

Политехнический институт
Факультет приборостроения,
информационных технологий и
электроники

Кафедра
«Информационная безопасность
систем и технологий»

Специальность 100503 –

Специализация -

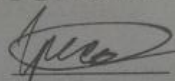
Информационная безопасность
автоматизированных систем
Защищенные автоматизированные
системы управления

ДИПЛОМНЫЙ ПРОЕКТ

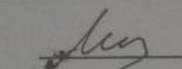
на тему

ПРАКТИКУМ ДЛЯ АНАЛИЗА УЯЗВИМОСТЕЙ КРИПТОГРАФИЧЕСКИХ
ПРОТОКОЛОВ

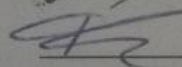
Студент

 Гресько Владислав Петрович

Руководитель

 Липилин О. В.

Нормоконтролёр

 Фатеев А. Г.

Рецензент заместитель ген. директора
по спецвопросам АО «ПНИЭИ»

 Сомкин С. А.

Работа допущена к защите (протокол заседания кафедры от 20.06.17 № 14)

Заведующий кафедрой

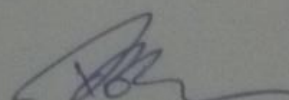
 С.Л. Зефирова

Работа защищена с отметкой

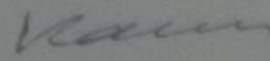
хорошо (протокол заседания ГЭК

от 23.06.17 № 9)

Секретарь ГЭК

 А.П. Иванов

Пенза 2017


9500
21.06.17

РЕФЕРАТ

Отчет содержит 71с., 14 рис., 3 таблицы, 9 источников.

КРИПТОГРАФИЧЕСКИЙ ПРОТОКОЛ, ФОРМАЛЬНОЕ ОПИСАНИЕ ПРОТОКОЛА, СРЕДСТВА АВТОМАТИЗИРОВАННОГО АНАЛИЗА ПРОТОКОЛОВ, ЛАБОРАТОРНЫЙ ПРАКТИКУМ

Объектом данного исследования являются средства автоматизированного анализа криптографических протоколов.

Цель дипломного проекта – разработка лабораторного практикума для применения навыков формального описания криптографических протоколов и средств автоматизированного анализа криптографических протоколов.

В процессе работы был проведен обзор основных свойств безопасности криптографических протоколов, обзор основных методов и средств автоматизированного анализа криптографических протоколов, разработана архитектура лабораторного стенда для изучения способов формального описания криптографических протоколов и лабораторный практикум, в котором отрабатываются навыки использования средств автоматизированного анализа криптографических протоколов.

В результате был разработан лабораторный стенд для применения навыков формального описания криптографических протоколов и оперирования средствами автоматизированного анализа криптографических протоколов, а так же лабораторный практикум, включающий в себя рекомендации по выполнению заданий.

СОДЕРЖАНИЕ

Обозначения и сокращения.....	5
Введение.....	6
1 Обзор основных свойств безопасности криптографических протоколов	7
1.1 Понятие криптографического протокола	7
1.2 Свойства, характеризующие безопасность протоколов	12
1.3 Разбор свойств безопасности криптографических протоколов.	13
2 Обзор методов анализа уязвимостей криптографических протоколов..	18
2.1 Алгебраические свойства, анализируемые при проверке корректности криптографических протоколов.....	18
2.2 Проверка корректности криптографических протоколов с использованием формальных методов анализа	20
3 Обзор средств автоматизированного анализа протоколов	25
3.1 Значение автоматизированного анализа протоколов.....	25
3.2 AVISPA	25
3.3 Scyther	26
3.4 ProVerif.....	28
3.5 Isabelle	28
3.6 SPIN	29
4 Обзор языка спецификации протоколов высокого уровня HLPSL	32
4.1 Понятие ролей и переходов	32
4.2 Синтаксис языка HLPSL	42
5 Разработка лабораторного стенда для проведения лабораторного практикума	44
6 Реализация лабораторного практикума для средств AVISPA и SPAN..	45
6.1 Ознакомление с языком описания криптографических протоколов HLSPL и инструментом анализа протоколов AVISPA	45
6.2 Реализация протокола обеспечения конфиденциальности и целостности передачи сообщения между двумя участниками на языке	

HLPSL и проверка его корректности.....	51
6.3 Реализация протокола явного ключевого аутентифицированного обмена с использованием языка спецификации протоколов высокого уровня HLPSL и инструмента анализа протоколов AVISPA.....	58
Заключение	69
Список использованных источников	71

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

КП – криптографический протокол

САА – средство автоматизированного анализа

ОС – операционная система

ВВЕДЕНИЕ

В сфере безопасности информационных технологий для безопасной передачи сообщений по большей части используются технологии криптографии. Процесс зашифрования, передачи, дешифрования регламентируется особым порядком действий – протоколом. Развитие современных технологий не обошло стороной и криптографию: протоколы, считавшиеся надежными в течение десятилетий, обнаруживают свою неработоспособность в определенных условиях. Это возникло благодаря формализации и автоматизации средств анализа протоколов.

Студентам, обучающимся по специальности 10.03.05 «Информационная безопасность автоматизированных систем» необходимо получить знания и выработать практические навыки по применению средств автоматизированного анализа криптографических протоколов, так как установление причин уязвимостей и изучение атак на криптографические протоколы, которые реализуются благодаря им – отличная база для создания защищенных протоколов. Это позволит избежать прошлые ошибки и синтезировать надежные, относительно предыдущих, протоколы. Таким образом, тема выпускной квалификационной работы является актуальной.

1 Обзор основных свойств безопасности криптографических протоколов

1.1 Понятие криптографического протокола

Криптографический протокол - это процедура, выполняемая между двумя сторонами, которая используется для выполнения некоторой задачи безопасности. Обычно криптографические протоколы используют один или несколько криптографических примитивов и / или схем. Примером может служить передача номера кредитной карты от Боба на веб-сайт электронной коммерции Алисы. Такой протокол может включать в себя схему цифровой подписи (так что Боб знает, что он разговаривает с Алисой), и форма шифрования (для обеспечения того, чтобы данные кредитной карты Боба не перехватывались в пути). Примерами развернутых протоколов, которые выполняют такие операции, являются TLS или IPSec.

С понятием протокола непосредственно связаны понятия цикла, шага, роли и сеанса. Последовательность шагов протокола группируется в циклы. Цикл (проход) протокола (round, pass of cryptographic protocol) в криптографических протоколах с двумя участниками временной интервал, в котором активен только один из участников. Цикл (проход) завершается формированием и отсылкой сообщения с последующим переходом активного участника в состояние ожидания и передачей активности другому участнику. В протоколах с тремя и более участниками в синхронном случае цикл период времени между двумя точками синхронизации. К очередной точке синхронизации каждый участник должен отослать все сообщения, которые ему предписано передать другим участникам в текущем цикле. В протоколах интерактивного доказательства циклом (раундом) часто называют комбинацию из трех шагов: заявка, запрос, ответ. В асинхронном случае понятие цикла условно. Шаг (протокола) (step of a protocol, protocol action) конкретное законченное действие, выполняемое участником (протокола) во время одного цикла (прохода) протокола[1].

Коммуникационный протокол устанавливает последовательность действий участников при передаче информации или информационном обмене.

Обычный коммуникационный протокол обеспечивает установку соединения/сеанса, выбор маршрута, обнаружение искажений и восстановление передаваемой информации и т. п.

Безопасность протокола выражается в обеспечении гарантий выполнения таких свойств, характеризующих безопасность, как доступность, конфиденциальность, целостность и др. На системном языке говорят о функциях, выполняемых системой безопасности, либо о предоставляемых ею сервисах. Чтобы не путать с обычным математическим понятием функции, будем в подобных случаях говорить о функциях-сервисах.

Протокол, обеспечивающий поддержку хотя бы одной из функций-сервисов безопасности, называется защищенным или, точнее, протоколом обеспечения безопасности(security protocol). Защитные механизмы либо дополняют, либо встраиваются в коммуникационный протокол.[1]

Функции-сервисы безопасности. Понятие безопасности распределенных информационных систем в настоящее время конкретизируется как степень надежности реализации разнообразных функций (сервисов) безопасности. Указано также их желательное расположение в эталонной семиуровневой модели взаимодействия открытых систем.

Функция-сервис безопасности(security services) защитная функция, выполняемая подсистемой безопасности и определяемая ее целевым назначением.

В соответствии со стандартом ISO 7498.2 для архитектуры безопасности эталонной модели взаимодействия OSI выделено пять классов таких функций:

- аутентификация сторон и аутентификация источника данных;
 - разграничение доступа;
 - конфиденциальность;
 - целостность;
 - невозможность отказа от факта отправления/получения сообщения,
- которые могут конкретизироваться для конкретных условий применения.[1]

Определение. Криптографический протокол(cryptographic protocol) протокол, предназначенный для выполнения функций криптографической системы, в процессе выполнения которого участники используют криптографические алгоритмы.

Криптографическая система система обеспечения безопасности информации криптографическими методами.

Основными функциями криптографической системы являются обеспечение конфиденциальности, целостности, аутентификации, невозможности отказа и неотслеживаемости.

В качестве подсистем она может включать:

- системы шифрования;
- системы идентификации;
- системы имитозащиты;
- системы цифровой подписи и др., а также ключевую систему.

Для криптографических систем понятие стойкости хорошо известно их способность противостоять атакам противника и/или нарушителя, как правило, имеющим целью нейтрализацию одной или нескольких функций безопасности и, прежде всего, получение секретного ключа.

Пусть далее нарушитель внутренний нарушитель, т. е. участник протокола, нарушающий предписанные протоколом действия, а противниквнешний субъект (или коалиция субъектов), наблюдающий за передаваемыми сообщениями и, возможно, вмешивающийся в работу участников путем перехвата, искажения (модификации),вставки (создания новых), повтора и перенаправления сообщений, блокирования передачи и т. п. с целью нарушения одной или нескольких функций-сервисов безопасности. Часто допускается, что противник может образовывать коалицию с нарушителем[1]

Таблица 1 – Функции-сервисы безопасности

Функция-сервис	Назначение
Аутентификация источника данных	Обеспечивает возможность проверки того, что полученные данные действительно созданы конкретным источником. Данная функция не обеспечивает защиты от повторного навязывания или модификации данных
Аутентификация сторон	Обеспечивает возможность проверки того, что одна из сторон информационного взаимодействия действительно является той, за которую она себя выдает. Применяется с целью защиты от атаки типа имитации и от атаки на протокол с повторной передачей
Конфиденциальность данных	Обеспечивает невозможность несанкционированного получения доступа к данным или раскрытия данных.
Невозможность отказа	Обеспечивает невозможность отказа одной из сторон от факта участия в информационном обмене (полностью или в какой-либо его части)
Невозможность отказа с доказательством получения	Обеспечивает невозможность отказа одной из сторон от факта получения сообщения.

Продолжение таблицы 1

Функция-сервис	Назначение
Невозможность отказа с доказательством источника	Обеспечивает невозможность отказа одной из сторон от факта отправления сообщения.
Целостность данных	Обеспечивает возможность проверки того, что защищаемая информация не подверглась несанкционированной модификации и разрушению.
Обеспечение целостности соединения без восстановления	Обеспечивает возможность проверки того, что все данные, передаваемые при установленном соединении, не подверглись модификации, без восстановления этих данных.
Обеспечение целостности соединения с восстановлением	Обеспечивает возможность проверки того, что все данные, передаваемые при установленном соединении, не подверглись модификации, с восстановлением этих данных.
Разграничение доступа	Запрещает возможность несанкционированного использования ресурсов системы. На практике решение о предоставлении доступа основывается на аутентификации сторон.

1.2 Свойства, характеризующие безопасность протоколов

Криптографическая система может обеспечивать различные функции безопасности, для реализации которых применяются разнообразные криптографические протоколы.

Свойств, характеризующих безопасность криптографического протокола, также достаточно много. Обычно свойства протоколов, характеризующие их стойкость к различным атакам, формулируют как цели (goals) или требования к протоколам. Трактовка этих целей со временем меняется и уточняется. Наиболее полное и современное толкование этих целей дается в документах международной организации IETF.

Под свойствами безопасности в документах IETF в настоящее время понимаются следующие 20 целей, сгруппированные в 10 групп (см. таблицу 2).

Приведем определения перечисленных там свойств.

Таблица 2 – Свойства безопасности протоколов

№	Код	Название
1	G1	Аутентификация субъекта
	G2	Аутентификация сообщения
	G3	Защита от повторов
2	G4	Неявная (скрытая) аутентификация получателя
	G5	Аутентификация источника
3	G6	Авторизация (доверенной третьей стороной)
4	G7	Аутентификация ключа
	G8	Подтверждения правильности ключа
	G9	Защищенность от чтения назад
	G10	Формирование новых ключей
	G11	Защищенная возможность договориться о параметрах безопасности
5	G12	Конфиденциальность
6	G13	Обеспечение анонимности при прослушивании (несвязываемость)
	G14	Обеспечение анонимности при работе с другими участниками

Продолжение таблицы 2

№	Код	Название
7	G15	Ограниченная защищенность от атак типа отказ в обслуживании
8	G16	Неизменность отправителя
9	G17	Подотчетность
	G18	Доказательство отправки
	G19	Доказательство получения
10	G20	Безопасное временное свойство

1.3 Разбор свойств безопасности криптографических протоколов

(G1) Аутентификация объекта (аутентификация однорангового объекта): обеспечение одной стороны путем представления доказательств и / или удостоверений личности второго участника, участвующего в протоколе, и что вторая фактически участвовала во время выполнения текущего протокола. Обычно это делается путем представления части данных, которые могли быть созданы только второй стороной (например, ответом на вызов). Таким образом, обычно аутентификация объекта подразумевает, что некоторые данные могут быть однозначно прослежены до определенного объекта, что подразумевает аутентификацию источника данных.

