

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Политехнический институт
Факультет приборостроения,
информационных технологий и
электроники

Специальность 10.05.03 –

Специализация –

Кафедра
«Информационная безопасность
систем и технологий»

Информационная безопасность
автоматизированных систем

Защищенные автоматизированные
системы управления

ДИПЛОМНЫЙ ПРОЕКТ

на тему

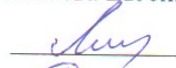
ПРАКТИКУМ ПО АНАЛИЗУ УЯЗВИМОСТЕЙ КРИПТОГРАФИЧЕСКИХ
ПРОТОКОЛОВ

Студент



Канаева Евгения Александровна

Руководитель



Липилин О.В.

Нормоконтролёр



Фатеев А.Г.

Рецензент зам. ген. директора по



Сомкин С.А.

специальным вопросам АО «ПНИЭИ»

Работа допущена к защите (протокол заседания кафедры от 21.06.19 № 16)

Заведующий кафедрой



С.Л. Зефилов

Работа защищена с отметкой Отлично (протокол заседания ГЭК от 25.06.19 № 19)

Секретарь ГЭК



А.П. Иванов

Проверка ВКР Лобанов
С.Л. Зефилов
Пенза 2019

Оглавление

Введение.....	4
1 Обзор методов анализа уязвимостей криптографических протоколов.....	5
1.1 Понятие криптографического протокола, виды криптографических протоколов, свойства безопасности.....	5
1.2 Атаки на криптографические протоколы.....	8
1.3 Классификация методов анализа уязвимостей криптографических протоколов.....	10
2 Обзор средств анализа уязвимостей криптографических протоколов.....	14
3 Разработка протоколов с уязвимостями.....	22
3.1 Разработка протокола односторонней аутентификации со внесением уязвимости в виде отсутствия защиты от атак повтора.....	22
3.2 Разработка протокола взаимной аутентификации со внесением уязвимостей в виде отсутствия защиты от атак повтора и отсутствия контроля целостности потока сообщений.....	29
3.3 Разработка протокола передачи данных с обеспечением свойства конфиденциальности со внесением уязвимостей в виде отсутствия защиты от атак повтора и отсутствия контроля целостности сообщений.....	40
3.4 Разработка протокола передачи данных с обеспечением свойства неотказуемости со внесением уязвимостей в виде отсутствия защиты от атак повтора, отсутствия контроля целостности потока сообщений, вычисления цифровой подписи не от всех полей сообщения, а также отсутствия аутентификации одного из участников протокола.....	47
4 Разработка виртуального стенда для проведения лабораторного практикума.....	60
5 Разработка рекомендаций по проведению лабораторного практикума...	63
5.1 Разработка методики проведения практикума.....	63

5.2 Разработка руководства пользователя.....	65
6 Вредные психофизиологические факторы, влияющие на органы зрения специалиста по защите информации. Методы и средства физической культуры, снижающие их воздействие.....	88
Заключение.....	91
Список используемых источников.....	92

Введение

При подготовке специалистов по информационной безопасности автоматизированных систем предусмотрено изучение дисциплины «Криптографические протоколы», в которой у студентов формируется компетенция ПК-3 «способность проводить анализ защищенности автоматизированных систем». При освоении данной компетенции студенты должны знать основные уязвимости криптографических протоколов и уметь проводить анализ их защищенности.

На сегодняшний момент для анализа защищенности криптографических протоколов применяются различные средства автоматизированного анализа, поэтому в рамках изучения дисциплины на практических занятиях студенты должны получить навыки работы с современными средствами анализа уязвимостей. Одним из таких средств является Tamarin Prover. Данное программное средство является одним из новейших инструментов для символьного моделирования и анализа криптографических протоколов. Сочетание автоматизированного и интерактивного режима построения доказательств позволит студентам наиболее детально понять процесс анализа уязвимостей.

Таким образом, разработка практикума по анализу уязвимостей криптографических протоколов с использованием Tamarin Prover является актуальной.

1 Обзор методов анализа уязвимостей криптографических протоколов

1.1 Понятие криптографического протокола, виды криптографических протоколов, свойства безопасности

Криптографический протокол — протокол, предназначенный для выполнения функций криптографической системы, в процессе выполнения которого участники используют криптографические алгоритмы [1]. Протокол основан на наборе правил, определяющих использование криптографических преобразований и алгоритмов в информационном процессе.

Криптографические протоколы используются в распределенных системах для аутентификации, а также для различных целей, связанных с обеспечением контроля доступа, конфиденциальности и неотказуемости.

По целевому назначению криптографические протокола делятся на следующие виды:

- а) протоколы обеспечения целостности сообщений:
 - 1) протоколы с аутентификацией источника;
 - 2) протоколы без аутентификации источника;
- б) протоколы цифровой подписи:
 - 1) протоколы индивидуальной или групповой цифровой подписи;
 - 2) протоколы с восстановлением или без восстановления сообщения;
 - 3) протоколы цифровой подписи вслепую;
 - 4) протоколы конфиденциальной цифровой подписи;
 - 5) протоколы цифровой подписи с доказуемостью подделки;
- в) протоколы идентификации (аутентификации участников):
 - 1) протоколы односторонней аутентификации;
 - 2) протоколы двусторонней (взаимной) аутентификации;
- г) протоколы конфиденциальной передачи:

- 1) протоколы обмена сообщениями;
 - 2) протоколы широковещательной или циркулярной передачи;
 - 3) протоколы честного обмена секретами;
 - 4) протоколы забывающей передачи;
 - 5) протоколы привязки к биту (строке);
- д) протоколы распределения ключей:
- 1) протоколы предварительного распределения ключей;
 - 2) протоколы передачи ключа (ключевого обмена);
 - 3) протоколы совместной выработки ключа (открытого распределения ключей);
 - 4) протоколы разделения секрета.

Так как криптографические протоколы могут обеспечивать различные функции безопасности, имеется широкий набор требований к криптографическим протоколам (иными словами, необходимых свойств, характеризующих безопасность криптографических протоколов). Конкретные требования к тому или иному криптографическому протоколу могут определяться в зависимости от системы, в которой будет применяться данный криптографический протокол.

В настоящее время в документах международной организации Internet Engineering Task Force (IETF) определено двадцать свойств безопасности, которые распределены по десяти группам. Данные свойства представлены в таблице 1.

Таблица 1 – Свойства безопасности криптографических протоколов

Код свойства	Наименование свойства
1. Нешироковещательная аутентификация	
G1	Аутентификация субъекта
G2	Аутентификация сообщения
G3	Защита от повтора

Продолжение таблицы 1

Код свойства	Наименование свойства
2. Аутентификация при рассылке по многим адресам или при подключении к службе подписки/уведомления	
G4	Неявная (скрытая) аутентификация получателя
G5	Аутентификация источника
3. Авторизация	
G6	Авторизация (доверенной третьей стороной)
4. Свойства совместной генерации ключа	
G7	Аутентификация ключа
G8	Подтверждение правильности ключа
G9	Защищенность от чтения назад
G10	Формирование новых ключей
G11	Защищенная возможность договориться о параметрах безопасности
5. Конфиденциальность	
G12	Конфиденциальность
6. Анонимность	
G13	Защита идентификаторов от прослушивания (несвязываемость)
G14	Защита идентификаторов от других участников
7. Ограниченная защищенность от атак типа «отказ в обслуживании»	
G15	Ограниченная защищенность от атак типа «отказ в обслуживании»
8. Инвариантность отправителя	
G16	Инвариантность отправителя
9. Невозможность отказа от ранее совершенных действий	
G17	Подотчетность
G18	Доказательство источника
G19	Доказательство получателя
10. Безопасное временное свойство	
G20	Безопасное временное свойство

При разработке и внедрении криптографических протоколов необходимо проводить их анализ на предмет обеспечения требуемых свойств безопасности.

Невыполнение свойства безопасности криптографического протокола означает наличие в нем уязвимости.

1.2 Атаки на криптографические протоколы

Среди основных атак, использующих уязвимости криптографических протоколов можно выделить следующие:

а) подмена — попытка подменить одного пользователя другим. В ходе проведения такой атаки нарушитель выступает от имени одной из сторон протокола и полностью имитирует ее действия, при этом получая в ответ сообщения определенного формата, необходимые для подделки отдельных шагов протокола. Методы противодействия от атак подмены состоят в:

- 1) сохранении в тайне от противника информации, определяющей алгоритм идентификации;
- 2) использовании различных форматов сообщений, передаваемых на разных шагах протокола;
- 3) вставке в сообщения специальных идентификационных меток и номеров сообщений;

б) повторное навязывание сообщения — повторное использование ранее переданного в текущем или предыдущем сеансе сообщения или какой-либо его части в текущем сеансе протокола. Например, повторная передача информации, переданной в сообщении ранее проведенного протокола идентификации, может привести к повторной успешной идентификации того же самого или другого пользователя. Методы противодействия атакам повтора состоят в обеспечении целостности сеанса и невозможности вставки в него лишних сообщений. Для этого используются следующие приёмы:

- 1) построение протокола по типу «запрос — ответ»;
- 2) включение в передаваемые сообщения временных меток, случайных чисел или возрастающих последовательностей чисел;

в) атака с использованием параллельных сеансов — специальное одновременное открытие противником нескольких параллельных сеансов с целью использования сообщений из одного сеанса в другом. В качестве метода противодействия подобным атакам применяется включение в сообщения протокола специальных идентификационных меток сеанса;

г) атака с использованием специально подобранных текстов — атака на протоколы типа «запрос — ответ», при которой противник по определенному правилу выбирает запросы с целью получить информацию о долговременном ключе доказывающего. Методы противодействия данной атаке состоят в:

- 1) включении случайных чисел в запросы или ответы;
- 2) использовании протоколов с нулевым разглашением;

д) использование противником собственных средств в качестве части телекоммуникационной структуры — атака, при которой в протоколе, реализуемом между участниками А и В, противник подключается к телекоммуникационному каналу и становится его частью при обмене сообщениями между А и В. Данная атака известна под именем «человек посередине». В качестве методов противодействия подобным атакам возможно применение:

- 1) шифрования сообщений;
- 2) механизма цифровой подписи.

При анализе криптографических протоколов необходимо учитывать все действия и возможности противника, анализирующего криптографический протокол, существующие и возможные методы атак на криптографические протоколы, а также уязвимости, заранее имеющиеся в самих криптографических протоколах. Иными словами, необходимо гарантировать корректность криптографического протокола.

Корректность — это способность криптографического протокола противостоять угрозам со стороны противника или нарушителя, не располагающего необходимой секретной информацией, но пытающегося

выполнить протокол за участника, который по определению должен такой информацией владеть. При этом под нарушителем понимается участник протокола, нарушающий определенные протоколом действия, а под противником – внешний субъект, наблюдающий за передаваемыми сообщениями и, возможно, вмешивающийся в работу участников путем перехвата, искажения, вставки, повтора и перенаправления сообщений, блокирования передачи с целью нарушения одного или нескольких свойств безопасности криптографического протокола.

1.3 Классификация методов анализа уязвимостей криптографических протоколов

Анализ криптографических протоколов сталкивается с трудностями, поскольку такие протоколы обладают повышенными свойствами безопасности по сравнению с обычными коммуникационными протоколами.

Однако повышенные свойства безопасности криптографических протоколов создают препятствия не только для злоумышленников, но и для разработчиков таких протоколов, а также лиц, рассматривающих те или иные протоколы для применения в рамках сетей и систем, так как не позволяет вовремя обнаруживать имеющиеся уязвимости.

Для поиска уязвимостей криптографических протоколов применяются различные методы анализа. Наиболее часто для классификации таких методов используется классификация, предложенная Мидоуз [2]:

а) моделирование и верификация протоколов с использованием спецификационных языков и инструментов для верификации, которые не были специально разработаны для анализа безопасности. Основная идея данного подхода заключается в том, чтобы истолковать криптографический протокол как любую другую программную систему и попытаться доказать ее корректность. Критика данного подхода заключается в том, что из доказательства

корректности не обязательно следует обеспечение безопасности. Применение подобных методов и методик для анализа безопасности протоколов сталкивается с тем, что не рассматриваются некоторые тонкости, специфические для области обеспечения безопасности;

б) разработка экспертных систем для создания и изучения различных сценариев, в которых исследуются свойства безопасности протоколов. Верификация в таких системах стартует с некоторого нежелательного состояния, и производится попытка обнаружить, достижимо ли данное нежелательное состояние из некоторого начального состояния. Данный подход позволяет более основательно осуществлять поиск уязвимостей, чем при использовании универсальных формальных методов, однако он также не гарантирует безопасность протоколов и не предполагает автоматизированных методик для анализа атак на протокол. Экспертные системы позволяют обнаружить, действительно ли рассматриваемый протокол содержит некий дефект, но с весьма малой вероятностью смогут обнаружить неизвестные заранее уязвимости протокола;

в) моделирование требований к протоколам с использованием логик, разработанных специально для анализа знаний и доверия. Формальные системы анализа на основе логик используют дедуктивный подход, основанный на логической проверке корректности рассматриваемого протокола. Проверяемые свойства формулируются как утверждения в рамках некоторой логики с определенными синтаксисом и семантикой и для верификации протокола ищется логический вывод данных утверждений. Данный подход применяется, как правило, для доказательства корректности системы. Целью анализа является вывод утверждения, которое будет представлять корректность протокола. Невозможность вывода такого утверждения означает, что протокол, вероятно, некорректен. Преимуществом такого подхода является возможность применения техники автоматического доказательства теорем. При анализе протоколов обычно используют следующие модальные логики [3]:

1) дохастическую логику, или логику доверия – систему выводов, которая применяет правила о том, как вывести новое доверие на основании имеющегося доверия;

2) эпистемическую логику, основанную на знаниях;

3) темпоральную (или временную) логику, описывающую последовательности состояний и анализирующую правила вывода формул двух типов: формулы состояния (обращаются в истину в некотором состоянии) и формулы пути (обращаются в истину на протяжении некоторого пути);

г) разработка формальной модели, основанной на алгебраических свойствах криптографических систем.

Описанная классификация также отражена на рисунке 1.



Рисунок 1 – Классификация Мидоуз

Гритзалис и Спинелис предлагают несколько иную классификацию. Согласно данной классификации подходы к анализу протоколов разделяются на две большие группы:

а) методы построения логических выводов, использующие специализированные логики, основанные на понятиях знания и доверия о том, что участники протокола могут достичь желаемых результатов;

б) методы построения возможных атак проникновения в защищенную систему с использованием алгебраических свойств алгоритмов, лежащих в основе протоколов.

Различные методы анализа уязвимостей криптографических протоколов могут применяться совместно в целях выявления большего количества уязвимостей.

2 Обзор средств анализа уязвимостей криптографических протоколов

Для автоматизированного анализа уязвимостей криптографических протоколов применяются средства двух типов:

- средства дедуктивного логического вывода;
- средства верификации.

Средства дедуктивного логического вывода используют дедуктивный подход, основанный на логической проверке корректности рассматриваемого протокола. При таком подходе свойства протокола формулируются как утверждения в рамках некоторой логики, и для их проверки ищется логический вывод данных утверждений. Такие средства применяются для доказательства корректности системы, того, что протокол действительно удовлетворяет определённым требованиям.

Средства дедуктивного логического вывода можно подразделить на три основные группы, в основе которых лежат соответственно:

- логика доверия и генерация теории;
- автоматическое доказательство теорем;
- логическое программирование.

Средства верификации основаны на использовании методов верификации, прежде всего метода проверки на модели. При использовании данного метода по протоколу строится формальная модель – система переходов, состояниями которой являются множества высказываний о свойствах протокола. Затем построенная модель проверяется на удовлетворение некоторому свойству безопасности в каждом своем состоянии, которого можно достичь, выходя из определенного множества начальных состояний.

Основные трудности такого подхода связаны с бесконечностью числа состояний моделирующей системы, что требует необходимости сведения к конечной поведенческой модели. С другой стороны, в результате строится контрпример, состоящий из траектории, ведущей к опасному состоянию

системы, называемому состоянием атаки, тем самым явно указывается атака на протокол.

Средства верификации можно разделить на следующие основные группы:

- основанные на автоматах;
- основанные на пространстве нитей;
- основанные на логике.

В рамках настоящей главы будут рассмотрены несколько средств автоматизированного анализа уязвимостей криптографических протоколов, являющихся наиболее известными либо одними из наиболее эффективных.

NRL Protocol Analyzer – одно из первых средств автоматизированного анализа уязвимостей криптографических протоколов, разработанное в 1996 году. NRL Protocol Analyzer автоматически генерирует инварианты с целью ограничения потенциально бесконечного пространства состояний. Полученное пространство состояний исследуется на возможность проведения атак на протоколы и для доказательства безопасности протоколов. Движение в пространстве состояний начинается из небезопасного состояния, и с помощью анализатора производится попытка обнаружить состояние, из которого существовал бы маршрут в данное небезопасное состояние и которое бы принадлежало множеству начальных состояний. Фактически данный процесс аналогичен доказательству теоремы. Во время процесса верификации требуется вмешательство пользователя, кроме того, данный процесс не всегда можно завершить. NRL Protocol Analyzer снабжен языком темпоральной логики NPATRL, позволяющим определить требования к протоколу в терминах желательной или нежелательной последовательности событий.

AVISPA позволяет анализировать протокол одним из четырех выходных модулей: TA4SP, SATMC, OFMC, CL-AtSe. Спецификация протокола, основанная на ролевом представлении, записывается на языке высокого уровня HPSL, а затем транслируется во внутренний язык IF. Проверяемые свойства описываются в терминах темпоральной логики. Модуль TA4SP реализует

технику, основанную на построении древовидных автоматов и развитую для систем автоматического доказательства. Строится верхняя аппроксимация древовидного автомата, реализующего систему переписывания термов, которая описывает максимальные знания нарушителя. Исследование свойства конфиденциальности в таком случае сводится к проверке наличия в данном множестве терма, содержащего секрет. Модули SATMC, OFMC, CL-AtSe осуществляют верификацию методом проверки на модели. При использовании данного метода протокол представляется как бесконечная система переходов, а задача верификации сводится к проверке выполнимости формулы, решения которой соответствуют атакам на протокол. Для сведения к конечному случаю применяются разные подходы. В модуле SATMC используются методы, разработанные в рамках теории решения задач планирования, в модуле OFMC – символический метод, позволяющий группировать различные состояния в бесконечные классы, а в CL-AtSe применяется техника, основанная на построении ограничений.

Scyther верифицирует ограниченное и неограниченное число сеансов протокола. Данное программное средство использует символический анализ в сочетании с обратным поиском, основанный на частично упорядоченных шаблонах. Scyther не требует задания сценария атаки. Необходимо только задание параметров, ограничивающих либо максимальное число запусков, либо пространство перебираемых траекторий. В первом случае Scyther всегда дает результат и показывает найденные траектории атаки. Во втором случае завершение процесса верификации не гарантировано. В качестве ответа возможен один из трех вариантов:

- установлено, что проверяемое свойство выполнено;
- свойство не выполнено, так как найдена атака;
- свойство может быть корректно для заданного пространства траекторий.

HERMES использует для задания спецификаций протоколов и их свойств язык верхнего уровня LaEva. Данный язык транслируется во внутреннее представление на языке cpl. HERMES позволяет проверять свойства протокола при неограниченной длине сообщения и неограниченном числе участников. Данное программное средство дает на выходе условия на исходные знания противника, которые гарантируют, что он не сможет узнать секрета. HERMES не оценивает знания нарушителя, а исследует множество безопасных сообщений. Для аппроксимации бесконечного множества сообщений используется символическое представление, основанное на шаблонах. Если в результате найдена атака, то предоставляется траектория атаки. В случае, когда в результате получается ответ, что свойство доказано, выдается также дерево полного доказательства, что бывает полезным при сертификации протокола.

