Новая математическая модель протоколов аутентификации и основанный на ней метод верификации

Миронов Андрей Михайлович

Протоколы аутентификации – это распределенные алгоритмы, предназначенные для обеспечения аутентификации агентов и передачи конфиденциальной информации (криптографических ключей, и т.п.) в небезопасной среде. Они используются, например, в электронных платежах, электронных процедурах голосования, системах доступа к базам данных, и т.д. Учитывая большой финансовый и социальный ущерб в случае неправильной работы таких протоколов, необходимо использовать математические методы для обоснования их корректности и безопасности. В настоящей работе вводится новая математическая модель таких протоколов аутентификации, позволяющая описывать как сами протоколы, так и их свойства. Показывается, как на базе данной модели можно решать задачи верификации протоколов аутентификации.

Ключевые слова: протоколы аутентификации, распределенные алгоритмы, верификация.

1. Введение

Протоколы аутентификации (ПА) – это распределенные алгоритмы, используемые для защиты электронных транзакций. ПА представляет собой описание порядка обмена сообщениями между несколькими агентами. Примеры таких агентов – компьютерные системы, банковские карточки, люди, и т.д. Для обеспечения свойств безопасности, (таких например как конфиденциальность передаваемых данных) в ПА используются криптографические преобразования (шифрование, электронная подпись, хэш-функции, и т.п.). Мы предполагаем, что криптографические преобразования, используемые в ПА, являются идеальными, т.е. удовлетворяют некоторым аксиомам, выражающим, например, невозможность

извлечения открытых текстов из шифртекстов без знания соответствующих криптографических ключей.

Наиболее известным ПА является протокол Kerberos [1]. Исходное описание данного протокола содержало уязвимости. Данные уязвимости были обнаружены формальными методами и исправлены, после чего новая версия данного протокола была формально верифицирована [2].

Есть много других ПА, используемых, например, для аутентификации перед провайдерами мобильной телефонной связи, для снятия денег в банкомате, и т.п. Цифровизация современного общества требует более широкого использования ПА для решения самых разных задач, таких например как работа с электронными паспортами (которые могут включать в себя чипы RFID), проведение электронных выборов (в которых предполагается голосование по интернету), и т. д.

Многие уязвимости ПА связаны не с плохими криптографическими качествами используемых в них криптографических примитивов, а с логическими ошибками в протоколах. Например, в ПА входа в портал Google, позволяющем пользователю идентифицировать себя только один раз, а затем обращаться к различным приложениям (таким, например, как Gmail или календарь Google), обнаружена логическая ошибка, позволяющая нечестному поставщику услуг выдавать себя за любого из своих пользователей для другого поставщика услуг.

Существует много других примеров ПА, которые продолжительное время использовались в критических по безопасности системах, однако затем обнаружилось что эти ПА содержат уязвимости следующего вида:

- агенты, участвующие в этих ПА, могут получать искаженные сообщения (или вообще терять их) в результате перехвата, удаления или искажения противником передаваемых сообщений, в результате чего нарушается свойство целостности,
- противник может узнать конфиденциальную информацию, содержащуюся в перехваченных сообщениях, в результате ошибочных или злонамеренных действий участников ПА.

Эти примеры являются обоснованием того что для ПА, используемых в критических по безопасности системах, недостаточно неформального анализа требуемых свойств, необходимо

- построить **математическую модель** анализируемого ПА,
- описать проверяемые свойства анализируемого ПА в виде математического объекта, называемого **спецификацией** этого ПА, и

- построить доказательство утверждения о том, что анализируемый ПА удовлетворяет (или не удовлетворяет) спецификации, процедура построения такого доказательства называется **верификацией** анализируемого ПА.

Исторически первым формальным аппаратом для математического анализа ПА была логика Бэрроуза-Абади-Нидхэма [3]. Данный аппарат имеет очень большие ограничения, в частности он не позволяет рассматривать случай неограниченного порождения сеансов анализируемого протокола.

Более популярным подходом к верификации ПА является формализм пространств нитей (strand spaces), развитый в работах сотрудников корпорации MITRE F. Javier Thayer Fabrega, Jonathan C. Herzog, Joshua D. Guttman [4]. Данный формализм также ограничен невозможностью верификации протоколов, порождающих неограниченное число сеансов.

Ссылки на работы, посвященные описанию различных формализмов, предназначенных для моделирования и верификации ПА, можно найти на странице курса по верификации криптопротоколов профессора Университета Эйндховена Jerry den Hartog [5]. См. также [6].

Одним из формализмов верификации ПА является подход, связанный с использованием клауз Хорна и систем уравнений с ограничениями (Constraint Systems), развитый в работах Абади, Блаше, Кортье и других специалистов (приведем лишь ссылку на современный вводный текст с изложением этого формализма [7], в нем содержатся многочисленные ссылки на недавние научные работы в этой области). И этот формализм имеет свои ограничения, главное из которых – нахождение ложных атак на анализируемый протокол. Говоря неформально, в данном подходе каждой паре (a_1, a_2) соседних действий участников протокола, первое из которых (a_1) - прием сообщения, а второе (a_2) - посылка сообщения, сопоставляется правило вывода, заключающееся в том, что если противник послал участнику какое-либо сообщение, совместимое с форматом сообщения в действии a_1, то он может получить от этого участника сообщение, соответствующее действию a_2, т.е. участники рассматриваются как автоматы с заданной реакцией. Фундаментальным недостатком такого подхода является отсутствие в модели противника хронологической связи между этими парами, о чем авторы [7] пишут на стр. 91 (см. пункт 8.2.4: Approximations), что приводит к тому что некоторые атаки на протокол, обнаруженные в этой модели, являются ложными. Судя по недавним публикациям авторов данного подхода, у них нет идей по поводу того, как можно было бы избавиться от данного недостатка.

