Межфакультетский курс «Интеллектуальные методы анализа протоколов безопасности» Лекция 1 Понятие протокола безопасности

А.М.Миронов

1 Понятие протокола безопасности

Протокол безопасности (**ПБ**), называемый также просто **протоколом** – это распределённый алгоритм, определяющий порядок обмена сообщениями между несколькими **агентами**, в качестве которых могут выступать, например, люди, компьютерные программы, вычислительные комплексы, базы данных, сети связи, банковские карточки, и т.д. Агенты, принимающие участие в работе ПБ, называются **участниками** этого ПБ.

Действия, выполняемые каждым из участников ПБ, могут иметь следующий вид:

- **посылка сообщения** другому участнику этого ПБ (или группе участников),
- **приём сообщения** от другого участника,
- **внутренние действия**, к числу которых относятся
  - выполнение участником некоторых вычислений,
  - проверка логических условий, и
  - обновление значений переменных.

ПБ предназначены для обеспечения безопасности передачи, обработки и хранения информации в небезопасной среде. Свойства безопасности, которые должен обеспечивать ПБ, могут иметь, например, следующий вид:

- **целостность** передаваемых сообщений, которая заключается в том, что всякое изменение сообщений в процессе их передачи будет обнаружено в ходе выполнения ПБ,

- **секретность** передаваемых сообщений, которая заключается в отсутствии неавторизованной утечки информации в процессе работы ПБ.

2 Шифрование сообщений

2.1 Понятие шифрования и дешифрования

Некоторые из сообщений, пересылаемых участниками ПБ, могут быть зашифрованными. Шифрование сообщений делается для того, чтобы противник, которому станут доступны пересылаемые сообщения, не смог ознакомиться с их содержанием.

Шифрование сообщения m представляет собой применение некоторого алгоритма $Encr$ (называемого **алгоритмом шифрования**) к паре (k, m), где

- k – битовая строка, называемая **ключом шифрования** (или просто **ключом**), и

- m – шифруемое сообщение, называемое в данном случае **открытым текстом (ОТ)**.

Мы будем обозначать результат применения алгоритма $Encr$ к паре (k, m) записью $k(m)$ и называть её **шифртекстом (ШТ)** сообщения m на ключе k.

Для того, чтобы извлечь из ШТ $k(m)$ исходное сообщение m, должен быть задан алгоритм $Decr$, называемый **алгоритмом дешифрования**, и обладающий следующими свойствами:

- алгоритм $Decr$ получает на вход пару вида (d, u), где

  - d – битовая строка, называемая **ключом дешифрования**, и
  - u – сообщение,

  результат применения алгоритма $Decr$ к паре (d, u) будем обозначать записью $d(u)$,

- каждому ключу шифрования k должен соответствовать ключ дешифрования d_k, такой, что для каждого сообщения m верно равенство

$$d_k(k(m)) = m. \tag{1}$$

2.2 Системы шифрования

Система шифрования (СШ) представляет собой набор следующих данных:

- пара алгоритмов *Encr* и *Decr* шифрования и дешифрования соответственно, и
- набор ограничений, которым должны удовлетворять ключи шифрования и дешифрования, а также шифруемые сообщения.

Системы шифрования принято подразделять на два класса: симметричные и асимметричные.

1. В **симметричных** СШ **(ССШ)** каждый ключ шифрования k совпадает с соответствующим ему ключом дешифрования d_k. Как правило, алгоритмы шифрования и дешифрования в ССШ тоже совпадают.
2. В **асимметричных** СШ **(АСШ)** ключи шифрования могут отличаться от соответствующих им ключей дешифрования, и
   - ключи шифрования и дешифрования в АСШ, как правило, связаны с конкретными агентами, мы будем обозначать эти ключи записями A^+ и A^- соответственно, где A – идентификатор агента, с которым связаны эти ключи, и
   - ключ A^+ является **открытым** (т.е. известен всем агентам), в то время как ключ A^- **закрыт**: он не должен быть известен никому кроме агента A (и, возможно, некоторым доверенным агентам).

Некоторые АСШ обладают следующим свойством: для каждого сообщения m верно равенство

$$A^+A^-(m) = m. \tag{2}$$

Такие АСШ можно использовать для **аутентификации** агентов: если какой-либо агент A, использующий эту АСШ, хочет доказать свою подлинность (т.е. доказать, что он действительно является тем, за кого себя

выдаёт), то он может сделать это путем предъявления пары (m, m'), где m – произвольное сообщение, и $m' = A^-(m)$. Доказательство подлинности A будет принято, если будет выполнено условие

$$A^+(m') = m. \tag{3}$$

Данное условие обосновывается предположением о том, что

- без знания закрытого ключа A^- создать сообщение m', удовлетворяющее условию (3), невозможно, и
- никто, кроме агента A, не должен знать ключ A^-.

3 Электронная подпись

Электронная подпись является одним из средств, предназначенных для доказательства подлинности передаваемых сообщений и их отправителей.

Протокол электронной подписи – это алгоритм, преобразующий пару (m, A), где m – сообщение и A – имя агента, в строку,

- обозначаемую записью $\langle m \rangle_A^s$, и
- называемую **электронной подписью (ЭП)** сообщения m, созданной агентом A.

При вычислении $\langle m \rangle_A^s$ используется **закрытый ключ** A^-, который должен быть известен только агенту A.

Тройку $(m, A, \langle m \rangle_A^s)$ мы будем обозначать записью $\langle m \rangle_A$.

ЭП должна обладать следующими свойствами:

- **возможность проверки подлинности ЭП**: существует открытый алгоритм позволяющий по тройке (m, A, u) проверить истинность равенства $u = \langle m \rangle_A^s$,
- **невозможность подделки ЭП**: задача вычисления $\langle m \rangle_A^s$ без знания A^- является труднорешаемой,
- **невозможность подмены**: задача нахождения по $\langle m \rangle_A^s$ такого $m' \neq m$, что $\langle m' \rangle_A^s = \langle m \rangle_A^s$, является труднорешаемой.

В том случае, когда подписывающий агент A и проверяющий агент B могут использовать одну и ту же АШС, $\langle m \rangle_A^s$ может иметь один из следующих видов: $A^-(m)$, $B^+A^-(m)$, $A^-(A, A^-(m), t)$, где t – метка времени, и т.п.

4 Хэш-функции

Хэш-функция (ХФ) - это функция h вида

$$h : D \to \{0,1\}^n, \quad \text{где } D \subseteq \{0,1\}^*$$

(где n – заданное число, $\{0,1\}^n$ – множество битовых последовательностей длины n, $\{0,1\}^*$ – множество конечных битовых последовательностей), алгоритм вычисления которой должен быть общеизвестен, и которая обладает следующими свойствами:

- h – **односторонняя функция**, т.е. не существует быстрого алгоритма нахождения по заданному $y \in \{0,1\}^n$ такого $x \in D$, что $y = h(x)$,
- h **устойчива к коллизиям**, т.е. сложно найти различные $x_1, x_2 \in D$, такие, что $h(x_1) = h(x_2)$.

ХФ применяются для контроля целостности данных, аутентификации источников данных, и многих других целей.

5 Формальные описания и примеры протоколов

5.1 Понятие формального описания протокола

Одним из простейших видов формального описания ПБ является список P записей вида

$$A \to B : [\![\varphi]\!]\, m, \tag{4}$$

$$A \to \{B_1, \ldots, B_n\} : [\![\varphi]\!]\, m, \tag{5}$$

или

$$A : [\![\varphi]\!] \text{ внутреннее действие}, \tag{6}$$

где A, B, $B_1, \ldots, B_n$ – имена агентов, φ – условие, m – выражение, значением которого является сообщение (**сообщением** мы называем любой объект, который один из участников ПБ передаёт другому в процессе функционирования ПБ, этот объект может быть как символьной строкой, так и набором банкнот, товаром, и т.п.). Записи (4), (5) и (6) изображают действия, которые должны выполнять агенты в процессе функционирования ПБ.

Действия, входящие в список P, выполняются последовательно, их выполнение происходит следующим образом:

- (4) и (5) выполняется путем проверки φ, и
  - если φ выполнено, то A посылает сообщение m агенту B (в случае (4)), или агентам $B_1, \ldots, B_n$ (в случае (5)),
  - если же φ не выполнено, то данное действие пропускается, и выполняется следующее по списку действие,
- (6) выполняется аналогично, только в данном случае происходит не посылка сообщения от A, а вычисления и изменение значений переменных агента A.

Если φ верно всегда, то компонента $[\![\varphi]\!]$ в записи действий опускается.

Если текущее исполняемое действие не относится к какому-либо из участников ПБ, то этот участник не функционирует в момент исполнения этого действия.

5.2 Пример формального описания протокола

В этом пункте мы рассмотрим пример формального описания протокола, решающего задачу продажи компьютера агента A агенту B. Мы предполагаем, что у A есть компьютер, а у B есть деньги, и B хочет на эти деньги купить у A компьютер. Агенты A и B не доверяют друг другу, поэтому протоколы продажи компьютера, имеющие вид

$$\begin{array}{l} A \to B: \text{ компьютер} \\ B \to A: \text{ деньги} \end{array}$$

и

$$\begin{array}{l} B \to A: \text{ деньги} \\ A \to B: \text{ компьютер} \end{array}$$

для них неприемлемы: каждый из них не верит, что если он выполнит первое действие в первом или втором протоколе, то его коллега обязательно выполнит второе действие.

Одним из возможных решений задачи продажи компьютера агента A агенту B может быть протокол, в котором, помимо A и B, принимает участие доверенный посредник J. Данный протокол может иметь, например, следующий вид:

- $A \to J$: компьютер
- $B \to A$: деньги

- $A \to J$: $\left(\begin{array}{l}\text{подтверждение или опровержение}\\ \text{того, что полученная от } B \text{ сумма}\\ \text{соответствует стоимости компьютера}\end{array}\right)$
- $J \to B$: $[\![\text{от } A \text{ поступило подтверждение}]\!]$ компьютер
- $J \to A$: $[\![\text{от } A \text{ поступило опровержение}]\!]$ компьютер

A и B могут считать данный протокол приемлемым, например, по следующим причинам:

- A верит, что до окончания проверки денег агент J не передаст компьютер агенту B, и J вернёт компьютер A, если B передаст A недостаточную сумму,
- B верит, что пока A не пошлёт J подтверждение, компьютер будет находиться у J, и сразу после того, как A пошлёт J подтверждение, J передаст B компьютер.

Однако данный протокол некорректно работает в том случае, когда какой-либо из агентов ведёт себя нечестно (например, B посылает A правильную сумму, но A посылает опровержение, получает обратно свой компьютер, и не отдает B полученные от него деньги).

Межфакультетский курс «Интеллектуальные методы анализа протоколов безопасности» Лекция 2 Уязвимости протоколов безопасности

А.М.Миронов

1 Уязвимости протоколов

1.1 Понятие уязвимости протокола

Нарушения свойств безопасности в процессе работы ПБ могут происходить по причине противодействия со стороны агентов, называемых **противниками**. Противники могут быть участниками протоколов.

Противники подразделяются на следующие два класса:

- **пассивные противники**, они могут перехватывать сообщения, пересылаемые участниками ПБ, и анализировать их,
- **активные противники**, они могут делать то же, что и пассивные противники, а также
  - модифицировать или удалять перехваченные сообщения,
  - генерировать новые сообщения и посылать их участникам ПБ,
  - выдавать себя за участников ПБ.

Кроме того, нарушения свойств безопасности ПБ возможны из-за действий участников ПБ, которые могут нарушать (умышленно или неумышленно) предписанные протоколом правила взаимодействия с другими участниками этого ПБ.

Возможность нарушения свойств безопасности в процессе работы ПБ называют **уязвимостями** этого ПБ.

При решении задач анализа уязвимостей ПБ используются следующие предположения о противнике:

- противник полностью знает ПБ,
- противнику доступны все сообщения, пересылаемые между участниками ПБ, он может их модифицировать, удалять, и заменять своими сообщениями,
- противник не может извлечь из перехваченных ШТ соответствующие ОТ, если он не знает необходимых ключей.

1.2 Пример уязвимости протокола

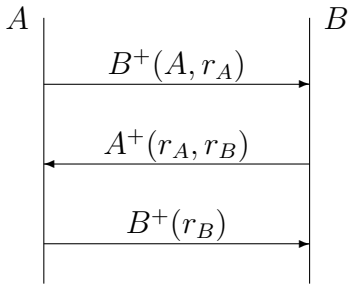
В этом пункте мы рассмотрим пример уязвимости ПБ, который использовался много лет в банковских транзакциях. Данный протокол очень прост, он состоит всего из трёх действий, и сначала его правильность не вызывала сомнений, однако после 15 лет его использования выяснилось, что он содержит уязвимость (которая была обнаружена автоматической системой верификации). Этот ПБ называется **протоколом Нидхема--Шредера** (Needham-Schroeder, 1979), мы будем обозначать его записью NS. Целью данного ПБ является взаимная аутентификация агентов A и B.

