Инженерный вестник Дона, №11 (2025)
ivdon.ru/ru/magazine/archive/n11y2025/10490

Обзор методов верификации для протоколов доказательства с нулевым разглашением

Д. А. Панченко

Южный федеральный университет, Таганрог

Аннотация: Информационные технологии стали все больше применяться в различных сферах, будь то документооборот или платежные системы. Одной из наиболее популярных и перспективных технологий являются криптовалюты. Так как они требуют обеспечение безопасности и надежности данных в системе, то большинство из них используют блокчейн и сложные криптографические протоколы, например, протоколы доказательства с нулевым разглашением. Поэтому важным аспектом для достижения безопасности данных систем является верификация, так как с помощью неё можно оценить устойчивость системы к различным атакам, а также её соответствие требованиям безопасности. В данной работе будет рассмотрено как само понятие верификации, так и методы для её осуществления, а также проведено сравнение методов для выявления подходящего для протоколов доказательства с нулевым разглашением. И в результате сделан вывод о том, что необходим комплексный подход к верификации, так как выбор только одного метода не может покрыть все потенциальные уязвимости, в связи с чем необходимо применять на различных этапах проектирование системы различные методы верификации.

Ключевые слова: криптовалюта, блокчейн, верификация, формальный метод, статический анализ, динамический метод, протокол доказательства с нулевым разглашением.

Введение

В данный момент всё большую популярность обретают различные информационные технологии. Например, базы данных, электронный документооборот и электронные платежи, которые всё чаще стали использоваться, в связи с чем для данных систем требуется обеспечение надёжности и безопасности. Поэтому планирование и проектирование таких систем [1] является важным этапом при их создании.

Также одним из перспективных направлений развития информационных технологий является криптовалюты. Поэтому они также требуют надёжной защиты. Таким образом, протоколы доказательства с нулевым разглашением [2] также стали более популярны, так как используются в криптовалютах, например Zcash, для обеспечения анонимности. Кроме того, данные протоколы можно использовать и для

аутентификации в веб-приложениях [3]. Это говорит о том, что данные протоколы являются перспективным направлением развития криптографии [4].

В связи с этим всё острее становится проблема безопасности систем доказательства, которые используют протоколы доказательства с нулевым разглашением. Так как данные протоколы используют различные криптографические механизмы, то поиск уязвимостей в них не является простой задачей, и сами криптовалюты построены на блокчейн-системах, в которых также используются различные криптографические алгоритмы [5], в связи с чем блокчейн играет важную роль в кибербезопасности [6]. Поэтому целью данной работы является рассмотрение текущих методов верификации и определение оптимального метода верификации для данных протоколов.

Понятие верификация

Проблему безопасности решают с помощью верификации, так как с помощью неё обеспечивается правильная работа системы ещё на этапе её проектирования. «Верификация – подтверждение на основе представления объективных свидетельств того, что установленные требования были выполнены» [7]. Поэтому верификация является неотъемлемой частью разработки различных систем и протоколов, и строгой проверкой, которая обеспечивает выполнение требований безопасности [8]. При этом исправление несоответствий и противоречий, возникающих в процессе верификации, является другой задачей, которая выходит за рамки верификации. Стандарт института инженеров электротехники и электроники (Institute of Electrical and Electronics Engineers – IEEE) 1012 [9] регулирует весь процесс верификации. Хоть методы верификации и развиваются, они всё равно не успевают за текущим развитием программного обеспечения (ПО). «Поэтому предельная сложность ПО, которое можно сделать надежно

и корректно работающим, существенно меньше сложности систем, востребованных современным обществом» [10].

Методы верификации

На данный момент есть несколько основных способов верификации: экспертизы, статический анализ, формальные методы, динамические и синтетические методы. Рассмотрим каждый из них подробнее.

Экспертиза

Согласно стандарту IEEE 1028 [11] данный метод нацелен на обнаружение формальных дефектов, с помощью поверхностной проверки. Экспертизы бывают нескольких видов, например: техническая экспертиза, сквозной контроль, инспекция и аудит. Есть общие методы, основанные на методе оценки по Фагану [12], а также специализированные, такие как организационная экспертиза и эвристическая оценка. Ещё одним немаловажным методом является систематический, например - метод анализа архитектуры программного обеспечения [13].