В настоящем тексте вскрыта главная причина недостатка этой модели, она заключается в том что каждое правило вывода, соответствующее паре действий "ввод-вывод" какого-либо участника протокола может быть применено к состоянию s противника только с учетом "предыстории", хранящей последовательность действий, приводящих противника в состояние s. Настоящий текст посвящен построению математической модели ПА, реализующей описанный выше подход. Показывается, как на базе данной модели можно решать задачи описания свойств ПА. Одно из наиболее важных достоинств предложенной модели ПА заключается в высокой степени автоматизации решения задачи верификации ПА на основе данной модели.

2. Вспомогательные понятия

2.1. Термы

Мы будем предполагать, что заданы следующие множества:

- множество $\mathcal{T}$, элементы которого называются **типами**,
- множества $\mathcal{X}$ и $\mathcal{C}$, элементы которых называются **переменными** и **константами** соответственно, причем каждой переменной или константе e сопоставлен некоторый тип $t(e) \in \mathcal{T}$,
- множество $\mathcal{F}$, элементы которого называются **функциональными символами (ФС)**, причем каждому ФС $f \in \mathcal{F}$ сопоставлен тип $t(f)$, представляющий собой запись вида $(t_1, \ldots, t_n) \to t$, где $t_1, \ldots, t_n, t \in \mathcal{T}$.

Термы строятся из переменных, констант и ФС. Множество всех термов обозначается символом $\mathcal{E}$. Каждому терму e сопоставлен тип $t(e) \in \mathcal{T}$, определяемый структурой терма e. Правила построения термов имеют следующий вид:

- каждая переменная и константа является термом того типа, который сопоставлен этой переменной или константе, и
- если $e_1, \ldots, e_n \in \mathcal{E}$, $f \in \mathcal{F}$, и $t(f)$ имеет вид $(t(e_1), \ldots, t(e_n)) \to t$, то знакосочетание $f(e_1, \ldots, e_n)$ является термом типа t.

Будем считать, что $\mathcal{T}$ содержит следующие типы:

- `agent`, термы этого типа называются **именами агентов** (или просто **агентами**), они обозначают имена участников ПА,
- `key`, термы этого типа называются **ключами**, они обозначают криптографические ключи, которые участники ПА могут использовать для шифрования или дешифрования сообщений,
- `message`, термы этого типа называются **сообщениями**, они обозначают сообщения, которые участники ПА могут пересылать друг другу в процессе своей работы.
- `start`, `end`, $\texttt{initiator}_i$, $\texttt{initiator}_r$, $\texttt{responder}_i$, $\texttt{responder}_r$, $\texttt{value}_i$, $\texttt{value}_r$, данные типы носят служебный характер, существует единственная переменная каждого из этих типов, мы будем обозначать каждую из этих переменных той же записью, которой обозначается соответствующий ей тип.

Будем использовать перечисляемые ниже ФС.

- ФС h, h_i (где i – индекс) типа $(\texttt{message}) \to \texttt{message}$, данные ФС являются именами **хэш-функций (ХФ)**.
- ФС $tuple_n$ (где n – произвольное натуральное число) типа

$$(\underbrace{\texttt{message}, \ldots, \texttt{message}}_{n}) \to \texttt{message}.$$

  Терм вида $tuple_n(e_1, \ldots, e_n)$ будет обозначаться более короткой записью $(e_1, \ldots, e_n)$ и называться **списком** термов $e_1, \ldots, e_n$.
- ФС $pr_{n,i}$ (где n – произвольное натуральное число, и $i \in \{1, \ldots, n\}$) типа $(\texttt{message}) \to \texttt{message}$.

  Терм вида $pr_{n,i}(e)$ будет обозначаться записью $e_{(i)}$.

  Терм вида $(e_1, \ldots, e_n)_{(i)}$, где $1 \le i \le n$, будет называться **i–й компонентой списка** $(e_1, \ldots, e_n)$ и считаться равным терму e_i.
- ФС enc и dec типа $(\texttt{key}, \texttt{message}) \to \texttt{message}$.

  Терм вида $enc(k, e)$ (или $dec(k, e)$) обозначает сообщение, получаемое шифрованием (или дешифрованием, соответственно) сообщения e на ключе k.

  Терм вида $enc(k, e)$ будет обозначаться записью $k(e)$ и называться **шифрованным сообщением (ШС)**.

- ФС pk и sk типа $(\texttt{agent}) \to \texttt{key}$.

  Терм вида $pk(a)$ (или $sk(a)$) обозначает **открытый** (или **закрытый**, соответственно) ключ агента a.

  Будем предполагать, что

  - ключ вида $pk(a)$ может использоваться любым агентом для шифрования сообщений, и
  - ключ вида $sk(a)$ может использоваться только агентом a и предназначен для дешифрования сообщений, зашифрованных на ключе $pk(a)$.

