УДК 003.26(075.8)

КРИПТОГРАФИЧЕСКИЕ ПРОТОКОЛЫ: ОСНОВНЫЕ СВОЙСТВА И УЯЗВИМОСТИ

А. В. Черемушкин

Институт криптографии, связи и информатики, г. Москва, Россия

E-mail: avc238@mail.ru

В лекции рассматриваются основные понятия, связанные с криптографическими протоколами, определяются их основные свойства и уязвимости. Приводятся примеры атак на протоколы. Изложение сопровождается примерами, иллюстрирующими слабости некоторых известных протоколов. Приводится описание некоторых современных систем автоматизированного анализа протоколов.

Ключевые слова: *криптографический протокол, аутентификация.*

Вводные замечания

Целью настоящей лекции является привлечение внимания к проблематике, связанной с анализом протоколов. Первая ее часть посвящена ознакомлению с основными задачами, возникающими при анализе криптографических протоколов. Лекция имеет не столько обзорный, сколько вводный характер. Поэтому основное внимание уделено определению, примерам и основным свойствам протоколов. Понятие безопасности протокола является очень сложным и многогранным. Современный подход к определению свойств, характеризующих безопасность протокола, заключается в точной формулировке и проверке этих свойств. В современной литературе имеется много различных трактовок этих свойств. Поэтому за основу взят подход, принятый в ряде систем формального анализа и одобренный комитетом IETF, который определяет 20 таких свойств. Приводятся примеры наиболее известных атак на протоколы, демонстрирующие нарушение некоторых из заявленных свойств этих протоколов. Более подробно с этими вопросами можно познакомиться по учебному пособию [1].

В настоящее время имеется очень большое число различных подходов к анализу протоколов, причем для этого применяется разнообразный математический аппарат. В обзорной лекции не представляется возможным осветить хотя бы в краткой форме все многообразие применяемых идей. Поэтому во второй части лекции дается общая характеристика некоторых современных систем автоматического анализа протоколов и упоминаются методы, лежащие в их основе.

1. Основные понятия

Определение. *Протокол* (*protocol*) — описание распределенного алгоритма, в процессе выполнения которого два участника (или более) последовательно выполняют определенные действия и обмениваются сообщениями.

В качестве *участников* (иначе — субъектов, сторон) протокола могут выступать не только пользователи или абоненты, но и процессы, выполняющие какую-либо функциональную роль, например клиентские и серверные приложения.

Предполагается, что все участники выполняют в нем какую-либо активную роль, а пассивные наблюдатели не являются участниками протокола.

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. Например, шагами протокола могут быть:

— вычисление значения некоторой функции;

— проверка правильности сертификата ключа;

— генерация случайного числа;

— отправка сообщения и т. п.

Сеанс (*session*) — это конкретная реализация протокола с конкретными участниками.

Пример. Рассмотрим протокол конфиденциальнорго обмена.

Цикл 1:

Шаг 1: A формирует текстовую последовательность M_1.

Шаг 2: A вычисляет $S_1 = E_{k_{AB}}(M_1)$.

Шаг 3: A отправляет участнику B сообщение S_1.

Шаг 4: B получает сообщение S_1 и из заголовка узнает идентификатор отправителя A.

Шаг 5: B вычисляет текст $M_1 = D_{k_{AB}}(S_1)$.

Цикл 2:

Шаг 1: B формирует текстовую последовательность M_2.

Шаг 2: B вычисляет $S_2 = E_{k_{BA}}(M_2)$.

Шаг 3: B отправляет участнику A сообщение S_2.

Шаг 4: A получает сообщение S_2 и из заголовка узнает идентификатор отправителя B.

Шаг 5: A вычисляет текст $M_2 = D_{k_{BA}}(S_2)$.

Схематическая запись этого протокола имеет следующий вид:

$$(1)\ A \to B:\ E_{k_{AB}}(M_1),$$
$$(2)\ A \leftarrow B:\ E_{k_{BA}}(M_2).$$

В дальнейшем для краткости будем использовать в основном краткую схематическую запись протоколов. Она не заменяет полного описания, но по ней, как правило, можно восстановить описание, используя общие интуитивные соображения.

Заметим, что данный протокол использует традиционные криптографические механизмы — шифрование на общих симметричных ключах, причем ключи привязаны к направлению для защиты от возможности обратной отсылки переданного сообщения. Вместе с тем этот протокол не защищен от атаки повторением, при которой повторно передается ранее переданное сообщение от имени того же участника.

Введем еще одно важное понятие. *Роль* — это та функция, которую выполняет конкретный участник в конкретном сеансе. Роль определяет порядок действий участника. Набор конкретных реализаций каждой из ролей протокола образует сеанс. В предыдущем протоколе две роли:

— участник A выступает в роли *инициатора* сеанса протокола:

$$\begin{array}{lll} (1) & A \to ? & : \quad E_{k_{AB}}(M_1), \\ (2) & A \leftarrow ? & : \quad E_{k_{BA}}(M_2?); \end{array}$$

— участник B выступает в роли *ответчика*:

$$\begin{array}{lll} (1) & ? \to B & : \quad E_{k_{AB}}(M_1?), \\ (2) & ? \leftarrow B & : \quad E_{k_{BA}}(M_2). \end{array}$$

Вопросительный знак указывает те моменты, которые могут быть подменены при реализации конкретного сеанса и которые каждая из сторон должна проверить и получить подтверждение их аутентичности.

Следует заметить, что:

— в описаниях протоколов на самом деле фигурируют не конкретные участники, а их роли;

— в стенограммах (записях) сеансов фигурируют уже конкретные участники.

Если все участники честно выполняют все предписанные протоколом действия, то сеанс полностью повторяет описание протокола, отличаясь лишь тем, что в нем фигурируют имена и идентификаторы конкретных участников.

Если в работу протокола вмешивается противник либо некоторые из участников пытаются одновременно выступать в нескольких ролях, то в результате могут получаться такие комбинации действий, при которых происходит нарушение одного или нескольких свойств протокола, что приводит к реализации сценария атаки на протокол.

2. Понятие криптографического протокола

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.

Функции-сервисы безопасности. Понятие безопасности распределенных информационных систем в настоящее время конкретизируется как степень надежности реализации разнообразных функций (сервисов) безопасности. Впервые в наиболее полном виде концепция функций-сервисов безопасности изложена в международном стандарте: «*Базовая эталонная модель взаимодействия открытых систем. Часть 2: Архитектура безопасности*», утвержденном в 1989 г. В 1991 г. этот стандарт был повторен в «*Рекомендации X.800: Архитектура безопасности взаимодействия открытых систем, для применений МККТТ*». Он содержит описание основных (базовых) *функций-сервисов безопасности* для случая взаимодействия двух систем, а также основных *механизмов*, обеспечивающих эти услуги, включая криптографические средства. 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. Их определения содержатся в табл. 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 *противник* — внешний 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.

Таблица 1

Функции-сервисы безопасности

Функция-сервис	Назначение
Аутентификация источника данных (data origin authentication service)	Обеспечивает возможность проверки того, что полученные данные действительно созданы конкретным источником. Данная функция не обеспечивает защиты от повторного навязывания или модификации данных
Аутентификация сторон (peer entity authentication service)	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 имитация и от атаки на протокол с повторной передачей
Конфиденциальность данных (data confidentiality service)	Обеспечивает невозможность несанкционированного получения доступа к данным или раскрытия данных
Невозможность отказа (non-repudiation service)	Обеспечивает невозможность отказа одной из сторон от факта участия в информационном обмене (полностью или в какой-либо его части)
Невозможность отказа с доказательством получения (non-repudiation service with proof of delivery)	Обеспечивает невозможность отказа получателя от факта получения сообщения
Невозможность отказа с доказательством источника (non-repudiation service with proof of origin)	Обеспечивает невозможность отказа одной из сторон от факта отправления сообщения
Целостность данных (data integrity service)	Обеспечивает возможность проверки того, что защищаемая информация не подверглась несанкционированной модификации или разрушению
Обеспечение целостности соединения без восстановления (connection integrity service without recovery)	Обеспечивает возможность проверки того, что все данные, передаваемые при установленном соединении, не подверглись модификации, без восстановления этих данных
Обеспечение целостности соединения с восстановлением (connection integrity service with recovery)	Обеспечивает возможность проверки того, что все данные, передаваемые при установленном соединении, не подверглись модификации, с восстановлением этих данных
Разграничение доступа (access control service)	Обеспечивает невозможность несанкционированного использования ресурсов системы. Данный термин понимается в самом широком смысле. На практике решение о предоставлении доступа основывается на аутентификации сторон

Заметим, что во многих формальных методах анализа протоколов противник отождествляется с сетью.

3. 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 (целями, требованиями) безопасности в документах IETF в настоящее время понимаются следующие 20 целей, сгруппированные в 10 групп (табл. 2). Приведем определения некоторых из перечисленных там свойств.

Таблица 2

Свойства безопасности протоколов

№	Код	Название
1	G1	Аутентификация субъекта
	G2	Аутентификация сообщения
	G3	Защита от повтора
2	G4	Неявная (скрытая) аутентификация получателя
	G5	Аутентификация источника
3	G6	Авторизация (доверенной третьей стороной)
4	G7	Аутентификация ключа
	G8	Подтверждение 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
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

(G1) Аутентификация субъекта (Аутентификация сторон, Peer Entity Authentication) — проверка с подтверждением подлинности одной из сторон наличия или полномочий (посредством представленных доказательств и/или документов) идентичности второй стороны, участвующей в выполнении протокола, а также того, что она действительно принимает участие в выполнении текущего сеанса протокола.

Обычно она осуществляется посредством набора данных, который мог быть сгенерирован только вторым участником (как отклик на запрос, например).

Таким образом, обычно аутентификация субъекта предполагает, что некоторые данные могут быть безошибочно возвращены некоторому субъекту, что предполагает аутентификацию источника данных (Data Origin Authentication).

(G2) Аутентификация сообщения (Message authentication) — обеспечение аутентификации источника данных и целостности передаваемого сообщения.

Аутентификация источника данных (Data Origin Authentication) означает, что протокол должен обеспечивать средства гарантии того, что полученное сообщение или часть данных были созданы некоторым участником в некоторый (как правило, неопределенный) момент времени, предшествующий получению сообщения, и что эти данные не были искажены или подделаны, но без предоставления гарантий однозначности и своевременности.

Поскольку уверенность в том, что данные были созданы некоторым участником, без гарантии того, что они не были модифицированы, не представляет практического интереса, то обычно полагают, что требование аутентификации сообщения влечет требование его целостности.

(G3) Защита от повтора (Replay Protection) — гарантирование одним участником того, что аутентифицированное сообщение не является старым.

В зависимости от контекста, это может иметь разный смысл:

— сообщение было сгенерировано в данном сеансе протокола;

— сообщение было сгенерировано в течение известного промежутка времени;

— сообщение не было принято ранее.

(G5) Аутентификация источника (Source Authentication) — законные группы участников должны быть способны аутентифицировать источник и содержание информации или групповой коммуникации.

Это относится к случаям, когда группы участников не доверяют друг другу.

(G7) Аутентификация ключа (Key Authentication) — это свойство предполагает, что один из участников получает подтверждение того, что никакой другой участник, кроме заранее определенного второго участника (и, возможно, других доверенных участников), не может получить доступа ни к одному секретному ключу.

(G8) Подтверждение правильности ключа (Key Confirmation, Key Proof of Possession) — один из участников получает подтверждение того, что второй участник (возможно, неопределенный) действительно обладает конкретным секретным ключом (либо имеет доступ ко всем ключевым материалам, необходимым для его вычисления).

(G9) Защищенность от чтения назад / Совершенная секретность в будущем (Perfect Forward Secrecy, PFS) — протокол обладает этим свойством, если компрометация долговременных ключей не приводит к компрометации старых сеансовых ключей.