ProVerif анализирует неограниченное число сеансов протокола с использованием верхней аппроксимации и представления протокола с помощью хорновских выражений. ProVerif предлагает два типа входных файлов: хорновские выражения и подмножество Pi-исчисления. При использовании Pi-исчисления ProVerif основывается на описании множества процессов, каждый из которых может выполняться неограниченное число раз. На выходе возможны четыре ситуации:

- свойство не выполнено;
- доказано, что свойство выполнено;
- свойство не может быть доказано, так как есть пример атаки (могут быть найдены ложные атаки);
- работа не завершается.

ProVerif моделирует множество траекторий, соответствующих определенному сценарию, и осуществляет полный перебор возможных траекторий.

Tamarin prover принимает в качестве входных данных модель протокола безопасности, определяющую действия, выполняемые агентами,

выполняющими протокол в различных ролях (например, инициатор протокола), спецификацию злоумышленника и спецификацию требуемых свойств протокола [4]. Данные описываются с помощью специализированного языка, при этом используются такие сущности, как теоремы, леммы и правила. Правила определяют систему переходов, состояние которой состоит из символического представления знаний противника, сообщений в сети, информации о недавно сгенерированных значениях и состоянии протокола. Противник и протокол взаимодействуют, обновляя сетевые сообщения и генерируя новые сообщения. Tamarin prover поддерживает спецификацию некоторых криптографических операторов, например, возведение в степень Диффи-Хеллмана. Свойства безопасности моделируются как свойства трасс, сверяются со следами системы переходов, иными словами, проверяется эквивалентность двух систем переходов.

Tamarin prover предлагает два режима анализа: автоматизированный и интерактивный. Автоматизированный режим использует дедукцию и регулярные выражения в сочетании с эвристическим подходом для поиска доказательств соответствия. Результатом является либо доказательство правильности (для неограниченного количества ролей и сгенерированных значений), либо контрпример, представляющий собой успешную атаку. Интерактивный режим Tamarin Prover позволяет производить мониторинг состояния доказательств, проверить графики атаки, тем самым комбинируя методы ручного и автоматизированного поиска доказательств. Является новейшим инструментом среди рассматриваемых в настоящем отчете.

Для сравнения описанных в настоящей главе средств анализа уязвимостей криптографических протоколов будут применены следующие критерии:

- уровень автоматизации (низкий уровень автоматизации неудобен при использовании и порождает ошибки из-за невнимательности либо недостатка знаний анализирующего лица);

- гибкость в смысле выразительных свойств модели (наличие ограничений на моделируемые свойства безопасности сужают пространство уязвимостей, которые возможно выявить при использовании того или иного программного средства);
- ограничения на модель в теоретико-вычислительном смысле (моделирование более сложных конфигураций позволяет анализировать большее число протоколов);
- ложные срабатывания при определении уязвимостей протокола;
- извлечение алгоритма атаки (возможность узнать способ использования уязвимости протокола предпочтительнее по сравнению с индикацией наличия уязвимости);
- наличие сообщества исследователей и/или разработчиков, использующих данное средство верификации (его наличие косвенным образом свидетельствует о достаточной выразительности средства и привлекательности для решения реальных задач).

Результаты сравнения были сведены в таблицу 2. Как видно из таблицы 2, среди рассматриваемых из средств анализа уязвимостей криптографических протоколов ни одно не удовлетворяет всем выработанным критериям. Однако Tamarin Prover обладает высокой гибкостью подхода и при этом позволяет извлекать алгоритм атаки в доступном для понимания виде.

Таблица 2 – Сравнение средств анализа уязвимостей криптографических протоколов

Критерий	NRL Protocol Analyzer	AVISPA	Scyther	HERMES	ProVerif	Tamarin prover
Уровень автоматизации	Автоматизированная верификация	Автоматическая верификация	Автоматическая верификация	Автоматическая верификация	Автоматическая верификация	Автоматическая верификация
Гибкость подхода	Нет	С точки зрения подходов к моделированию — высокая, с точки зрения моделируемых свойств - низкая	Нет	Высокая	Нет	Высокая
Ложные срабатывания	Возможны	Нет	Нет	Нет	Возможны	Нет

Продолжение таблицы 2

Критерий	NRL Protocol Analyzer	AVISPA	Scyther	HERMES	ProVerif	Tamarin prover
Ограничения	Фиксированный набор свойств, подлежащих верификации	Зависят от применяемого подхода	Фиксированный набор свойств, подлежащих верификации	Фиксированный набор свойств, подлежащих верификации	Фиксированный набор свойств, подлежащих верификации	Фиксированный набор свойств, подлежащих верификации
Извлечение алгоритма атаки	Есть возможность	Есть возможность	Есть возможность	Есть возможность, интерпретация результатов вызывает сложность	Не гарантировано	Есть возможность
Сообщество разработчиков	Нет	Есть	Нет	Есть	Нет	Есть

3 Разработка протоколов с уязвимостями

3.1 Разработка протокола односторонней аутентификации со внесением уязвимости в виде отсутствия защиты от атак повтора

Был разработан протокол односторонней аутентификации, содержащий уязвимость — отсутствие проверки уникальности нонса.

Целью данного протокола является передача информации (нонса) от участника A участнику B с обеспечением идентичности A и аутентичности цифровой подписи A . Нонс может впоследствии использоваться, например, в качестве идентификатора сеанса.

Предполагается, что участники A и B имеют безопасную схему цифровой подписи, A имеет ключ подписи SK_A , B имеет соответствующий ключ проверки подписи PK_A . Также B имеет свой ключ подписи SK_B , и A имеет соответствующий ключ проверки подписи PK_B . Оба участника имеют ключевые запоминающие устройства $KЗУ_A$ и $KЗУ_B$. Также участник A имеет датчик случайных чисел $ДСЧ_A$, который используется для генерации случайных чисел — нонсов N_A .

При получении сообщения участником B информация, содержащаяся в сообщении, считается недоказанной и обозначается знаком $\sim$. После проверки подписи информация считается доказанной и знак $\sim$ снимается.

Формальное описание данного протокола:

1. A :

$$KЗУ_A \rightarrow SK_A$$

$$ДСЧ_A \rightarrow N_A$$

$$S_{AB} = SK_A(A||N_A)$$

$$[A||N_A||S_{AB}] \Rightarrow B$$

2. B :

$$[\tilde{A}||\tilde{N}_A||\tilde{S}_{AB}]$$

$$\begin{aligned}
& KZU_B \rightarrow PK_{\tilde{A}} \\
& PK_A(\tilde{A} || \tilde{N}_{\tilde{A}}, \tilde{S}_{AB}) \stackrel{?}{=} 1 \\
& \tilde{A} = A \quad \tilde{N}_{\tilde{A}} = N_A
\end{aligned}$$

Ниже приведено словесное описание протокола.

Участник A :

- а) генерирует случайное число N_A с использованием датчика случайных чисел $ДСЧ_A$;
- б) формирует блок $[A || N_A]$ и вычисляет на него цифровую подпись S_{AB} на своем ключе SK_A ;
- в) формирует сообщение $[A || N_A || S_{AB}]$ и посылает участнику B по открытому каналу связи.

Участник B :

- а) формирует блок $[\tilde{A} || \tilde{N}_{\tilde{A}}]$ из принятых значений $\tilde{A}$, $\tilde{N}_{\tilde{A}}$ и проверяет принятую цифровую подпись $\tilde{S}_{AB}$ с использованием ключа проверки подписи PK_A в соответствии с принятым адресом отправителя $\tilde{A}$;
- б) при отрицательном результате проверки сообщение не принимается; при положительном результате делается вывод о том, что $\tilde{A}$ равно A , $\tilde{N}_{\tilde{A}}$ равно N_A , сообщение принимается, доказывающий A считается опознанным.

Описание протокола в нотации Tamarin:

theory auth1

begin

builtins: signing

rule Register_pk:

[Fr(~ltkX)]
 -->[!Ltk(\$X, ~ltkX)
 , !Pk(\$X, pk(~ltkX))
 , Out(pk(~ltkX))]

rule Reveal_ltk:

```

[ !Ltk($X, ltkX) ]
--[Reveal($X)]->
[ Out(ltkX)]

```

rule A_1_send:

```

let m1 = <$A, ~na>
    sign_a = sign(m1, ltkA)
in
    [ Fr(~na)
      , !Ltk($A, ltkA)]
    --[ Send($A, ~na)]->
    [ St_A_1($A, ltkA, ~na)
      , Out(<m1, sign_a>)]

```

rule B_1_receive:

```

let m1 = <$A, na>
in
    [ !Pk($A, pkA)
      , In(<m1, sign_a_check>)]
    --[ Recv($B, na)
      , Eq(verify(sign_a_check,m1,pkA),true)
      , Authentic($A,na), Honest($B), Honest($A)]->
    [ St_B_1($B, pkA, $A, <$A, na>)]

```

restriction Equality:

"All x y #i. Eq(x, y) @i ==> x=y"

lemma executable:

exists-trace

"Ex A B m #i #j. Send(A,m)@i & Recv(B,m) @j"

lemma message_authentication:

"All b m #i. Authentic(b,m) @i
==> (Ex #j. Send(b,m) @j & j<i)

| (Ex B #r. Reveal(B)@r & Honest(B) @i & r < i)"

end

Для идентификации внесенной уязвимости необходимо внести в него следующие изменения:

— добавить к описанию лемму, проверяющую уникальность нонса.

Пример подобной леммы:

lemma distinct_nonces:

"All n #i #j. CheckNonce(n)@i & CheckNonce(n)@j ==> #i=#j";

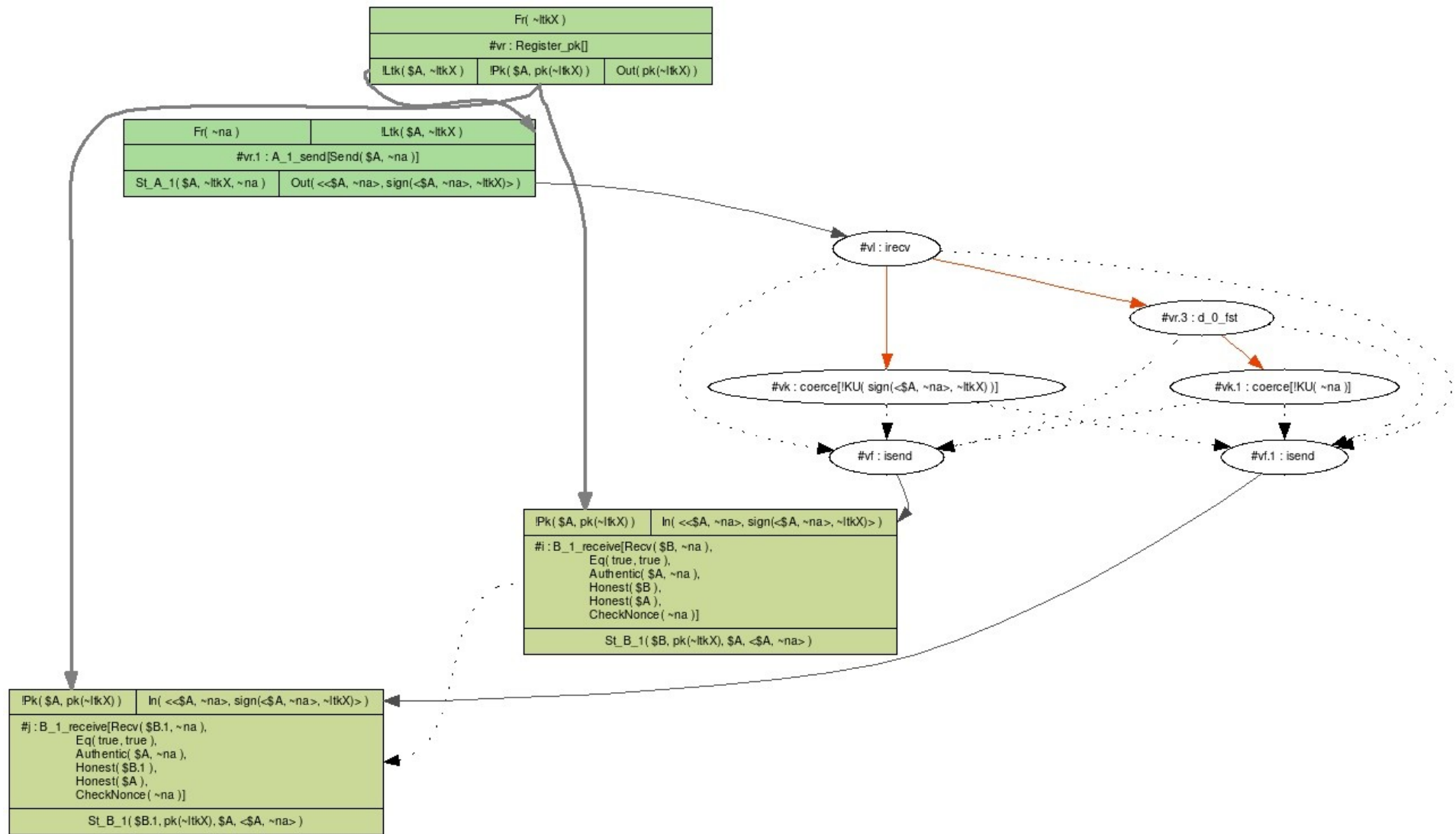
— добавить в правило B_1_receive факт действия CheckNonce(na).

В ходе проверки указанной леммы Tamarin prover окрашивает лемму в красный цвет, что показано на рисунке 2, и строит вектор атаки, изображенный на рисунке 3.

```
lemma distinct_nonces:
  all-traces
  "v n #i #j.
    ((CheckNonce( n ) @ #i) ^
    (CheckNonce( n ) @ #j)) -> (#i = #j)"
  simplify
  solve( (#i < #j) | (#j < #i) )
  case case_1
  solve( !Pk( $A, pk(x) ) > #i )
  case Register_pk
  solve( !Pk( $A.1, pk(x) ) > #j )
  case Register_pk
  solve( !KU( sign(<$A, n>, ~ltkX) ) @
  #vk.6 )
  case A_1_send
  solve( !KU( sign(<$A.1, ~na>,
  ~ltkX.1) ) @ #vk.8 )
  case A_1_send
  solve( !KU( ~na ) @ #vk.4 )
  case A_1_send
  SOLVED // trace found
  qed
  qed
  qed
  qed
  qed
  qed
```

Рисунок 2 – Идентификация отсутствия защиты от атак повтора

Для устранения заданной уязвимости необходимо включить проверку уникальности нонса в протокол и исключить из трассировок случаи, когда нонсы совпадают. Для этого можно использовать следующее ограничение:

Рисунок 3 – Вектор атаки на лемму `distinct_nonces`

restriction Unique:

"All x #i #j. UniqueNonce(x) @ #i & UniqueNonce(x) @ #j ==> #i = #j"

Для применения данного ограничения необходимо включить факт действия UniqueNonce(na) в правило B_1_receive. Описание исправленного протокола в нотации Tamarin представлено ниже:

theory auth1

begin

builtins: signing

rule Register_pk:

[Fr(~ltkX)]
-->[!Ltk(\$X, ~ltkX)
, !Pk(\$X, pk(~ltkX))
, Out(pk(~ltkX))]

rule Reveal_ltk:

[!Ltk(\$X, ltkX)]
--[Reveal(\$X)]->
[Out(ltkX)]

rule A_1_send:

let m1 = <\$A, ~na>
sign_a = sign(m1, ltkA)
in
[Fr(~na)
, !Ltk(\$A, ltkA)]
--[Send(\$A, ~na)]->
[St_A_1(\$A, ltkA, ~na)
, Out(<m1, sign_a>)]

rule B_1_receive:

let m1 = <\$A, na>
in

```

[ !Pk($A, pkA)
, In(<m1, sign_a_check>)]
--[ Recv($B, na)
, Eq(verify(sign_a_check,m1,pkA),true)
, Authentic($A,na), Honest($B), Honest($A)
, UniqueNonce(na)
, CheckNonce(na)]->
[ St_B_1($B, pkA, $A, <$A, na>)]

```

restriction Equality:

"All x y #i. Eq(x, y) @i ==> x=y"

restriction Unique:

"All x #i #j. UniqueNonce(x) @ #i & UniqueNonce(x) @ #j ==> #i = #j"

lemma executable:

exists-trace

"Ex A B m #i #j. Send(A,m)@i & Recv(B,m) @j"

lemma message_authentication:

"All b m #i. Authentic(b,m) @i
==> (Ex #j. Send(b,m) @j & j<i)
| (Ex B #r. Reveal(B)@r & Honest(B) @i & r < i)"

lemma distinct_nonces:

"All n #i #j. CheckNonce(n)@i & CheckNonce(n)@j ==> #i=#j"

end

Результат проверки исправленного протокола с использованием Tamarin prover показан на рисунке 4.

Как видно из рисунка 4, лемма окрашена в зеленый цвет, значит, Tamarin prover не нашел вектора атаки на исправленный протокол. Следовательно, уязвимость закрыта.

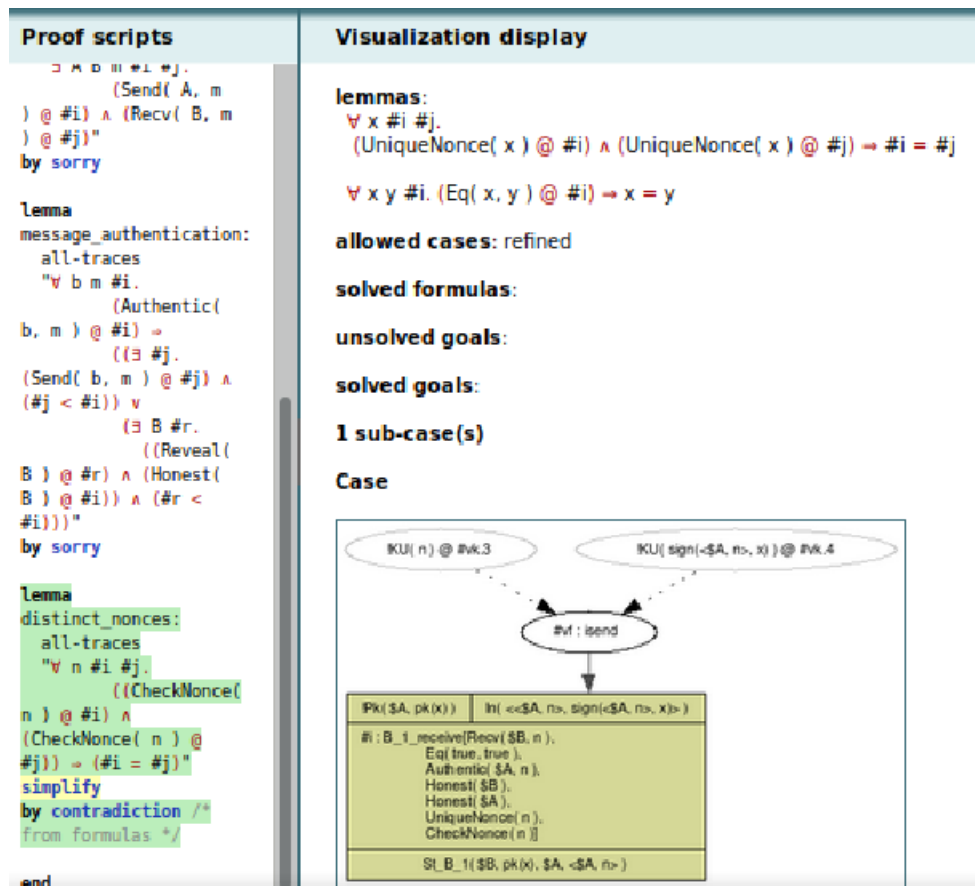


Рисунок 4 – Результат проверки исправленного протокола односторонней аутентификации на уязвимость отсутствия защиты от атак повтора

3.2 Разработка протокола взаимной аутентификации со внесением уязвимостей в виде отсутствия защиты от атак повтора и отсутствия контроля целостности потока сообщений

Был разработан протокол взаимной аутентификации, содержащий две уязвимости: отсутствие проверки уникальности нонса, как и в параграфе 3.1, а также отсутствие проверки различия порядковых номеров сообщений.

