P_{old}$ и N . Если пара присутствует в множестве $P \cup P_{old}$ (множестве N), функция возвращает истину (ложь соответственно). Если же пара не найдена в обоих множествах, тогда это означает, что она по-

сещается в первый раз, и в этом случае пара заносится в множество неподтверждённых пар P , а функция возвращает истину.

Функции `buildFront1` и `buildFront2` строят множества финальных переходов из заданного состояния LTS M_1 и M_2 соответственно (фронты).

В качестве верификатора моделей ПО, используемого для проверки формул темпоральной логики, выбран верификатор Spin. На вход ИТС подаются описания прототипов процессов и сетевой грамматики. По сетевой грамматике строятся модели, порождаемые нетерминальными символами грамматики. Для каждого нетерминала грамматики проводится поиск модели-инварианта среди моделей, выводимых из данного нетерминала. Проверка того, что модель является инвариантом, осуществляется с помощью построения полублочной симуляции. Когда для каждого нетерминала найдена модель-инвариант, может быть построен инвариант проверяемого семейства моделей.

В тех случаях, когда найти инвариант не представляется возможным, предлагается использовать подсистему нахождения контрпримеров по построенному отношению полублочной симуляции между моделями. С помощью контрпримеров могут быть обнаружены различия в поведении моделей семейства с различным числом процессов.

Протокол резервирования ресурсов предоставляет механизм резервирования сетевых ресурсов для соблюдения заданного качества сервиса (QoS). Протокол реализован на транспортном уровне стека протоколов TCP/IP. Данный протокол используется для обеспечения определённой скорости передачи аудио- и видеотрафика от сервера к потребителю. На уровне протокола выделяются три вида взаимодействующих хостов: производители данных (*senders*), потребители данных (*receivers*) и маршрутизаторы (*routers*). Потребители, производители и маршрутизаторы объединены сетью. Потребители подписываются на получение данных от заданных производителей, производители рассыпают данные подписанным потребителям. Подписка потребителя проводится рассылкой соответствующим производителям запроса на резервирование канала от производителя к потребителю. В запросе на резервирование задаются ограничения на пропускную способность канала и скорость реакции. При прохождении запроса через маршрутизатор принимается решение, есть ли у него необходимые ресурсы для поддержания качества сервиса или нет. Если ре-

сурсов недостаточно, маршрутизатор отсылает потребителю отказ в резервировании. Если ресурсы имеются, тогда маршрутизатор пересыпает запрос следующему хосту. После успешного резервирования канала создаётся сессия и производитель отсылает данные потребителю в рамках созданной сессии до тех пор, пока одна из сторон не закроет сессию.

Так как протокол находится на транспортном уровне, предполагается, что маршруты между хостами построены на сетевом уровне. В протоколе используются два вида рассылки: адресная (unicast) и групповая (multicast). В RSVP предусмотрена возможность подтверждения резервирования при изменении маршрутов в сети. В общем случае хост-потребитель может быть и хостом-производителем.

В протоколе определены следующие вспомогательные механизмы: классификатор пакетов (packet classifier), планировщик пакетов (packet scheduler), механизм контроля доступа (policy control), механизм контроля соединения (admission control). Классификатор пакетов выбирает уровень качества сервиса и возможные пути следования пакетов. Планировщик пакетов определяет порядок отправки пакетов. Механизм контроля доступа определяет права пользователя на резервирование ресурсов в заданном объёме. Механизм контроля соединения проверяет наличие необходимых сетевых ресурсов на промежуточных узлах сети.

В протоколе определены следующие типы сообщений, которыми обмениваются производители и потребители:

- сообщение path рассыпается производителями для оповещения потребителей о наличии сетевого ресурса;
- сообщение resv рассыпается потребителями после получения сообщения path с целью подписки на ресурс производителя;
- сообщение path_teardown рассыпается производителем для закрытия всех существующих соединений;
- сообщение resv_teardown рассыпается потребителем с целью прекращения подписки на сетевой ресурс производителя;
- сообщение path_refresh периодически рассыпается производителем для проверки работоспособности установленных соединений;
- сообщение resv_refresh периодически рассыпается потребителем для проверки работоспособности соединения;
- сообщение path_error посылается маршрутизатором в случае ошибки во время рассылки path;

- сообщение `resv_error` посыпается маршрутизатором в ответ на запрос о резервировании (сообщение `resv`) в случае отсутствия необходимых ресурсов;
- сообщение `data` рассыпается производителям для передачи данных потребителям;
- сообщение `resv_conf` (необязательное) рассыпается потребителям для подтверждения резервирования ресурса.

Схема взаимодействия процессов. После установления сетевых маршрутов (на сетевом уровне) производители рассыпают сообщение типа `path` для уведомления потребителей о наличии сетевых ресурсов. Маршрутизаторы пересыпают эти сообщения по дереву маршрутизации. После приёма сообщения `path` потребитель может послать в ответ запрос на резервирование ресурса `resv` с заданными параметрами качества сервиса. Маршрутизаторы проверяют, выполняются ли требования к качеству сервиса. Если требования не выполняются, маршрутизатор отвечает сообщением `resv_error`. В противном случае маршрутизатор пересыпает сообщение следующему хосту в таблице маршрутизации. В случае получения сообщения `resv` производитель создаёт сессию для обмена данными с потребителем и начинает посыпать сообщения типа `data`, содержащие данные для потребителя. В любой момент производитель, потребитель или маршрутизатор могут проверить работоспособность соединения с помощью сообщений типов `aresv_refr` и `path_refr`. Производитель (потребитель) может закрыть сессию рассылкой сообщения `path_teardown` (сообщения `refr_teardown` соответственно).

При создании модели протокола по его описанию необходимо провести абстракцию с учётом выбранного класса свойств. Используем модель, в которой можно проверять спецификации для производителей, потребителей и маршрутизаторов по отправке/приёму сообщений в терминах логики линейного времени.

Ниже перечислены основные абстракции, которые использовались при построении модели рассматриваемого протокола.

Сообщения. В протоколе используются два вида рассылки сообщений: адресная и групповая. При адресной рассылке каждый потребитель задаёт адрес производителя, на сообщения которого он хочет подписаться, и сохраняет в пакете свой собственный адрес. В случае адресной рассылки модель должна описывать адрес узла, при этом размер типа адреса растёт с числом узлов в сети. Так как метод не поддерживает модели с бесконечным числом состояний, адресная

рассылка из модели исключена. В модели используется групповая рассылка в пределах одной группы.

Логическая топология сети. Протокол располагается на транспортном уровне и использует протоколы маршрутизации сетевого уровня. Так как в модели рассматривается групповая рассылка сообщений, процессы объединены с помощью дерева групповой рассылки, в корне которого располагается производитель сообщения (source based multicast distribution tree). В модели используется бинарное дерево. Однако принципиальных ограничений на моделирование дерева с заданной степенью ветвления (для всей топологии) в методе нет.

Типы сообщений. В модели используются сообщения типов: path, resv, data, path_teardown, resv_teardown, path_refresh, resv_refresh, path_error, resv_error. Необязательные сообщения типа path_conf, подтверждающие успешное резервирование, не используются.

Механизм контроля соединения. В протоколе RSVP описан механизм задания ограничений на качество сервиса с помощью фильтров. В каждом маршрутизаторе, через который проходит запрос на резервирование, проводится проверка возможности резервирования с помощью механизма контроля соединения. В модели не рассматриваются различные варианты контроля соединения. Предполагается, что каждому маршрутизатору всегда хватает ресурсов для выполнения запроса на резервирование.

Сбои. Описание протокола предусматривает возможность изменения топологии сети и реакции на отключение маршрутизаторов. В модели все маршрутизаторы и каналы считаются надёжными.

Другие абстракции. В модели не рассматривается механизм контроля доступа. Также не моделируются тайм-ауты.

Описание модели. Модель протокола реализована на языке TinyPromela. В модели участвуют процессы, порождаемые из описаний прототипов Sender (производитель), Router (маршрутизатор) и Receiver (потребитель). Указанные прототипы представлены размечеными системами переходов, содержащими 9408, 24 и 126 состояний соответственно.

Листинг. Сетевая грамматика модели RSVP (RSVPNet).

```

processes {
    r|parent , left , right | = Router ;
    p| child | = Sender ;
    c| parent | = Receiver ;
}

nonterminals {
    S||;
    N|parent , left , right |;
}

rules {
    S => N|c/parent| || p|c/child |;
    N => N|lc/parent| || N|rc/parent| || r|lc/left , rc/right |;
    N => c||;
}

```

На рис. 28 приведена архитектура подсистемы построения полублочной симуляции. По описанию прототипов процессов и описанию привязки процессов к каналам на подмножестве языка Promela порождается описание на промежуточном языке описания моделей PTY. По описанию моделей на языке PTY порождаются функции вычисления следующих состояний системы по предыдущему на языке C++. Сгенерированные функции собираются вместе с модулями построения полублочной симуляции, в результате чего создаётся исполняемый файл, настроенный на построение симуляции между двумя заданными моделями. Язык PTY служит низкоуровневым текстовым представлением LTS, в котором описываются непосредственно переходы процессов из одного состояния в другое.

Основные отличия языка PTY от MDL:

- в качестве механизма синхронизации используется синхронизация действий и ко-действий вместо разделяемых переменных;
- разрядность целочисленных переменных не указывается явно, а вычисляется по телу процессов, так как над целочисленными типами допустимы лишь операции присваивания и сравнения;
- наряду с целочисленными типами поддерживаются перечислимые типы.

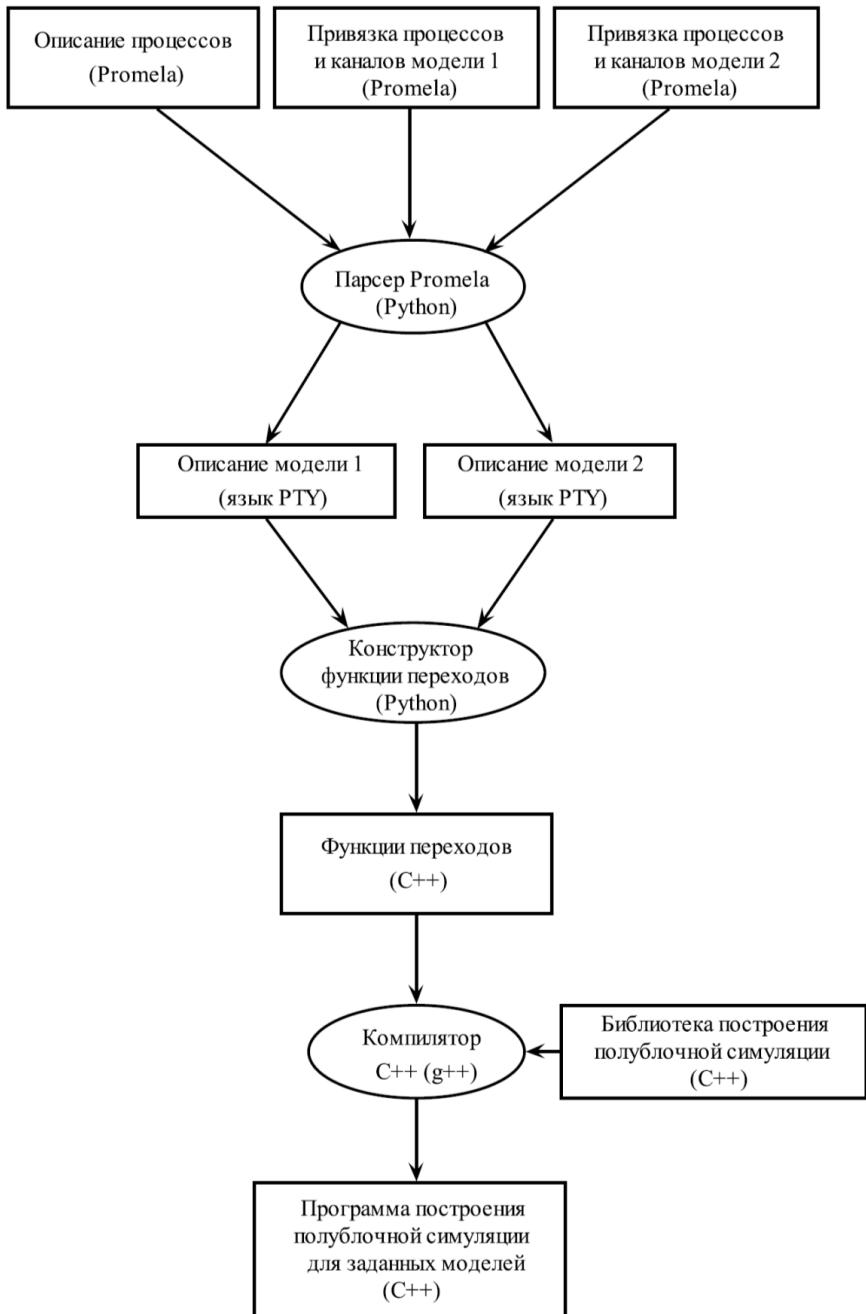


Рис. 28. Архитектура подсистемы проверки симуляции

На языке PTY LTS задаётся с помощью трёх секций: секции прототипов процессов, секции экземпляров процессов и секции связывания действий различных процессов.

В секции прототипов процессов для каждого прототипа описываются локальные переменные процесса и множество переходов вида: $\text{action: } x_1 = u_1, \dots, x_m = u_m \Rightarrow y_1 = v_1, \dots, y_n = v_n$, где action соответствует обозначению действия в LTS, пары $x_i = u_i$, $1 \leq i \leq m$, задают условие на значения переменных, при котором срабатывает переход, а пары $y_j = v_j$, $1 \leq j \leq n$, задают новые значения переменных в результате выполнения перехода.

В секции экземпляров процессов задаются процессы, порождаемые с помощью описаний в прототипах. Описание процесса выглядит следующим образом: $p1 = \text{new Proto}()$. Данная запись означает рождение процесса с именем $p1$ и телом, изоморфным LTS, заданной в прототипе Proto .

В секции связывания действий процессов задаются пары синхронизации действий экземпляров процессов вида: $p.a / q.b$, где p и q – имена экземпляров процессов, a и b – названия синхронизируемых действий.

Листинг. Состояние распределённой системы.

```
struct State {
    unsigned int p1State: <n1>; // состояние процесса P1
    ...
    unsigned int pKState: <nK>; // состояние процесса P_K
    int invalid: 1; // признак валидности состояния
};
```

Функция переходов системы задаётся с помощью функций переходов процессов. Для каждого процесса p_i задаются массивы, указанные в следующем листинге.

Листинг. Массивы описания переходов.

```
static int piSuccCnt[nSi][nAi];
static int piSucc[nSi][nAi]||maxa∈Ai(nTi(a))|;
```

Данное представление позволяет быстро перебирать возможные переходы из заданного состояния. При этом создаваемые массивы не потребляют критического объёма памяти, так как системы переходов

всех процессов малы, в то время как система переходов композиции процессов велика.

Переходы LTS распределённой системы вычисляются «на лету» с использованием массивов переходов процессов.

Листинг Б.6. Схема генерации функции построения переходов «на лету»

```
// Поиск следующего состояния по переходу с номером ind
// из состояния state по действию с номером action
State nextState(const State& state, int action, int ind) {
    State next = state; int cnt1, cnt2;
    switch(action) {
        // #для каждого процесса p
        // #для всех номеров а наблюдаемых действий процесса
        case <i>:
            cnt1 = pSuccCnt[state.pState][a];
            if (ind < cnt1) {
                next.pState = pSucc[state.pState][a][ind];
                return next;
            } else {
                next.invalid = 1;
                return next;
            }
        // #конец цикла
        // #конец цикла
        // #для каждого процесса p
        // #для всех номеров а ненаблюдаемых действий процесса
```

```

case 0:
    cnt1 = pSuccCnt[state.pState][a];
    if (ind < cnt1) {
        next.pState = pSucc[state.pState][a][ind];
        return next;
    } else {
        next.invalid = 1;
        return next;
    }
// #конец цикла
// #конец цикла

case 1: /* взаимодействие процессов */ {
    // #для каждой пары процессов p и q
    // #для всех номеров пар коммуникационных действий a, b
    // процессов p и q

    next = state;
    cnt1 = pSuccCnt[state.pState][a];
    cnt2 = qSuccCnt[state.qState][b];
    if (ind < cnt1 * cnt2) {
        next.pState = pSucc[state.pState][a][ind % cnt1];
        next.qState = qSucc[state.qState][b][ind / cnt1];
        return next;
    }
    if ((ind == cnt1 * cnt2) < 0) {
        next.invalid = 1;
        return next;
    }
}

```

```
// #конец цикла
// #конец цикла
} // switch
} // nextState
```

Аналогично функции nextState порождается функция prevState, вычисляющая состояния, из которых можно попасть в заданное состояние по заданному действию. Функция prevState используется для проверки пар состояний в отношении полублочной симуляции в том случае, когда пара состояний заносится в множество опровергнутых пар.

Благодаря использованному подходу нам удалось избежать предварительного построения и хранения в памяти графа переходов распределённой системы без заметной потери производительности.

В качестве хранилища пар состояний выбрали стандартный контейнер `std::set<State1, std::set<State2>>` (в дальнейшем это хранилище называется `std`). Однако с ростом числа процессов в моделях потребление памяти при построении отношения симуляции росло довольно быстро. При запуске тестов даже на 5 процессах со 100 состояниями объём используемой памяти превышал два гигабайта.

Из-за нехватки памяти при использовании хранилища `std` решили использовать одно из символьных представлений множества состояний.

Используется реализация `dfa` из исходных кодов Spin версии 4.2.7.

Для операций над `dfa` верны следующие оценки сложности:

- время выполнения операции вставки слова оценивается величиной $O(k \cdot |\Sigma_A|)$,
- время выполнения операции удаления слова так же оценивается величиной $O(k \cdot |\Sigma_A|)$,
- время выполнения операции проверки вхождения слова в язык автомата оценивается величиной $O(k)$.

Данное представление хорошо подходит для хранения множеств неподтверждённых и опровергнутых пар, т.к. проверка наличия состояния в множествах проводится чаще, чем вставка и удаление. При этом множество состояний довольно сильно сжимается, благодаря разметке рёбер интервалами целых чисел вместо значений. В используемой реализации нам не хватает операции перебора слов автомата

для организации цикла по неподтверждённым состояниям. Для реализации перебора слов автомата мы добавляем к слову дополнительный байт – признак чётности итерации. При переборе слов автомата на итерации i в языке автомата ищется произвольное слово $(i \bmod 2) \cdot w$ (с признаком чётности $i \bmod 2$). Если такое слово найдено, то слово $(i \bmod 2) \cdot w$ удаляется из языка автомата, добавляется слово $((i + 1) \bmod 2) \cdot w$ и выдаётся слово w .

Для быстрого поиска произвольного слова в автомате переходам приписывается флаг empty, информирующий о пустоте языка автомата по пройденному префикску.

Листинг. Перебор состояний в DFA.

```

int depth; // длина слова в байтах (k)
int iter = 0; // номер итерации
char *word; // массив, в который заносится новое слово
// начало новой итерации
void dfa_begin() { iter = (iter + 1) % 2; }
// получение следующего слова на текущей итерации
const char* dfa_next() {
    dfa_node* node = root; dfa_edge* edge = 0;
    // перейти к словам с префиксом iter % 2
    edge = root->first_edge;
    while (edge && edge->value != iter % 2)
        edge = edge->next;
    assert(edge);
    node = edge->next_node;
}

```

```

// найти произвольное слово из вершины node
for (int i = 1; i < depth; i++) {
    edge = node->first_edge;
    while (edge) {
        if (!edge->empty) {
            // существует хотя бы одно слово с
            // пройденным префиксом
            word[i] = edge->value;
            node = edge->next_node;
            break;
        }
        edge = edge->next;
    }
    if (!edge)
        // не существует слов с префиксом iter % 2,
        // перебор окончен
    return 0;
}
// перенести слово на следующую итерацию
dfa_remove(word);
word[0] = (iter + 1) % 2;
dfa_insert(word);
return word;
} // dfa_next

```

На каждом уровне автомата необходимо перебрать не более $|\Sigma_A|$ рёбер, поэтому сложность операции поиска произвольного слова с заданным префиксом оценивается величиной $O(k \cdot |\Sigma_A|)$, последующие операции удаления и вставки также занимают $O(k \cdot |\Sigma_A|)$. Сложность функции `dfa_next` равна $O(k \cdot |\Sigma_A|)$.

Более эффективным представлением оказалось представление dfa, потребляющее в экспериментах 20-30 мегабайт памяти в тех случаях, когда представление std требует более 2 гигабайт. Применение представления dfa позволило сократить и время вычисления на больших примерах.

3.2. Верификация вычислительных систем банка

В качестве примера предложенного метода верификации моделей разного уровня рассмотрена графовая модель абстрактного и структурного автоматов схемы пересчета, его схемная реализация и построение комплексного покрытия синтезированной схемы методом пересечения всех кубов вырожденных покрытий подсхем и методом пересечения с ограничениями. На основе комплексного кубического покрытия схемы был восстановлен граф переходов схемы и сделано заключение об изоморфизме графов и, следовательно, об их идентичности, что позволило сделать заключение о верификации моделей разного уровня.

Для моделей одного уровня абстракции предложен метод верификации с использованием комплексных покрытий на основании операций $\#$ и \cap кубов комплексного покрытия анализируемых схем. Рассмотрим применение условий необходимости и достаточности при верификации схемных решений методом моделирования. Пусть задана некоторая булева функция f своими покрытиями $C_1(f)$, при $f = 1$, и $C_0(f)$ при $f = 0$, которые построены произвольным методом, например по картам Карно. Требуется верифицировать некоторое схемное решение для f в виде логической схемы N (с внешним выходом ZN), спроектированной, например, эвристическим способом в произвольном базисе с применением методов факторизации, декомпозиции или их сочетания.

Необходимым условием верификации является совпадение реакций схемы N $ZN = 1$ ($ZN = 0$) для $\forall c \in C_1(f)$ ($\forall c \in C_0(f)$).

Достаточным условием верификации является совпадение реакций схемы $ZN = 0$ ($ZN = 1$) для $\forall c \in C_0(f)$ ($\forall c \in C_1(f)$).

Таким образом необходимым и достаточным условиями верификации схемы N является совпадение реакций схемы ZN со значе-

ниями булевой функции f для всех кубов $c \in C1(f) \cup C0(f)$. Комплексное покрытие $KP = C1(f) \cup C0(f)$ и есть метамодель схемы N , Верификацию схемы N можно осуществить и другим способом, отличным от метода моделирования.

Построим метамодель схемы для значений выхода ZN , равных 0 и 1, в виде комплексного покрытия $KP = C0(ZN) \cup C1(ZN)$. В этом случае требуется установить соответствие между покрытиями $C1(f) \cup C0(f)$ и $C0(ZN) \cup C1(ZN)$, которое может быть выполнено либо с использованием операции вычитания кубов покрытий ($\#$), либо с помощью операции пересечения кубов покрытий (\cap). При использовании операции вычитания для верификации схемы N необходимо, чтобы $C1(f) \# C1(ZN) = \emptyset$, и достаточно, чтобы $C1(ZN) \# C1(f) = \emptyset$. Аналогично для покрытий $C0(f)$ и $C0(ZN)$. Из этого следует, что при использовании операции вычитания достаточно иметь только один тип покрытий: либо единичное $C1(f)$ и $C1(ZN)$, либо нулевое $C0(f)$ и $C0(ZN)$.

Применение алгебро-топологических операций вычитания и пересечения покрытий позволяет при сравнении множеств избежать точного соответствия элементов множеств между собой, поэтому $C1(f) \# C1(ZN)$ и $C1(ZN) \# C1(f)$ не соответствуют аналогам (не топологическим) обычного вычитания множеств ($A \setminus B$ и $B \setminus A$).