(G2) Аутентификация сообщения (аутентификация источника данных): протокол должен обеспечивать средства для уверенности в том, что полученное сообщение или часть данных было создано определенной стороной в некотором (обычно неопределенном) времени в прошлом и что это Данные не были повреждены или не подделаны, но без предоставления уникальности или гарантии времени. Уверенность в том, что данные были созданы определенной стороной, но без уверенности, что она не была изменена, не представляет для нас интереса. Таким образом, аутентификация сообщения подразумевает целостность. Только очень немногие интернет-протоколы предлагают аутентификацию источника данных без предоставления аутентификации по объектам (IPsec AH или PKI Signatures будут примерами, как относительно

тривиальными для проверки). В списке протоколов-кандидатов для AVISPA нет протокола в этой категории.

(G3) Защита от повторного воспроизведения. Некоторые документы IETF определяют защиту воспроизведения как «Протокол должен обеспечивать средства для уверенности в том, что принятое сообщение не было записано и воспроизведено противником». Таким образом, это свойство не поддается проверке. Мы определяем его скорее как: Обеспечение одной стороны, что аутентифицированное сообщение не является старым. В зависимости от контекста это может иметь разные значения:

- что сообщение было создано во время этого сеанса;
- что сообщение было создано во время известного последнего временного окна;
- что сообщение ранее не принималось.

Аутентификация в многоадресной рассылке или через службу подписки/уведомления: это требования к аутентификации для групп с одним источником и очень большим количеством потенциальных получателей (многоадресной рассылки) или источника и службы, которые размещают информацию для подписки (и Авторизованных) пользователей. Основными требованиями для решения являются выполнение свойств G4 и G5.

(G4) Неявная аутентификация назначения Протокол должен обеспечивать средства для обеспечения того, чтобы отправленное сообщение было доступно для чтения только сущностями, которые разрешает отправитель. То есть, только законные уполномоченные члены будут иметь доступ к текущей информации, многоадресному сообщению или групповому общению. Сюда входят группы с очень динамичным членством.

(G5) Аутентификация источника. Законные члены группы смогут аутентифицировать источник и содержимое информации или групповое общение. Это включает случаи, когда члены группы не доверяют друг другу.

(G6) Авторизация (доверенной третьей стороной): В некоторых протоколах доверенная сторонняя сторона ТЗР вводит один главный В в

другого принципала А и А, что В «доверен» ТЗР и является ` Разрешено ", в необходимом смысле протокола. Когда протокол выполняется между А, В и ТЗР, тогда А, возможно, не может использовать списки управления доступом или другие механизмы для авторизации В (поскольку имя В неизвестно А или может быть даже псевдонимом) Но А уверен, что В разрешен ТЗР.

(G7). Ключевая аутентификация - это свойство, согласно которому одна сторона уверена, что никакая другая сторона, кроме специально идентифицированной второй стороны (и, возможно, дополнительных идентифицированных доверенных сторон), не сможет получить доступ к определенному секретному ключу.

(G8) Подтверждение ключа (Key Proof of Possession): одна сторона уверена, что вторая (возможно, неопознанная) сторона фактически владеет определенным секретным ключом (или всеми материалами, необходимыми для его расчета).

(G9) Perfect Forward Secrecy (PFS): протокол имеет это свойство, если компромисс долгосрочных ключей не ставит под угрозу прошлые сеансовые ключи.

(G10) Свежая деривация ключа В протоколе используется динамическое управление ключами для получения свежих ключей сеанса.

(G11) Согласование безопасных возможностей (устойчивость к понижающим и атакам на переговоры). Когда протокол соглашения о ключах также обнаруживает криптографические возможности и предпочтения сверстников и согласовывает параметры безопасности (такие как идентификаторы ассоциации безопасности, сила ключа и шифры), важно обеспечить, чтобы объявленные возможности и согласованные параметры не были подделаны Атакующим.

(G12) Конфиденциальность (секретность): свойство, что конкретный элемент данных или информация (обычно отправляемая или полученная как часть содержимого сообщения «защищена» или построенная на основе обменных данных) не является доступной или раскрытой для

неавторизованных лиц, организаций или процессов и остается неизвестной злоумышленнику. Мы выбираем соглашение о том, что секретность ключа сеанса, сгенерированного во время соглашения о ключах, здесь не рассматривается, а в разделе «Проверка ключа» выше. Также секретность долгосрочного ключа, используемого в протоколе, не рассматривается в качестве цели секретности протокола.

Анонимность. Многие протоколы не предоставляют анонимности, поскольку эксперт должен знать, с кем он говорит, чтобы определить, какой криптографический ключ использовать. Но некоторые протоколы позволяют скрыть действительное лицо.

(G13) Защита удостоверений от подслушивающих устройств: злоумышленник (тот, кто подслушивает) не должен связывать обмен информацией, обмениваемый одной стороной с реальной личностью участника.

(G14) Защита идентификатора от однорангового узла: одноранговое соединение в сообщении не должно связывать обмен информацией, обмен которой осуществляется одной стороной, с подлинной личностью стороны, а не несвязанным псевдонимом или частным идентификатором.

(G15) Соппротивление отказам в обслуживании (DoS). Трудно проверить сопротивление DoS. Одна из причин заключается в том, что протокол может подвергаться атакам DoS по разным причинам, наиболее распространенным из которых является то, что он потребляет слишком много ресурсов (память, вычислительная мощность), прежде чем эксперт аутентифицирует себя. Но есть много других причин: среди прочего, протоколы могут быть уязвимы для:

- DoS по распределению памяти;
- DoS на вычислительной мощности;
- атака с помощью третьих лиц (это побуждает один или несколько хостов отправлять большие суммы пакетов жертве). И, как известно, есть некоторые принципы, которые уже использовались, чтобы сделать протоколы более устойчивыми к отказу в обслуживании, могут быть формализованы.

(G16) Инвариантность отправителя. Сторона уверена, что источник сообщения остался таким же, как тот, который начал сообщение, хотя фактическая идентификация источника не важна для получателя.

Неотказание. Предотвращение неправильного отказа пользователя от выполнения действия.

(G17)Подотчетность: свойство системы (включая все ее системные ресурсы), которая гарантирует, что действия системного объекта могут быть однозначно отслежены для этого объекта, который может нести ответственность за его действия.

(G18) Доказательство происхождения. Неопровержимые доказательства того, что вы отправили сообщение.

(G19) Доказательство доставки Неопровержимые доказательства получения сообщения.

(G20). Временное свойство безопасности.[2]

2 Обзор методов анализа уязвимостей криптографических протоколов.

2.1 Алгебраические свойства, анализируемые при проверке корректности криптографических протоколов

Криптографические протоколы применяются во всех распределенных системах для обеспечения конфиденциальности, целостности, аутентификации, невозможности отказа от авторства и отслеживаемости. Существуют определенные требования, связанные с объектом их использования, предъявляемые к криптографическим протоколам

При анализе криптографических протоколов надлежит принимать во внимание все действия и возможности аналитика криптографического протокола, имеющиеся и возможные способы атак на криптографические протоколы, а также незащищенности, изначально присутствующие в самих криптографических протоколах, т.е. должна быть установлена гарантия корректности криптографического протокола. Корректность - это возможность криптографического протокола противостоять опасности со стороны противника и/или злоумышленника, как правило, имеющим целью нейтрализацию одной или нескольких функций безопасности, не имеющего необходимой секретной информации, но стремящегося выполнить протокол вместо участника, по определению владеющего такой информацией. [3]

В данный момент при создании и анализе большинства криптографических протоколов используются неформальные и интуитивные методы. При этом изучаются криптографические примитивы, применяемые в криптографических протоколах или алгебраические свойства криптографических протоколов. К основным алгебраическим свойствам криптографических протоколов относятся следующие:

- ассоциативность: $f(f(x, y), z) = f(x, f(y, z))$ или $(x+y)+z = x+(y+z)$;
- коммутативность: $\{\{x\}_y\}_z = \{\{x\}_z\}_y$ или $\{\{x\}_k\}_{k-1} = x$ and $\{\{x\}_{k-1}\}_k = x$;
- свойство исключающего ИЛИ (XOR), эта операция имеет следующие свойства:

- $x \otimes (y \otimes z) = (x \otimes y) \otimes z$ (ассоциативность), здесь $\otimes$ - операция XOR;
- $x \otimes y = y \otimes x$ (коммутативность);
- $x \otimes 0 = x$ (существование единичного элемента);
- $x \otimes x = 0$ (нилпотентность или коммутативность групповой операции);
- абелева группа, имеющая следующие свойства:
 - $x \times (y \times z) = (x \times y) \times z$ (ассоциативность), здесь $\times$ - бинарная мультипликативная операция;
 - $x \times y = y \times x$ (коммутативность);
 - $x \times 1 = x$ (существование единичного элемента);
 - $x \times x^{-1} = 1$ (инверсность);
- гомоморфизм: $\{[u, v]\}k = [\{u\}k, \{v\}k]$;
- свойство префикса – это свойство, обеспечивающее защиту от раскрытия префикса сообщения, например если $\{x, y\}_z$ это зашифрованный открытый текст (x – префикс, y – само сообщение), то $\{x\}_z$ – это зашифрованный префикс;
- абелева группа и модулярное возведение в степень:
 - $\text{exp}(\text{exp}(x, y), z) = \text{exp}(x, y \times z)$;
 - $\text{exp}(x, 1) = x$;
- временной штамп.

Анализ алгебраических свойств криптографического протокола необходим в том случае, когда:

- при атаке на криптографический протокол используются уязвимости криптографических примитивов, применяемых в криптографическом протоколе;
- криптографический протокол не имеет возможности функционировать без использования определенного криптографического примитива. [3]

Впрочем, анализ алгебраических свойств не дает гарантий, что криптографический протокол предоставляет остальные свойства криптографических протоколов, установленные в документах IETF и является корректным.

2.2 Проверка корректности криптографических протоколов с использованием формальных методов анализа

Для анализа корректности криптографических протоколов, в первую очередь необходимо представить протокол с применением какой-либо модели.

Для анализа свойств криптографических протоколов, сейчас часто используются формальные методы анализа криптографических протоколов. В формальных методах анализа криптографических протоколов применяется модель злоумышленника Далева-Яо, т.е. в этой модели злоумышленник идеализируется или подразумевается, что он обладает всеми возможностями для выполнения атаки на криптографический протокол и предполагается, что алгебраические примитивы, используемые в криптографическом протоколе не приводят к уязвимости криптографического протокола.

Формальные методы анализа криптографических протоколов дают возможность:

- определить точные границы между протоколом и средой его функционирования;
- указать точное функциональное поведение или поведение в реальном масштабе времени;
- установить точные необходимые свойства протокола;
- доказать реально ли протокол осуществляет возложенные задачи;
- выяснить условия, при которых протокол не имеет возможности реализовывать возложенные задачи.

Формальные методы анализа криптографических протоколов подразделяют на четыре вида:

- методы, созданные на модальной логике для анализа знаний и доверия (I вид);

- методы, основанные на экспертных системах, с применением которых создатель криптографических протоколов имеет возможность проследить за поведением протокола при различных схемах атак (II вид);

- методы, образованные на моделировании и анализе протоколов с использованием языков спецификаций и инструментариев анализа произведенных не для анализа криптографических протоколов (III вид);

- методы, созданные на формальных моделях, основывающихся на алгебраических и логических свойствах криптографических протоколов (алгебраическое моделирование) (IV вид).

Далее приведена классификация методов анализа корректности криптографических протоколов.

Условно, методы верификации корректности криптографических протоколов можно разделить на символические и алгебраические. К алгебраическим методам относятся две категории: на модальной логике для анализа знаний и доверия, на экспертных системах.[3]

Методы, работа которых основана на модальной логике:

- логика Каилара;
- логика Бьебера;
- логика Мао и Бойда;
- логика Мозера;
- логика Сиверсона;
- BAN-логика;
- GNY-логика.

К методам, работа которых основана на работе экспертных систем, относятся:

- ЭС Interrogator;
- системы, основанные на правилах.

Алгебраические же методы верификации корректности криптографических протоколов можно разделить на методы, основанные на

моделировании и анализе протоколов с использованием языков спецификаций и инструментов анализа, и на методы, работа которых основана на формальных моделях, базирующихся на алгебраических и логических свойствах криптографических протоколов.

Методы, основанные на моделировании и анализе протоколов с использованием языков спецификаций и инструментов анализа:

- формальный язык спецификации Ina Jo;
- конечные автоматы.