Целью данного протокола является передача информации (нонсов) между участниками A и B с обеспечением следующих свойств:

- идентичности A и аутентичности цифровой подписи A ;
- идентичности B и аутентичности цифровой подписи B .

Предполагается, что участники A и B имеют безопасную схему цифровой подписи. A имеет ключ подписи SK_A , B имеет соответствующий ключ проверки подписи PK_A , при этом B имеет свой ключ подписи SK_B и A имеет соответствующий ключ проверки подписи PK_B . Оба участника имеют ключевые запоминающие устройства $KЗУ_A$ и $KЗУ_B$. Оба участника имеют датчики случайных чисел $ДСЧ$, которые используются для генерации случайных чисел – нонсов N_A и N_B . Также оба участника имеют счетчики $Сч$.

Формальное описание данного протокола:

$$1. A: KЗУ_A \rightarrow SK_A$$

$$ДСЧ \rightarrow N_A$$

$$Сч \rightarrow X_A$$

$$S_{AB} = SK_A(A||N_A||X_A)$$

$$[A||N_A||X_A||S_{AB}] \Rightarrow B$$

2. B:

$$[\tilde{A}||\tilde{N}_A||\tilde{X}_A||\tilde{S}_{AB}]$$

$$KЗУ_B \rightarrow PK_{\tilde{A}}$$

$$PK_{\tilde{A}}(\tilde{A}||\tilde{N}_A||\tilde{X}_A, \tilde{S}_{AB}) \stackrel{?}{=} 1$$

$$\tilde{A} = A \quad \tilde{N}_A = N_A \quad \tilde{X}_A = X_A$$

$$KЗУ_B \rightarrow SK_B$$

$$ДСЧ \rightarrow N_B$$

$$Сч \rightarrow X_B$$

$$S_{BA} = SK_B(B||N_B||S_{AB}||X_B)$$

$$[B||N_B||S_{AB}||X_B||S_{BA}] \Rightarrow A$$

3. A:

$$[\tilde{B}||\tilde{N}_B||\tilde{S}_{AB}||\tilde{X}_B||\tilde{S}_{BA}]$$

$$\tilde{S}_{AB} \stackrel{?}{=} S_{AB}$$

$$KЗУ_A \rightarrow PK_{\tilde{B}}$$

$$PK_{\tilde{B}}(\tilde{B}||\tilde{N}_B||\tilde{S}_{AB}||\tilde{X}_B, \tilde{S}_{BA}) \stackrel{?}{=} 1$$

$$\widetilde{B} = B \quad \widetilde{N_B} = N_B \quad \widetilde{X_B} = X_B$$

Ниже приведено словесное описание протокола.

Участник A :

- а) генерирует случайное число N_A с использованием датчика случайных чисел ДСЧ;
- б) получает от счетчика $Cч$ номер сообщения X_A ;
- в) формирует блок $[A||N_A||X_A]$ и вычисляет цифровую подпись S_{AB} на своем ключе SK_A ;
- г) формирует сообщение $[A||N_A||X_A||S_{AB}]$ и посылает участнику B по открытому каналу связи.

Участник B :

- а) формирует блок $[\widetilde{A}||\widetilde{N_A}||\widetilde{X_A}]$ из принятых значений $\widetilde{A}$, $\widetilde{N_A}$, $\widetilde{X_A}$ и проверяет принятую цифровую подпись $\widetilde{S_{AB}}$ с использованием ключа проверки подписи PK_A в соответствии с принятым адресом отправителя A ;
- б) при отрицательном результате проверки сообщение не принимается; при положительном результате делается вывод о том, что $\widetilde{A}$ равно A , $\widetilde{N_A}$ равно N_A , $\widetilde{X_A}$ равно X_A , сообщение принимается, доказывающий $\widetilde{A}$ считается опознанным;
- в) с помощью датчика случайных чисел ДСЧ генерирует случайное число N_B ;
- г) получает от счетчика $Cч$ номер сообщения X_B ;
- д) формирует блок $[B||N_B||X_B]$ и вычисляет цифровую подпись S_{BA} на своем ключе SK_B ;
- е) формирует сообщение $[B||N_B||S_{AB}||X_B||S_{BA}]$ и посылает участнику A по открытому каналу связи.

Участник A :

- а) проверяет соответствие цифровой подписи $\widetilde{S_{AB}}$ из сообщения и собственной цифровой подписи S_{AB} ;

б) при отрицательном результате проверки сообщение не принимается; при положительном результате B формирует блок $[\widetilde{B}||\widetilde{N}_B||\widetilde{X}_B]$ из принятых значений $\widetilde{B}$, $\widetilde{N}_B$, $\widetilde{X}_B$ и проверяет принятую цифровую подпись $\widetilde{S}_{BA}$ с использованием ключа проверки подписи PK_B в соответствии с принятым адресом отправителя $\widetilde{B}$;

в) при отрицательном результате проверки сообщение не принимается; при положительном результате делается вывод о том, что $\widetilde{B}$ равно B , $\widetilde{N}_B$ равно N_B , $\widetilde{X}_B$ равно X_B , сообщение принимается, доказывающий B считается опознанным.

Описание протокола в нотации Tamarin:

theory auth2

begin

builtins: signing

rule Register_pk:

[Fr($\sim$ ltkX)] -->
 [!Ltk($\$X$, $\sim$ ltkX),
 !Pk($\$X$, pk($\sim$ ltkX)),
 Out(pk($\sim$ ltkX))]

rule Reveal_ltk:

[!Ltk($\$X$, ltkX)] –
 [Reveal($\$X$)]->
 [Out(ltkX)]

rule A_1_send:

let
 $m1 = \langle \$A, \sim na, '1' \rangle$
 $sign_a = sign(m1, ltkA)$
 in
 [Fr($\sim na$)
 , !Ltk($\$A$, ltkA)]

```

, !Pk($B, pkB)]
--[ Send1($A, ~na)
, MessageNum('1')]->
[ St_A_1($A, ltkA, pkB, $B, ~na, sign_a, '1')
, Out(<m1, sign_a>)]

```

rule B_1_receive:

```

let
  m1 = <$A, na, '1'>
  m2 = <$A, $B, ~nb, sign_a_check, '2'>
  sign_b = sign(m2, ltkB)
in
  [ !Ltk($B, ltkB)
  , !Pk($A, pkA)
  , In(<m1, sign_a_check>)
  , Fr(~nb)]
  --[ Recv1($B, na)
  , Eq(verify(sign_a_check,m1,pkA),true)
  , Authentic($A,na), Honest($B), Honest($A)
  , Send2($B, sign_a_check)]->
  [ St_B_1($B, ltkB, pkA, $A, na, ~nb)
  , Out(<m2, sign_b>)]

```

rule A_2_receive:

```

let m2 = <$A, $B, nb, sign_a_check, passnum2>
in
  [St_A_1($A, ltkA, pkB, $B, na, sign_a, passnum1)
  , In(<m2, sign_b>)]
  --[ Recv2($A, sign_a_check)
  , Eq(verify(sign_b,m2,pkB),true)
  , Authentic($B,sign_a_check), Honest($B), Honest($A)

```

, Eq(sign_a,sign_a_check)]->
 [St_A_2(\$A, ltkA, pkB, \$B, na, nb)]

restriction Equality:

"All x y #i. Eq(x, y) @i ==> x=y"

lemma executable:

exists-trace

"Ex A B m1 m2 #i #j #k #n. Send1(A,m1)@i & Recv1(B,m1) @j
 & Send2(A,m2)@k & Recv2(B,m2) @n"

lemma message_authentication:

"All b m #i. Authentic(b,m) @i
 ==> (Ex #j. Send1(b,m) @j & j<i)
 | (Ex #q. Send2(b,m) @q & q<i)
 | (Ex B #r. Reveal(B)@r & Honest(B) @i & r < i)"

end

Действия по идентификации уязвимости в виде отсутствия проверки нонса и ее закрытию идентичны описанным в параграфе 3.1. Однако в данном протоколе необходимо также добавить факты действия UniqueNonce(nb) и CheckNonce(nb) в правило A_2_receive.

Для идентификации уязвимости в виде отсутствия проверки различия порядковых номеров сообщений необходимо внести в протокол следующие изменения:

— добавить к описанию лемму, проверяющую различие порядковых номеров сообщений протокола. Пример подобной леммы:

lemma flow_integrity:

"All n1 n2 #i #j. MessageNum(n1) @i & CheckPassNum(n2) @j==>
 not n1 = n2";

— добавить в правило A_2_receive факт действия CheckPassNum(passnum2).

В ходе проверки указанной леммы Tamarin prover окрашивает лемму в красный цвет, что показано на рисунке 5, и строит вектор атаки, изображенный на рисунке 6.

```
lemma flow_integrity:
  all-traces
  "∀ n1 n2 #i #j.
    ((MessageNum( n1 ) @ #i) ∧
    (CheckPassNum( n2 ) @ #j)) →
    (¬(n1 = n2))"
  simplify
  solve( !Ltk( $A, ltkA ) @ #i )
  case Register_pk
  solve( !Pk( $B, pkB ) @ #i )
  case Register_pk
  solve( St_A_1( $A.1, ltkA, pk(x), $B.1,
na.1, sign_a, passnum1
  ) @ #j )
  case A_1_send
  solve( !KU( sign(<$A.1, $B.1, nb,
sign(<$A.1, ~na.1,
'1'>, ~ltkX.2), '1'>,
~ltkX.3)
  ) @ #vk.10 )
  case c_sign
  solve( !KU( ~ltkX.3 ) @ #vk.11 )
  case Reveal_ltk
  solve( !KU( sign(<$A.1, ~na.1,
'1'>, ~ltkX.2)
  ) @ #vk.10 )
  case A_1_send
  SOLVED // trace found
```

Рисунок 5 – Идентификация отсутствия контроля целостности потока сообщений

Для устранения заданной уязвимости необходимо включить в протокол проверку различия порядковых номеров сообщений и исключить из трассировок случаи, когда указанные номера совпадают. Для этого можно использовать следующее ограничение:

restriction Inequality:

"All x y #i. Neq(x, y) @i ==> not x=y"

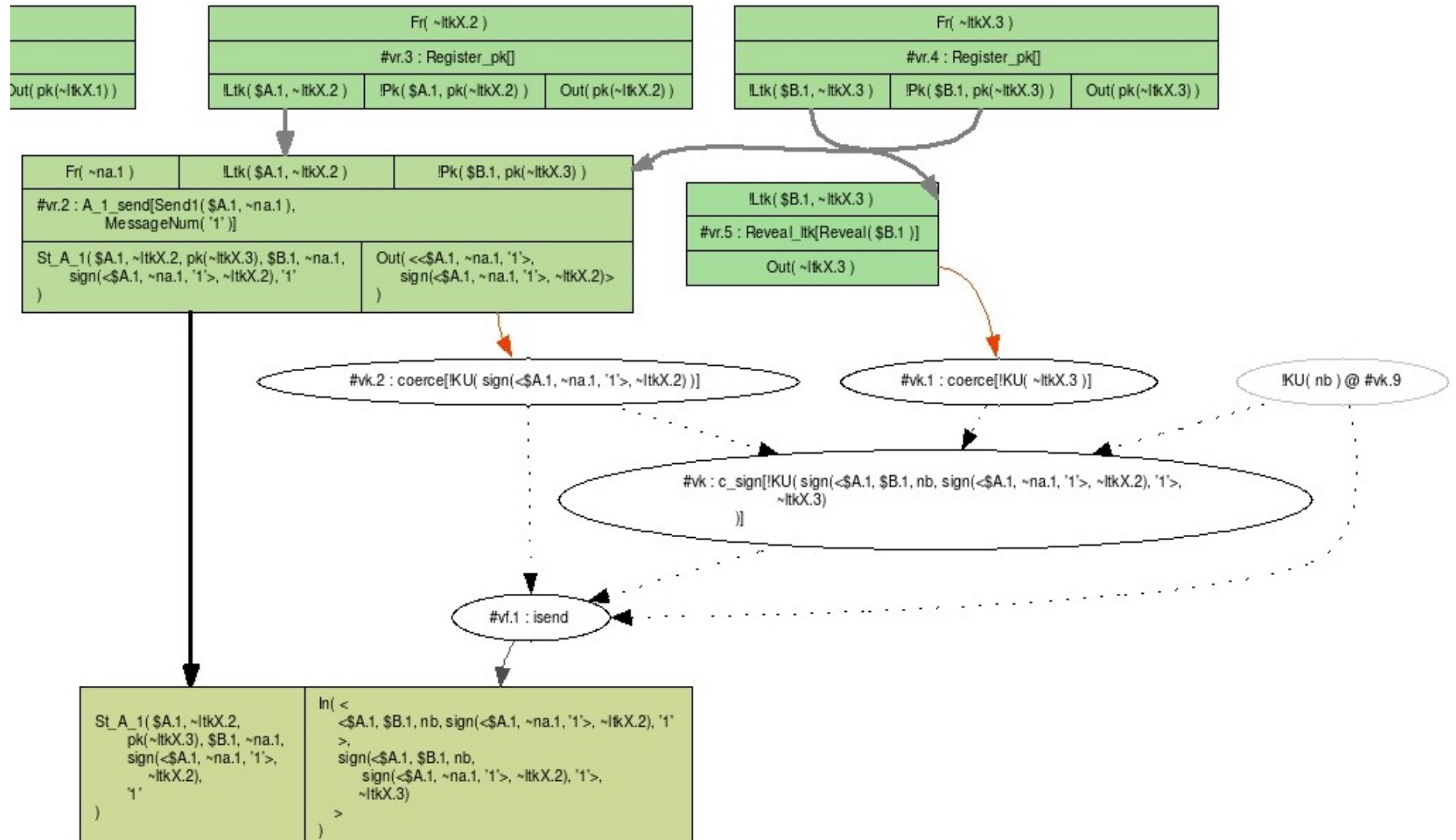
Для применения данного ограничения необходимо включить факт действия Neq(passnum1, passnum2) в правило A_2_receive.

Итоговое описание протокола в нотации Tamarin представлено ниже:

theory auth2

begin

builtins: signing

Рисунок 6 – Вектор атаки на лемму `flow_integrity`

rule Register_pk:

```
[Fr(~ltkX)]
-->[!Ltk($X, ~ltkX),
!Pk($X, pk(~ltkX)),
Out(pk(~ltkX))]
```

rule Reveal_ltk:

```
[!Ltk($X, ltkX)]
--[Reveal($X)]->
[Out(ltkX)]
```

rule A_1_send:

```
let   m1 = <$A, ~na, '1'>
      sign_a = sign(m1, ltkA)
in
      [Fr(~na)
      , !Ltk($A, ltkA)
      , !Pk($B, pkB)]
      --[ Send1($A, ~na)
      , MessageNum('1')]->
      [ St_A_1($A, ltkA, pkB, $B, ~na, sign_a, '1')
      , Out(<m1, sign_a>)]
```

rule B_1_receive:

```
let m1 = <$A, na, '1'>
      m2 = <$B, ~nb, sign_a_check, '2'>
      sign_b = sign(m2, ltkB)
in
      [ !Ltk($B, ltkB)
      , !Pk($A, pkA)
      , In(<m1, sign_a_check>)
      , Fr(~nb)]
```

```

--[ Recv1($B, na)
, Eq(verify(sign_a_check,m1,pkA),true)
, Authentic($A,na), Honest($B), Honest($A)
, Send2($B, sign_a_check)
, UniqueNonce(na)
, CheckNonce(na)]->
[ St_B_1($B, ltkB, pkA, $A, na, ~nb)
, Out(<m2,sign_b>)]

```

rule A_2_receive:

```

let m2 = <$A, $B, nb, sign_a_check, passnum2>
in
[St_A_1($A, ltkA, pkB, $B, na, sign_a, passnum1)
, In(<m2, sign_b>)]
--[ Recv2($A, sign_a_check)
, Eq(verify(sign_b,m2,pkB),true)
, Authentic($B,sign_a_check), Honest($B), Honest($A)
, Eq(sign_a,sign_a_check)
, Neq(passnum1, passnum2)
, CheckPassNum(passnum2)
, UniqueNonce(nb)
, CheckNonce(nb)]->
[ St_A_2($A, ltkA, pkB, $B, na, nb)]

```

restriction Unique:

"All x #i #j. UniqueNonce(x) @ #i & UniqueNonce(x) @ #j ==> #i = #j"

restriction Equality:

"All x y #i. Eq(x, y) @i ==> x=y"

restriction Inequality:

"All x y #i. Neq(x, y) @i ==> not x=y"

lemma executable:

exists-trace

"Ex A B m1 m2 #i #j #k #n. Send1(A,m1)@i & Recv1(B,m1) @j
& Send2(A,m2)@k & Recv2(B,m2) @n"

lemma message_authentication:

"All b m #i. Authentic(b,m) @i
==> (Ex #j. Send1(b,m) @j & j<i)
| (Ex #q. Send2(b,m) @q & q<i)
| (Ex B #r. Reveal(B)@r & Honest(B) @i & r < i)"

lemma distinct_nonces:

"All n #i #j. CheckNonce(n)@i & CheckNonce(n)@j ==> #i=#j"

lemma flow_integrity:

"All n1 n2 #i #j. MessageNum(n1) @i & CheckPassNum(n2) @j==>
not n1 = n2"

end

Результаты проверки итогового протокола с использованием Tamarin prover показаны на рисунках 7 и 8.

```
lemma distinct_nonces:
  all-traces
  "∀ n #i #j.
    ((CheckNonce( n ) @ #i) ∧
    (CheckNonce( n ) @ #j)) → (#i = #j)"
  simplify
  solve( (#i < #j) | (#j < #i) )
  case case_1
  solve( CheckNonce( n ) @ #i )
  case A_2_receive
  solve( St_A_1( $A, ltkA, pk(x), $B, na,
  sign_a, passnum1 ) ▶ #i )
  case A_1_send
  solve( CheckNonce( n ) @ #j )
  case A_2_receive
  by contradiction /* cyclic */
qed
qed
next
```

Рисунок 7 – Результат проверки исправленного протокола взаимной аутентификации на уязвимость отсутствия защиты от атак повтора

```

lemma flow_integrity:
  all-traces
  "v n1 n2 #i #j.
    ((MessageNum( n1 ) @ #i) ∧
    (CheckPassNum( n2 ) @ #j)) →
    (¬(n1 = n2))"
  simplify
  solve( !Ltk( $A, ltkA ) ▷1 #i )
  case Register_pk
  solve( !Pk( $B, pkB ) ▷2 #i )
  case Register_pk
  solve( St_A_1( $A.1, ltkA, pk(x), $B.1,
na.1, sign_a, passnum1
) ▷3 #j )
  case A_1_send
  by contradiction /* from formulas */
qed
qed
qed

```

Рисунок 8 – Результат проверки исправленного протокола взаимной аутентификации на уязвимость отсутствия контроля целостности потока сообщений

Как видно из рисунков 7 и 8. леммы окрашены в зеленый цвет, значит, Tamarin prover не нашел векторов атаки на исправленный протокол. Следовательно, уязвимости закрыты.