Но данные методы имеют одни серьёзный недостаток – нет возможности автоматизации. А также необходимо непосредственное участие специалистов при верификации.

Статический анализ

Статический анализатор осуществляет поиск строк исходного кода, которые потенциально не удовлетворяют установленным нормам кодирования, и выводит диагностические сообщения с целью разъяснения разработчику причин возникновения проблемы. Процесс статического анализа структурно схож с компиляцией, за исключением того, что при его выполнении не производится генерация объектного или исполняемого кода. [14].

Статический анализ позволяет проверить исходный код программы до её выполнения, поэтому его можно легко автоматизировать, и большинство

компиляторов имеют встроенный анализатор. Но есть и специальные инструменты, которые могут сделать более детальный анализ кода, например:

Plum Hall, INC (USA, Hawaii) - мировой лидер в разработке пакетов проверки компиляторов и программного обеспечения. Тестовые наборы Plum Hall проверяют соответствие компиляторов и библиотек C и C++ стандартам международной организация по стандартизации (International Organization for Standardization – ISO). Тестовые наборы развиваются: тесты добавляются, отключаются и изменяются по мере развития стандартов [15];

Ливерпульская ассоциация по исследованию данных (Liverpool Data Research Associates – LDRA) - С 1975 года LDRA является пионером в области качества и верификации программного обеспечения. То, что начиналось как инновационное видение в области тестирования программного обеспечения, превратилось в глобальную силу, которая формирует отраслевые стандарты, стимулирует технологический прогресс и обеспечивает безопасность и надёжность критически важных систем по всему миру. LDRA формирует будущее качества программного обеспечения, помогая разработчикам обеспечивать безопасность, надёжность и соответствие требованиям [16];

Jtest - это комплексный инструмент тестирования Java, разработанный компанией Parasoft, INC (USA, California) и предназначенный для упрощения модульного тестирования, статического анализа и анализа покрытия кода Java-приложений. Он предоставляет возможности автоматизированного тестирования, позволяя разработчикам выявлять и устранять дефекты на ранних этапах разработки. Jtest интегрируется с популярными средами разработки, такими как Eclipse и IntelliJ, обеспечивая бесперебойные процессы тестирования. Его расширенные функции включают в себя

автоматическую генерацию тестовых случаев, обратную связь в режиме реального времени и поддержку непрерывной интеграции [17];

Svace - Данный инструмент предназначен для выявления ошибок и потенциальных уязвимостей в исходном коде программ, написанных на языках программирования C, C++, C# и Java. Его отличительными чертами являются легкость использования, обширная библиотека возможных предупреждений, возможность обработки больших объемов программного кода (до миллионов строк), а также высокое качество анализа (от 30 до 80 процентов достоверных предупреждений) [18, 19].

Несомненным преимуществом данного метода является автоматизация, однако он имеет и другие преимущества:

- Повышение качества кода, так как анализаторы помогают выявить неэффективные алгоритмы, неправильное использование программного интерфейса приложений и дублирование кода. Это помогает сделать код более читабельным и улучшает его структуру. Также это помогает поддерживать единые стандарты программирования, так как анализатор может проверять соблюдение правил форматирования кода.

- Обнаружение ошибок на ранних этапах разработки, что в будущем усиливает безопасность кода.

Но у данного метода есть и минусы, это предварительная настройка, так как для правильной работы требуется корректная настройка, иначе могут возникнуть ложные срабатывания из-за слишком строгой настройки или же несрабатывание в сложных случаях, так как при использовании редких библиотек или языков программирования, анализатор может не понять код и не обнаружить ошибку.

