УДК 536.421

А.В. Никешин, Н.В. Пакулин, В.З. Шнитман

АВТОМАТИЗАЦИЯ ТЕСТИРОВАНИЯ СООТВЕТСТВИЯ РЕАЛИЗАЦИЙ СТАНДАРТУ ПРОТОКОЛА БЕЗОПАСНОСТИ ТРАНСПОРТНОГО УРОВНЯ TLS

A.V. Nikeshin, N.V. Pakulin, V.Z. Shnitman

CONFORMANCE TESTING AUTOMATION FOR TRANSPORT LAYER SECURITY PROTOCOL TLS

Протокол TLS используется для защиты обмена данными между клиентом и сервером в различных прикладных сценариях: при обмене данными между браузером и веб-сервером, передаче электронной почты, создании виртуальных сетей, голосовой и видеосвязи и т. п. В настоящее время широко используются более десятка различных реализаций TLS. Обеспечение совместимости реализаций TLS является очень актуальной задачей: несовместимость двух реализаций может привести к различным последствиям, начиная с невозможности установить соединение вплоть до разглашения конфиденциальных данных.

Представлен подход к тестированию соответствия стандарту TLS, основанный на автоматизированном тестировании соответствия формальным спецификациям. Приведены результаты тестирования нескольких общедоступных реализаций TLS. Описаны направления дальнейших исследований.

ТЕСТИРОВАНИЕ; ВЕРИФИКАЦИЯ; ФОРМАЛЬНЫЕ МЕТОДЫ; ФОРМАЛЬНЫЕ СПЕЦИФИКАЦИИ; ТЕСТИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ МОДЕЛЕЙ; МОДЕЛИ ПРОГРАММ; TLS.

TLS protocol is widely used to protect data exchange between clients and servers in various scenarios: while browsing Internet, sending and receiving e-mails, establishing VPN, etc. There are dozens implementations in the market at the moment; such as, ensuring interoperability is highly important. Incompatibility between two implementations might result in disconnection or even disclosure of sensitive data.

Conformance testing is the primary tool to ensure interoperability between implementations of a protocol. The paper presents a model-based approach to conformance testing of TLS implementations. It discusses the formal model of TLS protocol, the structure of the test suite. We applied the test suite to a several popular implementations of TLS, and present brief results. The paper concludes with discussion of the directions for future research.

TESTING; VERIFICATION; FORMAL METHODS; FORMAL SPECIFICATION; MODEL BASED TESTING; SOFTWARE MODELS; TLS.

Среди всех современных протоколов, предназначенных для защиты передачи информации в открытых сетях, наиболее широко используется протокол Transport Layer Security (TLS)[1–4, 6]. Разработчики протокола TLS поставили перед собой цели обеспечить вычислительно эффективную криптографическую безопасность, совместимость реализаций и расширяемость протокола [4].

Протокол TLS применяется для защиты обменов между клиентом и сервером в различных прикладных сценариях: для защиты www-трафика (протокол HTTPS), отправки и получения электронной почты (с почтовыми протоколами SMTP, IMAP, POP3), при организации виртуальных защищенных сетей (OpenVPN), для защиты голосовой и видеосвязи (с SIP и основанными на нем приложениями, например VoIP), и многих др.

Как и у других популярных телекомму-

никационных протоколов, у протокола TLS в настоящее время широко используются более десятка различных реализаций. Несовместимость двух реализаций может привести к различным последствиям, начиная с невозможности установить соединение до разглашения конфиденциальных данных.

Наивный подход к решению задачи обеспечения совместимости заключается в том, чтобы каждую реализацию испытать на возможность взаимодействия с каждой. Очевидны недостатки этого подхода. Необходимо собрать все реализации TLS и провести попарные сценарии обменов данными. Так как для протоколов безопасности сценарии отказов не менее важны, чем сценарии нормальной передачи данных, необходимо разработать большое количество сценариев для каждой пары, причем может оказаться, что отдельные сценарии невозможно реализовать для тех или иных пар. Кроме того, такие испытания плохо поддаются автоматизации, многие сценарии приходится выполнять вручную.