3.3 Разработка протокола передачи данных с обеспечением свойства конфиденциальности со внесением уязвимостей в виде отсутствия защиты от атак повтора и отсутствия контроля целостности сообщений

Был разработан протокол передачи данных с обеспечением свойства конфиденциальности, содержащий две уязвимости: отсутствие проверки уникальности нонса, как и в параграфе 3.1, а также отсутствие контроля целостности сообщений.

Целью данного протокола является доставка сообщения M от A к B с обеспечением свойства конфиденциальности.

Предполагается, что участники A и B имеют преобразования E и D шифрования и расшифрования соответственно. A знает публичный ключ шифрования B PK_B , B имеет соответствующий секретный ключ SK_B . Оба

участника имеют ключевые запоминающие устройства $KЗУ_A$ и $KЗУ_B$. A имеет датчик случайных чисел $ДСЧ$, который используется для генерации случайных чисел – нонсов N_A .

Формальное описание данного протокола:

$$1. A: KЗУ_A \rightarrow PK_B$$

$$ДСЧ \rightarrow N_A$$

$$C_{AB} = E_{PK_B}(B||A||N_A||M)$$

$$[B||A||N_A||C_{AB}] \Rightarrow B$$

$$2. B:$$

$$[B||A||N_A||\tilde{C}_{AB}]$$

$$KЗУ_B \rightarrow SK_B$$

$$D_{SK_B}(\tilde{C}_{AB}) = [B||A||N_A||M]$$

Ниже приведено словесное описание протокола.

Участник A :

- а) извлекает ключ PK_B из своего $KЗУ_A$;
- б) с помощью датчика случайных чисел $ДСЧ$ генерирует случайное число N_A ;
- в) формирует блок $[B||A||N_A||M]$ и зашифровывает его на ключе PK_B ;
- г) формирует сообщение $[B||A||N_A||C_{AB}]$ и посылает участнику B по открытому каналу связи.

Участник B :

- а) извлекает ключ SK_B из своего $KЗУ_B$;
- б) расшифровывает $\tilde{C}_{AB}$ на ключе SK_B и получает $[B||A||N_A||M]$.

Сообщение M принимается.

Описание протокола в нотации Tamarin:

```
theory conf_protocol
```

```
begin
```

builtins: asymmetric-encryption , hashing

rule Register_pk:

```
[Fr(~ltkB)]
-->[!Ltk($B, ~ltkB),
!Pk($B, pk(~ltkB)),
Out(pk(~ltkB))]
```

rule Reveal_ltk:

```
[!Ltk($B, ltkB)]
--[Reveal($B)]->
[Out(ltkB)]
```

rule A_1_send:

```
let
    ma = <$B, $A, ~Na, ~x>
    Cab = aenc(ma, pkB)
in
    [Fr(~Na),
    Fr(~x),
    !Pk($B, pkB)]
    --[Send($A, <$B, $A, ~Na, ~x>),
    Secret(~x)]->
    [Out(<$B, $A, ~Na, Cab>),
    St_A_1($A, pkB, $B, ~x)]
```

rule B_1_receive:

```
let
    ma = <$B, A, Na, x>
    Cab = aenc(ma, pk(ltkB))
in
    [!Ltk($B, ltkB),
    In(<$B, A, Na, Cab>)]
```

```

--[Recv($B, <$B, A, Na, x>)]->
[St_B_1($B, ltkB, x)]
lemma executable:
  exists-trace
    "Ex A B m #i #j. Send(A,m)@i & Recv(B,m) @j"
lemma secrecy:
  "All x #i. Secret(x) @i & not (Ex B #r. Reveal(B) @ r)
  ==> not (Ex #j. K(x)@j)"
end

Действия по идентификации уязвимости в виде отсутствия проверки
нонса и ее закрытию идентичны описанным в параграфе 3.1.

Для идентификации уязвимости в виде отсутствия контроля целостности
сообщений необходимо внести в протокол следующие изменения:
— добавить к описанию лемму, сравнивающую результат
расшифрования служебных данных с полученными из сообщения значениями
A, B и NA. Пример подобной леммы:
lemma integrity:
  "All h1 h2 #i. CheckIntegrity(h1, h2)@i ==> h1 = h2";
— добавить в правило B_1_receive обозначение mb = <$B, A, Na> и
факт действия CheckIntegrity(ma,mb).

В ходе проверки указанной леммы Tamarin prover окрашивает ее в
красный цвет, что показано на рисунке 9, и строит вектор атаки, изображенный
на рисунке 10.

Для устранения заданной уязвимости необходимо включить в протокол
хэширование данных сообщений.

Также исходный протокол содержит в сообщении одни и те же данные в
открытом и зашифрованном виде, что является нежелательным. Поэтому
целесообразно зашифровывать только сообщение M и значение хэш-функции.

```

```

lemma integrity:
  all-traces
  "∀ h1 h2 #i. (CheckIntegrity( h1, h2 ) @
  #i) → (h1 = h2)"
  simplify
  solve( !Ltk( $B, ltkB ) ▶ #i )
  case Register_pk
  solve( !KU( aenc(<h1, x>, pk(~ltkB)) ) @
  #vk.6 )
  case A_1_send
  SOLVED // trace found
  qed
  qed

```

Рисунок 9 – Идентификация отсутствия контроля целостности сообщений

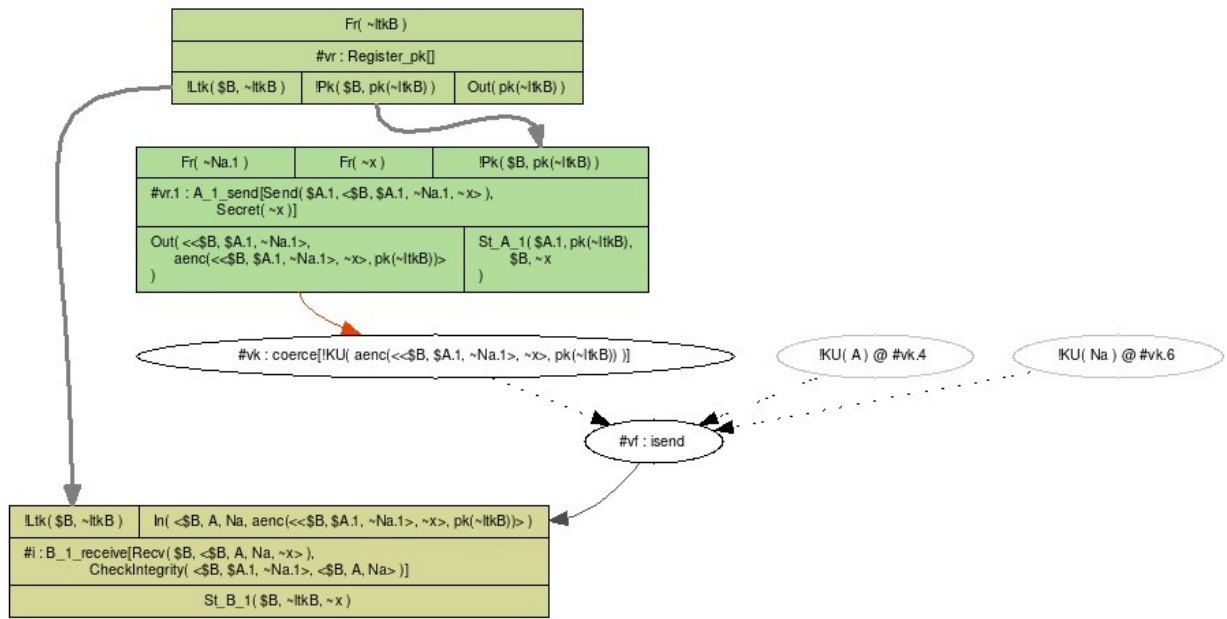


Рисунок 10 – Вектор атаки на лемму integrity

Описание исправленного протокола в нотации Tamarin представлено

ниже:

theory conf_protocol

begin

builtins: asymmetric-encryption , hashing

rule Register_pk:

[Fr(~ltkB)]

-->[!Ltk(\$B, ~ltkB),

!Pk(\$B, pk(~ltkB)),

Out(pk(~ltkB))]

rule Reveal_ltk:

```

[!Ltk($B, ltkB)]
--[Reveal($B)]->
[Out(ltkB)]
rule A_1_send:
  let
    ha = h(<$B, $A, ~Na, ~x>)
    Cab = aenc(<~x,ha>, pkB)
  in
    [Fr(~Na),
     Fr(~x),
     !Pk($B, pkB)]
    --[Send($A, <$B, $A, ~Na, ~x>),
     Secret(~x)]->
    [Out(<$B, $A, ~Na, Cab>),
     St_A_1($A, pkB, $B, ~x)]
rule B_1_receive:
  let
    Cab = aenc(<x,ha>, pk(ltkB))
    hb = h(<$B, A, Na, x>)
  in
    [!Ltk($B, ltkB),
     In(<$B, A, Na, Cab>)]
    --[Recv($B, <$B, A, Na, x>),
     CheckIntegrity(hb, ha),
     Eq(hb, ha),
     UniqueNonce(Na),
     CheckNonce(Na),
     Honest($B)]->
    [St_B_1($B, ltkB, A, x)]

```

restriction Equality:

"All x y #i. Eq(x, y) @i ==> x=y"

restriction Unique:

"All x #i #j. UniqueNonce(x) @ #i & UniqueNonce(x) @ #j ==> #i = #j"

lemma executable:

exists-trace

"Ex A B m #i #j. Send(A,m)@i & Recv(B,m) @j"

lemma integrity:

"All h1 h2 #i. CheckIntegrity(h1, h2)@i ==> h1 = h2"

lemma secrecy:

"All x #i. Secret(x) @i & not (Ex B #r. Reveal(B) @ r)
==> not (Ex #j. K(x)@j)"

lemma distinct_nonces:

"All n #i #j. CheckNonce(n)@i & CheckNonce(n)@j ==> #i=#j"

end

Как видно из описания итогового протокола, лемма *integrity* стала проверять целостность всего сообщения, а не только служебных данных, что позволяет проверить также целостность сообщения *M*. Чтобы указать на то, что в случае несовпадения хэш-функций *ha* и *hb* протокол разрывается, было добавлено ограничение *Equality*, а также факт действия *Eq(hb, ha)*.

Результаты проверки итогового протокола с использованием Tamarin prover показаны на рисунках 11 и 12.

```
lemma distinct_nonces:
  all-traces
  "V n #i #j.
    ((CheckNonce( n ) @
    #i) ^ (CheckNonce( n ) @
    #j)) -> (#i = #j)"
  simplify
  by contradiction /* from
  formulas */
```

Рисунок 11 – Результат проверки исправленного протокола передачи сообщений с обеспечением конфиденциальности на уязвимость отсутствия защиты от атак повтора

```

lemma integrity:
  all-traces
  "∀ h1 h2 #i.
  (CheckIntegrity( h1, h2 ) @
  #i) → (h1 = h2)"
  simplify
  by contradiction /* from
  formulas */

```

Рисунок 12 – Результат проверки исправленного протокола передачи сообщений с обеспечением конфиденциальности на уязвимость отсутствия контроля целостности

Как видно из рисунков 11 и 12, леммы окрашены в зеленый цвет. Это значит, что Tamarin prover не нашел векторов атаки на исправленный протокол. Следовательно, уязвимости закрыты.

3.4 Разработка протокола передачи данных с обеспечением свойства неотказуемости со внесением уязвимостей в виде отсутствия защиты от атак повтора, отсутствия контроля целостности потока сообщений, вычисления цифровой подписи не от всех полей сообщения, а также отсутствия аутентификации одного из участников протокола

Был разработан протокол передачи данных с обеспечением свойства неотказуемости, содержащий четыре уязвимости: отсутствие проверки уникальности нонса и проверки различия порядковых номеров сообщений, как и в параграфах 3.1 и 3.2, а также вычисление цифровой подписи не от всех полей сообщения и отсутствие аутентификации одного из участников протокола.

Целью данного протокола является передача сообщения M от A к B с обеспечением свойства неотказуемости.

Предполагается, что участники A и B имеют безопасные схемы вычисления цифровой подписи и хэш-функции. A имеет ключ подписи SK_A , B имеет соответствующий ключ проверки подписи PK_A , при этом B имеет свой ключ подписи SK_B и A имеет соответствующий ключ проверки подписи PK_B .

Оба участника имеют ключевые запоминающие устройства $KЗУ_A$ и $KЗУ_B$. Оба участника имеют датчики случайных чисел $ДСЧ$, которые используются для генерации случайных чисел – нонсов N_A и N_B . Также оба участника имеют счетчики $Сч$.

Формальное описание данного протокола:

$$1. A: KЗУ_A \rightarrow SK_A$$

$$ДСЧ \rightarrow N_A$$

$$Сч \rightarrow X_A$$

$$S_{AB} = SK_A(M)$$

$$H_{AB} = h(B||A||N_A||M||X_A||S_{AB})$$

$$[B||A||N_A||M||X_A||S_{AB}||H_{AB}] \Rightarrow B$$

$$2. B:$$

$$[B||\widetilde{A}||\widetilde{N_A}||\widetilde{M}||\widetilde{X_A}||\widetilde{S_{AB}}||\widetilde{H_{AB}}]$$

$$KЗУ_B \rightarrow PK_{\widetilde{A}}$$

$$PK_A(\widetilde{M}, \widetilde{S_{AB}}) \stackrel{?}{=} I$$

$$\widetilde{A} = A \quad \widetilde{N_A} = N_A \quad \widetilde{X_A} = X_A \quad \widetilde{M} = M \quad \widetilde{H_{AB}} = H_{AB}$$

$$ДСЧ \rightarrow N_B$$

$$Сч \rightarrow X_B$$

$$[A||B||N_B||H_{AB}||X_B] \Rightarrow A$$

$$3. A:$$

$$[A||\widetilde{B}||\widetilde{N_B}||\widetilde{H_{BA}}||\widetilde{X_B}]$$

$$\widetilde{H_{BA}} \stackrel{?}{=} H_{AB}$$

$$\widetilde{B} = B \quad \widetilde{N_B} = N_B \quad \widetilde{X_B} = X_B$$

Ниже приведено словесное описание протокола.

Участник A :

а) с помощью датчика случайных чисел $ДСЧ$ генерирует случайное число N_A ;

- б) получает со счетчика $Cч$ номер сообщения X_A ;
- в) формирует сообщение $[B||A||N_A||M||X_A||S_{AB}||H_{AB}]$ и посылает участнику B по открытому каналу связи.

Участник B :

- а) проверяет принятую цифровую подпись $\tilde{S}_{AB}$ от сообщения M с использованием ключа проверки подписи PK_A в соответствии с принятым адресом отправителя $\tilde{A}$;
- б) при отрицательном результате проверки сообщение не принимается; при положительном результате делается вывод о том, что $\tilde{A}$ равно A , $\tilde{N}_A$ равно N_A , $\tilde{X}_A$ равно X_A , $\tilde{M}$ равно M , $\tilde{H}_{AB}$ равно H_{AB} , сообщение принимается;
- в) с помощью датчика случайных чисел $ДСЧ$ генерирует случайное число N_B ;
- г) получает со счетчика $Cч$ номер сообщения X_B ;
- д) формирует сообщение $[A||B||N_B||H_{AB}||X_B]$ и посылает участнику A по открытому каналу связи.

Участник A :

- а) сравнивает полученное в сообщении значение $\tilde{H}_{BA}$ и сохраненное исходное значение H_{AB} ;
- б) при отрицательном результате проверки сообщение не принимается; при положительном результате делается вывод о том, что $\tilde{B}$ равно B , $\tilde{N}_B$ равно N_B , $\tilde{X}_B$ равно X_B , сообщение принимается.

Описание протокола в нотации Tamarin:

```
theory non_rep_protocol
begin
  builtins: signing, hashing
  rule Register_pk:
    [Fr(~ltkX)]
    ->[!Ltk($X, ~ltkX),
    !Pk($X, pk(~ltkX)),
```

```

    Out(pk(~ltkX))
rule Reveal_ltk:
    [!Ltk($X, ltkX)]
    --[Reveal($X)]->
    [Out(ltkX)]
rule A_1_send:
    let
        m1 = <$B, $A, ~na, ~m, '1'>
        sign_a = sign(~m, ltkA)
    in
        [Fr(~na)
        ,Fr(~m)
        , !Ltk($A, ltkA)
        , !Pk($B, pkB)]
        --[ Send1($A, ~m, ~na)
        , MessageNum('1')]->
        [ St_A_1($A, ltkA, pkB, $B, ~m, h(<m1, sign_a>), '1')
        , Out(<m1, sign_a, h(<m1, sign_a>)>)]
rule B_1_receive:
    let
        m1 = <$B, $A, na, m, '1'>
        m2 = <$A, $B, ~nb, hash1, '2'>
    in
        [ !Ltk($B, ltkB)
        , !Pk($A, pkA)
        , In(<m1, sign_a_check, hash1>)
        , Fr(~nb)]
        --[ Recv1($B, m, na)
        , Eq(verify(sign_a_check,m,pkA),true)]

```

```

, Authentic($A,m,na), Honest($B), Honest($A)
, Send2($B, hash1, ~nb)]->
[ St_B_1($B, ltkB, pkA, $A, m)
, Out(<m2>)]

```

rule A_2_receive:

```

let m2 = <$A, $B, nb, hash2, passnum2>
in
[St_A_1($A, ltkA, pkB, $B, m, hash1, passnum1)
, In(<m2>)]
--[ Recv2($A, hash2, nb)
, Authentic($B,hash2,nb), Honest($B), Honest($A)
, Eq(hash2, hash1)]->
[ St_A_2($A, ltkA, pkB, $B, m)]

```

restriction Equality:

"All x y #i. Eq(x, y) @i ==> x=y"

lemma executable:

```

exists-trace
"Ex A B m1 m2 n1 n2 #i #j #k #n.
Send1(A,m1,n1)@i & Recv1(B,m1,n1) @j
& Send2(A,m2,n2)@k & Recv2(B,m2,n2) @n"

```

lemma message_authentication:

```

"All b m n #i. Authentic(b,m,n) @i
==> (Ex #j. Send1(b,m,n) @j & j<i)
| (Ex B #r. Reveal(B)@r & Honest(B) @i & r < i)"

```

end

Действия по идентификации уязвимостей в виде отсутствия проверки нонса и проверки различия порядковых номеров сообщений, а также их закрытию идентичны описанным в параграфах 3.1 и 3.2.