Формальные методы

Данный метод базируется на математическом моделировании программ и требований к ПО. Следовательно, исследованию подвержена

математическая модель, которая является идеализированным описанием программы. В основе формальных методов верификации лежат логико-алгебраические, исполнимые и промежуточные формальные модели требований к ПО, поведения и окружения ПО [20]. «На сегодняшний день особого внимания заслуживает применение формальной верификации для проверки криптографических протоколов на обеспечение специальных свойств безопасности» [21]. В данный момент широко используются такие средства, как:

Автоматизированная проверка протоколов и приложений безопасности интернета (Automated Validation of Internet Security Protocols and Applications – AVISPA) - это инструмент для автоматизированной проверки протоколов и приложений, чувствительных к интернет-безопасности. Он предоставляет модульный и выразительный формальный язык для описания протоколов и их свойств безопасности, а также интегрирует различные «бэкэнды», реализующие множество современных методов автоматического анализа. Ни один другой инструмент не обладает таким же уровнем охвата и надежности, обеспечивая при этом такую же производительность и масштабируемость. [22];

Pro Verif - это автоматический символьный верификатор протоколов. Он поддерживает широкий спектр криптографических примитивов, определяемых правилами перезаписи или уравнениями. Он может доказывать различные свойства безопасности: секретность, аутентификацию и эквивалентность процессов для неограниченного пространства сообщений и неограниченного количества сеансов. В качестве входных данных он принимает описание протокола для проверки на диалекте прикладного исчисления числа Пи, расширения исчисления числа Пи с использованием криптографии. Он автоматически переводит это описание протокола в

предложения Хорна и определяет, выполняются ли требуемые свойства безопасности путем разрешения этих предложений [23];

Простой интерпретатор на язык промела (Simple Promela Interpreter – SPIN) - широко используемый инструмент верификации программного обеспечения с открытым исходным кодом. Он может использоваться для формальной верификации многопоточных приложений. Разработка инструмента началась в Bell Labs в группе Unix Исследовательского центра вычислительных наук в 1980 году, и он доступен бесплатно с 1991 года. Spin продолжает развиваться, чтобы идти в ногу с новыми разработками в этой области. В апреле 2002 года инструмент был удостоен награды ассоциации вычислительной техники System Software Award [24];

Tamarin - успешно использовался для анализа и поддержки разработки современных протоколов безопасности. Преимущества Tamarin:

- Расширенный графический пользовательский интерфейс, который автоматически находит атаки или доказательства. Графики атак показывают, как именно может быть нарушено свойство;

- Интерактивное построение доказательств или обнаружение атак. Создавайте частичные доказательства или управляйте поиском доказательств или атак вручную. Для сложных протоколов пользователи могут просматривать частичные доказательства и записывать вспомогательные леммы;

- Интерфейс командной строки. Выполняйте анализ протокола в пакетном режиме с помощью командной строки. [25];

Scyther - инструмент для проверки криптографических протоколов, который может проверять большинство протоколов для неограниченного количества сеансов менее чем за секунду. Он даёт «полную характеристику» ролей протокола, позволяя разработчикам рано обнаруживать возможные неожиданные поведения. Также данный инструмент используется в

обучении. Scyther использовался для предоставления студентам практического опыта верификации протоколов и, следовательно, для ознакомления их с некоторыми подводными камнями проектирования протоколов путем проверки существующих протоколов или протоколов, разработанных ими самими [26].

Данный метод хорошо подходит для верификации криптографических протоколов, так он позволяет удостовериться в том, что протоколы устойчивы к различных атакам, но он проверят модель, а не саму реализацию. Поэтому ошибки, допущенные при переносе модели на язык программирования, будут незамечены.

Динамические методы

С помощью данного метода происходит проверка соответствия работы ПО с требованиями к нему. Выполняется он посредством тестирования по сценариям либо с помощью мониторинга. Способы мониторинга разделяются по способу получения характеристик ПО [27]: основанные на событиях и статические методы. Вот некоторые инструменты:

AppDynamics – лидирующий инструмент для контроля производительности приложений. Он обеспечивает мониторинг состояния инфраструктуры приложений и даёт детальное понимание происходящего на уровне кода. Поддерживает большинство современных технологий разработки. AppDynamics доступен как в форме локального решения, так и в виде облачного сервиса.