В настоящей статье предлагается подход к оценке совместимости реализаций протокола TLS, основанный на оценке степени соответствия реализации стандарту протокола: если две реализации соответствуют спецификации протокола, то они совместимы. Необходимо отметить, что указанная гипотеза о совместимости соответствующих (conforming) реализаций принимается как постулат и в рамках данной работы не проверяется. Можно сказать, что мы доверяем разработчикам протокола TLS, которые заявили указанное свойство как одну из целей проектирования протокола [4].

Несмотря на популярность протокола TLS и ценность информации, которая защищается этим протоколом, открытых тестовых наборов для тестирования соответствия TLS, как это ни парадоксально, нет. Фактически, TLS является единственным из всех основных протоколов стека TCP/IP, для которого нет тестового набора для тестирования соответствия стандарту. Подробнее ситуация с тестами для TLS рассмотрена в следующем разделе.

Практическая значимость разработки такого тестового набора несомненна. С другой стороны, разработка тестового набора традиционным образом, как набора отдельных тестовых программ, каждая из которых проверяет определенное требование, является трудоемкой задачей. Прежде всего тесты должны проверять большое количество вариантов поведения протокола TLS в зависимости от выбора параметров TLS-соединения, таких как выбор криптографических алгоритмов, алгоритма сжатия, схемы построения общего секрета.

Альтернативой ручной разработке тестов как отдельных независимых программ является автоматизация построения тестов, основанная на модели протокола. Существует большое число методов, использующих модели протоколов для уменьшения ручного труда при создании тестов, эти методы получили совокупное название Model Based Testing или MBT, что переводится как «Тестирование, основанное на моделях» или «Тестирование с использованием моделей».

Цель данной работы – создание тестового набора для тестирования соответствия спецификации протоколу TLS, причем с самого начала было принято решение разрабатывать тестовый набор в рамках методологии тестирования, основанного на моделях.

В настоящей статье мы рассматриваем вопросы разработки модели протокола, тестов и результаты тестирования некоторых серверных реализаций TLS на соответствие стандарту протокола. Также в статье представлены направления дальнейших работ по развитию тестового набора для протокола TLS.

Тестирование TLS

В настоящее время используются более десяти семейств реализаций протокола TLS. Безусловно, все они тестировались в процессе разработки. Мы проанализировали тестовые наборы для открытых реализаций TLS и пришли к следующим выводам:

реализации тестируются по принципу «белого ящика», при этом существенно используются внутренние особенности реализации;

тестирование нацелено на выявление

ошибок в «недрах» реализации, соответственно, тестовые наборы не содержат специально выделенного подмножества тестов соответствия стандарту.

Тестовые наборы для проприетарных реализаций недоступны для использования вне компании-разработчика, поэтому мы ничего не можем сказать про их устройство.

Тестовый набор для самой популярной свободной реализации SSL/TLS – проект OpenSSL – содержит большой набор тестов для проверки различных аспектов поведения библиотеки OpenSSL. Однако эти тесты непереносимы на другие реализации, т. к. существенно используют особенности реализации OpenSSL.

С некоторой натяжкой можно сказать, что в OpenSSL частично реализовано тестирование совместимости: имеется возможность запустить приложение, реализующее клиентскую сторону TLS, в режиме тестирования. При запуске необходимо указать адрес и порт, на которых находится тестируемая реализация сервера. Результатом выполнения тестов будет вердикт о совместимости целевой реализации с OpenSSL.

Необходимо отметить, что при этом нет возможности как-либо соотнести результат работы тестов со стандартом. Результат говорит только о совместимости с OpenSSL, но ничего не говорит о том, какие пункты стандарта проверялись. При этом нельзя считать OpenSSL эталоном, т. к. не исследовалось, как данная реализация сама по себе соотносится со стандартом. Кроме того, очень важно тестировать поведение реализации в случае ошибок и отказов другой стороны. При тестировании на соответствие OpenSSL проверяется только поведение в нормальных сценариях.

Тестовый набор для реализации TLS в проекте открытой реализации языка Java OpenJDK насчитывает порядка 140 классов. Из них для тестирования соответствия стандарту предназначены только четыре, остальные проверяют различные компоненты реализации TLS в Java Secure Socket Extension.