  Терм вида $enc(pk(a), e)$ будет обозначаться записью $a(e)$.

  Термы вида $pk(a)$ и $sk(a)$ будут обозначаться записями вида a^+ и a^- соответственно.

- ФС key типа $(\texttt{key}) \to \texttt{message}$.

  Терм вида $key(k)$ обозначает сообщение, совпадающее с ключом k, термы такого вида используются для обозначения действий, связанных с передачей ключей.

  В записи терма вида $key(k)$ ФС key и скобки будут опускаться.

- ФС $sign$ типа $(\texttt{message}, \texttt{agent}) \to \texttt{message}$.

  Терм вида $sign(e, a)$ обозначает **цифровую подпись** сообщения e, сделанную агентом a.

  Тройка вида $(e, a, sign(e, a))$ будет обозначаться записью $\langle e \rangle_a$.

$\forall\, e \in \mathcal{E}$ запись X_e обозначает множество переменных, входящих в e.

$\forall\, X \subseteq \mathcal{X}$ запись $\mathcal{E}_X$ обозначает множество $\{e \in \mathcal{E} \mid X_e \subseteq X\}$.

Будем называть **подстановкой** произвольную частичную функцию $\theta : \mathcal{X} \to \mathcal{E}$, такую, что $\forall\, x \in \mathcal{X}$, если $\theta(x)$ определено, то $t(\theta(x)) = t(x)$. Ниже символ Θ обозначает множество всех подстановок, и $\forall\, \theta \in \Theta$

$$dom(\theta) \stackrel{\text{def}}{=} \{x \in \mathcal{X} \mid \theta(x) \text{ определено}\}.$$

$\forall\, X \subseteq \mathcal{X}$ запись Θ_X обозначает множество $\{\theta \in \Theta \mid dom(\theta) \subseteq X\}$.

$\forall\, \theta \in \Theta, \forall\, e \in \mathcal{E}$, запись θe обозначает терм определяемый рекурсивно: если $e = x \in dom(\theta)$, то $\theta e \stackrel{\text{def}}{=} \theta(x)$, если $e = x \in \mathcal{X} \setminus dom(\theta)$, то $\theta e \stackrel{\text{def}}{=} x$, если $e = c \in \mathcal{C}$, то $\theta e \stackrel{\text{def}}{=} c$, и если $e = f(e_1, \ldots, e_n)$, то $\theta e \stackrel{\text{def}}{=} f(\theta e_1, \ldots, \theta e_n)$.

$\forall\, \theta_1, \theta_2 \in \Theta$ записи $\theta_1 \subseteq \theta_2$ и $\theta_2 \supseteq \theta_1$ означают, что $dom(\theta_1) \subseteq dom(\theta_2)$, и $\forall\, x \in dom(\theta_1)\ \ \theta_1(x) = \theta_2(x)$.

2.2. Формулы

Понятие формулы определяется индуктивно следующим образом:

- если $e, e' \in \mathcal{E}$, то запись $[\![e = e']\!]$ является формулой,
- если $\beta_1, \ldots, \beta_n$ – формулы, то запись $\beta_1 \wedge \ldots \wedge \beta_n$ является формулой.

Произвольная формула обозначается символом β (возможно, с индексами). Множество всех формул обозначается символом $\mathcal{B}$.

$\forall\, e, e' \in \mathcal{E}, \forall\, \beta \in \mathcal{B}$ запись $[\![e = e']\!] \in \beta$ означает, что формула β содержит конъюнктивный член $[\![e = e']\!]$.

$\forall\, \beta \in \mathcal{B}$ запись X_β обозначает множество переменных, входящих в β.

$\forall\, X \subseteq \mathcal{X}$ запись $\mathcal{B}_X$ обозначает множество $\{\beta \in \mathcal{B} \mid X_\beta \subseteq X\}$.

$\forall\, \beta \in \mathcal{B}$ запись $\underset{\beta}{=}$ обозначает наименьшую конгруэнцию на $\mathcal{E}$ (относительно операций, соответствующих ФС из $\mathcal{F}$), удовлетворяющую условию: $\forall\, [\![e = e']\!] \in \beta, \forall\, \theta \in \Theta \quad (\theta e, \theta e') \in \underset{\beta}{=}$.

$\forall\, \beta_1, \beta_2 \in \mathcal{B}$ записи $\beta_1 \subseteq \beta_2$ и $\beta_2 \supseteq \beta_1$ означают, что

$$\forall\, [\![e = e']\!] \in \beta_1 \quad (e, e') \in \underset{\beta_2}{=} \tag{1}$$

(нетрудно видеть, что (1) равносильно включению $\underset{\beta_1}{=} \subseteq \underset{\beta_2}{=}$).

Формулы β_1 и β_2 считаются равными, если $\beta_1 \subseteq \beta_2$ и $\beta_2 \subseteq \beta_1$ (т.е. если конгруэнции $\underset{\beta_1}{=}$ и $\underset{\beta_2}{=}$ совпадают).

Нетрудно видеть, что следующие формулы равны:

- $[\![h(e) = h(e')]\!]$ и $[\![e = e']\!]$, где h – имя ХФ,
- $[\![k(e) = k'(e')]\!]$ и $[\![k = k']\!] \wedge [\![e = e']\!]$ где k, k' – ключи.