Для сбора необходимых данных используется специальное программное обеспечение («агент»), которое интегрируется непосредственно в приложение. Собранные агентом показатели отправляются на центральный компонент – контроллер. Контроллер обрабатывает полученные метрики и выводит аналитику через удобный веб-интерфейс. Пользователи имеют

возможность самостоятельно настроить уведомления и формировать отчёты через тот же интерфейс [28];

Applications Manager - предоставляет вам полный контроль над стеком облачных и локальных приложений. Получайте глубокие проактивные аналитические данные для выявления и предотвращения узких мест производительности, оптимизации работоспособности приложений и обеспечения бесперебойной работы пользователей. Эта унифицированная платформа упрощает информационные процессы, оптимизируя мониторинг приложений, обеспечивая максимальную производительность [29];

AQtime – это инструмент для повышения производительности процесса отладки памяти и ресурсов для компиляторов Microsoft, Intel, Borland, Compaq, разработанный AutomatedQA, INC (USA, Massachusetts) и неоднократно отмеченный наградами. Можно интегрировать AQtime в Microsoft Visual Studio или запускать его отдельно. Отлаживайте, оптимизируйте, поставляйте цельные приложения с помощью AQtime, он поможет вам не только запустить проблему, но и выяснить, что стало ее причиной [30];

Данный метод позволяет обнаруживать ошибки во время работы программы, а также позволяет оценить характеристики работы в реальных условиях, но для применения данного метода нужно иметь готовую и работоспособную программу, что не позволяет использовать данный метод на ранних этапах разработки. А также для программ с большим числом состояний невозможно за приемлемое время проверить их все, из-за этого требуются различные методы сокращения обхода всех состояний.

Синтетические методы

Современные направления синтетических методов охватывают тестирование на основе моделей [31], при котором используются формальные модели требований к программному обеспечению для создания

тестовых наборов, а также мониторинг формальных свойств [32], основанный на интеграции проверок формальных свойств программного обеспечения в систему мониторинга с целью контроля корректности функционирования системы.

В настоящее время тестирование на основе моделей включает совокупность методов, таких, как проверка согласованности автоматов и систем переходов [33 – 35], разработка тестов на основе формального анализа свойств программного обеспечения [36, 37] и применение методов символического исполнения для построения тестовых сценариев [38].

Синтетические методы объединяют сильные стороны статических и динамических методов и предоставляют оптимальные решения. Так, в работе [39] предлагается синтетический метод верификации ПО, который позволяет покрыть большое количество классов ошибок и решает проблему экспоненциального роста числа состояний системы, которая встречается в формальных методах верификации при охвате всех заданных свойств проверяемой программы.

Выбор оптимально метода верификации

Рассмотрев приведенные методы верификации, можно прийти к выводу, что для верификации криптографических протоколов лучше всего подойдут формальные средства верификации. И действительно, в большинстве случаев при разработке криптографических протоколов используются средства формальной верификации, так как они помогают проверить соответствие логической модели протокола требования безопасности, но при практической реализации на конкретном языке могут возникнуть ошибки. «Таким образом, получается, что изначально сам протокол считался безопасным, но его реализация на деле не является таковой» [40].

Так как большинство средств формальной верификации производят анализ протоколов в абстрактном виде, то это не позволяет проверить протокол полностью. Таким образом, верификация прокола по исходному коду является более актуальной, чем с помощью средств формальной верификации [41].

Поэтому также важна проверка самого кода протокола, так как при переносе логики протокола в код возникают некие допущения из-за специфики языка программирования. Например, в статье [42] представлена ошибка в описании протокола «Zero-Knowledge Succinct Non-Interactive Argument of Knowledge». По сравнению с исходной версией данная схема включает избыточные элементы. Поэтому, при наличии корректного доказательства для одного входа, можно создать корректное доказательство для любого входа.

Теперь рассмотрим анализ исходного кода протокола Bulletproofs [43]. Данный анализ был выполнен Quarkslab, INC (France, Paris) [44].

В процессе оценки безопасности реализации Bulletproof были обнаружены несколько уязвимостей. И все они связаны с отсутствием надлежащих проверок входных данных как для функций вычисления доказательств, так и для функций проверки доказательств. Это приводит к другим алгоритмическим и арифметическим ошибкам, так как входные данные полностью контролируются злоумышленником. Поэтому в функции мультиэкспонентации (Bos-Coster и Pippenger) возникает ошибка, из-за которой ошибочно выводится точка идентичности, либо отбрасывается элемент в вычислении. А в функции двойного скалярного умножения возникает арифметическое переполнение.

Помимо возможности создания неверных выходных значений, они могут быть первыми шагами к тому, чтобы заставить верификатор принять ложное доказательство.