(G12) Конфиденциальность (Confidentiality, Secrecy) — свойство, состоящее в том, что специфический набор данных (обычно посылаемый или полученный как часть «защищенного» сообщения, а также сформированный на основе данных, полученных в результате обмена) не станет доступным или раскрытым для неавторизованных субъектов или процессов, а останется неизвестным противнику.

Мы принимаем соглашение, что секретность сеансового ключа, сгенерированного в результате процедуры открытого распределения ключей, рассматривается не здесь, а в п. «Аутентификация ключа» (см. выше).

Заметим, что секретность долговременного ключа, используемого в протоколе, не рассматривается как целевое свойство безопасности протокола, а относится к исход-

ным предположениям. В табл. 3 указаны свойства безопасности некоторых протоколов.

Таблица 3

Примеры свойств безопасности, характеризующих протоколы

Протокол \ Цель G	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
EAP-IKEv2	×	×	×			×	×			×					×
EKE	×	×										×			
IKE	×	×	×				×		×	×	×		×	×	×
IKEv2	×	×	×				×		×	×	×				×
DHCP-IPSec-tunnel	×	×										×			
kerberos	×	×	×			×	×			×					
SSH	×	×	×				×			×	×				
TLS	×	×	×				×			×	×		×		
TLS-v1.1	×	×	×				×			×	×		×		
TLS-SRP	×	×	×				×			×	×		×		
tls-sharedkeys	×	×	×				×			×	×		×		
SET	×	×	×										×		

4. Основные атаки на безопасность протоколов

Определение. Под *атакой на протокол* понимается попытка проведения анализа сообщений протокола и/или выполнения не предусмотренных протоколом действий с целью нарушения работы протокола и/или получения информации, составляющей секрет его участников.

Атака считается *успешной*, если нарушено хотя бы одно из заявленных свойств, характеризующих безопасность протокола.

В основе атак могут лежать различные *методы анализа протоколов*.

Пример. Рассмотрим протокол передачи секретного ключа k от A к B без использования какой-либо общей секретной информации. Его названия: «бесключевой» протокол А. Шамира, либо — трехпроходный протокол Шамира – Ривеста – Адлемана:

$$\begin{aligned}
&(1)\quad A \to B:\ E_{k_A}(k),\\
&(2)\quad A \leftarrow B:\ E_{k_B}(E_{k_A}(k)),\\
&(3)\quad A \to B:\ D_{k_A}(E_{k_B}(E_{k_A}(k))) = E_{k_B}(k).
\end{aligned}$$

Здесь E — коммутирующее шифрующее преобразование

$$E_{k_1}(E_{k_2}(x)) = E_{k_2}(E_{k_1}(x))$$

при всех сообщениях x и произвольных ключах k_1 и k_2. Заметим, что в этом протоколе можно использовать не каждое коммутирующее преобразование E. Например, для

$$E_{k_A}(k) = k \oplus \Gamma_A$$

протокол оказывается заведомо нестойким:

$$\begin{aligned}
&(1)\quad A \to B:\ E_{k_A}(k) = k \oplus \Gamma_A,\\
&(2)\quad A \leftarrow B:\ E_{k_B}(E_{k_A}(k)) = k \oplus \Gamma_A \oplus \Gamma_B,\\
&(3)\quad A \to B:\ D_{k_A}(E_{k_B}(E_{k_A}(k))) = E_{k_B}(k) = k \oplus \Gamma_B.
\end{aligned}$$

Поэтому в протоколе Шамира рекомендуется использовать преобразование вида

$$E_{k_A}(k) = k^a \bmod p,$$

в котором константа a определяется ключом k_A, а p — большое простое число.

Укажем некоторые слабости и приведем примеры атак на этот протокол.

Слабость 1: отсутствует аутентификация сторон.

Можно применить атаку путем *подмены* участника.

Атака 1: (*impersonation attack*)

$$\begin{array}{ll} (1) & A \to C(B):\ E_{k_A}(k), \\ (2) & A \leftarrow C(B):\ E_{k_C}(E_{k_A}(k)), \\ (3) & A \to C(B):\ E_{k_C}(k). \end{array}$$

Противник C блокирует передачу к B, выступая от имени участника B. В результате C получил от A ключ k для связи с ним от имени B.

Слабость 2: сообщения участников симметричны.

Поэтому протокол не является стойким и к *атаке повторением*.

Атака 2: (*replay attack*)

$$\begin{array}{ll} (1) & A \to C(B):\ E_{k_A}(k), \\ (2) & A \leftarrow C(B):\ E_{k_A}(k), \\ (3) & A \to C(B):\ k. \end{array}$$

Для ее проведения нарушитель C осуществляет доступ к сети от имени участника B, повторяя первое сообщение участника A. В результате ключ k появляется в канале связи в явном виде.

Можно защитить протокол от этой атаки, осуществляя проверку на втором шаге с целью отбраковки повторно переданных сообщений. Но, как заметил Карлсен (Carlsen), все равно можно осуществить аналогичную атаку путем *чередования сообщений* двух различных сеансов:

Атака 3: (*interleaving attack*)

$$\begin{array}{ll} (1) & A \to C(B):\ E_{k_A}(k), \\ (1') & A \to C(B):\ E_{k_A}(k'), \\ (2') & A \leftarrow C(B):\ E_{k_A}(k), \\ (3') & A \to C(B):\ k, \\ (2) & A \leftarrow C(B):\ M, \\ (3) & A \to C(B):\ D_{k_A}(M). \end{array}$$

Здесь M — произвольное фиктивное сообщение.

Стойкость протокола зависит от многих факторов:

— от надежности криптографических механизмов;

— от правильности реализации;

— от точности выполнения порядка предписанных действий участниками протокола и др.

Основное внимание будем концентрировать на тех **слабостях** протоколов, которые обусловлены способом формирования и порядком отправки сообщений участниками, то есть ошибками, допущенными при синтезе самих протоколов, представляющих собой предписания, определяющие порядок действий всех участников.

Приведем основные **предположения**, которые применяют при анализе протоколов.

Предположение 1. *Perfect cryptography assumption* — все стандартные криптографические примитивы удовлетворяют условию совершенной стойкости; слабости могут быть вызваны только непродуманным порядком действий, предписанных самим протоколом.

Предположение 2. *Strong typing assumption* (строгое соблюдение типов) — все участники правильно понимают форматы получаемых сообщений и корректно распознают типы полей в записи передаваемых сообщений.

Предположение 3. *Honest participants* (честность участников) — все участники точно выполняют предписанный согласно протоколу порядок действий.

Предположение 4. *Bounded number of sessions* (ограниченное число сеансов) — для сведения задачи к конечному числу состояний.

Заметим, что при неограниченном числе сеансов проблема безопасности протокола неразрешима (S. Even, O. Goldreich, 1983).

При **классификации атак** на безопасность протоколов можно применять различные подходы:

— по способу воздействия (активная, пассивная);

— по цели (на какое свойство протокола осуществляется атака);

— по области применения (универсальные, зависящие от реализации);

— по классу протоколов;

— по применяемому математическому аппарату (без дополнительных средств, например, используя только повтор, отражение или задержку сообщений; статистические; алгебраические и др.);

— по способу применения (число необходимых сеансов, можно ли ограничиться старыми или нужны вновь открытые параллельные сеансы);

— по требуемым ресурсам (ограниченные или неограниченные вычислительные возможности);

— по наличию и характеру дополнительной информации (известные сообщения, сеансовые или долговременные ключи и т. п.), и т. д.

5. Перечень наиболее широко известных атак на криптографические протоколы

1. *Подмена (impersonation)* — попытка подменить одного пользователя другим. Нарушитель, выступая от имени одной из сторон и полностью имитируя ее действия, получает в ответ сообщения определенного формата, необходимые для подделки отдельных шагов протокола.

Методы противодействия состоят в:

— сохранении в тайне от противника информации, определяющей алгоритм идентификации;

— использование различных форматов сообщений, передаваемых на разных шагах протокола;

— вставка в них специальных идентификационных меток и номеров сообщений.

В протоколах с использованием третьей стороны возможны атаки, основанные на подмене доверенного сервера.

Например, одна из сторон, имеющая доверительные отношения с сервером, выступает от его имени, подменяет его трафик обмена с другими сторонами и в результате получает возможность раскрывать значения генерируемых центром ключей.

Эта атака может быть успешной для протоколов, в которых аутентификация при доступе к серверу основана только на идентификаторах сторон и случайных числах, генерируемых при каждом взаимодействии.

Для защиты от таких атак применяют средства привязки ключей не к одной, а к обеим взаимодействующим сторонам путем передачи обоих идентификаторов в зашифрованном виде.

2. *Повторное навязывание сообщения* (*replay attack*) — повторное использование ранее переданного в текущем или предыдущем сеансе сообщения или какой-либо его части в текущем сеансе протокола.

Например, повторная передача информации ранее проведенного протокола идентификации может привести к повторной успешной идентификации того же самого или другого пользователя.

В протоколах передачи ключей данная атака часто применяется для повторного навязывания уже использованного ранее сеансового ключа — *атака на основе новизны* (*freshness attack*).

Методы противодействия состоят в обеспечении целостности сеанса и невозможности вставки в него лишних сообщений. Для этого используется:

— техника типа «запрос — ответ»;

— вставка в передаваемые сообщения временных меток, случайных чисел или возрастающих последовательностей чисел.

3. Еще один тип подобных атак связан с обратной передачей адресату ранее переданных им сообщений и получил название *атака отражением* (*reflection attack*). Часто атаки данного типа относят к классу атак с повторным навязыванием сообщения.

Для защиты от таких атак протоколы специально делают несимметричными, включая в зашифрованные сообщения идентификаторы сторон либо изменяя процедуры так, чтобы стороны должны были выполнять разные действия, вводят в протокол идентификационную информацию, используют различные ключи для приема и передачи сообщений.

4. *Задержка передачи сообщения* (*forced delay*) — перехват противником сообщения и навязывание его в более поздний момент времени. Это также разновидность атаки с повторным навязыванием сообщения.

Методы противодействия включают использование случайных чисел совместно с ограничением временного промежутка для ответа, использование временных меток.

5. *Комбинированная атака* (*interleaving attack*) — подмена или другой метод обмана, использующий комбинацию данных из ранее выполненных протоколов, в том числе протоколов, ранее навязанных противником.

Метод противодействия состоит в обеспечении целостности сеансов протоколов и отдельных сообщений.

6. Специальный частный случай предыдущей атаки, в котором противник специально открывает одновременно несколько параллельных сеансов с целью использования сообщений из одного сеанса в другом, получил название *атака с параллельными сеансами* (*parallel-session attack*).

Пример: протокол NSPK (Needham — Schroeder public key protocol). Это протокол осуществления взаимной аутентификации, предложенный Нидхэмом и Шредером в 1978 г.:

$$A \to B: \ E_B(r_A, A),$$
$$A \leftarrow B: \ E_A(r_A, r_B),$$
$$A \to B: \ E_B(r_B).$$

В данном случае дважды используется техника «запрос — ответ», причем случайные числа спрятаны внутри зашифрованных сообщений.

17 лет он считался стойким, пока G. Lowe в 1995 г. с использованием автоматизированного средства FDR не обнаружил слабость: нарушитель C, используя свой законный обмен с участниками A и B, может открыть второй сеанс протокола с участником B и при этом, чередуя сообщения обоих сеансов (interleaving attack), может найти случайное число r_B, сгенерированное участником B:

$$\begin{array}{lllllll}
(1) & A & \to & C: & & & E_C(r_A, A), \\
(1') & & & C(A) & \to & B: & E_B(r_A, A), \\
(2') & & & C(A) & \leftarrow & B: & E_A(r_A, r_B), \\
(2) & A & \leftarrow & C: & & & E_A(r_A, r_B), \\
(3) & A & \to & C: & & & E_C(r_B), \\
(3') & & & C(A) & \to & B: & E_B(r_B).
\end{array}$$

В результате C может убедить участника B в том, что он выступает от имени A.