Для идентификации уязвимости в виде вычисления цифровой подписи не от всех полей сообщения необходимо проверить существующую лемму `message_authentication` с помощью `Tamarin prover`. В ходе проверки леммы `Tamarin prover` окрашивает ее в красный цвет, что показано на рисунке 13, и строит вектор атаки, изображенный на рисунке 14.

```
lemma message_authentication:
  all-traces
  "∀ b m n #i.
    (Authentic( b, m, n ) @ #i) →
    ((∃ #j. (Send1( b, m, n ) @ #j) ∧
    (#j < #i)) ∨
    (∃ B #r.
      ((Reveal( B ) @ #r) ∧ (Honest( B
    ) @ #i)) ∧ (#r < #i)))"
  simplify
  solve( Authentic( b, m, n ) @ #i )
  case A_2_receive
  solve( St_A_1( $A, ltkA, pkB, $B, m, m.l,
  passnum1 ) @ #i )
  case A_1_send
  solve( !KU( h(<<$B, $A, ~na, ~m, '1'>,
    sign(~m, ~ltkX)>)
    ) @ #vk.7 )
  case A_1_send
  SOLVED // trace found
```

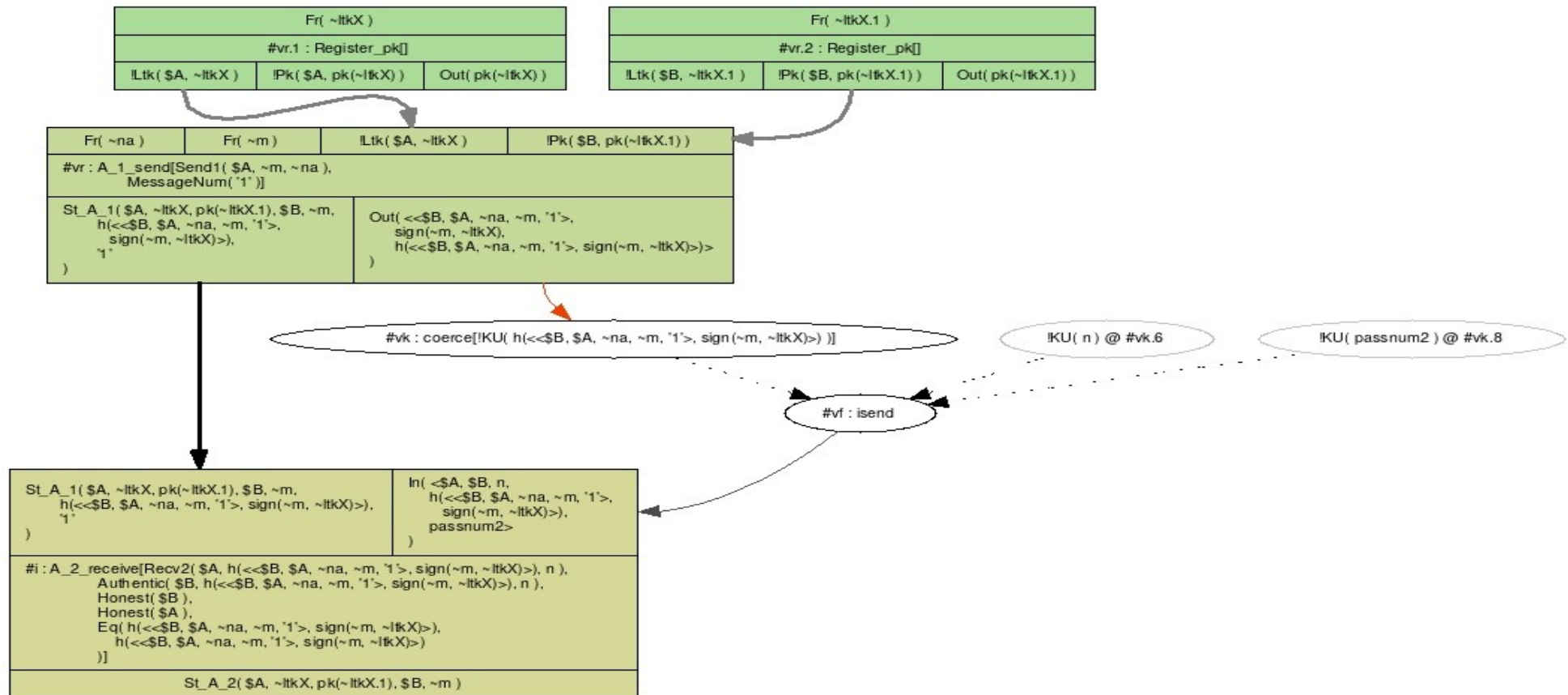
Рисунок 13 – Идентификация вычисления цифровой подписи не от всех полей сообщения

Для закрытия данной уязвимости необходимо вычислять цифровую подпись не только от сообщения M , но и от служебных данных.

Для идентификации уязвимости в виде отсутствия аутентификации одного из участников протокола необходимо добавить в лемму `message_authentication` строку, проверяющую аутентификацию участника B . Итоговая лемма может выглядеть следующим образом:

lemma message_authentication:

```
"All b m n #i. Authentic(b,m,n) @i
==> (Ex #j. Send1(b,m,n) @j & j<i)
| (Ex #q. Send2(b,m,n) @q & q<i)
| (Ex B #r. Reveal(B)@r & Honest(B) @i & r < i)"
```

Рисунок 14 – Вектор атаки на лемму `message_authentication`

В ходе проверки указанной леммы Tamarin prover строит вектор атаки, изображенный на рисунке 15.

Для устранения заданной уязвимости необходимо включить в протокол вычисление цифровой подписи участников В и ее проверку участником А.

Итоговое описание исправленного протокола в нотации Tamarin представлено ниже:

```
theory non_rep_protocol
begin
  builtins: signing, hashing
  rule Register_pk:
    [Fr(~ltkX)]
    --> [!Ltk($X, ~ltkX),
        !Pk($X, pk(~ltkX)),
        Out(pk(~ltkX))]
  rule Reveal_ltk:
    [!Ltk($X, ltkX)]
    --[Reveal($X)]->
    [Out(ltkX)]
  rule A_1_send:
    let
      m1 = <$B, $A, ~na, ~m, '1'>
      sign_a = sign(m1, ltkA)
    in
      [Fr(~na)
      ,Fr(~m)
      , !Ltk($A, ltkA)
      , !Pk($B, pkB)]
      --[ Send1($A, ~m, ~na)
      , MessageNum('1')]->
```

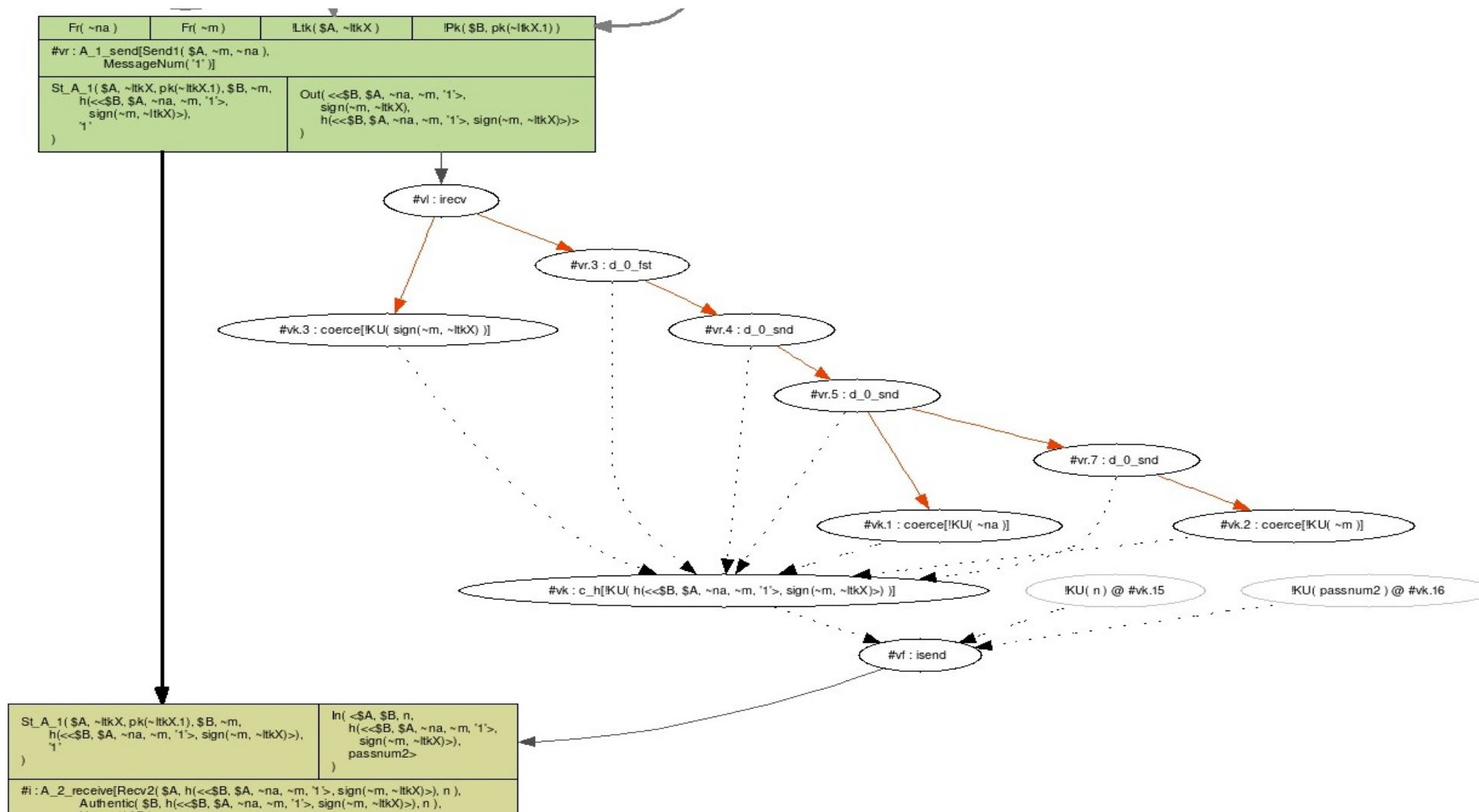


Рисунок 15 – Вектор атаки на дополненную лемму message_authentication

[St_A_1(\$A, ltkA, pkB, \$B, ~m, h(<m1, sign_a>), '1')
 , Out(<m1, sign_a, h(<m1, sign_a>)>)]

rule B_1_receive:

let m1 = <\$B, \$A, na, m, '1'>
 m2 = <\$A, \$B, ~nb, hash, '2'>
 sign_b = sign(m2, ltkB)
 in
 [!Ltk(\$B, ltkB)
 , !Pk(\$A, pkA)
 , In(<m1, sign_a_check, hash>)
 , Fr(~nb)]
 --[Recv1(\$B, m, na)
 , Eq(verify(sign_a_check,m1,pkA),true)
 , Authentic(\$A,m,na), Honest(\$B), Honest(\$A)
 , Send2(\$B, hash, ~nb)
 , CheckNonce(na)
 , Eq(hash, h(<m1, sign_a_check>))
 , UniqueNonce(na)]->
 [St_B_1(\$B, ltkB, pkA, \$A, m)
 , Out(<m2,sign_b>)]

rule A_2_receive:

let m2 = <\$A, \$B, nb, hash2, passnum2>
 in
 [St_A_1(\$A, ltkA, pkB, \$B, m, hash1, passnum1)
 , In(<m2,sign_b>)]
 --[Recv2(\$A, hash2, nb)
 , Eq(verify(sign_b,m2,pkB),true)
 , Authentic(\$B,hash2,nb), Honest(\$B), Honest(\$A)
 , Neq(passnum1, passnum2)]

```
, CheckNonce(nb)
, CheckPassNum(passnum2)
, Eq(hash1, hash2)
, UniqueNonce(nb)]->
[ St_A_2($A, ltkA, pkB, $B, m)]
```

restriction Unique:

```
"All x #i #j. UniqueNonce(x) @i & UniqueNonce(x) @j ==> #i = #j"
```

restriction Equality:

```
"All x y #i. Eq(x, y) @i ==> x=y"
```

restriction Inequality:

```
"All x y #i. Neq(x, y) @i ==> not x=y"
```

lemma executable:

exists-trace

```
"Ex A B m1 m2 n1 n2 #i #j #k #n.
Send1(A,m1,n1)@i & Recv1(B,m1,n1) @j
& Send2(A,m2,n2)@k & Recv2(B,m2,n2) @n"
```

lemma message_authentication:

```
"All b m n #i. Authentic(b,m,n) @i
==> (Ex #j. Send1(b,m,n) @j & j<i)
| (Ex #q. Send2(b,m,n) @q & q<i)
| (Ex B #r. Reveal(B)@r & Honest(B) @i & r < i)"
```

lemma distinct_nonces:

```
"All n #i #j. CheckNonce(n)@i & CheckNonce(n)@j ==> #i=#j"
```

lemma flow_integrity:

```
"All n1 n2 #i #j. MessageNum(n1) @i & CheckPassNum(n2) @j==>
not n1 = n2"
```

end

Результаты проверки исправленного протокола с использованием Tamarin prover показаны на рисунках с 16 по 18.

```

lemma distinct_nonces:
  all-traces
  "∀ n #i #j.
    ((CheckNonce( n ) @ #i) ∧
    (CheckNonce( n ) @ #j)) → (#i = #j)"
  simplify
  solve( (#i < #j) ∨ (#j < #i) )
  case case_1
  solve( CheckNonce( n ) @ #i )
  case A_2_receive
  solve( St_A_1( $A, ltkA, pk(x), $
m, hash1, passnum1 ) ▷0 #i )
  case A_1_send
  solve( CheckNonce( n ) @ #j )
  case A_2_receive
  by contradiction /* cyclic
*/
qed
qed
next
case B_1_receive
solve( !Ltk( $B, ltkB ) ▷0 #i )
case Register_pk
solve( !Pk( $A, pk(x) ) ▷1 #i
)
case Register_pk
solve( CheckNonce( n ) @ #j

```

Рисунок 16 – Результат проверки исправленного протокола передачи сообщений с обеспечением неотказуемости на уязвимость отсутствия защиты от атак повтора

```

lemma flow_integrity:
  all-traces
  "∀ n1 n2 #i #j.
    ((MessageNum( n1 ) @ #i) ∧
    (CheckPassNum( n2 ) @ #j)) →
    (¬(n1 = n2))"
  simplify
  solve( !Ltk( $A, ltkA ) ▷2 #i )
  case Register_pk
  solve( !Pk( $B, pkB ) ▷3 #i )
  case Register_pk
  solve( St_A_1( $A.1, ltkA, pk(x),
$B.1, m.1, hash1, passnum1
) ▷0 #j )
  case A_1_send
  by contradiction /* from formulas
*/
qed
qed
qed

```

Рисунок 17 – Результат проверки исправленного протокола передачи сообщений с обеспечением неотказуемости на уязвимость отсутствия контроля целостности потока сообщений

```

lemma message_authentication:
  all-traces
    "∀ b m n #i.
      (Authentic( b, m, n ) @ #i) →
      (((∃ #j. (Send1( b, m, n ) @
#j) ∧ (#j < #i)) ∨
      (∃ #q. (Send2( b, m, n ) @
#q) ∧ (#q < #i))) ∨
      (∃ B #r.
        ((Reveal( B ) @ #r) ∧
(Honest( B ) @ #i)) ∧ (#r < #i)))"
  simplify
  solve( Authentic( b, m, n ) @ #i )
  case A_2_receive
  solve( St_A_1( $A, ltkA, pk(x), $B,
m, hash1, passnum1 ) @ #i )
  case A_1_send
  solve( !KU( sign(<$A, $B, n,
h(<<$B, $A, ~na,
~m, '1'>,
sign(<$B, $A,
~na, ~m, '1'>, ~ltkX)>),
passnum2>,
~ltkX.1)
) @ #vk.10 )
  case B_1_receive
  by contradiction /* from formulas
*/
next
case c_sign
solve( !KU( ~ltkX.1 ) @ #vk.11

```

Рисунок 18 – Результат проверки исправленного протокола передачи сообщений с обеспечением неотказуемости на уязвимости вычисления отсутствия аутентификации одного из участников протокола и вычисления цифровой подписи не от всех полей сообщения

Как видно из рисунков с 16 по 18, все леммы окрашены в зеленый цвет, значит, Tamarin prover не нашел векторов атаки на исправленный протокол. Следовательно, уязвимости закрыты.

4 Разработка виртуального стенда для проведения лабораторного практикума

Для проведения лабораторного практикума был разработан виртуальный стенд состоящий из следующих компонентов:

- хостовая операционная система Debian;
- гостевая операционная система Linux Mint;
- программное средство Tamarin prover;
- браузер Mozilla Firefox;
- файлы FirstExample.spthy, auth1.spthy, auth2.spthy, conf.spthy, non-rep.spthy.

UML-диаграмма компонентов лабораторного стенда изображена на рисунке 19.

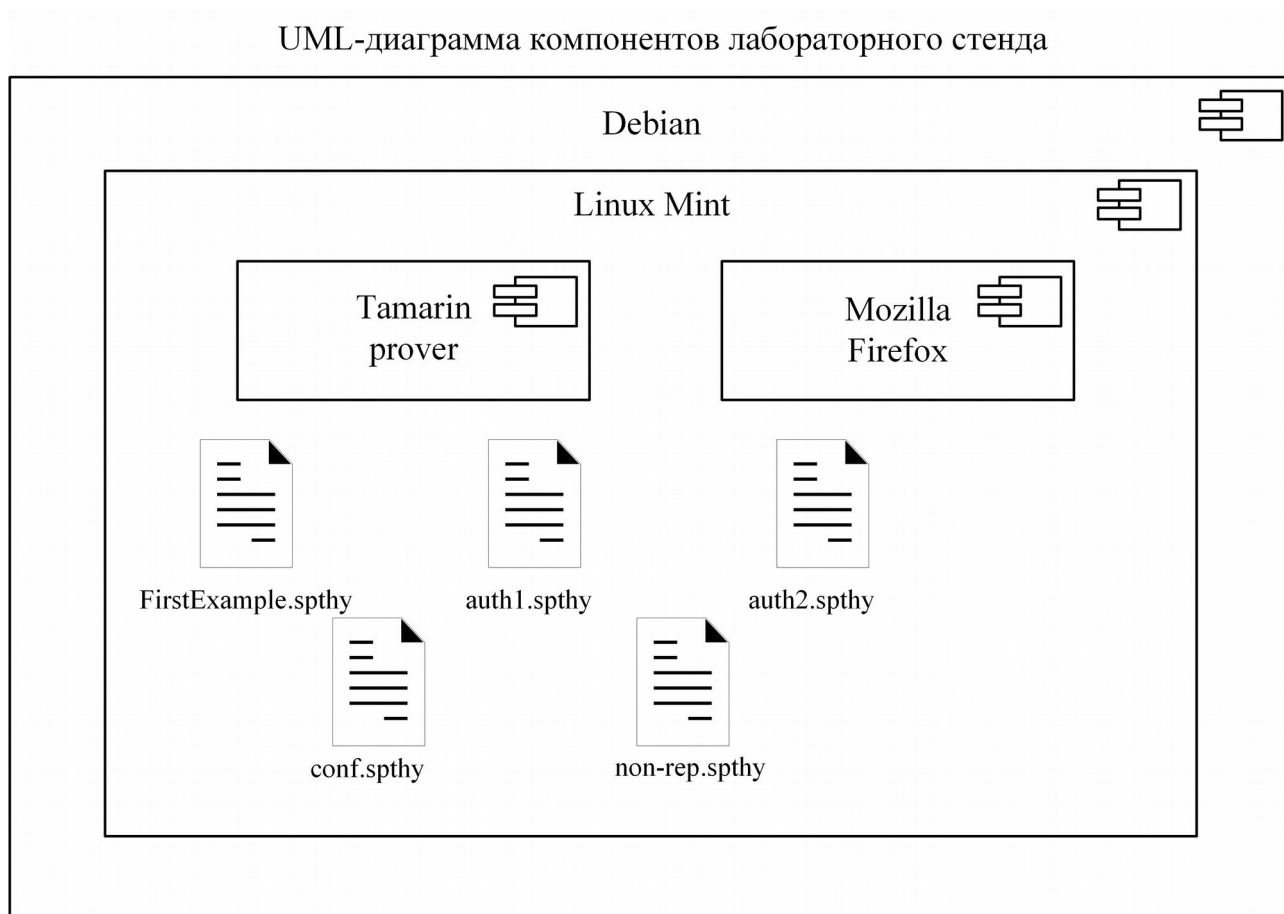


Рисунок 19 – UML-диаграмма компонентов лабораторного стенда

Для реализации лабораторного стенда на компьютерах кафедры «Информационная безопасность систем и технологий» был разработан сценарий автоматического развертывания.