За время оценки ни одна из обнаруженных уязвимостей не привела к практическому использованию, позволяющему либо создать ложное доказательство, принятое проверяющим, либо раскрыть информацию о доказательстве [45].

Хоть найденные уязвимости и не привели к практическому использованию, это не означает, что это невозможно, так как найденные слабые стороны могут превратиться в уязвимости во время эволюции кода.

Заключение

Таким образом, верификации исходного кода протокола показывают те уязвимости, которые присущи конкретной реализации. Это позволяет найти даже самые мелкие уязвимости. Исходя из этого, можно сделать вывод, что для нахождения всех уязвимостей мало использовать формальные средства верификации: необходим комплексный подход к верификации протокола.

Таким образом, на этапе создания протокола его логическую модель нужно проверить с помощью методов формальной верификации, на этапе переноса логики протокола в код необходимо использовать статические анализаторы кода, а на конечном этапе разработки - использовать динамические методы. Такой подход может занять больше времени, но обеспечит большую надёжность и безопасность.

Литература

1. Кузина Е.Л., Полторак А.В., Бозин М.М., Назаров Е.А., Чурилов А.И., Шкуратов Г.И. Планирование и проектирование информационной системы организации: этапы и методы // Инженерный вестник Дона. 2024. №6. URL: ivdon.ru/ru/magazine/archive/n6y2024/9272

2. Юрцев А.Н. Доказательства с нулевым разглашением // Международный журнал гуманитарных и естественных наук. 2023. - №4-4 (79). – С. 138- 141

3. Сиганов И.Д. Доказательство с нулевым разглашением как метод аутентификации в веб-приложениях // Математические структуры и моделирование. 2016. - № 4(40). - С. 143–150

4. Мурзагалиев А.Р. Перспективные направления развития криптографии // Скиф. Вопросы студенческой науки. 2022. - №5 (69). – С. 302-306

5. Ищукова Е. А., Панасенко С. П., Романенко К. С., Салманов В. Д. Криптографические основы блокчейн-технологий – Москва: ООО "ДМК Пресс. Электронные книги", 2022. – 301 с. – ISBN 978-5-9706-0865-4.

6. Пахаев Х.Х., Айгумов Т.Г., Абдулмукминова Ф.М. Роль технологии блокчейн в реализации кибербезопасности // Инженерный вестник Дона. 2022. №10. URL: ivdon.ru/ru/magazine/archive/n10y2022/7958

7. Буренков В. С., Иванов С. Р., Савельев А. Я. Проблемы формальной верификации технических систем // Наука и образование: научное издание МГТУ им. Н.Э. Баумана. – 2012. – № 4. URL: technomag.edu.ru/doc/373672.html

8. Резник С.А., Котенко И.В. Методы и средства верификации для комбинированного анализа протоколов безопасности // Защита информации. Инсайд. 2009. № 3 (27). С. 56-72.

9. IEEE 1012-2004 Standard for Software Verification and Validation. IEEE, 2005, P. 110.

10. Кулямин В. В. Перспективы интеграции методов верификации программного обеспечения // Труды Института системного программирования РАН. 2009. – Том 16. – С. 73-88.

11. IEEE 1028 Standard for Software Reviews. New York: IEEE, 1998, P. 48.

12. Fagan M. E. Design and Code Inspections to Reduce Errors in Program Development // IBM Systems Journal. 1976. V. 15, № 3. pp. 182–211.

13. Kazman R., Bass L., Abowd G., Webb M. SAAM: A Method for Analyzing the Properties of Software Architectures // Proc. 16th International Conference on Software Engineering, 1994. pp. 81–90.

14. Колосов А. П., Рыжков Е. А. Применение статического анализа при разработке программ // Известия Тульского государственного университета. Технические науки. 2008. № 3. С. 185-190

15. Plum Hall Inc. URL: plumhall.com/ (дата обращения: 10.06.2025).

16. LDRA. URL: ldra.com/ (дата обращения: 10.06.2025).

17. Jtest. URL: parasoft.com/ (дата обращения: 10.06.2025).

18. Иванников В.П., Белеванцев А.А., Бородин А. Е., Игнатьев В. Н., Журихин Д. М., Аветисян А.И., Леонов М.И. Статический анализатор Svace для поиска дефектов в исходном коде программ. Труды ИСП РАН, том 26, вып. 1, 2014 г., С. 231-250

19. Бородин А. Е., Белеванцев А. А. Статический анализатор Svace как коллекция анализаторов разных уровней сложности. Труды ИСП РАН, том 27, вып. 6, 2015 г., С. 111-134

20. Кулямин В. В. Методы верификации программного обеспечения. М.: Институт системного программирования, 2008. С. 111.