Применив операции вычитания ($\#$) и пересечения (\cap) покрытий при верификации объектов получим четыре возможных отношения:

1. $C1(f) \# C1(ZN) = \emptyset$ и $C1(ZN) \# C1(f) = \emptyset$, или $C1(f) \cap C0(ZN) = \emptyset$ и $C0(f) \cap C1(ZN) = \emptyset$ - условия полной верификации.
2. $C1(f) \# C1(ZN) \neq \emptyset$ и $C1(ZN) \# C1(f) = \emptyset$, или $C1(f) \cap C0(ZN) \neq \emptyset$ и $C0(f) \cap C1(ZN) = \emptyset$ - есть необходимое, но недостаточное условие верификации.
3. $C1(f) \# C1(ZN) = \emptyset$ и $C1(ZN) \# C1(f) \neq \emptyset$, или $C1(f) \cap C0(ZN) = \emptyset$ и $C0(f) \cap C1(ZN) \neq \emptyset$ - есть достаточное, но не необходимое условие верификации.
4. $C1(f) \# C1(ZN) \neq \emptyset$ и $C1(ZN) \# C1(f) \neq \emptyset$, или $C1(f) \cap C0(ZN) \neq \emptyset$ и $C0(f) \cap C1(ZN) \neq \emptyset$ - условия верификации отсутствуют.

Во 2-ом и 3-ем случаях можно говорить о верификации частично определенных функций (объектов) с использованием «размытых» множеств, иначе говоря, один объект «делает» больше, чем другой. Полная верификация существует только при выполнении соотноше-

ний 1-го случая. Данное рассуждение проведено для случая одновыходной схемы N и исходного покрытия булевой функции f , которое является простейшим случаем метамодели высшего ранга.

Аналогичные методы можно применить при верификации граф-схем алгоритмов, конечных автоматов (абстрактных и структурных), многовыходных последовательностных схем и ПО.

На рис. 29 один и тот же вычислительный процесс реализован двумя различными способами. На схеме показано, что на основании технических заданий (ТЗ) или разработанных программ, строятся ГАМ и комплексные покрытия двух рассматриваемых вычислительных процессов. Из них методом алгебро-топологического вычитания исключаются значения, на которых функция не определена (don't care), и строятся контролирующие тесты.

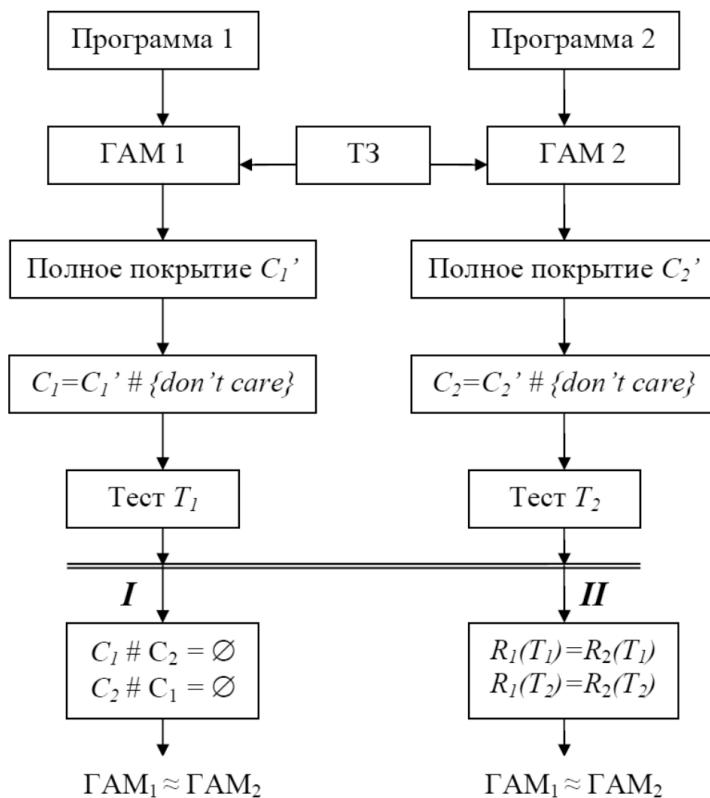


Рис. 29. Общая схема процесса верификации вычислительных процессов

Предложены два способа верификации этих процессов:

- метод алгебро-топологического вычитания покрытий каждого из каждого. При условии пустого значения результата делается заключение о эквивалентности данных вычислительных процессов;
- метод построения тестовых наборов по комплексным покрытиям путем пересечения кубов из интервальных частей покрытий и перекрестное тестирование, по результатам которого делается заключение о результатах верификации.

Рассмотрим пример верификации ациклического процесса. Пусть задана некоторая интервальная формула:

$$r = \begin{cases} FR1, & \text{при } x \leq k_1; \\ FR2, & \text{при } k_1 < x < k_2; \\ FR3, & \text{при } x \geq k_2, \end{cases}$$

реализующая вычисления некоторой переменной r по различным формулам: FR1, FR2 и FR3 произвольного вида в зависимости от двух булевых переменных, задающих некоторые условия-предикаты в виде неравенств: a: $x \leq k_1$ и b: $x \geq k_2$.

Переход от неравенств к булевым переменным при проектировании вычислительного процесса позволяет абстрагироваться от конкретного смысла неравенств и им соответствующих условий-предикатов и рассмотреть решение задачи верификации в общем виде.

ГАМ вычисления переменной r приведены на рис. 30. На рис. 30,а показана функциональная декомпозиция булевой функции $f = f(a,b)$ с начальной вершиной условия-предиката a (ГАМ 1), а на рис. 30,б – с начальной вершиной b (ГАМ 2).

Построим комплексные кубические покрытия $C_1(r)$ и $C_2(r)$ для вычисляемой переменной r по ГАМ 1 и ГАМ 2:

$$C_1(r) = \left\{ \begin{array}{c|ccccc|cc|c} z_1 & a & b & r & r' & z_2' & \{c\} \\ \hline 1 & 1 & \times & \times & /FR1/ & 1 & c_1 \\ 1 & 0 & 1 & \times & /FR3/ & 1 & c_2 \\ 1 & 0 & 0 & \times & /FR2/ & 1 & c_3 \\ 0 & \times & \times & p & p & 0 & c_4 \end{array} \right\}$$

$$C_2(r) = \left\{ \begin{array}{c|ccccc|cc|c} z_1 & a & b & r & r' & z_2' & \{c\} \\ \hline 1 & \times & 1 & \times & /FR3/ & 1 & c_1 \\ 1 & 0 & 0 & \times & /FR2/ & 1 & c_2 \\ 1 & 1 & 0 & \times & /FR1/ & 1 & c_3 \\ 0 & \times & \times & p & p & 0 & c_4 \end{array} \right\}$$

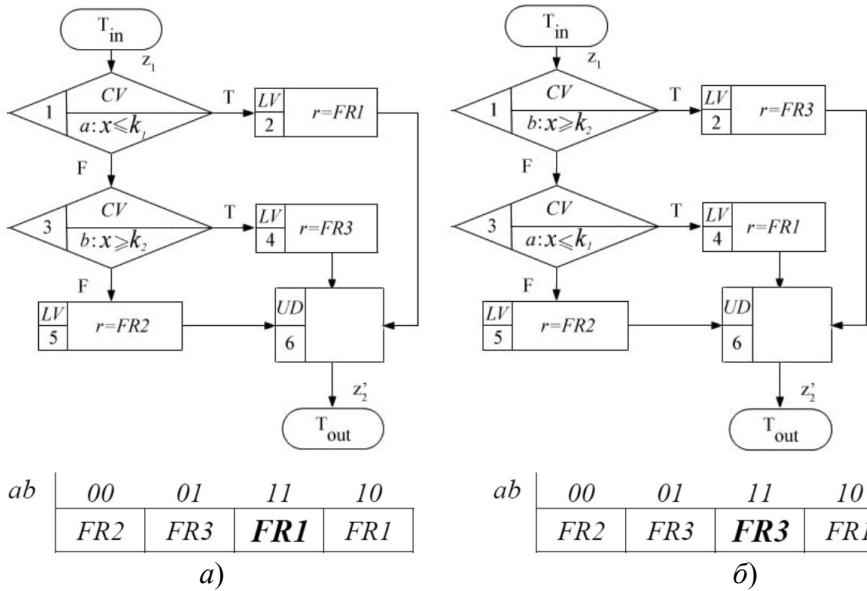


Рис. 30. ГАМ вычисления интервальнойной формулы

По покрытиям $C_1(r)$ и $C_2(r)$ построим тесты $T_1(r)$ и $T_2(r)$ путем пересечения кубов из интервальных частей покрытий $C_1(r)$ и $C_2(r)$, соответственно. Получим тесты:

$$T_1(r) = \left\{ \begin{array}{c|ccccc} z_1 & a & b & r & r' & z_2' \\ \hline 1 & 1' & 0 & \times & /FR1/ & 1 \\ 1 & 0' & 0 & \times & /FR2/ & 1 \\ 1 & 0 & 1' & \times & /FR3/ & 1 \\ 1 & 0 & 0' & \times & /FR2/ & 1 \\ 1' & 0 & 1 & p & /FR3/ & 1 \\ 0' & 0 & 1 & p & p & 0 \end{array} \right\} \{t\}$$

Примеч. $t_1 / t_1' \in c_1 \cap c_3$ из $C_1(r)$

| z_1 | a | b | r | r' | z_2' | $\{t\}$ |
|-------|-----|-----|-----|-------|--------|---------------|
| 1 | 0 | 1' | x | /FR3/ | 1 | t_1 |
| 1 | 0 | 0' | x | /FR2/ | 1 | \tilde{t}_1 |
| 1 | 1' | 0 | x | /FR1/ | 1 | t_2 |
| 1 | 0' | 0 | x | /FR2/ | 1 | \tilde{t}_2 |
| 1' | 1 | 0 | p | /FR1/ | 1 | t_3 |
| 0' | 0 | 1 | p | p | 0 | \tilde{t}_3 |

Примеч. $t_1 / \tilde{t}_1 \in c_1 \cap c_3$ из $C_2(r)$

Для формул должны выполняться условия: /FR1/ \neq /FR2/ \neq /FR3/ \neq p, т.е. вычисляемые и хранимые значения должны различаться на разных наборах теста. Штрихами в тестах отмечены значения активно изменяемых условий-предикатов.

С учетом удаления ab=11 верификация по покрытиям дает $C_1 \# C_2 = \emptyset$ и $C_2 \# C_1 = \emptyset$, что и свидетельствует об эквивалентности вычислительных процессов. Это наглядно можно наблюдать на картах Карно приведенных на рис. 30.

Перекрестное тестирование дает следующий результат: $R_1(T_1)=R_2(T_1)$ и $R_1(T_2)=R_2(T_2)$, что также подтверждает эквивалентность вычислительных процессов. Если переменную r вычислять по разным упрощенным формулам FR1, FR2 и FR3, тогда метод перекрестного тестирования является предпочтительным, так как не требует приведения выражений формул к каноническому виду.

Далее приводятся машинно-ориентированные алгоритмы реализации основных операций по вычислению комплексных кубических покрытий. Алгоритм построения комплексного кубического покрытия представлен на рис. 31.

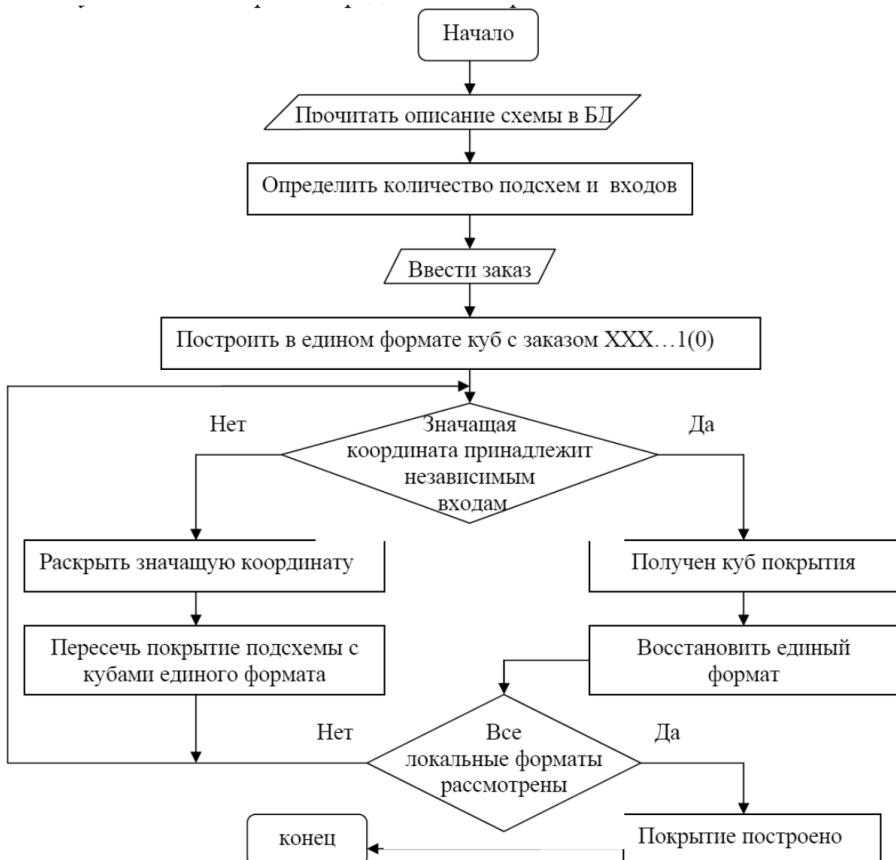


Рис. 31. Алгоритм построения комплексного кубического покрытия

3.3. Совершенствования верификации симуляций между моделями программ

Решается задача создания универсальной среды проверки симуляций между моделями программ, заданными конечными структурами Крипке.

Определение 2.1. Структура Крипке задаётся пятёркой $\langle S, S_0, R, \Sigma, L \rangle$, где S – конечное множество состояний, $S_0 \in S$ – начальное

состояние, $R \subseteq S \times S$ – отношение переходов, Σ – множество атомарных переменных, $L: S \rightarrow 2^\Sigma$ – функция разметки состояний.

Для задания спецификаций поведения программ наиболее часто используется темпоральная логика ветвящегося времени CTL^* и её фрагменты LTL , $ACTL^*$, CTL^*_X и $ACTL^*_X$. Эти логики имеют различные выразительные способности и области применения. Так, CTL^* и $ACTL^*$ позволяют описывать свойства деревьев вычислений, а LTL – свойства отдельных вычислений. В логиках CTL^*_X , $ACTL^*_X$ и LTL_X отсутствует оператор X . Данный оператор используется для задания свойств, которые должны выполняться в следующем состоянии вычисления. Поэтому он не применяется для описания свойств параллельных асинхронных систем, поскольку порядок выполнения процессов в таких системах недетерминирован.

Бинарное отношение \leq на множестве структур Кripке сохраняет выполнимость формул логики Λ , если для любой формулы $\varphi \in \Lambda$ и любых структур Кripке M_1 и M_2 таких, что $M_1 \leq M_2$, из $M_2 \models \varphi$ следует $M_1 \models \varphi$.

Отношения симуляции – это специальная разновидность бинарных отношений на множестве моделей программ; говоря неформально, модель M_2 симулирует модель M_1 , если дерево вычислений, порождённое моделью M_1 , подобно некоторому фрагменту дерева вычислений, порождённому моделью M_2 . Определяя по-разному подобие деревьев вычислений, можно получить большое разнообразие отношений симуляции (строгую симуляцию, прореженную симуляцию, ...). Наибольший интерес представляют отношения симуляции, сохраняющие выполнимость формул различных темпоральных логик.

Определение 2.2 (Строгая симуляция). Пусть даны две структуры Кripке M_1 и M_2 , где $M_i = \langle S_i, s_{i0}, R_i, \Sigma_i, L_i \rangle$, $i = 1, 2$, и $\Sigma_1 = \Sigma_2$. Будем говорить, что между M_1 и M_2 выполняется строгая симуляция (обозначается $M_1 \leq M_2$), если существует такое $H \subseteq S_1 \times S_2$, что $(s_{10}, s_{20}) \in H$ и для любой пары $(s_1, s_2) \in H$ выполняется:

$$1. L_1(s_1) = L_2(s_2).$$

2. Если $(s_1, s'_1) \in R_1$, то существует такое состояние $s'_2 \in S_2$, что $(s_2, s'_2) \in R_2$ и $(s'_1, s'_2) \in H$.

Будем говорить, что между M_1 и M_2 выполняется строгая бисимуляция, если в определении 2.2 отношение H является симметричным (т. е. $H = H^{-1}$).

Определение прореженной симуляции отличается от определения строгой симуляции тем, что каждому переходу $(s_1, s'_1) \in R_1$ может соответствовать последовательность переходов t_0, t_1, \dots, t_n ($n \geq 0$) в R_2 , такая, что $t_0 = s_2, (s'_1, t_n) \in H$ и для всех $i < n$ выполняется $(t_i, t_{i+1}) \in R_2$ и $(s_1, t_i) \in H$.

Если добавить к этому определению требование симметричности, то мы придём к определению прореженной бисимуляции.

Строгая симуляция сохраняет выполнимость формул $ACTL^*$, строгая бисимуляция — выполнимость формул CTL^* , прореженная симуляция — выполнимость формул $ACTL^*_x$, прореженная бисимуляция — выполнимость формул CTL^*_x .

Здесь преследуется цель разработки универсальной программноинструментальной среды проверки симуляций. Разрабатываемая среда должна состоять из двух компонент: формального языка, позволяющего описывать различные симуляции, и программноинструментального средства, которое по двум размеченным системам переходов и описанию симуляции проверяет, выполняется ли заданная симуляция между этими системами переходов.

В основу универсального средства описания и проверки симуляций целесообразно положить теоретико-игровой подход. Этот подход годится для проверки всех известных отношений симуляции, и более того, во многих случаях его использование даёт оптимальные по сложности алгоритмы проверки симуляций.

В теоретико-игровом подходе задача проверки симуляции сводится к задаче нахождения выигрышной стратегии в антагонистической игре двух игроков Spoiler и Duplicator (или для краткости S и D).

Формально такая игра задаётся пятёркой $\langle V_S, V_D, E_S, E_D, v_0 \rangle$, где V_S и V_D — множества состояний игроков Spoiler и Duplicator соответственно, $E_S \subseteq V_S \times V_D$ и $E_D \subseteq V_D \times V_S$ — множества допустимых ходов, $v_0 \in V_S$ — начальное состояние. История игры — это такая конечная или бесконечная последовательность игровых состояний (v_0, v_1, v_2, \dots) , что $(v_i, v_{i+1}) \in E_S \cup E_D$ для любого $i \geq 0$. Стратегией игрока Duplicator — это функция $W : V_D \rightarrow V_S \cup \{halt\}$.

История игры (v_0, v_1, v_2, \dots) удовлетворяет стратегии W игрока Duplicator, если для любого состояния $v_i \in V_D$ из истории игры, за исключением, возможно, последнего, выполняется $v_{i+1} = W(v_i)$, а для последнего состояния (если история игры конечна) выполняется $W(v_i) = halt$. Игрок Duplicator выигрывает в игре, если у него существует стратегия, для которой любая удовлетворяющая ей история

игры либо является бесконечной, либо завершается состоянием игрока Spoiler. Игры проверки симуляции проектируются таким образом, что выполнимость отношения симуляции равносильна существованию выигрышной стратегии игрока Duplicator в соответствующей игре проверки симуляции.

Различные варианты игр для проверки многих отношений симуляции имеют много общего. В них всегда участвуют два игрока и условия выигрыша совпадают (один игрок выигрывает, если другой игрок не может сделать ход). Игровые состояния представляются кортежами, компонентами которых являются компоненты моделей. Множества допустимых ходов могут быть описаны булевыми формулами над фиксированной сигнатурой. Это наблюдение было использовано для создания теоретико-игрового языка задания симуляций. Определение некоторого отношения на этом языке представляет собой описание правил игры для проверки этого отношения и включает в себя описание устройства игровых состояний и правил, по которым игроки совершают свои ходы.

Самый простой вид имеет определение правил игры проверки строгой симуляции. Предположим, что необходимо проверить симуляцию между M_1 и M_2 , где $M_i = \langle S_i, s_{i0}, R_{i,i}, L_i \rangle$. Состояния игроков Spoiler и Duplicator в этой игре представляются кортежами, состоящими из одного состояния из S_1 и одного состояния из S_2 , т. е. $V_S = V_D = \{(s_1, s_2) | s_1 \in S_1 \wedge s_2 \in S_2\}$. Если состояния игрока P описывается парой значений типизированных переменных x_{1P} и x_{2P} (x_{iP} принимает значения из S_i), то допустимые переходы игроков описываются формулами

$$\varphi_S(x_S^1, x_S^2, x_D^1, x_D^2) \equiv (x_S^2 = x_D^2) \wedge R_1(x_S^1, x_D^1)$$

$$\varphi_D(x_D^1, x_D^2, x_S^1, x_S^2) \equiv (x_S^1 = x_D^1) \wedge R_2(x_S^2, x_D^2) \wedge (L_1(x_D^1) = L_2(x_D^2))$$

Поэтому множества ходов будут иметь следующий вид:

$E_S = \{((s_1, s_2), (s'_1, s'_2)) | (s_1, s'_1) \in R_1\}$ и $E_D = \{((s_1, s_2), (s_1, s'_2)) | (s_1, s'_2) \in R_2 \wedge L(s_1) = L(s'_2)\}$. Начальное игровое состояние полагается равным $v_0 = (s_{10}, s_{20})$.

Всего на разработанном языке были записаны правила проверки следующий отношений: строгой симуляции и бисимуляции, прореженной симуляции и бисимуляции, слабой симуляции (сохраняющей выполнимость формул LTL_x), k_2 симуляции (сохраняющей выполнимость формул LTL). В работе доказана корректность игр для проверки

прореженной симуляции и бисимуляции, корректность игр для проверки остальных симуляций была доказана ранее в литературе.

Алгоритм проверки симуляций реализован по двум конечным структурам Кripке и по отношению симуляции, заданном на разработанном языке, проверяет выполнимость этого отношения между этими моделями. В алгоритме последовательно проводится построение множества достижимых состояний игры, построение множества выигрышных для игрока Spoiler состояний и проверка начального состояния на принадлежность этому множеству. Разработанный алгоритм является символьным, т.е. для повышения эффективности в нём проводятся операции над формулами, характеризующими множества, а не над отдельными элементами этих множеств. Доказана корректность приведённого алгоритма.

Разработано универсальное программно-инструментальное средство проверки симуляций между парами структур Кripке. Данное средство реализовано на языке Python. В качестве языка задания моделей используется язык широко используемого верификатора NuSMV. Для представления формул в символьном алгоритме были использованы упорядоченные двоичные разрешающие диаграммы (OBDD). В равной мере для символьного представления моделей и игры могут быть использованы и другие математические конструкции: регулярные выражения, прямое формульное представление и др. OBDD были выбраны ввиду широкой распространённости библиотек для манипуляции с ними. В качестве библиотеки работы с OBDD используется библиотека Cudd.

В инкрементальном методе последовательно строятся модели M_1, \dots, M_n так, что каждая следующая модель является уточнением предыдущей модели и содержит больше подробностей о конструируемой программе. Такой способ построения помогает избежать ошибок при построении моделей. Чтобы убедиться, что M_{i+1} уточняет M_i , достаточно проверить, что между этими моделями выполняется некоторое отношение симуляции (которое выбирается исходя из характера отличий между M_{i+1} и M_i). Если же это отношение симуляции не выполняется, то контрпример, выданный средством проверки симуляций, поможет найти ошибку в модели M_{i+1} .

Инкрементальный подход был применён для решения задачи справедливого разделения ресурсов в системе асинхронных взаимодействующих процессов (задача “обедающих философов”). Всего были построены три модели. В модели M_1 не накладывалось никаких

ограничений на действия процессов (философов). В модели M_2 появляется маркер; каждый философ может получить маркер только когда вилки слева и справа от него лежат на столе. После того, как философ получил маркер, он может взять вилки и начать есть; после того как он закончил есть, он освобождает маркер. В окончательной модели M_3 маркер движется строго по кругу, и каждый философ обязан закончить есть прежде, чем он передаст маркер своему соседу.