С каждой подстановкой $\theta \in \Theta$ связана формула, обозначаемая тем же символом θ, и определяемая как конъюнкция формул вида $[\![x = \theta(x)]\!]$, где $x \in dom(\theta)$.

2.3. Замкнутые множества термов

Пусть $E \subseteq \mathcal{E}$ и $\beta \in \mathcal{B}$. Множество E называется β–**замкнутым**, если

1) $\forall\, e \in \mathcal{E}$, если $e = f(e_1, \ldots, e_n)$, где $f \in \mathcal{F}$ и $e_1, \ldots, e_n \in E$, то $e \in E$,
2) $\forall\, e \in E \quad \{e\}_\beta \subseteq E$, где $\{e\}_\beta$ – класс конгруэнции $\underset{\beta}{=}$, содержащий e.

Замкнутые множества термов используются для представления множеств сообщений, которые м.б. известны противнику, и

1) первое из указанных выше условий соответствует операциям, которые противник I может выполнять с имеющимися у него сообщениями, например,

   - если I имеет сообщения e_1, ..., e_n, то он может скомпоновать из них сообщение $(e_1, \ldots, e_n)$,
   - если I имеет сообщение $(e_1, \ldots, e_n)$, то он может получить его компоненты e_1, ..., e_n,
   - если I имеет сообщения k и e, где k – ключ шифрования, то I может создать ШС $k(e)$,
   - если I имеет ШС e и ключ дешифрования k, то он может дешифровать e, т.е. получить $dec(k, e)$, и т.д.,

2) второе из указанных выше условий отражает тот факт, что разные термы могут соответствовать одинаковым сообщениям, и если формула β имеет вид $[\![e_1 = e'_1]\!] \wedge \ldots \wedge [\![e_n = e'_n]\!]$, то она выражает предположение об одинаковости сообщений, соответствующих e_i и e'_i $\forall\, i = 1, \ldots, n$, из которого следует, что включение $\beta \supseteq [\![e = e']\!]$ влечет одинаковость сообщений, представляемых термами e и e', т.е. если противник имеет сообщение, представляемое термом e, то он имеет сообщение, представляемое каждым элементом класса $\{e\}_\beta$.

Нетрудно доказать, что $\forall\, e \in \mathcal{E}$, $\forall\, \beta \in \mathcal{B}$ существует наименьшее (по включению) β–замкнутое множество $cl(e, \beta) \subseteq \mathcal{E}$, такое, что $e \in cl(e, \beta)$.

3. Протоколы аутентификации

3.1. Неформальное понятие протокола аутентификации

Будем рассматривать каждый ПА как распределенный алгоритм, в котором участвуют несколько последовательных процессов. Каждый из этих процессов связан с некоторым агентом. Допускается, что несколько из этих процессов м.б. связаны с одним и тем же агентом. Если в ПА P участвуют процессы $p_1, \ldots, p_n$, то будем обозначать данный факт записью

$$P = (p_1, \ldots, p_n). \tag{2}$$

Выполнение ПА (2) происходит путем выполнения каждого из входящих в него процессов $p_1, \ldots, p_n$. Выполнение каждого из этих процессов заключается в порождении им последовательности действий.

Работа каждого из процессов P_i в протоколе происходит путем последовательного выполнения **внешних действий** процесса P_i, такие действия заключаются в передаче или приеме сообщений, и **внутренних действий** процесса P_i.

4. Действия, процессы и выполнения

Для формального определения понятия протокола аутентификации введем понятия действия, процесса и выполнения процесса.

4.1. Действия

Действие – это запись одного из трех видов: ввод, вывод, внутреннее действие. Понятие действия связано с понятием **выполнения**. В начале выполнения какого-либо действия α определена подстановка θ, называемая **текущей подстановкой**, в результате выполнения этого действия подстановка θ заменяется на подстановку $\theta' \supseteq \theta$, которая будет текущей подстановкой в начале выполнения следующего действия.

Виды действий определяются следующим образом.

- **Ввод** имеет вид $[a?e]$, где a – агент (называемый **отправителем**), и $e \in \mathcal{E}$.

  Выполнение этого действия заключается в

  - получении какого-либо сообщения e' от агента a, и
  - замене текущей подстановки θ на новую текущую подстановку $\theta' \supseteq \theta$, такую, что $\theta' \supseteq [\![e = e']\!]$.

    Если не существует подстановки θ', удовлетворяющей этому условию, то выполнение данного действия невозможно.

- **Вывод** имеет вид $[a!e]$, где a – агент (называемый **получателем**), и $e \in \mathcal{E}$.

  Выполнение этого действия заключается в посылке сообщения e агенту a. Новая текущая подстановка совпадает с θ.

- **Внутреннее действие** имеет вид β, где $\beta \in \mathcal{B}$.

Выполнение этого действия заключается замене текущей подстановки θ на новую текущую подстановку $\theta' \supseteq \theta$, удовлетворяющую условию: $\theta' \supseteq \beta$.

Если не существует подстановки θ', удовлетворяющей этому условию, то выполнение данного действия невозможно.

Множество всех действий обозначается символом $\mathcal{A}$.

4.2. Понятие процесса

В этом параграфе определяется понятие **процесса**. Процессы описывают работу участников ПА.