21. Мерзлякова Е. Ю., Янченко Е. В. Обзор методов верификации и оценки качества программного обеспечения // Вестник СибГУТИ. 2023. Т. 17, № 1. С. 92–106.

22. Automated Validation of Internet Security Protocols and Applications URL: avispa-project.org/ (дата обращения: 11.06.2025).

23. Blanchet B. Modeling and Verifying Security Protocols with the Applied Pi Calculus and ProVerif. Foundations and Trends in Privacy and Security, V. 1 No. 1-2, P. 1-135, October 2016. dx.doi.org/10.1561/3300000004

24. Перевышина Е. А., Бабенко Л. К. Анализ стойкости к атакам криптографических протоколов с использованием формального верификатора SPIN // Известия ЮФУ. Технические науки. 2019. № 5 (207). С. 58–68.

25. Tamarin Prover URL: tamarin-prover.com (дата обращения: 11.06.2025).

26. Cremers C. J. F. Scyther: semantics and verification of security protocols: Technische Universiteit Eindhoven, 2006. Proefschrift. P. 185. – ISBN 90-386-0804-7.

27. Гурин Р. Е., Рудаков И. В., Ребриков А. В. Методы верификации программного обеспечения // Сетевое научное издание. Наука и образование. 2015. № 10. С. 235–251.

28. Splunk AppDynamics Documentation URL: docs.appdynamics.com/ (дата обращения: 20.07.2025).

29. Applications Manager. URL: manageengine.com/products/applications_manager/ (дата обращения: 20.07.2025).

30. SmartBear AQtime URL: syssoft.ru/SmartBear/SmartBear-AQtime/ (дата обращения: 20.07.2025).

31. Utting M., Pretschner A., Legeard B. A Taxonomy of Model-Based Testing. Technical Report, Department of Computer Science, The University of Waikato, New Zealand, 2006, P. 17, DOI:10.1002/stvr.456.

32. Delgado N., Gates A. Q., Roach S. A Taxonomy and Catalog of Runtime Software-Fault Moni toring Tools // IEEE Transactions on Software Engineering. December 2004. V. 30, № 12. pp. 859–872.

33. Broy M., Jonsson B., Katoen J.-P., Leucker M., Pretschner A. Model Based Testing of Reactive Systems. LNCS 3472, Springer, 2005, P. 659.

34. Utting M., Legeard B. Practical Model-Based Testing: A Tools Approach. Morgan-Kaufmann, 2007, P. 433, DOI:10.1016/B978-0-12-372501-1.X5000-5.

35. Ambert F., Bouquet F., Chemin S., Guenaud S., Legeard B., Peureux F., Vacelet N., Utting M. BZ-TT: A tool-set for test generation from Z and B using constraint logic programming // Proc. FATES'2002, August 2002. P. 105–119.

36. Ammann P., Black P. E. Abstracting formal specifications to generate software tests via model checking // Proc. 18th IEEE Digital Avionics Systems Conference, October 1999, V. 2, P. 25, DOI:10.1109/DASC.1999.822091.

37. Engel C., Hahnle R. Generating unit tests from formal proofs // Proc. TAP 2007. LNCS 4454, Springer-Verlag, 2007. P. 169–188.

38. Sen K., Agha G. CUTE and jCUTE: Concolic unit testing and explicit path model-checking tools // Proc. Computer Aided Verification, August 2006. P. 419–423.

39. Рудаков И. В. Разработка и исследование синтетического метода верификации программы с помощью SMT-решателей // Вестник Московского государственного технического университета им. Н. Э. Баумана. Серия Приборостроение. 2016. № 4 (109). С. 49–64.