В работе [7] описана верификация самого протокола TLS на исполнимой модели. Модель разрабатывалась на функциональном языке F# и представляет собой упрощенную реализацию TLS, которая, тем не менее, была способна взаимодействовать с полноценными (mainstream) реализациями TLS. Разработанная исполнимая модель была верифицирована на соответствие требованиям безопасности, другими словами, верифицировался сам протокол TLS. Авторы упомянули, что идет разработка соответствующего тестирующего программного обеспечения для протокола TLS на базе разработанной модели. Но подробности не приведены, других работ этих авторов на тему тестирования реализаций TLS обнаружить не удалось.

Подход, используемый в работе. В данной работе используется подход к автоматизации тестирования, основанный на технологии автоматизированного тестирования UniTESK [8], которая предоставляет средства автоматизации тестирования соответствия формальным спецификациям.

Требования, представленные в тексте стандарта, формализуются как набор булевских предикатов и императивных конструкций, заданных средствами исполнимого формализма. Для упрощения разработки и анализа модели в данной работе в качестве формализма использовалось расширение языка Java[9]. Указанный подход уже применялся нами в аналогичных проектах [10–14].

В UniTESK тест представляет собой конечный автомат. С каждым переходом автомата сопоставлено определенное тестовое воздействие. При выполнении перехода это воздействие подается на тестируемую реализацию, регистрируются реакции реализации и автоматически выносится вердикт о соответствии наблюдаемого поведения спецификации. Вердикт выносится на основании модели, которая проверяет, допустима ли зарегистрированная реакция в текущем состоянии тестируемой реализации. Так как тестирование проводится по схеме «черного ящика», т. е. внутреннее состояние реализации недоступно тесту, то модель также обновляет модельное состояние в соответствии с требованиями стандарта. Тем самым модель выступает как

эталонная реализация протокола.

Спецификация автомата теста на конкретном формализме называется *тестовым сценарием*. В данной работе для записи тестовых сценариев используется то же расширение языка Java [9], что и для разработки моделей. Тестовые воздействия, реализующие дуги в автомате теста, представляют собой методы на языке Java, в которых тестовые воздействия оказываются на объект модели протокола. Эти методы разрабатываются вручную, но обход осуществляется в полностью автоматическом режиме. Алгоритм обхода не зависит от протокола, тестируемой реализации или конкретного теста; он реализован как библиотечный компонент UniTESK, разработчикам тестов не требуется адаптировать его под целевой протокол.

Модель протокола TLS. Построение тестовых воздействий и верификация полученных ответов реализации опирается на модель протокола.

Стандарт протокола не требует, чтобы реализации протокола использовали в точности те структуры данных, которые введены в стандарте для описания протокола. Требуется только соответствие внешне наблюдаемого поведения требованиям. Однако в модели мы не гонимся за производительностью, поэтому набор внутренних переменных модели состоит из тех же самых переменных, что указаны в стандарте: параметры сессии TLS и по четыре состояния чтения-записи на сессию.

Помимо состояний в модель протокола входят спецификационные методы – исполнимые модели стимулов и реакций TLS-сервера. Каждый спецификационный метод содержит предусловие, в котором проверяется правильность структуры тестового сообщения и его своевременность, и на основании этого делается вывод о том, должен ли на него быть ответ, сообщение об ошибке или реализация должна его проигнорировать.

Параметрами спецификационных методов служат сообщения TLS в модельном представлении, т. е. объекты Java. Наборы полей соответствующих классов следуют структуре сообщений, описанных в стандарте TLS. В состав тестового набора входят кодеки (codecs) для преобразования сообщений из объектов в байтовые массивы для отправки в целевую систему и для обратного преобразования полученных сообщений.

В отличие от большинства телекоммуникационных протоколов, кодеки для TLS не являются контекстно-независимыми. Для корректного кодирования и декодирования сообщений кодек должен иметь доступ к текущему состоянию приема и отправки, прежде всего к ключам шифрования и криптографической контрольной суммы.

Другая особенность кодеков для TLS отражает тот факт, что протокол TLS двухуровневый. Сообщения служебных протоколов (например, TLS Handshake) и данные прикладных протоколов преобразуются в двоичный вид и фрагментируются, а полученные фрагменты обрабатываются протоколом TLS Records – упаковываются и шифруются. Фактически, именно в кодеках реализуется модель протокола TLS Records и его взаимодействие со служебными протоколами.