Процесс – это совокупность $p = (a, S, s^0, R, \beta, e, U)$, где

- a – **агент**, с которым связан процесс p,
- S – множество **состояний**, $s^0 \in S$ – **начальное состояние**,
- $R \subseteq S \times \mathcal{A} \times S$ – множество **переходов**, каждый переход (s, α, s') из R обозначается записью $s \xrightarrow{\alpha} s'$,
- $\beta \in \mathcal{B}$ – **начальное условие**,
- $e \in \mathcal{E}$ – **раскрытый терм** (где под раскрытостью терма понимается предположение о том, что этот терм м.б. известен противнику),
- $U \subseteq \mathcal{X}$ – множество **уникальных переменных** процесса p. (смысл понятия уникальной переменной будет объяснен ниже)

Множество всех процессов обозначается символом $\mathcal{P}$.

$\forall\, p \in \mathcal{P}$ записи a_p, S_p, s^0_p, R_p, β_p, e_p, U_p обозначают соответствующие компоненты процесса p. Множество всех переменных, входящих в p, обозначается записью X_p. Множество всех отправителей и получателей, входящих в p, обозначается записью A_p. Процесс p, такой, что $R_p = \emptyset$ обозначается символом $\mathbf{0}$. Состояние $s \in S_p$ называется **терминальным**, если R_p не содержит переходов вида $s \xrightarrow{\alpha} s'$. Переход $s \xrightarrow{\alpha} s'$ называется **вводом**, **выводом**, или **внутренним переходом**, если α – ввод, вывод, или внутреннее действие, соответственно.

Процесс p можно представлять себе как граф (обозначаемый тем же символом p), вершинами которого являются состояния из S_p, а ребра соответствуют переходам из R_p: каждый переход $s \xrightarrow{\alpha} s'$ соответствует ребру из s в s' с меткой α.

4.3. Понятие выполнения процесса

Выполнение процесса $p \in \mathcal{P}$ можно понимать как обход графа p, начиная с состояния s_p^0, с выполнением действий, которые являются метками проходимых ребер.

Выполнение процесса $p \in \mathcal{P}$ – это граф V вида

$$v_0 \stackrel{\alpha_1}{\to} v_1 \stackrel{\alpha_2}{\to} \ldots \stackrel{\alpha_n}{\to} v_n \quad (n \geq 0) \tag{3}$$

каждое ребро которого помечено некоторым действием $\alpha_i \in \mathcal{A}$, и каждая вершина v которого имеет метку $(s_v, \theta_v, e_v) \in S_p \times \Theta \times \mathcal{E}$, причем $s_{v_0} = s_p^0$, $\theta_{v_0} \supseteq \beta_p$, $e_{v_0} = e_p$, $\forall\, i = 1, \ldots, n \;\; (s_{v_{i-1}} \stackrel{\alpha_i}{\to} s_{v_i}) \in R_p$, $\theta_{v_{i-1}} \subseteq \theta_{v_i}$, и

- если $\alpha_i = [a?e]$ или $[a!e]$, то $e_{v_i} = (e_{v_{i-1}}, e)$,
- если $\alpha = \beta \in \mathcal{B}$, то $\theta_{v_i} \supseteq \beta$, $e_{v_i} = e_{v_{i-1}}$.

$\forall\, i = 0, \ldots, n \;\; e_{v_i}$ можно понимать как раскрытый терм в вершине v_i.

Число n в выполнении (3) называется **длиной** этого выполнения, и обозначается записью $|V|$. Вершина v_n в (3) называется **последней**.

Если какая-либо вершина v выполнения (3) является концом ребра с меткой вида $[a?e]$, или началом ребра с меткой вида $[a!e]$, то

- вершина v называется **получателем** (от агента a) или **отправителем** (агенту a), соответственно, и
- терм e обозначается записью e_v, и называется **сообщением**, **полученным** (или **отправленным**, соответственно) в вершине v.

5. Протоколы аутентификации

5.1. Определение протокола аутентификации

Протокол аутентификации (называемый ниже просто **протоколом**) – это конечная совокупность P процессов $p_1, \ldots, p_n$, удовлетворяющая следующим условиям:

- $\forall\, i = 1, \ldots, n \;\; A_{p_i} \subseteq \{a_{p_1}, \ldots, a_{p_n}\}$,
- множества $X_{p_i} \setminus A_{p_i}$ $(i = 1, \ldots, n)$ дизъюнктны.

Если протокол состоит из процессов p_1, ..., p_n, то такой протокол обозначается записью $(p_1, \ldots, p_n)$.

5.2. Нормальное выполнение протокола

Нормальное выполнение протокола $P = (p_1, \ldots, p_n)$ заключается в совместном выполнении процессов $p_1, \ldots, p_n$, входящих в этот протокол. При нормальном выполнении протокола P выполнение каждого невнутреннего действия α каким-либо процессом p_i, входящим в P, происходит синхронно с выполнением некоторого действия α' (называемого **комплементарным к** α) другим процессом p_j, входящим в P, причем

- либо α имеет вид $[a_{p_j}?e]$, α' имеет вид $[a_{p_i}!e']$,
- либо α имеет вид $[a_{p_j}!e]$, α' имеет вид $[a_{p_i}?e']$,

т.е. пара (α, α') представляет собой пересылку либо сообщения e' от процесса p_j процессу p_i, либо сообщения e от процесса p_i процессу p_j.