Предполагается, что A и B используют общую АСШ.

Последовательность действий, из которых состоит протокол NS, имеет следующий вид:

$$\begin{array}{l} A \to B : B^+(A, r_A) \\ B \to A : A^+(r_A, r_B) \\ A \to B : B^+(r_B) \end{array} \qquad (1)$$

Для формальной записи ПБ используется также нотация в виде диаграмм. Каждое действие в ПБ, связанное с пересылкой сообщений, изображается в диаграмме горизонтальной стрелкой, помеченной пересылаемым сообщением. Диаграмма, соответствующая ПБ (1), имеет вид



Действия, входящие в ПБ NS, имеют следующий смысл.

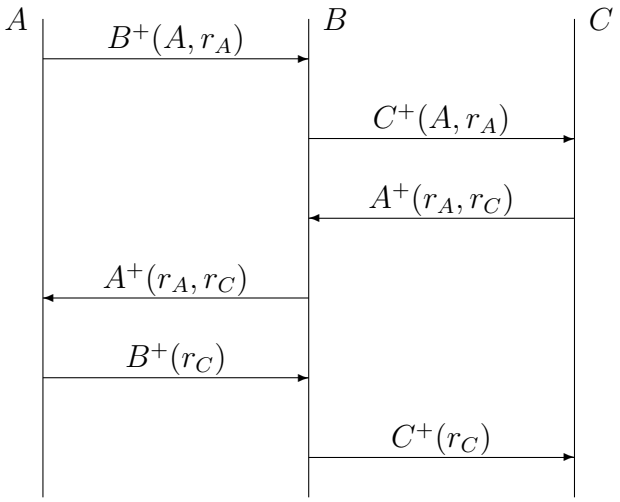
- Первое действие заключается в пересылке от A к B ШТ $B^+(A, r_A)$ где соответствующий ОТ – пара, состоящая из имени агента A, и уникального значения r_A, которое представляет собой большую (несколько сотен битов) строку, случайным образом сгенерированную агентом A (строки такого типа называются **нонсами** (nonce, это слово является аббревиатурой словосочетания number used once)). Нонс r_A используется для того, чтобы дать возможность B доказать свою подлинность путем извлечения данного нонса из $B^+(A, r_A)$.

- Второе действие заключается в пересылке от B к A ШТ $A^+(r_A, r_B)$ где соответствующий ОТ – пара, состоящая из нонса r_A, который агент B извлёк из полученного ШТ $B^+(A, r_A)$, и нонса r_B, сгенерированного агентом B. После его получения A убеждается, что его отправителем может быть только B, т.к. никто кроме B не может получить r_A. Нонс r_B предназначен для того, чтобы дать возможность A тоже доказать свою подлинность.

- Третье действие заключается в пересылке от A к B ШТ $B^+(r_B)$. После получения r_B агент B убеждается в подлинности агента A.

Таким образом, после успешного завершения данного протокола A и B будут убеждены в подлинности друг друга.

Одна из уязвимостей данного ПБ связана с тем, что агент B может использовать свой статус участника ПБ NS для того, чтобы во взаимодействии с другими агентами выдавать себя за агента A. Использование агентом B своего статуса участника ПБ NS для совершения мошеннических действий можно изобразить следующей диаграммой:



Данную диаграмму можно рассматривать как объединение двух диаграмм, изображающих выполнение двух сеансов выполнения ПБ NS:

- в первом сеансе участвуют агенты A и B,
- а во втором – агенты B и C.

Действия агентов A, B и C во время выполнения этих сеансов выглядят следующим образом.

- После того, как A и B выполнят первое действие первого сеанса, агент B создает с использованием полученного нонса r_A ШТ $C^+(A, r_A)$, и посылает этот ШТ агенту C в качестве своего первого действия во втором сеансе ПБ NS (выдавая себя за A).
- Получив этот ШТ и дешифруя его, агент C предполагает, что этот ШТ был послан агентом A (о чём говорит ему первая компонента соответствующего ОТ). Поэтому, согласно протоколу NS, в качестве второго действия агент C должен сгенерировать нонс r_C и послать ШТ $A^+(r_A, r_C)$ тому агенту, от которого он получил первый ШТ (думая, что он посылает этот ШТ агенту A).
- B пересылает полученный от C ШТ $A^+(r_A, r_C)$ агенту A.
- A рассматривает полученный от B ШТ как тот ШТ, который B должен послать ему в соответствии со вторым шагом ПБ NS, т.е. A рассматривает извлечённый из полученного ШТ нонс r_C как нонс, который был сгенерирован агентом B.
- Согласно третьему шагу NS, A посылает B ШТ $B^+(r_C)$.
- B извлекает из полученного ШТ нонс r_C, и выполняет

  $$B \to C : C^+(r_C)$$

  как третье действие во втором сеансе ПБ NS.

После этого C верит, что тот агент, с которым он выполнял этот сеанс ПБ NS, является агентом A.

2 Другие примеры уязвимостей протоколов

2.1 Обедающие криптографы

Рассмотрим следующую задачу.

За круглым столом сидят три криптографа и обедают. После того, как они пообедали и хотят заплатить, официант сообщает им, что их обед уже оплачен, но не уточняет, кто именно платил.

Возможен один из двух вариантов:

- обед оплатил один из криптографов,
- обед оплатила ФСБ.

Криптографы хотят выяснить, какой именно из вариантов имеет место, причём, если имеет место первый вариант, то те криптографы, которые не платили, не должны узнать, кто же конкретно оплатил их обед.

Для решения этой задачи предлагается следующий ПБ.

Поскольку участники сидят за круглым столом, то каждая пара соседей может подбрасывать монету между собой, так, чтобы результат был известен только им двоим. Подбрасывание монеты двумя участниками можно рассматривать как получение ими сообщения, состоящего из одного случайным образом порождённого бита, от доверенного посредника.

После того, как все три пары подбросили монету, каждый участник знает результаты двух подбрасываний (решка или орёл). Эти результаты могут быть

- либо одинаковыми (т.е. оба раза была решка, или оба раза был орёл),
- либо разными (т.е. при одном подбрасывании была решка, а при другом - орёл).

Каждый участник говорит другим “одинаково” или “по разному”, причём тот, кто заплатил, утверждает противоположное (т.е. если надо сказать “одинаково” то он говорит “по-разному”, и наоборот).

Если число ответов “по разному” чётно, то это значит, что обед оплатила ФСБ, иначе - один из них.

Уязвимость этого протокола – в отсутствии контроля честности участников.

2.2 Протокол с подтверждением приёма

Рассмотрим следующую ситуацию: агенты A и B используют общую АСШ для секретного обмена сообщениями, и для контроля правильности передачи каждое получаемое сообщение отсылается назад отправителю (чтобы отправитель был уверен, что сообщение получено в неискажённом виде), согласно следующему протоколу:

$$\begin{array}{l} A \to B : B^+A^-(m) \\ B \to A : A^+B^-(m) \end{array} \tag{2}$$

(из полученного ШТ извлекается сообщение m и возвращается отправителю в зашифрованном виде в качестве подтверждения приёма).

К сожалению, данный ПБ уязвим к следующей атаке: активный противник E может перехватить первое сообщение и послать его B (от своего имени), в результате чего будут выполнены следующие действия:

$$\begin{array}{l} E \to B : B^+A^-(m) \\ B \to E : E^+B^-E^+A^-(m) \end{array}$$

т.к. при взаимодействии с E агент B действует согласно протоколу (2): получив сообщение m' $(= B^+A^-(m))$, он посылает в ответ сообщение

$$E^+B^-E^+B^-(m')\ (= E^+B^-E^+A^-(m)). \tag{3}$$

Получив (3), E может извлечь из него m.

2.3 Вычисление суммы

Агенты $A_1, \ldots, A_n$ имеют числа $x_1, \ldots, x_n$ соответственно. Они хотят вычислить $\sum_{i=1}^{n} x_i$, причём каждый A_i не хочет разглашать своё число x_i.

Один из протоколов для решения данной задачи имеет вид:

- $A_1 \to A_2 : {A_2}^+(x_1 + r)$, где r – случайно выбранное число из $\mathbf{Z}$
- $A_2 \to A_3 : {A_3}^+(x_2 + x_1 + r)$
- $\ldots$
- $A_n \to A_1 : {A_1}^+(x_n + \ldots + x_1 + r)$
- A_1 вычитает r из полученного результата.

Уязвимости данного протокола: 1). отсутствие контроля честности участников, 2). если все участники кроме одного откроют свои значения x_i друг другу, то они смогут узнать и значение оставшегося участника.

Межфакультетский курс «Интеллектуальные методы анализа протоколов безопасности» Лекция 3 Протоколы электронного голосования

А.М.Миронов

1 Понятие протокола голосования

Задача **голосования** заключается в том, что несколько агентов должны совместно выбрать решение из некоторого множества возможных решений. Каждый агент заполняет свой бюллетень, отражающий решение этого агента. Совместное решение вырабатывается путём обработки всех бюллетеней.

Например, в качестве данного решения может выступать избрание некоторого лица на какую-либо должность, в этом случае

- каждый бюллетень представляет собой список возможных кандидатов,
- заполнение бюллетеня агентом заключается в том, что агент оставляет в своём бюллетене только того кандидата, которого он хотел бы видеть избранным на эту должность,
- обработка бюллетеней заключается в подсчёте голосов, поданных за каждого кандидата,
- избранным считается тот кандидат, за которого подано наибольшее количество голосов.

Часто при процедуре голосования важно обеспечить конфиденциальность решений, принимаемых агентами (и некоторые другие условия,

связанные с контролем правильности подсчёта голосов). Данная задача решается при помощи специальных протоколов, которые называются **протоколами голосования**.

Наиболее часто используются такие протоколы голосования, в которых

- каждый агент отсылает свой бюллетень некоторому доверенному агенту, называемому **Центральной Избирательной Комиссией (ЦИК)**, которого мы будем обозначать ниже символом J,
- ЦИК обрабатывает полученные от агентов бюллетени, и публикует результат голосования.

Условия на процедуру голосования, которые должен обеспечить протокол голосования, могут иметь, например, следующий вид.

1. Голосовать могут только те агенты, которые имеют на это право (такие агенты называются **избирателями**).
2. Каждый избиратель может голосовать только один раз (т.е. может послать только один бюллетень).
3. Невозможно установить, за кого проголосовал каждый избиратель.
4. Невозможно использовать дубликат заполненного бюллетеня.
5. Невозможно изменить результат голосования каждого избирателя.
6. Каждый избиратель может проверить, что его бюллетень учтён.
7. Всем известно, кто участвовал в голосовании.

2 Примеры протоколов голосования

1. Простой протокол голосования:

   - каждый избиратель A_i создаёт свой бюллетень v_i, шифрует его и посылает J:
     $$A_i \to J : J^+(v_i)$$
   - J вычисляет результат и публикует его.

   В данном протоколе не выполняются почти все вышеперечисленные условия.

2. Другой протокол:

   - каждый избиратель A_i создаёт свой бюллетень v_i, подписывает его, шифрует, и посылает J:

     $$A_i \to J : J^+\langle v_i \rangle_{A_i}$$

   - J вычисляет результат и публикует его.

   В данном протоколе условия 1 и 2 выполняются, условие 3 не выполняется, условие 4 выполняется, и т.д.

3 Протоколы голосования с использованием слепой электронной подписи

Предположим, что целью голосования является выбор одной из двух альтернатив. Каждый избиратель A_i должен иметь два бюллетеня (каждый из которых соответствует некоторой альтернативе), которые подписаны J. Избиратель A_i должен выбрать один из этих бюллетеней, и послать его J. Победившей считается та альтернатива, за которую подано больше голосов.

В излагаемом ниже протоколе взаимодействие избирателя A_i с J осуществляется следующим образом.

1. A_i создает 10 троек вида $(v_j^{(1)}, v_j^{(2)}, r_j)$ $(j = 1, \ldots, 10)$, где

   - $v_j^{(1)}$ – j-й бюллетень агента A_i, соответствующий первой альтернативе,
   - $v_j^{(2)}$ – j-й бюллетень агента A_i, соответствующий второй альтернативе,
   - r_j – нонс (т.е. уникальное псевдослучайное значение, которое генерируется заново при каждом сеансе выполнения протокола), который в данном случае рассматривается как уникальный номер A_i.

2. $A_i \to J : \{\left(\xi_j v_j^{(1)},\ \xi_j v_j^{(2)},\ \xi_j r_j\right) \mid j = 1, \ldots, 10\}$, где ξ_j – маскирующие множители, причём операция умножения на маскирующий множитель коммутирует с функцией вычисления ЭП $\langle m \rangle_J$.