Тестовый набор. Тестовый стенд состоит из двух сетевых узлов: инструментального и целевого. На целевом узле функционирует тестируемая реализация. На инструментальном узле исполняется обход тестового автомата и верификация наблюдаемых реакций. Инструментальный и целевой узлы могут располагаться в разных сегментах сети.

Стимулами в разработанном тестовом наборе являются сообщения от инструментального узла, а реакциями – сообщения со стороны тестируемого узла. Основная часть требований спецификации TLS проверяется в модели протокола. Часть требований проверяется в декодерах сообщений – прежде всего требования к структуре сообщений и форматам данных.

Процесс тестирования начинается с создания TCP соединения к целевому узлу и инициализации начальных значений переменных модели. Все обмены сообщениями происходят по созданному соединению. Из модельного представления тестового сооб-

щения TLS строится реализационное, которое и отправляется в сеть. Сообщения TLS, полученные из установленного соединения, декодируются, и строятся модельные представления этих сообщений. Последовательность блоков данных в полученных сообщениях рассматривается как последовательность реакций целевой системы.

В модели полученные сообщения проверяются на соответствие требованиям спецификации. Проверка разделена на несколько стадий. Сначала проверяется допустимость такого сообщения от реализации и его своевременность, затем – структура самого сообщения (присутствующие поля и их значения должны соответствовать текущему обмену). После проверки всех требований результат передается тестовому сценарию, где в зависимости от плана сценария принимается решение о продолжении или завершении информационного обмена. В случае выявления нарушения требований принимается решение о критичности ошибки и возможности отправки следующих запросов.

Первыми по установленному TCP соединению передаются сообщения протокола TLS Handshake. В случае успешного завершения обменов по этому протоколу в модели создается новая TLS-сессия и устанавливаются параметры следующих состояний чтения-записи (pending read-write states). Кроме того, модель протокола поддерживает сокращенный вариант обмена, в котором используется уже существующая TLS-сессия.

При создании TLS-сессии клиент и сервер согласовывают различные параметры: применяемые алгоритмы сжатия, шифрования и цифровой подписи, схемы генерации разделяемых секретов, методы аутентификации. Для TLS Handshake разработано несколько тестовых сценариев, в которых перебираются различные сочетания криптографических алгоритмов и соответствующих им схем построения секретов. Тест для Handshake разделен на несколько автоматов в силу особенности алгоритма обхода в UniTESK – длина тестовой последовательности (а значит, и время выполнения теста) быстро растут по мере увеличения числа вариантов тестовых воздействий (дуг в автомате). Для получения приемлемого времени работы теста иногда имеет смысл разделить тестовый автомат на несколько, которые в совокупности дают то же покрытие требований, что и «большой» тест, но исполняются за меньшее время.

Протокол TLS Handshake содержит львиную долю функциональности протокола TLS. Функции остальных служебных протоколов значительно проще, поэтому для них не требуется большое количество сценариев. Протокол смены состояния и протокол передачи прикладных данных тестируются в одном сценарии, т. к. невозможно передавать данные через соединение TLS, не осуществив смену состояния. Протокол оповещений покрыт в текущей реализации тестового набора не полностью. Часть из ошибочных ситуаций, предусмотренных стандартом протокола, не тестируются.

Апробация. Тестовый набор апробирован на примере тестирования нескольких открытых серверных реализаций TLS. В роли TLS-сервера реализация не генерирует запросы, а лишь поддерживает информационный обмен, инициированный другим узлом. Стимулами являются тестовые сообщения от инструментального узла.

Для тестирования на соответствие стандарту были выбраны следующие реализации:

почтовый сервер Postfix 2.9.3 с открытой реализацией протокола TLS OpenSSL 1.0.1c под управлением операционной системы Red Hat Enterprise Linux 5.5;

реализация TLS в виртуальной машине Java 1.7.0_05 (Java Secure Socket Extension, JSSE);

тестовый сервер TLS интернет-ресурса https://www.mikestoolbox.net.

В первой реализации было выявлено восемь отклонений от спецификации TLS. В остальных – по три отклонения от спецификации. Несмотря на некоторые особенности и нарушения, реализации в целом соответствуют спецификации. Однако третья реализация (https://www.mikestoolbox.net) нарушает критичное требование проверки версии протокола TLS в сообщении

ClientKeyExchange с использованием алгоритма RSA. Подробный разбор части результатов тестирования можно найти в [14].