Исходя из характера отличий между моделями был сделан вывод о том, что между M_2 и M_1 должно выполняться отношение строгой симуляции, а между M_3 и M_2 – отношение прореженной симуляции. Справедливость этих отношений была проверена при помощи разработанного средства для систем с различным количеством обедающих философов.

Далее решается задача разработки алгоритма проверки симуляции между временными игровыми автоматами.

Будем использовать запись $\mathcal{B}(X)$ для обозначения множества формул над множеством X , удовлетворяющих грамматике $\phi ::= x \sim k \mid \varphi \wedge \psi$, где $k \in \mathbb{Z}_{\geq 0}$, $x \in X$ и $\sim \in \{<, \leq, =, >, \geq\}$. Будем использовать запись X^Y для обозначения множества всех отображений из множества X в множество Y (это расходится с традиционной записью Y^X). Оценка переменных из X – это элемент $Rx \geq 0$. Пусть $\delta \in \mathbb{R}_{\geq 0}$, тогда будем обозначать $v + \delta$ такую оценку, что для всех $x \in X$ выполняется $(v + \delta)(x) = v(x) + \delta$. Пусть $Y \subseteq X$, тогда будем использовать запись $v[Y]$ для обозначения оценки, сопоставляющей 0 всем $x \in Y$, и $v(x)$ всем $x \in X \setminus Y$.

Определение 3.1. Временной автомат (timed automata, TA) – это шестёрка $\langle S, s_0, X, \Sigma, R, \text{Inv} \rangle$, где S – конечное множество дискретных состояний, $s_0 \in S$ – начальное дискретное состояние, X – конечное множество вещественных переменных (таймеров), Σ – множество пометок переходов (действий), включающее в себя так называемое невидимое действие τ , $R \subseteq S \times \mathcal{B}(X) \times \Sigma \times \wp(X) \times S$ – конечное множество переходов ($\wp(X)$ – множество всех подмножеств X), $\text{Inv} : S \rightarrow \mathcal{B}(X)$ – функция, сопоставляющая дискретным состояниям формулы (такие формулы называются инвариантами).

Семантика временного автомата определяется размеченной системой переходов (Q, q_0, \rightarrow) , где $Q = S \times \mathbb{R}_{\geq 0}$, $q_0 = (s_0, \vec{0})$. Из состояния q_1 существует дискретный переход по действию a в состояние q_2 (обозначается $q_1 \xrightarrow{a} q_2$), если существует такой переход $e = (s, g, a,$

$(Y, s') \in R$, что $v |= g$, $v' = v[Y]$ и $v' |= \text{Inv}(s')$. Из состояния $q_1 = (s, v)$ существует задержка по времени δ в состояние $q_2 = (s, v')$ (обозначается $q_1 \xrightarrow{\delta} q_2$), если $v' = v + \delta$ и $v' |= \text{Inv}(s)$.

Для простоты будем полагать, что временные автоматы детерминированы по действиям, т. е. для любой пары $q \in Q$ и $a \in \Sigma$ существует не более одного такого q' , что $q \xrightarrow{a} q'$. Временной игровой автомат (timed game automata, TGA) – это временной автомат, в котором множество действий Σ разбито на множество контролируемых (Σ^c) и множество неконтролируемых (Σ^u) переходов. Стратегия с нулевой памятью контроллера (соответственно, среды) – это функция f , определённая на множестве состояний Q , и возвращающая некоторый элемент множества $(R \geq 0 \times \Sigma^c)$ (соответственно, $(R \geq 0 \times \Sigma^u)$).

Определение 3.5. Пусть дан TGA $\langle S, l_0, \Sigma, X, R, \text{Inv} \rangle$, и пусть (δ^c, a^c) и (δ^u, a^u) – некоторые стратегии контроллера и среды соответственно. Вычисление

$$q_0 \xrightarrow{\delta_0} q_2 \xrightarrow{a_0} q_3 \xrightarrow{\delta_1} q_4 \xrightarrow{a_1} q_5 \dots$$

будем называть порождённым этими стратегиями, если для любого $i \in \mathbb{N}$ выполняется $\delta_i = \min(\delta^c(q_{2i}), \delta^u(q_{2i}))$ и:

$$a_i = \begin{cases} a^u(q_{2i}) & \text{если } \delta^u(q_{2i}) \leq \delta^c(q_{2i}) \\ a^c(q_{2i}) & \text{иначе} \end{cases}$$

Аналогично определяются понятие стратегии для недетерминированных по действиям временных игровых автоматов, в этом случае стратегия будет возвращать не пометки, а переходы.

Определена логика ATCTL, используемая для описания свойств временных игровых автоматов. Эта логика состоит из формул вида $A\sigma_1 U_t \sigma_2$ и $A\sigma_1 W_t \sigma_2$, где $t \in \mathbb{Z} \geq 0 \cup \{+\infty\}$, а σ_1 и σ_2 – некоторые множества видимых действий. Будем говорить, что вычисление π временного автомата M удовлетворяет формуле $\sigma_1 U_t \sigma_2$, если существует такой префикс этого вычисления π' , что все видимые действия π' лежат в множестве σ_1 , последнее действие π' лежит в σ_2 и длительность π' не превышает t . Вычисление π временного автомата M удовлетворяет формуле $\sigma_1 W_t \sigma_2$, если это вычисление удовлетворяет формуле $\sigma_1 U_t \sigma_2$, или все видимые действия π принадлежат множеству σ_1 . Будем говорить, что формула $A\varphi$ логики ATCTL выполняется на TGA M , если существует такая стратегия контроллера f_c , что для любой стратегии

среды f_u формула φ выполняется на вычислении, порождённом стратегиями f_c и f_u .

Задача, которая решается в данной главе, заключается в определении отношения \preceq , сохраняющего выполнимость формул логики $ATCTL$, и разработке алгоритма проверки этого отношения. На основании обзора сделан вывод о том, что в качестве искомого отношения следует использовать “комбинацию” определений временной, игровой и ослабленной отношений симуляции, описанных ранее в литературе.

Определение 3.7 (twa-симуляция). Пусть даны два TGA M_1 и M_2 с множествами состояний Q_1 и Q_2 , где $M_i = \langle S_i, s_{i0}, X_i, \Sigma_i, R_i, \text{Inv}_i \rangle$. Будем говорить, что между M_1 и M_2 выполняется отношение слабой альтернирующей временной (timed weak alternating, twa) симуляции, если существует такое отношение $H \subseteq Q_1 \times Q_2$, что пара начальных состояний (q_{10}, q_{20}) находится в этом отношении и для любой пары $(q_1, q_2) \in H$ и для любых $q'_1 \in Q_1$, $q'_2 \in Q_2$, $\delta \in \mathbb{R} \geq 0$, $a \in \Sigma \setminus \{\tau\}$ выполняются следующие требования:

1. Если $(q_2 \xrightarrow{a} c q'_2)$, то существует такое $n \geq 1$ и такая последовательность состояний $q'_{1,1}, q'_{1,2}, \dots, q'_{1,n}$ автомата M_1 , что $q'_{1,1} = q_1$, $q'_{1,n-1} \xrightarrow{a} c q'_{1,n}$, $(q'_{1,n}, q'_2) \in H$ и при этом для всех $i = 2..n-1$ выполняется $(q'_{1,i}, q_2) \in H$ и $q'_{1,i-1} \xrightarrow{\tau} c q'_{1,i}$.
2. Если $(q_1 \xrightarrow{a} u q'_1)$, то существует такое $n \geq 1$ и такая последовательность состояний $q'_{2,1}, q'_{2,2}, \dots, q'_{2,n}$ автомата M_2 , что $q'_{2,1} = q_2$, $q'_{2,n-1} \xrightarrow{a} u q'_{2,n}$, $(q'_1, q'_{2,n}) \in H$ и при этом для всех $i = 2..n-1$ выполняется $q'_{1,i-1} \xrightarrow{\tau} u q'_{1,i}$.
3. Если $q_2 \xrightarrow{\tau} c q'_2$, то $(q_1, q'_2) \in H$.
4. Если $q_1 \xrightarrow{\tau} u q'_1$, то $(q'_1, q_2) \in H$.
5. Если $q_2 \xrightarrow{\delta} q'_2$, то существует такое $q''_1 \in Q_1$, что $q_1 \xrightarrow{\delta} q''_1$ и $(q''_1, q'_2) \in H$

Ранее доказано, что введённое отношение twa-симуляции сохраняет выполнимость формул логики $ATCTL$.

Разработан теоретико-игровой алгоритм проверки твасимуляции и доказана его корректность. Поскольку в общем случае множество состояний временных автоматов бесконечно, то бесконечна и игра проверки симуляции. Поэтому в разработанном алгоритме ведётся работа не над отдельными игровыми состояниями, а над формулами, описывающими потенциально бесконечные множества состояний. Для операций над такими формулами используется аппарат матриц ограниченных разностей (difference bounded matrixes).

Осуществлено экспериментальное исследование реализации разработанного алгоритма твасимуляции для решения задачи синтеза алгоритма работы контроллера в системе автоматического управления.

Рассмотрим следующий пример. Пусть имеется движущийся конвейер, на который ставится изделие. Затем изделие движется по конвейеру и последовательно проходит N этапов обработки, каждый этап занимает от 1 до 2 единиц времени. Задача контроллера – после завершения последнего этапа переместить готовое изделие с конвейера в коробку. Если этого не произойдёт в течение $2N + 2$ единиц времени с начала работы конвейера, то изделие будет потеряно, и, таким образом, задача контроллера не будет выполнена. Данная система описывается ТГА S, изображённом на рис. 32 слева. Сплошными дугами обозначены контролируемые переходы, а прерывистыми – неуправляемые.

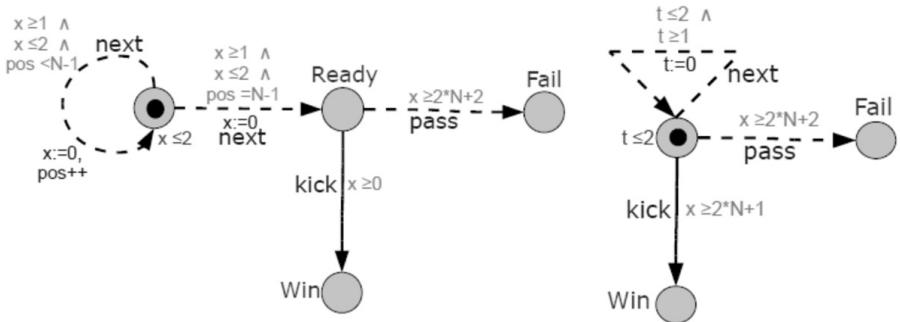


Рис. 32. Временные автоматы, на которых проводилось экспериментальное исследование разработанного алгоритма

Задача контроллера в автомате S – перевести автомат в дискретное состояние *Win*, т. е. обеспечить выполнимость формулы

$A\{next\}U\{kick\}$. Одна из возможных выигрышных стратегий контроллера f_c – подождать, пока автомат перейдёт в дискретное состояние *Ready*, а затем выполнить переход в состояние *Win*. Предположим теперь, что в реальной ситуации контроллеру неизвестно текущее положение изделия на конвейере. Тогда стратегия f^c не может использоваться в моделируемой (реальной) ситуации.

Поэтому была построена другая модель А, изображённая на рис. 13 справа. Эта модель уже не содержит невидимую контроллеру информацию, и для неё у контроллера существует выигрышная стратегия (её можно автоматически построить при помощи средства UPPAAL Tiga). Для того, чтобы проверить, что та же стратегия будет выигрышной и для исходной модели, достаточно проверить существование twa-симуляции между S и А. Корректность такого сведения была доказана в теореме. Существование twa-симуляции между S и А было проверено для нескольких значений N .

Взаимное исключение с помощью мьютекса. Мьютекс (одноместный семафор) является одним из наиболее часто используемых на практике примитивов синхронизации, позволяющим обеспечивать монопольный доступ клиентов к разделяемому ресурсу. Приводится представленное ниже описание модели мьютекса на языке Promela.

```

1: mtype = { p, v }           // пользовательский тип данных
2: chan sema = [0] of { mtype } // канал для обмена сообщениями типа mtype
3: active proctype Semaphore() { // мьютекс
4:     do
5:         :: sema!p -> sema?v;      // предоставление доступа
6:     od
7: }
8: active [3] proctype User() { // три клиентских процесса
9:     do
10:        :: sema?p; skip; sema!v;   // получение доступа
11:    od
12: }
```

Пусть требуется проверить, позволяет ли данная структура гарантировать каждому клиенту доступ к разделяемому ресурсу. Более формально: необходимо установить выполнимость свойства «каждый клиент рано или поздно получит доступ к ресурсу, защищенному мьютексом».

Предложенная задача может быть решена в ИС SPIN путем проверки достижимости циклов без прогресса. Для верификации использовалась следующая последовательность команд.

- 1: spin -a ds.pml
- 2: gcc -DNCORE=4 -DNP -o pan pan.c
- 3: pan -l > log.txt
- 4: spin -p -t ds.pml > trace.txt
- 5: del pan.* ds.pml.trail

Результат верификации представлен на рис. 33.

```

edit log.txt - Far 3.0.4000 x86 Administrator
E:\dissertation\spin\log.txt
pan:1: non-progress cycle (at depth 3)
pan: wrote ds.pml.trail
1251 Ln 32/32 Cc

(Spin Version 6.3.2 -- 17 May 2014)
Warning: Search not completed
+ Partial Order Reduction

Full statespace search for:
never claim          + (:np_...)
assertion violations + (if within scope of claim)
non-progress cycles  + (fairness disabled)
invalid end states   - (disabled by never claim)

State-vector 36 byte, depth reached 10, errors: 1
  4 states, stored
  0 states, matched
  4 transitions (= stored+matched)
  0 atomic steps

hash conflicts:      0 (resolved)

Stats on memory usage (in Megabytes):
  0.000 equivalent memory usage for states (stored*(State-vector + overhead))
  0.292 actual memory usage for states
  64.000 memory used for hash table (-w24)
  0.343 memory used for DFS stack (-m10000)
  64.539 total actual memory usage

pan: elapsed time 0.031 seconds
pan: rate      200 states/second

```

Указание на обнаружение ошибки

Время верификации

Рис. 33. Результат верификации в ИТС SPIN

По описанию на языке Promela была построена эквивалентная структура Кripke, изображенная на рис. 34. Атомарным предикатом *a* отмечено состояние, в котором доступ к ресурсу получает третий процесс.

Желаемое поведение описывается LTL-формулой *Fa* (когда-нибудь в будущем третий клиентский процесс обязательно получит доступ к ресурсу). Невыполнение спецификации будет означать, что существует как минимум одна последовательность состояний, на которой требуемое событие не происходит. И если такая траектория существует, то, следовательно, и существуют аналогичные траектории, на которых доступ к ресурсу не получают первый и второй процессы.

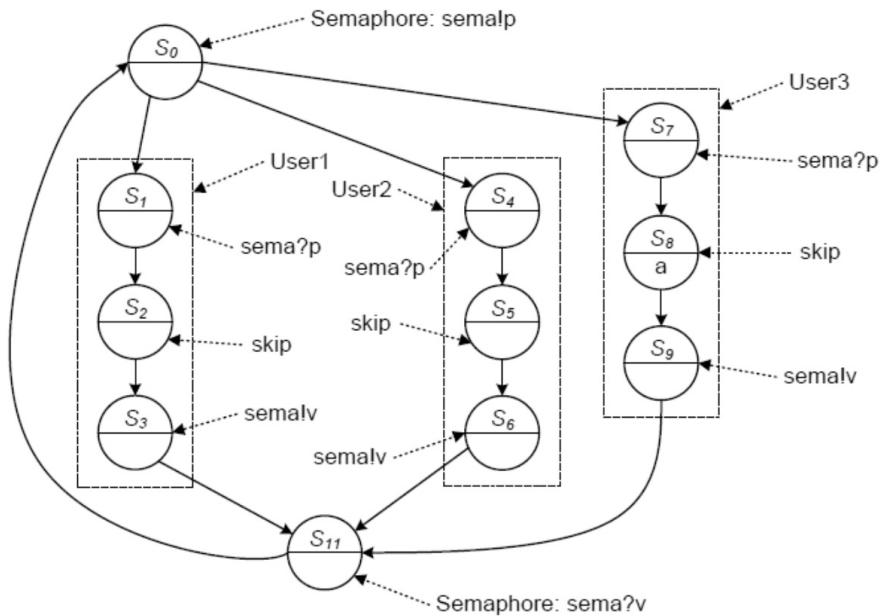


Рис. 34. Эквивалентная описание структура Крипке

Сформированная на основе структуры Крипке и спецификации база знаний подавалась на вход программной реализации МЛВ. Полученные результаты представлены на рис. 35.

Применение как ИС SPIN, так и предлагаемого в работе подхода показали несоответствие модели и спецификации. Таким образом, может быть сделан вывод о том, что исследуемая реализация мютекса не позволяет гарантировать клиентам доступ к ресурсу. Для устранения ошибки необходимо ввести дополнительную FIFO-подобную структуру для размещения запросов от клиентских процессов.

Алгоритм работы сетевого генератора данных. Используется модель алгоритма, циклически генерирующего через фиксированный временной промежуток сообщение одного из двух возможных типов и отправляющего данное сообщение получателю через буферизированный канал.

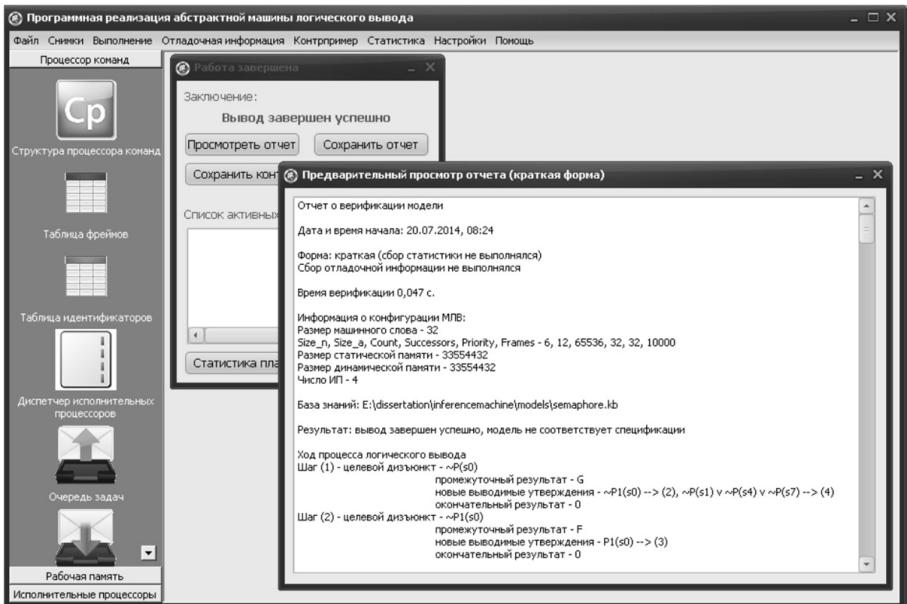


Рис. 35. Результат верификации с помощью МЛВ

Описание модели на языке Promela представлено ниже.

```

1: #define N 512                                // размер буфера
2: bool x = false;
3: init {
4:     chan dummy = [N] of { byte };           // канал для передачи сообщений
5:     do
6:         :: dummy!85                         // передача сообщения первого типа
7:         :: dummy!170; x = true;             // передача сообщения второго типа
8:     od
9: }
```

Нежелательным поведением генератора является ситуация, при которой канал оказывается заполнен сообщениями первого типа. То есть, корректное поведение предполагает, что рано или поздно генератор отправит сообщение второго типа, а переменная x примет истинное значение. Для верификации в ИС SPIN применялась следующая последовательность команд.

- 1: spin -f “!(\lozenge (x == true))” > ltl.nvr
- 2: spin -m -a -N ltl.nvr gen.pml
- 3: gcc -DNCORE=4 -o pan pan.c
- 4: pan -a -n > log.txt
- 5: spin -p -t gen.pml > trace.txt
- 6: del pan.* ltl.nvr gen.pml.trail

Результат верификации представлен на рис. 36.

```

edit log.txt - Far 3.0.4000 x86 Administrator
E:\dissertation\spin\log.txt * 1251 Ln 34/34
warning: for p.o. reduction to be valid the never claim must be stutter-invariant
(Never claims generated From LTL formulae are stutter-invariant)
pan:1: acceptance cycle (at depth 1024)
pan: wrote gen.pml.trail

(Spin Version 6.3.2 -- 17 May 2014)
Warning: Search not completed
+ Partial Order Reduction

Full statespace search for:
    never claim      + (never_0)
    assertion violations + (if within scope of claim)
    acceptance cycles   + (fairness disabled)
    invalid end states - (disabled by never claim)

State-vector 536 byte, depth reached 1025, errors: 1
  513 states, stored
    0 states, matched
    513 transitions (= stored+matched)
      0 atomic steps
hash conflicts:          0 (resolved)

Stats on memory usage (in Megabytes):
  0.270   equivalent memory usage for states (stored*(State-vector + overhead))
  0.487   actual memory usage for states
  64.000  memory used for hash table (-w24)
  0.343  memory used for DFS stack (-m10000)
  64.734  total actual memory usage

pan: elapsed time 0.587 seconds
pan: rate 912.81139 states/second

```

Рис. 36. Результат верификации в ИТС SPIN

Эквивалентная описанию на языке Promela структура Кripке изображена на рис. 37.

Как и в предыдущем примере, корректное поведение описывается LTL-формулой Fx . Сформированная на основе структуры Крипке и спецификации база знаний подавалась на вход программной реализации МЛВ.

Полученные результаты представлены на рис. 38.

ИТС SPIN и предлагаемый логический подход показали, что модель не соответствует спецификации.