7. *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.

Эта атака может включать специально подобранные

— открытые тексты, если доказывающий должен подписать или зашифровать запрос,

— шифрованные тексты, если доказывающий должен расшифровать запрос.

Методы противодействия этой атаке состоят:

— во включении случайных чисел в запросы или ответы, а также

— в использовании протоколов с нулевым разглашением.

8. *Использование противником своих средств в качестве части телекоммуникационной структуры* — атака, при которой в протоколе идентификации между A и B противник входит в телекоммуникационный канал и становится его частью при реализации протокола между A и B.

При этом противник может подменить информацию, передаваемую между A и B.

Эта атака особенно опасна в случае формирования участниками A и B общего ключа по протоколу Диффи — Хеллмана. Она известна как «*противник в середине*» и заключается в полной подмене всех сообщений между сторонами.

Пример: протокол Диффи – Хеллмана. Первый алгоритм открытого распределения ключей был предложен У. Диффи и М. Хеллманом (Diffie W., Hellman M. E.) в 1976 г. Для его выполнения стороны должны договориться о значениях большого простого числа p и образующего элемента α мультипликативной группы $\mathrm{Z}_p^* = \{1, 2, \ldots, p-1\}$. Для выработки общего ключа k они должны сгенерировать случайные числа x, $1 \leqslant x \leqslant p-2$, и y, $1 \leqslant y \leqslant p-2$, соответственно. Затем они должны обменяться сообщениями.

Протокол DH:

$$\begin{array}{l}
A \to B: \ \alpha^x \bmod p, \\
A \leftarrow B: \ \alpha^y \bmod p.
\end{array}$$

Искомый общий ключ теперь вычисляется по формуле

$$k = (\alpha^y)^x = (\alpha^x)^y \bmod p.$$

Слабость: отсутствие аутентификации сторон.

Атака: (*«противник в середине»*)

$$\begin{array}{lll} A \to \ C(B): & & \alpha^x \bmod p, \\ \quad C(A) & \to B: & \alpha^{x^*} \bmod p, \\ \quad C(A) & \leftarrow B: & \alpha^y \bmod p, \\ A \leftarrow \ C(B): & & \alpha^{y^*} \bmod p. \end{array}$$

Противник C:
— выбирает числа x^* и y^*;
— подменяет α^x и α^y на α^{x^*} и α^{y^*} соответственно;
— вычисляет ключи $k_{AC} = (\alpha^x)^{y^*} \bmod p$ и $k_{CB} = (\alpha^y)^{x^*} \bmod p$.

В результате C полностью контролирует обмен сообщениями между A и B, A и B не замечают подмену — они уверены, что связываются непосредственно друг с другом.

Пример. Протокол MTI. Интересный подход к защите протокола DH от атаки «противник в середине» был предложен Т. Мацумото, И. Такашима и Х. Имаи (Matsumoto T., Takashima Y., Imai H.) в 1986 г. Они предложили серию протоколов, предполагающих наличие у абонентов открытых ключей и использующих различные модификации процедуры выработки общего ключа.

Рассмотрим протокол MTI/A0. Предположим, что участники A и B имеют секретные ключи a, $1 \leqslant a \leqslant p-2$, и b, $1 \leqslant b \leqslant p-2$, соответственно и публикуют свои открытые ключи $\beta_A = \alpha^a \bmod p$ и $\beta_B = \alpha^b \bmod p$. Для выработки общего секретного ключа k они должны сгенерировать случайные числа x, $1 \leqslant x \leqslant p-2$, и y, $1 \leqslant y \leqslant p-2$, соответственно, а затем обменяться следующими сообщениями:

$$\begin{array}{l} A \to B: \ \alpha^x \bmod p, \\ A \leftarrow B: \ \alpha^y \bmod p. \end{array}$$

Теперь участники A и B вычисляют общий ключ $k = \alpha^{xb+ya} \bmod p$ по формулам

$$\begin{array}{l} k_A = (\alpha^y)^a \, \beta_B^x \bmod p, \\ k_B = (\alpha^x)^b \, \beta_A^y \bmod p \end{array}$$

соответственно.

Любая подмена сообщений приводит к тому, что все стороны получают различные значения ключа, в результате чего становится невозможным чтение всей передаваемой информации. Тем самым свойство аутентификации ключа протокола при атаке «противник в середине» не нарушено. Вместе с тем этот протокол не обеспечивает аутентификации сторон и подтверждения правильности получения ключа.

Покажем, что даже после усиления этого протокола путем добавления к пересылаемым сообщениям сертификатов он остается уязвимым к подмене содержания передаваемых сообщений, а именно, этот протокол не обеспечивает аутентификации сторон.

Атака на протокол MTI/A0, в которой нарушитель C подменяет сертификат участника A на свой собственный. Пусть участник A получил сертификат

$$\mathrm{cert}_A = (A, \beta_A, \mathrm{Sig}_T(A, \beta_A)).$$

Нарушитель C регистрирует для себя сертификат

$$\mathrm{cert}_C = (C, \beta_A^e, \mathrm{Sig}_T(C, \beta_A^e)),$$

где e — произвольное число, предназначенное для того, чтобы скрыть связь с ключом участника A. Далее он использует следующий протокол:

$$\begin{aligned} A \to \ & C(B): && \mathrm{cert}_A, \alpha^x \bmod p, \\ & C \quad \to B: && \mathrm{cert}_C, \alpha^x \bmod p, \\ & C \quad \leftarrow B: && \mathrm{cert}_B, \alpha^y \bmod p, \\ A \leftarrow \ & C(B): && \mathrm{cert}_B, (\alpha^y)^e \bmod p. \end{aligned}$$

В результате участник B будет уверен, что полученный в результате выполнения этого протокола ключ

$$k = \alpha^{aey+bx} \bmod p$$

будет общим ключом с участником C, а не с участником A, который ничего об этом не подозревает. При этом участник A вычислит тот же ключ k. Поэтому вся информация, полученная от участника A и зашифрованная на этом ключе, будет восприниматься участником B как полученная от участника C, а не от A.

Таким образом, данный протокол хотя и обеспечивает аутентификацию ключа для участника A, в то же время не обеспечивает аутентификации сторон, а также аутентификации ключа для участника B.

9. Во многих случаях проведение атаки облегчает дополнительная информация. Например, *атака с известным сеансовым ключом* (*known-key attack*) заключается в попытке получения информации о долговременном ключе или любой другой ключевой информации, позволяющей восстанавливать сеансовые ключи для других сеансов протокола.

Для защиты от такой атаки обеспечивают *независимость* между различными применяемыми ключами, которая достигается с помощью протоколов совместной выработки ключа, гарантирующих свойство *новизны ключа* (*freshness*) и не позволяющих ни одному из участников заранее предсказать значение ключа.

10. *Атака с неизвестным общим ключом* (*unknown key-share attack*) — атака, при которой нарушитель C открывает два сеанса с A и B, выступая в первом случае от имени B, хотя последний может ничего не знать об этом. При этом в результате будет сформирован общий ключ между A и B, причем A будет уверен, что сформировал общий ключ с B, а B будет уверен, что сформировал общий ключ с C. Сам ключ может быть не известен C.

Примерный сценарий может выглядеть так: пусть B — филиал банка и A — держатель счета. Сертификат выдается центральным банком и содержит информацию о держателе счета. Пусть протокол электронного депозита вкладов должен выработать ключ для филиала. После аутентификации B как отправителя зашифрованный вклад помещается на счет, указанный в сертификате. Если не проводится никакой последующей аутентификации в зашифрованном сообщении депозита, то при успешно проведенной атаке вклад будет положен на счет C вместо счета A.

Пример: протокол STS. Попытка построения аутентифцированного протокола на базе **DH** была предпринята в протоколе STS (*station-to-station*), созданном У. Диффи (W. Diffie), П. Ван Ооршотом (P. Van Oorschot) и М. Вейнером (M. Wiener) в 1992 г.

$$\begin{aligned} & A \to B: \ A, B, m_A = \alpha^x \bmod p, \\ & A \leftarrow B: \ B, A, m_B = \alpha^y \bmod p, E_k(\mathrm{Sig}_B(m_B, m_A)), \\ & A \to B: \ A, B, E_k(\mathrm{Sig}_A(m_A, m_B)). \end{aligned}$$

Здесь Sig_A и Sig_B — цифровые подписи пользователей A и B соответственно, $k = \alpha^{xy} \bmod p$ — искомый общий ключ.

Вставка во второе и третье сообщения протокола значений цифровых подписей позволяет гарантировать достоверность получения сообщения именно от того пользователя, от которого это сообщение получено.

Шифрование значений подписей пользователей с помощью симметричного алгоритма E введено для того, чтобы обеспечить взаимное подтверждение правильности вычисления значения ключа, так как при неверно вычисленном ключе невозможно получить верные значения цифровых подписей.

Атака: (Lowe G., 1996) (*атака с неизвестным общим ключом*) (unknown key-share attack, UKS attack)

$$\begin{array}{lll} A \to & C(B): & A, B, m_A, \\ & C \to B: & C, B, m_A, \\ & C \leftarrow B: & B, C, m_B,\ E_k(\mathrm{Sig}_B(m_B, m_A)), \\ A \leftarrow & C(B): & B, A, m_B,\ E_k(\mathrm{Sig}_B(m_B, m_A)), \\ A \to & C(B): & A, B, E_k(\mathrm{Sig}_A(m_A, m_B)). \end{array}$$

Нарушитель C, используя свой законный обмен с участником B, может применить против участника A атаку, в результате которой он может убедить A в том, что он выступает от имени B. Нарушитель C использует свой законный обмен с участником B и убеждает A в том, что он выступает от имени B. Сеанс с участником B остается незавершенным, так как C, не зная секретного ключа участника A, не сможет подобрать правильный ответ для B. Поэтому любое его сообщение на третьем шаге будет участником B отвергнуто.

Данная атака не представляет реальной опасности, так как при этом нарушитель не будет знать секретного ключа $k = \alpha^{xy} \bmod p$ и поэтому не сможет читать передаваемые сообщения, передаваемые от A к B. Однако в результате участник A не будет ничего подозревать и примет участника C за B.

Таким образом, протокол не обеспечивает: аутентификации сторон, аутентификации ключа для участника B, так как последний будет полагать, что сформировал общий ключ с участником C, а на самом деле — с участником A.

Пример: модифицированный протокол STS. В 2004 г. К. Бойд и А. Матура предложили следующую модификацию протокола STS:

$$\begin{array}{l} A \to B:\ A, B, m_A = \alpha^x \bmod p, \\ A \leftarrow B:\ B, A, m_B = \alpha^y \bmod p, \mathrm{Sig}_B(m_B, m_A), h_{k_0}(m_B, m_A) \\ A \to B:\ A, B, \mathrm{Sig}_A(m_A, m_B), h_{k_0}(m_A, m_B), \end{array}$$

где $k_0 = f(k)$ — ключевой параметр хеш-функции, вычисляемый как значение некоторой функции от результирующего сеансового ключа $k = \alpha^{xy} \bmod p$.

Атака: для данной версии протокола можно применить двустороннюю атаку с неизвестным общим ключом (*bilateral unknown key-share attack, BUKS attack*):

$$\begin{array}{lll} A \to & C: & A, B, m_A, \\ & C \to D: & A, B, m_A, \\ & D \to B: & D, B, m_A, \\ & D \leftarrow B: & B, D, m_B, \mathrm{Sig}_B(m_B, m_A), h_{k_0}(m_B, m_A), \\ & C \leftarrow D: & B, A, m_B, h_{k_0}(m_B, m_A) \\ A \leftarrow & C: & C, A, m_B,\ \mathrm{Sig}_C(m_B, m_A), h_{k_0}(m_B, m_A), \\ A \to & C: & A, C, \mathrm{Sig}_A(m_A, m_B), h_{k_0}(m_A, m_B), \\ & C \to D: & A, B, h_{k_0}(m_A, m_B), \\ & D \to B: & D, B, \mathrm{Sig}_D(m_A, m_B), h_{k_0}(m_A, m_B). \end{array}$$

В результате участники C и D, вступившие в сговор, вводят в заблуждение участников A и B, сформировавших общий ключ. При этом участник A уверен, что он сформировал общий ключ с участником C, а участник B уверен, что он сформировал общий ключ с участником D.