3. J проверяет в своей базе данных, что раньше от A_i не было сообщений, и заносит имя A_i в свою базу данных.

4. J выбирает 9 сообщений из полученных 10, просит у A_i их маскирующие множители, и открывает их.

5. Если все сообщения корректны, J подписывает в невскрытом кортеже $\Big(\xi v^{(1)},\ \xi v^{(2)},\ \xi r\Big)$ все 3 компоненты, и посылает результат A_i:

$$J \to A_i : \Big(\langle \xi v^{(1)} \rangle_J,\ \langle \xi v^{(2)} \rangle_J,\ \langle \xi r \rangle_J\Big)$$

6. A_i удаляет ξ, выбирает нужный $\langle v^{(\cdot)} \rangle_J$, и

$$A_i \overset{\circ}{\to} J : J^+\Big(\langle v^{(\cdot)} \rangle_J,\ \langle r \rangle_J\Big)$$

   ($\overset{\circ}{\to}$ – анонимная посылка)

7. J проверяет уникальность нонса (если нонс новый, то заносит его в свою БД), после чего вычисляет результат, и публикует

   - результат голосования, и
   - нонсы голосовавших, вместе с волеизъявлением каждого нонса.

Недостаток: нечестный J может фальсифицировать выборы, генерируя бюллетени и посылая их самому себе.

4 Голосование с ЦИК + ЦУР

Для того чтобы противодействовать возможной нечестности со стороны ЦИК, можно использовать второго доверенного посредника, называемого **Центральным Управлением Регистрации (ЦУР)**, и обозначаемого записью J'. Необходимым условием корректности нижеследующего протокола является отсутствие обмена информацией между ЦУР и ЦИК.

Взаимодействие между A_i, J и J' осуществляется следующим образом. В каждой из нижеследующих пересылок пересылаемые сообщения должны быть подписаны и зашифрованы открытым ключом получателя.

1. $A_i \to J'$: просьба дать регистрационный номер

2. $J' \to A_i : r_i$ (случайный рег. номер)

3. $J' \to J : \mathcal{R} :=$ список всех выданных рег. номеров

4. $A_i \to J : (ID_i, r_i, v_i)$, где ID_i – случайный идентификатор избирателя A_i, v_i – бюллетень избирателя A_i

5. J проверяет: $r_i \in \mathcal{R}$? Если это верно, то

   - $\mathcal{R} := \mathcal{R} \setminus \{r_i\}$
   - $\Im := \Im \sqcup \{ID_i\}$ (в начале работы $\Im = \emptyset$)

6. после получения всех бюллетеней J публикует результат, и список записей вида $(ID_i,\ v_i)$.

7. J' публикует список зарегистрированных A_i.

5 Улучшенный протокол голосования

Данный протокол удовлетворяет условиям 1 - 6, а также обладает следующим свойством: если избиратель A_i обнаружит, что его бюллетень был заполнен или обработан неправильно, то он может переголосовать.

Подготовка:

1. J публикует список всех агентов, имеющих право голосовать

2. $A_i \to J :$ намерение голосовать

3. J публикует список избирателей, собирающихся принять участие в выборах
   (это мешает J включать поддельные бюллетени)

Голосование (ниже $\overset{\circ}{\to}$ – анонимная передача):

1. $J \overset{\circ}{\to} A_i : ID_i$ (идентификационный номер)

2. $A_i \overset{\circ}{\to} J : \left(ID_i,\ A_i^+(ID_i, v_i)\right)$

3. J публикует $A_i^+(ID_i, v_i)$ (это позволяет A_i проверить, что J корректно учёл его v_i)

4. $A_i \to J : ID_i, A_i^-$

5. J расшифровывает бюллетени и обрабатывает их

6. J публикует результаты голосования, и все

$$v_i,\ A_i^+(ID_i, v_i)$$

7. Если A_i обнаружил, что его v_i учтён неверно, то

$$A_i \to J:\ \left(ID_i,\ A_i^+(ID_i, v_i),\ A_i^-\right)$$

8. Если A_i хочет изменить выбор с v_i на v_i', то

$$A_i \to J:\ \left(ID_i,\ A_i^+(ID_i, v_i'),\ A_i^-\right)$$

Если на этапе 1 голосования J обнаруживает, что два избирателя получили одинаковые ID, то J

- генерирует новый ID',
- выбирает одного из избирателей (=: A_i) с этим ID, и
- публикует $ID', A_i^+(ID, v_i)$

A_i узнаёт о путанице, и повторно отсылает свой v_i (этап 2 голосования) с новым идентификационным номером ID'.

Недостатки: преступная ЦИК может

- воспользоваться бюллетенями избирателей, которые зарегистрировались, но не голосовали
- безнаказанно "потерять" некоторые бюллетени

6 Выборы без ЦИК

Для простоты мы рассмотрим случай, когда в голосовании участвуют 4 избирателя: A_1, A_2, A_3, A_4. Все избиратели используют одну и ту же асимметричную ШС.

1. $\forall\, i\ \ A_i \to A_1 : m_i$, где

$$\begin{array}{rcl} m_i & := & A_1^+(u_{i1}, r_{i1}) \\ u_{i1} & := & A_2^+(u_{i2}, r_{i2}) \\ u_{i2} & := & A_3^+(u_{i3}, r_{i3}) \\ u_{i3} & := & A_4^+(u_{i4}, r_{i4}) \\ u_{i4} & := & A_1^+(v_{i1}) \\ v_{i1} & := & A_2^+(v_{i2}) \\ v_{i2} & := & A_3^+(v_{i3}) \\ v_{i3} & := & A_4^+(v_i, r_{i5}) \end{array}$$

(где v_i – бюллетень избирателя A_i)

2. $A_1 \to A_2$: перетасованный набор u_{i1}
3. $A_2 \to A_3$: перетасованный набор u_{i2},
4. $A_3 \to A_4$: перетасованный набор u_{i3}
5. $A_4 \to A_1$: перетасованный набор u_{i4}
6. $A_1 \to \{A_2, A_3, A_4\}$: перетасованный набор $\langle v_{i1} \rangle_{A_1}$
7. $A_2 \to \{A_1, A_3, A_4\}$: перетасованный набор $\langle v_{i2} \rangle_{A_2}$
8. $A_3 \to \{A_1, A_2, A_4\}$: перетасованный набор $\langle v_{i3} \rangle_{A_3}$
9. $A_4 \to \{A_1, A_2, A_3\}$: перетасованный набор $\langle A_i, r_{i5} \rangle_{A_4}$

Каждый раз, когда избиратель получает набор сообщений, одной из компонентов которых является нонс, он проверяет наличие среди полученных сообщений такого, в котором присутствует его нонс.

Межфакультетский курс «Интеллектуальные методы анализа протоколов безопасности» Лекция 4 Протоколы электронных платежей

А.М.Миронов

1 Протоколы электронных платежей

В протоколах **электронных платежей**, называемых также протоколами **электронной коммерции (ПЭК)** некоторые взаимодействия между участниками имеют коммерческий характер. Сообщения, передаваемые в таких взаимодействиях, являются электронными аналогами бумажных денег и называются **электронными банкнотами (ЭБ)**.

Свойства, которыми могут обладать ЭБ, могут иметь, например, следующий вид.

1. По ЭБ невозможно вычислить тех участников, которые использовали её в своих платежах (анонимность).
2. ЭБ можно передавать другим участникам, которые могут использовать её по своему усмотрению.
3. ЭБ невозможно использовать более одного раза путём изготовления дубликата.

Ниже в этой главе некоторые участники ПЭК обозначаются специальными символами:

- символом B обозначается банк, и
- символом T обозначается продавец.

2 Примеры ПЭК

- $A \to B : (\xi_1 m_1, \ldots, \xi_k m_k)$, где
  - $m_1, \ldots, m_k$ – переводы на одну и ту же сумму,
  - $\xi_1, \ldots, \xi_k$ – маскирующие множители (маскировка коммутирует с ЭП $\langle m \rangle_B$)
- B выбирает $i \in \{1, \ldots, k\}$, снимает маскировку у всех переводов кроме i-го, проверяет вскрытые переводы, после чего
  - $B \to A : \langle \xi_i m_i \rangle_B$
  - B списывает со счёта A соотв. сумму.

  (в результате B не сможет определить, что $m_i \in A$)
- $A \to T : \langle m_i \rangle_B$ (платёж)
- $T \to B : \langle m_i \rangle_B$ (депонирование)
- $B \to T$: деньги.

Недостаток: можно сделать копию ЭБ и использовать её. Чтобы этого не было, надо заменить первое действие на

$$A \to B : \xi_1(m_1, r_1), \ldots, \xi_k(m_k, r_k)$$

Во 2-м действии B проверяет уникальность всех нонсов у демаскированных переводов. Когда в 4-м действии B получает $\langle m_i, r_i \rangle_B$, он проверяет, депонировалась ли ЭБ с r_i.

Однако этот КП не может определить мошенника.

Можно заменить действие 4 на

$$T \to B : \langle m_i, r_i \rangle_B, r'$$

где нонс r' генерируется совместно A и T. Когда B получает это сообщение, он проверяет наличие r_i и r' в своей БД:

- если r' есть в БД банка, то присланное сообщение является не законной ЭБ, а копией, которую сделал T,
- если r_i есть, а r' нет, то присланное сообщение является не законной ЭБ, а копией, которую сделал A.

3 ПЭК, распознающий жулика

1. $A \to B : \{\xi_i(m_i, r_i, h(\alpha_{i1}), h(\alpha_{i2})) \mid i = 1, \ldots, k\}$, где
   - $m_1, \ldots, m_k$ – переводы на одну и ту же сумму,
   - $\xi_1, \ldots, \xi_k$ – маскировка, коммутирует с ЭП,
   - α_{i1} и α_{i2} – доли секрета: $id_A = \alpha_{i1} \oplus \alpha_{i2}$,
   - h – ХФ.
2. B выбирает один из переводов, вскрывает остальные и проверяет их (в том числе, просит A сообщить α_{i1} и α_{i2} у вскрытых переводов), после чего подписывает оставшийся перевод и посылает его A:
   - $B \to A : \langle \xi(m, r, h(\alpha_1), h(\alpha_2)) \rangle_B$
   - B списывает со счёта A соотв. сумму.
3. $A \to T : \langle m, r, h(\alpha_1), h(\alpha_2) \rangle_B$ (ЭБ для платежа)
4. $T \to A : e := (e_1, \ldots, e_k) \underset{r}{\in} \{1, 2\}^k$
5. $A \to T : a_e := (\alpha_{1e_1}, \ldots, \alpha_{ke_k})$
   (T принимает платёж, если среди $\{h(\alpha_{ie_i}) \mid i = 1, \ldots, k\}$ есть третья или четвёртая компонента полученной ЭБ)
6. $T \to B : (\langle m, r, h(\alpha_1), h(\alpha_2) \rangle_B, a_e)$ (депонирование)
7. B проверяет: $r \in \mathcal{R}$?
   - Если $r \notin \mathcal{R}$, то $\mathcal{R} := \mathcal{R} \cup \{r\}$, $B \to T$: деньги
   - Если $r \in \mathcal{R}$, то B сравнивает a_e в ЭБ и в своей БД.
     - Если они совпадают, то ЭБ скопировал T.
     - Если они не совпадают, то ЭБ скопировал A.
       Т.к. она уже использовалась при покупке у другого продавца T', который послал A список $e' \neq e$, то $\exists\, i$: одному из продавцов T, T' A послал α_{i1}, а другому – α_{i2}.
       B получает $id_A = \alpha_{i1} \oplus \alpha_{i2}$.

Межфакультетский курс «Интеллектуальные методы анализа протоколов безопасности» Лекция 5 Теория процессов. Примеры верификации свойств процессов

А.М.Миронов

1 Неформальное понятие процесса и примеры процессов

Прежде чем сформулировать формальное понятие процесса, мы опишем данное понятие неформально и рассмотрим простейшие примеры процессов.

1.1 Неформальное понятие процесса

Мы понимаем под процессом модель поведения динамической системы на некотором уровне абстракции.

Процесс можно представлять себе как граф P, компоненты которого имеют следующий смысл.

- Вершины графа P называются **состояниями** и изображают ситуации (или классы ситуаций), в которых может находиться моделируемая система в различные моменты своего функционирования.

  Одно из состояний является выделенным, оно называется **начальным состоянием** процесса P.

- Рёбра графа P имеют метки, изображающие **действия**, которые может исполнять моделируемая система.

- Функционирование процесса P описывается переходами по рёбрам графа P от одного состояния к другому. Функционирование начинается из начального состояния.

  Метка каждого ребра изображает действие процесса, исполняемое при переходе от состояния в начале ребра к состоянию в его конце.

1.2 Пример процесса

В качестве первого примера рассмотрим процесс, представляющий собой простейшую модель поведения некоторого торгового автомата.