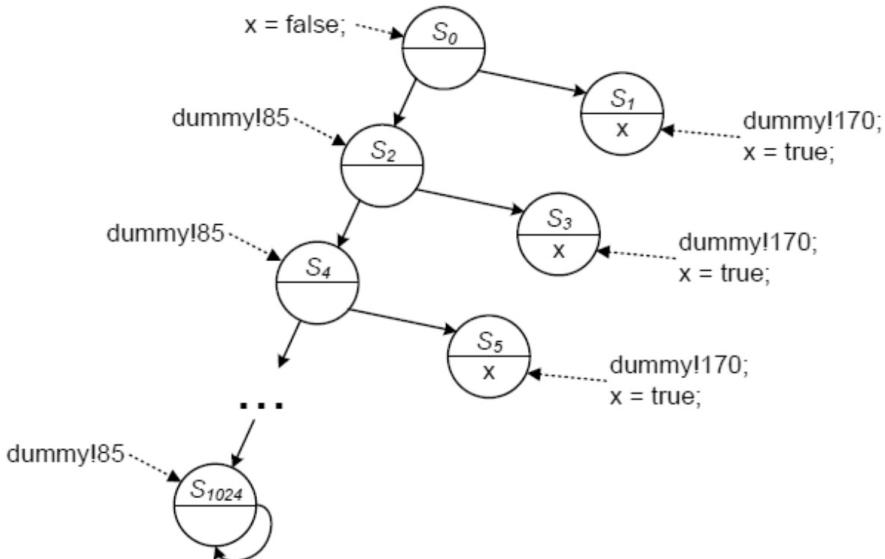


Рис. 37. Эквивалентная описание структура Крипке

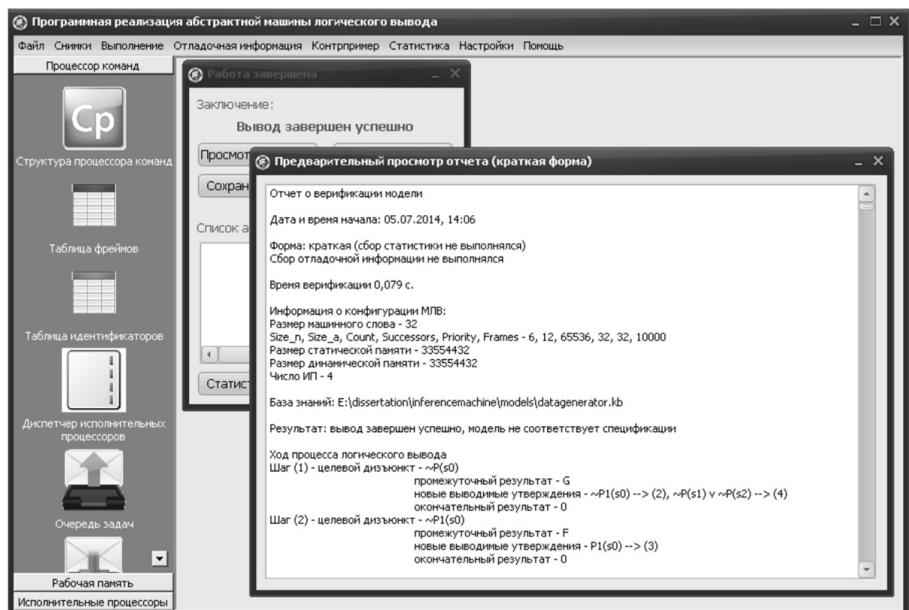


Рис. 38. Результат верификации с помощью МЛВ

Алгоритм планирования sleep-wakeip. В работе рассматриваются реализации нескольких процедур ядра распределенной операционной системы UTS. Предложено представленное ниже описание модели алгоритма работы планировщика этой операционной системы.

```

1: mtype = { Wakeme, Running }           // пользовательский тип данных
2: bit lk, sleep_q, r_lock, r_want;
3: mtype State = Running;                // текущее состояние клиентского процесса
4:
5: active proctype Client() {            // клиентский процесс
6:   sleep:
7:     atomic { (lk == 0) -> lk = 1 };    // захват спинлока
8:     do
9:       :: (r_lock == 1) ->              // проверка условия выполнения задачи
10:      r_want = 1;                     // запрос на пробуждение
11:      State = Wakeme;               // смена состояния клиентского процесса
12:      lk = 0;                      // освобождение спинлока
13:      (State == Running);          // ожидание смены состояния
14:    :: else -> break
15:    od
16:   progress:
17:     assert(r_lock == 0);             // проверка условия корректности
18:     r_lock = 1;                   // отметка о выполнении задачи
19:     lk = 0;                      // освобождение спинлока
20:     goto sleep
21: }
22:
23: active proctype Server() {           // управляющий процесс
24:   wakeup:
25:     r_lock = 0;
26:     (lk == 0);                  // ожидание освобождения спинлока
27:     if
28:       :: r_want ->              // если требуется смена состояния
29:         // клиентского процесса
30:         atomic { (sleep_q == 0) -> sleep_q = 1 }; // захват спинлока
31:         r_want = 0;                // запрос на смену состояния обработан
32:         (lk == 0);                // ожидание освобождения спинлока
33:         if
34:           :: (State == Wakeme) -> State = Running; // смена состояния
35:           :: else ->
36:         fi;

```

```

37:     sleep_q = 0           // освобождение спинлока
38:     :: else ->
39:     fi;
40:     goto wakeup
41: }

```

Одним из основных требований, предъявляемых к планировщику, является недостижимость ситуации голодания, при которой клиентский процесс всегда будет находиться в состоянии *Wakeme*. Для верификации в ИС SPIN применялась следующая последовательность команд, рис. 39.

- 1: spin -f "< ([] (State == Wakeme))" > ltl.nvr
- 2: spin -a -N ltl.nvr sw.pml
- 3: gcc -DNCORE=4 -o pan pan.c
- 4: pan -a -n > log.txt
- 5: spin -p -t sw.pml > trace.txt
- 6: del pan.* ltl.nvr sw.pml.trail

```

edit log.txt - Far 3.0.4000 x86 Administrator
E:\dissertation\spin\log.txt          1251 Ln      34/34
warning: for p.o. reduction to be valid the never claim must be stutter-invariant
(Never claims generated from LTL formulae are stutter-invariant)
pan:1: acceptance cycle (at depth 80)
pan: wrote sw.pml.trail

(Spin Version 6.3.2 -- 17 May 2014)
Warning: Search not completed
+ Partial Order Reduction

Full statespace search for:
    never claim      + (never_0)
    assertion violations + (if within scope of claim)
    acceptance cycles   + (fairness disabled)
    invalid end states - (disabled by never claim)

State-vector 24 byte, depth reached 85, errors: 1
  104 states, stored (120 visited)
    71 states, matched
    191 transitions (= visited+matched)
      0 atomic steps
hash conflicts: 0 (resolved)

Stats on memory usage (in Megabytes):
  0.004 equivalent memory usage for states (stored*(State-vector + overhead))
  0.289 actual memory usage for states
  64.000 memory used for hash table (-w24)
  0.343 memory used for DFS stack (-m10000)
  64.539 total actual memory usage

pan: elapsed time 0.129 seconds
pan: rate 2553.1915 states/second

```

Рис. 39. Результат верификации в ИС SPIN

По описанию алгоритмов работы процессов *Client* и *Server* были построены изображенные на рис. 40 структуры Крипке. Атомарным предикатом а отмечены состояния клиентского процесса, в которых выполняется условие *State == Wakeme*.

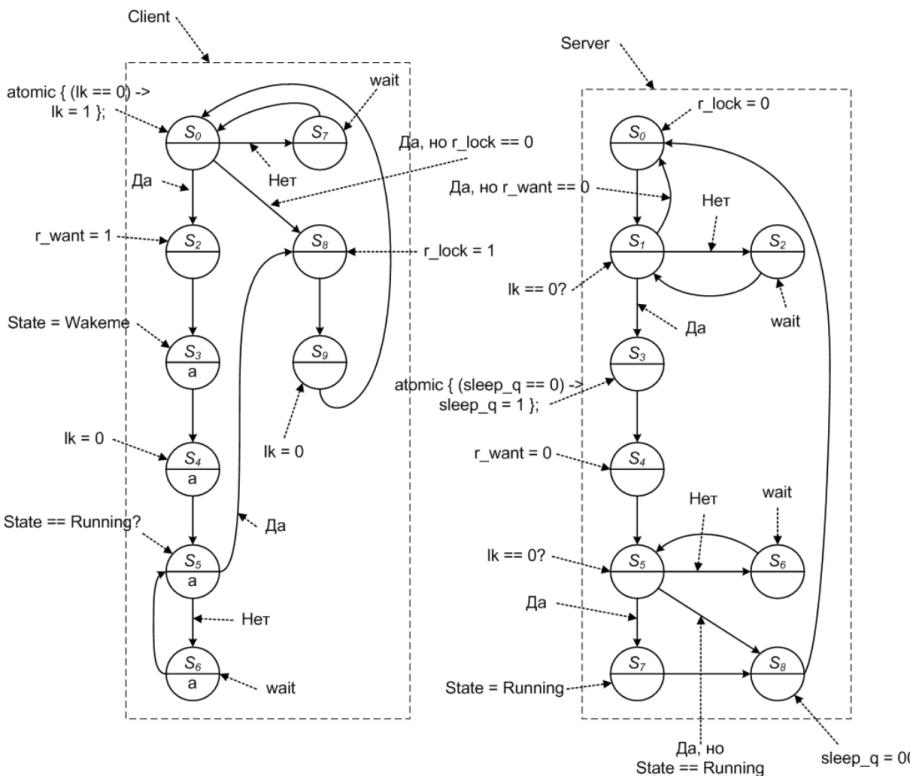


Рис. 40. Эквивалентные процессам структуры Крипке

Модель, соответствующая исследуемому алгоритму планирования, была сформирована путем построения асинхронной композиции представленных структур и последующего исключения недостижимых состояний. Граф полученной структуры Крипке содержит 83 вершины.

Предъявляемое требование может быть выражено с помощью LTL-формулы $GF\neg a$ (для каждого состояния модели всегда в будущем будет достижимо состояние, в котором клиентский процесс функционирует). На рис. 41 представлены результаты верификации алгоритма с помощью МВОЭ.

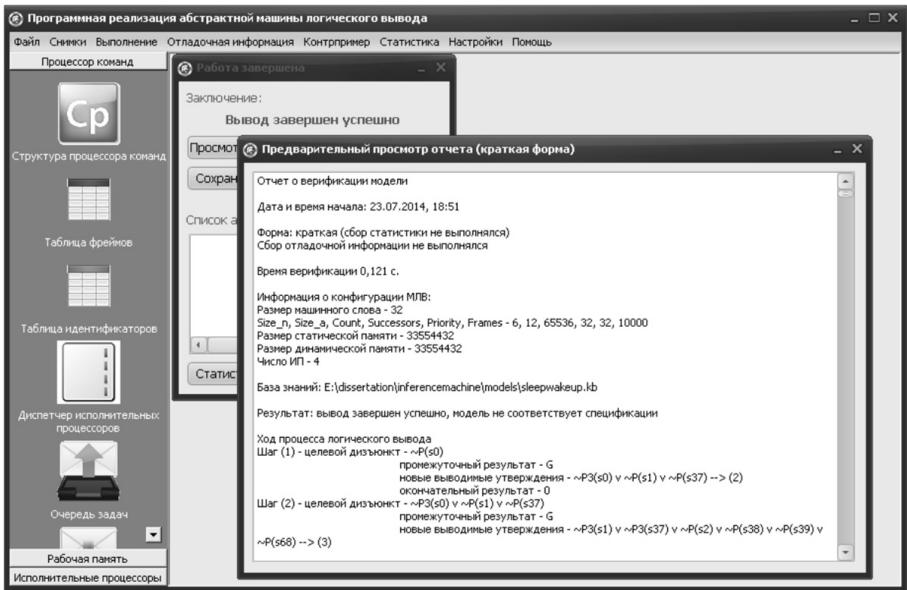


Рис. 41. Верификация с помощью МЛВ

Результаты, полученные в ИС SPIN и в программной реализации МЛВ, совпали с точностью до контрпримера. Исследуемый алгоритм не отвечает предъявляемому требованию – возможна ситуация, при которой клиентский процесс перейдет в состояние Wakeme и не выйдет из этого состояния в дальнейшем. Для устранения ошибки необходимо обеспечить атомарное выполнение смены состояния клиентского процесса и установки флага запроса на пробуждение.

4. АВТОМАТИЧЕСКИЙ АНАЛИЗ БЕЗОПАСНОСТИ ПЛАТЕЖНЫХ ПРОТОКОЛОВ

4.1. Протоколы бесконтактных платежей по банковским картам

В последние годы бесконтактные банковские карты получили в мире широкое распространение. По оценкам Всемирного банка, к концу 2020 года было задействовано около 900 миллионов бесконтактных кредитных карт и 1300000 бесконтактных считывающих устройств. Однако не все платежные протоколы, используемые этими банковскими картами, являются безопасными. В независимом исследовании [54] был изучен набор из 20 бесконтактных банковских карт от нескольких платежных организаций и 111 крупных банков-эмитентов в США. Исследование обнаружило уязвимости безопасности разной степени во всех 20 картах.

Эксперимент Хейдта-Бенджамина и др. [43] проводился со считывающими устройствами банковских карт от двух независимых производителей. Бесконтактные карты связываются со считывающими устройствами через транспортный уровень ISO14443-B [58]. Электронные данные карты были захвачены, и сообщения, передаваемые между картой и считывателем, были получены из последовательного порта считывателей. Затем выходные данные были визуализированы в формате магнитной полосы ISO7813 [59], а протоколы оплаты, используемые банковскими картами, были преобразованы в обратную инженерию.

На основании полученных данных протоколы, используемые 20 банковскими картами, были разделены на четыре категории. Далее формально смоделируем два из них и покажем выполнение этих двух протоколов и их атак.

Протокол оплаты банковской картой запускается, когда бесконтактная банковская карта касается считывающего устройства. Считыватель пересыпает информацию о карте и строку описания транзакции M на внутренний сервер, который затем проверяет согласованность информации и, если он удовлетворен, сообщает шлюзу биллинга. Поскольку считыватель и внутренний сервер часто совместно используют безопасный канал связи, такой как TLS [57], а считыватель просто пересыпает полученную информацию, то опустим явное мо-

делирование считывателя. Вместо этого предполагаем, что бесконтактная банковская карта напрямую взаимодействует с платежным сервером с M как часть передачи сообщения. Принимая это упрощение, теперь предоставляем высокуровневые описания двух протоколов бесконтактных платежей, типа A и типа B , из [54].

I. Протокол карты A: этот тип карты всегда отправляет считывающему устройству набор статической информации, которая содержит его основной номер счета (PAN), имя держателя карты и дату истечения срока действия. Пример последовательного вывода коммерческого считывателя с этим типом карты:

Первая строка – это сообщение, содержащееся на основной памяти карты; вторая строка содержит почти ту же информацию и представляет собой вторую запись в дополнительной памяти карты.

Здесь «xxxxxx7532xxxxxx» – это номер PAN, «Erok/Victor» – имя держателя карты, а «2106» – срок действия. Пусть данные владельца карты (ДВК) будет термином, который представляет собой комбинацию PAN, имени держателя карты и даты истечения срока действия. Предполагаем, что оставшаяся строка является подписью банка-эмитента над всеми данными держателя карты. Делаем упрощающее предположение, что платежный сервер Se отвечает за выпуск банковских карт. У Se имеется открытый и закрытый ключи $\{k_{se.open}, k_{se.lock}\}$. Формализация нашего сообщения следующее:

$M_{\cdot \text{ДВК}}.r_v(\theta(\text{ДВК}), Se)$,

где r_v – симметричное шифрование; $\Theta(\text{ДВК})$ – краткое сообщение на основе хеширования.

Короткое сообщения: $\theta(\text{ДВК})$. Пусть переменная ДВК представляет сообщение. Определите функцию $\theta(\text{ДВК})$ как короткое сообщение ДВК , вычисленный с использованием криптографической хеш-функции [55, 56]. Предполагаем, что платежный терминал использует идеальное хеширование:

1. Если $(ДВК_1 \neq ДВК_2)$, тогда $(\theta(ДВК_1) \neq \theta(ДВК_2))$.
 2. Учитывая $\theta(ДВК)$, невозможно восстановить $ДВК$.

3. Пусть f – любая функция, тогда, если $(\text{ДВК} \neq f(\text{ДВК}))$, тогда $(\theta(\text{ДВК}) \neq \theta(f(\text{ДВК})))$.

$M.\text{ДВК}.r_v(\theta(\text{ДВК}), Se)$ представляет собой объединение простого текста ДВК и подписи его хеша. Поскольку ДВК передается в открытом виде, этот протокол явно подвержен атаке с перехватом. Более того, у этого типа карт нет компонента изменчивости – он всегда отправляет одну и ту же строку при общении со считывателем. Следовательно, он также подвержен атаке повторного воспроизведения. Наконец, M не подписан, поэтому протокол уязвим для атак вырезания и вставки.

II. Протокол карты B :

Этот тип карты по-прежнему отправляет данные владельца карты (ДВК) в открытом виде. Тем не менее, он значительно повысил свою безопасность, сохраняя значение счетчика, которое монотонно увеличивается при каждом нажатии. Пример последовательного вывода для этого типа карты при обмене данными со считывателем:

```
Bxxxxxxxx3272xxxxxx^ Erok/Victor ^21061011000000000001000000000000
xxxxxxxx3272xxxxxx=21061011000001800431
```

В этом примере выходных данных «**0043**» – это значение счетчика, которое увеличивается при каждой банковской транзакции. Это обеспечивает изменчивость каждого сообщения о транзакции. Три предшествующие цифры «**018**» изменяются при каждой транзакции (пин код, получаемый от банка), и предполагается, что они являются подписью над ДВК и значением счетчика. В этом случае предполагаем, что платежный сервер сохраняет сопоставление каждого ДВК с его последним значением счетчика, которое видел сервер. Более того, платежный сервер использует уникальный симметричный ключ k с каждой картой, которая идентифицируется ДВК. Формализованное сообщение:

$M.\text{ДВК}.C.w(\text{ДВК}.C, k)$

где C – конкретное значение счетчика, а $w(\text{ДВК}.C, k)$ – код аутентификации сообщений на основе хеша через ДВК.C с использованием ключа k .

Поскольку протокол карты A всегда отправляет одно и то же статическое сообщение, он уязвим для атак повторного воспроизве-

дения. Один из шагов, предпринимаемых платежным сервером после получения сообщения от бесконтактной карты и специфической для транзакции полезной нагрузки M , – это проверка этой информации на актуальность и согласованность. В этом протоколе имеется три принципала: банковская карта (Bc), платежный сервер (Se) и биллинговый шлюз (Bi), который обрабатывает транзакции для выставления счетов. Предполагаем, что канал между Se и Bi безопасен, и тем самым опускаем моделирование бесконтактного считывателя, поскольку он только добавляет информацию описания транзакции M и отправляет все на платежный сервер. Диаграмма последовательности UML протокола карты типа A представлена на рис. 42.

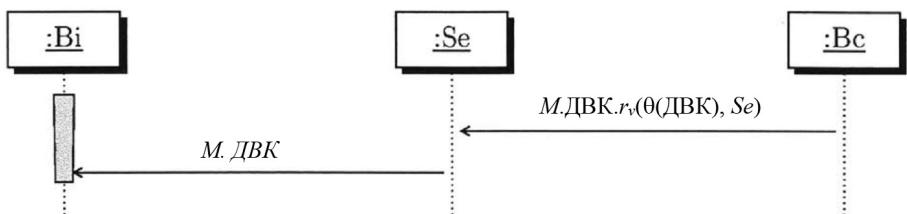


Рис. 42. Схема последовательности UML для протокола бесконтактных платежей типа А

Первоначальные предположения - это признание того, что Se имеет собственные открытые и закрытые ключи. Se моделируются в виде двух формул:

$$\begin{aligned} Se &\equiv \gamma(Se, k_{se.open}); \\ Se &\equiv \psi(Se, k_{se.lock}), \end{aligned}$$

где $\gamma(Se, k_{se.open}) = k_{se.open}$ является открытым ключом Se ; $\psi(Se, k_{se.lock}) = k_{se.lock}$ является закрытым ключом Se .

Набор предварительных условий – это набор целей безопасности, которые, как ожидается, будут достигнуты перед каждым этапом обмена сообщениями. Приведем возможный список целей и кратко обсудим, как их можно формализовать. Рассмотрим совокупность предварительных условий, которые Se устанавливает перед взаимодействием с биллинговым шлюзом Bi . Предположим, Se получил $M.ДВК.rv(\theta(ДВК), Se)$. Прежде чем Se отправит сообщение на сервер биллинга, необходимо выполнить ряд требований безопасности.

1а. Se должен полагать, что ДВК представляет собой информацию о карте, выпущенной Se . Мы моделируем это, заявляя, что Se однажды изготовил ДВК:

$$Se \equiv \mathbf{G}(\text{ДВК}, Se),$$

где \mathbf{G} (ДВК, Se) – Se один раз произнес (изготовил) ДВК.

2а. Se необходимо убедиться, что ДВК не поврежден, поскольку он был изготовлен Se . Именно этот ДВК изготовил Se :

$$Se \equiv \eta(\text{ДВК}, Se),$$

где η – целостность сообщения.

3а. Se необходимо убедиться, что транзакция М.ДВК должна была быть совершена Bc . Сделка, представленная приобретенными товарами M и финансовым инструментом, используемым для оплаты ДВК, была инициирована Bc , а не каким-либо другим принципалом:

$$Se \equiv \mathbf{G}(M.\text{ДВК}, Bc).$$

То есть M .ДВК произнес Bc .

4а. Запрашивается именно эта транзакция М.ДВК, а не какой-либо другой M .ДВК':

$$Se \equiv \eta(M.\text{ДВК}, Bc).$$

5а. Se необходимо убедиться, что запрос транзакции был последним. Se должен быть в состоянии исключить любую атаку повторного воспроизведения, в которой старый запрос транзакции был записан в атаке подслушивания и повторно отображен здесь. Формализуем это требование как следующее убеждение Se :

$$Se \equiv \#(M.\text{ДВК}),$$

где $\#$ – изменение $(\#(M.\text{ДВК}) - \text{изменение } M.\text{ДВК})$.

Здесь для простоты опускаем постусловия, которые обычно включают требования к оплате, такие как неотказ от авторства.

Протокол бесконтактных платежей типа A , представленный на рис. 42, позволяет надежно реализовать только первый и второй пункты безопасности (смотри выше: 1а, 2а). Третий, четвертый и пятый пункты безопасности (3а, 4а, 5а) не могут быть надежно реализованы по протоколу A . В частности, третий (3а) не выполняется, потому что сообщение $M.ДВК$ не было подписано Bc ; четвертый (4а) не работает по той же причине; и пятый (5а) не функционирует, потому что в $M.ДВК$ нет компонента изменчивости.

Протокол для банковской карты типа B использует счетчик C и симметричное шифрование, а также код аутентификации сообщений на основе хеша. Предполагаем, что Se знает последнее значение счетчика, связанного с данной банковской картой. Диаграмма последовательности UML этого протокола представлена на рис. 43:

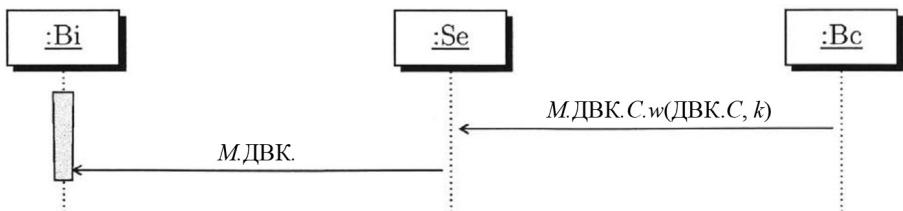


Рис. 43. Схема последовательности UML для протокола бесконтактных платежей типа В

Предположим, что Bc имеет общий симметричный ключ k с Se . Ключ k унаследован для Bc и ни Bc , ни Se не используют тот же k для других приложений. Более того, Se знает, является ли значение счетчика C , отправленное Bc , новым, поскольку Se отслеживает последнее значение счетчика Bc . Наконец, Se считает, что он никогда не создавал код аутентификации сообщений на основе хеша $w(\text{ДВК}.C, k)$, поэтому Bc не может его воспроизвести. Только Se и Bc , которые являются единственными принципалами, обладающими k , могут построить этот код аутентификации сообщений на основе хеша. Se считает, что ключи надежно хранятся и извлекаются на всех картах. Только протокол, работающий на Bc , может иметь доступ к k в соответствии с конструкцией аппаратной защиты.

Следовательно, если Se считает, что он никогда не раскрывал $w(\text{ДВК}.C, k)$, тогда Se знает, что код аутентификации сообщений на

основе хеша является непередаваемым доказательством аутентификации Bc (он должен быть сгенерирован Bc). Таким образом, нам потребуется следующий набор исходных предположений:

1. Предположение утверждает, что Bc знает симметричный ключ:

$$Bc \equiv y_K(k, Bc, Se),$$

где $y_K(k, Bc, Se)$ – симметричный ключ k между Bc и Se .

2. Предположение утверждает, что Se знает симметричный ключ:

$$Se \equiv y_K(k, Bc, Se).$$

3. Предположение утверждает, что Se знает, что C изменился

$$Se \equiv \#(C).$$

4. Предположение утверждает, что Se никогда не генерировал код аутентификации сообщений на основе хеша.

$$Se \equiv G_{lock}(w(\text{ДВК.}C, k), Se),$$

где $G_{lock}(w(\text{ДВК.}C, k), Se)$ – Se никогда не генерировал $w(\text{ДВК.}C, k)$.