Направления дальнейшего развития

Разработанный тестовый набор для протокола TLS покрывает существенную часть функциональных требований сервера TLS, изложенных в стандартах протокола.

Мы видим два направления дальнейшего развития разработанного тестового набора.

С одной стороны, необходимо расширить тестовый набор тестами для клиента TLS. Основная трудность здесь заключается в том, что необходимо разработать средства для взаимодействия теста с тестируемым клиентом: необходимо выдавать стимулы для установления связи с сервером или отправки данных, а также для сбора реакций клиента. Если для реализации TLS в Java реализовать подобное средство не составит труда, то тестирование клиента в браузере требует разработать и установить расширения, которые позволят воздействовать на браузер извне.

Кроме того, для протоколов безопасности помимо тестирования соответствия есть еще одна важная задача: тестирование устойчивости. Протокол TLS используется для защиты общедоступных серверов, поэтому реализация TLS должна быть «готова» к приему произвольных сообщений.

Спецификация TLS определяет более 40 ошибочных ситуаций, однако их список не является исчерпывающим. Помимо тестирования на выделенные в спецификации особые случаи необходимо тестировать устойчивость к ошибкам в сообщениях, не указанным в спецификации протокола.

На практике тестирование устойчивости к некорректным сообщениям осуществляется преимущественно посредством случайного тестирования (fuzz testing, random testing). При случайном тестировании генерируются более-менее произвольные последовательности байтов в качестве целых пакетов или отдельных полей.

Построение случайных сообщений легко реализовать. Существует большое число исследовательских и коммерческих генераторов случайных сообщений, например IxDefend компании Ixia, Mu Dynamix одноименной компании, EMRET компании Microsoft, Dranzer организации CERT и т. д. Однако вероятность построения сообщения, которое способно преодолеть хотя бы часть механизмов валидации (сжатие, шифрование и контрольные суммы TLS Records, согласованные значения полей сообщений TLS Handshake) настолько мала, что случайное тестирование, дающее сколько-нибудь значимую долю покрытия кода реализации, займет неприемлемо длительное время.

Мы предполагаем дополнить имеющийся тестовый набор средствами мутационного тестирования (mutation testing), при котором ошибочные сообщения строятся как искажения корректных сообщений. При мутационном тестировании необходимо сначала построить корректное сообщение, а затем трансформировать («мутировать») его в некорректное. При мутационном тестировании число тестовых воздействий существенно меньше, чем при случайном тестировании, а покрытие реализации выше. Применение мутационного тестирования нацелено на выявление нарушений функционирования из-за переполнений буфера, разрушения стека, индексирования за пределами массива.

Применение мутационного тестирования к TLS связано с нетривиальной задачей построения корректного сообщения, т. к. содержимое сообщения связано с внутренним состоянием протокола: сообщение должно быть корректно зашифровано и подписано, в противном случае оно будет отброшено на ранних этапах обработки. Мы предполагаем использовать мутации совместно с тестированием на основе моделей: благодаря наличию модели можно целенаправленно привести реализацию в необходимое состояние и подготовить разумные данные для корректного сообщения.

Тестирование соответствия спецификации или стандарту является в настоящее время основным механизмом обеспечения совместимости между различными реализациями: если две реализации корректно

реализуют протокол, то они обязательно смогут взаимодействовать. Такой подход используется для протоколов всех уровней в стеке TCP/IP: существуют тесты для канального уровня (Ethernet), сетевого (IPv4 и IPv6), транспортного (TCP и UDP), многих протоколов прикладного уровня (DNS, DHCP, SMTP/POP/IMAP и т. п.).

Однако в сфере TLS сложилась парадоксальная ситуация. Протокол TLS используется на сотнях тысяч серверов по всему миру, сотни миллионов людей пользуются им ежедневно для защиты соединений с веб-серверами, отправки и получения электронной почты, существует более десятка реализаций, но при этом нет ни одного достаточно содержательного тестового набора, с помощью которого можно было бы удостовериться в соответствии реализации стандарту TLS.