В данном сценарии для настройки параметров виртуальной машины используется утилита `vboxmanage`, которая входит в комплект поставки VirtualBox.

Текст сценария приведен ниже:

```
#!/bin/bash

#переменные с параметрами виртуальной машины
NAME="Mint"
DISK="/VDisk/Mint.vdi"
RAM="2048"
VRAM="16"
OS="Linux_64"
NETNAME="SomeNet"
STRNAME="CONTROLLER_NAME"
_AUDIO="none"
_ACPI="on"
_IOAPIC="on"
_PAE="on"
_HWVIRTEX="on"
_VTXVPID="on"
_USB="on"
_CONTROLLER="PIIX4"
_MTYPE="immutable" #создает неизменяемые диски
SHAREPATH="/home/user/share"

#создание машины, указание основных параметров
vboxmanage createvm --name "$NAME" --ostype $OS --register
vboxmanage modifyvm "$NAME" --memory $RAM --vram $VRAM --acpi $_ACPI
```

```
--ioapic $_IOAPIC --pae $_PAE --chipset $_CHIPSET --hwvrtex $_HWVIRTEX --  
vtxvpid $_VTXVPID  
#настройки сети  
vboxmanage modifyvm "$NAME" --nic1 nat  
#настройка других устройств  
vboxmanage modifyvm "$NAME" --audio $_AUDIO --usb $_USB  
#добавление контроллера  
vboxmanage storagectl "$NAME" --name "$STRNAME" --add ide --controller  
$_CONTROLLER  
vboxmanage storageattach "$NAME" --storagectl "$STRNAME" --port 0 --device 0  
--type hdd --medium "$DISK" --mtype $_MTYPE  
##настройка общей папки и общего буфера обмена  
#vboxmanage sharedfolder add $NAME --name share --hostpath $SHAREPATH  
vboxmanage modifyvm "$NAME" --clipboard bidirectional  
vboxmanage modifyvm "$NAME" --draganddrop bidirectional
```

5 Разработка рекомендаций по проведению лабораторного практикума

5.1 Разработка методики проведения практикума

В целях изучения Tamarin Prover и синтаксиса его языка предлагается выполнить следующие задания:

- а) ознакомиться с руководством пользователя Tamarin prover;
- б) запустить виртуальную машину «Mint»;
- в) запустить Tamarin prover в интерактивном режиме для файла FirstExample.spthy;
- г) ознакомиться с интерфейсом Tamarin prover. Доказать все приведенные в файле леммы. Объяснить смысл каждой леммы.

В рамках исследования протокола односторонней аутентификации предлагается выполнить следующие задания:

- а) ознакомиться с протоколом односторонней аутентификации, описание которого приведено в параграфе 3.1 настоящего отчета;
- б) открыть файл auth1.spthy, содержащий описание протокола в нотации Tamarin. Объяснить смысл правил и лемм, содержащихся в файле;
- в) в протокол была внесена уязвимость. Найти данную уязвимость и доказать ее существование, построив лемму и внеся необходимый для доказательства факт действия в правила протокола;
- г) исправить уязвимость, поставив необходимое ограничение и внеся необходимый для применения ограничения факт действия в правила протокола;
- д) доказать с помощью Tamarin prover, что уязвимость закрыта.

В рамках исследования протокола взаимной аутентификации предлагается выполнить следующие задания:

- а) ознакомиться с протоколом взаимной аутентификации, описание которого приведено в параграфе 3.2 настоящего отчета;

б) открыть файл `auth2.spthy`, содержащий описание протокола в нотации Tamarin. Объяснить смысл правил и лемм, содержащихся в файле;

в) в протокол были внесены две уязвимости. Найти данные уязвимости и доказать их существование;

г) исправить уязвимости;

д) доказать с помощью Tamarin prover, что уязвимости закрыты.

В рамках исследования протокола передачи данных с обеспечением свойства конфиденциальности предлагается выполнить следующие задания:

а) ознакомиться с протоколом передачи данных с обеспечением свойства конфиденциальности, описание которого приведено в параграфе 3.3 настоящего отчета;

б) открыть файл `conf.spthy`, содержащий описание протокола в нотации Tamarin. Объяснить смысл правил и лемм, содержащихся в файле;

в) в протокол были внесены две уязвимости. Найти данные уязвимости и доказать их существование;

г) исправить уязвимости;

д) доказать с помощью Tamarin prover, что уязвимости закрыты.

В рамках исследования протокола передачи данных с обеспечением свойства неотказуемости предлагается выполнить следующие задания:

а) ознакомиться с протоколом передачи данных с обеспечением свойства неотказуемости, описание которого приведено в параграфе 3.4 настоящего отчета;

б) открыть файл `non-rep.spthy`, содержащий описание протокола в нотации Tamarin. Объяснить смысл правил и лемм, содержащихся в файле;

в) в протокол были внесены четыре уязвимости. Найти данные уязвимости и доказать их существование;

г) исправить уязвимости;

д) доказать с помощью Tamarin prover, что уязвимости закрыты.

Для закрепления полученных знаний и умений предлагается выполнить следующие задания:

- а) объяснить синтаксис и назначение правила;
- б) объяснить синтаксис и назначение леммы;
- в) объяснить синтаксис и назначение ограничения;
- г) объяснить смысл любых двух правил, встроенных в Tamarin prover, из раздела Message Theory при исследовании файлы FirstExample.spthy;
- д) объяснить смысл встроенных правил isend и irectv;
- е) описать порядок запуска Tamarin prover в графическом интерфейсе.

5.2 Разработка руководства пользователя

Tamarin prover - программа для символьного моделирования и анализа протоколов безопасности. Он принимает в качестве входных данных модель протокола безопасности, определяющую действия, выполняемые различным ролями агентов протокола (например, инициатором протокола, ответчиком и доверенным сервером ключей), спецификацию противника и спецификацию требуемых свойств протокола. На основании входных данных Tamarin prover строит доказательство того, что, даже когда произвольно много экземпляров ролей протокола, включая противника, работают параллельно, протокол отвечает заданным свойствам.

Tamarin предоставляет два способа построения доказательств: автоматический и интерактивный. Если автоматический поиск доказательств завершается, Tamarin возвращает либо доказательство правильности, либо контрпример, представляющий собой атаку, нарушающую указанное свойство. При необходимости можно прибегнуть к интерактивному режиму для подробного изучения векторов атак.

Для установки Tamarin prover на Linux или Mac OS необходимо ввести следующую команду:

brew install tamarin-prover/tap/tamarin-prover

Для ОС Windows графический интерфейс Tamarin prover не поддерживается.

Для исследования с использованием Tamarin prover криптографический протокол необходимо описать при помощи специального языка.

Протокол представляется как множество состояний. Действия, выполняемые агентами, называются фактами. Факты складываются в правила (rule), которые представляют собой переход от одного состояния (исходного состояния, или предпосылки) к другому состоянию (результату). Иными словами, состояние - это набор фактов. Переход из состояния 1 в состояние 2 - правило.

Доказываемое свойство протокола описывается леммой (lemma).

Ниже приведено описание синтаксиса языка Tamarin.

Для описания каких-либо значений (сообщение, ключ, имя участника протокола) в языке Tamarin используются переменные. Вид переменной выражается с помощью префиксов:

- $\sim x$ – новое значение;
- $\$x$ – публичное значение;
- $\#i$ – временное значение;
- m — сообщение.

Также существуют строковые константы, которые записываются в одинарных кавычках 'c'.

Множества обозначаются их элементами в скобках $\langle \rangle$, записанными через запятую. Например, $\langle a, b, '1', '2' \rangle$.

Описание протокола называется теорией (theory) и обрамляется ключевыми словами begin и end.

В целом теория представляет собой конструкцию, описываемую следующим образом:

$$\begin{aligned}
& security_protocol_theory := \\
& 'theory' ident // ident - название теории \\
& \quad 'begin' \\
& \quad \quad body \\
& \quad 'end'
\end{aligned}$$

где *body* – тело протокола, состоящее из следующих множеств:

$$body := (signature_spec \mid global_heuristic \mid rule \mid restriction \mid lemma \mid formal_comment)$$

Особые характеристики протокола (*signature_spec*) – описываются следующим образом:

$$signature_spec := functions \mid equations \mid built_in$$

Особые характеристики включают в себя три множества:

— *functions* – функции, используемые протоколом. Чтобы объявить функции *f1*, ..., *fn* с арностью (количеством переменных) *a1*, ..., *an*, необходимо включить в код следующую строку:

functions: f1/a1, ..., fn/an;

— *equations* – равенства, которые используются для задания свойств функций, например, чтобы показать, что симметричное шифрование обратное расшифрованию на том же ключе. Равенства задаются с помощью ключевого слова *equations*:

equations: left1 = right1, ..., leftn = rightn;

— *built_in* – встраиваемые библиотеки функций.

Возможно включить в описание протокола библиотеки, описанные в таблице 3.

Таблица 3 – Библиотеки функций Tamarin prover

Библиотека	Описание
hashing	Моделирует хеш-функцию. Определяет символ функции $h/1$.
asymmetric-encryption	Моделирует схему шифрования с открытым ключом. Определяет функциональные символы $aenc/2$, $adec/2$ и $pk/1$, которые связаны уравнением $adec(aenc(m, pk(sk)), sk) = m$.
signing	Моделирует схему подписи. Определяет функциональные символы $sign/2$, $verify/3$, $pk/1$, и $true$, которые связаны уравнением $verify(sign(m,sk),m,pk(sk)) = true$.
revealing-signing	Моделирует схему подписи, раскрывающую сообщения. Определяет функциональные символы $revealSign/2$, $revealVerify/3$, $getMessage/1$, $pk/1$ и $true$, которые связаны уравнениями $revealVerify(revealSign(m,sk),m,pk(sk)) = true$ и $getMessage(revealSign(m,sk)) = m$.
symmetric-encryption	Моделирует симметричную схему шифрования. Определяет символы функции $senc/2$ и $sdec/2$, которые связаны уравнением $sdec(senc(m,k),k) = m$.
diffie-hellman	Моделирует группы Диффи-Хеллмана. Определяет функциональные символы $inv/1$, $1/0$ и символы $^$ и $*$. g^a используется для обозначения возведения g в степень a . Символы $*$, inv и 1 используются для моделирования показателей абелевой группы.
bilinear-pairing	Моделирует билинейные группы. Расширяет библиотеку <code>diffie-hellman</code> функциональными символами $pmult/2$ и $em/2$. $pmult(x,p)$ обозначает умножение точки p на скаляр x , $em(p,q)$ обозначает применение билинейного отображения к точкам p и q .
xor	Добавляет символы функций $\oplus/2$ (также записывается как $XOR/2$) и $zero/0$. Функция $\oplus$ является ассоциативной и коммутативной.

Продолжение таблицы 3

Библиотека	Описание
multiset	Вводит ассоциативно-коммутативный оператор +, который обычно используется для моделирования мультимножеств.

Глобальная эвристика (`global_heuristic`) устанавливает эвристику по умолчанию, которая будет использоваться при автоматическом подтверждении лемм в файле. Она определяет порядок подтверждения лемм. При работе с `Tamarin prover` в интерактивном режиме глобальная эвристика не используется. При автоматическом доказательстве теорий данный пункт также не является обязательным.

Одним из главных элементов синтаксиса языка `Tamarin` являются так называемые правила переписывания мультимножества (`rule`), описывающие действия участников протокола.

Правила состоят из предпосылки – исходного набора фактов – и заключения – окончного набора фактов. Также правило может включать в себя факты действия. Факты действия не появляются в состояниях, а только при трассировке. Они показывают, что происходит в процессе перехода из одного состояния в другое.

Таким образом, у правила есть предпосылка, заключение и факты действия. Синтаксис правил следующий:

‘rule’ rule_name:

[Fact1, Fact2]

-->[Fact3, Fact4]

или

‘rule’ rule_name:

[Fact1, Fact2]

--[ActionFact]

->[Fact3, Fact4]

Для более удобного восприятия кода Tamarin предусматривает конструкцию “let – in”. Данную конструкцию можно использовать в правилах для обозначения множества или функции через одну переменную. Например:

‘rule’ rule_name:

```
let vars = <var1, var2>
    ,hash = h(vars)
in
    [Fact1(var1), Fact2(var2)]
    --[ ActionFact ]
    ->[Fact3(vars), Fact4(hash)]
```

Ниже приведены примеры правил на языке Tamarin.

Правило генерации ключевой пары:

rule Register_pk:

```
[ Fr(~ltk) ] //генерация новое значение ключа
--> [
    !Ltk($A, ~ltk), //установка нового значения в качестве секретного ключа A
    !Pk($A, pk(~ltk))
    //создание из секретного ключа открытого ключа (функцией pk(~ltk))
    //и установка его как открытого ключа абонента A
]
```

Правило раскрытия секретного ключа:

rule Reveal_ltk:

```
[ !Ltk(A, ltk) ] //если участник A устанавливает секретный ключ ltk
--[ LtkReveal(A) ]-> //действие — произошло раскрытие ключа
[ Out(ltk) ] //то ключ ltk отсылается в сеть
```

Элементы ограничения (restriction) определяют ограничения на набор рассматриваемых трассировок, иными словами, фильтруют набор трассировок протокола.

Синтаксис для указания ограничений и свойств безопасности определяется следующим образом:

- All – универсальность;
- Ex – существование;
- $\implies$ – импликация;
- & – конъюнкция;
- | – дизъюнкция;
- not – отрицание;
- $f @ i$ – ограничение действия;
- $\#i = \#j$ – равенство между временными переменными 'i' и 'j';
- $x = y$ – равенство переменных " x " и "y".

Ниже приведены примеры ограничений на языке Tamarin.

Ограничение равенства:

restriction Equality:

"All x y $\#i$. $\text{Eq}(x,y) @ i \implies x = y$ "

Данное ограничение устанавливает, что если в правиле есть факт действия $\text{Eq}(x,y)$, то Tamarin при построении доказательств лемм учитывает только такие случаи, когда $x=y$. Остальные случаи отбрасываются. Ограничение может быть полезно, например, при применении цифровой подписи или хэш-функции в протоколе. Например, если в теории присутствует ограничение Equality, а в каком-либо правиле есть факт действия $\text{Eq}(h(x), \text{hash})$, то все случаи, когда не выполняется равенство $h(x)=\text{hash}$, Tamarin prover рассматривать не будет. Иными словами, если данное равенство не выполняется, протокол разрывается.

Ограничение неравенства:

restriction Inequality:

"All x $\#i$. $\text{Neq}(x,x) @ \#i \implies F$ "

Данное ограничение может быть полезно, например, при реализации механизма порядковых номеров. Применение в теории ограничения Inequality

гарантирует, что при указании в правиле факта действия $\text{Neq}(\text{passnum1}, \text{passnum2})$, случаи, когда passnum1 будет совпадать с passnum2 , будут отбрасываться.

Ограничение уникальности:

restriction unique:

"All x #i #j. UniqueFact(x) @#i & UniqueFact(x) @#j ==> #i = #j"

Данное ограничение может быть полезно при использовании механизма нонсов. Применение в теории ограничения unique и указание факта действия UniqueFact(nonce) гарантирует, что одинаковых nonce в одной трассировке появиться не может.

Леммы (lemma) определяют свойства безопасности.

Для определения свойства могут использоваться следующие ключевые слова:

— all-traces — указатель того, что указанное свойство должно выполняться для всех трассировок;

— exists-trace — указатель того, что свойство должно выполняться как минимум для одной трассировки.

Ниже приведены примеры лемм на языке Tamarin.

Лемма, проверяющая возможность реализации протокола:

lemma executable:

exists-trace //существует трассировка

"Ex A B m #i #j. //существуют такие переменные A, B, m,

//а также временные переменные i и j

Send(A,m)@i //что если на i-м шаге протокола был выполнен факт Send
& Recv(B,m) @j" //то на j-м шаге протокола будет выполнен факт Recv

Лемма, проверяющая уникальность переменной (nonce):

lemma distinct_nonces:

all-traces //на всех трассировках

"All n #i #j.

$Act1(n)@i \ \& \ Act1(n)@j$ //если факт $Act1$ был выполнен на i -м и j -м шаге
 $\implies \#i=\#j$ //то i -ый шаг совпадает с j -м

Лемма, проверяющая аутентификацию отправителя:

lemma message_authentication:

"All b m #i. Authentic(b,m) @i //если на i -м шаге был выполнен Authentic
 $\implies$ (Ex #j. Send(b,m) @j & $j < i$) //то ранее был выполнен факт Send
 | (Ex B #r. Reveal(B)@r & Honest(B) @i & $r < i$)"
 //или ранее был выполнен факт Reveal,
 //и на i -м шаге был выполнен факт Honest

Лемма, проверяющая конфиденциальность сообщения:

lemma secrecy:

"All x #i.
 Secret(x) @i $\implies$ //если был выполнен факт Secret
 not (Ex #j. K(x)@j) //то не был выполнен факт K
 | (Ex B #r. Reveal(B)@r & Honest(B) @i)"
 //или были выполнены факты Reveal и Honest

Формальные комментарии (formal_comment) используются, чтобы сделать код более читабельным. Их синтаксис приведен ниже:

formal_comment := ident '{* ident* '*'}

Также могут применяться Си-подобные комментарии:

/* многострочный

комментарий */

// однострочный комментарий

В языке Tamarin есть несколько встроенных фактов, в числе которых:

— $Fr(\sim x)$ – создание нового значения переменной x ;

— $Ltk(A, key)$ – установление значения переменной key в качестве секретного ключа абонента A ;

— $Pk(A, key)$ – установление значения переменной key в качестве открытого ключа абонента A ;

- $\text{Out}(x)$ – отправление сообщения x в сеть;
- $\text{In}(x)$ – прием сообщения x из сети.

Также существует набор фактов, касающихся противника, например:

- $\text{KD}(x)$ – противник знает x ;
- $\text{KU}(x)$ – противник генерирует свое значение x .

Ниже приведен пример реализации криптографического протокола в нотации Tamarin.

В рассматриваемом протоколе клиент C генерирует новый симметричный ключ k , зашифровывает его с помощью открытого ключа pk_S сервера S ($aenc$ обозначает асимметричное зашифрование) и отправляет его S . Сервер подтверждает получение ключа, отправляя хэш ключа клиенту.

Формальное описание протокола приведено ниже.