Теперь перечислим возможный набор предварительных и дополнительных условий. Нам не нужны предварительные условия для сообщения номер один от Bc до Se на рис. 43. Набор предварительных условий указан для второго сообщения, прежде чем Se сообщит Bi :

16. Se необходимо сделать вывод, что информация о банковской карте $\text{ДВК.}C$ отправляется Bc , а не какой-либо другой банковской картой Bc' . Это требование моделируется следующим образом:

$$Se \equiv \mathbf{G}(\text{ДВК.}C, Bc).$$

26. Se необходимо получить уверенность в том, что $\text{ДВК.}C$ не поврежден, поскольку Bc произнес (сгенерировал) его:

$$Se \equiv \eta(\text{ДВК.}C, Bc).$$

3б. Se необходимо убедиться, что $M.\text{ДВК.}C$ изменился:

$$Se \equiv \#(M.\text{ДВК.}C).$$

4б. Se необходимо убедиться, что именно Bc запросил транзакцию $M.\text{ДВК.}C$:

$$Se \equiv \mathbf{G}(M.\text{ДВК.}C, Bc).$$

5б. Se необходимо убедиться, что запрос транзакции $M.\text{ДВК.}C$ не поврежден, поскольку он был передан Bc :

$$Se \equiv \eta(M.\text{ДВК.}C, Bc).$$

6б. Укажем одно постусловие неотказуемости. То есть Se считает, что может доказать, что транзакция (включая описание товаров, дату, дату и платежный инструмент) была запрошена Bc . Это утверждение моделируется в виде:

$$Se \equiv \phi(Se, \mathbf{G}(M.\text{ДВК.}C, Bc)).$$

При программной реализации этой спецификации протокола были получены следующие результаты. Первое, второе и третье предварительные условия (смотри выше: 1б, 2б, 3б) были подтверждены. Они имеются, Se считает, что Bc сказал $\text{ДВК.}C$; Se верит в целостность $\text{ДВК.}C$; и Se считает, что $\text{ДВК.}C$ изменился. Однако протокол не смог надежно отработать четвертое и пятое условие (4б, 5б). Это означает, что намерение Bc послать M не может быть доказано. M не подписано Bc . Заключаем, что этот протокол не является корректным в отношении указанных исходных предположений, предварительных и постусловий. Обратите внимание, что не проверяется постусловие 6б. Это связано с тем, что отказов любых предварительных условий достаточно, чтобы сделать протокол-кандидат недействительным.

Упростим условие моделирования, отбросим два предварительных условия 4б и 5б, а также постусловие 6б. Сравниваем исходную спецификацию протокола с этим набором

смягченных предварительных и постусловий и обнаруживаем, что теперь это корректно для функционирования банковских карт. Теперь перейдем к этому набору упрощенных предварительных и постусловий.

Исходный протокол для карты типа *B* состоит из двух передач сообщений:

$$Bc \rightarrow Se:M.\text{ДВК}.C.w(\text{ДВК}.C, k);$$

$$Se \rightarrow Bi:M.\text{ДВК},$$

где \rightarrow – заслуживает доверие.

Поскольку предполагается, что вторая передача осуществляется по защищенному каналу, сосредоточимся на атаке, которая изменяет первую передачу. Используя стратегию модификации, можем заменить $M.\text{ДВК}.C.w(\text{ДВК}.C, k)$ на путь, который начинается с *Bc* и заканчивается в *Se* через принципала-нарушителя H^* модели злоумышленника Dolev-Yao. Конкретно, если позволим H^* быть злоумышленником модели Dolev-Yao, первая передача может быть заменена следующим путем:

$$Bc \rightarrow H^*:M.\text{ДВК}.C.w(\text{ДВК}.C, k),$$

где

$$H^* \rightarrow Se:M.\text{ДВК}.C.w(\text{ДВК}.C, k).$$

Такой путь можно рассматривать как атаку с подслушиванием, которая незаметно прослушивает канал связи между *Bc* и *Se*. Поскольку нет никаких модификаций содержимого сообщения, т.е. никакие существующие предварительные условия или постусловия не изменяются. Однако для передачи сообщения добавляется пустой набор предварительных условий ($H^* \rightarrow Se:M.\text{ДВК}.C.w(\text{ДВК}.C, k)$). Набор предварительных условий для передачи сообщения ($Bc \rightarrow H^*:M.\text{ДВК}.C.w(\text{ДВК}.C, k)$) остается пустым набором, а предварительные условия для ($Se \rightarrow Bi:M.\text{ДВК}$) состоят из: 16, 26 и 3б.

Утверждение злоумышленника, которое моделируется, касается секретности информации о банковской карте ДВК. Поскольку

передачи между бесконтактными банковскими картами и считывателями транслируются, их может прочитать любой, кто их подслушивает. В этом случае необходимо, чтобы злоумышленник-перехватчик H^* не мог получить финансовую информацию ДВК в открытом виде. Формально такое утверждение злоумышленника имеет вид:

$$\neg H^* \equiv \Xi(H^*, \text{ДВК}),$$

где \neg – отрицание; $\Xi(H^*, \text{ДВК}) - H^*$ может восстановить ДВК.

То есть злоумышленник H^* не верит, что может реконструировать ДВК.

При программной реализации этой спецификации протокола протокол успешно отработал до конца. В этом протоколе атаки не было поступления для проверки. Однако было обнаружено, что утверждение злоумышленника неверно. Наша программная реализация правильно вычислила $H^* \equiv \Xi(H^*, \text{ДВК})$ как действительную формулу. То есть злоумышленник H^* действительно может реконструировать ДВК, потому что ДВК был отправлен в открытом виде от Bc до Se . Заключаем, что исходный протокол неверен при столкновении с этой специфической атакой секретности.

Вывод. Выявили ряд недостатков в вышеописанных протоколах. В частности, протоколу типа A не удалось убедить Se в том, что транзакция была недавно инициирована Bc и содержимое транзакции было повреждено. Протокол типа B не смог учесть намерение Bc провести транзакцию для определенного M . Протокол уязвим для атаки методом «вырезать и вставить» на M и нарушает требование секретности данных ДВК держателя банковской карты.

EMV [47, 48] – ведущий стандарт оплаты, администрируемый EMVCo (<http://www.emvco.com>). EMV не является спецификацией для одного протокола. Он состоит из нескольких легко настраиваемых модулей, предназначенных для различных компонентов платежных протоколов. Далее формализуем аутентификационную часть протокола EMV и проведем его анализ.

Протокол аутентификации, используемый в EMV, можно смоделировать с использованием двух принципов: считывателя Re и банковской карты Bc . Мы предполагаем, что все банковские карты выпущены доверенным третьим принципалом N . Принципал N имеет пару открытого и закрытого ключей $\{k_{N.open}, k_{N.lock}\}$. Закрытый ключ

$k_{N.lock}$ принципала N хранится в секрете, а открытый ключ $k_{N.open}$ известен всем считывающим устройствам. Предполагаем, что каждая банковская карта надежно хранит свои личные ключи $k_{Bc.lock}$.

На этапе инициализации Bc и Re обмениваются последовательностью сообщений подтверждения для настройки протокола (эти шаги не моделируем). В конце инициализации Bc отправляет свой ДВК, включая дату истечения срока, PAN и некоторую вспомогательную информацию, такую как список объектов данных управления рисками карты (Card Risk Management Data Object List, CDOL), в Re . Теперь протокол EMV готов к аутентификации карты.

Цель аутентификации – проверить целостность ДВК и проверить, действительно ли он исходит от заявленного Bc . Спецификация EMV предоставляет три типа методов аутентификации: статическая аутентификация данных (САД), динамическая аутентификация данных (ДАД) и комбинированная аутентификация данных (КАД). КАД похож на ДАД, но с дополнительной транзакционной информацией, включенной в сообщения аутентификации. Далее проведем формализацию протокола аутентификации САД и ДАД, а затем выполним их анализ.

В САД карта возвращает подпись хеш-кода ДВК, подписанную эмитентом карты N . Эта подпись была предварительно вычислена во время производства и хранится на карте. Протокол аутентификации САД можно формализовать как передачу следующего сообщения:

$$Bc \rightarrow Re; \text{ДВК}.r_v(\theta(\text{ДВК}), N).$$

Первоначальные предположения – это создание ключей до аутентификации. Формально имеем:

1. Принципал N считает, что ему принадлежит открытый ключ $k_{N.open}$:

$$N \equiv \gamma(N, k_{N.open}).$$

2. Принципал N считает, что ему принадлежит закрытый ключ $k_{N.lock}$:

$$N \equiv \psi(N, k_{N.lock}).$$

3. Re считает открытый ключ $k_{N.open}$ принадлежит N :

$$Re \equiv \gamma(N, k_{N.open}).$$

4. Re считает, что N заслуживает доверия:

$$Re \equiv N \rightarrow \text{trustworthy}.$$

Bc не предъявляет никаких предварительных условий перед отправкой этого сообщения. Таким образом, предварительным условием для САД является пустой набор.

Набор требуемых Re постусловий формализуется следующим образом:

1в. Re считает, что ДВК был выпущен N :

$$Re \equiv \mathbf{G}(\text{ДВК}, N).$$

2в. Re верит в целостность ДВК от N . Именно ДВК выдал N :

$$Re \equiv \eta(\text{ДВК}, N).$$

3в. Re считает, что этот запрос на транзакцию отправил Bc :

$$Re \equiv \mathbf{G}(\text{ДВК}, Bc).$$

4в. Re считает, что ДВК был недавно отправлен Bc и не является повтором. Моделируя это требование, утверждаем

$$Re \equiv \#(\text{ДВК}).$$

5в. Re считает целостность ДВК от Bc :

$$Re \equiv \eta(\text{ДВК}, Bc).$$

Выполнив программную реализацию этой спецификации протокола, обнаружили, что последние три постуслугия 3в, 4в и 5в недействительны. Использование САД оставляет злоумышленникам возможность воспроизвести некоторые $r_v(\theta(\text{ДВК}), N)$ в Re . Re может проверить, действительно ли ДВК был выдан N . Однако Re не может

доказать, что это был конкретный Bc , который недавно отправил ДВК, или что ДВК не поврежден, поскольку он был отправлен этим Bc .

Заключаем, что аутентификация САД не является корректной в отношении указанного набора исходных предположений, предварительных и последующих условий. Обратите внимание, что EMV понимает эти ограничения и рекомендует использовать САД только в условиях ограниченного и низкого риска.

Некоторые ограничения САД устраняются ДАД. В ДАД аутентификация карты осуществляется случайным вызовом и применением цифровой подписи. Карта также может аутентифицировать считыватель; мы не будем это моделировать. Bc сначала отправляет ДВК и его сертификат Re . Сертификат не является стандартным сертификатом X.509 [53]. Сертификат может быть оформлен в виде подписи с отметкой времени истечения срока действия, открытого ключа Bc и хеша ДВК, подписанного N закрытым ключом $k_{N.lock}$:

$$Bc \rightarrow Re:\text{ДВК}.r_v(\Theta^*(T_{exp}).\gamma(Bc, k_{Bc.open}).\theta(\text{ДВК}), N),$$

где Θ^* – отметка времени истечения срока.

Далее Re отправляет объект данных динамической аутентификации данных (Dynamic Data Authentication Data Object, DDOL) в Bc . DDOL содержит сгенерированный считывателем одноразовый номер N_{Re} , который, как известно Re , является измененным, и дополнительно содержит некоторую информацию, которую мы не будем моделировать.

$$Re \rightarrow Bc:\chi(N_{Re}),$$

где χ – в данное время (в режиме реального времени).

Получив N_{Re} , Bc создает объект динамических данных ICC (ICCDDO, ICC Dynamic Data Object), который состоит из только что сгенерированного одноразового номера N_{Bc} . Затем Bc возвращает свою подпись над ICCDDO и хеш-значениями ICCDDO и DDOL.

$$Bc \rightarrow Re:r_v(\chi(N_{Bc}).\theta(\chi(N_{Bc}).\chi(N_{Re})), Bc).$$

Сертификат в $Bc \rightarrow Re:\text{ДВК}.r_v(\Theta^*(T_{exp}).\gamma(Bc, k_{Bc.open}).\theta(\text{ДВК}), N)$ подписан эмитентом карты R . Он распознает, что открытый ключ $k_{Bc.open}$ связан с ДВК. Таким образом, ДВК ассоциируется с любым, кто может продемонстрировать владение соответствующим закрытым ключом. В $Re \rightarrow Bc:\chi(N_{Re})$ новый вызов отправляется Bc . В $Bc \rightarrow Re:r_v(\chi(N_{Bc}).\theta(\chi(N_{Bc}).\chi(N_{Re})), Bc)$ Bc предоставляет подпись над этим одноразовым номером и демонстрирует, что он имеет соответствующий закрытый ключ, и, следовательно, аутентифицируется в Re . Опишем формализации исходных предположений.

Исходные предположения.

1. Эмитент банковской карты N знает ее открытый и закрытый ключи:

$$N \equiv \gamma(N, k_{N.open});$$

$$N \equiv \psi(N, k_{N.lock}).$$

2. Re знает открытый ключ, принадлежащий N :

$$Re \equiv \gamma(N, k_{N.open}).$$

3. Bc знает, что N_{Bc} изменился, а Re знает, что N_{Re} изменился:

$$Bc \equiv \#(\chi(N_{Bc}));$$

$$Re \equiv \#(\chi(N_{Re})).$$

4. Re знает, что метка времени $\Theta^*(T_{exp})$ еще не истекла:

$$Re \equiv \Delta(\Theta^*(T_{exp})),$$

где Δ – изменился или срок годности не истек.

5. Re считает, что эмитент банковской карты N заслуживает доверия:

$$Re \equiv N \rightarrow \text{trustworthy}.$$

Три предварительных условия для первого, второго и третьего сообщения соответственно:

1. Нуль 0 (пустой набор): Bc всегда запускает аутентификационную часть протокола ДАД.

2. После получения первого сообщения и до второго сообщения Re должен поверить, что ДВК был объявлен N , не поврежден и все еще действителен. Re также должен верить, что открытый ключ Bc – это $k_{Bc.open}$:

$$Re \equiv \mathbf{G}(\text{ДВК}, N);$$

$$Re \equiv \eta(\text{ДВК}, N);$$

$$Re \equiv \gamma(Bc, k_{Bc.open}).$$

3. Нуль 0 (пустой набор): Bc всегда отвечает на DDOL.

В конце протокола Re должен полагать, что ответ от Bc был недавно сгенерирован с использованием закрытого ключа, соответствующего открытому ключу $k_{Bc.open}$. Если это предположение может быть подтверждено, то Re считает, что Bc владеет закрытым ключом, который соответствует $k_{Bc.open}$. Следовательно, Bc аутентифицирован. В этом случае постусловие имеет следующее утверждение:

$$Re \equiv \mathbf{G}(\chi(N_{Re}), Bc).$$

То есть Bc аутентифицирует себя и подтверждает в данный момент времени сгенерированный одноразовый номер N_{Re} .

Программная реализация этой спецификации протокола и успешно завершилась. Все постусловия действительны. Заключаем, что протокол является корректным.

4.2. Протоколы оплаты с использованием мобильных устройств

В этом параграфе рассматривается мобильный протокол бесконтактных платежей, который обеспечивает высокий уровень безопасности, а также дополнительные функции, такие как предотвращение отказа от авторства. Обсуждается его формализация и анализ.

Рассматриваемый протокол мобильных платежей позволяет мобильному телефону с поддержкой NFC производить оплату через считывающее устройство NFC, которое надежно подключается к шлюзу обработки платежей. В этой главе мы подробно рассмотрим первый тип протокола мобильных платежей и его анализ.

Протокол оплаты с использованием мобильного устройства, обозначаемый как P_D , позволяет мобильному телефону с поддержкой NFC взаимодействовать с устройством чтения NFC для совершения транзакции через NFCIPI (NFC-LLCP) [71]. Предполагаем, что считыватель NFC заслуживает доверия (заслуживающий доверия принципал обслуживается и использует правильный и аутентифицированный протокол) и подключен к платежному серверу через безопасный канал TLS [57, 55]. Предполагается, что связь между считывателем NFC и сервером безопасна и выходит за рамки этого анализа. Платежный сервер отвечает за сбор доказательств транзакций и выполнение таких операций, как выставление счетов и аудит. Мы будем использовать атомарные переменные Ph для обозначения телефона, Re для обозначения считывающего устройства и N для обозначения центра сертификации, который выдает сертификаты и управляет ими.

Обозначим $\psi(Ph, \delta(ID_{Ph}), \Theta(T_{1Ph}, T_{2Ph}), k_{ph.open}, N)$ как сертификат, выданный Ph . Обозначим $\psi(Re, \delta(ID_{Re}), \Theta(T_{1Re}, T_{2Re}), k_{Re.open}, N)$ как сертификат, выданный Re . Далее будем использовать $\psi(P)$ в качестве сокращенного обозначения для обозначения полного сертификата X.509 принципала P .

Телефон с функцией NFC не поддерживает конечный автомат для платежной информации, такой как баланс. Вместо этого эта информация хранится на платежном сервере. Телефон идентифицируется по сертификату и некоторой вспомогательной

информации, такой как связанный с ним постоянный номер счета (Permanent Account Number, PAN).

На телефоне с функцией NFC и считывателе имеются элементы защиты [63, 44, 38, 64, 65, 39, 68, 67, 45]. Встроенный элемент безопасности хранит закрытый ключ, а также любые секреты, которые могут быть удаленно переданы оператором на телефон или считающее устройство. Секреты состоят как из недолговечных симметричных ключей, так и из протокольных программ. Невозможно прочитать или изменить содержимое за пределами защищенного элемента, и только аутентифицированные программы могут получить доступ к его соответствующим ключам или секретам.

Предполагаем, что существует один платежный сервер и много аутентифицированных считывателей и телефонов. Кроме того, предполагаем, что один *Re* может обслуживать только одного *Ph* в любое время, согласование версии и согласование были успешно выполнены, и *Ph* и *Re* согласовали протокол связи. Рассмотрение этой процедуры выходят за рамки нашего анализа.

Исходное предположение – это набор обоснованных формул, которые считаются действительными до запуска протокола. Закрытые ключи *Ph* и *Re* генерируются внутри защищенных элементов. После создания они никогда не покидают защищенные элементы, и соответствующие открытые ключи отправляются на *N* для сертификации. Предполагаем, что сертификаты были подписаны *N*, а затем переданы *Ph* и *Re* до начала сеанса отработки протокола. Дополнительно предполагаем:

1. *Ph* и *Re* знают открытый ключ центра сертификации *N*:

$$\text{Ph} \equiv \gamma(N, k_{N.open});$$

$$\text{Re} \equiv \gamma(N, k_{N.open}).$$

2. И *Ph*, и *Re* считают, что *N* заслуживает доверия. *Ph* считает, что *Re* заслуживает доверия:

$$\text{Ph} \equiv N \rightarrow \text{trustworthy};$$

$$\text{Re} \equiv N \rightarrow \text{trustworthy};$$

$$\text{Ph} \equiv \text{Re} \rightarrow \text{trustworthy}.$$

3. Ph и Re имеют доступ к своим закрытым и открытым ключам:

$$Ph \equiv \gamma(Ph, k_{ph.open});$$

$$Re \equiv \gamma(Re, k_{Re.open});$$

$$Ph \equiv \psi(Ph, k_{ph.lock});$$

$$Re \equiv \psi(Re, k_{Re.lock}).$$

4. ID_{Ph} и ID_{Re} не включены в список аннулированных сертификатов, а отметки времени на сертификатах Ph и Re соответствуют текущему времени:

$$Ph \equiv \chi(\delta(ID_{Re}));$$

$$Re \equiv \chi(\delta(ID_{Ph}));$$

$$Ph \equiv \Delta(\Theta(T_{1Re}, T_{2Re}));$$

$$Re \equiv \Delta(\Theta(T_{1Ph}, T_{2Ph})),$$

где δ – идентичность.

5. Re может восстановить сообщение server-hello, которое должно быть сгенерировано им самим в сообщении 2 (второе сверху) на рис. 44:

$$Re \equiv \Xi(Re, \chi(N_{Re}).\Theta(T_{Re}).D_{Tran}),$$

где D_{Tran} – дескриптор (описание) транзакции, которое может включать финансовую информацию о взимаемой сумме, продавце, местонахождении и т.д.

6. Ph может реконструировать D_{Ph} сообщения, который должен быть сгенерирован сам по себе в сообщении 3 (третье сверху) на рис. 44:

$Ph \equiv \Xi(Ph, D_{Ph})$.

7. Re и Ph считают, что метка времени $\Theta(T_{Re})$ изменилась:

$Re \equiv \#(\Theta(T_{Re}))$;

$Ph \equiv \#(\Theta(T_{Re}))$.

8. Re считает генерируемый им в данное время $\chi(N_{Re})$ поменялся:

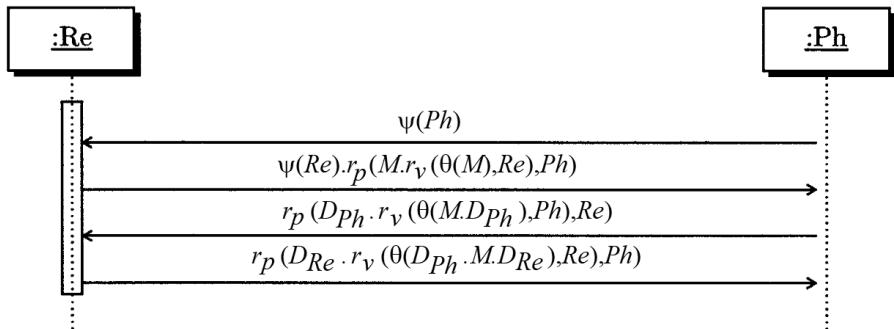


Рис. 44. Схема последовательности UML для протокола P_D оплаты с использованием мобильных устройств на основе NFC

Ниже приводится пошаговое описание протокола P_D .

Шаг 1 (Client Hello). Ph сначала отправляет свой сертификат в качестве приветствия клиента Re .

$Ph \rightarrow Re: \psi(Ph)$.

Перед выполнением шага 2 необходимо выполнить повторную проверку сертификата Ph . Важным в этом процессе является обеспечение истечения срока действия сертификата и его отзыв. Проверка должна быть успешной, и поэтому Re может получить открытый ключ Ph :

$Re \equiv \gamma(Ph, k_{Ph.open})$.

Шаг 2 (Server Hello). После получения приветствия клиента от Ph , Re создает запрос M , состоящий из нового (измененного) одноразового номера $\chi(N_{Re})$, метки времени T_{Re} , принадлежащей Re , описания транзакции D_{Tran} и идентификатор получателя $\delta(ID_{Ph})$, который Re получает из $\psi(Ph)$, полученного на предыдущем этапе. M оформляется в виде:

$$M = \chi(N_{Re}).\Theta(T_{Re}).D_{Tran}.\delta(ID_{Ph}).$$

Re имеет требование конфиденциальности по M , поскольку D_{Tran} содержит транзакционную информацию, которую нельзя раскрыть. Затем Re отправляет свой сертификат и шифрованный текст, зашифрованный с использованием открытого ключа Ph , на Ph :

$$Re \rightarrow Ph: \psi(Re).r_p(M.r_v(\theta(M), Re), Ph).$$

До реализации шага 3 Ph должен подтвердить сертификат Re . Он также проверяет целостность, актуальность и является ли сервер предполагаемым получателем (если его идентификатор подтвержден). Он также может выполнять некоторые семантические проверки, зависящие от протокола транзакции. Например, он проверяет, является ли пакет запроса транзакции D_{Tran} правильным и ожидаемым. Поскольку они не имеют непосредственного отношения к модели безопасности, не будем их включать в предварительные условия. Однако их легко добавить. Таким образом, предварительными условиями для этой передачи являются:

1. Ph считает, что открытый ключ принадлежит Re :

$$Ph \equiv \gamma(Re, k_{Re.open}).$$

2. Ph считает, что M происходит от Re :

$$Ph \equiv \mathbf{G}(M, Re)$$

3. Ph считает M измененным:

$$Ph \equiv \#(M).$$

4. Ph считает, что M не повреждено с тех пор, как произнесено Re :

$$Ph \equiv \eta(M, Re).$$

5. Ph считает, что это предполагаемое считающее устройство для M , и что M подтверждает свой идентификатор:

$$Ph \equiv \mathfrak{R}(M, Ph);$$

$$Ph \equiv w_P(M, \delta(ID_{Ph})),$$

где $w_P(M, \delta(ID_{Ph})) = \delta(ID_{Ph})$ является частью M ; $\mathfrak{R}(M, Ph) = Ph$ является получателем от M .

$\delta(ID_{Ph})$ в $Ph \equiv w_P(M, \delta(ID_{Ph}))$ является идентификатором Ph , содержащийся в $\psi(Ph)$. Это подтверждение моделирует проверку, которую выполняет Ph . Если личность, подтвержденная Re , отличается от $\delta(ID_{Ph})$, тогда Ph знает, что транзакция фальсифицирована.