11. Имеется большое число типов атак, которые зависят от *конкретной реализации* протокола.

Например, для криптографических протоколов на основе симметричных шифрсистем можно использовать особенности работы самих шифрсистем и, в частности, реализованных способов и режимов шифрования, синхронизации и т. п.

Чтобы защититься от подобных атак, необходимо провести анализ архитектуры протокола и структуры передаваемых сообщений с целью определения возможных уязвимостей, позволяющих, например, осуществить навязывание сообщений с известными или одинаковыми значениями определенных полей, либо с помощью подмены типа некоторых полей.

12. Для криптографических протоколов, построенных на основе асимметричных шифрсистем, основной уязвимостью является возможность осуществления подмены открытого ключа одного из участников на другой открытый ключ с известной противнику секретной половиной этого ключа. В частности, это позволяет противнику узнавать содержание зашифрованных сообщений, отправляемых данному участнику.

В данном случае нарушается свойство связанности открытого ключа и идентификатора участника. Поэтому атаку данного типа называют *атакой на основе связывания* (*binding attack*).

Для защиты от подобных атак вместо открытых ключей используют их *сертификаты*, создавая специальную инфраструктуру (PKI) для выдачи, отзыва и проведения проверки их правильности.

Обычно используют следующие сокращенные названия атак:

Сокр.	Название
MITM	«*Противник в середине*» (man-in-the-middle attack)
Replay	*Атака с повторной передачей* (replay attack)
TF	*подмена типа* (type flaw attack)
STS	*Атака с известным разовым ключом* (short-term secret)
PS	*Атака с параллельными сеансами* (parallel-session)
KN	*Атака с известным сеансовым ключом* (known-key attack)
UKS	*Атака с неизвестным сеансовым ключом* (unknown key-share attack)

Примеры атак на криптографические протоколы приведены в табл. 4.

6. Мощность пространства атак

При совершении атак участники могут выступать в разных ролях. Кроме того, одновременно может выполняться несколько сеансов, причем в разных сеансах один и тот же участник может выполнять различные роли. Каждая *роль* состоит в последовательном выполнении определенных действий, связанных с отправлением и получением сообщений. Противник «q» может выступать от имени любого из участников протокола. При проведении атаки может одновременно выполняться несколько сеансов протокола.

Таблица 4

Примеры атак на криптографические протоколы

Протокол	Атака
ISO с симметричными ключами 1-проходный	Replay
ISO с симметричными ключами 2-проходный	Replay
Andrew Secure RPC	TF, Replay
Needham–Shroeder с симметричными ключами	STS
Denning–Sacco с симметричными ключами	TF
Otway–Rees	TF
Wide Mouthed Frog	PS
Yahalom	TF
Woo–Lam с симметричными ключами	Replay, PS
Woo–Lam с симметричными ключами	PS
ISO с открытыми ключами 1-проходный	Replay
ISO с открытыми ключами 1-проходный	Replay
NSPK (Needham–Shroeder Public Key)	MITM
NSPK с ключевым сервером	MITM
NSL (NSPK, модифицированный Lowe)	MITM
Denning–Sacco Key Distr. (с открытыми ключами)	MITM
Shamir–Rivest–Adleman 3-проходный	Replay, PS
CCITT X.509	TF
EKE (Encrypted Key Exchange)	PS

Поэтому общее описание атаки на протокол представляет собой композицию параллельных процессов, соответствующих различным ролям каждого из сеансов выполнения протокола.

Пространство атак. Общим описанием различных траекторий выполнения протокола с двумя ролями при реализации всевозможных атак на протокол будет запись $q||\, r_1^*||\, r_2^*$, где $r^* = r||\, r||\, r||\, \ldots$ Эта запись означает, что одновременно может выполняться несколько процессов, один из которых соответствует противнику, а остальные — двум ролям, каждая из которых может выполняться произвольное конечное число раз.

Запись $q||\, r_1^*||\, r_2^*$ задает все пространство всевозможных траекторий выполнения протокола, которые могут получаться при всевозможных атаках. Будем называть его *пространством поиска атак*. Его элементами, называемыми *траекториями*, являются мультимножества вида $\{q^1, r_1^{k_1}, r_2^{k_2}\}$. Здесь запись $r_i^{k_i}$ означает, что роль r_i выполняется k_i раз.

Пространство всевозможных траекторий выполнения протокола является бесконечным, поэтому обычно вводят ограничения на число n различных реализаций ролей протокола (в данном случае это $n = k_1 + k_2$).

Каждую конкретную реализацию роли называют *запуском* (run). В различных моделях протокола этому понятию соответствуют термины: «регулярная нить» (regular strand), «процесс» (process), «нить» (thread) и др. Конкретную реализацию траектории, представляющую собой мультимножество описаний запусков, называют *сценарием* (scenario). В некоторых моделях для данного понятия используют термин «*связка*» (bandle).

Будем обозначать участников протокола маленькими буквами $a, b, c \dots$ Для символического описания каждого запуска протокола будем использовать запись (run description)

$$r_i(x_1, x_2, \dots, x_t),$$

где r_i — роль, выполняемая участником x_1, $x_1 \in \{a, b, c \dots\}$; $x_2, \dots, x_t \in \{q, a, b, c \dots\}$ — другие участники, с которыми участник x_1 осуществляет взаимодействие; t — число ролей протокола.

Участник x_1 изначально не знает, с кем он на самом деле осуществляет взаимодействие.

Обычно при анализе протокола предполагается, что имеется два честных участника и один противник.

Мощность пространства атак. Подсчитаем мощность пространства атак при ограничении на число запусков протокола.

Утверждение 1 (Cas Cremers, Pascal Lafourcade, 2006). Пусть в спецификации протокола предусмотрено m участников и t ролей. Тогда мощность пространства поиска атак, проводимых при n запусках протокола, равна

$$\binom{tm(m+1)^{t-1} + n - 1}{n}.$$

Доказательство. Число M всевозможных запусков протокола равно числу различных описаний ролей вида $r(x_1, x_2, \dots, x_t)$, где число ролей r равно t, значение переменной x_1 может быть выбрано m способами, значение каждой из переменных $x_2, \dots, x_t$ может быть выбрано $m + 1$ способом.

Поэтому

$$M = tm(m+1)^{t-1}.$$

Теперь надо вычислить число сценариев, т. е. мультимножеств порядка n, состоящих из всевозможных описаний запусков протокола. Оно совпадает с числом размещений n элементов по M ящикам и равно

$$\binom{M + n - 1}{n}.$$

■

Пример. При $t = m = n = 2$ пространство всевозможных сценариев атак на протокол равно

$$\binom{2 * 2 * 3^{2-1} + 2 - 1}{2} = \binom{13}{2} = 78.$$

Данное пространство включает сценарии всех возможных атак. Например, атаке «противник в середине» соответствует сценарий

$$\{r_1(a, q), r_2(b, q)\}.$$

Не каждому сценарию соответствует реальная атака.

Мощность пространства атак с учетом эквивалентных сценариев с точностью до перестановки участников. Введем отношение эквивалентности на пространстве поиска атак, т. е. множестве различных сценариев. Рассмотрим группу преобразований, действующую путем перестановки участников. Для двух участников она имеет

порядок два и определяется двумя перестановками:

$$\binom{a,\ b}{a,\ b}, \quad \binom{a,\ b}{b,\ a}.$$

Воспользуемся леммой Бернсайда, которая утверждает, что число орбит при действии группы G на множестве X равно

$$\frac{1}{\mid G \mid} \sum_{g \in G} i(g),$$

где $i(g)$ обозначает число неподвижных элементов подстановки $g \in G$.

В нашем случае $\mid G \mid = 2$.

Следствие. Число классов эквивалентности мультимножеств равно

$$\frac{1}{\mid G \mid} \sum_{g \in G} i(g) = \frac{\binom{2t3^{t-1}+n-1}{n} + \varepsilon_n \binom{t3^{t-1}+\frac{n}{2}-1}{\frac{n}{2}}}{2},$$

где $\varepsilon_n = 1$ при четных n и 0 в противном случае.

Действительно, для тождественной подстановки неподвижными будут все мультимножества. Для транспозиции участников неподвижными будут только такие мультимножества, в которых входящие в них запуски разбиваются на пары, отличающиеся в своей записи заменой $\{a \to b, b \to a\}$. При нечетных n такое разбиение невозможно.

При $t = m = n = 2$ получаем 42 класса эквивалентности вместо 78 мультимножеств. Например, атаке «противник в середине» соответствует два эквивалентных сценария $\{r_1(a, q), r_2(b, q)\}$ и $\{r_1(b, q), r_2(a, q)\}$.

7. Виды криптографических протоколов

Есть много подходов к классификации протоколов. Приведем некоторые из них.

- Классификация *по числу участников*:
  - двусторонний,
  - трехсторонний и т. п.,
  - многосторонний.
- Классификация *по числу передаваемых сообщений*:
  - интерактивный (есть взаимный обмен сообщениями),
  - неинтерактивный (только однократная передача). Неинтерактивные протоколы часто называют *схемами*.
- Классификация *по целевому назначению* протокола:
  - протокол обеспечения целостности сообщений,
    - с аутентификацией источника,
    - без аутентификации источника;
  - протокол (схема) цифровой подписи,
    - протокол индивидуальной / групповой цифровой подписи,
    - с восстановлением / без восстановления сообщения,
    - протокол цифровой подписи вслепую,
    - протокол конфиденциальной цифровой подписи,
    - протокол цифровой подписи с доказуемостью подделки;
  - протокол идентификации (аутентификации участников),
    - односторонней аутентификации,

  - двусторонней (взаимной) аутентификации;
- конфиденциальная передача,
  - обычный обмен сообщениями,
  - широковещательная / циркулярная передача,
  - честный обмен секретами,
  - забывающая передача,
  - протокол привязки к биту (строке);
- протокол распределения ключей,
  - протокол (схема) предварительного распределения ключей,
  - протокол передачи ключа (обмена ключами),
  - протокол совместной выработки ключа (открытого распределения ключей),
  - протокол парный / групповой,
  - протокол (схема) разделения секрета,
  - протокол (распределения ключей для) телеконференции, и др.

Групповые протоколы (*group-oriented protocol*) предполагают одновременное участие групп участников, например:

— *протокол разделения секрета* (*secret sharing protocol*) — если все группы, имеющие на это право, формируют одинаковые ключи;

— *протокол телеконференции* — если у различных групп должны быть разные ключи;

— *протокол групповой подписи* (*group signature protocol*) — предполагается одновременное участие заранее определенной группы участников, причем в случае отсутствия хотя бы одного участника из группы формирование подписи невозможно.

Примитивный криптографический протокол (*primitive cryptographic protocol*) — это криптографический протокол, который не имеет самостоятельного прикладного значения, но используется как базовый компонент при построении прикладных криптографических протоколов. Как правило, он решает какую-либо одну абстрактную задачу. Примеры: протокол обмена секретами, протокол привязки к биту, протокол подбрасывания монеты (по телефону).

Прикладной криптографический протокол (*application cryptographic protocol*) предназначен для решения практических задач обеспечения функций — сервисов безопасности с помощью криптографических систем. Следует заметить, что прикладные протоколы, как правило, обеспечивают не одну, а сразу несколько функций безопасности. Более того, такие протоколы, как IPsec, на самом деле являются большими семействами различных протоколов, включающими много разных вариантов для различных ситуаций и условий применения.