У участников перед выполнением протокола доступна следующая информация: собственный секретный ключ, публичный ключ другого участника протокола. Между участниками определена операция шифрования и расшифрования, а также операция вычисления хэш-функции. Участник C обладает датчиком случайных чисел $ДСЧ$.

$$1. C: KZU_C \rightarrow PK_S$$

$$ДСЧ \rightarrow k$$

$$C_C = E_{PK_S}(k)$$

$$[k] \Rightarrow S$$

$$2. S:$$

$$[C_C]$$

$$KZU_S \rightarrow PK_S$$

$$D_S(C_C) = k$$

$$H_S = h(k)$$

$$[H_S] \Rightarrow C$$

$$3. C:$$

$[H_S]$

Описание в нотации Tamarin:

```
theory FirstExample //объявление теории под названием FirstExample
begin
builtins: hashing, asymmetric-encryption /*подключение библиотек хэширования
и асимметричного шифрования*/
// правило регистрации открытого ключа
rule Register_pk:
  [ Fr(~ltk) ] //генерация нового значения ключа
  -->
  [ !Ltk($A, ~ltk), !Pk($A, pk(~ltk)) ]
/*установка нового значение в качестве секретного ключа A,
создание из секретного ключа открытого ключа (функцией pk(~ltk))
установка его за открытый ключ абонента A
восклицательный знак ! значит, что ключ устанавливается надолго
*/
/* В данном примере злоумышленнику может получить любой открытый ключ,
используя следующее правило. По сути, он считывает запись базы данных с
открытым ключом и отправляет открытый ключ в сеть, используя встроенный
факт Out, который обозначает отправку сообщения в сеть*/
rule Get_pk:
  [ !Pk(A, pubkey) ]
  -->
  [ Out(pubkey) ]
/* Моделирование раскрытия долгосрочных закрытых ключей. Интуитивно -
читается запись базы данных с закрытым ключом и отправляется злоумышлен-
нику. Данное правило имеет факт действия LtkReveal, утверждающий, что дол-
госрочный ключ агента A был скомпрометирован.
```

*/

rule Reveal_ltk:

[!Ltk(A, ltk)]

--[LtkReveal(A)]->

[Out(ltk)]

// Правило, определяющее стартовые действия клиента при выборе сервера

rule Client_1:

[Fr(~k) // генерация нового ключа

, !Pk(\$S, pkS) // получение публичного ключа сервера

]

-->

[Client_1(\$S, ~k) // сохранение значения имени сервера

//и ключа для последующего использования

, Out(aenc(~k, pkS)) // отправка зашифрованного значения ключа в сеть

]

// Правило, определяющее дальнейшие действия клиента

rule Client_2:

[Client_1(S, k) // получение значений имени сервера и ключа,

//сохраненных на предыдущем шаге

, In(h(k)) // получение из сети хэша ключа

]

--[SessKeyC(S, k)]-> // установка k в качестве ключа сессии с сервером S

[]// дальнейших действий в данном протоколе не предусматривается

// Правило, определяющее действия сервера при ответе на запрос клиента

rule Serv_1:

[!Ltk(\$S, ~ltkS) // получение секретного ключа сервера

, In(request) // получение из сети запроса

]

--[AnswerRequest(\$S, adec(request, ~ltkS))]-> // факт действия,

```

//устанавливающий, что произошел факт ответа сервера на запрос
[ Out( h(adec(request, ~ltkS)) ) ] //отправка в сеть
// хэш-функции от расшифрованного запроса
//Лемма, описывающая свойство секретности ключа сессии
lemma Client_session_key_secrecy:
" /* Не может быть такого, что*/
not(
  Ex S k #i #j.
    /* клиент устанавливает ключ сессии 'k' для сервера 'S' */
    SessKeyC(S, k) @ #i
    /* и противник знает 'k' */
    & K(k) @ #j
    /* не зная долговременного ключа 'S'. */
    & not(Ex #r. LtkReveal(S) @ r)
)"
//Лемма, описывающая аутентификацию клиента
lemma Client_auth:
" /* Для всех ключей сессии 'k', установленных клиентами для сервера 'S' */
( All S k #i. SessKeyC(S, k) @ #i
  ==>
    /* существует сервер, ответивший на запрос */
    ( (Ex #a. AnswerRequest(S, k) @ a)
      /* или же злоумышленник раскрыл долговременных ключ сервера прежде,
чем ключ был установлен */
      | (Ex #r. LtkReveal(S) @ r & r < i)
    )
  )
"
//Лемма, описывающая инъективную аутентификацию клиента

```

lemma Client_auth_injective:

```
" /* Для всех ключей сессии 'k', установленных клиентами для сервера 'S' */
  ( All S k #i. SessKeyC(S, k) @ #i
    ==>
      /* существует сервер, ответивший на запрос */
      ( (Ex #a. AnswerRequest(S, k) @ a
        /* и не существует других клиентов с таким же запросом */
        & (All #j. SessKeyC(S, k) @ #j ==> #i = #j)
      )
      /* или же злоумышленник раскрыл долговременных ключ сервера прежде,
чем ключ был установлен */
      | (Ex #r. LtkReveal(S) @ r & r < i)
    )
  )
"
```

//Лемма, описывающая возможность установки ключа сессии без раскрытия долговременного ключа сервера

lemma Client_session_key_honest_setup:

exists-trace

```
" Ex S k #i.
  SessKeyC(S, k) @ #i
  & not(Ex #r. LtkReveal(S) @ r)
"
```

end

Для доказательства корректности лемм необходимо ввести в командную строку команду запуска Tamarin prover в интерактивном режиме для файла, содержащего в себе описание протокола:

tamarin-prover interactive FirstExample.spthy

После ввода команды в командной строке появятся такие строки:

GraphViz tool: 'dot'

checking version: dot - graphviz version 2.39.20150613.2112 (20150613.2112). OK.

maude tool: 'maude'

checking version: 2.7. OK.

checking installation: OK.

The server is starting up on port 3001.

Browse to <http://127.0.0.1:3001> once the server is ready.

Loading the security protocol theories './*.spthy' ...

Finished loading theories ... server ready at

<http://127.0.0.1:3001>

21/Jun/2019:09:16:01 +0200 [Info#yesod-core] Application launched

@(yesod_83PxojfltaB8w9Rj9nFdZm:Yesod.Core.Dispatch
./Yesod/Core/Dispatch.hs:157:11)

Если при описании протокола были допущены синтаксические или иные ошибки (например, использование в одном правиле одной и той же переменной с символом $\sim$ и без него), то информация о данных ошибках будет отображена в командной строке. Также будет отображена информация о предупреждениях, если, к примеру, ограничение (restriction) было определено, но не было использовано в правилах.

Информация об ошибке включает в себя описание ошибки, а также ее расположение в файле с расширением `.spthy` (номер строки и номер символа в строке). Это позволяет быстро отыскать в файле и исправить ошибку.

В независимости от наличия или отсутствия ошибок, при вызове `Tamarin prover` запустит веб-сервер, который загрузит все теории протоколов

безопасности из рабочего каталога командной строки. Для просмотра теорий необходимо ввести в адресной строке браузера `http://localhost:3001`.

Стартовая страница Tamarin prover показана на рисунке 20.

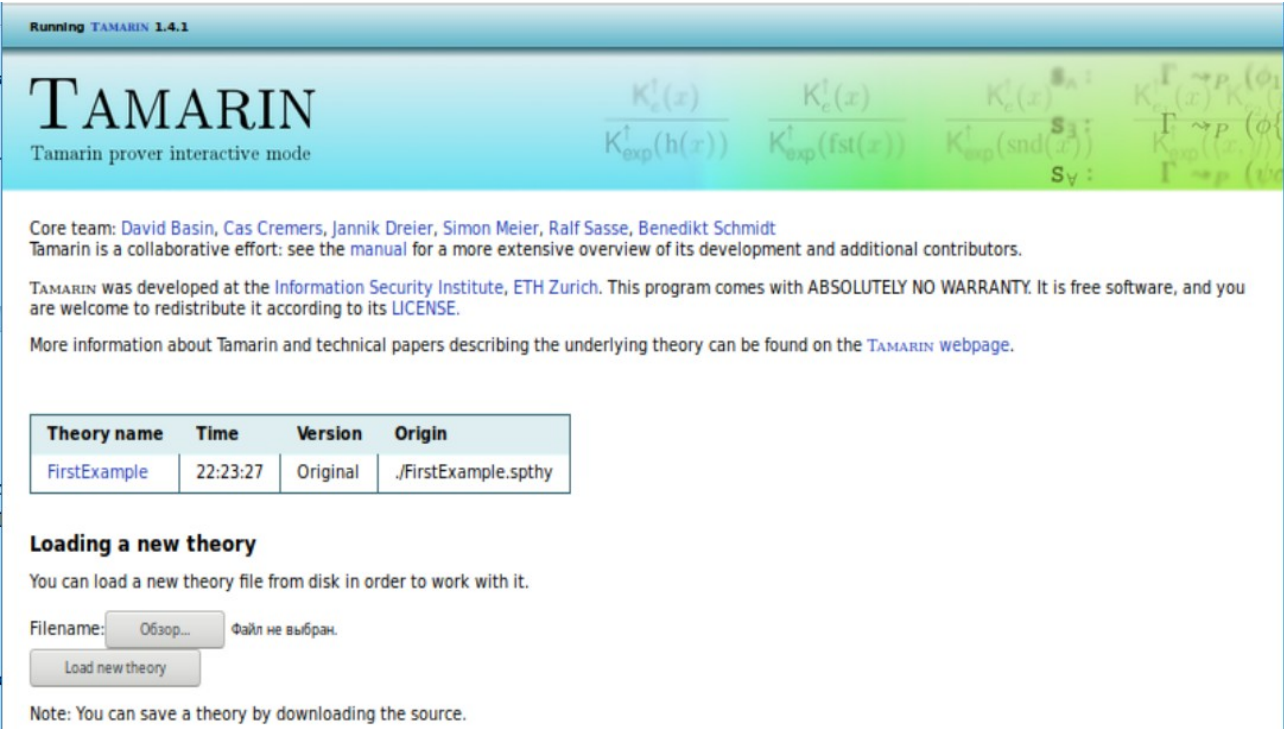


Рисунок 20 – Стартовая страница Tamarin prover

Таблица в середине показывает все загруженные теории. Существует возможность выбрать теорию, чтобы исследовать ее и доказать свойства безопасности, либо загрузить дальнейшие теории, используя форму загрузки внизу (в таком случае не будут отображены предупреждения). Рекомендуется запускать Tamarin из командной строки в соответствующем каталоге.

При выборе записи «FirstExample» в таблице загруженных теорий появляется страница, изображенная на рисунке 21.

Слева на рисунке 21 показана теория: ссылки на теорию сообщений (message theory), описывающую противника, правила и ограничения перезаписи мультимножества (multiset rewrite rules), описывающие протокол, необработанные и уточненные источники (raw sources) и леммы, которые предстоит доказать.

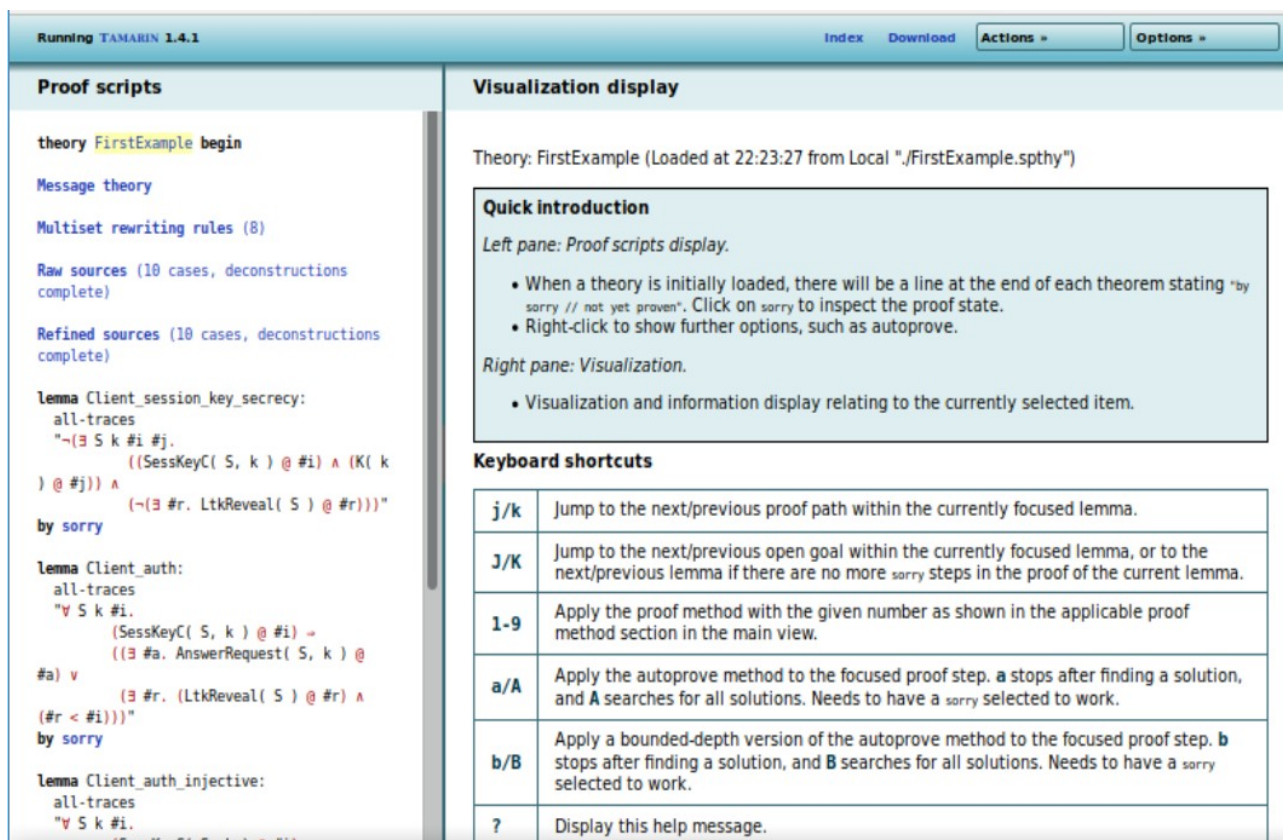


Рисунок 21 – Описание протокола и доступные команды

Справа – краткая сводка доступных команд и сочетаний клавиш, которые можно использовать для навигации внутри теории. В правом верхнем углу есть несколько ссылок: Index ведет обратно на стартовую страницу, Download позволяет загрузить теорию, Actions-->Show source показывает исходный код теории, а Options позволяют настроить уровень детализации в визуализации графика.

Вкладка Message theory показана на рисунке 22.

Справа на рисунке 22 показана теория сообщений, начиная с так называемой подписи (Signature), которая состоит из всех функций и уравнений. Они могут быть определены пользователем или импортированы с помощью встроенных модулей.

Ниже приведены правила построения (Construction rules). Эти правила описывают функции, которые может применять противник. Например, следующее правило:

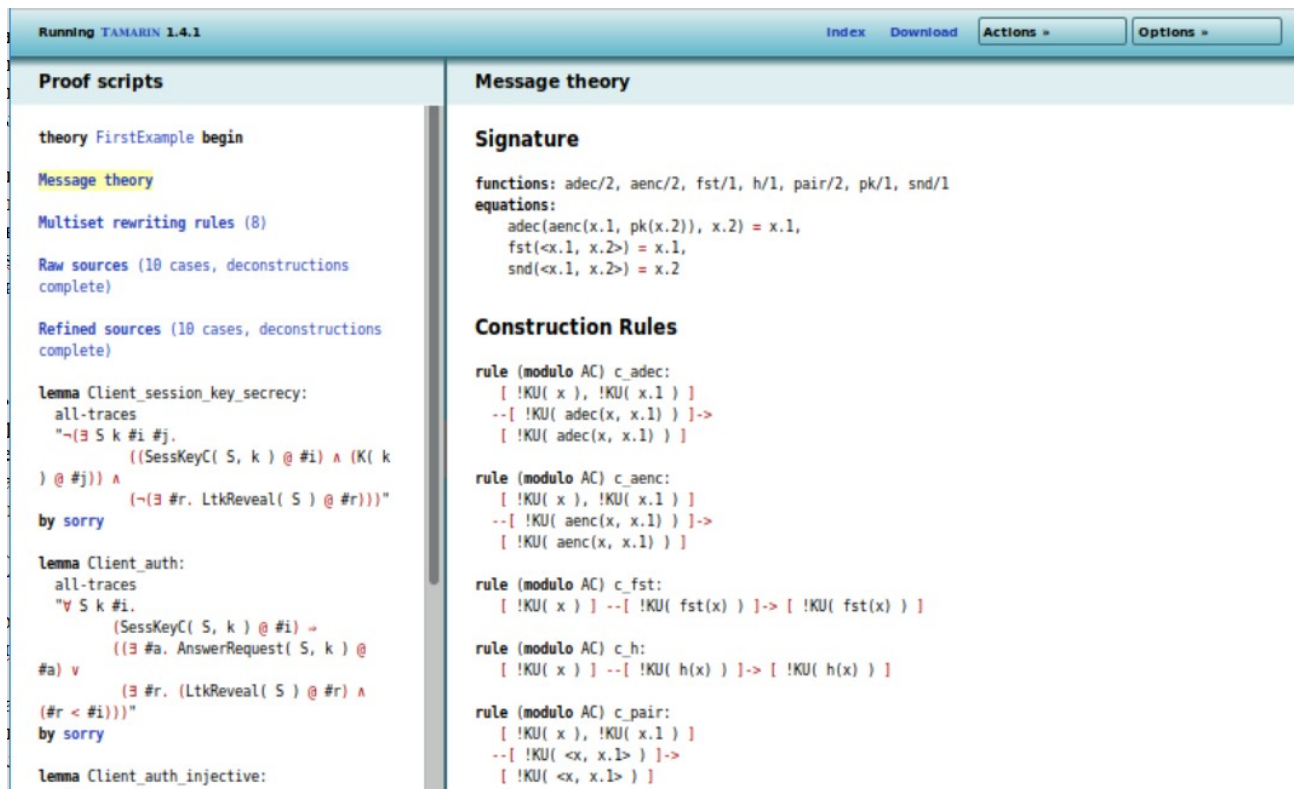


Рисунок 22 – Вкладка Message theory

rule (modulo AC) ch:

$$[!KU(x)] \text{ -- } [!KU(h(x))] \text{ --> } [!KU(h(x))]$$

выражает, что если противник знает x (представленное фактом $!KU(x)$ в предпосылке), то он может вычислить $h(x)$ (представленный фактом $!KU(h(x))$ в заключении), то есть хэш x . Факт действия $!KU(h(x))$ посередине также записывает это для целей рассуждения.

Также существуют правила деконструкции (Deconstruction rules). Эти правила описывают, какие термины противник может извлечь из более крупных терминов, применяя функции. Например, следующее правило:

rule (modulo AC) dfst:

$$[!KD(<x.1, x.2>)] \text{ --> } [!KD(x.1)]$$

гласит, что если противник знает пару $<x.1, x.2>$ (представлено фактом $!KD(<x.1, x.2>)$), то он может извлечь первое значение $x.1$ (представлен фактом $!KD(x.1)$) от него. Это происходит в результате применения функции fst к паре, а затем с помощью уравнения $fst(<x.1, x.2>) = x.1$.

Вкладка Multiset rewriting rules изображена на рисунке 23.