Мы будем представлять себе этот автомат как машину, которая имеет монетоприемник, кнопку и лоток для выдачи товара. Когда покупатель хочет приобрести товар, он опускает монету в монетоприемник, нажимает на кнопку, и после этого в лотке появляется товар.

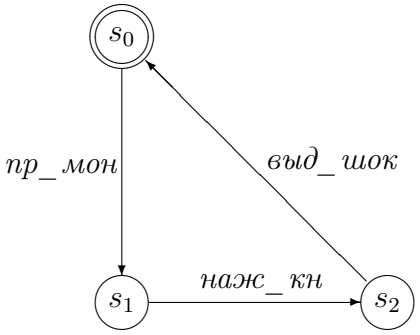
Предположим, что наш автомат торгует шоколадками по цене 1 монета за штуку. Опишем действия такого автомата.

- По инициативе покупателя в автомате могут происходить следующие действия: попадание в щель монеты и нажатие кнопки.
- В ответ автомат может осуществлять реакцию: выдавать в лоток шоколадку.

Обозначим действия короткими именами:

- прием монеты мы обозначим записью *пр_мон*,
- нажатие кнопки – записью *наж_кн*, и
- выдачу шоколадки – записью *выд_шок*.

Процесс нашего торгового автомата выглядит следующим образом:



(1)

Диаграмма (1) объясняет, как функционирует торговый автомат:

- вначале автомат находится в состоянии s_0, в этом состоянии он ожидает появления в приемнике монеты (то, что состояние s_0 является начальным, изображается на диаграмме двойным кружочком вокруг идентификатора этого состояния),
- когда монета появляется, автомат переходит в состояние s_1 и ждет нажатия на кнопку,
- после нажатия кнопки автомат переходит в состояние s_2, выдает шоколадку и возвращается в состояние s_0.

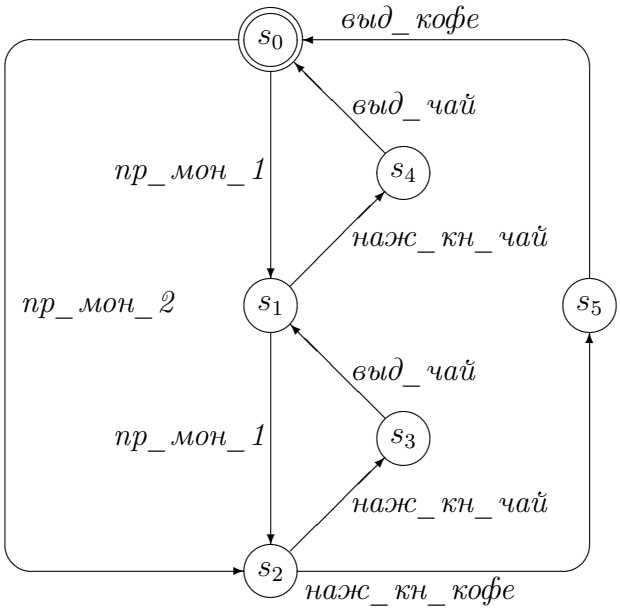
1.3 Другой пример процесса

Рассмотрим более сложный пример торгового автомата, который отличается от предыдущего тем, что продаёт два вида товаров: чай и кофе, причём стоимость чая – 1 рубль, а стоимость кофе – 2 рубля.

Автомат имеет две кнопки: одну – для чая, другую – для кофе.

Покупатель может платить монетами достоинством в 1 рубль и 2 рубля. Данные монеты будут обозначаться знакосочетаниями *мон_1* и *мон_2* соответственно. Если покупатель опустил в монетоприемник монету *мон_1*, он может купить только чай. Если же он опустил монету *мон_2*, он может купить кофе или два чая. Кофе можно купить также, опустив в монетоприёмник две монеты *мон_1*.

Процесс такого торгового автомата выглядит следующим образом:



Для формального определения понятия процесса мы должны уточнить понятие действия. Это уточнение излагается в параграфе 2.

2 Действия

Для задания процесса P, представляющего собой модель поведения некоторой динамической системы, должно быть указано множество $Act(P)$ **действий**, которые может выполнять процесс P. Будем предполагать, что действия всех процессов являются элементами некоторого универсального множества Act всех возможных действий, которые может выполнить какой-либо процесс, т.е. для любого процесса P $Act(P) \subseteq Act$.

Выбор множества $Act(P)$ действий процесса P зависит от целей моделирования. В разных ситуациях для представления модели анализируемой системы в виде некоторого процесса могут выбираться разные множества действий.

Будем предполагать, что Act делится на 3 следующих класса.

1. **Входные действия**, которые изображаются записями вида $?\alpha$. Действие вида $?\alpha$ понимается как прием объекта с именем α.

2. **Выходные действия**, которые изображаются записями вида $!\alpha$. Действие вида $!\alpha$ понимается как посылка объекта с именем α.

3. **Внутреннее** (или **невидимое**) действие, которое обозначается символом τ. Внутренним мы называем такое действие процесса P, которое не связано с его взаимодействием с **окружающей средой** (т.е. с процессами, являющимися внешними по отношению к процессу P и с которыми он может взаимодействовать). Например, внутреннее действие может быть связано со взаимодействием компонентов процесса P.

   В действительности внутренние действия могут быть самыми разнообразными, но для обозначения всех внутренних действий мы будем использовать один и тот же символ τ. Это отражает наше желание не различать все внутренние действия, т.к. они не являются «наблюдаемыми» извне процесса P.

Обозначим знакосочетанием $Names$ совокупность имён объектов, которые могут приниматься или посылаться процессами. Множество $Names$ предполагается бесконечным.

Множество Act представляет собой дизъюнкное объединение

$$Act = \{?\alpha \mid \alpha \in Names\} \sqcup \{!\alpha \mid \alpha \in Names\} \sqcup \{\tau\}.$$

Отметим, что объекты, которые принимаются и посылаются процессами, могут иметь самую различную природу (как материальную, так и нематериальную). Например, ими могут быть материальные ресурсы, люди, деньги, информация, энергия и т.д.

Кроме того, сами понятия приема и посылки могут иметь виртуальный характер, т.е. слова «прием» и «посылка» могут использоваться лишь как метафоры, а в действительности никакого приема или посылки какого-либо реального объекта может и не происходить.

Для каждого имени $\alpha \in Names$ действия $?\alpha$ и $!\alpha$ называются **комплементарными**. Будем использовать следующие обозначения:

- для каждого действия $a \in Act \setminus \{\tau\}$
  - $\bar{a}$ обозначает действие, комплементарное к a: $\overline{?\alpha} = !\alpha$, $\overline{!\alpha} = ?\alpha$,
  - $name(a)$ обозначает имя в a, т.е. $name(?\alpha) = name(!\alpha) = \alpha$,
- $\forall\, L \subseteq Act \setminus \{\tau\}\ \ \overline{L} \stackrel{\text{def}}{=} \{\overline{a} \mid a \in L\}$, $names(L) \stackrel{\text{def}}{=} \{name(a) \mid a \in L\}$.

3 Определение понятия процесса

Процессом (над множеством действий Act) называется тройка P вида (S, s^0, R), компоненты которой имеют следующий смысл:

- S – множество, элементы которого называются **состояниями**,
- $s^0 \in S$ – состояние, называемое **начальным состоянием**,
- $R \subseteq S \times Act \times S$, элементы множества R называются **переходами**, если переход из R имеет вид (s_1, a, s_2), то
  - будем говорить, что этот переход является переходом из состояния s_1 в состояние s_2 с выполнением действия a,
  - состояния s_1 и s_2 называются **началом** и **концом** этого перехода, а действие a называется **меткой** этого перехода, и
  - будем обозначать данный переход записью $s_1 \stackrel{a}{\rightarrow} s_2$.

Совокупность всех процессов обозначается записью $Proc$.

Процесс (S, s^0, R) можно представлять себе как граф с множеством вершин S, выделенной вершиной s^0, рёбра которого соответствуют переходам из R: если переход имеет вид $s_1 \stackrel{a}{\rightarrow} s_2$, то ему соответствует ребро с началом s_1, концом s_2 и меткой a.

Функционирование процесса $P = (S, s^0, R)$ заключается в порождении последовательности переходов вида $s^0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \dots$ и выполнении действий $a_0, a_1, a_2 \dots$, соответствующих этим переходам. Более подробно: на каждом шаге функционирования $i \geq 0$

- процесс находится в некотором состоянии s_i $(s_0 = s^0)$,
- если есть хотя бы один переход из R с началом в s_i, то P
  - недетерминированно выбирает переход с началом в s_i, помеченный таким действием a_i, которое можно выполнить в текущий момент времени (если таких переходов нет, то процесс временно приостанавливает свою работу до того момента, когда появится хотя бы один такой переход),
  - выполняет действие a_i, после чего переходит в состоние s_{i+1}, которое является концом выбранного перехода,
- если в R нет переходов с началом в s_i, то P заканчивает работу.

Ниже для каждого процесса P запись $Act(P)$ будет обозначать множество невнутренних действий процесса P:

$$Act(P) = \{a \in Act \setminus \{\tau\} \mid \exists\, s \xrightarrow{a} s' \in R\}.$$

Процесс (S, s^0, R) называется **конечным**, если S – конечное множество. Конечный процесс можно изображать диаграммой, в которой

- каждому состоянию соответствует кружочек на плоскости, в котором м.б. написан идентификатор (имя этого состояния),
- каждому переходу соответствует стрелка из начала этого перехода в его конец, на которой написана метка этого перехода,
- начальное состояние обозначается двойным кружочком.

Состояние s процесса $P = (S, s^0, R)$ называется

- **достижимым**, если $s = s^0$ или существует последовательность переходов в P, имеющая вид $s^0 \xrightarrow{a_1} \dots \xrightarrow{a_n} s$,
- **недостижимым**, если оно не является достижимым,
- **терминальным**, если не существует переходов с началом в s.

Будем считать процессы $P_1 = (S_1, s_1^0, R_1)$ и $P_2 = (S_2, s_2^0, R_2)$ равными, если они изоморфны как графы, т.е. существует биективное отображение $f : S_1 \to S_2$, такое, что $f(s_1^0) = s_2^0$, и для каждой пары вершин $s, s' \in S_1$ и каждого $a \in Act$ существует ребро $s \xrightarrow{a} s' \in R_1$ тогда и только тогда, когда существует ребро $f(s) \xrightarrow{a} f(s') \in R_2$.

Кроме того, будем считать процессы $P_1 = (S_1, s_1^0, R_1)$ и $P_2 = (S_2, s_2^0, R_2)$ равными, если P_2 получается из P_1 удалением недостижимых состояний и связанных с ними рёбер.

Процесс (S, s^0, R) называется **детерминированным**, если $\forall\, s \in S$, $\forall\, a \in Act$ существует не более одного $s' \in S$, такого, что $s \xrightarrow{a} s' \in R$.

4 Операции на процессах

В этом пункте мы определим операции на процессах, при помощи которых из одних процессов можно строить другие, более сложные процессы.

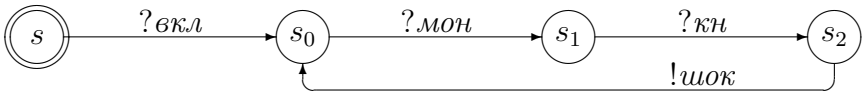
4.1 Префиксное действие

Первая такая операция – префиксное действие.

Пусть заданы процесс $P = (S, s^0, R)$ и действие $a \in Act$. Применением операции **префиксного действия** к паре (a, P) является процесс, обозначаемый записью $a; P$ и имеющий вид $(S \sqcup \{s\}, s, R \sqcup \{s \xrightarrow{a} s^0\})$, т.е.

- множество состояний процесса $a; P$ получается добавлением к S нового состояния s, которое будет начальным в $a; P$, и
- множество переходов процесса $a; P$ получается добавлением к R нового перехода $s \xrightarrow{a} s^0$.

Проиллюстрируем действие данной операции на примере торгового автомата из параграфа 1.2. Обозначим процесс, представляющий поведение этого автомата, записью $P_{\text{та}}$. Расширим множество действий данного автомата новым действием $?вкл$, которое будет означать включение этого автомата в сеть. Процесс $?вкл; P_{\text{та}}$ представляет поведение нового торгового автомата, который в начальном состоянии не может ни принимать монет, ни воспринимать нажатия на кнопку, ни выдавать шоколадок. Единственное, что он может, – это стать включенным, после этого его поведение ничем не будет отличаться от поведения исходного автомата. Графовое представление процесса $?вкл; P_{\text{та}}$ имеет вид

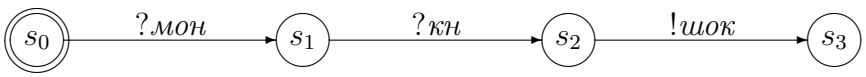


4.2 Пустой процесс

Среди всех процессов существует один наиболее простой. Этот процесс имеет всего одно состояние и не имеет переходов. Для обозначения такого процесса используется константа (т.е. нульарная операция) **0**.