Методы, основанные на формальных моделях, базирующихся на алгебраических и логических свойствах криптографических протоколов:

- модель Меррита;
- метод Таусанта;
- KPL логика;
- модель Далева-Ляо;
- алгоритм Нарровера;
- анализатор протоколов NRL.

На данный момент, чаще всего используемыми методами анализа корректности криптографических протоколов, являются методы, базируемые на модальной логике, в частности самым результативным методом анализа криптографических протоколов на основе модальной логики является BAN логика (Burrows-Abadi-Needham logic) и его расширения (GNY логика, Логика Сиверсона и т.д.). Задачей использования модельной логики является вывод из определения протокола утверждение корректности протокола. Если не имеется возможности осуществить такой вывод в случае полноты логики это указывает на то, что протокол имеет уязвимость.

BAN логика предоставляет возможность получить ответы на следующие вопросы:

- необходимо ли для протокола больше допущений чем для других протоколов?

- существуют ли в протоколе нежелательные шаги?
- присутствует ли в протоколе шифрование сообщений, передача которых в открытом виде не влияет на безопасность протокола?
- каковы источники информации передаваемой в протоколе?
- новизна передаваемой в протоколе информации: первая («новая») или повторная передача?
- кому еще известна (или может быть известна) информация используемая в протоколе?

Последствием использования BAN логики для анализа криптографического протокола являются комбинация аксиом или логических умозаключений, доказывающих корректность криптографического протокола. Недостаток возможности совершения вывода означает существование уязвимости в протоколе. Ниже указаны примеры выводов по BAN логике.

Если Р верит, что К это ключ общий с Q И Р имеет $K[X]$ ТОГДА Р верит, что Q действительно отправил X.

Если Р верит, что сообщение X «новое» (т.е. не передано повторно) И Р верит, что сообщение X передано Q, ТОГДА Р верит, что Q верит в сообщение X.

Если Р верит, что X это «новое» сообщение, ТОГДА Р верит, что «X,Y» это новые сообщения.

Если Р верит, что Q доверяет сообщению X И если Р верит, что Q обладает сообщением X, ТОГДА Р доверяет сообщению X.

Исходя из существует ряд инструментальных программных средств анализа криптографических протоколов, но все они имеют специфицированные границы, т.е. либо не имеют права использования для анализа любых криптографических протоколов, либо имеют какие-то исключения или условия для предоставления заключений о стойкости установленного криптографического протокола. К данным средствам относится ЭС Interrogator, анализатор протоколов NRL и многие другие средства созданные для анализа определенных протоколов.

Наличие огромного количества формальных методов анализа криптографических протоколов и отсутствие единого суждения, какой из данных методов следует использовать и о результативности их употребления, на сегодняшний день, относится к одной из основных проблем.

Существует свод требований, которым надлежит следовать при создании криптографических протоколов:

- исход атаки на криптографический протокол должен обнаруживаться в ходе выполнения протокола;
- протокол должен разрабатываться для определенного уровня доверия протокола (Protocol Assurance Level);
- каждое сообщение, применяемое в протоколе должно иметь свое установленное назначение;
- если для благополучного воплощения протокола необходимо использовать данные о сторонах, то в протоколе в качестве протокольных сообщений следует использовать информацию, точно распознающую стороны;
- разработчик протокола должен быть уверен в целях осуществления шифрования в криптографическом протоколе, т.е. неверное применение шифрования в протоколе может привести к ошибкам.

Данные правила являются основными при разработке криптографических протоколов.[4]

3 Обзор средств автоматизированного анализа протоколов

3.1 Значение автоматизированного анализа протоколов

Установление причин уязвимостей и изучение атак на криптографические протоколы, которые реализуются благодаря им – отличная база для создания защищенных протоколов. Это позволяет избегать прошлые ошибки и синтезировать надежные, относительно предыдущих, протоколы.

Вопросы определения уровня безопасности протокола достаточно трудны и многообразны. Например, для тех протоколов, что считались на протяжении длительного времени вполне надежными, даже спустя десятилетие находились атаки, что выявляли уязвимость протоколов в определенных условиях. Все это становилось возможным благодаря тому, что предпринимались попытки формально обосновать свойства протокола и более определенной формулировки изначальных предположений, на основе которых изучают уровень безопасности протокола.

Рассмотрим средства автоматизированного анализа криптографических протоколов, которые в данный момент считаются самыми эффективными(, и рассмотрим математические аспекты работы этих систем). Упомянутые средства позволяют не только проверять требуемые свойства протоколов, но и находить конкретные атаки на протоколы в случае, когда эти свойства не выполняются.

3.1 AVISPA

AVISPA нацелена на разработку кнопочной, промышленной технологии для анализа широкомасштабных протоколов и приложений, чувствительных к безопасности в Интернете. Эта технология ускорит разработку следующего поколения сетевых протоколов, улучшит их безопасность и, следовательно, повысит общественное признание передовых, распределенных ИТ-приложений на их основе. В AVISPA предлагается достичь этого, продвигая технологию спецификации и дедукции до такой степени, что отраслевые протоколы могут быть указаны и автоматически проанализированы. Одной из центральных задач проекта является интеграция этой технологии в надежный автоматизированный

инструмент, ориентированный на практические масштабные проблемы и переносимый в органы стандартизации, разработчикам протоколов которых крайне нужны такие инструменты.

Цели:

- разработать богатый язык спецификаций для формализации протоколов, целей безопасности и моделей угроз промышленной сложности;
- продвигать современное состояние в автоматизированных методах вычета, чтобы увеличить эту сложность;
- создать инструмент на основе этих методов, который позволит организациям промышленности и стандартизации автоматически проверять или обнаруживать ошибки в своих продуктах;
- настроить этот инструмент и продемонстрировать эту концепцию в большом наборе практически соответствующих промышленных протоколов;
- начать миграцию этой технологии в такие организации стандартизации, как IETF, чтобы как научное, так и промышленное сообщество могло извлечь выгоду из достижений, достигнутых в рамках этого проекта.

AVISPA – это открытый проект RTD (FET open), финансируемый Европейской комиссией в рамках Программы технологий информационного общества

Проект AVISS (автоматизированная проверка систем с бесконечным состоянием) – это проект оценки, который проложил путь для AVISPA. Начатый в мае 2001 года и продолжавшийся один год, он оказался успешным, и он повысил интерес как к теме, так и в исследовательской группе. AVISPA является результатом последующего предложения по гранту, которое было принято летом 2002 года.[4]

3.2 Scyther

Инструмент предоставляет графический интерфейс пользователя, который дополняет интерфейсы командной строки и интерфейса Python. Графический интерфейс предназначен для пользователей, заинтересованных в проверке или понимании протокола. Интерфейсы командной строки и скриптов

облегчают использование Scyther для широкомасштабных проверок проверки протокола.

Scyther сочетает в себе ряд новых функций. Во-первых, Scyther гарантированно завершает работу, позволяя доказать правильность протоколов для неограниченного количества сеансов и необязательно выводить дерево доказательств (используя бэкэнд). В отличие от других неограниченных инструментов проверки, инструмент дает полезные результаты даже в том случае, если не обнаружена атака, но также не может быть установлена неограниченная корректность. В таких случаях результаты Scyther имеют аналогичную интерпретацию как ограниченные инструменты проверки, заявляя, что в пределах определенной привязки не существует атаки. Во-вторых, Scyther помогает в анализе протоколов, предоставляя классы поведения протоколов (или классов атак), а не только следы одиночной атаки, предоставляемые другими инструментами. В-третьих, Scyther облегчает так называемый многопротокольный анализ. При таком анализе анализируется параллельный состав двух (суб) протоколов. Инструмент можно использовать тремя способами:

- автоматически генерировать соответствующие требования безопасности для протокола и проверять их; Для анализа протокола путем формирования полной характеристики. Ниже описывается каждый из этих режимов;

- проверка требований. Язык ввода Scyther позволяет специфицировать свойства безопасности в терминах событий требования, то есть в спецификациях ролей можно утверждать, что определенное значение является конфиденциальным (секретность) или определенные свойства должны выполняться для партнеров связи (аутентификация). Scyther можно использовать для проверки этих свойств или фальсификации их.

- автоматические требования. Если спецификация протокола не содержит требований безопасности, Scyther может автоматически генерировать претензии. В конце каждой роли добавляются требования аутентификации,

утверждая, что предполагаемые партнеры по коммуникации должны были выполнить протокол, как ожидалось.[5]

3.3 ProVerif

ProVerif – это программный инструмент для автоматического анализа свойств безопасности, обнаруженных в криптографических протоколах. Инструмент был разработан Бруно Бланшетом.

Поддержка криптографических примитивов включает в себя: симметричную и асимметричную криптографию; Цифровые подписи; Хэш-функции; бит-обязательство; И сигнатурные доказательства знаний. Инструмент способен оценивать свойства достижимости, утверждения соответствия и наблюдательную эквивалентность. Эти возможности рассуждения особенно полезны для домена компьютерной безопасности, поскольку они позволяют анализировать свойства секретности и аутентификации. Также можно рассмотреть новые свойства, такие как конфиденциальность, прослеживаемость и проверяемость. Анализ протокола рассматривается в отношении неограниченного количества сеансов и неограниченного пространства сообщений. Инструмент способен восстанавливать атаки: когда свойство не может быть доказано, строится трассировка выполнения, которая фальсифицирует требуемое свойство.[6]

3.4 Isabelle

Isabelle – средство для автоматизированного логического вывода, основанное на логике предикатов первого порядка. При использовании Isabelle разработчик протокола должен построить логическую теорию и доказать в ее рамках различные утверждения. В силу теоретических ограничений данного подхода автоматическое доказательство невозможно. Однако отдельные фрагменты логического вывода можно выполнить автоматически. Основная трудность состоит в том, что пока определенное утверждение не доказано, нельзя сказать определенно, почему оно не доказано: потому, что оно не выводимо (и можно вывести его отрицание); или же потому, что еще не найден подходящий вывод. Это означает, что эффективность работы с Isabelle во

многое зависит от начальных предположений, которые разработчик протокола делает на основании опыта и интуиции. [7]

3.5 SPIN

SPIN обеспечивает эффективную проверку программного обеспечения, а не проверку оборудования. Инструмент поддерживает язык высокого уровня, чтобы указать описания систем, называемые PROMELA (PROcess MEta Language). Spin был использован для отслеживания ошибок логического проектирования в проектировании распределенных систем, таких как операционные системы, протоколы передачи данных, системы коммутации, параллельные алгоритмы, протоколы железнодорожной сигнализации и т. Д. Инструмент проверяет логическую согласованность спецификации. В нем сообщается о тупиках, неуказанных приемах, незавершенности флагов, условиях гонки и необоснованных предположениях относительно относительной скорости процессов.

Свойства корректности могут быть указаны как системные или технологические инварианты (с использованием утверждений), как линейные требования временной логики (LTL), как формальные Büchi Automata или более широко как общие омега-регулярные свойства в синтаксисе никогда не требует.

Инструмент поддерживает динамически растущее и сокращающееся количество процессов.

Также поддерживаются смешанные системы, использующие синхронную и асинхронную связь. Идентификаторы канала сообщений для randevu и буферизованных каналов могут передаваться из одного процесса в другой в сообщениях.

Инструмент поддерживает случайное, интерактивное и управляемое моделирование и как исчерпывающие, так и частичные методы доказательства, основанные на первом или первом поиске по ширине. Инструмент специально разработан, чтобы хорошо масштабироваться и эффективно обрабатывать даже очень большие размеры проблем. Чтобы оптимизировать прогон проверки,

инструмент использует эффективные методы сокращения частичного порядка и (необязательно) методы хранения, подобные BDD.

Как мы можем видеть, имеется целый спектр средств проверки корректности работы протокола, что подталкивает нас к определению четких свойств средств верификации, которые будут сравниваться между собой для каждой отдельной ситуации реализации безопасного протокола. Предлагается выбрать следующие атрибуты средств верификации:

- уровень автоматизации (чем больше степень автоматизации, тем лучше);
- простоту модели (ошибки возможны не только в протоколе, но и в его формальной модели, и вероятность таких ошибок возрастает с усложнением модели);
- гибкость в смысле выразительных свойств модели (чем меньше ограничений на моделируемые свойства, тем лучше);
- ограничения на модель в теоретико-вычислительном смысле (чем более сложные конфигурации можно промоделировать, тем лучше);
- ложные срабатывания при определении уязвимостей протокола (лучший вариант — без ложных срабатываний, приемлемый вариант — с небольшой вероятностью ложного обнаружения возможной атаки);
- извлечение алгоритма атаки (с точки зрения проектирования протокола 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). [8]

Таблица 3– Сравнительная оценка средств

Критерий	Murphy/SPIN	AVISPA	Isabelle	ProVerif	Scyther
Уровень автоматизации	Автоматическая	Автоматическая	Автоматизированная	Автоматическая	Автоматическая
Сложность модели	Высокая	Низкая	Высокая	Низкая	Низкая
Гибкость подхода	Да	Да	Да	Низкая	Низкая
Ложные срабатывания	Без оптимизации – нет, с оптимизацией – возможны	Нет	Возможность неопределенности в ходе доказательства	Возможны	Нет
Рамки	Значительные	Устанавливаются используемым подходом	Теоретические ограничения логики предикатов 1-го порядка	Строгий набор свойств, подлежащих верификации	Строгий набор свойств, подлежащих верификации
Установлен ли механизм атаки	Да	Да	Нет гарантий	Нет гарантий	Да
Сообщество разработчиков	Отсутствует	Да	Да	Отсутствует	Отсутствует

С учетом всех факторов и условий было принято решение реализовывать лабораторный практикум с использованием средства автоматизированного анализа криптографических протоколов AVISPA, основанное на работе языка спецификации протоколов высокого уровня HLPSL.