Нормальное выполнение протокола $P = (p_1, \ldots, p_n)$ представляет собой ациклический граф V, получаемый из дизъюнктного объединения $V_1 \sqcup \ldots \sqcup V_n$ (где $\forall\, i = 1, \ldots, n$ V_i является выполнением процесса p_i) добавлением новых ребер (называемых **горизонтальными** ребрами, эти ребра соответствуют комплементарным действиям), причем каждое горизонтальное ребро имеет вид $v_i \to v_j$, где

- v_i и v_j – вершины графов V_i и V_j соответственно, причем $i \neq j$, v_i – отправитель агенту a_{p_j}, v_j – получатель от агента a_{p_i}, и
$$\theta_{v_i} \subseteq \theta_{v_j}, \quad [\![e_{v_i} = e_{v_j}]\!] \subseteq \theta_{v_j},$$
- для каждой вершины-отправителя (или вершины-получателя) v графа V существует единственное горизонтальное ребро графа V с началом (или концом, соответственно) в v.

5.3. Выполнение протокола с участием противника

Противник – это процесс (обозначаемый символом I), который может

- перехватывать и уничтожать сообщения от любого процесса, и
- создавать новые сообщения (используя перехваченные сообщения), и посылать их (от имени любого агента) любому процессу.

Из сообщений, которые I либо имел до начала своего выполнения, либо перехватил во время своего выполнения, он может строить новые сообщения, путем применения операций из $\mathcal{F}$.

Выполнение протокола с участием противника I

- представляет собой совокупность выполнений процессов, входящих в этот протокол, и
- происходит так, что выполнение каждого невнутреннего действия какого-либо процесса p, входящего в этот протокол, может выполняться
  - либо так же, как при нормальном выполнении этого протокола, т.е. это действие выполняется синхронно с комплементарным действием другого процесса, входящего в этот протокол,
  - либо синхронно с выполнением соответствующего действия I:
    - при выполнении процессом p действия вида $[a!e]$ посланное сообщение e может получить не агент a, а I, и
    - при выполнении процессом p действия вида $[a?e]$ процесс p может получить сообщение не от агента a, а от I.

Без ограничения общности можно считать, что если протокол выполняется с участием I, то все невнутренние действия, выполняемые процессами этого протокола, заключаются в их взаимодействии с I, т.к. если какое-либо из этих действий заключается в передаче сообщения e от процесса p_i процессу p_j этого протокола, то можно считать что данное действие является комбинацией двух действий:

- перехват противником I того сообщения e, которое послал p_i, и
- посылка этого сообщения e от I процессу p_j.

5.4. Система переходов протокола

Система переходов (СП) протокола P – это граф Σ_P с определяемыми ниже множеством вершин S_P (называемых **состояниями**) и множеством ребер R_P (называемых **переходами**). В Σ_P представлены все возможные совместные выполнения процессов, входящих в P, с участием противника.

Для определения множеств S_P и R_P введем обозначения:

- $\forall\, \theta \in \Theta$, $\forall\, p \in \mathcal{P}$ запись θp обозначает процесс (называемый **вариантом**) процесса p, получаемый из p заменой каждой переменной $x \in X_p$ на терм θx,
- запись V_P обозначает множество пар вида (p, V), где p – вариант процесса, входящего в P, и V – какое-либо выполнение p.

Каждое состояние s из S_P – это тройка (W_s, θ_s, e_s), где

- W_s – конечное подмножество множества V_P, причем если W_s имеет вид $\{(p_i, V_i) \mid i = 1, \ldots, n\}$, то множества $X_{p_1} \setminus A_{p_1}$, ..., $X_{p_n} \setminus A_{p_n}$ дизъюнктны,
- $\theta_s \in \Theta$ и $e_s \in \mathcal{E}$.

Компоненты состояния s понимаются следующим образом:

- W_s представляет собой одно из возможных совместных выполнений (до текущего момента времени) некоторых вариантов процессов, входящих в P, с участием противника,
- θ_s и e_s – текущая подстановка и раскрытый терм в состоянии s.

Множество R_P переходов СП Σ_P состоит из всех пар (s, s') состояний, таких, что $\theta_s \subseteq \theta_{s'}$, и верно одно из двух следующих утверждений:

- $W_{s'}$ получается из W_s путем активизации некоторого варианта p одного из процессов, входящих в P, т.е. $W_{s'} = W_{s'} \cup \{(p, V)\}$, где $|V| = 0$, $\theta_{s'} \supseteq \theta$, $e_{s'} = (e_s, e)$, где θ и e – вторая и третья компонента соответственно метки единственной вершины графа V, или
- $W_{s'}$ получается из W_s путем продвижения на один шаг одного из выполнений (p, V), входящих в W_s, т.е. W_s можно представить в виде $W \cup \{(p, V)\}$, и
  - $W_{s'} = W \cup \{(p, V')\}$, где V' получается из V добавлением ребра $v \xrightarrow{\alpha} v'$, выходящего из последней вершины v выполнения V,
  - * если $\alpha = [a?e]$, то $\theta_{s'} e \in cl(e_s, \theta_s)$, $e_{s'} = e_s$,
    * если $\alpha = [a!e]$, то $e_{s'} = (e_s, e)$,
    * если $\alpha = \beta \in \mathcal{B}$, то $\theta_{s'} \supseteq \beta, e_{s'} = e_s$.