Возвращаясь к примерам с торговыми автоматами, можно сказать, что процесс **0** представляет поведение сломанного автомата, то есть такого автомата, который вообще не может выполнять никаких действий.

Путем применения операций префиксного действия к процессу **0** можно определять поведение более сложных автоматов. Рассмотрим, например, процесс $P = ?мон; ?кн; !шок; \mathbf{0}$. Графовое представление этого процесса имеет вид



Этот процесс задает поведение автомата, который обслуживает ровно одного покупателя и после этого ломается.

4.3 Альтернативная композиция

Следующая операция на процессах – альтернативная композиция. Данная операция используется в том случае, когда по паре процессов P_1, P_2 требуется построить процесс P, который будет функционировать либо как процесс P_1, либо как процесс P_2, причём выбор процесса, в соответствии с которым P будет функционировать, может определяться как самим P, так и окружающей средой, в которой функционирует P.

Например, если P_1 и P_2 имеют вид $P_1 = ?\alpha; P_1'$, $P_2 = ?\beta; P_2'$ и в начальный момент времени окружающая среда может предложить P ввести объект α, но не может предложить P ввести объект β, то P должен выбрать то поведение, которое является единственно возможным в данной ситуации, т.е. работать так же, как процесс P_1.

Отметим, что в данном случае выбирается такой процесс, первое действие в котором может быть выполнено в текущий момент времени. Выбрав P_1 и выполнив действие $?\alpha$, процесс P обязан продолжать работу в соответствии со своим выбором, т.е. в соответствии с процессом P_1'.

Если же в начальный момент времени окружающая среда может предложить P ввести как объект α, так и объект β, то P недетерминированно (т.е. произвольно) выбирает процесс, в соответствии с которым он будет работать, или же этот выбор производится с учётом дополнительных факторов.

Операция альтернативной композиции определяется следующим образом. Пусть процессы P_1, P_2 имеют вид $P_i = (S_i, s_i^0, R_i)$ $(i = 1, 2)$, причём множества состояний S_1 и S_2 не имеют общих элементов (если S_1 и S_2 имеют общие элементы, то для определения процесса $P_1 + P_2$ в качестве второго слагаемого надо взять не сам P_2, а его изоморфную копию, которая дизъюнктна с P_1).

Альтернативная композиция процессов P_1 и P_2 – это процесс

$$P_1 + P_2 = (S_1 \sqcup S_2 \sqcup \{s^0\}, s^0, R),$$

где $R = R_1 \sqcup R_2 \sqcup \{s^0 \xrightarrow{a} s \mid s_i^0 \xrightarrow{a} s \in R_i, i = 1, 2\}$ (т.е. R получается добавлением к $R_1 \sqcup R_2$ переходов из начального состояния, дублирующих переходы из начальных состояний процессов-слагаемых).

Рассмотрим в качестве примера торговый автомат, который продаёт газированную воду, причём

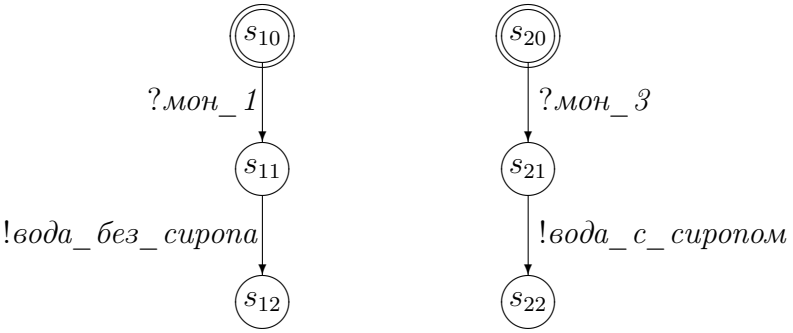
- если покупатель опускает в него монету *мон_ 1* достоинством в 1 копейку, то автомат выдаёт стакан воды без сиропа,
- а если покупатель опускает в него монету *мон_ 3* достоинством в 3 копейки, то автомат выдаёт стакан воды с сиропом

и сразу после продажи одного стакана воды автомат ломается.

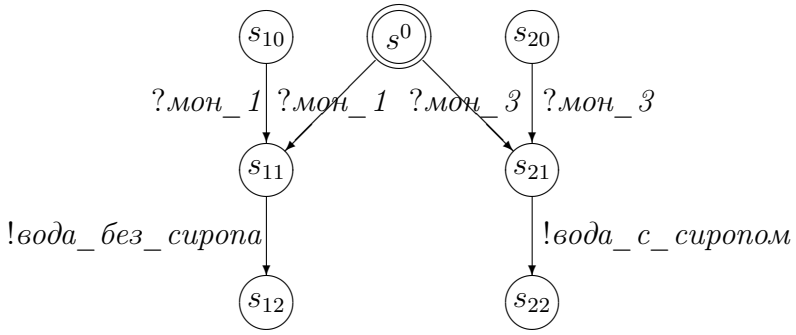
Поведение данного автомата описывается следующим процессом:

$$\begin{array}{rcl} P_{\text{газ\_вода}} & = & ?\textit{мон\_ 1}; !\textit{вода\_ без\_ сиропа}; \mathbf{0} \;\; + \\ & + & ?\textit{мон\_ 3}; !\textit{вода\_ с\_ сиропом}; \mathbf{0} \end{array} \qquad (2)$$

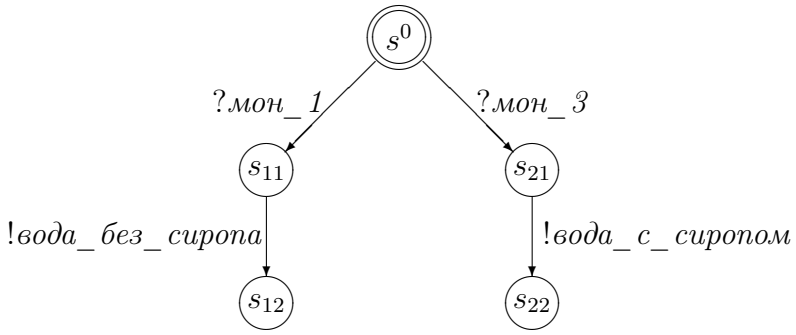
Графовые представления слагаемых в сумме (2) имеют вид



Согласно определению альтернативной композиции, графовое представление процесса (2) получается добавлением к предыдущей диаграмме нового состояния и переходов из этого состояния, дублирующих переходы из начальных состояний слагаемых, в результате чего получается следующая диаграмма:



Поскольку состояния s_{10} и s_{20} недостижимы, то их и связанные с ними переходы можно удалить, в результате чего получится диаграмма



которая и является искомым графовым представлением процесса (2).

Рассмотрим другой пример. Опишем разменный автомат, который может принимать купюры достоинством в 1000 рублей. Автомат должен выдать либо 2 купюры по 500 рублей, либо 10 купюр по 100 рублей, причем выбор способа размена осуществляется независимо от желания клиента. Сразу после одного сеанса размена автомат ломается.

$$P_{\text{размен}} = ?1\_no\_1000; (!2\_no\_500; \mathbf{0} + !10\_no\_100; \mathbf{0})$$

На этих двух примерах видно, что альтернативная композиция может использоваться для описания как минимум двух принципиально различных ситуаций.

1. Во-первых, она может выражать зависимость поведения системы от поведения её окружения.

   Например, в случае автомата $P_{\text{газ\_вода}}$ реакция автомата определяется действием покупателя, а именно достоинством монеты, которую он ввел в автомат. В данном случае процесс, представляющий

поведение моделируемого торгового автомата, является **детерминированным**, то есть его функционирование однозначно определяется входными действиями.

2. Во-вторых, на примере автомата $P_{\text{размен}}$ мы видим, что при одних и тех же входных действиях возможна различная реакция автомата. Это – пример **недетерминизма**, то есть неопределенности поведения системы. Неопределённость в поведении систем может происходить по крайней мере по двум причинам.

   (a) Во-первых, поведение систем может зависеть от **случайных факторов**. Такими факторами могут быть, например, сбои в аппаратуре, коллизии в компьютерной сети, отсутствие купюр необходимого достоинства в банкомате или что-либо еще.

   (b) Во-вторых, модель всегда есть некоторая абстракция или упрощение реальной системы. А при этом некоторые факторы, которые влияют на поведение этой системы, могут быть просто исключены из рассмотрения.

   В частности, на примере процесса $P_{\text{размен}}$ мы видим, что реальная причина выбора варианта поведения автомата может не учитываться в процессе, представляющем модель поведения этого автомата.

4.4 Параллельная композиция

Операция параллельной композиции используется для построения моделей поведения динамических систем, состоящих из взаимодействующих компонентов.

Если система состоит из двух компонентов, поведение которых описывается процессами P_1 и P_2, и функционирование системы заключается в совместном функционировании её компонентов, то поведение этой системы описывается процессом, который называется **параллельной композицией** процессов P_1 и P_2 и определяется следующим образом.

Пусть процессы P_1 и P_2 имеют вид $P_i = (S_i, s_i^0, R_i)$ $(i = 1, 2)$. **Параллельной композицией** процессов P_1 и P_2 называется процесс

$$P_1 \,|\, P_2 = (S_1 \times S_2, (s_1^0, s_2^0), R),$$

где R содержит следующие переходы:

- переход $(s_1, s_2) \xrightarrow{a} (s_1', s_2)$, если $s_1 \xrightarrow{a} s_1' \in R_1$, $s_2 \in S_2$,
- переход $(s_1, s_2) \xrightarrow{a} (s_1, s_2')$, если $s_1 \in S_1$, $s_2 \xrightarrow{a} s_2' \in R_2$,

- переход $(s_1, s_2) \xrightarrow{\tau} (s_1', s_2')$, если $s_1 \xrightarrow{a} s_1' \in R_1$, $s_2 \xrightarrow{\overline{a}} s_2' \in R_2$.

Таким образом, каждое выполнение действия процессом $P_1|P_2$ представляет собой

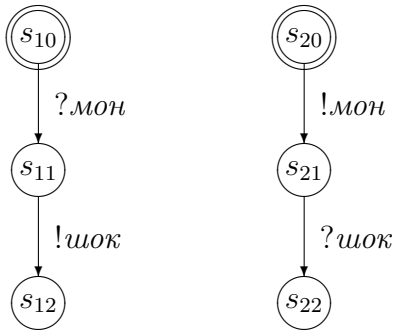
- либо одиночное выполнение действия одним из процессов P_1, P_2, во время выполнения которого другой из этих процессов приостанавливает свою работу,
- либо одновременное совместное выполнение действий обоими процессами P_1, P_2, которое заключается в том, что
  - один из этих процессов (P_i) передаёт другому процессу (P_j) некоторый объект и
  - процесс P_j в тот же самый момент времени принимает от процесса P_i этот объект,

  такой вид совместного выполнения называется **синхронным взаимодействием**.

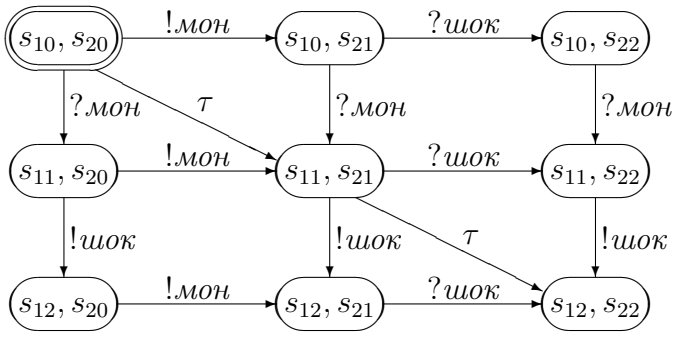
Приведём в качестве примера процесс $P_1 \,|\, P_2$, где процессы P_1 и P_2 представляют поведение торгового автомата и покупателя:

- $P_1 = ?\mathit{мон}; !\mathit{шок}; \mathbf{0}$, т.е. P_1 получает монету, выдаёт шоколадку и после этого ломается,
- $P_2 = !\mathit{мон}; ?\mathit{шок}; \mathbf{0}$, т.е. P_2 опускает монету, получает шоколадку и после этого завершает свою работу.

Графовые представления P_1 и P_2 имеют вид



Графовое представление процесса $P_1|P_2$ имеет вид



В данной диаграмме представлены все возможные варианты совместного функционирования P_1 и P_2. Каждое действие процесса $P_1|P_2$ м.б.