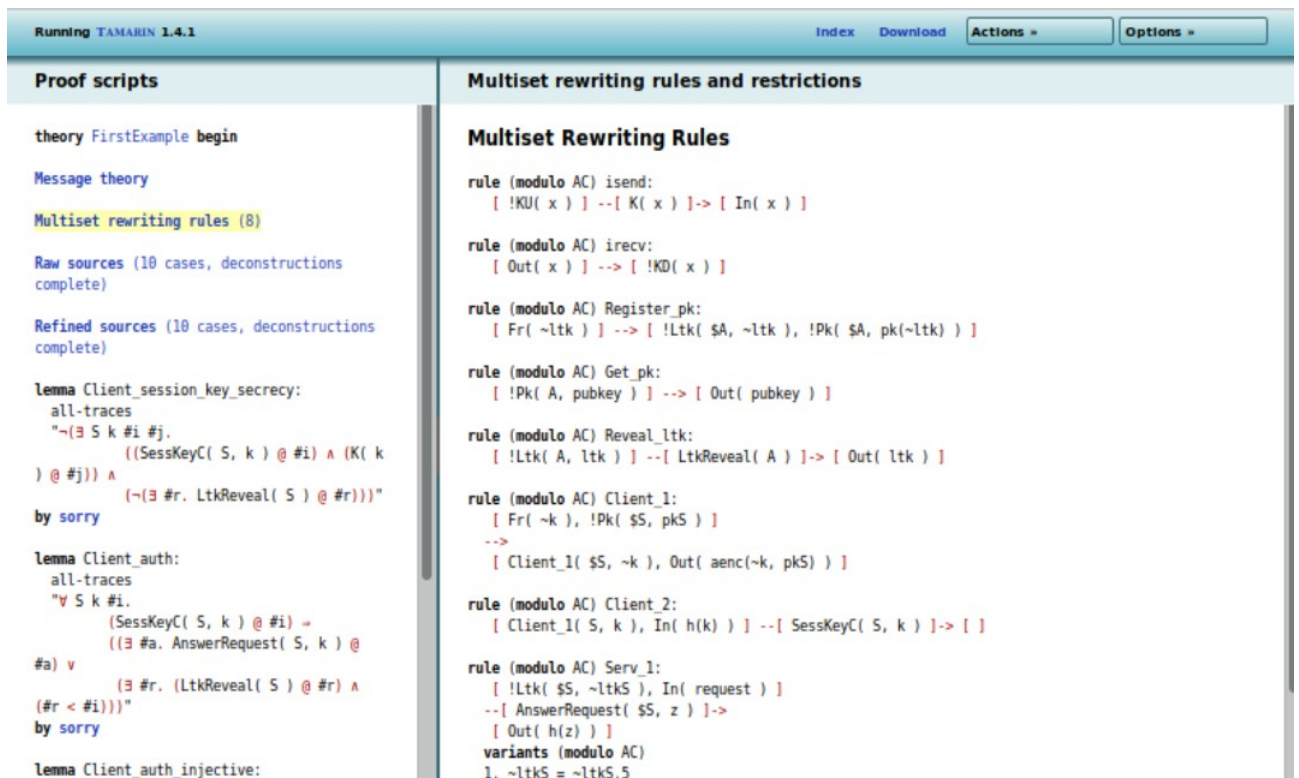


Рисунок 23 – Вкладка Multiset rewriting rules

В правой части рисунка 23 находятся правила перезаписи протокола, а также два дополнительных правила: *isend* и *irecv*, моделирующими контроль противника над сетью:

rule irecv:

$$[\text{Out}(x)] \rightarrow [!\text{KD}(x)]$$

rule isend:

$$[!\text{KU}(x)] \rightarrow [\text{K}(x)] \rightarrow [\text{In}(x)]$$

Правило *irecv* гласит, что любое сообщение, отправленное агентом с использованием факта *Out*, изучается противником.

В правиле *isend* говорится, что любое сообщение, полученное агентом протокола, может быть построено противником.

Данные два правила важны при рассмотрении графиков атаки, построенных Tamarin.

Вкладка Refined sources изображена на рисунке 24.

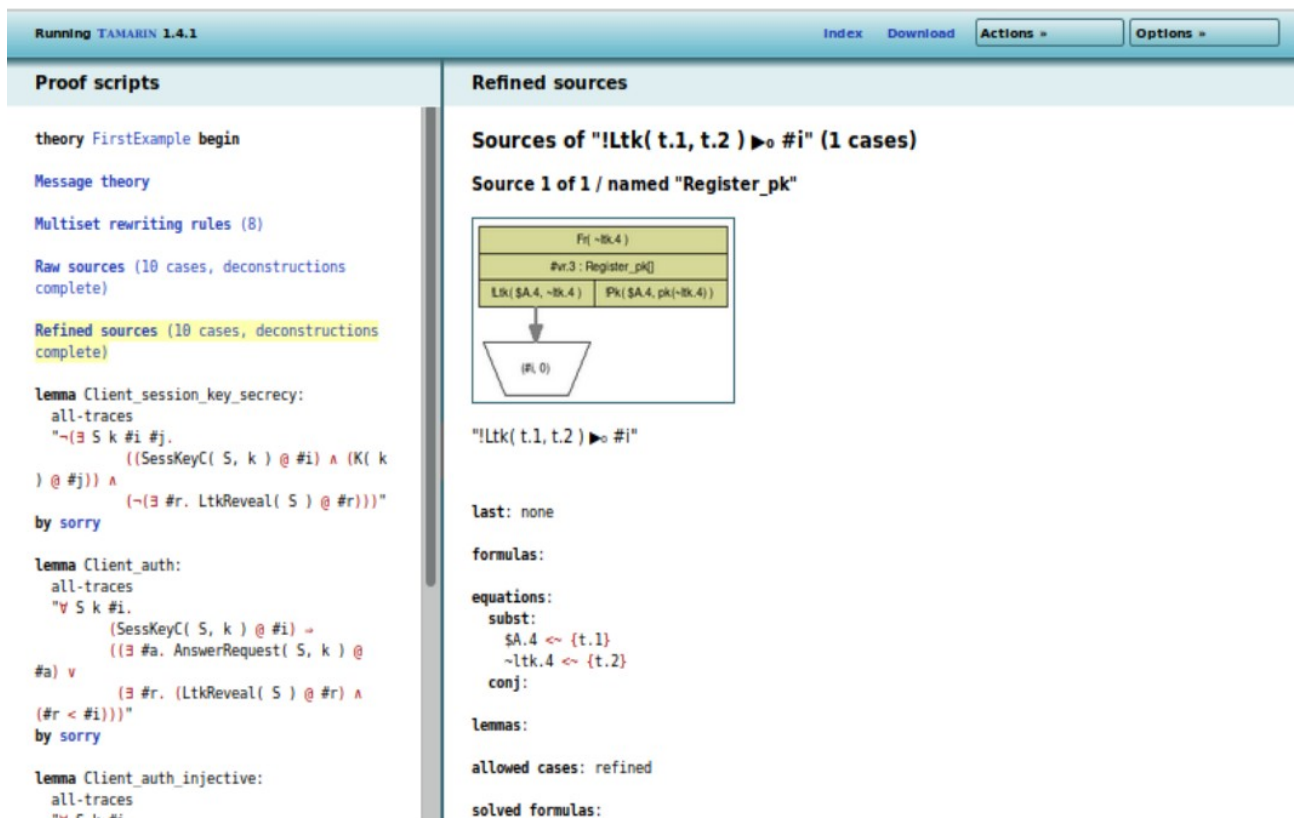


Рисунок 24 – Вкладка Refined sources

Чтобы повысить эффективность своих внутренних рассуждений, Tamarin предварительно вычисляет различия между ситуациями, рассматриваемыми программой. Предварительные вычисления используются, чтобы избежать возможного дублирования рассматриваемых ситуаций.

Для доказательства леммы в интерактивном режиме необходимо нажать на надпись *sorry* (данное слово показывает, что доказательство не было запущено) после первой леммы в левом окне. Вкладка доказательства леммы изображена на рисунке 25.

При доказательстве леммы Tamarin уточняет знания, которые он имеет о свойстве и протоколе (так называемую систему ограничений), пока не сможет либо заключить, что свойство выполняется во всех возможных случаях, либо пока не найдет контрпример к лемме. На рисунке 25 можно видеть возможные шаги доказательства вверху и текущее состояние системы ограничений ниже (которое пусто, поскольку доказательство еще не начато).

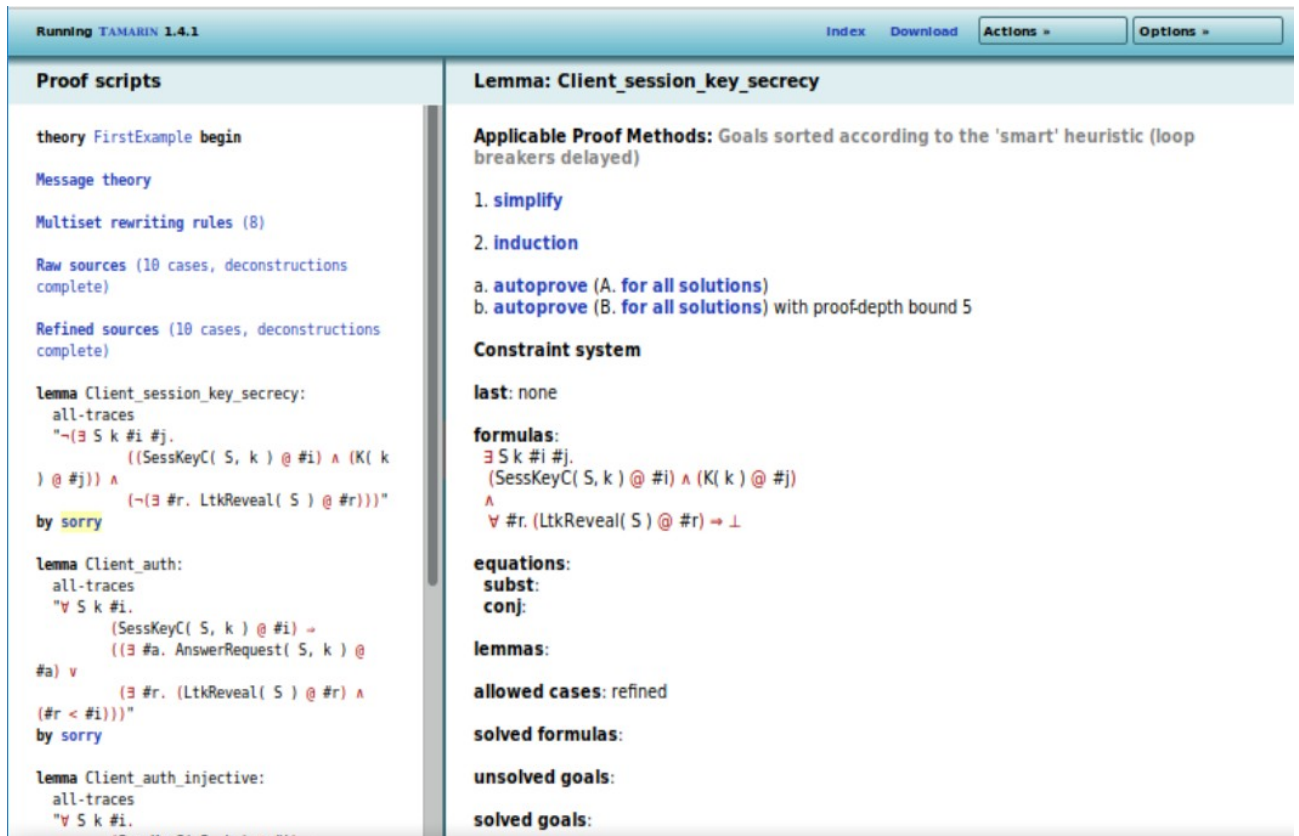


Рисунок 25 – Вкладка доказательства леммы

Доказательство всегда начинается с шага упрощения (*1. simplify*), который переводит лемму в исходную систему ограничений, которая должна быть решена, или с шага установки индукции (*2. induction*), которая генерирует необходимые ограничения для доказательства леммы с использованием индукции. Стратегия доказательства по умолчанию – шаг упрощения. Вид вкладки доказательства леммы после упрощения показан на рисунке 26.

Tamarin переводит лемму в систему ограничений. Поскольку Tamarin ищет контрпримеры к лемме, он ищет выполнение протокола, которое содержит действия $SessKeyC(S, k)$ и $K(k)$, но не использует $LtkReveal(S)$.

Это визуализируется на графике следующим образом. Единственный способ получить действие $SessKeyC(S, k)$ - использовать экземпляр правила *Client_2* слева, а правило $K(k)$ символизируется справа с помощью круглого поля (рассуждение противника всегда визуализируется с помощью круглых полей).

Running TAMARIN 1.4.1
Index
Download
Actions
Options

Proof scripts

```

theory FirstExample begin

  Message theory

  Multiset rewriting rules (8)

  Raw sources (10 cases, deconstructions complete)

  Refined sources (10 cases, deconstructions complete)

  lemma Client_session_key_secret:
    all-traces
    "~(∃ S k #i #j.
      ((SessKeyC( S, k ) @ #i) ∧ (K( k
        ) @ #j)) ∧
      (¬(∃ #r. LtkReveal( S ) @ #r)))"
  simplify
  solve( Client_1( S, k ) @ #i )
  case Client_1
  solve( !KU( ~k ) @ #vk.1 )
  case Client_1
  solve( !KU( ~ltk ) @ #vk.2 )
  case Reveal_ltk
  by contradiction /* from formulas
  qed
qed
qed

lemma Client_auth:
  all-traces
  "∀ S k #i.
    (SessKeyC( S, k ) @ #i)

```

Case: Reveal_ltk

1. **contradiction** /* from formulas */
2. **solve**(!KU(h(~k)) @ #vk.2) // nr. 4

a. **autoprove** (A. for all solutions)
b. **autoprove** (B. for all solutions) with proofdepth bound 5

Constraint system

Рисунок 27 – Результат доказательства леммы

6 Вредные психофизиологические факторы, влияющие на органы зрения специалиста по защите информации. Методы и средства физической культуры, снижающие их воздействие

Специалист по защите информации на протяжении всего дня работает как с бумажными носителями информации, так и за компьютером, что негативно сказывается на работе органов зрения. Чтение оказывает большую нагрузку на глаза, так как восприятие текста связано с чрезвычайно быстрым выполнением зрительным анализатором работы по различению огромного количества крайне мелких деталей и по перемещению вдоль строк и от строки к строке [5]. Работа же на персональном компьютере (ПК) связана с постоянным напряжением функций зрительного аппарата. Если при обычном чтении световой поток падает непосредственно на бумагу и специалист считывает отраженный текст, то при работе на персональном компьютере глаза работника смотрят непосредственно на источник света, поэтому органы зрения вынуждены работать в постоянном стрессовом режиме в течение 5-8 часов [6].

Такой режим работы приводит к возникновению так называемого «компьютерного зрительного синдрома» (КЗС) [7]. Его симптомами являются жжение в глазах, покраснение век и конъюнктивы, чувство инородного тела или песка под веками, боли в области глазниц и лба, затуманивание зрения, замедленная перефокусировка с ближних объектов на дальние. Одной из причин КЗС является то, что миллионы лет эволюции у человека сформировалась определенная зрительная система, которая приспособлена для восприятия объектов в отраженном свете (рисунки, печатные тексты). Изображение же на дисплее кардинально отличается от бумажных, так как состоит из дискретных точек, светится, мерцает. Чтобы выбрать необходимую информацию из ее чрезмерного потока в виде светящихся строчек, человек задерживает моргание, что приводит к перенапряжению глазных мышц. Более

того, специалист вынужден часто переводить взгляд с экрана на клавиатуру, которые по-разному освещены и находятся на разном расстоянии.

Всё вышеперечисленное приводит к тому, что у специалистов по защите информации могут возникнуть заболевания органов зрения. Так, по данным Всемирной организации здравоохранения, у операторов ПК отмечается повышенная частота заболеваний глаукомой и катарактой и такие операторы вынуждены каждые 6-9 минут менять очки в сторону их усиления [8].

Наиболее негативно влияют на зрения следующие факторы:

- отсутствие или недостаток естественного света;
- недостаточная искусственная освещенность рабочей зоны;
- повышенная яркость света;
- повышенная контрастность;
- прямая и отраженная блескость;
- недостаточно высокое напряжение монитора;
- несоблюдение расстояния от глаз оператора до экрана;
- зрительное напряжение.

К средствам физической культуры, снижающим воздействие приведенных факторов, относятся [9]:

- физические упражнения;
- оздоровительные силы природной среды;
- гигиенические факторы.

Под физическими упражнениями понимается двигательная деятельность человека, специально организуемая для снижения воздействия негативных факторов на органы человека [10]. Примерами таких упражнений являются вращение глазами, быстрые моргания, зажмуривание, массаж глаз.

Оздоровительные силы природы дополняют физические упражнения такими важными факторами, которыми они сами не обладают. Закаливание солнцем, воздухом и водой не только укрепляет здоровье органов зрения, но и

обеспечивает физическую, психическую и волевую подготовленность к жизни, особенно в стрессовых условиях.

Гигиенические факторы дополняют оздоровительное воздействие физических упражнений. Гигиенически обоснованные режимы жизни способствуют укреплению и сохранению здоровья и поддержанию физической и умственной работоспособности. Для соблюдения гигиены зрения можно выделить следующие рекомендации:

- правильное обустройство рабочего места (освещение, расстояние от экрана или бумаги до глаз);
- организация перерывов в работе каждые 30 минут;
- правильное питание;
- защита глаз от воздействия ультрафиолетового излучения.

Заключение

В ходе выполнения выпускной квалификационной работы был произведен анализ методов анализа уязвимостей криптографических протоколов. Был рассмотрен ряд средств анализа уязвимостей криптографических протоколов, при этом было произведено сравнение указанных средств по разработанным критериям. Затем был изучен синтаксис языка Tamarin, а также были получены навыки работы с Tamarin prover.

В результате был разработан лабораторный стенд, содержащий компоненты, необходимые для проведения лабораторного практикума. Был разработан ряд протоколов со внесенными уязвимостями, их формальное описание и описание в нотации Tamarin. Была разработана методика проведения практикума. Было оформлено руководство пользователя Tamarin prover, содержащее в себе синтаксис используемого языка, инструкцию по поиску уязвимостей с помощью Tamarin prover и пример такого поиска.

Таким образом, задание на выпускную квалификационную работу выполнено в полном объеме.

Лабораторный практикум, разработка которого осуществлялась в ходе выпускной квалификационной работы, был внедрен для использования на кафедре «Информационная безопасность систем и технологий».

Список используемых источников

- 1 А. В. Черемушкин. Криптографические протоколы: основные свойства и уязвимости. М.: Академия, 2009.
- 2 А.С. Косачев, В.Н. Пономаренко. Анализ подходов к верификации функций безопасности и мобильности. М.: Институт системного программирования РАН, 2004.
- 3 Основные принципы конструирования и анализа криптографических протоколов [электронный ресурс] – http://cryptowiki.net/index.php?title=%D0%9E%D1%81%D0%BD%D0%BE%D0%B2%D0%BD%D1%8B%D0%B5_%D0%BF%D1%80%D0%B8%D0%BD%D1%86%D0%B8%D0%BF%D1%8B_%D0%BA%D0%BE%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F_%D0%B8_%D0%B0%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0_%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D1%85_%D0%BF%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB%D0%BE%D0%B2 – режим доступа свободный – дата последнего обращения 01.06.2019.
- 4 Tamarin Prover Introduction [электронный ресурс] – https://tamarin-prover.github.io/manual/book/001_introduction.html – режим доступа свободный – дата последнего обращения 01.06.2019.
- 5 Влияние чтения на зрение [электронный ресурс] – <https://www.glazmed.ru/news/news-0097.shtml> – режим доступа свободный – дата последнего обращения 09.06.2019.
- 6 Вредные и опасные факторы при работе с компьютером [электронный ресурс] – <http://www.grandars.ru/shkola/bezopasnost->

zhiznedeyatelnosti/vrednye-factory-pri-rabote-na-pk.html – режим доступа свободный – дата последнего обращения 09.06.2019.

7 Почему работа за компьютером наносит вред здоровью человека [электронный ресурс] – <http://headinsider.info/1x773b.html> – режим доступа свободный – дата последнего обращения 09.06.2019.

8 Синдром компьютерного стресса [электронный ресурс] – http://e-ikt.uginfo.sfedu.ru/lectures/lecture15/lecture15_3.html – режим доступа свободный – дата последнего обращения 09.06.2019.

9 Оздоровительные силы природы и гигиенические факторы как средства физического воспитания [электронный ресурс] – <https://www.fizkulturaisport.ru/fizvospitanie/teoriya/273-ozdosrovitelnye-sily-prirody-i-gigienicheskie-factory-ka> – режим доступа свободный – дата последнего обращения 09.06.2019.

10 Физические упражнения как основное средство физического воспитания [электронный ресурс] – <https://www.fizkulturaisport.ru/fizvospitanie/teoriya/272-fizic> – режим доступа свободный – дата последнего обращения 09.06.2019.