Сформулированные выше условия на переход (s, s') представляют собой возможные варианты продолжения совместного выполнения вариантов процессов, входящих в P:

- либо активизируется новый вариант какого-либо процесса из P,
- либо продолжается выполнение одного из процессов, входящих в s.

Состояние $s \in S_P$ называется **начальным**, если $W_s = \emptyset$.

Путь в СП $\Sigma_P = (S_P, R_P)$ – это последовательность $\pi = (s_0, \ldots, s_n)$ состояний из S_P, такая, что $\forall\, i = 1, \ldots, n \;\; (s_{i-1}, s_i) \in R_P$. Состояние s_0 в пути $\pi = (s_0, \ldots, s_n)$ обозначается записью $\hat{\pi}$.

Множество всех путей в Σ_P обозначается записью Π_P, а множество всех путей π в Σ_P, таких, что $\hat{\pi}$ – начальное состояние – записью Π_P^0.

$\forall\, \pi = (s_0, \ldots, s_n) \in \Pi_P$, $\forall\, s, s' \in S_P$

- запись $s \in \pi$ означает, что $\exists\, i \in \{0, \ldots, n\} : s = s_i$,
- запись $s <_\pi s'$ означает, что $\exists\, i, j \in \{0, \ldots, n\}$: $i < j$, $s = s_i$, $s' = s_j$.

6. Спецификация свойств протоколов

6.1. Истинность формул в состояниях

$\forall\, \pi \in \Pi_P$, $\forall\, s \in \pi$, $\forall\, \beta \in \mathcal{B}$ запись $s \models_\pi \beta$ обозначает утверждение

$$\beta \subseteq \theta_s, \quad \forall\, s' <_\pi s \quad \beta \not\subseteq \theta_{s'}. \tag{4}$$

(4) интерпретируется как утверждение о том, что s – первое состояние на пути π, в котором истинна формула β.

6.2. Свойство соответствия

Свойство соответствия – это запись вида $\beta \rightsquigarrow \beta_1, \beta_2$, где $\beta, \beta_1, \beta_2 \in \mathcal{B}$.

Будем говорить, что протокол P обладает свойством соответствия $\beta \quad \beta_1, \beta_2$, если $\forall\, \pi \in \Pi_P$, $\forall\, s \in \pi$ верна импликация

$$s \models_\pi \beta \quad \Rightarrow \quad \exists\, s' <_\pi s : \;\; s' \models_\pi \beta_1,\; s \models_\pi \beta_2. \tag{5}$$

Если в (5) $\exists\, s'$ заменить на $\exists !\, s'$ (существует единственное s'), то соответствующее свойство протокола P называется **инъективным свойством соответствия**. Такое свойство обозначается записью $\beta \rightsquigarrow^! \beta_1, \beta_2$.

6.3. Корректность протоколов аутентификации как свойство соответствия

В этом параграфе показывается, как задача обоснования корректности ПА сводится к задаче проверки некоторого инъективного свойства соответствия.

Простейший ПА имеет вид (p_a, p_b). В нем участвуют агенты a и b, где

- агент a называется **доказывающим агентом**, он передает агенту b свое имя, и его цель в этом ПА – доказать агенту b, что переданное им имя является его подлинным именем,

- агент b называется **проверяющим агентом**, его цель в этом ПА – установить, совпадает ли полученное им имя с именем того агента, который взаимодействует с ним при выполнении этого ПА.

Будем предполагать, что

- множество X_{p_b} содержит переменную n_a, предназначенную для занесения в нее того значения, которое проверяющий агент должен получить от доказывающего агента в качестве его имени, и

- множество U_{p_a} содержит переменную r_a, значение которой (в составе зашифрованного сообщения) передается процессу p_b и записывается им в переменную $x_a \in X_{p_b}$.

В выполнении этого ПА может принимать участие противник, поэтому не исключено, что в действительности с агентом b взаимодействует совсем не тот агент, имя которого заносится в переменную n_a (т.е. который выдает себя за n_a, но в действительности не является им).

Будем говорить, что описанный выше ПА является **корректным**, если при любом его выполнении с участием противника, всякий раз, когда процесс p_b успешно завершает свою работу и принимает положительное решение (т.е. решает, что значение, записанное в n_a, совпадает с истинным именем агента, с которым взаимодействовал p_b), данное решение является правильным.

Формальное описание корректности ПА (p_a, p_b) выражается в виде излагаемого ниже свойства соответствия.

Обозначим записями $\tilde{p}_a$ и $\tilde{p}_b$ процессы, получаемые из p_a и p_b соответственно следующими модификациями:

- $\tilde{p}_a$ получается из p_a добавлением перехода вида $s \xrightarrow{\alpha} s^0_{p_a}$, где

  - s – новое состояние, которое будет начальным в $\tilde{p}_a$, и
  - $\alpha = [\![(start, initiator_i, responder_i, value_i) = (1, a_p, b, r_a)]\!]$, где r_a – вышеупомянутая переменная из U_{p_a},

- $\tilde{p}_b$ получается из p_b добавлением для каждого терминального состояния $s \in S_{p_b}$ перехода вида $s \xrightarrow{\alpha} s'$, где

– s' – новое состояние, которое будет терминальным в $\tilde{p}_b$, и

– $\alpha = [\![(end, initiator_r, responder_r, value_r) = (1, n_a, b, x_a)]\!]$, где n_a и x_a – вышеупомянутые переменные из X_{p_b}.