Примерами прикладных протоколов являются:

- система электронного обмена данными:
  - протоколы электронного документооборота;
- система электронных платежей:
  - протоколы систем с виртуальными деньгами,
    - удаленный платеж по электронным чекам;
  - протоколы систем с электронными деньгами,
    - протокол удаленного платежа по кредитным картам,
    - протокол электронного денежного перевода,
    - протокол электронного дебетового поручения;
  - протоколы систем с цифровыми деньгами,

  - протокол снятия со счета цифровой наличности,
  - платежный протокол с цифровыми деньгами,
  - протокол депозита для сдачи цифровых денег в банк,
  - протокол с идентификацией повторной траты монеты;
- система электронной коммерции:
  - протокол подписания контракта,
  - протокол сертифицированной электронной почты,
  - протокол электронного аукциона;
- поддержка правовых отношений:
  - протоколы голосования (электронные выборы);
- игровые протоколы,
  - протокол бросания жребия (по телефону),
  - протокол игры в покер (по телефону) и т. д.

Пример. Задача жребия по телефону (*coin-flipping by telephone*) была впервые достаточно ярко сформулирована в 1982 г. М. Блюмом: «Муж и жена недавно развелись, живут в разных городах и хотят решить, кому достанется машина. Жребий бросает жена. Муж загадывает, что выпадет решка и сообщает об этом жене по телефону, после чего слышит, как жена (на другом конце провода) говорит: "Ну хорошо... Я бросаю ... Орел! Ты проиграл!"»

Для решения этой задачи может быть применен следующий **протокол привязки к биту**:

$$\begin{aligned} A \to B: &\quad \gamma = f(a, x), \\ A \leftarrow B: &\quad b, \\ A \to B: &\quad x. \end{aligned}$$

Здесь a — загадываемое A значение; b — угадываемое B значение; γ — свидетельство; x — случайное значение. Можно подобрать функцию f так, чтобы для этого протокола выполнялись свойства сокрытия и связывания. Первое свойство означает, что B не может извлечь значение a из полученного сообщения γ, что позволит правильно «угадать» значение b. Второе свойство гарантирует невозможность подбора значений x_1 и x_2, удовлетворяющих равенству $f(0, x_1) = f(1, x_2)$, т. е. позволяющих осуществить со стороны A подмену выбранного значения.

Классификацию криптографических протоколов можно проводить также и по другим признакам:

- *по типу используемых криптографических систем*:
  - на основе симметричных криптосистем,
  - на основе асимметричных криптосистем,
  - смешанные;
- *по способу функционирования*:
  - интерактивный / неинтерактивный,
  - однопроходный / двух- / трех- и т. д. проходный,
  - протокол с арбитром (протокол с посредником),
  - двусторонний / с доверенной третьей стороной (с центром доверия), и т. п.

Протокол с арбитром (*arbitrated protocol*), синоним *протокол с посредником* — криптографический протокол, в котором для разрешения споров между участниками требуется арбитраж.

Если обмен сообщениями осуществляется с участием специально выделенного участника, обладающего доверием других участников, то говорят о *протоколах с доверенной третьей стороной* (ТЗР) или о *протоколах с центром доверия.*

Различают протоколы, в которых третья ТЗР сторона работает в режимах: on-line — в реальном времени, off-line — в отложенном режиме, in-line — в режиме посредника.

Протоколы доказательства

Доказательство интерактивное (*interactive proof*) осуществляется путем выполнения протокола с двумя участниками: доказывающий — убеждает проверяющего в истинности некоторого утверждения; проверяющий — либо принимает, либо отвергает доказательство. Характеризуется двумя условными вероятностями:

— если доказываемое утверждение верно, то доказательство должно быть верным с вероятностью, стремящейся к единице при увеличении числа повторений протокола;

— если же доказываемое утверждение ложно, то при увеличении числа повторений протокола вероятность правильности доказательства должна стремиться к нулю.

Наиболее распространены протоколы доказательства знания (какого-либо факта).

Доказательство знания (*proof of knowledge*) — доказательство интерактивное, при котором доказывающий убеждает проверяющего в том, что он владеет секретной информацией, не раскрывая её.

Протокол интерактивного доказательства должен учитывать возможность обмана со стороны обоих участников. Если участник A (доказывающий) на самом деле не знает доказываемого утверждения (либо от имени участника A выступает кто-либо другой), то B должен обнаружить факт обмана. Поэтому доказательство знания характеризуется двумя свойствами: полнотой и корректностью.

Полнота (*completeness property*) — свойство криптографического протокола, означающее, что при выполнении честными участниками протокол решает ту задачу, для которой он создан.

Корректность (*soundness property*) — способность криптографического протокола противостоять угрозам со стороны противника и/или нарушителя, не располагающего необходимой секретной информацией, но пытающегося выполнить протокол за участника A, который по определению должен такой информацией владеть.

С другой стороны, протокол должен обеспечить возможность доказательства владения секретной информацией, не раскрывая её. Если проверяющий B захочет получить какую-либо информацию об этом утверждении, помимо самого факта владения ею, то его попытки сделать это должны быть обречены на неуспех.

Поэтому центральным свойством таких протоколов является *разглашение нулевое* (*Zero-knowledge property*) — свойство протокола доказательства знания, обеспечивающее такое его выполнение, что никакая информация о доказываемом утверждении, кроме факта его истинности, не может быть получена нечестным проверяющим из переданных сообщений за время, полиномиально зависящее от суммарной длины этих сообщений.

Как правило, протоколы доказательства выполняются в виде последовательности независимых *циклов* (раундов), каждый из которых состоит из трех шагов определенного вида:

$$\begin{array}{lll} (1) & A \to B: & \gamma \quad (\text{заявка, } witness), \\ (2) & A \leftarrow B: & x \quad (\text{запрос, } challenge), \\ (3) & A \to B: & y \quad (\text{ответ, } response). \end{array}$$

После выполнения каждого такого цикла проверяющий принимает решение об истинности доказательства.

К категории доказательства знания относятся, например, протоколы идентификации.

8. Средства автоматизированного анализа

По принципам работы их можно сгруппировать в два основных класса: средства дедуктивного логического вывода, средства верификации.

Средства **первого типа** используют дедуктивный подход, основанный на логической проверке корректности рассматриваемого протокола. Свойства формулируются как утверждения в рамках некоторой логики, и для их проверки ищется *логический вывод* этого утверждения. Применяется для доказательства корректности системы, того, что протокол действительно удовлетворяет определённым требованиям.

Средства первого типа можно подразделить на три основные группы, в основе которых лежат соответственно: логика доверия и генерация теории; автоматическое доказательство теорем; логическое программирование.

Средства **второго типа** основаны на использовании методов верификации, и прежде всего метода проверки на модели (*model checking*).

По протоколу строится формальная модель — *система переходов*, состояниями которой являются множества высказываний о свойствах протокола. Затем она 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 второго типа можно подразделить на следующие основные группы: основанные на автоматах; основанные на пространстве нитей; основанные на логике.

Соберем в табл. 5 сводную информацию о существующих средствах автоматизированного анализа протоколов. В ней использованы следующие обозначения: A — алгебраический подход, TP — автоматическое доказательство теорем (theorem proving), MC — проверка моделей (model checking), BL — логика доверия (believe logic).

Алгебраический подход. Д. Долев и А. Яо (Dolev, Yao) в 1983 г. предложили алгоритм анализа протоколов, основанный на сведении задачи анализа протокола к *обобщенной проблеме тождества слов* в некотором алфавите.

Алгебраический подход для записи сценариев работы протокола состоит в следующем: каждому сценарию выполнения протокола соответствует последовательность преобразований слов; проблема безопасности сводится к проверке выводимости слов определенного вида из множества слов, соответствующих возможным сценариям выполнения протокола. Правила преобразования слов определяются на основе *модели противника*, в которой для каждого протокола указано, что может и чего не может делать противник.

Модель противника Долева — Яо. Основные положения этой модели сводятся к следующему.

1) Противник может использовать любые доступные ему комбинации стандартных операций для построения новых сообщений из тех, которые ему известны. Предполагается, что, наблюдая в канале связи сообщения, противник всегда знает их структуру.

Таблица 5

Известные средства автоматизации анализа протоколов

Название	Авторы	Год	Тип	Примечание
DY	Dolev-Yao	1983	A	Множество слов
BAN	Burrow, Abadi	1989	BL	
GNY	Gong и др.	1990	BL	
AT	Abadi, Tuttle	1991	BL	
VO	van Oorshot	1993	BL	
SvO	Syverson и др.	1996	BL	
AUTLOG	Kessler и др.	1994	BL	
Accountability	Kailar	1996	BL	
Revere (RV)	Kindred	1999	BL	Генерация теории
NRL	Meadows	1996	TP	Специальное
Isabelle	Paulson	1997	TP	Универсальное
PVS	Dutertre и др.	1997	TP	Универсальное
Maude	Denker и др.	1998	TP	Универсальное
SPASS	Weidenbach	1999	TP	Универсальное, хорновские выражения
daTac	Rusinowitch и др.	2000	TP	Универсальное
TA4SP	Genet, Clay и др.	2004	TP	Специальное, древовидные автоматы
Athena	Song и др.	1999	MC	Модель пространства нитей
Interrogator	Millen и др.	1987	MC	Пространство состояний
Casper/FDR	Lowe	1996	MC	Специальное, компилятор Casper, алгебра процессов CSP
Murφ	Mitchell и др.	1997	MC	Универсальное
Cosec	Focardi и др.	1997	MC	Алгебры процессов (Security Process Algebra)
Brutus	Marrero	2001	MC	Системы переписывания термов, темпор. логика
Athena	Song и др.	1999	MC	Модель пространства нитей
STA	Boreale	2001	MC	
ProVerif	Blanchet	2001	MC	Аппроксимация, Pi-исчисление, хорновские выражения, PROLOG
Autofocus	Wimmel и др.	2002	MC	Диаграммы переходов состояний
CoProVe	Corin, Etalle	2002	MC	
HERMES	Bozga и др.	2003	MC	Логики CAPSL, SPL и TTL
CL-AtSe	Turuani	2003	MC	Символические модели
OFMC	Basin и др.	2004	MC	Символические модели
SATMC	Armando и др.	2005	MC	Выполнимость, планирование
Spider	Lenzini и др.	2004	MC	Темпоральная логика, алгебра процессов (spi-исчисление)
Scyther	Cremers	2006	MC	Симв. анализ, обр. поиск, шаблоны

Неизвестными ему могут быть только конкретные значения отдельных фрагментов сообщения. В частности, он может выделять части из перехваченных сообщений, составлять из них новые, расшифровывать и зашифровывать сообщения с помощью известных ему ключей, вычислять значения хеш-функций и т. п.

При этом ключи, случайные метки, идентификаторы абонентов и т. п. всегда рассматриваются противником как неделимые фрагменты, то есть они могут извлекаться противником из сообщений участников протокола только целиком.

2) Если в некотором сообщении содержится зашифрованная информация, то противник может использовать ее только в данном, зашифрованном виде, либо, чтобы извлечь ее, противник должен знать соответствующий ключ расшифрования. Более точно, предполагается, что все криптографические алгоритмы обладают свойством совершенности:

— если это система с открытыми ключами, все односторонние функции являются односторонними, общедоступный каталог открытых ключей является общедоступным и защищенным от подмены, расшифровать сообщение может только тот, кто обладает секретным ключом;

— если это симметричная шифрсистема, то, не зная ключа, нельзя ни зашифровать, ни расшифровать сообщение;

— если это система имитозащиты, то нельзя подделать значения кодов аутентичности сообщений;

— если это система цифровой подписи, то нельзя ни подделать значения цифровой подписи по сообщением, ни сформировать корректно подписанные сообщения, и т. п.

3) Предполагается, что противник обладает полным контролем над всеми каналами связи. Это означает, что противник может прослушивать все сообщения, удалять сообщения из каналов связи и перенаправлять их другим адресатам, формировать и отправлять любым участникам сообщения.