40. Бабенко Л.К., Писарев И.А. Алгоритм анализа исходного кода C# для извлечения структуры криптографических протоколов // Вопросы кибербезопасности. 2018. №4(28) С. 46-53

41. Бабенко Л.К., Писарев И.А., Верификация безопасности криптографических протоколов по исходным кодам системы электронного голосования с применением множественного бросания бюллетеней // Известия ЮФУ. Технические науки. 2019. - №5 (207). – С. 46-57.

42. Gabizon A. On the Security of the BCTV Pinocchio zk-SNARK Variant. Cryptology ePrint Archive. Paper 2019/119. 2019. P. 9. URL: eprint.iacr.org/2019/119.

43. Benedikt B., Jonathan B., Dan B., Andrew P., Pieter W., Greg M. Bulletproofs: Short Proofs for Confidential Transactions and More // Cryptology ePrint Archive. Paper 2017/1066. 2017. P. 46 URL: eprint.iacr.org/2017/1066

44. Quarkslab Anticipate. Detect. Protect. URL: quarkslab.com/ (дата обращения: 05.06.2025).

45. Evaluation of Bulletproof Implementation URL: 18-06-439-REP-monero-bulletproof-sec-assessment.pdf (дата обращения: 05.06.2025).

References

1. Kuzina E.L., Poltorak A.V., Bozin M.M., Nazarov E.A., Churilov A.I., Shkuratov G.I. Inzhenernyj vestnik Dona 2024. №6. URL: ivdon.ru/ru/magazine/archive/n6y2024/9272

2. Jurcev A.N. Mezhdunarodnyj zhurnal gumanitarnyh i estestvennyh nauk. 2023. №4-4 (79). pp. 138- 141.

3. Siganov I.D. Matematicheskie struktury i modelirovanie. 2016. № 4(40). pp. 143–150.

4. Murzagaliev A.R., Skif. Voprosy studencheskoj nauki. 2022. №5 (69). pp. 302-306.

5. Ishhukova E. A., Panasenko S. P., Romanenko K. S., Salmanov V. D. Kriptograficheskie osnovy blokchejn-tehnologij [Cryptographic foundations of blockchain technologies]. Moskva: OOO "DMK Press. Jelektronnye knigi", 2022. 301 p. ISBN 978-5-9706-0865-4.

6. Pahaev H.H., Ajgumov T.G., Abdulmukminova F.M. Inzhenernyj vestnik Dona 2022. №10. URL: ivdon.ru/ru/magazine/archive/n10y2022/7958.

7. Burenkov, V. S., Ivanov S. R., Savel'ev A. Ya. Nauka i obrazovanie: nauchnoe izdanie MGTU im. N.Je. Baumana. 2012. № 4. URL: technomag.edu.ru/doc/373672.html.

8. Reznik S.A., Kotenko I.V. Zashhita informacii. Insajd. 2009. № 3 (27). pp. 56-72.

9. IEEE 1012-2004 Standard for Software Verification and Validation. IEEE, 2005, P. 110.

10. Kuljamin V. V. Trudy Instituta sistemnogo programmirovanija RAN. 2009. Tom 16. pp. 73-88.

11. IEEE 1028 Standard for Software Reviews. New York: IEEE, 1998, P. 48.

12. Fagan M. E. IBM Systems Journal. 1976. V. 15, № 3. pp. 182–211.

13. Kazman R., Bass L., Abowd G., Webb M. Proc. 16th International Conference on Software Engineering, 1994. pp. 81–90.

14. Kolosov A. P., Ryzhkov E. A. Izvestija Tul'skogo gosudarstvennogo universiteta. Tehnicheskie nauki. 2008. № 3. pp. 185-190

15. Plum Hall Inc URL: plumhall.com (date of access: 10.06.2025).

16. LDRA URL: ldra.com (date of access: 10.06.2025).

17. Jtest URL: parasoft.com (date of access: 10.06.2025).

18. Ivannikov V.P., Belevancev A.A., Borodin A. E., Ignat'ev V. N., Zhurihin D. M., Avetisjan A.I., Leonov M.I. Trudy ISP RAN, tom 26, vyp. 1, 2014 g., pp. 231-250

19. Borodin A. E., Belevancev A. A. Trudy ISP RAN, tom 27, vyp. 6, 2015 g., pp. 111-134

20. Kuljamin V. V. Metody verifikacii programmnogo obespechenija [Software verification methods]. M.: Institut sistemnogo programmirovanija, 2008. P. 111.