4 Обзор языка спецификации протоколов высокого уровня HLPSL

4.1 Понятие ролей и переходов

При описании протокола на языке HLPLS удобно использовать A-B нотацию (Alice-Bob), которая заключается в описании протокола в виде последовательности передаваемых сообщений, например:

$A \rightarrow B : A$

$B \rightarrow A : Nb$

$A \rightarrow B : Na, H(k(A, B), (Na, Nb, A))$

$B \rightarrow A : H(k(A, B), Na)$

В данном примере описана последовательность передачи сообщений протокола MS-CHAPv2 – протокола аутентификации между сервером и клиентом на основе пароля. В роли сервера выступает участник с идентификатором B, в роли клиента – участник с идентификатором A. $k(A, B)$ является паролем пользователя, на основе которого вычисляются коды аутентификации сообщений $H()$, передаваемых сервером и клиентом.

Эта нотация не позволяет описать сложные протоколы, содержащие условные переходы или потоки управления, однако позволяет перейти к более высокоуровневому описанию протокола на языке HLPLS.

Структура описания протокола на языке HLPLS.

Представление протокола на языке HLPLS состоит из описания ролей, целей и вызова главной роли.

Действия каждого участника протокола указываются в модуле, который называется основной ролью (basic role). Взаимодействие основных ролей осуществляется с помощью их объединения в композиционную роль (composed role). Первоначальное составление описания протокола на языке HLPLS в A-B нотации позволяет выделить основные роли, т.е. для каждого участника протокола будет выделена роль, в которой будет указана его последовательность действий. Таким образом, роль представляет собой спецификацию участника протокола, которая затем реализуется с помощью одного или нескольких агентов, выполняющих данную роль.[9]

Роль — это та функция, которую выполняет конкретный участник в конкретном сеансе, т.е. определяет порядок действий участника. В приведенном выше примере:

Участник А выполняет роль инициатора:

(1) $A \rightarrow ? : A$ %участник А передает свой идентификатор

(2) $A \leftarrow ? : Nb?$ %участник А получает нонс (? означает, что

%это значение может быть подменено злоумышленником)

(3) ...

Участник В выполняет роль ответчика:

(1) $? \rightarrow B : A?$ %участник В получает идентификатор

(2) $? \leftarrow B : Nb$ %участник В отправляет нонс

(3) ...

В описаниях протоколов на самом деле фигурируют не конкретные участники, а их роли.

Набор конкретных реализаций каждой из ролей протокола образует сеанс.

Наименование роли приводится строчными буквами, например: *alice*, *bob*, *server*, *session*, *initiator*. Для обозначения агента, выполняющего данную роль используется ключевое слово *played_by*.

Основная роль содержит следующие поля:

- параметры (*parameters*) – информация, используемая при инициализации участника протокола, включает заголовок роли, привязку роли к определенному агенту;

- начальное состояние (*initial state*);

- переходы (*transitions*) – для описания смены состояния роли в зависимости от событий или фактов.

Блок описания параметров по сути является блоком объявления переменных. Используются следующие типы переменных:

- агенты (`agent`) – для объявления переменных, обозначающих участников протоколов, по сути – идентификатор участника протокола. (При переходе к описанию на языке HLPSL этими переменными становятся используемые идентификаторы участников сторон из нотации A-B);

- ключи, используемые в протоколе:

- `symmetric_key` – тип данных для объявления переменной, являющейся ключом симметричной схемы шифрования;

- `public_key` – тип данных для объявления переменной, являющейся открытым ключом асимметричной схемы шифрования;

- хэш-функция (`hash_func`) – тип данных для объявления переменной, являющейся однонаправленной функцией; (`function`)

- канал (`channel(dy)`) – тип данных для объявления переменной для описания каналов, через которые будет выполняться взаимодействие участников протокола (обмен сообщениями). В скобках указывается тип канала, описывающий модель нарушителя, действующего в этом канале: `dy` означает, что модель нарушителя соответствует модели Долева-Яо (в HLPSL реализована только эта модель);

- текст (`text`) – тип данных для объявления переменной, являющейся произвольной битовой строкой (например, нонсом);

- сообщение (`message`) – тип данных для объявления переменной, являющейся битовой строкой. Используется для тех переменных, которые являются результатом вычисления, т.к. функции (например, хэш-функции) возвращают значение с этим типом данных;

- натуральное число (`nat`) – тип данных для объявления целочисленной переменной;

- логическое значение (`bool`) – тип данных для объявления логической переменной, принимающей значение 0 (ложь) или 1 (истина);

- набор данных (set) – тип данных для неупорядоченного набора значений стандартных типов данных.

Для объявления роли используется ключевое слово `role`, после чего следует название роли (начинается с прописной буквы) за которой в круглых скобках приводятся ее внешние параметры – список переменных с приведением их типа, разделяемые запятой. Объявление роли заканчивается ключевыми словами `end role`.

Ключевое слово `played_by` используется для обозначения агента, выполняющего роль. Ключевое слово `def=` означает начало блока объявления параметров роли и описания переходов.

Роль может содержать следующие объявления параметров:

- объявления локальных переменных с их типом;
- объявления констант, которые не являются локальными для роли и могут использоваться в других (являются глобальными);
- инициализация локальных переменных;
- объявление условий, при которых роль можно считать выполненной;
- объявление знаний нарушителя – информации, которая доступна нарушителю при начале выполнения роли.

При объявлении переменной сначала приводится ее название (с заглавной буквы), затем через `:` приводится ее тип. Запятая служит для разделения переменных одного типа друг от друга при перечислении и отделения переменных разных типов.

Объявление локальных переменных выполняется в секции, обозначаемой ключевым словом `local`, после которого с заглавной буквы приводится название переменной (или нескольких) с указанием типа. Следует учесть, что даже если переменная является локальной по отношению к роли, то переменные с таким же именем, объявленные в других ролях, должны иметь такой же тип.[9]

Объявление констант выполняется в секции, обозначаемой ключевым словом `const`, после которого со строчной буквы приводится список констант с указанием их типа. Константы являются глобальными, поэтому рекомендуется их объявлять в главной роли.

Секция инициализации переменных обозначается ключевым словом `init`. Символ `:=` используется для присвоения локальной переменной значения.

Вся информация о базовой роли (параметры и локальные переменные) представляет собой знание игрока роли.

Пример объявления роли:

```
role alice(A,B,S : agent,  
Kas : symmetric_key,  
SND, RCV : channel (dy))  
played_by A  
def=  
local  
State: nat,  
Kab: symmetric_key  
init State := 0  
transition  
...  
end role
```

В примере объявляется роль `alice`, при вызов которой будет осуществляться с параметрами: `A, B, S` – агенты, выполняющие в протоколе различные роли, `Kas` – симметричный ключ шифрования, `SND, RCV` – каналы, через которые участник протокола, играющий эту роль будет взаимодействовать. В секции `played_by` объявляется участник протокола, который будет выполнять эту роль. После ключевого слова `def=` выполняется объявление локальных переменных (натуральное число `State` и

симметричный ключ K_{ab}), затем в секции инициализации переменной присваивается значение 0. В секции `transition` затем описываются переходы роли. Заканчивается описание роли ключевыми словами `end role`.

Переходы в основной роли представляют собой *действия*, которые выполняются в случае истинности заданных в левой части выражения логических условий (предикатов), либо *реакции*, выполняющиеся немедленно при наступлении некоторых событий. Объявление переходов выполняется в секции, обозначаемой ключевым словом `transition`. Каждый переход может быть обозначен меткой (константа или натуральное число), отделенной от описания перехода точкой.[9]

Структура описания перехода:

метка. логические условия (предикаты) `--|>` действия
в случае использования действий, и
метка. событие `=|>` реакция
в случае использования реакций на события.

Описание реакций и предикатов идентичны, они отличаются только контекстом, в котором они используются.

Пример:

```
transition
step1.  State = 1 /\ RCV(start) =|>
State' := 2 /\ Na' := new() /\ SND({Na'}_K)
step2.  State = 2 /\ RCV({Nb'}_K) =|>
State' := 3 /\ SND({Nb'}_Hash(Na.Nb'))
```

Секция описания переходов начинается после ключевого слова `transition`. Константой `step1` обозначен первый переход, который представляет собой реакцию (`=|>`) на два события: Значение переменной `State` равно 1 И из канала `RCV` получено сообщение `start` (сообщение, которое посылается одному из участников протокола, после получения которого он запускает протокол – т.е. становится инициатором выполнения протокола). После одновременного наступления этих событий, происходит

следующая реакция: Значение переменной *State* устанавливается равном 2 И генерируется новое значение переменной *Na* И новое значение переменной *Na*, зашифрованное на ключе *K* передается в канал *SND*.

Второй переход, обозначенный константой *step2*, срабатывает при наступлении событий: значение переменной *State* равно 2 И из канала *RCV* получено сообщение, содержащее новое значение переменной *Nb*, зашифрованное на ключе *K*. После одновременного наступления этих событий, происходит следующая реакция: Значение переменной *State* устанавливается равном 3 И в канал *SND* передается полученное из канала *RCV* значение переменной *Nb*, зашифрованное на значении хэш-функции от конкатенации переменных *Na* и *Nb*.

Переходы могут содержать четыре предварительно определенных предикатов цели:

- *secret*(*E*, *id*, *S*) – объявление информации *E* общим секретом между агентами множества *S*, *id* является идентификатором цели;
- *witness*(*A*, *B*, *id*, *E*) – для слабой аутентификации агента *A* для агента *B* на основе информации *E*: агент *A* является обладателем информации *E*, *id* является идентификатором цели;
- *request*(*B*, *A*, *id*, *E*) – для сильной аутентификации – агент *B* запрашивает проверку *E*, *id* является идентификатором цели;
- *wrequest*(*B*, *A*, *id*, *E*) – аналогично *request*, но предназначено для слабой аутентификации.

Идентификатор *id* объявляется в роли верхнего уровня и используется для связи запросов с ответами на них. Идентификатор также используется в секции описания целей протокола.

Пример:

```
1.  State = 1 /\ RCV({Na'}_K) =|>
    State' := 3 /\ Nb' := new()
    /\ SND({Nb'}_K)
```

```

/\ K1' := Hash (Na' .Nb' )
/\ secret (K1' , k1 , {A,B})

```

В предикате цели `secret (K1' , k1 , {A,B})` объявляется, что информация `K1'` является общим секретом между участниками протокола А и В. Причем, объявление цели производится сразу после появления (вычисления) информации `K1'` в этой роли, поскольку после этого будет постоянно проводиться проверка сохранения этого значения в тайне на протяжении всего протокола.

```

2.  State = 2 /\ RCV ({Nb' }_K) =|>
State' := 4 /\ K1' := Hash (Na.Nb' )
/\ SND ({Nb' }_K1' )
/\ witness (A,B,bob_alice_nb,Nb' )

```

В данном примере после получения значения `Nb'` и отправления его, зашифрованном на новом ключе `K1'`, выполняется объявление цели аутентификации участника А на основе знания им информации `Nb'`. Объявление цели секретности `K1'` уже не выполняется, поскольку она уже была объявлена в другой роли. Идентификатор `bob_alice_nb` используется связи этого запроса аутентификации с проверкой, т.е. в цели `request (B,A,bob_alice_nb,Nb)` (объявленной в другой роли) будет выполняться проверка именно этого запроса.

Композиционная роль сочетает в себе другие роли, либо параллельно, либо последовательно. Она объявляется после описания основных ролей, которые описывают сеансы протокола. Композиционная роль создает один экземпляр каждой основной роли и описывает один сеанс протокола. Обычно эта роль называется `session`.

Композиционная роль не содержит секции переходов (`transitions`), а содержит секцию `composition`, в которой выполняется вызов других ролей. Оператор `/\` указывает, что эти роли должны выполняться параллельно.

В роли `session` обычно объявляются все каналы, используемые базовыми ролями.

Пример:

```
role session (A,B,S : agent,  
Kas, Kbs : symmetric_key)  
def=  
local  
SA, RA, SB, RB, SS, RS: channel (dy)  
composition  
alice (A, B, S, Kas, SA, RA)  
\ bob (B, A, S, Kbs, SB, RB)  
\ server(S, A, B, Kas, Kbs, SS, RS)  
end role
```

В примере параметрами композиционной роли являются агенты (A, B, S) и ключи (Kas, Kbs). В секции объявления переменных `local` объявляются каналы, через которые будут взаимодействовать роли. В секции `composition` параллельно запускаются роли `alice`, `bob` и `server`. При запуске ролей указываются агенты, причем при запуске роли `alice` первым параметром указывается агент A , который будет выполнять эту роль (поскольку при описании роли `alice` было указано `played_by A`). Другие агенты при запуске роли `alice` указываются в соответствии с описанием роли. Объявленные ранее в блоке `local` каналы также распределены по ролям.