- синхронным взаимодействием: первое выполнение действия процессом $P_1|P_2$ м.б. результатом одновременного выполнения действий автоматом и покупателем (переход $(s_{10}, s_{20}) \xrightarrow{\tau} (s_{11}, s_{21})$: покупатель опускает в автомат монету, и автомат её принимает), или
- одиночным выполнением действия каким-либо из процессов P_1, P_2, например
  - переход $(s_{10}, s_{20}) \xrightarrow{!мон} (s_{10}, s_{21})$ интерпретируется следующим образом: покупатель опускает в автомат монету, но эта монета по каким-либо причинам не принимается автоматом, или
  - переход $(s_{11}, s_{21}) \xrightarrow{!шок} (s_{12}, s_{21})$ интерпретируется следующим образом: автомат выдал шоколадку, но покупатель по каким-либо причинам её не получил (например, к автомату подошёл вор и взял эту шоколадку, прежде чем её смог взять покупатель).

4.5 Ограничение

Пусть заданы процесс $P = (S, s^0, R)$ и подмножество $L \subseteq Names$.

Ограничением P по L называется процесс

$$P \setminus L = (S, s^0, \{s \xrightarrow{a} s' \mid a = \tau, \text{ или } name(a) \notin L\}),$$

т.е. $P \setminus L$ получается из P удалением переходов, метки которых содержат имена из L.

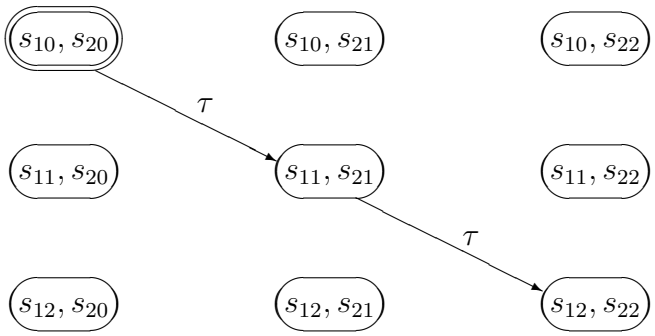
Операция ограничения используется, как правило, совместно с операцией параллельной композиции, для представления процессов, состоящих из нескольких компонентов, взаимодействие между которыми должно удовлетворять ограничениям. Эти ограничения заключаются в невозможности одиночного выполнения некоторых действий.

Например, пусть P_1 и P_2 – торговый автомат и покупатель, которые рассматривались в предыдущем параграфе. Мы хотели бы описать процесс, являющийся моделью такого параллельного функционирования процессов P_1 и P_2, при котором эти процессы могут совершать действия, связанные с покупкой-продажей шоколадки, только совместно.

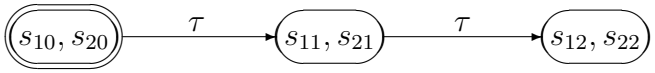
Искомый процесс имеет вид

$$P \stackrel{\text{def}}{=} (P_1|P_2) \setminus \{\textit{мон}, \textit{шок}\} \tag{3}$$

Графовое представление процесса (3) имеет вид



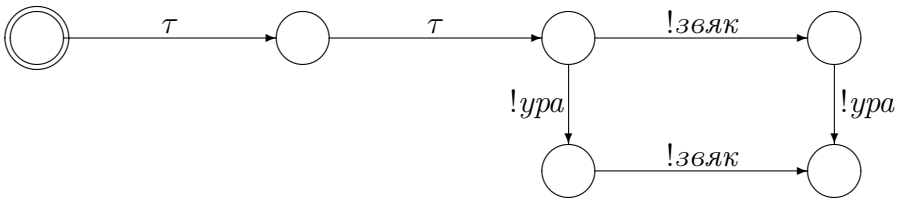
После удаления недостижимых состояний получится процесс, имеющий следующее графовое представление:



Рассмотрим другой пример. Немного изменим определения торгового автомата и покупателя: пусть они еще и сигнализируют об успешном выполнении своей работы, т.е. их процессы имеют следующий вид:

$$P_1 \stackrel{\text{def}}{=} ?\textit{мон}; !\textit{шок}; !\textit{звяк}; \mathbf{0}, \quad P_2 \stackrel{\text{def}}{=} !\textit{мон}; ?\textit{шок}; !\textit{ура}; \mathbf{0}$$

В этом случае графовое представление процесса (3) после удаления недостижимых состояний имеет вид



Данный процесс допускает одиночное выполнение тех невнутренних действий, которые не связаны с покупкой и продажей шоколадки.

4.6 Переименование

Пусть заданы процесс $P = (S, s^0, R)$ и функция $\theta : Names \to Names$.

Переименованием P относительно θ называется процесс

$$P^\theta = (S, s^0, \{s \xrightarrow{a^\theta} s' \mid s \xrightarrow{a} s' \in R\}),$$

где $(!\alpha)^\theta = !\theta(\alpha)$, $(?\alpha)^\theta = ?\theta(\alpha)$, $\tau^\theta = \tau$, т.е. P^θ получается из P заменой имён в метках переходов: каждое имя α заменяется на $\theta(\alpha)$.

Операция переименования позволяет многократно использовать один и тот же процесс P в качестве компоненты при построении более сложного процесса P'. Эта операция используется для предотвращения «конфликтов» между именами действий, используемых в различных вхождениях P в P'.

Если θ действует нетождественно лишь на имена $\alpha_1, \ldots, \alpha_n$ и отображает их в имена $\beta_1, \ldots, \beta_n$ соответственно, то для P^θ используется обозначение $P(\beta_1/\alpha_1, \ldots, \beta_n/\alpha_n)$.

5 Свойства операций на процессах

В этом параграфе мы приводим некоторые свойства определённых выше операций на процессах. В излагаемых ниже свойствах символы P, L и θ (возможно, с индексами) изображают произвольные процессы, множества имен и переименования соответственно.

1. $(P_1 + P_2) + P_3 = P_1 + (P_2 + P_3)$.

   Из данного свойства следует, что допустимы выражения вида

$$P_1 + \ldots + P_n, \tag{4}$$

так как при любой расстановке скобок в выражении (4) получится один и тот же процесс. Данный процесс можно описать явно следущим образом. Пусть процессы P_i $(i = 1, \ldots, n)$ имеют вид

$$P_i = (S_i, s_i^0, R_i) \quad (i = 1, \ldots, n), \tag{5}$$

причём множества $S_1, \ldots, S_n$ дизъюнктны. Тогда (4) имеет вид

$$(S_1 \sqcup \ldots \sqcup S_n \sqcup \{s^0\}, s^0, R),$$

где $\forall\, i = 1, \ldots, n$ R содержит все переходы из R_i и $\forall\, s_i^0 \xrightarrow{a} s \in R_i$ R содержит переход $s^0 \xrightarrow{a} s$.

2. $(P_1 \mid P_2) \mid P_3 = P_1 \mid (P_2 \mid P_3)$.

   Из данного свойства следует, что допустимы выражения вида

   $$P_1 \mid \ldots \mid P_n, \tag{6}$$

   так как при любой расстановке скобок в выражении (6) получится один и тот же процесс. Данный процесс можно описать явно следущим образом. Пусть процессы P_i $(i = 1, \ldots, n)$ имеют вид (5), тогда процесс (6) имеет вид

   $$P = (S_1 \times \ldots \times S_n, (s_1^0, \ldots, s_n^0), R),$$

   где R содержит следующие переходы:

   - $(s_1, \ldots, s_n) \xrightarrow{a} (s_1, \ldots, s_{i-1}, s_i', s_{i+1}, \ldots, s_n)$, где $s_i \xrightarrow{a} s_i' \in R_i$,
   - $(s_1, \ldots, s_n) \xrightarrow{\tau} (s_1, \ldots, s_{i-1}, s_i', s_{i+1}, \ldots, s_{j-1}, s_j', s_{j+1}, \ldots, s_n)$, где для некоторого $a \in Act \setminus \{\tau\}$ $s_i \xrightarrow{a} s_i' \in R_i$, $s_j \xrightarrow{\bar{a}} s_j' \in R_j$.

3. $P_1 + P_2 = P_2 + P_1$.

4. $P_1 \mid P_2 = P_2 \mid P_1$.

5. $P \mid \mathbf{0} = P$.

6. $\mathbf{0} \setminus L = \mathbf{0}$.

7. $\mathbf{0}^\theta = \mathbf{0}$.

8. $P \setminus L = P$, если $L \cap names(Act(P)) = \emptyset$.

9. $(a; P) \setminus L = \begin{cases} \mathbf{0}, & \text{если } a \neq \tau \text{ и } name(a) \in L, \\ a; (P \setminus L) & \text{иначе.} \end{cases}$

10. $(P_1 + P_2) \setminus L = (P_1 \setminus L) + (P_2 \setminus L)$.

11. $(P_1 \mid P_2) \setminus L = (P_1 \setminus L) \mid (P_2 \setminus L)$, если

$$L \cap names(Act(P_1) \cap \overline{Act(P_2)}) = \emptyset.$$

12. $(P \setminus L_1) \setminus L_2 = P \setminus (L_1 \cup L_2)$.

13. $P^\theta \setminus L = (P \setminus \theta^{-1}(L))^\theta$.

14. $P^{\theta_{id}} = P$, где θ_{id} – тождественная функция.

15. $P^\theta = P^{\theta'}$, если $\forall\, \alpha \in names(Act(P))\ \ \theta(\alpha) = \theta'(\alpha)$.

16. $(a; P)^\theta = a^\theta; (P^\theta)$.

17. $(P_1 + P_2)^\theta = P_1^\theta + P_2^\theta$.

18. $(P_1 \mid P_2)^\theta = P_1^\theta \mid P_2^\theta$, если сужение θ на $names(Act(P_1) \cup Act(P_2))$ является инъективной функцией.

19. $(P \setminus L)^\theta = P^\theta \setminus \theta(L)$, если функция θ инъективна.

20. $P^{\theta\theta'} = P^(\theta' \circ \theta)$.

6 Эквивалентности процессов

В этой главе мы излагаем несколько определений понятия эквивалентности процессов. Выбор того или иного варианта определения эквивалентности процессов для его применения в конкретной ситуации должен определяться тем, как именно в данной ситуации понимается одинаковость процессов. В параграфах 6.1 и 6.2 вводятся понятия простой и сильной эквивалентности процессов. Данные понятия используются в тех ситуациях, когда все действия, выполняющиеся в процессах, имеют одинаковый статус. В параграфах **??** и **??** предлагаются другие варианты понятия эквивалентности процессов: наблюдаемая эквивалентность и наблюдаемая конгруэнция. Данные понятия используются в тех ситуациях, когда действие τ рассматривается как ненаблюдаемое.

С каждым из вариантов понятия эквивалентности процессов связаны естественные задачи: распознавание для двух заданных процессов, являются ли они эквивалентными и построение по заданному процессу P процесса P', имеющего минимальное число состояний среди всех процессов, которые эквивалентны P.

6.1 Простая эквивалентность

6.1.1 Поведение процесса

Будем использовать следующие обозначения: $\forall\, P = (S, s^0, R) \in Proc$

- $\forall\, s \in S$ запись P^s обозначает процесс (S, s, R) (т.е. P^s отличается от P только начальным состоянием),
- $\forall\, P' \in Proc$, $\forall\, a \in Act$ запись $P \xrightarrow{a} P'$ обозначает утверждение: в P имеется ребро вида $s^0 \xrightarrow{a} s$ и $P' = P^s$.

Утверждение $P \xrightarrow{a} P'$ может интерпретироваться следующим образом: P может выполнить действие a, после чего вести себя как P'.

Запись $P \xrightarrow{a} P'$ называется **переходом** из P.

Поведением процесса P называется дерево $Beh(P)$,

- каждая вершина v которого помечена некоторым процессом P^v,
- каждое ребро которого помечено некоторым действием из Act и
- для каждой вершины $v \in Beh(P)$ множество ребер, выходящих из v, находится во взаимно однозначном соответствии с множеством переходов из P^v: ребру $v \xrightarrow{a} v'$ соответствует переход вида $P^v \xrightarrow{a} P^{v'}$.

6.1.2 Понятие простой эквивалентности

Будем говорить, что процессы $P, P' \in Proc$ находятся в отношении **простой эквивалентности**, если деревья $Beh(P)$ и $Beh(P')$ изоморфны, т.е. существует биективное отображение f множества вершин дерева $Beh(P)$ на множество вершин дерева $Beh(P')$, такое, что $\forall\, v, v' \in Beh(P)$, $\forall\, a \in Act$ существует ребро $v \xrightarrow{a} v'$ тогда и только тогда, когда существует ребро $f(v) \xrightarrow{a} f(v')$.

$\forall\, P, P' \in Proc$ запись $P \equiv P'$ обозначает, что процессы P и P' находятся в отношении простой эквивалентности.

6.1.3 Примеры процессов, находящихся в отношении простой эквивалентности

В этом пункте мы приводим несколько примеров процессов, находящихся в отношении простой эквивалентности. Обоснование утверждений о простой эквивалентности этих процессов является несложным.