4) В общем случае противник может быть как внутренним (если это один или несколько участников протокола, называемых обычно нарушителями, *dishonest participants*), так и внешним (в этом случае он часто отождествляется с сетью), а также коалицией из участников и внешних субъектов (случай сговора). В связи с этим часто предполагается также, что каждый из участников может получать все сообщения, отправляемые другими участниками. Нечестные участники протокола раскрывают контролирующему каналы связи субъекту всю доступную им информацию, в том числе ключи. Таким образом, противник может взаимодействовать с другими участниками протокола от лица тех участников, которые вступили с ним в сговор.

Модель знаний противника Долева — Яо. Возможные изменения множества сообщений, составляющих совокупное знание противника, моделируются в терминах системы переходов с бесконечным числом состояний. Эта система переходов определяет дерево, представляющее возможные переходы состояний, стандартным образом: корень — это начальное состояние, а потомки представляют состояния, в которые система может перейти за один переход. Состоянием системы в каждый момент времени является множество сообщений, которые могут стать известны противнику на этом этапе выполнения протокола. Дерево переходов имеет бесконечное число состояний, так как по определению модели Долева — Яо каждый узел имеет бесконечное число преемников. Это дерево имеет бесконечную глубину, так как мы не можем ограничить число различных сеансов протокола.

Знания противника можно определить в терминах логических правил вывода. В соответствии с моделью Долева — Яо *знание противника* (*intruder knowledge*) представляет собой множество сообщений $\mathcal{DY}(M)$, которые сможет сгенерировать противник, зная множество сообщений M. Множество сообщений $\mathcal{DY}(M)$ определяется как минимальное множество, замкнутое относительно следующих правил, объединенных в две группы: правила генерации и правила анализа. Правила генерации показывают, как

противник может составлять сообщения, используя известные ему ключи и фрагменты сообщений; правила анализа показывают, как противник может разбирать сообщения. Таким образом, знание противника — это множество термов $\mathcal{DY}$, которые могут быть выведены из M с помощью определенных выше правил.

Процесс изменения этого множества в ходе реализации различных сценариев выполнения протокола формально описывается как последовательное применение к множеству термов формальных правил, называемых *правилами переписывания*, которые строятся на основе определенных выше правил вывода.

Формальная система, описывающая порядок применения этих правил, называется *системой переписывания термов*.

NRL Protocol Analyzer

NRL *Protocol Analyzer* разработан в US *Navy Research Laboratory* под руководством Catherine A. Meadows, 1996 г. Он осуществляет автоматическую генерацию инвариантов с целью ограничить потенциально бесконечное пространство состояний. Полученное пространство исследуется на возможность проведения атак на протоколы и для доказательства безопасности протоколов, даже при потенциально неограниченном числе выполнений протокола или неограниченном числе атак противника.

Небезопасное состояние специфицируется не только в терминах:

— значений переменных локального состояния,

— знаний противника,

— в терминах последовательности событий, которые должны или не должны произойти.

Например, можно запросить осуществить поиск состояния, в котором один и тот же ключ принят дважды участником (происходят два события), или состояния, в котором отвечающая сторона B принимает некий ключ для взаимодействия с инициатором A, в то время как A не инициировал этот протокол.

NRL *Protocol Analyzer* начинает движение из этого состояния и проходит в обратном направлении все маршруты в пространстве состояний, которые заканчиваются либо начальным состоянием, либо недостижимым состоянием.

Таким образом, этот анализатор можно использовать для доказательства того, что никакая сторона не будет аутентифицирована некорректно, но не для того, чтобы доказать, что аутентификация завершается всегда.

NRL *Protocol Analyzer* не делает никаких предположений об ограничениях на число сеансов выполнения протокола, на количество участников или на количество применяемых криптографических функций, что приводит к бесконечности пространства состояний. Однако он обеспечивает средства для специфицирования и доказательства индуктивных лемм о недостижимости бесконечных классов состояний, что позволяет во многих случаях проводить исчерпывающее исследование пространства состояний.

NRL *Protocol Analyzer* снабжен языком темпоральной логики NPATRL (NRL *Protocol Analyzer Temporal Requirements Language*) (P. Syverson and C. Meadows, 1993), позволяющим специфицировать требования к протоколу в терминах желательной или нежелательной последовательности событий.

NRL *Protocol Analyzer* и во многом похожий на него *Interrogator* (J. Millen, S. Clark and S. Freedman, 1987) были первыми инструментами для анализа протоколов, которые можно отнести к инструментам для исследования пространства состояний в обратном направлении. Движение в пространстве состояний начиналось из небезопасного состояния, и с помощью анализатора делалась попытка обнаружить состояние, из которого существовал маршрут в это небезопасное состояние и которое бы удовлетво-

ряло начальной конфигурации. Такие средства имели дело с бесконечными моделями и являлись фактически доказывателем теорем. Во время верификационного процесса требовалось вмешательство пользователя, кроме того, этот процесс не всегда можно было завершить.

FDR

G. Lowe (1996) исследовал применение FDR к анализу криптографических протоколов, специфицированных с помощью языка процессов CSP (*Communicating sequential processes*) (1980). Каждый участник протокола моделируется с помощью процесса CSP, который находится либо в состоянии ожидания сообщения, либо посылает сообщение. Каналы используются для взаимодействия между процессами и моделирования противника. Противник — параллельная композиция нескольких процессов, по одному на каждый факт или сообщение, из которых он может получить какие-либо сведения о выполнении протокола. Специально разработанный компилятор Casper (Lowe, 1997) позволил автоматизировать конструирование противника.

Под безопасным понимается такой процесс, в котором отсутствуют последовательности событий, соответствующие получению противником секретных сведений.

Murφ

J. Mitchell, M. Mitchell, and U. Stern в 1997 г. описали применение универсального анализатора моделей Murφ к анализу криптографических протоколов. В Murφ состояние системы определяется с помощью набора глобальных переменных состояния, включая разделяемые переменные, которые используются для моделирования взаимодействия. Вводятся правила, которые описывают, как участники осуществляют переход между состояниями и как новые сообщения поступают в сеть, а также поведение противника. Эти правила авторами подробно не описываются, только упоминается, что описание противника выглядит довольно сложным.

Athena

Athena (D. Song, S. Berezin, A. Perrig, 2001) интегрирует подходы на основе проверки на модели и автоматического доказательства теорем. Как и NRL Protocol Analyzer, *Athena* работает в обратном направлении, стартуя из запрещенного состояния, и пытается выявить начальные условия, необходимые для достижения этого состояния. Как и в NRL Protocol Analyzer, состояния поддерживаются как можно более абстрактными, и Athena старается соединить это состояние с правой стороной правила. В случае NRL Protocol Analyzer — это правило переписывания, в случае *Athena* — это правило вывода. По мере продвижения поиска состояния конкретизируются в том смысле, что все больше переменных становятся связанными.

Athena использует расширение модели пространства нитей (*strand space*) (F. Thayer Fabrega, J. Herzog, and J. Guttman, 1998) в качестве модели вычислений. Благодаря простоте и интуитивности этой модели средство *Athena* обладает высокой эффективностью.

Isabelle

L. C. Paulson в 1998 г. исследовал применение *Isabelle*, которое является инструментом для доказательства теорем универсального назначения, для доказательства корректности протоколов.

Подобно моделям, используемым в Murφ и NRL Protocol Analyzer, протокол в *Isabelle* специфицируется с помощью набора правил, которые описывают поведение подлинных участников протокола. Эти правила описывают, при каких обстоятельствах участник создает и посылает сообщение.

Murφ и NRL Protocol Analyzer используют эти правила для описания состояния, в которое переходит система, когда совершается некое действие или посылается сообщение. В противоположность этому, здесь эти правила используются для индуктивного определения множества возможных траекторий.

Каждое правило имеет форму: «если некая траектория содержит определенные события, тогда она может быть продолжена с помощью добавления определенного нового события в ее конец». Благодаря тому, что дается индуктивное определение для множества трасс в протоколе, доказательство корректности справедливо для произвольного числа протокольных сессий, а не только для модели с конечным числом состояний.

Однако, как и в случае NRL Protocol Analyzer, это средство не гарантирует завершения процесса анализа. К тому же не ясно, как получить информацию о возможных ошибках в протоколе, для которого доказательство корректности не проходит. Тем не менее это средство хорошо подходит для проектировщиков протоколов в целях их отладки.

Все средства, основанные на методе доказательства теорем, имеют общий недостаток, потому что требуют тесного взаимодействия с пользователем и глубокого проникновения в суть проблемы при верификации протоколов.

REVERE

Revere (Kindred, 1999) является инструментом для автоматизированного анализа протоколов, который основан на применении логики доверия RV (от *Responsibility Verification*), обобщающей логику доверия BAN. В ее основе лежит алгоритм генерации теории (*theory generation*), предназначенный для получения конечного представления логики доверия, являющегося по сути полным списком всех утверждений, выводимых из описания протокола и исходных предположений. Сравнивая такие представления для двух различных протоколов, можно выделять утверждения, справедливые для одного из протоколов и не справедливые для другого, и, тем самым, выяснять их индивидуальные особенности и уязвимости. Поскольку этот подход является разновидностью метода доказательства теорем, данное средство дает возможность доказывать положительные утверждения о протоколах, но не находит контрпримеры.

AVISPA

В начале осени 2005 г. появился новый программный продукт — AVISPA (*Automated Validation of Internet Security Protocols and Applications*) (A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, и др.) (http://www.avispa-project.org), разработанный в рамках международного проекта с участием LORIA-INRIA (Франция), ETH Цюрих (Швейцария), Ai-Lab Univ. Genova (Италия), Siemens AG (Германия).

Главное преимущество AVISPA состоит в том, что ее применение позволяет не только определить, есть ли недостатки у конкретного протокола, но и найти атаки на данный протокол, если это возможно. Интегрирует различные современные подходы к анализу протоколов, такие, как проверка на модели (model-checking), древовидные автоматы, темпоральная логика. При этом используются последние достижения, полученные после 2000 г. В отличие от других средств, исполняемый программный код этого средства доступен через Интернет на сайте http://www.avispa-tool.org.

В настоящее время AVISPA включает четыре выходных модуля (рис. 1):

— OFMC (On-the-fly Model-Checker),

— CL-AtSe (CL-based Attack Searcher),

— SATMC (SAT-based Model-Checker),

— TA4SP (Tree Automata-based Protocol Analyser).

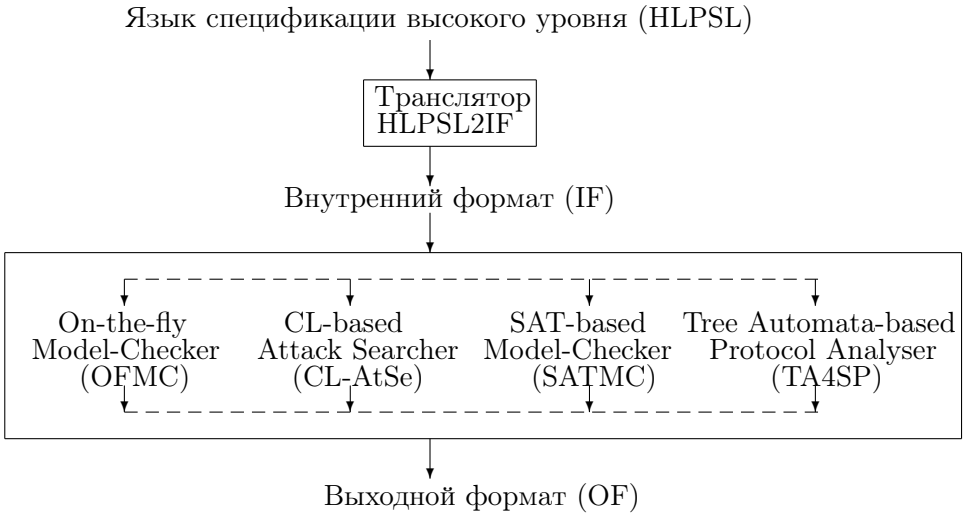

Рис. 1. Архитектура AVISPA

Данные модули частично дублируют и взаимно дополняют друг друга. Первые три модуля (CL-AtSe, OFMC и SATMC) реализуют верификацию протокола путем проверки на модели. Приведем их краткое описание. Модуль TA4SP использует технику, разработанную для систем автоматического доказательства.