Главной ролью является роль `environment`. Эта роль содержит глобальные константы и композицию из одного или нескольких сеансов (композиционной роли), где злоумышленник может играть определенные роли в качестве законного пользователя. Эта роль не содержит параметров. В теле роли объявляются константы, обозначающие агентов, ключи, функции, идентификаторы целей и т.п. Также в теле роли объявляются знания нарушителя (`intruder_knowledge`), где указываются имена всех агентов,

все открытые ключи, его личный ключ, все ключи, общие с другими агентами, и все общеизвестные функции. Для обозначения нарушителя используется зарезервированная константа `i`. Главная роль не содержит переходов, а содержит секцию `composition`, в которой параллельно запускается несколько сеансов роли `session`.

Пример:

```
role environment()
def=
const
  bob_alice_nb,k1 : protocol_id,
  kab,kai,kib : symmetric_key,
  a,b : agent,
  h : hash_func
  intruder_knowledge = {a,b,h,kai,kib}
  composition
    session(a,b,kab,h)
  /\ session(a,i,kai,h)
  /\ session(i,b,kib,h)
end role
```

В примере объявляются константы, описывающие всех участников протокола, ключи, хэш-функция и идентификаторы целей (`protocol_id`). Затем описываются знания нарушителя: агенты (`a` и `b`), хэш-функция (`h`), общие ключи злоумышленника с агентами `a` (`kai`) и `b` (`kib`). В секции `composition` выполняется запуск 3 сессий. Первая описывает нормальное функционирование протокола (`session(a,b,kab,h)`), во второй и третьей сессии злоумышленник выступает в роли одного из легальных участников протокола.

В последней секции спецификации после ключевого слова `goal` приводятся цели протокола. Цели бывают следующие:

- `secrecy_of` – обеспечение конфиденциальности некоторой информации;
- `authentication_on` – сильная аутентификация участников по некоторой информации;
- `weak_authentication_on` – слабая аутентификация участников по некоторой информации.

После указания цели приводится идентификатор цели, объявленный в главной роли.

Заканчивается описание целей ключевыми словами `end goal`.

Пример:

```
goal
secrecy_of n1b, k1ab
authentication_on bob_alice_nb
end goal
```

В примере объявлены две цели: конфиденциальность информации (сама информация будет указана в предикате цели `secret`) и сильная аутентификация. Идентификатор цели `bob_alice_nb` будет использоваться в предикатах цели `witness` и `request`. [9]

Описание завершается запуском главной роли:

```
environment()
```

4.2 Синтаксис языка HLPSL

Все переменные в HLPSL начинаются с заглавной буквы, а все константы начинаются со строчной буквы.

Символ `%` обозначает комментарий (до конца строки).

Зашифрование обозначается в виде фигурных скобок `{ }`, внутри которых помещается информация, подлежащая зашифрованию. Ключ, на котором будет зашифрована информация, приводится после скобок через нижнее подчеркивание:

```
{Pt}_Kab
```

В данном примере зашифровывается информация, обозначенная переменной P_t , ключом зашифрования является K_{ab} .

Логическое И обозначается символом $\wedge$.

X' обозначает новое значение переменной.

5 Разработка лабораторного стенда для проведения лабораторного практикума

Для успешного выполнения лабораторных работ потребуется виртуальная машина с установленной операционной системой Ubuntu 10, программный продукт AVISPA и аниматор протоколов безопасности SPAN, которые позволят выполнять следующие задачи:

- создавать модель криптографического протокола на языке высокого уровня;
- проверять работоспособность криптографического протокола, описанного на языке высокого уровня;
- проверять корректность выполнения свойства безопасности криптографического протокола;
- демонстрировать варианты атак на описанный криптографический протокол.

Аниматор протоколов безопасности SPAN по своей сущности является графической оболочкой программного средства автоматизированного анализа криптографических протоколов AVISPA.

Диаграмма разработанного лабораторного стенда приведена на рисунке 1.

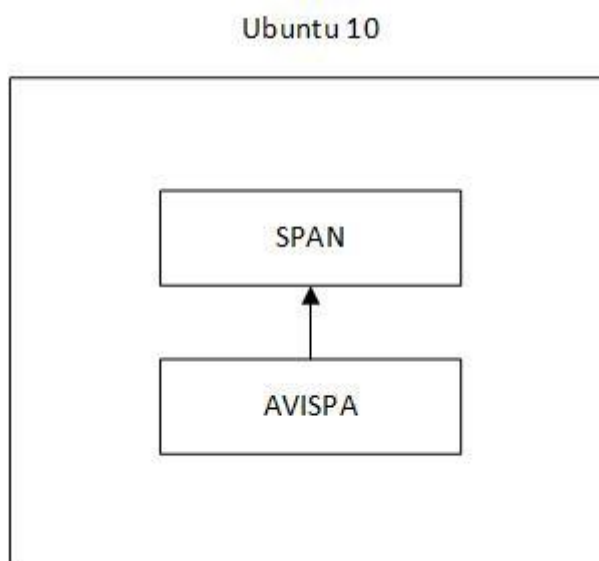


Рисунок 1 – Диаграмма компонентов лабораторного стенда

6 Реализация лабораторного практикума для средств AVISPA и SPAN

6.1 Ознакомление с языком описания криптографических протоколов HLPSL и инструментом анализа протоколов AVISPA.

В данной части практикума требуется, чтобы студент изучил предоставленные ниже сведения о языке HLPSL и произвел анализ протокола, описанного на языке HLPSL с помощью графического фронт-энда для AVISPA – «SPAN».

HLPSL – язык высокого уровня, описывающий криптографические протоколы.

AVISPA – инструмент, производящий анализ протоколов на предмет уязвимостей.

В языке HLPSL участники протокольного взаимодействия выполняют разные роли. Каждая роль описывается начальным набором параметров (состоянием) и способами изменения параметров (переходами). Название ролей пишется строчными буквами (например, *alice*), а участники протокольного взаимодействия (агентов) обозначаются прописными (например, *A*).

Пример определения роли на языке HLPSL:

```
role alice( A, B, S : agent,  
Kas : symmetric_key,  
SND, RCV : channel (dy) )  
played_by A def =  
local State: nat, Kab: symmetric_key  
init State := 0  
transition  
...  
end role
```

В описанной роли параметры *A*, *B*, *S* являются участниками протокольного взаимодействия, *Kas* – симметричный ключ, *SND* и *RCV* – каналы взаимодействия с выбором модели нарушителя, применяемой к этим каналам (в данном случае это модель носит название *dy*). Все переменные

языка HLPSPSL пишутся с большой буквы, а все константы — с маленькой. Все константы и переменные должны иметь тип.

Локальные переменные роли определяются в соответствующем разделе описания и определяются с использованием ключевого слова `local`, за которым следует их перечисление с указанием типов. Инициализация переменных происходит в разделе инициализации, начинающегося с ключевого слова `init`.

Описание переходов приводится в секции `transitions`. В общем случае, каждый переход показывает процесс приема сообщения и посылки ответа на него. Выполнение соответствующего действия определяется состоянием триггеров (переменных).

Пример описания перехода:

```
step1. State = 0 /\ RCV({Kab'}_Kas) =|>
State' := 2 /\ SND({Kab'}_Kbs)
```

В данном примере при получении сообщения из канала `RCV` и при состоянии `State`, равном 0, происходит изменение состояния триггера в значение 2 и в канал `SND` передается полученное сообщение `Kab'`, зашифрованное на новом ключе — `Kbs`.

Кроме простых ролей, HLPSPSL позволяет создавать составные роли. Например, составной ролью являются сессии, в которых описываются участники протокола и порядок их взаимодействия.

Роль высшего уровня или окружения (`environment`) является особенной и обязательной для описания в HLPSPSL. В ней определяются глобальные переменные и одну или несколько сессий. Здесь описываются участники взаимодействия, их ключи, а также знания нарушителя (агенты и ключи). Любой файл описания заканчивается инструкцией «вызова» роли высшего уровня `environment()`.

Далее необходимо начать работу с программой SPAN. Программа SPAN является графической надстройкой над программой AVISPA и служит для упрощения работы с ней.

Изображение окна программы показано на рисунке 2.

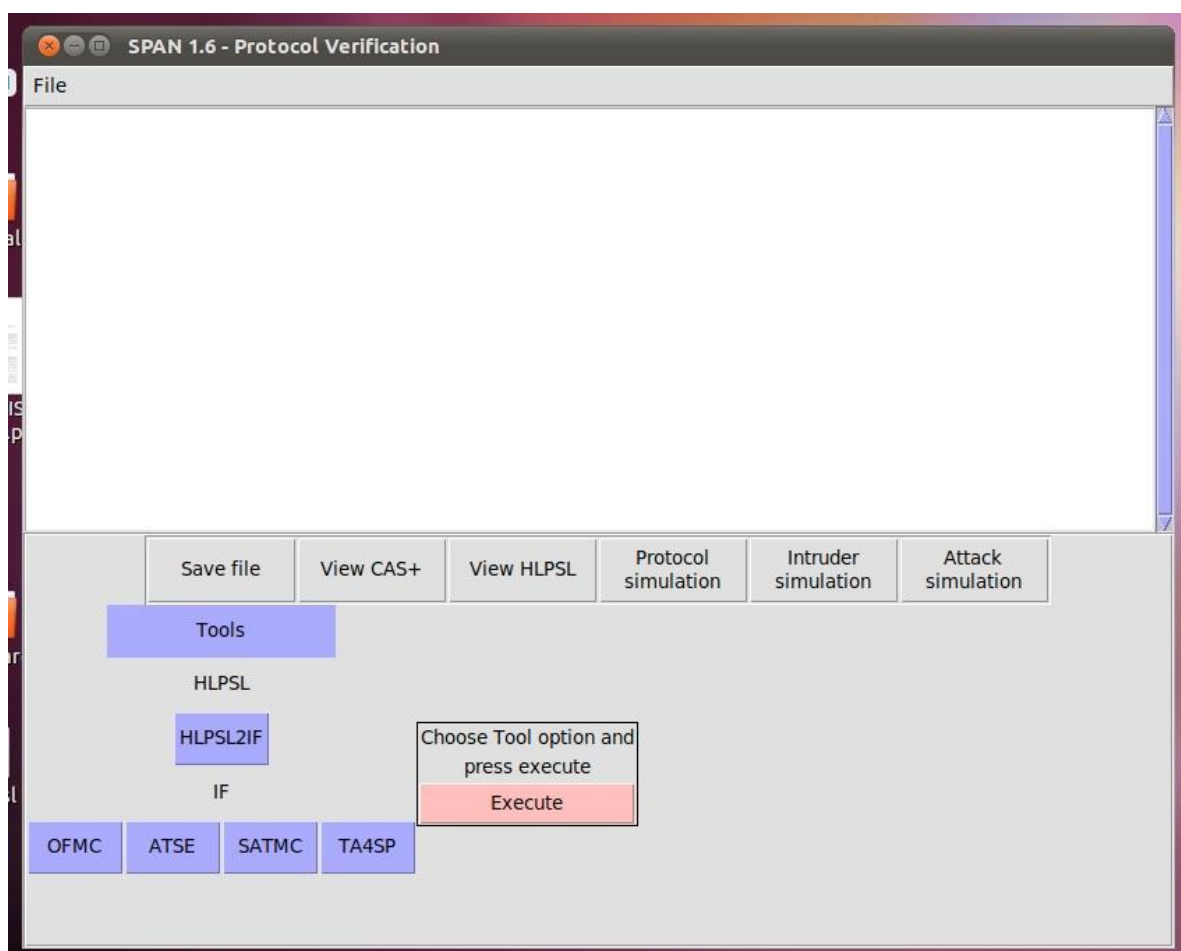


Рисунок 2 – Окно SPAN

На вход программы следует подать файл описания на языке HLPSP и запустить команду симуляции протокола (Protocol Simulation). С помощью данного инструмента можно проследить последовательность шагов протокола. В данном случае есть два участника обмена — А и В. Каждый из них поочередно может быть отправителем и получателем сообщения (для этого описаны роли initiator и responder). Участник А посылает участнику В сообщение m1, а участник В отвечает сообщением m2, как схематично показано на рисунке 3. Сообщения передаются в открытом виде.

Ожидаемые события и произошедшие события отображаются соответственно в списках Incoming events и Past events.