1. $P \equiv P'$, где P и P' имеют следующий вид:

2. Если графы процессов P и P' изоморфны или P' получается из P удалением недостижимых состояний, то $P \equiv P'$.

3. $\forall\, P \in Proc \;\; P + \mathbf{0} \equiv P$.

4. Если $P = (S, s^0, R)$ и список всех ребер, выходящих из s^0, имеет вид $\{s^0 \xrightarrow{a_i} s_i \mid i = 1, \ldots, n\}$, то $P \equiv a_1; P^{s_1} + \ldots + a_n; P^{s_n}$.

5. Пусть процесс P имеет вид $P_1| \ldots |P_n$, где

$$\forall\, i = 1, \ldots, n \quad P_i \equiv a_{i1}; P_{i1} + \ldots + a_{in_i}; P_{in_i}. \tag{7}$$

   Тогда $P \equiv P'$, где P' является суммой

   - всех процессов вида

$$a_{ij}; (P_1| \ldots |P_{i-1}|P_{ij}|P_{i+1}| \ldots |P_n),$$

     где $i = 1, \ldots, n,\; j = 1, \ldots, n_i$, и
   - всех процессов вида

$$\tau; (P_1| \ldots |P_{i-1}|P_{ik}|P_{i+1}| \ldots |P_{j-1}|P_{jl}|P_{j+1}| \ldots |P_n),$$

     где $1 \le i < j \le n, \quad a_{ik}, a_{jl} \neq \tau$, и $a_{ik} = \overline{a_{jl}}$.

6. **Теорема о разложении**: пусть P имеет вид $(P_1| \ldots |P_n)\backslash L$, причем верно (7). Тогда $P \equiv P'$, где P' является суммой

   - всех процессов вида

$$a_{ij}; \Big((P_1| \ldots |P_{i-1}|P_{ij}|P_{i+1}| \ldots |P_n) \setminus L\Big),$$

     где $i = 1, \ldots, n,\; j = 1, \ldots, n_i$, $a_{ij} = \tau$ или $name(a_{ij}) \notin L$, и
   - всех процессов вида

$$\tau; \Big((P_1| \ldots |P_{i-1}|P_{ik}|P_{i+1}| \ldots |P_{j-1}|P_{jl}|P_{j+1}| \ldots |P_n) \setminus L\Big),$$

     где $1 \le i < j \le n, \quad a_{ik}, a_{jl} \neq \tau$, и $a_{ik} = \overline{a_{jl}}$.

6.2 Сильная эквивалентность

6.2.1 Понятие сильной эквивалентности

Другим вариантом понятия эквивалентности процессов является **сильная эквивалентность**. Данное понятие основано на следующем понимании эквивалентности процессов: если мы рассматриваем процессы P_1 и P_2 как эквивалентные, то должно быть выполнено условие: если один из этих процессов (P_i) может выполнить некоторое действие $a \in Act$ и после этого вести себя как некоторый процесс P_i' (т.е. $P_i \xrightarrow{a} P_i'$), то другой процесс (P_j) должен обладать способностью выполнить то же самое действие a, после чего вести себя как процесс P_j' (т.е. $P_j \xrightarrow{a} P_j'$), который эквивалентен P_i'.

Обозначим символом $\mathcal{M}$ совокупность всех отношений μ на множестве $Proc$, удовлетворяющих условию: $\forall\,(P_1, P_2) \in \mu$

$$\begin{array}{l} \forall\, a \in Act, \forall\, P_1' : P_1 \xrightarrow{a} P_1' \;\; \exists\, P_2' : P_2 \xrightarrow{a} P_2' \text{ и } (P_1', P_2') \in \mu, \\ \forall\, a \in Act, \forall\, P_2' : P_2 \xrightarrow{a} P_2' \;\; \exists\, P_1' : P_1 \xrightarrow{a} P_1' \text{ и } (P_1', P_2') \in \mu. \end{array} \tag{8}$$

Естественно считать процессы P_1, P_2 эквивалентными, если существует хотя бы одно отношение $\mu \in \mathcal{M}$, содержащее пару (P_1, P_2). Данное свойство можно выразить соотношением $(P_1, P_2) \in \bigcup_{\mu\in\mathcal{M}} \mu$.

Определим $\sim$ как отношение $\bigcup_{\mu\in\mathcal{M}} \mu$. Данное отношение называется **сильной эквивалентностью**. Нетрудно видеть, что $\sim \in \mathcal{M}$.

Теорема 1
$\sim$ является отношением эквивалентности.

Доказательство.

- $\sim$ рефлексивно, т.к. $Id_{Proc} = \{(P, P) \mid P \in Proc\} \in \mathcal{M}$,
- $\sim$ симметрично, т.к. из того, что $\forall\, \mu \in \mathcal{M} \;\; \mu^{-1} \in \mathcal{M}$, следует, что

$$\sim^{-1} = (\bigcup_{\mu\in\mathcal{M}} \mu)^{-1} = \bigcup_{\mu\in\mathcal{M}} (\mu^{-1}) \subseteq \bigcup_{\mu\in\mathcal{M}} \mu = \sim,$$

- $\sim$ транзитивно, т.к. если $\mu_1, \mu_2 \in \mathcal{M}$, то $\mu_1 \circ \mu_2 \in \mathcal{M}$, поэтому

$$\sim \circ \sim = (\bigcup_{\mu_1\in\mathcal{M}} \mu_1) \circ (\bigcup_{\mu_2\in\mathcal{M}} \mu_2) = \bigcup_{\mu_1,\mu_2\in\mathcal{M}} (\mu_1 \circ \mu_2) \subseteq \bigcup_{\mu\in\mathcal{M}} \mu = \sim . \;\blacksquare$$

Процессы $P_1, P_2 \in Proc$ называются **сильно эквивалентными**, если $(P_1, P_2) \in \sim$. Если процессы P_1 и P_2 сильно эквивалентны, то этот факт обозначается записью $P_1 \sim P_2$.

Нетрудно доказать, что

$$\text{если } P \equiv P', \text{ то } P \sim P'. \tag{9}$$

6.2.2 Критерий сильной эквивалентности, основанный на понятии бимоделирования

Пусть заданы два процесса: $P_i = (S_i, s_i^0, R_i)$ $(i = 1, 2)$. **Бимоделированием (БМ)** между P_1 и P_2 называется отношение $\rho \subseteq S_1 \times S_2$, содержащее пару (s_1^0, s_2^0) и удовлетворяющее условию: $\forall\, (s_1, s_2) \in \rho$

$$\begin{array}{l} \forall\, a \in Act, \forall\, s_1' : s_1 \xrightarrow{a} s_1' \;\; \exists\, s_2' : s_2 \xrightarrow{a} s_2' \text{ и } (s_1', s_2') \in \rho, \\ \forall\, a \in Act, \forall\, s_2' : s_2 \xrightarrow{a} s_2' \;\; \exists\, s_1' : s_1 \xrightarrow{a} s_1' \text{ и } (s_1', s_2') \in \rho. \end{array} \tag{10}$$

Теорема 2
$P_1 \sim P_2$ тогда и только тогда, когда существует БМ между P_1 и P_2.

Доказательство.

1. Пусть $P_1 \sim P_2$. Определим отношение $\rho \subseteq S_1 \times S_2$:

$$\rho \stackrel{\text{def}}{=} \{(s_1, s_2) \in S_1 \times S_2 \mid P_1^{s_1} \sim P_2^{s_2}\}.$$

Докажем что ρ является БМ между P_1 и P_2.

Так как $P_1 = P_1^{s_1^0}$ и $P_2 = P_2^{s_2^0}$, то из $P_1 \sim P_2$ следует, что $(s_1^0, s_2^0) \in \rho$.

Докажем первое свойство в (10): из $(s_1, s_2) \in \rho$ и $s_1 \xrightarrow{a} s_1'$ следует

$$\exists\, s_2' : s_2 \xrightarrow{a} s_2', (s_1', s_2') \in \rho. \tag{11}$$

Из $(s_1, s_2) \in \rho$ следует $P_1^{s_1} \sim P_2^{s_2}$. Из $s_1 \xrightarrow{a} s_1'$ следует $P_1^{s_1} \xrightarrow{a} P_1^{s_1'}$.

Так как $\sim \in \mathcal{M}$, то $\exists\, P_2' : P_2^{s_2} \xrightarrow{a} P_2'$ и $P_1^{s_1'} \sim P_2'$.

Согласно определению понятия перехода из процесса, из $P_2^{s_2} \xrightarrow{a} P_2'$ следует, что $\exists\, s_2' : s_2 \xrightarrow{a} s_2'$ и $P_2' = P_2^{s_2'}$.

Из $P_1^{s_1'} \sim P_2^{s_2'}$ следует, что $(s_1', s_2') \in \rho$. Таким образом, верно (11).

Аналогично доказывается второе свойство в (10).

2. Пусть существует БМ ρ между P_1 и P_2. Докажем, что $P_1 \sim P_2$ путем построения отношения $\mu \in \mathcal{M}$, содержащего пару (P_1, P_2).

   Определим $\mu \stackrel{\text{def}}{=} \{(P_1^{s_1}, P_2^{s_2}) \mid (s_1, s_2) \in \rho\}$.

   Так как $(s_1^0, s_2^0) \in \rho$, то $(P_1, P_2) = (P_1^{s_1^0}, P_2^{s_2^0}) \in \mu$.

   Для доказательства соотношения $\mu \in \mathcal{M}$ докажем первое свойство в (8): из соотношений $(P_1^{s_1}, P_2^{s_2}) \in \mu$ и $P_1^{s_1} \stackrel{a}{\to} P_1'$ следует, что

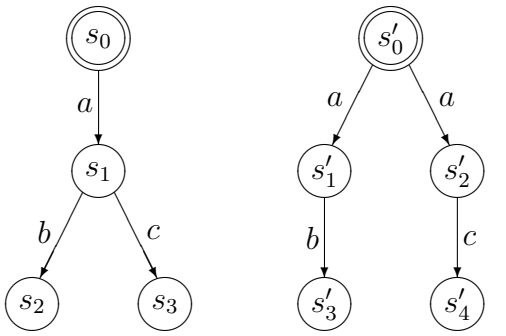
$$\exists\, P_2' : P_2^{s_2} \stackrel{a}{\to} P_2' \text{ и } (P_1', P_2') \in \mu. \tag{12}$$

   Так как $(s_1, s_2) \in \rho$ и из $P_1^{s_1} \stackrel{a}{\to} P_1'$ следует, что $\exists\, s_1' : s_1 \stackrel{a}{\to} s_1'$ и $P_1' = P_1^{s_1'}$, то по определению БМ $\exists\, s_2' : s_2 \stackrel{a}{\to} s_2'$, $(s_1', s_2') \in \rho$, поэтому в качестве P_2' в (12) можно взять $P_2^{s_2'}$.

   Аналогично доказывается второе свойство в (8). ■

Используя теорему 2, можно обосновать следующие утверждения.

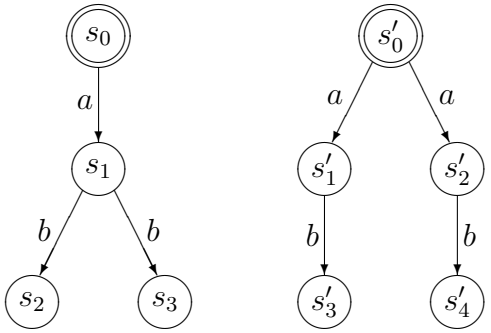
1. Процессы



(13)

   не являются сильно эквивалентными. Действительно, если они сильно эквивалентны, то по теореме 2 существует БМ ρ, содержащее пару (s_0, s_0'). Из $(s_0, s_0') \in \rho$ и $s_0' \stackrel{a}{\to} s_1'$ следует, что $\exists\, s : s_0 \stackrel{a}{\to} s$ и $(s, s_1') \in \rho$. Так как из s_0 выходит единственное ребро, то $s = s_1$. Из $(s_1, s_1') \in \rho$ и $s_1 \stackrel{c}{\to} s_3$ следует, что $\exists\, s : s_1' \stackrel{c}{\to} s$ и $(s_3, s) \in \rho$. Однако ребра вида $s_1' \stackrel{c}{\to} s$ не существует.

2. Процессы



сильно эквивалентны. БМ, обосновывающее это утверждение, имеет вид $\{(s_0, s'_0), (s_1, s'_1), (s_1, s'_2), (s_2, s'_3), (s_2, s'_4), (s_3, s'_3), (s_3, s'_4)\}$.

Межфакультетский курс «Интеллектуальные методы анализа протоколов безопасности» Лекция 6 Описание и спецификация протокола с чередованием битов

А.М.Миронов

1 Протоколы передачи данных

Важными примерами процессов являются **протоколы передачи данных** (называемые также просто **протоколами**). В этом параграфе рассматриваются некоторые протоколы передачи данных.