Шаг 3 (привязка телефона, Phone Binding). Затем Ph строит свое описательное сообщение D_{Ph} . D_{Ph} будет содержать любую дополнительную информацию, которую Ph должен предоставить для завершения этой транзакции, например, ДВК. Затем Ph производит подпись S_{Ph} . Подпись верифицирует $M.D_{Ph}$ в Ph .

$$S_{Ph} = r_v(\theta(M.D_{Ph}), Ph).$$

После этого Ph шифрует D_{ph} и подпись, используя открытый ключ, принадлежащий Re , и отправляет шифрованное сообщение Re .

$$Ph \rightarrow Re:r_p(D_{Ph}.S_{ph}, Re).$$

До реализации шага 4 необходимо провести повторную проверку целостности, и подлинности сообщения. Кроме того, Re проверяет, подтверждает ли Ph правильный M (M , созданный Re , а не отправленный Ph). Предварительные условия на этом этапе:

1. Re считает, что Ph прислала подтверждение транзакции в лице $M.D_{Ph}$:

$$Re \equiv \mathbf{G}(M.D_{Ph}, Ph).$$

2. Re считает, что $M.D_{Ph}$ – это только что сгенерированная транзакция, а не её повтор:

$$Re \equiv \#(M.D_{Ph}).$$

3. Re считает, что $M.D_{ph}$ не поврежден, так как он был сгенерирован Ph :

$$Re \equiv \eta(M.D_{Ph}, Ph).$$

Шаг 4 (Получение, Receipt). После получения D_{ph} и проверки его целостности и подлинности Re генерирует квитанцию и передает ее Ph . Квитанция, обозначаемая как D_{Re} , представляет собой композицию запроса транзакции D_{Tran} и ответа D_{ph} плательщика, а также распознавания считывающим устройством двойной привязки. Пусть $R_{Ph.Re}$ обозначает квитанцию, сгенерированную Re для отправки в Ph :

$$R_{Ph.Re} = D_{Re}.r_v(\theta(D_{Ph}.M.D_{Re}), Re).$$

Наконец, $R_{Ph.Re}$ зашифровывается и отправляется Ph . Таким образом, имеем

$$Re \rightarrow Ph:r_p(R_{Ph.Re}, Ph).$$

Постусловия:

A. Ph проверяет: подтверждает ли Re правильный D_{ph} (Ph недавно отправил D_{ph}) и что $D_{Ph}.M.D_{Re}$ от Re , изменился и неповрежденный.

$$Ph \equiv \mathbf{G}(D_{Ph}.M.D_{Re}, Re);$$

$$Ph \equiv \#(D_{Ph}.M.D_{Re});$$

$$Ph \equiv \eta(D_{Ph}.M.D_{Re}, Re)$$

Б. Неопровержимый запрос от Re : Ph содержит неопровержимые и неповрежденные доказательства того, что Re запросил платеж (в определенной сумме в определенное время, в определенном месте и т.д.). Если Re опровергает платеж, Ph может предъявить подлежащее передаче доказательство арбитру:

- обеспечение защиты авторства в отношении контента:

$$Ph \equiv \phi(Ph, \mathbf{G}(M, Re));$$

- обеспечение фиксации авторства в отношении целостности контента:

$$Ph \equiv \phi(Ph, \eta(M, Re)).$$

В. Неопровержимый платеж от Ph : Ph санкционировал платеж в ответ на запрос Re . Ph предоставил такую информацию, как номер PAN и сумму, которую он разрешил выплатить, а также свою готовность связать этот платеж с запросом Re . Re хранит это неопровержимое доказательство. Если Ph опровергает какие-либо сведения о сумме, PAN, продавце, времени и месте платежа, тогда Re может предоставить передаваемое доказательство честности перевода:

$$Re \equiv \phi(Re, \mathbf{G}(M.D_{Ph}, Ph));$$

$$Re \equiv \phi(Re, \eta(M.D_{Ph}, Ph)).$$

Г. Неопровержимое подтверждение платежа: Кроме того, Re не может позже опровергнуть тот факт, что он не получил приемлемый платеж от Ph . Ph может предъявить передаваемое доказательство арбитру о принятии Re оплаты от Ph :

$$Ph \equiv \phi(Ph, \mathbf{G}(D_{Ph}.M.D_{Re}, Re));$$

$$Ph \equiv \phi(Ph, \eta(D_{Ph}.M.D_{Re}, Re)).$$

Программная реализация этого алгоритма показала, что протокол успешно завершился. По завершении выполнения алгоритма все постусловия выполняются. Таким образом, мы

заключаем, что P_D является корректным. Это означает, что протокол P_D выполняет набор требований безопасности, отраженных в указанных предварительных условиях и постусловиях.

Далее рассмотрим две атаки, полученные путем применения функции преобразования F_A на P_D .

Далее рассмотрим подслушивающего злоумышленника, который прослушивает все передачи между Re и Ph , не внося никаких изменений в передаваемые сообщения. Эта атака на P_D обозначается как $P_{D.attack}$ – протокол атаки на протокол P_D . Наша цель – убедиться, что соблюдены требования к секретности конфиденциальной информации. Диаграмма последовательности $P_{D.attack}$ представлена на рис. 45.

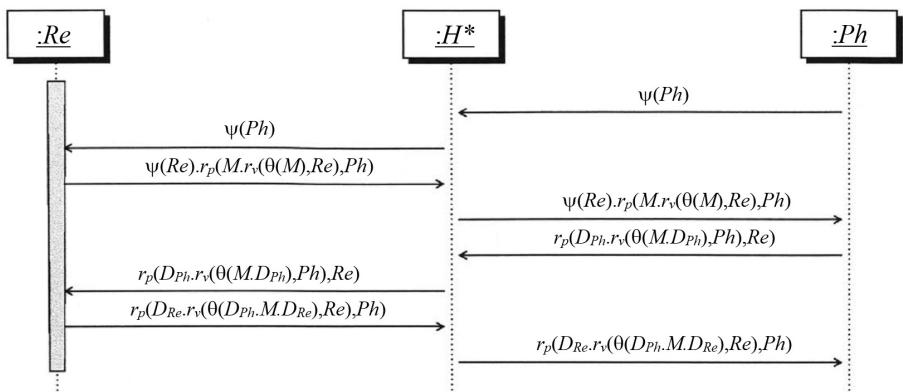


Рис. 45. Схема последовательности UML для протокола атаки $P_{D.attack}$.

Набор исходных предположений такой же, как и для P_D , за исключением того, что злоумышленник, обозначенный как H^* , знает открытые ключи центра сертификации N , Re и Ph . Таким образом, имеем

$$H^* \equiv \gamma(N, k_{N.open}),$$

$$\begin{aligned} H^* &\equiv \gamma(Ph, k_{ph.open}), \\ H^* &\equiv \gamma(Re, k_{Re.open}). \end{aligned}$$

Кроме того, предполагаем, что N и Re заслуживают доверия для злоумышленника:

$$H^* \equiv N \rightarrow \text{trustworthy};$$

$$H^* \equiv Re \rightarrow \text{trustworthy}.$$

Постусловия для атаки злоумышленника на P_D такие же, как для P_D , поскольку H^* не изменяет никакие передачи, и все участники P_D полностью участвуют в $P_{D.\text{attack}}$. Последовательность передач транзакций и предварительные условия представлены в табл. 16.

Таблица 16
Предварительные условия и обмен сообщениями при атаке $P_{D.\text{attack}}$

| Последовательность | Предварительные условия | Обмен сообщениями |
|--------------------|---|--|
| 1 | Нет | $Ph \rightarrow H^*: \psi(Ph)$ |
| 2 | Нет | $H^* \rightarrow Re: \psi(Ph)$ |
| 3 | $Re \equiv \gamma(Ph, k_{Ph.\text{open}})$ | $Re \rightarrow H^*: \psi(Re).r_p(M.r_v(\theta(M), Re), Ph)$ |
| 4 | Нет | $H^* \rightarrow Ph: \psi(Re).r_p(M.r_v(\theta(M), Re), Ph)$ |
| 5 | $Ph \equiv \gamma(Re, k_{Re.\text{open}})$, $Ph \equiv \mathbf{G}(M, Re)$, $Ph \equiv \#(M)$, $Ph \equiv \eta(M, Re)$, $Ph \equiv \mathfrak{R}(M, Ph)$, $Ph \equiv w_p(M, \delta(ID_{Ph}))$ | $Ph \rightarrow H^*: r_p(D_{Ph}.r_v(\theta(M.D_{Ph}), Ph), Re)$ |
| 6 | Нет | $H^* \rightarrow Re: r_p(D_{Ph}.r_v(\theta(M.D_{Ph}), Ph), Re)$ |
| 7 | $Re \equiv \mathbf{G}(M.D_{Ph}, Ph)$, $Re \equiv \#(M.D_{Ph})$, $Re \equiv \eta(M.D_{Ph}, Ph)$ | $Re \rightarrow H^*: r_p(D_{Re}.r_v(\theta(D_{Ph}.M.D_{Re}), Re), Ph)$ |
| 8 | Нет | $H^* \rightarrow Ph: r_p(D_{Re}.r_v(\theta(D_{Ph}.M.D_{Re}), Re), Ph)$ |

Набор утверждений злоумышленников моделирует секретность с помощью отрицаний трех формул. Они утверждают, что злоумышленник не может восстановить M , D_{Re} или D_{Ph} .

1. Злоумышленник H^* не может восстановить сообщение транзакционного запроса M (секретность M)

$$\neg H^* \equiv \Xi(H^*, M).$$

2. Злоумышленник H^* не может восстановить сообщение подтверждения транзакции D_{Ph} (секретность на D_{Ph}).

$$\neg H^* \equiv \Xi(H^*, D_{Ph}).$$

3. Злоумышленник H^* не может восстановить сообщение D_{Re} (Секретность на D_{Re})

$$\neg H^* \equiv \Xi(H^*, D_{Re}).$$

При программной реализации атаки для протокола $P_{D.attack}$. Протокол был доведен до конца, и все пост-условия и утверждения злоумышленников были действительны. Алгоритм возвращает истину и, следовательно, P_D является правильным при атаке $P_{D.attack}$.

В $P_{D.attack}$ мы вставили злоумышленника, который прослушивал все сообщения, которыми обменивались Re и Ph . Программная реализация протокола P_D показала, что злоумышленник не смог раскрыть текстовые сообщения M , D_{Re} или D_{Ph} . При этом атака не была обнаружена, а все постусловия и утверждения злоумышленников были выполнены. Это означает, что исходный протокол P_D защищен от этого специфического типа атак с перехватом, поскольку обеспечивает секретность передаваемой транзакционной информации.

Теперь смоделируем другую атаку, в которой злоумышленник H^* пытается выдать себя за Ph , воспроизводя его сертификат и некоторые из старых сообщений H^* , перехваченных в предыдущем сообщении (M'), аналогичном $P_{D.attack}$. Обозначим атаку как $P_{D2.attack}$. Последовательность передачи сообщений на языке UML представлена на рис. 46.

В этом протоколе Re предполагает, что разговаривает с Ph . Вместо этого Re взаимодействует с злоумышленником H^* , при этом Ph заблокирован от получения каких-либо сообщений. После получения сигнала $client-hello$, воспроизведения сертификата Ph от H^* , Re возвращает свой сертификат, а также $\tau_p(M.r_v(\Theta(M)), Re), Ph$.

Поскольку H^* не знает закрытый ключ Ph , H^* не может управлять этим сообщением, а H^* не может видеть его содержимое. Вместо этого H^* отправляет повтор третьей передачи в предыдущем прогоне P_D между Re и Ph , обозначенный как $r_p(D_{Ph}.r_v(\theta(M'.D_{Ph}), Ph), Re)$, где M' отличается от M , т.к. M изменился, и D_{Ph} использовалась в качестве платежной информации, которую использовал Ph , в надежде, что Re примет платежную информацию. Если протокол завершается, Re отправляет свою подпись через $D_{Ph}.M.D_{Re}$, где D_{Ph} был получен Re ; M и D_{Re} были построены Re .

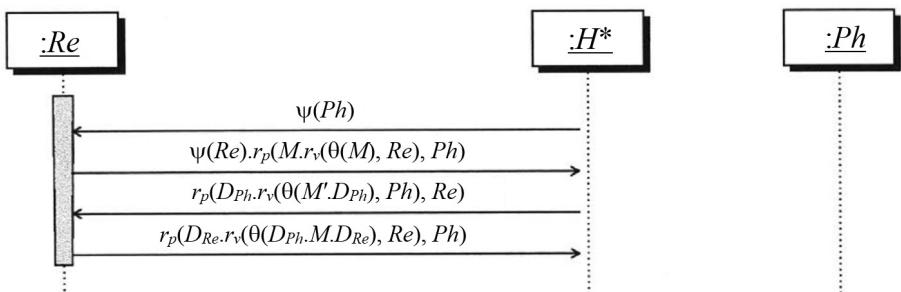


Рис. 46. Схема последовательности UML для протокола атаки $P_{D2.attack}$

Исходные предположения такие же, как и для $P_{D.attack}$. Предварительные условия и передача сообщений представлены в табл. 17.

Таблица 17

Протокол атаки $P_{D2.attack}$ для протокола оплаты мобильного считывателя

| Последовательность | Предварительные условия | Обмен сообщениями |
|--------------------|--|---|
| 1 | Нет | $H^* \rightarrow Re:\psi(Ph)$ |
| 2 | $Re \equiv \gamma(Ph, k_{Ph.open})$ | $Re \rightarrow H^*:\psi(Re).r_p(M.r_v(\theta(M), Re), Ph)$ |
| 3 | Нет | $H^* \rightarrow Re:r_p(D_{Ph}r_v(\theta(M'.D_{Ph}), Ph), Re)$ |
| 4 | $Re \equiv \mathbf{G}(M.D_{Ph}, Ph)$, $Re \equiv \#(M.D_{Ph})$, $Re \equiv \eta(M.D_{Ph}, Ph)$ | $Re \rightarrow H^*:r_p(D_{Re}.r_v(\theta(D_{Ph}.M.D_{Re}), Re), Ph)$ |

Все предварительные условия, связанные с Ph , удалены, поскольку Ph не участвовал в этом протоколе атаки. Все предварительные условия для передачи сообщений, в которых злоумышленник является отправителем, являются пустыми. Более того, три предварительных условия для сообщения 4 – это утверждения о M , а не о M' . Это связано с тем, что эти утверждения являются признанием Ph элемента M , созданного Re , а не признанием M элемента Re , получаемого от Ph . Они утверждают, что на этом этапе протокола Re должен иметь возможность вывести уверенность в том, что M , построенное Re , является подтверждено Ph . Ph признаёт другой M' и не изменяет цель подтверждения, которую ищет Re .

В этом случае имеет следующие постусловия:

1. Re убежден в неопровергимых доказательствах подтверждения сделки $Ph.M.D_{Ph}$:

$$Re \equiv \phi(Re, \mathbf{G}(M.D_{Ph}, Ph)).$$

2. Re убежден в целостность неопровергимых доказательств:

$$Re \equiv \phi(Re, \eta(M.D_{Ph}, Ph)).$$

Наконец, утверждения злоумышленника этого протокола атаки:

1. Злоумышленник H^* не может восстановить сообщение транзакционного запроса M (секретность M):

$$\neg H^* \equiv \Xi(H^*, M).$$

2. Злоумышленник H^* не может восстановить D_{Re} , т.е. квитанцию:

$$\neg H^* \equiv \Xi(H^*, D_{Re}).$$

Эти утверждения злоумышленника отражают требования конфиденциальности, которые были наложены на платежную информацию M и D_{Re} ; два утверждения были построены и отправлены честным принципалам Re .

Программная реализация алгоритма $P_{D2.attack}$ показала, что протокол P_D остановлен после того, как Re получил

$r_p(D_{Ph}, r_v(\theta(M'.D_{Ph}), Ph), Re)$ из-за нарушения предварительного условия $Re \equiv \mathbf{G}(M.D_{Ph}, Ph)$, где M сгенерирован в начале протокола и было вызовом Re . Более того, на момент нарушения утверждения злоумышленника были действительными. То есть злоумышленник не может восстановить M или D_{Re} . Это означает, что исходный протокол может обнаруживать этот тип атаки, и на момент обнаружения никакие требования к секретности не были нарушены.

Вывод: протокол P_D ведет себя корректно при атаке $P_{D2.attack}$.

В этом параграфе был представлен протокол бесконтактных платежей с помощью мобильного считывателя с высокой степенью защиты. Две ключевые особенности этого протокола, которых нет в EMV ДАД, – это секретность платежной информации (аналогичная ДВК в EMV) и обработка неотказуемости как для продавца, так и для клиента. Мы формализовали протокол P_D и показали, что он работает корректно. Были рассмотрены две формы атаки: одну на секретность ($P_{D.attack}$) и другую за выдачу себя за другое лицо ($P_{D2.attack}$). Пришли к выводу, что протокол-кандидат P_D защищен от этих двух атак в отношении набора начальных предположений, а также предварительных и последующих условий.

4.3. Протоколы оплаты с использованием мобильных токенов

В этом параграфе рассматривается мобильный протокол бесконтактных платежей, который обеспечивает высокий уровень безопасности, а также дополнительные функции, такие как предотвращение отказа от авторства. Обсуждается его формализация и анализ.

Рассматриваемый протокол мобильных платежей позволяет NFC-телефону с возможностью подключения к серверу приобретать услуги или товары, идентифицируемые пассивными тегами (смарт-картами). В отличие от протокола оплаты на основе считывателя (параграф 4.2), телефон теперь отвечает за сбор доказательств транзакции, а также за их отправку на платежный сервер.

Первоначально рассмотрим протокол мобильных платежей, который не использует NFC-считыватель. Протокол, обозначенный как P_B , позволяет NFC-телефону с возможностью передачи данных сервера покупать услуги или товары, идентифицированные пассивными тегами (смарт-картами). В отличие от протокола оплаты на основе считывателя, телефон теперь отвечает за сбор доказательств транзакции, а также за их отправку на платежный сервер. Пассивные теги размещаются с описанием покупаемых услуг или товаров. Они обесточены и общаются с телефоном по схеме модуляции нагрузки. Популярными примерами являются умные плакаты, где на стене вешают пассивные теги для рекламы услуг, а также умные дисплеи в торговом центре, где покупатели могут нажать, чтобы заплатить за отображаемые товары. После этого покупатели могут забрать купленные товары при выходе из магазина или торгового центра.

Основное различие между этим типом платежного протокола P_B и протоколом на основе считывателя P_D заключается в том, что пассивные теги не могут представлять произвольные услуги или товары. Вместо этого они представляют собой фиксированный набор услуг или товаров. Преимущество состоит в том, что зачастую к продаваемым товарам не предъявляются требования конфиденциальности – каждый может их проверить. После того, как телефон коснется пассивной метки, метка представляет «меню» товаров или услуг для покупки. Такое же «меню» возвращается при каждом таком запросе. Единственное требование, которое

предъявляет платежный сервер, – это то, что идентифицированный телефон (необязательно с информацией о выбранном платежном инструменте) согласился приобрести определенную услугу в определенное время.

Предполагаем, что пассивные теги могут выполнять примитивные криптографические операции, такие как шифрование и цифровая подпись. Кроме того, у тегов достаточно памяти для хранения своих сертификатов. Предполагается, что хранение закрытых ключей и секретов в этих тегах аналогично защищенному элементу, используемому в телефоне. Закрытые ключи и любые секреты не могут быть прочитаны или изменены злоумышленниками.

В этом протоколе задействованы три типа принципалов: уникальный платежный сервер Se , количество действующих NFC-телефонов Ph и количество действующих смарт-карт Sm . Предполагаем, что Se считается заслуживающим доверия как Ph , так и Sm . Ph и Sm не считаются заслуживающими доверия Se . Предположим, что выдающий орган управляет сертификатами для всех участников.

Высокоуровневое описание протокола выглядит следующим образом: после того, как Se получает сигнал *client-hello* от Ph , Se генерирует запрос $M_{Se} = \chi(N_{Se}).\Theta(T_{Se}).\delta(ID_{Ph})$, где $\chi(N_{Se})$ – глобально уникальный одноразовый номер, а $\Theta(T_{Se})$ – временная метка, сгенерированная сервером. Сообщение M_{Se} и его подтверждение происхождения затем отправляется в Ph . Ph затем пересыпает то же сообщение Sm для подписи. После получения подписи от Sm , Ph выполняет вторую подпись под подписью, возвращенной Sm . Полученная в результате двойная подпись передается обратно в Se для проверки. Наконец, если все верно, квитанция возвращается Ph . Общая последовательность потока сообщений представлена на рис. 47.

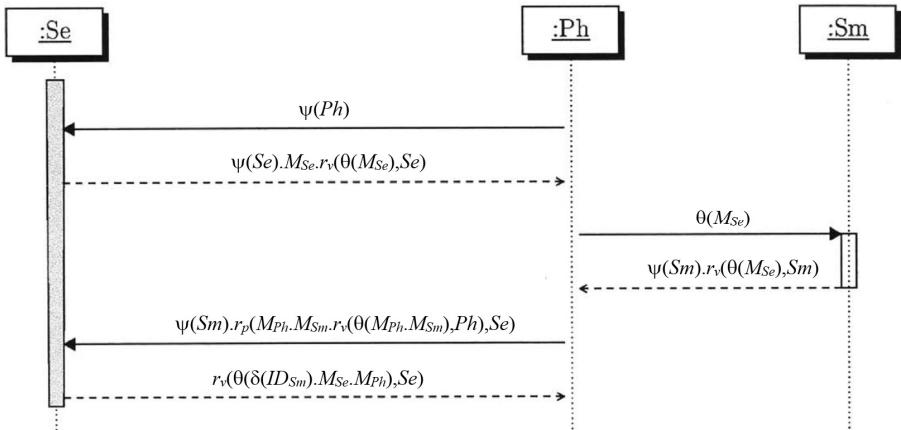


Рис. 47. Схема последовательности UML для телефона NFC, протокол P_B оплаты смарт-картой

Ниже представлена формализованная спецификация P_B , включая исходные предположения, передачу сообщений, предварительные условия, а также постусловия.

1. И Ph , и Sm считают, что Se заслуживает доверия. Ph , Sm и Se считают, что это транзакция заслуживает доверия. Таким образом:

$$Ph \equiv Se \rightarrow \text{trustworthy};$$

$$Sm \equiv Se \rightarrow \text{trustworthy};$$

$$Ph \equiv N \rightarrow \text{trustworthy};$$

$$Sm \equiv N \rightarrow \text{trustworthy};$$

$$Se \equiv N \rightarrow \text{trustworthy}.$$

2. Ph , Sm и Se имеют доступ к своим личным и открытым ключам:

$$Ph \equiv \gamma(Ph, k_{Ph.open});$$

$$Sm \equiv \gamma(Sm, k_{Sm.open});$$

$$Se \equiv \gamma(Se, k_{Se.open});$$

$$Ph \equiv \psi(Ph, k_{Ph.lock});$$

$$Sm \equiv \psi(Sm, k_{Sm.lock});$$

$$Se \equiv \psi(Se, k_{Se.lock}).$$

3. Ph , Sm и Se имеют root-сертификаты от центра сертификации и, следовательно, знают открытый ключ N :

$$Ph \equiv \gamma(N, k_{N.open});$$

$$Sm \equiv \gamma(N, k_{N.open});$$

$$Se \equiv \gamma(N, k_{N.open}).$$

4. ID_{Ph} , ID_{Se} , ID_{Sm} отсутствуют в списке аннулированных сертификатов, а отметки времени на сертификатах Ph , Se и Sm соответствуют текущему времени:

$$Ph \equiv \xi(\delta(ID_{Se}));$$

$$Sm \equiv \xi(\delta(ID_{Se}));$$

$$Se \equiv \xi(\delta(ID_{Se}));$$

$$Se \equiv \xi(\delta(ID_{Ph}));$$

$$Sm \equiv \xi(\delta(ID_{Ph}));$$

$$Ph \equiv \xi(\delta(ID_{Ph}));$$

$$Se \equiv \xi(\delta(ID_{Sm}));$$

$$Ph \equiv \xi(\delta(ID_{Sm}));$$

$$Sm \equiv \xi(\delta(ID_{Sm}));$$

$Ph \equiv \Delta(\Theta(T_{1Se}, T_{2Se}))$;

$Sm \equiv \Delta(\Theta(T_{1Se}, T_{2Se}))$;

$Se \equiv \Delta(\Theta(T_{1Se}, T_{2Se}))$;

$Se \equiv \Delta(\Theta(T_{1Ph}, T_{2Ph}))$;

$Sm \equiv \Delta(\Theta(T_{1Ph}, T_{2Ph}))$;

$Ph \equiv \Delta(\Theta(T_{1Ph}, T_{2Ph}))$;

$Ph \equiv \Delta(\Theta(T_{1Sm}, T_{2Sm}))$;

$Se \equiv \Delta(\Theta(T_{1Sm}, T_{2Sm}))$;

$Sm \equiv \Delta(\Theta(T_{1Sm}, T_{2Sm}))$,

где ξ – невозможность отзыва сертификата.