Модуль CL-AtSe

CL-AtSe (*constraint-Logic-based Model-Searcher*) — программа верификации, основанная на логике ограничений. Выполняет трансляцию произвольной спецификации протокола, записанной в терминах отношений переходов (*transition relation*) на языке IF, в множество ограничений (*constraints*), обязывающих нарушителя к выполнению определенных действий, которое может быть эффективно использовано для нахождения атак на протоколы. Как трансляция, так и проверка выполняются полностью автоматически внутри блока CL-AtSe.

Каждый шаг протокола моделируется ограничениями, которые заносятся в список знаний нарушителя. Например, сообщение, полученное честным участником, рассматривается как ограничение, обязывающее нарушителя к использованию возможности для подделки. С каждым шагом протокола осуществляется добавление новых ограничений для системы и сокращение/исключение других ограничений соответственно. Затем на каждом шаге состояние системы тестируется на предмет выполнения необходимого множества свойств безопасности.

Алгоритм анализа, применяемый в CL-AtSe, построен так, что он способен обрабатывать ограниченное число циклов, то есть ограниченное число шагов для любой траектории.

В процессе чтения IF файла CL-AtSe пытается с помощью выполняемых по умолчанию процедур упростить спецификацию протокола с целью сократить общее число шагов протокола, подлежащих проверке. Основная идея этого сокращения вариантов перебора состоит в предварительном выделении тех шагов протокола, которые можно выполнить в конце, и тех, которые надо выполнить немедленно, и последующем учете этой информации при проведении перебора. В заключение CL-AtSe пытается создать хорошо читаемое для человека описание атаки, если таковая найдена.

Модуль OFMC

Выходной модуль OFMC (*On-the-Fly Model-Checker*) (D. Basin, S. Mödersheim, and L. Viganò, 2003) разработан в Information Security Group ETH Zurich, является анализатором моделей для анализа протоколов защиты. Он комбинирует два метода:

— использование инертных (*lazy*) типов данных, позволяющих редуцировать бесконечное пространство состояний до конечного;

– применение символического метода для моделирования злоумышленника в модели Долева — Яо, чьи действия воспроизводятся оперативно в стиле управления по запросу (*demand-driven way*), т. е. на ходу.

Для сокращения числа вариантов поиска строится редуцированная конечная модель за счет рассмотрения бесконечных типов данных (таких, как потоки и деревья) и оперирования со свободными переменными. Использование таких типов данных, иначе называемых *инертными (lazy) типами данных*, позволяет редуцировать бесконечное пространство состояний до конечного.

Авторы формализовали применение символического метода для того, чтобы значительно сократить пространство состояний без исключения из рассмотрения каких-либо атак. Подход, названный авторами *методом инертного злоумышленника*, использует символическое представление. Сообщения злоумышленника представляются выражениями с переменными, значения которых не фиксируются.

OFMC может быть применен:

— не только для быстрого определения атак,

— для доказательства корректности протокола при ограниченном числе сессий.

Более того, OFMC имеет встроенную поддержку для алгебраических уравнений, а именно с использованием OFMC может быть проанализирован протокол с ключевым обменом по протоколу Диффи — Хеллмана; причем в настоящее время разрабатывается более общая поддержка алгебраических свойств (включая определенные самим пользователем).

При конкретной реализации метода проверки протокола с применением техники инертных состояний необходимо решить две проблемы: в каком порядке применять соответствующие правила, как реализовать приведение ограничений к редуцированному виду. В OFMC разработан новый подход, основанный на символическом задании сеансов протокола: вводится символическое представление инертного нарушителя вместо перенумерации сеансов протокола; вместо имен вводятся роли участников, обозначаемые переменными. Затем в процессе поиска переменные подвергаются унификации и принимают значения, равные именам конкретных участников, либо остаются переменными.

Модуль SATMC

SATMC (*SAT-based Model-Checker*) (M. Abadi, V. Cortier, 2004) строит пропозициональную формулу, кодирующую отношения переходов, специфицированные с помощью языка IF, начальное состояние и множество состояний, представляющих различные варианты свойств безопасности. В результате компиляции спецификации IF с помощью модуля SATMC осуществляется преобразование комбинации указанных проблем безопасности в задачи планирования, для решения которых используется современная техника решения задач выполнимости, разработанная для проблем планирования. Пропозициональная формула подается на вход алгоритма проверки выполнимости (SAT-солвера), причем любое найденное решение преобразуется обратно в атаку.

При разработке SATMC основное внимание было уделено гибкости, модульности и эффективности. Результатом этого является открытая и гибкая платформа для основанной на задаче выполнимости ограниченной проверки моделей протоколов безопасности. Например, продвижение технологии SAT может быть осуществлено путем интеграции с современными SAT-солверами (например, дающими наилучшую производительность в соревновании SAT средств, о которых можно узнать на сайте http://www.satlive.org). Аналогично, преимущества и новые подходы в AI планировании и технике SAT-редуцирования могут быть непосредственно применены в SATMC.

SATMC можно применять не только для открытия новых атак на протоколы, но и для верификации при ограниченном числе сессий — проблемы, которая, как было доказано M. Rusinowitch и M. Turuani, 2001, принадлежит тому же классу, что и проблема выполнимости, то есть NP-полная.

На вход выходного модуля (рис. 2) подается спецификация протокола и проблемы на языке IF, которая затем перекомпилируется с учетом линейного кодирования в проблему планирования. Далее осуществляется попытка решить проблему планирования с помощью какого-либо современного средства проверки выполнимости. Далее в зависимости от результата проверки либо процесс проверки продолжается при измененных параметрах, либо выдается результат — свойство выполнено, или же найденное решение анализируется и по нему выписывается найденная атака.

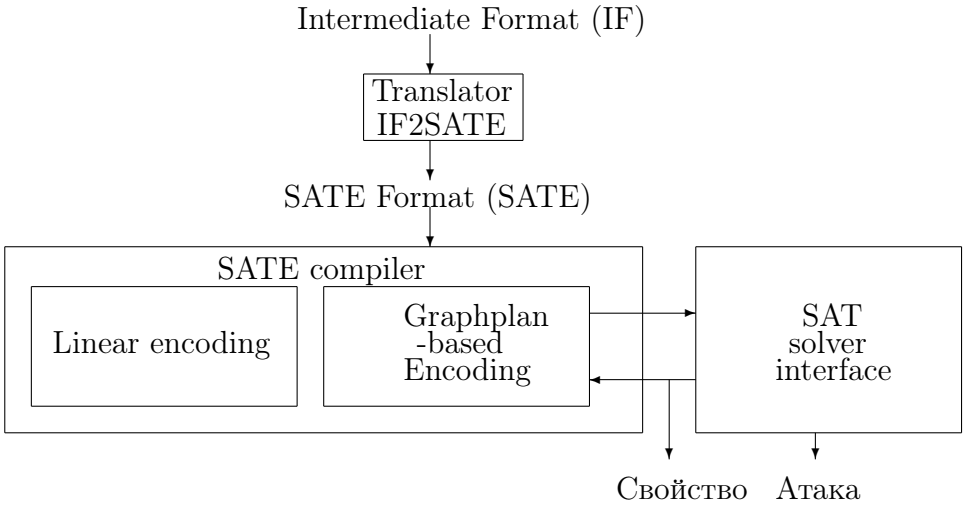

Рис. 2. Блок-схема выходного модуля SATMC

Модуль TA4SP

Модуль TA4SP вычисляет для заданного начального состояния как верхнюю, так и нижнюю аппроксимацию знаний нарушителя с помощью правил переписывания с применением техники аппроксимации древовидных автоматов (языков) при неограниченном числе сессий.

TA4SP использует библиотеку древовидных автоматов Timbuk 2.0, созданную Th. Genet IRISA, Rennes France и доступную на сайте http://www.irisa.fr/lande/genet-/timbuk/.

Верхняя оценка может привести к реальному доказательству проверяемого свойства безопасности для изучаемого протокола при неограниченном числе сессий.

В противном случае, в контексте верхнего оценивания TA4SP может только делать вывод, что свойство конфиденциальности выполняется только для данного начального состояния.

При нижнем оценивании без каких-либо абстракций средство может показать, что протокол не удовлетворяет данному свойству безопасности.

Для проверки протокола с помощью TA4SP применяется следующая эмпирическая стратегия.

1. Пользователь вычисляет верхнюю оценку и проверяет свойство безопасности.

2. Если первый шаг не позволяет сделать вывод о выполнении свойства безопасности, то пользователь вычисляет нижнюю оценку до тех пор, пока не будет получена атака в приемлемое время. Однако эта эмпирическая стратегия не всегда приводит к ожидаемому результату.

Промежуточный результат на основе верхнего оценивания не позволяет сделать вывод, что атака действительно существует.

Сочетает преимущества систем автоматического доказательства теорем с формальной абстрактной интерпретацией, называемой аппроксимацией.

Обладает многими важными для практики свойствами:

— не требуется завершение работы систем переписывания;

— системы переписывания могут включать AC-символы;

— доказательства получаются с помощью выполнения операции пересечения с аппроксимирующим автоматом $\mathcal{T}_{\mathcal{R}\uparrow}(\mathcal{A}_0)$ (что быстро выполняется в автоматическом режиме);

— построение $\mathcal{T}_{\mathcal{R}\uparrow}(\mathcal{A}_0)$ также выполняется автоматически, монотонно, причем можно гарантировать его завершение при подходящем выборе аппроксимационной функции γ.

Построение функции аппроксимации не требует никакого особого мастерства и специальных знаний из теории формального доказательства, так как оно заключается в указании некоторых множеств объектов, представляемых распознающими регулярные множества термов состояниями, которые надо объединять вместе для построения аппроксимирующей модели. Пока этот шаг выполняется вручную, но в скором времени он будет автоматизирован.

Scyther

Создан в ETH (Цюрих). Верифицирует ограниченное и неограниченное число сеансов протокола. Использует символический анализ в сочетании с обратным поиском, основанный на частично упорядоченных шаблонах (C. J. F. Cremers, 2006).

Scyther не требует задания сценария атаки. Необходимо только задать параметры, ограничивающие либо максимальное число запусков ($MaxRuns(n)$), либо пространство перебираемых траекторий ($Traces$).

В первом случае даже для маленьких значений n всегда дает результат и показывает найденные траектории атаки.

Во втором случае завершение не гарантировано. По умолчанию *Scyther* использует значение $MaxRuns(5)$, что гарантирует завершение, причем в качестве ответа получаем одну из трех ситуаций:

— установлено, что проверяемое свойство выполнено для $MaxRuns(5)$;

— свойство не выполнено, так как найдена атака;

— свойство может быть корректно для пространства траекторий $Traces$.

HERMES

Архитектура HERMES представлена на рис. 3. Он позволяет анализировать ограниченное и неограниченное число сеансов протокола.

Разработан под руководством Liana Bozga французским исследовательским центром VERIMAG из IMAG в рамках объединенного французского проекта EVA (*Explication et Verification Automatique pour les Protocoles Cryptographiques*) (http://www-eva.imag.fr/). Впервые был представлен на конференции CAV в 2003 г., доступ к нему в on-line появился в 2006 г. В настоящее время работа над ним приостановлена. Имеется две работы, описывающих особенности функционирования HERMES, однако нет теоретического обоснования принципа его действия. Исполняемых файлов, а также исходных кодов программы в открытом доступе также нет. Доступен в интерактивном режиме on-line на сайте http://www-verimag.imag.fr/Liana.Bozga/eva/hermes.php.