21. Merzljakova E. Yu., Janchenko E. V. Vestnik SibGUTI. 2023. T. 17, № 1. pp. 92–106.

22. Automated Validation of Internet Security Protocols and Applications URL: avispa-project.org (date of access: 11.06.2025).

23. Blanchet B. Foundations and Trends in Privacy and Security, Vol. 1 No. 1-2, pp. 1-135, October 2016. dx.doi.org/10.1561/3300000004

24. Perevyshina E. A., Babenko L. K. Izvestija JuFU. Tehniche skie nauki. 2019. № 5 (207). pp. 58–68.

25. Tamarin Prover URL: tamarin-prover.com (date of access: 11.06.2025).

26. Cremers C. J. F. Scyther: semantics and verification of security Protocols: Technische Universiteit Eindhoven, 2006. Proefschrift. 185 p. ISBN 90-386-0804-7.

27. Gurin R. E., Rudakov I. V., Rebrikov A. V. Setevoe nauchnoe izdanie. Nauka i obrazovanie. 2015. № 10. pp. 235–251.

28. Splunk AppDynamics Documentation URL: docs.appdynamics.com (date of access: 20.07.2025).

29. Applications Manager. URL: manageengine.com/products/applications_manager (date of access: 20.07.2025).

30. SmartBear AQtime URL: syssoft.ru/SmartBear/SmartBear-AQtime (date of access: 20.07.2025).

31. Utting M., Pretschner A., Legeard B. A Taxonomy of Model-Based Testing. Technical Report, Department of Computer Science, The University of Waikato, New Zealand, 2006, P. 17, DOI:10.1002/stvr.456.

32. Delgado N., Gates A. Q., Roach S. IEEE Transactions on Software Engineering. December 2004. V. 30, № 12. pp. 859–872.

33. Broy M., Jonsson B., Katoen J. P., Leucker M., Pretschner A. Model Based Testing of Reactive Systems. LNCS 3472, Springer, 2005, P. 659.

34. Utting M., Legeard B. Practical Model-Based Testing: A Tools Approach. Morgan-Kaufmann, 2007. P. 433, DOI:10.1016/B978-0-12-372501-1.X5000-5.

35. Ambert F., Bouquet F., Chemin S., Guenaud S., Legeard B., Peureux F., Vacelet N., Utting M. Proc. FATES'2002, August 2002. pp. 105–119.

36. Ammann P., Black P. E. Proc. 18th IEEE Digital Avionics Systems Conference, October 1999, P. 25, DOI:10.1109/DASC.1999.822091.

37. Engel C., Hahnle R. Proc. TAP 2007. LNCS 4454, Springer-Verlag, 2007. pp. 169–188.

38. Sen K., Agha G. Proc. Computer Aided Verification, August 2006. pp. 419–423.

39. Rudakov I. V. Vestnik Moskovskogo gosudarstvennogo tehnicheskogo universiteta im. N. Je. Baumana. Serija Priborostroenie. 2016. № 4 (109). pp. 49–64.

40. Babenko L.K., Pisarev I.A. Voprosy kiberbezopasnosti. 2018. №4(28). pp. 46-53.

41. Babenko L.K., Pisarev I.A., Izvestija JuFU. Tehnicheskie nauki. 2019. №5 (207). pp. 46-57.

42. Gabizon A. On the Security of the BCTV Pinocchio zk-SNARK Variant. Cryptology ePrint Archive. Paper 2019/119. 2019. P. 9. URL: eprint.iacr.org/2019/119.

43. Benedikt B., Jonathan B., Dan B., Andrew P., Pieter W., Greg M. Bulletproofs: Short Proofs for Confidential Transactions and More Cryptology ePrint Archive. Paper 2017/1066. 2017. P. 46. URL: eprint.iacr.org/2017/1066

44. Quarkslab Anticipate. Detect. Protect. URL: quarkslab.com (date of access: 05.06.2025).

45. Evaluation of Bulletproof Implementation. URL: 18-06-439-REP-monero-bulletproof-sec-assessment.pdf (date of access: 05.06.2025).

Дата поступления: 12.09.2025

Дата публикации: 25.10.2025