В данной статье представлен подход к разработке тестового набора для тестирования соответствия реализации протокола TLS на соответствие стандарту. Подход использует технологию UniTESK к автоматизации тестирования: тестовая последовательность строится динамически как обход некоторого автомата теста, для построения тестовых воздействий и вынесения вердикта о корректности наблюдаемого поведения реализации используется модель протокола.

Разработан тестовый набор, покрывающий большинство требований, изложенных в стандарте протокола TLS [4]. Тестовый набор апробирован на трех общедоступных реализациях TLS/SSL, во всех реализациях обнаружены отклонения от спецификации, в одном случае нарушение расценивается как критическое.

В качестве дальнейшего направления работ рассматривается расширение тестового набора тестами на некорректные сообщения. Такие тесты предполагается автоматизировать с использованием методов мутационного тестирования.

Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований, проект 13-07-00869.

СПИСОК ЛИТЕРАТУРЫ

1. **Freier A., Karlton P., Kocher P.** The Secure Sockets Layer (SSL) Protocol Version 3.0. Aug. 2011.

2. **Dierks T., Allen C.** IETF RFC 2246 // The TLS Protocol Version 1.0, Jan. 1999.

3. **Dierks T., Rescorla E.** IETF RFC 4346 // The Transport Layer Security (TLS) Protocol Version 1.1. Apr. 2006.

4. **Dierks T. Rescorla E.** IETF RFC 5246 // The Transport Layer Security (TLS) Protocol Version 1.2. Aug. 2008.

5. **Turner S., Polk T.** IETF RFC 6176 // Prohibiting Secure Sockets Layer (SSL) Version 2.0. March 2011.

6. Проект SSL Pulse консорциума Trustworthy Internet Movement [электронный ресурс] / URL: https://www.trustworthyinternet.org/ssl-pulse/

7. **Bhargavan K., Fournet C., Corin R., Zalinescu E.** Cryptographically verified implementations for TLS // Proc. of the 15th ACM Conf. on Computer and Communications Security. ACM New York, 2008.

8. **Bourdonov I., Kossatchev A., Kuliamin V., Petrenko A.** UniTesK Test Suite Architecture // Proc. of FME. Springer-Verlag, 2002. LNCS 2391. Pp. 77–88.

9. **Bourdonov I.B., Demakov A.V., Jarov A.A., Kossatchev A.S., Kuliamin V.V., Petrenko A.K., Zelenov S.V.** Java Specification Extension for Automated Test Development // Proc. of PSI-2001. Novosibirsk. Springer-Verlag, 2001. LNCS 2244. Pp. 301–307.

10. **Пакулин Н.В.** Формализация стандартов и тестовых наборов протоколов Интернета. Дис. ... канд. физ.-мат. наук. М., 2006.

11. **Пакулин Н.В., Хорошилов А.В.** Разработка формальных моделей и тестирование соответствия для систем с асинхронными интерфейсами и телекоммуникационных протоколов // Программирование. 2007. № 5. С. 1–29.

12. **Никешин А.В., Пакулин Н.В., Шнитман В.З.** Разработка тестового набора для верификации реализаций протокола безопасности IPsec v2 // Труды Института системного программирования РАН. 2010. Т. 18. С. 151–182.

13. **Никешин А.В., Пакулин Н.В., Шнитман В.З.** Верификация функций безопасности протокола IPsec v2 // Программирование. 2011. № 1. С. 36–56.

14. **Никешин А.В., Пакулин Н.В., Шнитман В.З.** Разработка тестового набора для верификации реализаций протокола безопасности TLS // Труды Института системного программирования РАН. 2012. Т. 23. С. 387–404.

REFERENCES

1. **Freier A., Karlton P., Kocher P.** *The Secure Sockets Layer (SSL) Protocol Version 3.0.* Aug. 2011.

2. **Dierks, T., Allen C.** IETF RFC 2246. *The TLS Protocol Version 1.0.* Jan. 1999.

3. **Dierks, T., Rescorla E.** IETF RFC 4346. *The Transport Layer Security (TLS) Protocol Version 1.1.* Apr. 2006.

4. **Dierks, T. and E. Rescorla** IETF RFC 5246. *The Transport Layer Security (TLS) Protocol Version 1.2.* Aug. 2008.

5. **Turner S., Polk T.** IETF RFC 6176. *Prohibiting Secure Sockets Layer (SSL) Version 2.0.* March 2011.