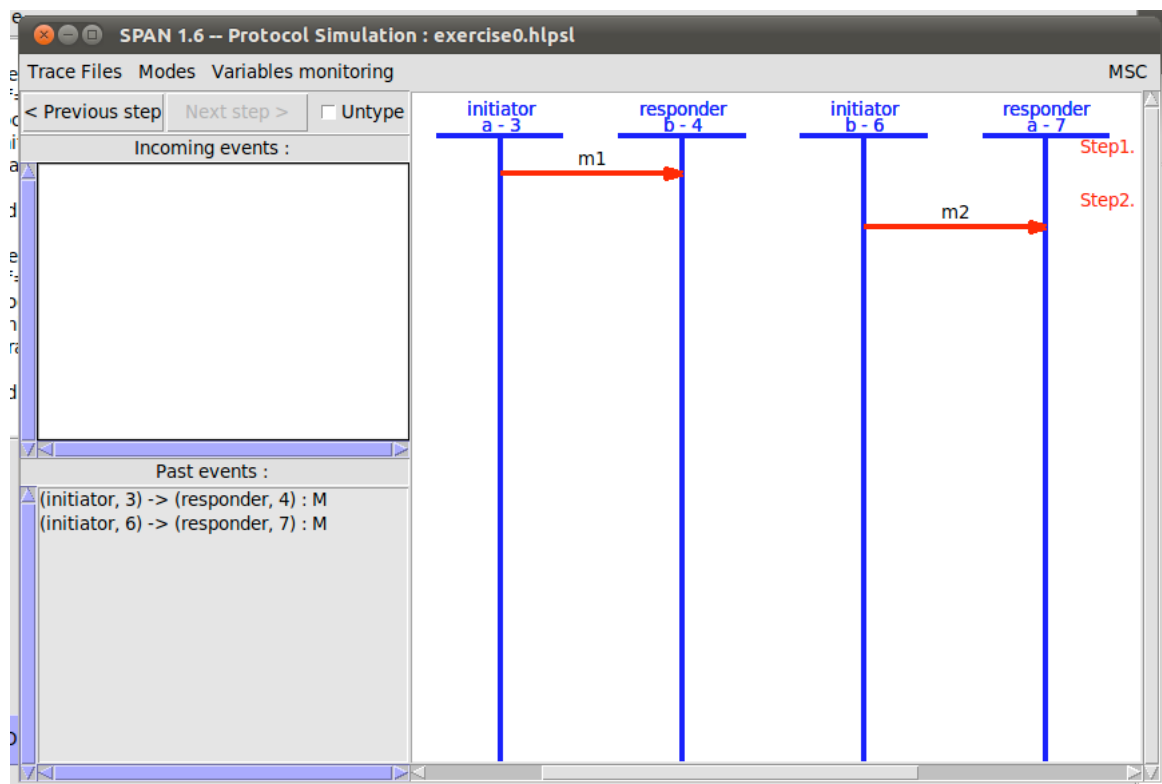


Рисунок 3 – Шаги протокола

Далее следует открыть другой файл описания — "exercise1.hlpst", где описано следующее взаимодействие:

Участник А посылает участнику В нонс, зашифрованный на секретном ключе между А и В. Участник В расшифровывает нонс и возвращает его А в открытом виде.

После чего требуется произвести проверку алгоритма путем нажатия на кнопку "Execute". Пример можно увидеть ниже:

```
% OFMC
% Version of 2006/02/13
SUMMARY
SAFE
DETAILS
BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
/home/span/span/testsuite/results/exercise1.if
GOAL
```

```

as_specified
BACKEND
OFMC
COMMENTS
STATISTICS
parseTime: 0.00s
searchTime: 0.00s
visitedNodes: 2 nodes
depth: 1 plies

```

Как видно из раздела SUMMARY, алгоритм признан безопасным, хотя на самом деле он таким не является. В описании перехода для роли участника В есть ошибка.

```

...
transition
1. State = 1 /\ RCV({Na}_Kab) =|> State' := 3 /\ SND
(Na)
...

```

Информация Na является новой для участника В и поэтому должна быть помечена апострофом. В программе это сделано не было, поэтому считается, что нонс известен участнику В. Для исправления этого необходимо поменять строку на следующую:

```

1. State = 1 /\ RCV({Na'}_Kab) =|> State':=3 /\
SND(Na')

```

После внесенных исправлений программа выдала следующий отчет:

```

% OFMC
% Version of 2006/02/13
SUMMARY
UNSAFE
DETAILS
ATTACK_FOUND

```

```

PROTOCOL
/home/span/span/testsuite/results/exercise1.if
GOAL
secrecy_of_sec1
BACKEND
OFMC
COMMENTS
STATISTICS
parseTime: 0.00s
searchTime: 0.00s
visitedNodes: 1 nodes
depth: 1 plies
ATTACK TRACE
i -> (a,3): start
(a,3) -> i: {Na(1)}_kab
i -> (b,3): {Na(1)}_kab
(b,3) -> i: Na(1)
i -> (i,17): Na(1)
i -> (i,17): Na(1)
% Reached State:
%
% secret(Na(1),sec1,set_41)
% contains(a,set_41)
% contains(b,set_41)
% state_role_B(b,a,kab,3,Na(1),3)
% state_role_A(a,b,kab,2,Na(1),set_41,3)

```

На рисунке 4 приведена симуляция атаки.

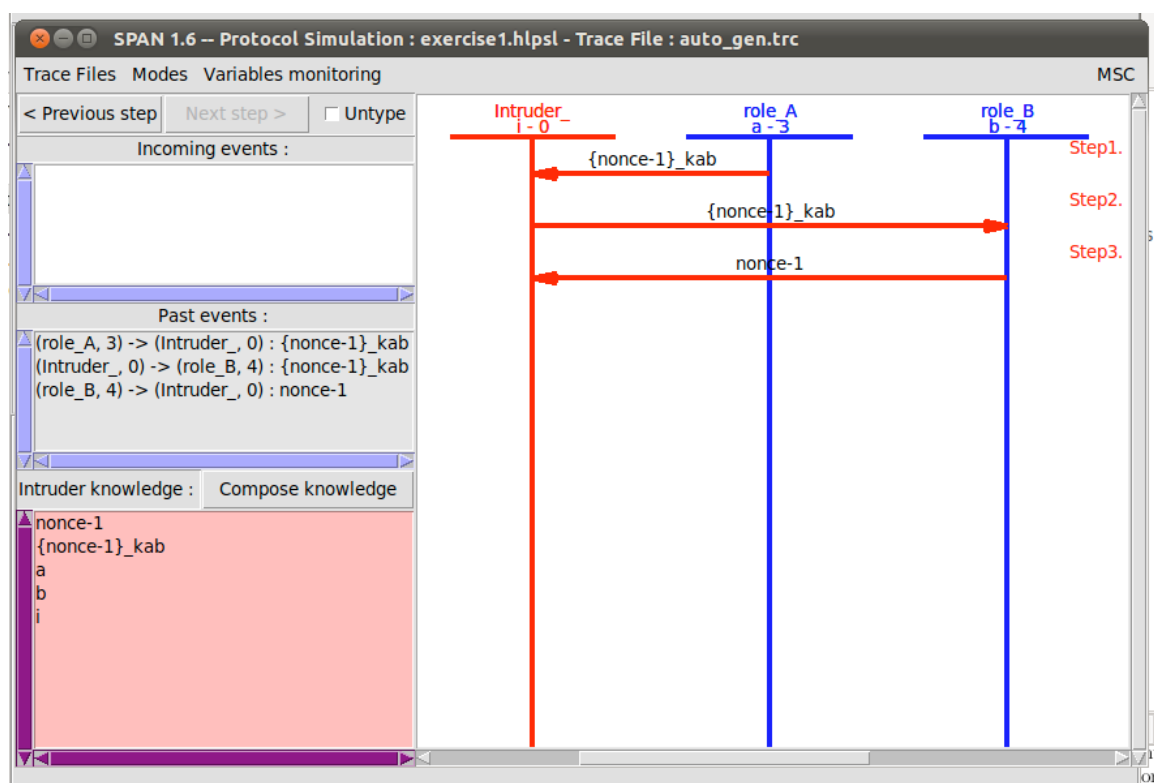


Рисунок 4 – Симуляция атаки

Далее студенту предлагается выполнить аналогичную работу, для закрепления навыка работы со средствами AVISPA и SPAN. По результатам выполнения должен быть представлен отчет в формате .pdf, сопровождаемый прикрепленными скриншотами окна программы SPAN, шагов протокола и симуляции атаки.

6.2 Реализация протокола обеспечения конфиденциальности и целостности передачи сообщения между двумя участниками на языке HLPSL и проверка его корректности

В данной части практикума студент изучает теоретический материал и знакомится с примерами реализации протокола, а также способами проверки его корректности.

Протокол обеспечения конфиденциальности и целостности сообщения имеет схему, показанную на рисунке 5. Участники А и В обладают общим секретным ключом и договорились использовать одну и ту же хеш-функцию.

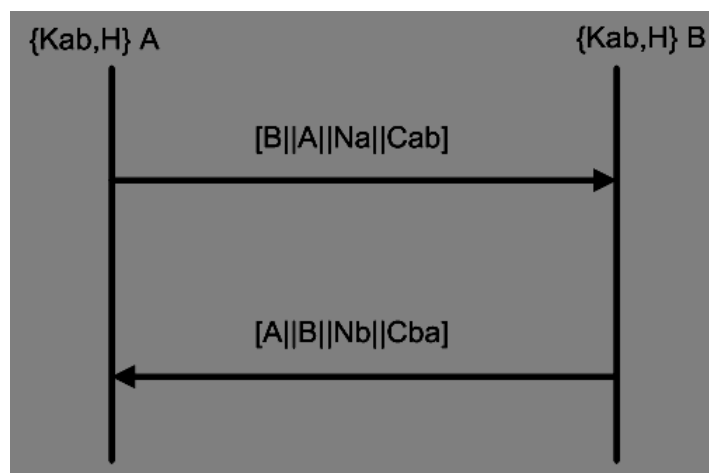


Рисунок 5 – Схема протокола

Алгоритм данного протокола представлен ниже:

1A КЗУ $\rightarrow$ Kab

ДСЧ $\rightarrow$ Na

$h_{ab} = H_{ab}(B||A||N_a||M)$

$C_{ab} = E_{ab}(M||h_{ab})$ — шифрование сообщения M и хеш-кода

$[B||A||N_a||C_{ab}] \rightarrow B$

2B Принимает $[B||A^*||N_a^*||C_{ab}^*]$

КЗУ $\rightarrow$ K_{ba}

$D_{ba^*}(C_{a^*b}) = [M^*||h_{ab}^*]$ — сообщение M и хеш-код расшифрованы

$h_{ba} = H_{ba^*}(B||A^*||N_a^*||M^*) \stackrel{?}{=} h_{ab}^*$ — сравнение хеш-кодов, т.е проверка

целостности сообщения M

ДСЧ $\rightarrow$ N_b

$h_{ba} = H_{ba^*}(A^*||B||N_b^*||h_{ab}^*)$

$C_{ba} = E_{ba^*}(h_{ab}^*||h_{ba})$

$[A^*||B||N_b||C_{ba}] \Rightarrow A^*$

3A Принимает $[A||B^*||N_b^*||C_{ba}^*]$

$D_{ab}(C_{ba}^*) = [h_{ab}^*||h_{ba}^*]$

Сравнивается возвращенное значение h_{ab}^* со сгенерированным на первом шаге абонентом A.

$H_{ab}(A||B||N_b^*||h_{ab}^*) \stackrel{?}{=} h_{ba}^*$

Делается вывод о том, что переданное сообщение было принято

правильно, целостность не нарушена. Цели протокола достигнуты.