5. Se может восстановить одноразовый номер и метку времени, сгенерированные им самим, $\chi(N_{Se}).\Theta(T_{Se})$. Ph может восстановить сообщение M_{Ph} , сгенерированное им самим:

$Se \equiv \Xi(Se, \chi(N_{Se}).\Theta(T_{Se}))$;

$Ph \equiv \Xi(Ph, M_{Ph})$.

6. Se и Ph полагают, что $\Theta(T_{Se})$, созданное Se , является новой отметкой времени:

$Se \equiv \Delta(\Theta(T_{Se}))$;

$Ph \equiv \Delta(\Theta(T_{Se}))$.

7. Se считает, что в данный момент времени $\chi(N_{Se})$ изменился:

Описание протокола P_B :

Шаг 1. Предварительных нет условий. Ph отправляет свой сертификат Se :

$$Ph \rightarrow Se:\psi(Ph).$$

Шаг 2. Предварительное условие – Se необходимо проверить сертификат Ph и тем самым получить открытый ключ Ph :

$$Se \equiv \gamma(Ph, k_{Ph.open}).$$

Se генерирует M_{Se} , который состоит из одноразового номера $\chi(N_{Se})$, отметки времени сервера $\Theta(T_{Se})$, а также тега получателя $\delta(ID_{Ph})$, который Se получает от $\psi(Ph)$, полученного в предыдущем сообщении. Se не требует конфиденциальности по отношению к M_{Se} . Затем он отправляет свой сертификат $\psi(Se)$, M_{Se} и свою подпись над хешем M_{Se} в Ph . Предполагаем, что Se отслеживает все M_{Se} , которые он создает для определенного Ph :

$$Se \rightarrow Ph:\psi(Se).M_{Se}.r_v(\theta(M_{Se}), Se).$$

Шаг 3. Предварительные три условия:

1. Ph должен извлечь открытый ключ Se и убедиться, что M_{Se} исходит от Se :

$$Ph \equiv \gamma(Se, k_{Se.open});$$

$$Ph \equiv \mathbf{G}(M_{Se}, Se).$$

2. Ph должен гарантировать, что M_{Se} изменился и неповрежденный:

$$Ph \equiv \#(M_{Se});$$

$$Ph \equiv \eta(M_{Se}, Se).$$

3. Ph должен убедиться, что M_{Se} подтверждает ID , принадлежащий Ph , чтобы он был предназначен для Ph , а не для других Ph' :

$$Ph \equiv w_p(M_{Se}, \delta(ID_{Ph})).$$

Затем Ph отправляет хешированный M_{Se} в Sm :

$$Ph \rightarrow Sm: \theta(M_{Se}).$$

Шаг 4. Предварительных нет условий. Sm вычисляет свою подпись по хешу M_{Se} и возвращает подпись, а также свой сертификат.

$$Sm \rightarrow Ph: \psi(Sm).r_v(\theta(M_{Se}), Sm).$$

Шаг 5. Предварительные два условия:

1. Ph должен сначала проверить сертификат Sm (срок действия, срок действия, CRL и т.д.) И извлечь открытый ключ Sm .

$$Ph \equiv \gamma(Sm, k_{Sm.open}).$$

2. Ph должен проверить, что Sm подтвердил (произнес) M_{Se} , чтобы исключить любую повторную атаку. Ph также проверяет, не повреждена ли M_{Se} :

$$Ph \equiv \mathbf{G}(M_{Se}, Sm);$$

$$Ph \equiv \eta(M_{Se}, Sm).$$

Пусть $M_{Sm} \equiv r_v(\theta(M_{Se}), Sm)$; M_{Sm} можно рассматривать как недавно созданный и глобально уникальный платежный запрос, в котором услуга и продавец идентифицируются с помощью Sm , а время и покупатель идентифицируются с помощью M_{Se} . Затем Ph готовит свои платежные реквизиты, такие как выбранный номер PAN, обозначаемый строкой M_{Ph} . Ph подписывает хеш $M_{Ph}.M_{Sm}$ и тем самым эффективно связывает свое согласие с M_{Ph} и M_{Sm} . Наконец, Ph передает сертификат Sm , зашифрованную строку через $M_{Ph}.M_{Sm}$ и подписанный хеш:

$$Ph \rightarrow Se: \psi(Sm).r_p(M_{Ph}.r_v(\theta(M_{Ph}.M_{Sm}), Ph), Se).$$

Шаг 6. Предварительные три условия – сервер должен проверить ряд элементов перед отправкой квитанции:

1. Сначала сервер проверяет сертификат Sm , $\psi(Sm)$ и извлекает открытый ключ Sm :

$$Se \equiv \gamma(Sm, k_{Sm.open}).$$

2. Затем Se извлекает M_{Se} , которое было отправлено Ph , из своей базы данных. Se проверяет, что M_{Se} подтвержден Sm и подтверждение не повреждено:

$$Se \equiv G(M_{Se}, Sm);$$

$$Se \equiv \eta(M_{Se}, Sm).$$

3. Se необходимо убедиться, что $M_{Ph}.M_{Sm}$ произносится Ph и не поврежден:

$$Se \equiv G(M_{Ph}.M_{Sm}, Ph);$$

$$Se \equiv \eta(M_{Ph}.M_{Sm}, Ph).$$

Далее Se выдает квитанцию. Он создает подпись поверх хеша $\delta(ID_{Sm})$, M_{Se} и M_{Ph} , которая эффективно подтверждает транзакцию между $\delta(ID_{Ph})$ и $\delta(ID_{Sm})$ в момент времени $\Theta(T_{Se})$ с платежной информацией, как описано в M_{Ph} :

$$Se \rightarrow Ph:r_v(\theta(\delta(ID_{Sm}).M_{Se}.M_{Ph}), Se).$$

Шаг 7. Алгоритм работы протокола закончен.

Постусловия:

1. Утверждения, принадлежащие Ph , по последнему сообщению, которое получил Ph . Ph утверждает, что он происходит от Se , он не поврежден, и делает вывод:

$$Ph \equiv G(\delta(ID_{Sm}).M_{Se}.M_{Ph}, Se);$$

$$Ph \equiv \eta(\delta(ID_{Sm}).M_{Se}.M_{Ph}, Se).$$

2. Неопровергимый платеж от Ph : Ph предоставил свою платежную информацию M_{Ph} и привязал себя к услуге и времени. Если Ph позже опровергнет, что он согласился оплатить услуги, идентифицированные идентификатором смарт-карты $\delta(ID_{Sm})$ в момент времени $\Theta(T_{Se})$ с платежными реквизитами M_{Ph} , платежный сервер Se может предоставить неопровергимые доказательства арбитру. Формально это:

- Se предлагает посредством свидетельства M_{Sm} и строки M_{Se} передаваемое доказательство того, что Sm связывает свою идентичность с M_{Se} :

$$Se \equiv \phi(Se, \mathbf{G}(M_{Se}, Sm));$$

$$Se \equiv \phi(Se, \eta(M_{Se}, Sm));$$

- Se предлагает передаваемое доказательство того, что Ph связал свою личность с платежной информацией M_{Ph} и тем же M_{Sm} ,

$$Se \equiv \phi(Se, \mathbf{G}(M_{Ph}.M_{Sm}, Ph));$$

$$Se \equiv \phi(Se, \eta(M_{Ph}.M_{Sm}, Ph)).$$

таким образом, доказывает, что Ph согласился приобрести услугу, обозначенную $\delta(ID_{Sm})$, с платежной информацией M_{Ph} и на конкретное время, как указано в $\Theta(T_{Se})$.

3. Неопровергимое подтверждение платежа. Кроме того, Se не может позже опровергнуть, что он не получил приемлемого платежа от Ph . Ph может предъявить передаваемое доказательство арбитру о принятии Se оплаты Ph . Доказательства связывают согласие Se с личностями Ph и Sm , а также с платежной информацией и временем транзакции.

$$Ph \equiv \phi(Ph, \mathbf{G}(\delta(ID_{Sm}).M_{Se}.M_{Ph}, Se));$$

$$Ph \equiv \phi(Ph, \eta(\delta(ID_{Sm}).M_{Se}.M_{Ph}, Se)).$$

Программная реализация алгоритм P_B успешно отработала до конца без нарушения предварительных условий. Кроме того, все

постусловия были действительны на момент завершения. P_B является надёжным протоколом. Это означает, что протокол P_B соответствует набору бизнес-требований и целей безопасности, отраженных в предварительных и постусловиях. Кроме того, он соответствует предпосылкам вышеупомянутого набора исходных предположений. Все исходные предположения должны быть выполнены, прежде чем можно будет утверждать, что P_B надёжный протокол.

Теперь рассмотрим две возможные атаки, полученные при применении функции преобразования F_A к P_B .

I. Секретная атака.

Представляем злоумышленника, который прослушивает весь передаваемый трафик, не внося никаких изменений в передачу. Это перехватывающая атака через P_B , обозначенная как $P_{B.Attack}$. Диаграмма последовательностей для $P_{B.Attack}$ представлена на рис. 48.

Протокол аналогичен на P_B , за исключением того, что каждая передача в P_B теперь проходит через злоумышленника H^* .

Набор исходных предположений аналогичен, как и для P_B , за исключением того, что H^* теперь знает открытый ключ центра сертификации N , Se , Ph и Sm . Таким образом, мы имеем:

$$H^* \equiv \gamma(N, k_{N.open});$$

$$H^* \equiv \gamma(Se, k_{Se.open});$$

$$H^* \equiv \gamma(Ph, k_{Ph.open});$$

$$H^* \equiv \gamma(Sm, k_{Sm.open}).$$

Кроме того, предполагаем, что N и Se заслуживают доверия злоумышленника:

$$H^* \equiv N \rightarrow \text{trustworthy};$$

$$H^* \equiv Se \rightarrow \text{trustworthy}.$$

Постусловия такие же, как и в исходном протоколе P_B , поскольку H^* не изменяет никаких сообщений, и Se , Ph и Sm участвуют в $P_{B.Attack}$ до завершения своих соответствующих ролей.

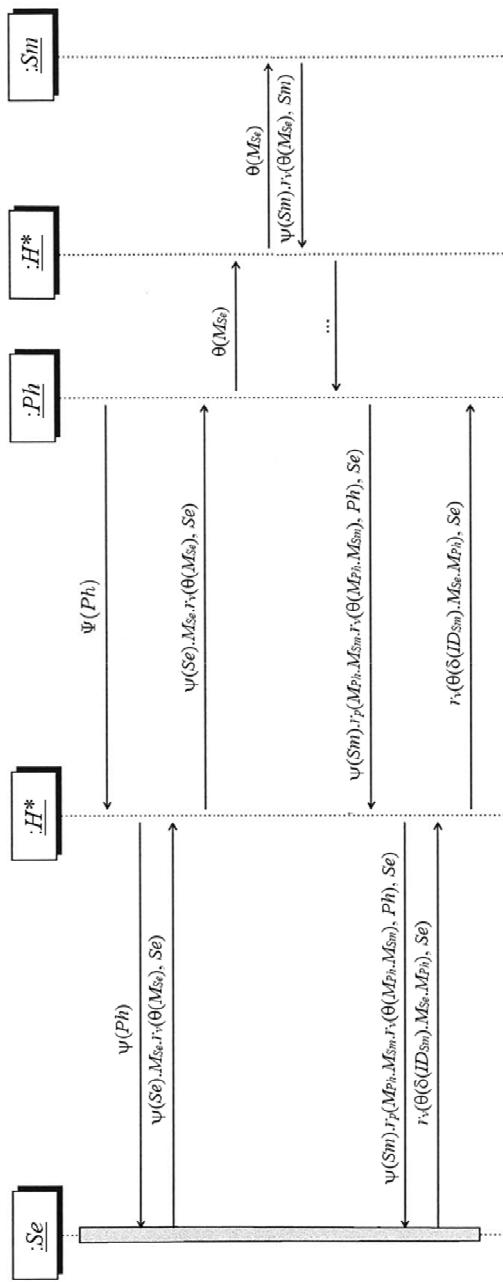


Рис. 48. Схема последовательности UML для протокола $P_{B.Attack}$, случай с подслушивателем

Предварительные условия и последовательность передачи сообщений представлены в табл. 18.

Таблица 18

Предварительные условия и обмен сообщениями для протокола атаки $P_{B.Attack}$

| Последовательность | Предварительные условия | Обмен сообщениями |
|--------------------|---|---|
| 1 | Нет | $Ph \rightarrow H^*: \psi(Ph)$ |
| 2 | Нет | $H^* \rightarrow Se: \psi(Ph)$ |
| 3 | $Se \equiv \gamma(Ph, k_{Ph.open})$ | $Se \rightarrow H^*: \psi(Se).M_{Se}.r_v(\theta(M_{Se}), Se)$ |
| 4 | Нет | $H^* \rightarrow Ph: \psi(Se).M_{Se}.r_v(\theta(M_{Se}), Se)$ |
| 5 | $Ph \equiv \gamma(Se, k_{Se.open}),$ $Ph \equiv \mathbf{G}(M_{Se}, Se),$ $Ph \equiv \#(M_{Se}),$ $Ph \equiv \eta(M_{Se}, Se),$ $Ph \equiv w_p(M_{Se}, \delta(ID_{Ph}))$ | $Ph \rightarrow H^*: \theta(M_{Se})$ |
| 6 | Нет | $H^* \rightarrow Sm: \theta(M_{Se})$ |
| 7 | Нет | $Sm \rightarrow H^*: \psi(Sm).r_v(\theta(M_{Se}), Sm)$ |
| 8 | Нет | $H^* \rightarrow Ph: \psi(Sm).r_v(\theta(M_{Se}), Sm)$ |
| 9 | $Ph \equiv \gamma(Sm, k_{Sm.open}),$ $Ph \equiv \mathbf{G}(M_{Se}, Sm),$ $Ph \equiv \eta(M_{Se}, Sm)$ | $Ph \rightarrow$ $H^*: \psi(Sm).r_p(M_{Ph}.M_{Sm}.r_v(\theta(M_{Ph}.M_{Sm}), Ph), Se)$ |
| 10 | Нет | $H^* \rightarrow Se:$ $\psi(Sm).r_p(M_{Ph}.M_{Sm}.r_v(\theta(M_{Ph}.M_{Sm}), Ph), Se)$ |
| 11 | $Se \equiv \gamma(Sm, k_{Sm.open}),$ $Se \equiv \mathbf{G}(M_{Se}, Sm),$ $Se \equiv \eta(M_{Se}, Sm),$ $Se \equiv \mathbf{G}(M_{Ph}.M_{Sm}, Ph),$ $Se \equiv \eta(M_{Ph}.M_{Sm}, Ph)$ | $Se \rightarrow H^*: r_v(\theta(\delta(ID_{Sm}).M_{Se}.M_{Ph}), Se)$ |
| 12 | Нет | $H^* \rightarrow Ph: r_v(\theta(\delta(ID_{Sm}).M_{Se}.M_{Ph}), Se)$ |

Последовательность передач теперь включает в себя те последовательности, которые были получены и переданы нарушителем H^* . Утверждения злоумышленника, которые мы требуем для $P_{B.Attack}$, просто заключаются в том, что перехватчик не получает реквизиты платежа, отправленные от Ph :

$$\neg H^* \equiv \Xi(H^*, M_{Ph}).$$

Предполагаем, что Sm и Se не предъявляют никаких требований к конфиденциальности. Se генерирует новую M_{Se} , которая может быть публично видна, а Sm может раскрыть свою идентичность, а также $\Theta(M_{Se})$.

Программная реализация алгоритма $P_{B.Attack}$ показала, что протокол успешно завершен, и все поступления и утверждения злоумышленников действительны. Программа возвращает *true*, и, таким образом, протокол P_B надежен и безопасен при атаке $P_{B.Attack}$.

II. Атака с использованием ретрансляции.

Далее моделируем ретрансляционную атаку на P_B , обозначенную как $P_{B2.Attack}$. Схема последовательности UML представлена на рис. 49.

Злоумышленник H^* не позволяет Sm получить $\Theta(M_{Se})$, и H^* передает $\Theta(M_{Se})$ на вторую аутентифицированную смарт-карту Sm' для подписи. Затем H^* возвращает эту подпись Ph . Ph думает, что он прослушивал Sm , но на самом деле он получил доказательства прослушивания для другой смарт-карты, возможно, в другом географическом месте. Обратите внимание, что Sm не участвует в этом протоколе.

Исходные предположения такие же, как у $P_{B.Attack}$. Последовательность передач и их предварительные условия представлены на табл. 19.

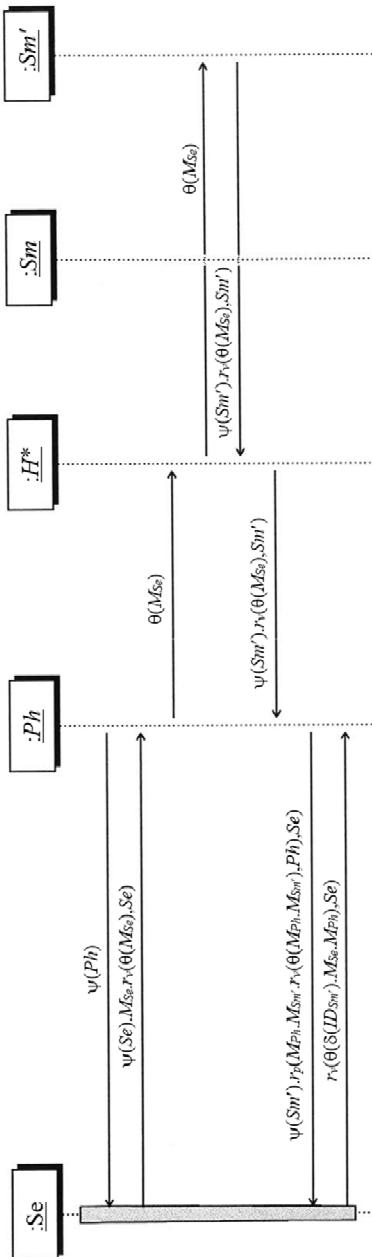


Рис. 49. Схема последовательности UML для атаки $P_{B2.Attack}$, случай со второй аутентифицированной смарт-картой.

Таблица 19

Предварительные условия и обмен сообщениями для протокола атаки $P_{B2.Attack}$

| Последовательность | Предварительные условия | Обмен сообщениями |
|--------------------|---|--|
| 1 | Нет | $Ph \rightarrow Se:\psi(Ph)$ |
| 2 | $Se \equiv \gamma(Ph, k_{Ph.open})$ | $Se \rightarrow Ph:\psi(Se).M_{Se}.r_v(\theta(M_{Se}), Se)$ |
| 3 | $Ph \equiv \gamma(Se, k_{Se.open})$, $Ph \equiv \mathbf{G}(M_{Se}, Se)$, $Ph \equiv \#(M_{Se})$, $Ph \equiv \eta(M_{Se}, Se)$, $Ph \equiv w_p(M_{Se}, \delta(ID_{Ph}))$ | $Ph \rightarrow H^*:\theta(M_{Se})$ |
| 4 | Нет | $H^* \rightarrow Sm':\theta(M_{Se})$ |
| 5 | Нет | $Sm' \rightarrow H^*:\psi(Sm').r_v(\theta(M_{Se}), Sm')$ |
| 6 | Нет | $H^* \rightarrow Ph:\psi(Sm').r_v(\theta(M_{Se}), Sm')$ |
| 7 | $Ph \equiv \gamma(Sm', k_{Sm'.open})$, $Ph \equiv \mathbf{G}(M_{Se}, Sm')$, $Ph \equiv \eta(M_{Se}, Sm')$ | $Ph \rightarrow$ $Se:\psi(Sm').r_p(M_{Ph}.M_{Sm'}.r_v(\theta(M_{Ph}.M_{Sm'}), Ph), Se)$ |
| 8 | $Se \equiv \gamma(Sm', k_{Sm'.open})$, $Se \equiv \mathbf{G}(M_{Se}, Sm')$, $Se \equiv \eta(M_{Se}, Sm')$, $Se \equiv \mathbf{G}(M_{Ph}.M_{Sm'}, Ph)$, $Se \equiv \eta(M_{Ph}.M_{Sm'}, Ph)$ | $Se \rightarrow Ph:r_v(\theta(\delta(ID_{Sm'}).M_{Se}.M_{Ph}), Se)$ |

Предварительные условия для обмена сообщениями 7 и 8 теперь относятся к Sm' . Это связано с тем, что они были семантически связаны со смарт-картой при передаче сообщения 6, которое возвращается злоумышленником, выполнившим перенаправление сообщения (табл. 19). Набор постусловий и утверждений злоумышленника модифицируется в соответствии с этим изменением по сравнению с исходным протоколом.

Далее представлены постусловия, которые накладываем на протокол атаки $P_{B2.Attack}$:

1. Утверждения Ph по последнему сообщению, которое получил Ph . Ph утверждает, что он происходит от Se , он не поврежден, и делает вывод $\delta(ID_{Sm'}).M_{Se}.M_{Ph}$:

$$Ph \equiv \mathbf{G}(\delta(ID_{Sm'}).M_{Se}.M_{Ph}, Se);$$

$$Ph \equiv \eta(\delta(ID_{Sm'}).M_{Se}.M_{Ph}, Se).$$

2. Утверждения по общему протоколу по завершении:

- Неопровергимая выплата от Ph :

$$Se \equiv \phi(Se, \mathbf{G}(M_{Se}, Sm'));$$

$$Se \equiv \phi(Se, \eta(M_{Se}, Sm'));$$

$$Se \equiv \phi(Se, \mathbf{G}(M_{Ph}.M_{Sm'}, Ph));$$

$$Se \equiv \phi(Se, \eta(M_{Ph}.M_{Sm'}, Ph)).$$

- Неопровергимое подтверждение платежа:

$$Ph \equiv \phi(Ph, \mathbf{G}(\delta(ID_{Sm'}).M_{Se}.M_{Ph}, Se));$$

$$Ph \equiv \phi(Ph, \eta(\delta(ID_{Sm'}).M_{Se}.M_{Ph}, Se)).$$

3. Утверждение злоумышленника, которое мы накладываем на $P_{B2.Attack}$, состоит в том, что H^* не может восстановить M_{Ph} :

$$\neg H^* \equiv \Xi(H^*, M_{Ph}).$$

Отличие спроектированного протокола P_B от стандартного протокола NFC в том, что до шага 5 в стандартном протоколе Ph не проверяет, является ли Sm предполагаемым продавцом / товаром для покупки. В стандартном протоколе не было указано никаких предварительных условий, отражающих это требование безопасности. Иначе говоря, Ph не заботится, было ли доказательство прослушивания от Sm или какого-то Sm' .