6. *Project «SSL Pulse» of Trustworthy Internet Movement Consortium*. Available: https://www.trustworthyinternet.org/ssl-pulse/

7. **Bhargavan K., Fournet C., Corin R., Zalinescu E.** Cryptographically verified implementations for TLS, *Proceedings of the 15th ACM Conference on Computer and Communications Security*, ACM New York, USA, 2008.

8. **Bourdonov I., Kossatchev A., Kuliamin V., Petrenko A.** UniTesK Test Suite Architecture, *Proceedings of FME*. Springer-Verlag, 2002. LNCS 2391, Pp. 77–88.

9. **Bourdonov I.B., Demakov A.V., Jarov A.A., Kossatchev A.S., Kuliamin V.V., Petrenko A.K., Zelenov S.V.** Java Specification Extension for Automated Test Development. *Proceedings of PSI-2001*, Novosibirsk, Russia, Springer-Verlag, 2001, LNCS 2244, Pp. 301–307.

10. **Pakulin N.V.** *Formalizatsiya standartov i testovyih naborov protokolov Interneta* [*Formalisation of specification and test suite for Internet protocols*]. Dis. ... kand. fiz.-mat. nauk [PhD thesis], Moscow, 2006. (rus)

11. **Pakulin N.V., Khoroshilov A.V.** Razrabotka formalnyih modeley i testirovanie sootvetstviya dlya sistem s asinhronnyimi interfeysami i telekommunikatsionnyih protokolov [Development of formal models and conformance testing for systems with asynchronous interfaces and telecommunications protocols], *Programmirovanie* [*Programming and Computing Software*], 2007, Vol. 33, Iss. 6, Pp. 316–335. (rus)

12. **Nikeshin A.V., Pakulin N.V., Shnitman V.Z.** Razrabotka testovogo nabora dlya verifikatsii realizatsiy protokola bezopasnosti IPsec v2 [Development of a test suite for the verification of implementations of the IPsec v2 security protocol], *Trudyi Instituta sistemnogo programmirovaniya RAN* [*Proceedings of IPS RAS*], 2010, Vol. 18, Pp. 151–182. (rus)

13. **Nikeshin A.V., Pakulin N.V., Shnitman V.Z.** Verifikatsiya funktsiy bezopasnosti protokola IPsec v2 [Development of a test suite for the verification of implementations of the IPsec v2 security protocol], *Programmirovanie* [*Programming and Computing Software*], 2011, Vol. 37, Iss. 1, Pp. 36–56. (rus)

14. **Nikeshin A.V., Pakulin N.V., Shnitman V.Z.** Razrabotka testovogo nabora dlya verifikatsii realizatsiy protokola bezopasnosti TLS [Development of a test suite for the verification of implementations of the TLS security protocol], *Trudyi Instituta sistemnogo programmirovaniya RAN* [*Proceedings of IPS RAS*], 2012, Vol. 23, Pp. 387–404. (rus)

НИКЕШИН Алексей Вячеславович – *младший научный сотрудник Института системного программирования РАН.*
109004, Россия, Москва, ул. Александра Солженицына, д. 25.
E-mail: alexn@ispras.ru

NIKESHIN, Alexey V. *Institute for System Programming of the Russian Academy of Sciences.*
195251, Alexander Solzhenitsyn Str. 25, Moscow, Russia.
E-mail: alexn@ispras.ru

ПАКУЛИН Николай Витальевич – *старший научный сотрудник Института системного программирования РАН, кандидат физико-математических наук.*
109004, Россия, Москва, ул. Александра Солженицына, д. 25.
E-mail: npak@ispras.ru

PAKULIN, Nikolay V. *Institute for System Programming of the Russian Academy of Sciences.*
195251, Alexander Solzhenitsyn Str. 25, Moscow, Russia.
E-mail: npak@ispras.ru

ШНИТМАН Виктор Зиновьевич – *заместитель директора Института системного программирования РАН, доктор физико-математических наук, профессор.*
109004, Россия, Москва, ул. Александра Солженицына, д. 25.
E-mail: vzs@ispras.ru

SNITMAN, Victor Z. *Institute for System Programming of the Russian Academy of Sciences.*
195251, Alexander Solzhenitsyn Str. 25, Moscow, Russia.
E-mail: vzs@ispras.ru