Роль инициатора (участник А):

```
role sender( A,B: agent,
             Kab : symmetric_key,
             H : function,
             SND, RCV : channel(dy))
played_by A
def=
    local
        State: nat,
        Na: text,
        Nb: text,
        M : message,
        Hab : message,
        Hba : message

    init State := 0
    transition
    step1.  State = 0    /\ RCV(start) =|>
           State' := 1 /\ Na' := new()
                               /\ M' := new()
                               /\ secret(M',m,{A,B})
                               /\ Hab' := H(A.Na'.M')
                               /\ SND( A.Na'.{M.Hab'}_Kab
)
           step2.  State = 1 /\ RCV(
B.Nb'.{Hba'.Hab'}_Kab ) =|>
           State' := 2
```

Здесь участник А производит дальнейшие действия, результаты которых не отправляются в канал.

```

end role

Роль получателя (агент В):

role reciever(  B,A: agent,
               Kab : symmetric_key,
               H: function,
               SND, RCV: channel(dy))

played_by B
  def=
    local
      State: nat,
      Na: text,
      Nb: text,
      M : message,
      Hab : message,
      Hba : message

      init State := 0
      transition
        step1.  State = 0          /\
A'.Na'.{M'.Hab'}_Kab ) =|>
               State ':= 1  /\ Nb' := new()
                           /\ Hba' := H( B.Nb'.Hab' )
                           /\
                           SND(
B.Nb'.{Hba'.Hab'}_Kab )

    end role

role session (  A,B : agent,
               Kab : symmetric_key,
               H : function)

```

```

def=
local
SA, RA, SB, RB : channel (dy)
composition
    sender (A, B, Kab, H, SA, RA)
    /\ reciever (B, A, Kab, H, SB, RB)
end role

```

```

role environment()

```

```

def=

```

```

const

```

```

m : protocol_id,

```

```

kab,kai,kib: symmetric_key,

```

```

a,b: agent,

```

```

h: function

```

```

intruder_knowledge = {a,b,h,kai,kib}

```

Злоумышленник знает адреса участников обмена, хеш-функцию и имеет общие секретные ключи со всеми участниками

```

composition

```

```

    session(a,b,kab,h)

```

```

    /\ session(a,i,kai,h)

```

```

    /\ session(i,b,kib,h)

```

```

end role

```

```

goal

```

```

    secrecy_of m

```

Выше указана цель — обеспечение конфиденциальности M.

end goal

environment()

Результат анализа протокола:

% OFMC

% Version of 2006/02/13

SUMMARY

SAFE

DETAILS

BOUNDED_NUMBER_OF_SESSIONS

PROTOCOL

/home/span/span/testsuite/results/1.if

GOAL

as_specified

BACKEND

OFMC

COMMENTS

STATISTICS

parseTime: 0.00s

searchTime: 0.01s

visitedNodes: 24 nodes

depth: 5 plies

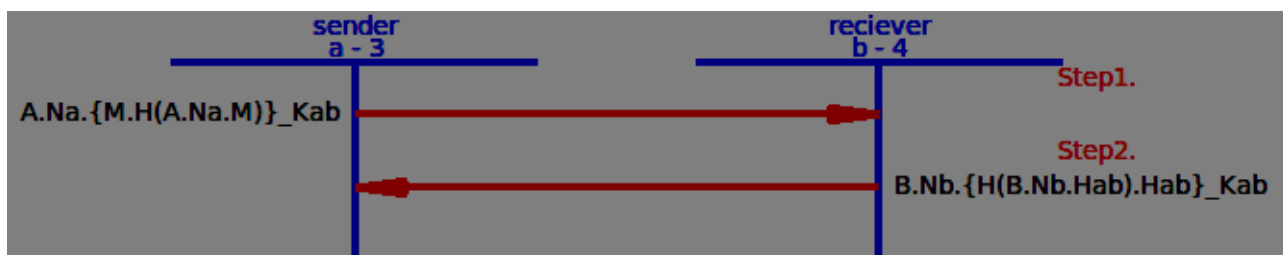


Рисунок 6 – Графическое представление передаваемых данных

Аутентификация не является целью данного протокола, поэтому тот случай, когда злоумышленник выдает себя за участника В, и пользуясь общим с участником А секретный ключ, получает сообщение М, система не распознала как уязвимость. Если же расширить знания злоумышленника, добавив к ним симметричный ключ Kab, то этот факт будет распознан как нарушение конфиденциальности передачи сообщения от А до В и анализ выдает следующий результат:

SUMMARY

UNSAFE

DETAILS

ATTACK_FOUND

TYPED_MODEL

PROTOCOL

home/span/span/testsuite/results/1.if

GOAL

Secrecy attack on (n1(M))

BACKEND

CL-AtSe

STATISTICS

Analysed : 0 states

Reachable : 0 states

Translation: 0.00 seconds

Computation: 0.00 seconds

ATTACK TRACE

i -> (a,3): start

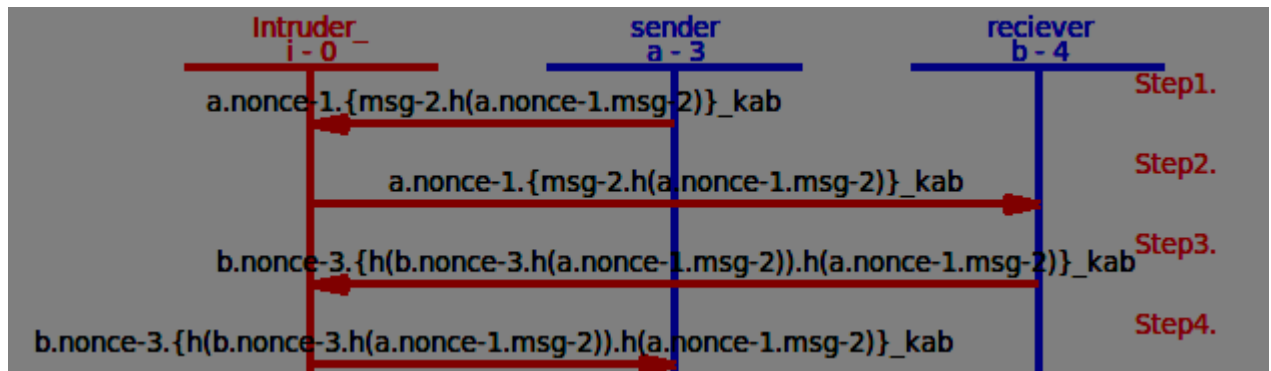


Рисунок 7 – Симуляция атаки (злоумышленник знает симметричный ключ Kab)

Злоумышленник здесь инициирует начала передачи сообщения, отправляя ему «start», а затем выступает посредником между участниками А и В, читая их сообщения.

После ознакомления с теорией и примерами реализации протокола студенту следует реализовать аналогичные по своей сути шаги и составить отчет в формате .pdf с прикреплением снимков экрана со схемой протокола, графическим изображением передаваемых данных и симуляцией атаки.

6.3 Реализация протокола явного ключевого аутентифицированного обмена с использованием языка спецификации протоколов высокого уровня HLPSL и инструмента анализа протоколов AVISPA

Реализуемый протокол состоит из следующих шагов:

1. А:

$KZU_a \rightarrow K_{ab}^1$

ДСЧ $\rightarrow N_a$

$h_{ab} = H_{ab}(B || A || N_a)$

$[B || A || N_a || h_{ab}] \Rightarrow B$

2. В:

принимает $[B || A' || N'_a || h'_{ab}]$

$KZU_b \rightarrow K_{ba}^1$

$H_{ba}'(B || A' || N'_a) \stackrel{?}{=} h'_{ab}$

$A' \cong A, N'_a \cong N_a$

ДСЧ $\rightarrow N_b, K_{ab}$

$$KZY_b \rightarrow K_{ba}^2,$$

$$K_{a',b}^* = E_{ba'}^2(K_{a',b})$$

$$h_{ba'} = H_{ba'}^1(A' || B || N'_a || N_b || K_{ab}^*)$$

$$[A' || B || N_b || K_{ab}^* || h_{ba'}] \Rightarrow A'$$

3. A:

$$\text{принимает } [A || B || N''_b || K''_{ab}^* || h''_{ba}]$$

$$H_{ba}^1(A || B || N_a || N''_b || K''_{ab}^*) \stackrel{?}{=} h''_{ba}$$

$$h''_{ba} \cong h_{ba}, N''_b \cong N_b, N'_a \cong N_a, K''_{ab}^* \cong K_{ab}^*$$

$$KZY_a \rightarrow K_{ab}^1$$

$$D_{ab}^2(K_{ab}^*) = K_{ab}$$

$$h_{ab1} = H_{ba}^1(B || A || N''_b || h''_{ba})$$

$$[B || A || h_{ab1}] \Rightarrow B$$

4. B:

$$\text{принимает } [B || A || h'_{ab1}]$$

$$H_{ba}^1(B || A || N_b || h_{ba}) \stackrel{?}{=} h'_{ab1}$$

ключ доставлен

Теперь следует ознакомиться с реализацией данной программы на языке HLPSL, код которой приведен ниже:

```

role alice(A,B: agent, Kab,K1:symmetric_key, Hash:
function, SND, RCV:channel(dy)) played_by A
def=
    local
        State :nat,
        NonsA,
        NonsB,
        Kabz      :text,
        Hab, Hba, Hab1 : message

    init
        State:=0

```

```

transition
1. State = 0 /\ RCV(start) =|>
    State' := 1 /\ NonsA' :=new()
        /\ Hab' :=Hash(B.A.NonsA'.K1)
        /\ SND(B.A.NonsA'.Hab')
        /\ witness(A,B,bob_alice_nb,NonsA')

2. State = 1 /\ RCV(A.B.NonsB'.{Kabz'}_Kab.Hba')
=|>
    State':=2 /\ Hab1':=
Hash(B.A.NonsB'.Hba'.K1)
        /\ SND(B.A.Hab1')
        /\ request(B,A,alice_bob_na,NonsB')
end role

role bob(A,B: agent, Kba,K1:symmetric_key, Hash:
function, SND, RCV:channel(dy)) played_by B
def=
    local
        State :nat,
        NonsA,
        NonsB,
        Kabz :text,
        Hab, Hba, Hab1, Habc : message
    init
        State:=0
    transition
1. State = 0 /\ RCV(B.A.NonsA'.Hab') =|>
    State' := 1 /\ Habc' := Hash(B.A.NonsA'.K1)
2. State = 1 /\ Hab = Habc =|>

```

```

        State' := 2 /\ NonsB' := new()
                /\ Kabz' := new()
                /\
Hba' := Hash(A.B.NonsA.NonsB'.{Kabz'}_Kba.K1)
                /\ SND(A.B.NonsB'.{Kabz'}_Kba.Hba')
                /\ secret(Kabz', kabz, {A,B})
                /\ request(B,A,bob_alice_nb,NonsA)
                /\ witness(A,B,alice_bob_na,NonsB')
3. State = 2 /\ RCV(B.A.Hab1') =|>
        State' := 3
end role

role session (A,B: agent, K,K1: symmetric_key, Hash:
function)
def=
    local SA,SB,RA,RB:channel (dy)
    composition
        alice(A,B,K,K1,Hash,SA,RA)
    /\
    bob(A,B,K,K1,Hash,SB,RB)
end role

role enviroment()
def=
    const
    a,b : agent,
    k,k1 : symmetric_key,
    kabz, bob_alice_nb, alice_bob_na: protocol_id,
    h: function

    intruder_knowledge = {a,b,h}

```

```

composition
    session(a,b,k,k1,h) /\ session(a,i,k,k1,h)
/\ session(i,b,k,k1,h)
end role

goal
    secrecy_of kabz
    authentication_on bob_alice_nb,  alice_bob_na
end goal

enviroment()

```

Результат симуляции протокола приведен на рисунках 8, 9, 10 и 11.