На практике приложение, работающее на телефоне, может представить окно (диалоговую форму) подтверждения и попросить клиента подтвердить, что все, что Ph получает обратно от Sm , является согласованным, прежде чем Ph примет транзакцию. Разработанная модель протокола может обеспечить эту дополнительную меру предосторожности в виде предварительного условия для шага 5 в P_B : $Ph \equiv w_p(\psi(Sm), \delta(ID_{Sm}))$, где $\psi(Sm)$ – это сертификат определенного Sm , который получает Ph , а $\delta(ID_{Sm})$ – это идентификатор смарт-карты (то есть подразумеваемые товары для покупки), которая является ожиданием покупателя.

Это поднимает интересную проблему для некоторых приложений бесконтактных платежей, таких как оформление билетов на общественный транспорт, в которых сервер должен иметь возможность определять географическое положение любого платежного устройства для целей расчета тарифов и аудита. В традиционных протоколах контактных платежей можно предположить, что действительная транзакция между покупателем и продавцом физически происходит на считывающем устройстве продавца (предполагаем, что считыватель карт заслуживает доверия). То есть банковская карта физически присутствовала в считывающем устройстве продавца на основе технологии бесконтактной передачи транзакций на банковском терминале.

С другой стороны, протокол бесконтактных мобильных платежей работает в непосредственной близости, но физический контакт не требуется. Это разделение, продемонстрированное на примере атаки $P_{B2Attack}$, может быть увеличено. Следовательно, в мобильных бесконтактных протоколах мы больше не можем вывести физическое присутствие клиента по транзакции. Сама транзакция все еще может быть действительной, поскольку в простейшей конфигурации требуются только законные отношения между идентифицированным покупателем и продавцом.

Другой интересный случай – когда продавец и покупатель являются злоумышленниками и вступают в сговор (и, таким образом, Se взаимодействует с двумя злоумышленниками). Например, Ph нечестен и игнорирует любые предварительные условия перед тем, как связаться с Se , после того, как он получил сообщение от Sm' , расположенного в другом географическом месте. Se просто проверяет связь между личностью Ph и идентичностью Sm' и записывает, что Ph хотел приобрести услугу, как определено $\delta(ID_{Sm'})$. Se не требует и не

может обеспечить, основываясь на ограниченной информации, что P_h физически находился в Sm' .

При оформлении транзитных билетов с помощью мобильных тегов клиенты входят в систему, нажимая на смарт-карты, установленные на платформах на станциях. Платежный сервер Se имеет бизнес-требование для проверки согласованности физического местоположения клиента и сообщаемых доказательств прослушивания с любым конкретным Sm . В этой ситуации мобильный телефон также может передавать свои аутентифицированные координаты GPS для проверки сервера. Мы можем смоделировать эту проверку согласованности, выполняемую на платежном сервере, введя дополнительную семантику в предикат *close By (GPS coordinates, Sm)*.

Подводя итог, можно указать на сложность разработки платежных протоколов. Протоколы, которые являются безопасными в одном варианте использования, могут не удовлетворять всем требованиям бизнеса и безопасности для другого. Однако с точки зрения анализа протокола представленная система моделирования платежного протокола с разграничением прав достаточно гибкая, чтобы формализовать различные наборы требований, и способна выполнять проверки в полной автоматизации (программная реализация).

ЛИТЕРАТУРА

1. Аверченков, В.И. Аудит информационной безопасности / В.И. Аверченков. – М.: ФЛИНТА, 2011. – 269 с.
2. Аверченков, В.И. Система обеспечения безопасности Российской Федерации / В.И. Аверченков, В.В. Ерохин. – Брянск: БГТУ, 2005. – 120 с.
3. Аверченков, В.И. Системы организационного управления / В.И. Аверченков, В.В. Ерохин. – Брянск: БГТУ, 2006. – 208 с.
4. Алиев, В.С. Информационные технологии и системы финансового менеджмента / В.С. Алиев. – М.: «ФОРУМ»: ИНФРА-М, 2007.
5. Бауэр, Ф. Расшифрованные секреты. Методы и принципы криптологии / Ф. Бауэр. – М.: Мир, 2007. – 550 с.
6. Бернет, С. Криптография. Официальное руководство RSA Security / С.Бернет, С.Пейн. – 2-е изд., стереотипное. – М.: «Бином-Пресс», 2009. – 384 с.
7. Ван-Тилборг, Х.К.А. Основы криптологии. Профессиональное руководство и интерактивный учебник / Х.К.А. Ван-Тилборг. – М.: Мир, 2006 – 471 с.
8. Василенко, О.Н. Теоретико-числовые алгоритмы в криптографии / О.Н. Василенко. – 2-е изд., доп. – М.: МЦНМО, 2006 – 326 с.
9. Гольдштейн, Б.С. Сети связи / Б.С. Гольдштейн, Н.А. Соколов, Г.Г. Яновский. – СПб.: БХВ-Петербург, 2011.
10. Гордеев, А.В. Операционные системы / А.В. Гордеев. – СПб.: Питер, 2006. – 416 с.
11. Дейт, К.Дж Введение в системы баз данных / К.Дж. Дейт – М.: Издательский дом «Вильямс», 2006. – 1328 с.
12. Ерохин, В.В. Безопасность информационных систем / В.В. Ерохин, Д.А. Погонышева, И.Г. Степченко. – М.: ФЛИНТА, 2015. – 184 с.
13. Ерохин, В.В. Верификация информационных систем коммерческого банка / В.В. Ерохин, Е.В. Елисеева // Вестник образовательного консорциума среднерусский университет. Информационные технологии. – 2017. – №1(9). – С. 20-23.
14. Ерохин, В.В. Защита информации электронных торговых сетей банков / В.В. Ерохин, Е.В. Елисеева и др. // Вопросы современ-

- ной науки: коллект. науч. монография; [под ред. Н.Р. Красовской]. – М.: Изд. Интернаука, 2017. Т. 18. – С. 123-140.
15. Ерохин, В.В. Защита программного обеспечения и верификация информации в информационно-телекоммуникационных системах банка. – М.: Изд-во МГУ, 2015. – 129 с.
 16. Ерохин, В.В. Управление доступом к информационному и программному обеспечению в коммерческом банке / В.В. Ерохин, Е.В. Елисеева, А.М. Хлопяников // Результаты социально-экономических и междисциплинарных научных исследований XXI века: монография. – Самара: Поволжская научная корпорация, 2016. – С. 189-202.
 17. Зильбербург, Л.И. Информационные технологии в проектировании и производстве / Л.И. Зильбербург, В.И. Молочник, Е.И. Яблочников. – СПб.: Политехника, 2008.
 18. Игнатьев, В.А. Информационная безопасность современного коммерческого предприятия / В.А.Игнатьев – Старый Оскол: ООО «ТНТ», 2005. – 448 с.
 19. Ключников, К.К. Методы снижения сбоев в работе биллинговой системы / К.К. Ключников // Мобильные системы. – 2007. – №08 – С. 46-48.
 20. Кузнецов, С.Л. Современные технологии документационного обеспечения управления / С.Л. Кузнецов. – М.: МЭИ, 2010. – 232 с.
 21. Кэрриэ, Б. Криминалистический анализ файловых систем / Б. Кэрриэ – СПб.: Питер, 2007. – 352 с.
 22. Марков, А.С. Методы оценки несоответствия средств защиты информации / А.С. Марков, В.Л. Цирлов, В.Л. Баранов. – М.: Горячая линия-Телеком, 2012. – 192 с.
 23. Мельников, В.П. Информационная безопасность / В.П. Мельников, С.А. Клейменов, А.М. Петраков – М.: Издательский центр «Академия», 2012. – 336 с.
 24. Олифер, В.Г. Компьютерные сети / В.Г. Олифер [и др.]. – СПб.: Питер, 2008. – 958 с.
 25. Олифер, В.Г. Сетевые операционные системы / В.Г. Олифер, Н.А. Олифер. – СПб.: Питер, 2009. – 672 с.
 26. Панасенко, С. Алгоритмы шифрования. Специальный справочник / С.Панасенко. – СПб.: БХВ-Петербург, 2009. – 578 с.
 27. Платонов, В.В. Программно-аппаратные средства обеспечения информационной безопасности вычислительных сетей / В.В. Платонов – М.: Издательский центр «Академия», 2006. – 240 с.

28. Скиба, В.Ю. Руководство по защите от внутренних угроз информационной безопасности / В.Ю. Скиба, В.А. Курбатов – СПб.: Питер, 2008. – 320 с.
29. Созыкин, А.В. Модели и методы создания интегрированной инфраструктуры управления доступом к сервисам / А.В. Созыкин // Системы управления и информационные технологии. – 2007. – №4.1(30). – С. 191-195.
30. Тихонов, В.А. Информационная безопасность: концептуальные, правовые, организационные и технические аспекты / В.А. Тихонов, В.В. Рай – М.: Солон-пресс, 2006. – 528 с.
31. Трофимов, В.В. Информационные системы и технологии в экономике и управлении / В.В. Трофимов, О.П. Ильина, Е.В. Трофимова, В.И. Кияев, Приходченко А.П. / под ред. проф. В.В.Трофимова. – М.: КНОРУС, 2011.
32. Хорев, П.Б. Методы и средства защиты информации в компьютерных системах / П.Б. Хорев. – М.: Издательский центр «Академия», 2008. – 256 с.
33. Anada, H., Yasuda, T., Kawamoto, J., Weng, J., Sakurai, K. (2019). RSA public keys with inside structure: Proofs of key generation and identities for web-of-trust. *Journal of Information Security and Applications*, 45, 10-19. doi:10.1016/j.jisa.2018.12.006.
34. Asokan, N., Nyman, T., Rattanavipanon, N., Sadeghi, A. -, Tsudik, G. (2018). ASSURED: Architecture for secure software update of realistic embedded devices. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(11), 2290-2300. doi:10.1109/TCAD.2018.2858422.
35. Bailey, D. V., Dürmuth, M., Paar, C. (2014). *Statistics on password re-use and adaptive strength for financial accounts*. doi:10.1007/978-3-319-10879-7_13.
36. Bailey, D.V., Brainard, J., Rohde, S., Paar, C. (2009). One-touch financial transaction authentication. Paper presented at the *ICETE 2009 - International Joint Conference on e-Business and Telecommunications*, 5-12.
37. Bailey, D.V., Brainard, J., Rohde, S., Paar, C. (2011). *Wireless authentication and transaction-confirmation token*. doi:10.1007/978-3-642-20077-9_13.
38. Brasser, F., Capkun, S., Dmitrienko, A., Frassetto, T., Kostiainen, K., Sadeghi, A. -. (2019). Dr.SGX: Automated and adjustable side-channel protection for sgx using data location randomization. Paper

presented at the *ACM International Conference Proceeding Series*, 788-800. doi:10.1145/3359789.3359809.

39. Brasser, F., Müller, U., Dmitrienko, A., Kostiainen, K., Capkun, S., Sadeghi, A. - (2017). Software grand exposure: SGX cache attacks are practical. Paper presented at the *11th USENIX Workshop on Offensive Technologies, WOOT 2017, Co-Located with USENIX Security 2017*.
40. Bulychev, P. Computing (bi)simulation relations preserving ctlx logic for ordinary and fair kripke structures / P. Bulychev, I. Konnov, V. Zakharov // Труды Института системного программирования РАН. – 2007. – Vol. 12. – Pp. 59-76.
41. Calder, M. Detecting feature interactions: how many components do we need? / M. Calder, A. Miller // Objects, Agents and Features. – 2004. – Vol. 2975. – Pp. 45-66.
42. Cheng, L., Liljestrand, H., Ahmed, M.S., Nyman, T., Jaeger, T., Asokan, N., Yao, D.D. (2019). Exploitation techniques and defenses for data-oriented attacks. Paper presented at the *Proceedings - 2019 IEEE Secure Development, SecDev 2019*, 114-128. doi:10.1109/SecDev.2019.00022.
43. Danev, B., Heydt-Benjamin, T. S., Čapkun, S. (2009). Physical-layer identification of RFID devices. Paper presented at the *Proceedings of the 18th USENIX Security Symposium*, 199-214.
44. Dhar, A., Puddu, I., Kostiainen, K., Capkun, S. (2020). Proximi-TEE: Hardened SGX attestation by proximity verification. Paper presented at the *CODASPY 2020 - Proceedings of the 10th ACM Conference on Data and Application Security and Privacy*, 5-16. doi:10.1145/3374664.3375726.
45. Ekberg, J. -, Kostiainen, K., Asokan, N. (2014). The untapped potential of trusted execution environments on mobile devices. *IEEE Security and Privacy*, 12(4), 29-37. doi:10.1109/MSP.2014.38.
46. Emerson, E.A. Parameterized model checking of ring-based message passing systems / E.A. Emerson, V. Kahlon // Computer Science Logic. – 2004. – Vol. 3210/2004 of LNCS. – Pp. 325-339.
47. EMVC0. The EMV 4.3 Specifications - Book 2 - Security and Key Management [online], 2011. Available at: <http://www.emvco.com/specifications.aspx?id=223>.
48. EMVC0. The EMV Contactless Specifications - Book D: Contactless Communication Protocol [online], 2011. Available at: <http://www.emvco.com/specifications.aspx?id=21>.

49. Erokhin V.V., Fetshchenko V.V., Panina I.S., Kazimirova N.P., Novikov S.P., Novikova A.V. Verification of Computer Systems of Commercial Bank // International Journal Of Applied Business and Economic Research. – 2017. – Volume 15, Number 12. – P. 297 – 306.
50. Erokhin V.V., Kulikova G.A., Mudrova N.V., Shadoba E.M., Romanov V.A. Controlling Access to the Information and Software in a Commercial Bank // International Journal Of Applied Business and Economic Research. – 2017. – Volume 15, Number 12. – P. 159 – 170.
51. Fan, J., Bailey, D.V., Batina, L., Güneysu, T., Paar, C., Verbauwhede, I. (2010). Breaking elliptic curve cryptosystems using reconfigurable hardware. Paper presented at the *Proceedings - 2010 International Conference on Field Programmable Logic and Applications, FPL 2010*, 133-138. doi:10.1109/FPL.2010.34.
52. Foltz, K., Simpson, W.R. (2020). Public key infrastructure issues for enterprise level security. Paper presented at the *ICEIS 2020 - Proceedings of the 22nd International Conference on Enterprise Information Systems*, 91-98.
53. Hawanna, V., Kulkarni, V. Y., Rane, R. A. (2017). Risk assessment of X.509 certificate by evaluating certification practice statements. Paper presented at the *International Conference on Computing, Analytics and Security Trends, CAST 2016*, 501-506. doi:10.1109/CAST.2016.7915020.
54. Heydt-Benjamin, T.S., Bailey, D.V., Fu, K., Juels, A., O'Hare, T. (2007). *Vulnerabilities in first-generation RFID-enabled credit cards*. doi:10.1007/978-3-540-77366-5_2.
55. IETF Internet Engineering Task Force. US Secure Hash Algorithm 1 (SHA1) [online], 2012. Available at: <http://tools.ietf.org/html/rfc3174>.
56. IETF Internet Engineering Task Force. Using SHA2 algorithms with cryptographic message syntax [online], 2012. Available at: <http://tools.ietf.org/html/rfc5754>.
57. IETF-RFC5246. The Transport Layer Security (TLS) protocol version 1.2 [online], 2008. Available at: <http://tools.ietf.org/html/rfc5246>.
58. ISO/IEC-14443. Identification cards Contactless integrated circuit cards Proximity cards [online] 2008. Available at: <http://www.iso.org/iso/iso-catalogue/catalogue-icscatalogue-detail-ics.htm?csnumber=28730>.

59. ISO/IEC-7813. Information technology Identification cards Financial transaction cards [online], 2008. Available at: <http://www.iso.org/iso/iso-catalogue/catalogue-tc/catalogue-detail.htm?csnumber=43317>.
60. Karthikeyan, S., Patan, R., Balamurugan, B. (2019). *Enhancement of security in the internet of things (IoT) by using X.509 authentication mechanism.* doi:10.1007/978-981-13-2685-1_22.
61. Kelly, D., Hammoudeh, M. (2018). Optimisation of the public key encryption infrastructure for the internet of things. Paper presented at the *ACM International Conference Proceeding Series*, doi:10.1145/3231053.3231098.
62. Khan, M.S.N., Marchal, S., Buchegger, S., Asokan, N. (2019). *Chowniot: Enhancing IoT privacy by automated handling of ownership change.* doi:10.1007/978-3-030-16744-8_14.
63. Kostiainen, K., Ekberg, J. -., Asokan, N., Rantala, A. (2009). On-board credentials with open provisioning. Paper presented at the *Proceedings of the 4th International Symposium on ACM Symposium on Information, Computer and Communications Security, ASI-ACCS'09*, 104-115. doi:10.1145/1533057.1533074.
64. Malisa, L., Kostiainen, K., Capkun, S. (2017). Detecting mobile application spoofing attacks by leveraging user visual similarity perception. Paper presented at the *CODASPY 2017 - Proceedings of the 7th ACM Conference on Data and Application Security and Privacy*, 289-300. doi:10.1145/3029806.3029819.
65. Malisa, L., Kostiainen, K., Knell, T., Sommer, D., Capkun, S. (2017). *Hacking in the blind: (almost) invisible runtime user interface attacks* doi:10.1007/978-3-319-66787-4_23.
66. Mannan, M., Asokan, N. (2020). Confronting the limitations of hardware-assisted security. *IEEE Security and Privacy*, 18(5), 6-7. doi:10.1109/MSEC.2020.3015413.
67. Marforio, C., Masti, R. J., Soriente, C., Kostiainen, K., Čapkun, S. (2016). Evaluation of personalized security indicators as an anti-phishing mechanism for smartphone applications. Paper presented at the *Conference on Human Factors in Computing Systems - Proceedings*, 540-551. doi:10.1145/2858036.2858085.
68. Marforio, C., Masti, R.J., Soriente, C., Kostiainen, K., Capkun, S. (2016). Hardened setup of personalized security indicators to counter phishing attacks in mobile banking. Paper presented at the *SPSM 2016 - Proceedings of the 6th Workshop on Security and Privacy in Smartphones*

and Mobile Devices, Co-Located with CCS 2016, 83-92.
 doi:10.1145/2994459.2994462.

69. Markert, P., Bailey, D. V., Golla, M., Durmuth, M., Avig, A.J. (2020). This PIN can be easily guessed: Analyzing the security of smartphone unlock PINs. Paper presented at the *Proceedings - IEEE Symposium on Security and Privacy*, 2020-May 286-303. doi:10.1109/SP40000.2020.00100.

70. Mumtaz, M., Akram, J., Ping, L. (2019). An RSA based authentication system for smart IoT environment. Paper presented at the *Proceedings - 21st IEEE International Conference on High Performance Computing and Communications, 17th IEEE International Conference on Smart City and 5th IEEE International Conference on Data Science and Systems, HPCC/SmartCity/DSS 2019*, 758-765. doi:10.1109/HPCC/SmartCity/DSS.2019.00112.

71. NFC-Forum. NFC Logical Link Control Protocol (LLCP) technical specification [online]. 2012. Available at: <http://www.nfcforum.org/specs/spec-list/>.

72. Ruan, O., Zhang, Y., Zhang, M., Zhou, J., Harn, L. (2018). After-the-fact leakage-resilient identity-based authenticated key exchange. *IEEE Systems Journal*, 12(2), 2017-2026. doi:10.1109/JSYST.2017.2685524.

73. Shivaprasad, S., Sadanandam, M., Sowjanya, M., Kondapalli, S. V. (2019). Deliver an accurate image on the digital certificate and digital signature using RSA and SHA. *Journal of Advanced Research in Dynamical and Control Systems*, 11(7), 501-514.

74. Wang, L., Asharov, G., Pass, R., Ristenpart, T., Shelat, A. (2019). Blind certificate authorities. Paper presented at the *Proceedings - IEEE Symposium on Security and Privacy*, 2019-May 1015-1032. doi:10.1109/SP.2019.00007.

ОГЛАВЛЕНИЕ

| | |
|---|-----|
| ВВЕДЕНИЕ | 3 |
| 1. АНАЛИЗ АВТОМАТИЗИРОВАННЫХ МЕТОДОВ ПОСТРОЕНИЯ УПРАВЛЕНИЯ ДОСТУПОМ К ИНФОРМАЦИОННОМУ И ПРОГРАММНОМУ ОБЕСПЕЧЕНИЮ БАНКА | |
| 1.1. Управления доступом к информационному и программному обеспечению в коммерческом банке | 34 |
| 1.2. Анализ методов проверки корректности работы программного обеспечения..... | 41 |
| 2. АВТОМАТИЗАЦИЯ ПРОЦЕССА УПРАВЛЕНИЯ ДОСТУПОМ К ИНФОРМАЦИОННОМУ И ПРОГРАММНОМУ ОБЕСПЕЧЕНИЮ БАНКА | |
| 2.1. Автоматизация разграничения доступа к информационному и программному обеспечению банка | 50 |
| 2.2. Обеспечение надёжности и доступности компьютерных систем банка | 74 |
| 3. АНАЛИЗ ДОСТОВЕРНОСТИ И ВЕРИФИКАЦИИ ИНФОРМАЦИИ В ИНФОРМАЦИОННО-ТЕЛЕКОММУНИКАЦИОННЫХ СИСТЕМАХ БАНКА | |
| 3.1. Алгоритм анализа достоверности и верификации информации | 84 |
| 3.2. Верификация вычислительных систем банка | 101 |
| 3.3. Совершенствования верификации симуляций между моделями программ..... | 107 |
| 4. АВТОМАТИЧЕСКИЙ АНАЛИЗ БЕЗОПАСНОСТИ ПЛАТЕЖНЫХ ПРОТОКОЛОВ | |
| 4.1. Протоколы бесконтактных платежей по банковским картам | 126 |
| 4.2. Протоколы оплаты с использованием мобильных устройств | 141 |
| 4.3. Протоколы оплаты с использованием мобильных токенов..... | 155 |
| ЛИТЕРАТУРА..... | |
| | 173 |



Уважаемые читатели!

Издательство «Спутник+»
предлагает:

- **ИЗДАНИЕ И ПЕЧАТЬ МОНОГРАФИЙ, КНИГ** любыми тиражами (от 50 экз.).
 - ✓ Срок - от 3-х дней в полноцветной и простой обложке или твердом переплете.
 - ✓ Присвоение ISBN, рассылка по библиотекам и регистрация в Книжной палате.
 - ✓ Оказываем помощь в реализации книжной продукции.
- **ПУБЛИКАЦИЯ НАУЧНЫХ СТАТЕЙ** для защиты диссертаций в журналах по гуманитарным, естественным и техническим наукам.
 - ✓ Журнал «Естественные и технические науки» входит в перечень ВАК.
- **ПРОВЕДЕНИЕ МЕЖДУНАРОДНЫХ НАУЧНО-ПРАКТИЧЕСКИХ ЗАОЧНЫХ КОНФЕРЕНЦИЙ** по всем научным направлениям для аспирантов, соискателей, докторантов и научных работников.
- **ПУБЛИКАЦИЯ СТИХОВ И ПРОЗЫ** в журналах «Российская литература», «Литературный альманах «Спутник» и «Литературная столица».
- + Набор, верстка, корректура и редактура текстов.
- + Печать авторефератов, переплет диссертаций (от 1 часа).
- Переплетные работы, тиснение, полноцветная цифровая печать.

*Наш адрес: Москва, 109428, Рязанский проспект, д. 8 А
тел. (495) 730-47-74, 778-45-60, 730-48-71 с 9 до 18 (обед с 14 до 15)
<http://www.sputnikplus.ru> e-mail: print@sputnikplus.ru*

Научное издание

Ерохин Виктор Викторович,
Бахадиров Хикмет Муратович

ВЕРИФИКАЦИЯ ИНФОРМАЦИИ И ЗАЩИТА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ В ИНФОРМАЦИОННО-ТЕЛЕКОММУНИКАЦИОННЫХ СИСТЕМАХ БАНКА

Монография

Издательство «Спутник +»
109428, Москва, Рязанский проспект, д. 8А.
Тел.: (495) 730-47-74, 778-45-60 (с 9.00 до 18.00)
Подписано в печать 17.05.2021. Формат 60×90/16.
Бумага офсетная. Усл. печ. л. 11,31. Тираж 25 экз. Заказ 161.
Отпечатано в ООО «Издательство «Спутник +»