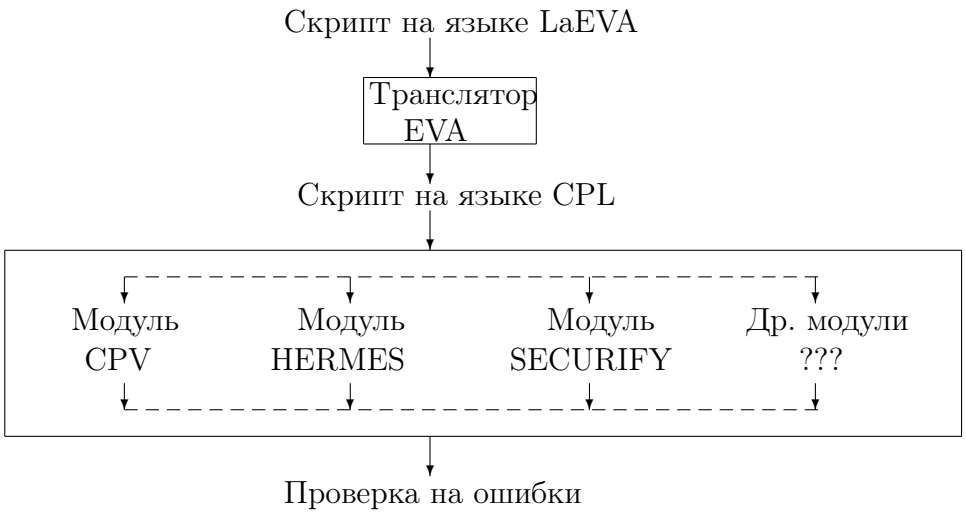

Рис. 3. Архитектура HERMES

HERMES использует язык LaEva верхнего уровня для задания спецификаций протоколов и их свойств, аналогичный языку HLPSL. Затем этот язык транслируется во внутреннее представление на языке *cpl*, которое является общим языком для нескольких продуктов, таких, как SECURIFY (2001), Cpv (2000) и HERMES.

Для заданных протокола и проверяемого свойства HERMES дает на выходе условия на исходные знания противника, которые гарантируют, что он не сможет узнать секрета. Работает при неограниченной длине сообщения и неограниченном числе участников. Строит инварианты для знаний противника, а не все множество сообщений, представляющее знания противника. Если в результате найдена атака, то предоставляет траекторию атаки. В случае, когда в результате получается ответ, что свойство доказано, выдает также дерево полного доказательства, что бывает полезным при сертификации протокола.

HERMES основывается на понятии безопасного сообщения для данного протокола, которое защищает секрет K (K-защищающее). Пусть $M(K)$ — множество сообщений, содержащих секрет. Идея, которая лежит в основе алгоритма, состоит в том, чтобы описать такое подмножество $M^* \subset M(K)$, которое позволит не раскрыть секреты, содержащиеся во всех посылаемых сообщениях. Сообщения из множества $M(K) \setminus M^*$,

которые не защищают секреты, назовем K-незащищающими. Оно вычисляется, уже непосредственно отталкиваясь от самой модели протокола.

Таким образом, в отличие от уже имеющихся методов проверки на модели, в которых каждое формализованное состояние протокола проверяется на удовлетворение свойству безопасности, HERMES проверяет сообщения — защищают ли они секрет или нет. После завершения работы алгоритма на выходе получаем множество секретов S и множество защищающих сообщений H.

Язык EVA, который используется в HERMES, основан на языке CAPSL и сам по себе легко читаем и довольно прост. Непрограммист может легко разобраться в особенностях языка, синтаксис языка несложен и интуитивно понятен. Имея в качестве примеров два-три формализованных протокола, новые протоколы задаются довольно легко, так как многие куски кода можно использовать практически без изменений. Поэтому с этих позиций у языка EVA налицо преимущество перед языком в AVISPA.

С другой стороны, HLPSL позволяет выражать свойства безопасности глубже и лаконичнее. Например, в EVA не существует терминов для задания различных видов сред, что может стать серьезным препятствием при формализации некоторых протоколов.

Сравнение HERMES и AVISPA

HERMES:

— легче в использовании;

— сложнее интерпретировать полученные результаты в случае обнаружения недостатков в протоколах. Даже малейшая ошибка в протоколе приводит к десяткам непонятных математических правил;

— ошибки в синтаксисе обнаруживаются непосредственно перед компиляцией;

— в случае безопасного протокола не выдается никаких сведений о том, с помощью каких инструментов был получен этот результат;

— протокол легче формализовать, но сложнее трактовать полученные результаты.

AVISPA:

— язык сложнее, но гибче, позволяет детально задавать протокол;

— требование задать свойства среды также усложняет процедуру проверки протокола;

— процедура отладки, в свою очередь, сложнее, чем в EVA;

— результаты, полученные с помощью AVISPA, более подробны и конструктивны;

— интерпретация полученных результатов не требует специальных навыков или большого опыта, в отличие от HERMES, в котором опыт играет значительную роль.

Таким образом, HERMES больше подходит для анализа простых протоколов, использующих стандартные криптографические примитивы; AVISPA стоит использовать для анализа сложных и запутанных протоколов (можно самому задавать модель нарушителя).

ProVerif

Разработан в рамках проекта, финансируемого INRIA. Анализирует неограниченное число сеансов протокола с использованием верхней аппроксимации и представления протокола с помощью хорновских выражений. Архитектура *ProVerif* представлена на рис. 4.

ProVerif (B. Blanchet, 2001) 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.

Позволяет в автоматическом режиме проверять свойство секретности (конфиденциальности), свойство аутентификации.

На выходе возможны четыре ситуации:

— свойство не выполнено;

— доказано, что свойство выполнено;

— есть пример атаки (могут быть найдены ложные атаки);

— работа не завершается.

Заметим, что *ProVerif* корректно моделирует множество траекторий, соответствующих определенному сценарию, и поэтому осуществляет полный перебор возможных траекторий.

Примеры, найденные в результате работы *ProVerif*, как правило, не требовали задания пространства траекторий $Traces$.

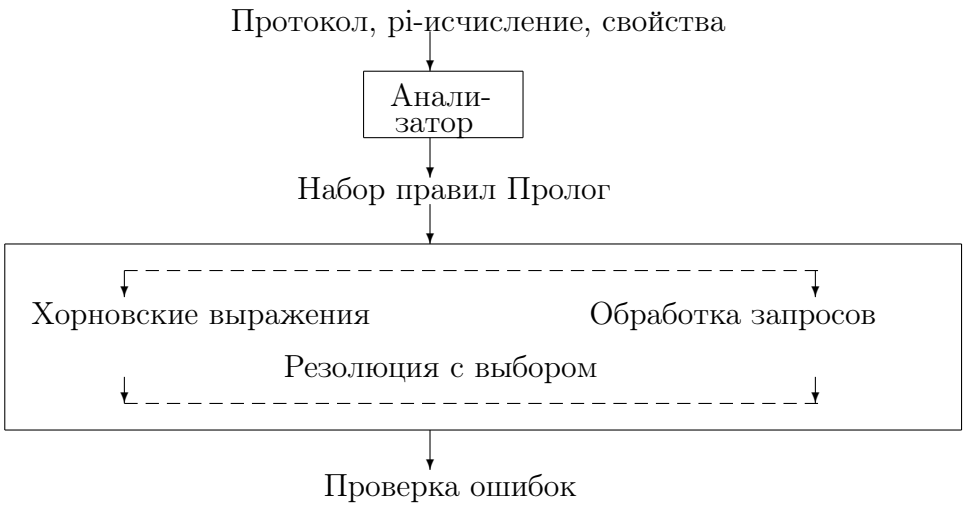

Рис. 4. Архитектура ProVerif

Сравнение AVISPA, ProVerif, Scyther

В заключение приведем диаграммы из работы [5], показывающие в логарифмической шкале трудоемкость проверки свойства секретности на примере протокола Нидхэма — Шредера с помощью программных средств AVISPA, ProVerif, Scyther (см. рис. 5).

Как видно на рис. 5, наибольший рост трудоемкости при увеличении числа параллельно открытых сеансов протокола наблюдается при использовании модуля SATMC (AVISPA). При использовании модулей ClStCe и OFMC также наблюдается рост трудоемкости. Модуль TA4SP дает постоянную трудоемкость, но большую, чем при использовании ProVerif, Scyther. Невозрастание трудоемкости в этих случаях объясняется тем, что они осуществляют проверку методом, не зависящим от числа открытых сеансов протокола.

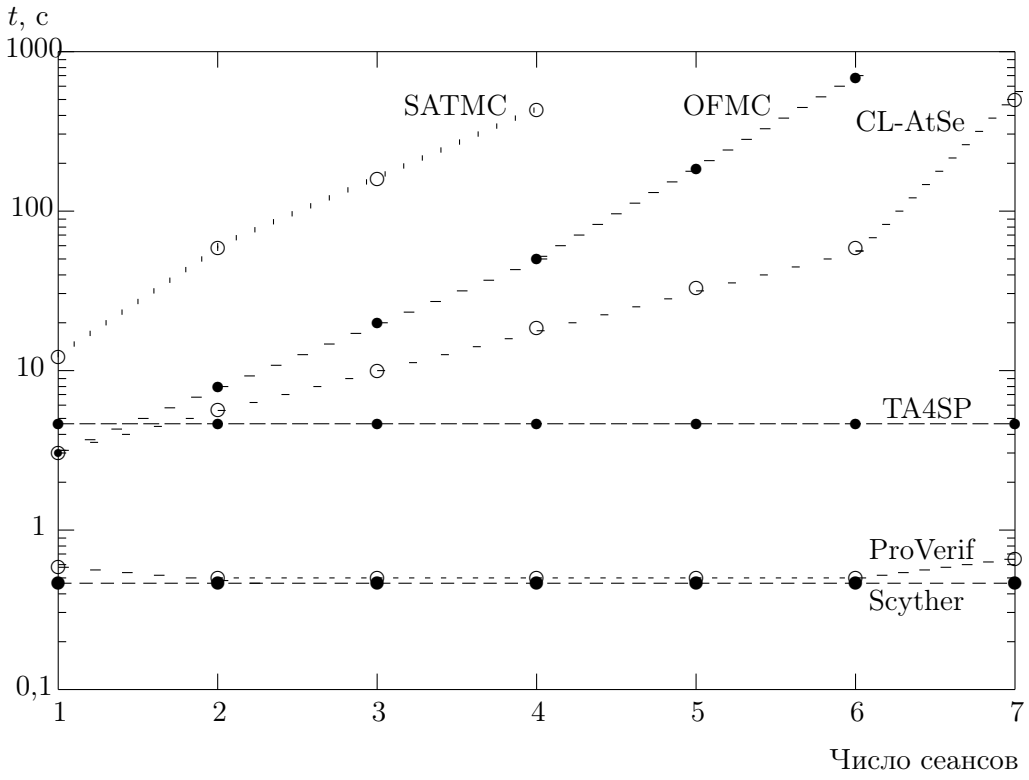

Рис. 5. Сравнение программных средств

ЛИТЕРАТУРА

1. *Черемушкин А. В.* Криптографические протоколы. Основные свойства и уязвимости: учебное пособие. М.: Изд. центр «Академия», 2009. 272 с.
2. Index of the security protocols repository (SPORE) // Laboratorie Spécification et Vérification. http://www.lsv.ens-cachan.fr/spore/table.html.
3. *Clark J., Jacob J.* A Survey of Authentication Protocol Literature: Version 1.0. 17 Nov. 1997. http://www.cs.york.ac.uk/ jac/papers/drareview.ps.gz, 1997.
4. *Menezes A. J., van Oorschot P. C., Vanstone S. A.* Handbook of applied cryptography. Boca Raton, New York, London, Tokyo: CRC Press, 1997. 780 p.
5. *Cremers C. J. F., Lafourcade P.* Comparing State Spaces in Automatic Security Protocol Verification. ETH Technical Report. 2007. No. 558. 26 p.