Свойство корректности ПА (p_a, p_b) можно сформулировать в виде инъективного свойства соответствия $\beta \rightsquigarrow \beta_1, \beta_2$ где

- $\beta = [\![end = 1]\!]$
- $\beta_1 = [\![start = 1]\!]$
- $\beta_2 = [\![initiator_i = initiator_r]\!] \wedge [\![responder_i = responder_r]\!] \wedge [\![value_i = value_r]\!]$

Поясним смысл изложенного выше описания свойства корректности: оно выражает утверждение о том, что если агент-респондер (b) завершил сеанс протокола (т.е. значение переменной `end` стало равно 1), и при этом

- он считает, что агентом-инициатором данного сеанса был агент, имя которого равно значению переменной n_a, и
- уникальное секретное значение, которое передал агент-инициатор, содержится в переменной x_a,

то в некоторый предшествующий момент времени агент, имя которого равно значению переменной n_a действительно начал сеанс выполнения этого протокола именно с b, и то уникальное секретное значение, которое он передал b, действительно равно значению переменной x_a.

7. Применение изложенного подхода

Для применения данного подхода к анализу какого-либо конкретного протокола необходимо описать конечное множество $\mathbf{P}$ протоколов, в выполнении которых агенты могут принимать участие, и при построении СП Σ_P в качестве P рассматривать не только совокупность процессов, входящих в анализируемый протокол, а все процессы, входящие в какой-либо протокол из $\mathbf{P}$, это необходимо для того чтобы установить существование атак, связанных с одновременным участием противника в нескольких сеансах различных протоколов. При атаке Лоу на протокол Нидхэма-Шредера (NSPK) противник участвует одновременно в двух сеансах NSPK, однако возможен случай одновременного участия

противника в различных сеансах разных протоколов, и описанная выше модель позволяет установить соответствующую атаку, если она действительно существует.

8. Заключение

За пределами настоящего исследования остались следующие вопросы.

1) Предложенный метод представляет собой по сути переборный алгоритм, связанный с построением всевозможных вариантов исполнения анализируемого протокола с участием противника (предполагая, что противник может быть одним из законных участников какого-либо протокола). Данный метод обладает свойством полноты: если атака действительно существует, то он её найдет, причем для поиска атаки достаточно построить конечный фрагмент СП Σ_P. Можно ли установить верхнюю границу на размер указанного выше фрагмента? Можно ли оптимизировать процесс поиска возможной атаки, т.е. строить описанный выше фрагмент аналогично тому как строится фрагмент модели Крипке при верификации свойств, темпоральными формулами, на основе метода on-the-fly, с использованием автоматов Бюхи?

2) Если анализируемый протокол не допускает атаки, то можно ли это доказать, не производя явное построение СП Σ_P, а используя например метод инвариантов (называемый также методом Флойда, или методом Хоара), который позволяет доказывать корректность программ без построения соответствующего множества их состояний?

Решению данных проблем будут посвящены дальнейшие исследования.

Список литературы

[1] Kerberos: The Network Authentication Protocol.
MIT Kerberos. 10 September 2015. Retrieved 31 October 2015
`http://web.mit.edu/kerberos/`

[2] Cervesato I., Jaggard A.D., Scedrov A., Tsay J.-K., Walstad C.,
Breaking and fixing public-key Kerberos,
Information and Computation Volume 206, Issues 2-4, (2008), 402-424.

[3] Burrows M., Abadi M., Needham R.,
A Logic of Authentication,
ACM Transactions on Computer Systems, 8(1), (1990) 18-36.

[4] Javier Thayer, Jonathan Herzog, and Joshua Guttman,
Презентация “Strand Spaces”,
http://www2.imm.dtu.dk/courses/02913/F05/

[5] Jerry den Hartog,
Страница курса Verification of Security protocols
https://www.win.tue.nl/~jhartog/CourseVerif/

[6] Proceedings of Joint Workshop on Foundations of Computer Security and Automated Reasoning for Security Protocol Analysis (FCS-ARSPA '06)
Information and Computation Volume 206, Issue 2, (2008).
https://www.sciencedirect.com/journal/information-and-computation/vol/206/issue/2

[7] Veronique Cortier, Steve Kremer
Formal Models and Techniques for Analyzing Security Protocols
Now Publishers Inc., Hanover, United States (2014).

New mathematical model of authentication protocols and verification method based on this model

Mironov Andrew M.

Authentication protocols are distributed algorithms designed to provide authentication of agents and the transfer of confidential information (cryptographic keys, etc.) in an insecure environment. They are used, for example, in electronic payments, electronic voting procedures, database access systems, etc. On the reason of the large financial and social damage in the case of the incorrect execution of such protocols, it is necessary to use mathematical methods to justify their correctness and security. In the present work, a new mathematical model of such authentication protocols is introduced, which provides a possibility to describe both the protocols and their properties. It is shown a possibility to solve problems of verification of authentication protocols.

Keywords: authentication protocols, distributed algorithms, verification