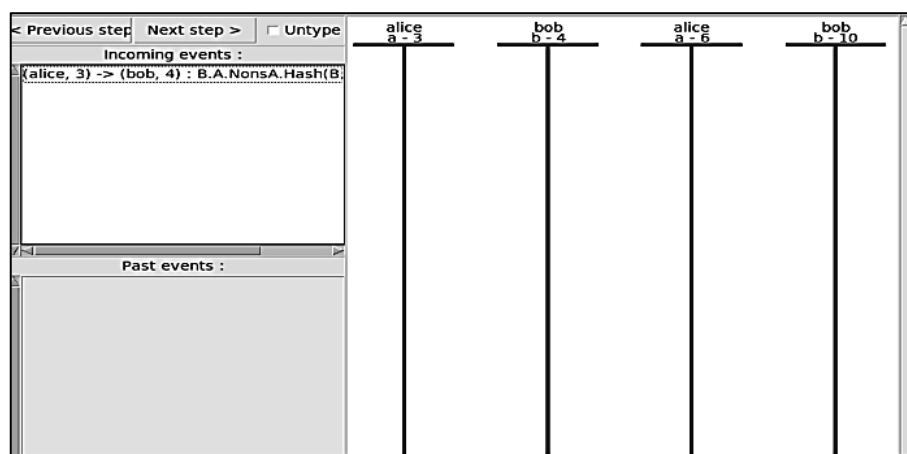


Рисунок 8 – Начальное состояние симуляции

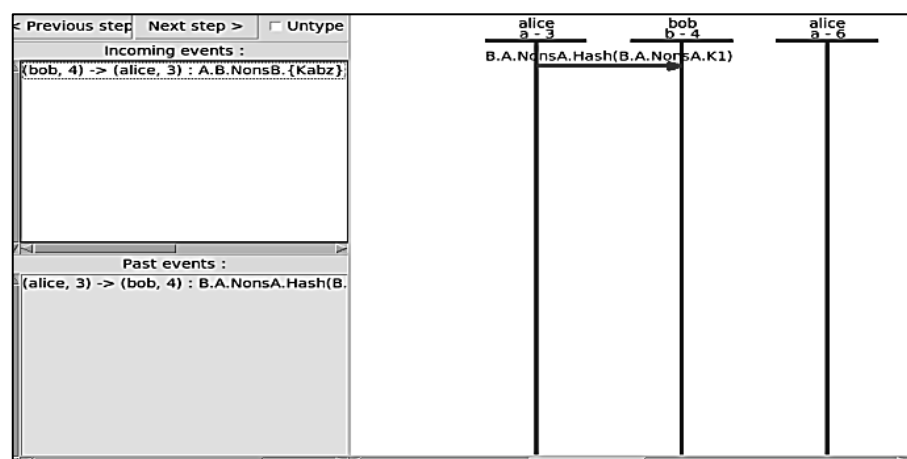


Рисунок 9 – Симуляция шага 1

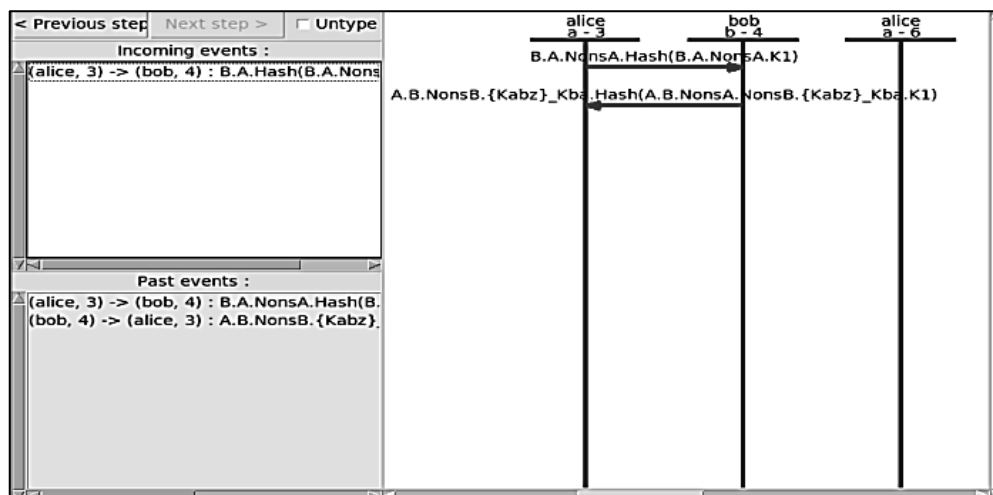


Рисунок 10 – Симуляция шага 2

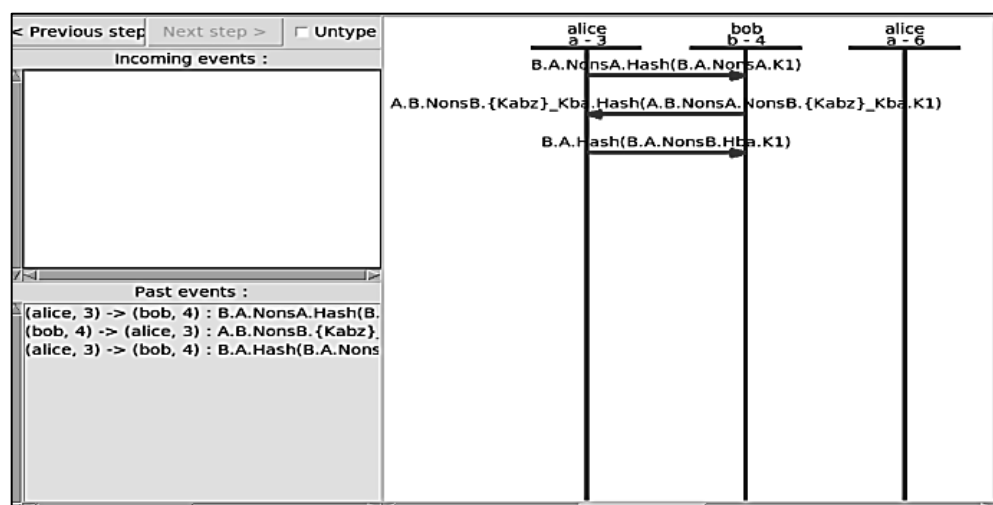


Рисунок 11 – Симуляция шага 3

Далее выполняется проверка стойкости протокола, путем симуляции атаки на протокол. Целью атаки является получение ключа K_{ab} . Злоумышленник знает адреса отправителя и получателя и хэш-функцию.

Результат поиска возможных атак:

% OFMC

% Version of 2006/02/13

SUMMARY

UNSAFE

DETAILS

ATTACK_FOUND

PROTOCOL

/home/span/span/testsuite/results/laba3.if

GOAL

authentication_on_alice_bob_na

BACKEND

OFMC

COMMENTS

STATISTICS

parseTime: 0.00s

searchTime: 0.01s

visitedNodes: 16 nodes

depth: 3 plies

ATTACK TRACE

i -> (a,3): start

(a,3) -> i: b.a.NonsA(1).h(b.a.NonsA(1).k1)

i -> (b,3): b.a.NonsA(1).h(b.a.NonsA(1).k1)

(b,3) ->

i:

a.b.NonsB(3).{Kabz(3)}_k.h(a.b.NonsA(1).NonsB(3).{Kabz(3)}_k.k1)

i -> (a,3): a.b.x275.{Kabz(3)}_k.x277

(a,3) -> i: b.a.h(b.a.x275.x277.k1)

На рисунке 12 показана реализация атаки на протокол.

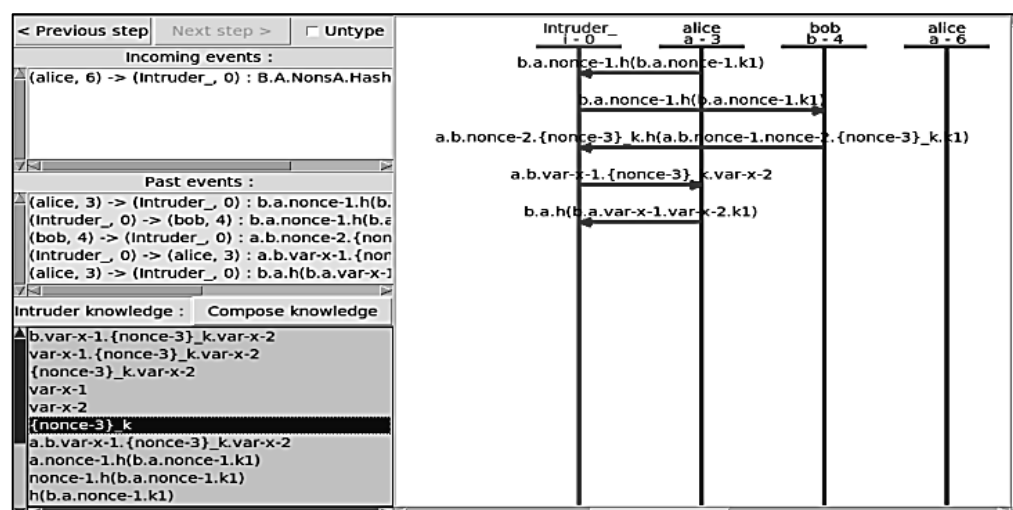


Рисунок 12 - Реализация атаки

По рисункам 11 и 12 видно, что был найден способ атаки на протокол, но результатом атаки стало нарушение аутентификации пользователя А.

Нарушитель встал между пользователем А и В и пересылал сообщения от пользователя к пользователю. Ключ K_{ab} нарушителем получен не был.

Далее проверяется стойкость протока при условии, что злоумышленнику известен еще и ключ хэш-функции. Результат поиска возможных атак:

Результат поиска возможных атак:

% OFMC

% Version of 2006/02/13

SUMMARY

UNSAFE

DETAILS

ATTACK_FOUND

PROTOCOL

/home/span/span/testsuite/results/laba3.if

GOAL

authentication_on_alice_bob_na

BACKEND

OFMC

COMMENTS

STATISTICS

parseTime: 0.00s

searchTime: 0.00s

visitedNodes: 1 nodes

depth: 1 plies

ATTACK TRACE

i -> (a,3): start

(a,3) -> i: b.a.NonsA(1).h(b.a.NonsA(1).k1)

i -> (a,3): a.b.x246.{x247}_k.x248

(a,3) -> i: b.a.h(b.a.x246.x248.k1)

На рисунке 13 показана реализация атаки на протокол.

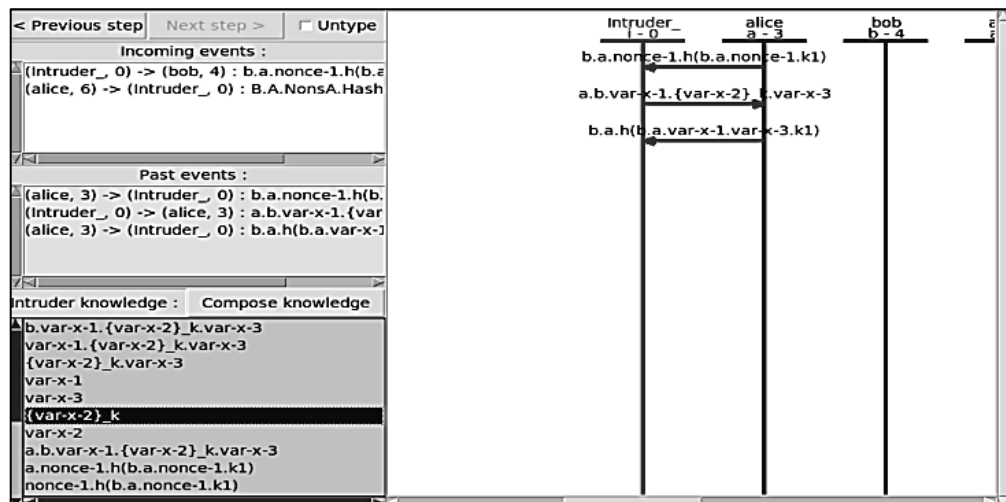


Рисунок 13 – Реализация атаки на протокол

По рисунку 13 видно, что был найден способ атаки на протокол, но результатом атаки стало нарушение аутентификации пользователя В. Нарушитель встал на место пользователя В и, зная ключ вычисления хэш-функции, смог выдать себя за пользователя В и навязать свой ключ K_{ab} пользователю А.

Далее проверяется стойкость протока при условии, что злоумышленнику известен еще и ключ хэш-функции. Результат поиска возможных атак показан далее

Результат поиска возможных атак:

%OFMC

%Version of 2006/02/13

SUMMARY

UNSAFE

DETAILS

ATTACK FOUND

PROTOCOL

/home/span/span/testsuite/results/laba3.if

GOAL

secrecy_of_kabz

BACKEND

OFMC

формате pdf, содержащий в себе снимки экрана с результатами симуляции протоколов (по шагам) и реализации атаки.

ЗАКЛЮЧЕНИЕ

В процессе выполнения задания на дипломное проектирование были решены следующие задачи:

- произведен обзор основных требований к безопасности криптографических протоколов;
- произведен обзор основных методов анализа криптографических протоколов;
- произведен обзор ряда средств автоматизированного анализа криптографических протоколов;
- разработан лабораторный стенд для проведения лабораторного практикума;
- разработан лабораторный практикум для практического закрепления навыков работы со средством автоматизированного анализа криптографических протоколов;
- разработаны рекомендации для проведения лабораторного практикума.

В результате разработан лабораторный практикум для применения процедур управления сертификатами ключей проверки электронной подписи, включающий лабораторный стенд, рекомендации для проведения практикума и комплект заданий.

В разработанном практикуме рассмотрены следующие вопросы:

- ознакомление со средством автоматизированного анализа криптографических протоколов;
- реализация протокола обеспечения конфиденциальности и целостности передачи сообщения между двумя участниками на языке HLPSL и проверка его корректности;
- реализация протокола явного ключевого аутентифицированного обмена.

Лабораторный стенд содержит виртуальную машину с установленной операционной системой Ubuntu 10, средством автоматизированного анализа криптографических протоколов AVISPA и аниматором протоколов безопасности для данного средства SPAN.

Таким образом, цель ВКР достигнута, требования технического задания на выпускную квалификационную работу выполнены в полном объеме.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1 Черемушкин А. В. Криптографические протоколы: основные свойства и уязвимости – [электронный ресурс]. – Режим доступа: <https://cyberleninka.ru/article/n/kriptograficheskie-protokoly-osnovnye-svoystva-i-uyazvimosti.pdf>, свободный. – Загл. с экрана.
- 2 Properties (Goals) [электронный ресурс]. – Режим доступа: <http://www.avispa-project.org/delivs/6.1/d6-1/node3.html>, свободный. – Загл. с экрана.
- 3 Методы анализа корректности криптографических протоколов [электронный ресурс]. – Режим доступа: <http://pandia.ru/text/79/313/20636.php>, свободный. – Загл. с экрана
- 4 This is AVISPA [электронный ресурс]. – Режим доступа: <http://www.avispa-project.org/theproject.html>, свободный. – Загл. с экрана
- 5 The Scyther Tool: Verification, Falsification, and Analysis of Security Protocols – Cas J. F. Cremes [электронный ресурс]. – Режим доступа: https://link.springer.com/chapter/10.1007/978-3-540-70545-1_38, свободный. – Загл. с экрана
- 6 ProVerif 1.96: Automatic Cryptographic Protocol Verifier, User Manual and Tutorial – Bruno Blanchet, Ben Smyth, Vincent Cheval, Marc Sylvestre [электронный ресурс] – Режим доступа: <http://prosecco.gforge.inria.fr/personal/bblanche/proverif/manual.pdf>, свободный. – Загл. с экрана
- 7 Перевышина Е. А. Методы и средства верификации криптографических протоколов – [электронный ресурс]. – Режим доступа: <https://www.scienceforum.ru/2017/2323/33304>, свободный. – Загл. с экрана
- 8 ON-THE-FLY, LTL MODEL CHECKING with SPIN [электронный ресурс]. – Режим доступа: <http://spinroot.com/spin/old.html>, свободный. – Загл. с экрана
- 9 HLPSL Tutorial [электронный ресурс]. – Режим доступа: <http://www.avispa-project.org/package/tutorial.pdf>, свободный. – Загл. с экрана