Каждый протокол представляет собой параллельную композицию процессов, выполняющих формирование, отправление, приём и обработку сообщений. Такие процессы называются **агентами** протокола, а сообщения, пересылаемые от одного агента другому, называются **кадрами** (**frame**). Агенты пересылают свои сообщения друг другу через каналы. Кадры в каналах могут искажаться или пропадать (например, в результате воздействия радиопомех). Поэтому каждый кадр должен содержать не только ту информацию, которую один агент желает передать другому, но также и средства, позволяющие получателю этого кадра выяснить, был ли этот кадр искажён в процессе передачи.

1.1 Однонаправленный протокол с чередующимися битами

Первый из рассматриваемых нами протоколов передачи данннх состоит из двух агентов: **отправителя** (*Sender*) и **получателя** (*Receiver*). Задачей протокола является организация надёжной доставки кадров от

отправителя к получателю через ненадёжный канал связи (который может искажать и терять передаваемые кадры). Данный протокол называется **однонаправленным протоколом с чередующимися битами**, в англоязычной литературе он называется **one-way Alternating Bit Protocol**, или, сокращённо, **one-way ABP**.

Работа протокола происходит следующим образом.

1. **Отправитель** получает сообщения от своего источника данных, эти сообщения называются **пакетами**. Задача отправителя заключается в периодическом выполнении следующих действий:

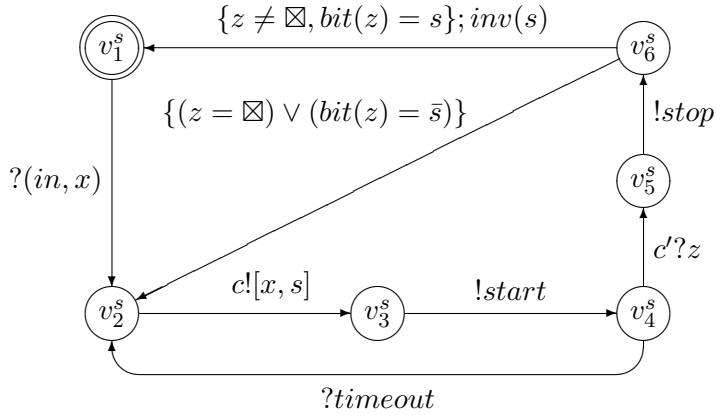
   - получить с порта in очередной пакет от своего источника данных, сформировать из этого пакета кадр, послать этот кадр в канал c и включить таймер действием $!start$,
   - если таймер пришлёт сигнал $timeout$ (который означает, что время ожидания подтверждения посланного кадра закончилось и, видимо, этот кадр до получателя не дошёл), то послать этот кадр в канал c ещё раз,
   - если придёт подтверждение от получателя (через канал c'), то это означает, что текущий кадр дошёл до него успешно, и нужно выключить таймер действием $!stop$, получить следующий пакет от своего источника данных и т.д.

   Механизм, с помощью которого получатель может отличить новый кадр от кадра, переданного повторно, реализован в данном протоколе следующим образом: среди переменных отправителя и получателя присутствуют булевы переменные s и r соответственно, где s – чётность номера очередного кадра, которого пытается послать отправитель, и r – чётность номера очередного кадра, которого ожидает получатель. В начальный момент $s = r = 0$.

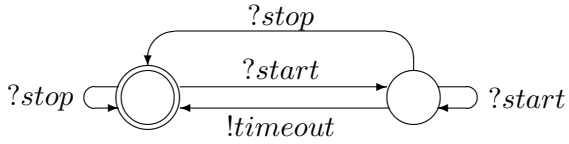
   Будем обозначать кадр, соответствующий пакету x, с которым связан бит s, записью $[x, s]$. Для извлечения пакета и бита из кадра используются функции $info$ и bit соответственно, обладающие свойствами:

   $$info([x, s]) = x, \quad bit([x, s]) = s.$$

   Работа отправителя представляется нижеследующим процессом P, в котором запись $inv(s)$ обозначает присваивание $s := 1-s$ и запись $\bar{s}$ обозначает терм $1-s$. Ниже записи вида $inv(x)$ и $\bar{x}$, где x – булева переменная, имеют тот же смысл.



Поведение таймера, с которым взаимодействует P, представляется процессом $Timer$, он имеет вид

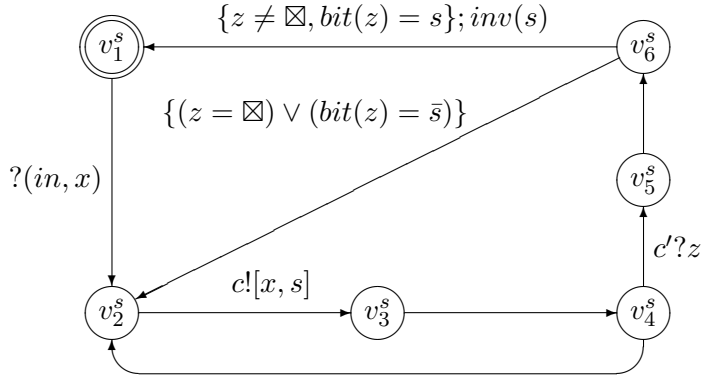


(1)

Процесс отправителя $Sender$ имеет вид

$$(P|Timer) \setminus \{start, stop, timeout\}.$$

Нетрудно доказать, что $Sender$ наблюдаемо эквивалентен процессу

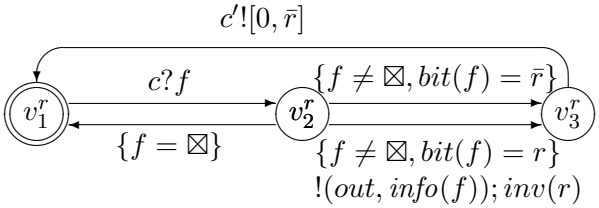


2. **Получатель** периодически выполняет следующие действия:

   - получение из канала c очередного кадра f и проверка наличия искажений в полученном кадре f,
   - если кадр f искажён, то получатель его игнорирует (предполагая, что отправитель, не дождавшись подтверждения, пошлёт это кадр ещё раз),

- если кадр f не искажён, то извлечение из него бита $bit(f)$, и
  - если $bit(f)$ совпадает с ожидаемым битом r, то извлечение из этого кадра пакета $info(f)$ и передача этого пакета с порта *out* потребителю данных и инвертирование бита r,
  - посылка отправителю через канал c' подтверждения полученного кадра (которое имеет вид $[0, \bar{r}]$).

Описанный выше алгоритм представляется процессом *Receiver*:

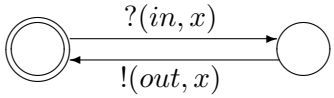


(2)

Поведение всего протокола описывается процессом

$$\begin{array}{l} Protocol \stackrel{\text{def}}{=} (Sender \mid Receiver), \\ Init(Protocol) = \{s = r = 0, c = c' = \varepsilon\}. \end{array} \tag{3}$$

Спецификация процесса (3) заключается в том, что данный процесс наблюдаемо эквивалентен следующему процессу *Buffer*:



(4)

1.2 Двунаправленный протокол передачи сообщений с чередующимися битами

В большинстве ситуаций пересылки данных между двумя агентами требуется **двунаправленная передача**, т.е. передача кадров в обоих направлениях, где каждый из агентов выступает как в роли отправителя, так и в роли получателя. Работа агентов в таких ситуациях выглядит следующим образом: если агент B успешно принял кадр f от агента A, то он посылает подтверждение получения кадра f в составе своего кадра, содержащего пакет для A.

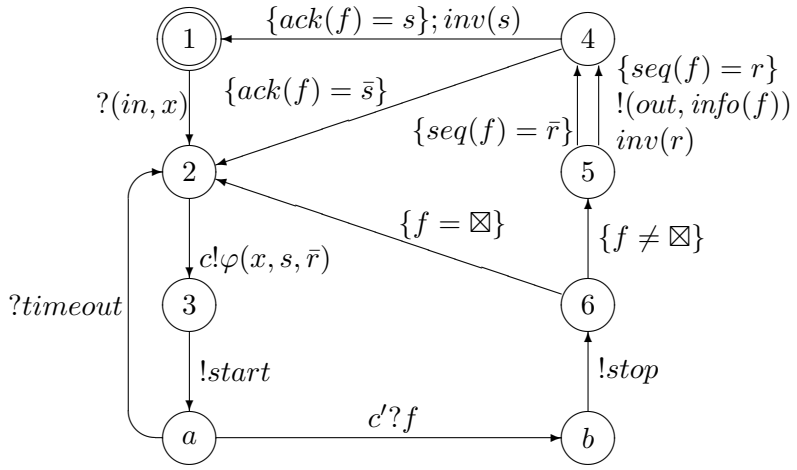
Простейшим протоколом двунаправленной надежной передачи является **двунаправленный протокол с чередующимися битами** (two-way ABP). Поведение каждого из участников этого протокола описывается одним и тем же процессом P, совмещающим в себе функции отправителя и получателя. У каждого из участников имеется свой источник данных (от которого он получает пакеты, пересылаемые в составе кадров

другому участнику) и свой потребитель данных (которому он передает пакеты, получаемые из кадров от другого участника).

Каждый пересылаемый кадр f имеет вид $[x, s, r]$, где x – пакет, s – бит, сопоставленный пакету x, r – бит подтверждения последнего полученного неискажённого кадра. Для извлечения пакетов и битов из кадров используются функции *info*, *seq* и *ack*, обладающие свойствами:

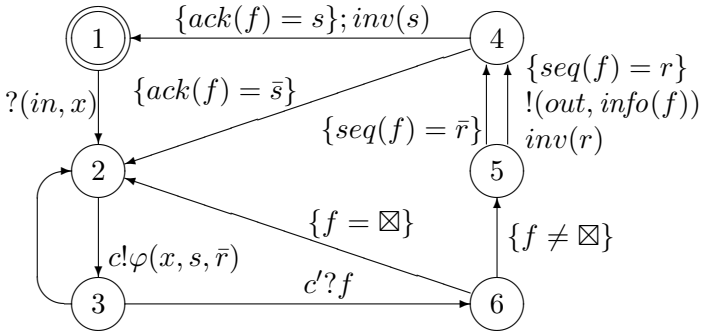
$$info([x, s, r]) = x, \quad seq([x, s, r]) = s, \quad ack([x, s, r]) = r.$$

Алгоритм работы участников данного протокола является объединением алгоритмов работы отправителя и получателя в протоколе из предыдущего пункта и представляется следующим процессом P:



Процесс *Agent*, описывающий поведение агента данного протокола, имеет вид $(P|Timer) \setminus \{start, stop, timeout\}$, где $Timer$ – процесс (1).

Нетрудно доказать, что *Agent* наблюдаемо эквивалентен процессу



(5)

Поведение всего протокола описывается процессом

$$\begin{array}{l} Protocol \stackrel{\text{def}}{=} (Agent_1 \,|\, Agent_2), \\ Init(Protocol) = \{s_1 = 0, r_1 = 0, s_2 = 0, r_2 = 0, c_{12} = \varepsilon, c_{21} = \varepsilon\}, \end{array} \tag{6}$$

где $Agent_1$ и $Agent_2$ получаются из (5) приписыванием индекса 1 или 2 соответственно к каждому имени и к каждой переменной, не являющейся каналом, заменой в $Agent_1$ каналов c и c' на c_{12} и c_{21} соответственно и заменой в $Agent_2$ каналов c и c' на c_{21} и c_{12} соответственно.

Межфакультетский курс «Интеллектуальные методы анализа протоколов безопасности» Лекция 7 Протоколы передачи данных через небезопасную среду

А.М.Миронов

Одними из широко используемых протоколов передачи данных через небезопасную среду являются протоколы скользящего окна.

1 Протокол скользящего окна с возвратом

В рассмотренных в предыдущей лекции протоколах каждый следующий кадр посылается только после получения подтверждения об успешной доставке текущего кадра. В этом и следующем пунктах рассматриваются протоколы двунаправленной передачи, в которых отправитель может послать в канал несколько кадров подряд, не дожидаясь подтверждений.

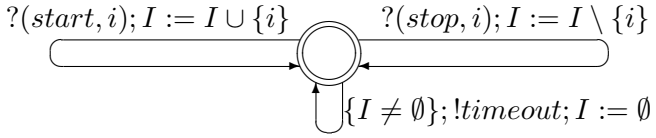
Первый из таких протоколов называется **протоколом скользящего окна (ПСО) с возвратом**. Так же как и в предыдущем протоколе, каждый из участников этого протокола совмещает функции отправителя и получателя. Процесс P, описывающий поведение участника этого протокола, содержит среди своих переменных массив x размера n, в компонентах $x_0, \ldots, x_{n-1}$ которого могут содержаться отправленные, но ещё не подтверждённые пакеты. Совокупность компонентов массива x, в которых содержатся такие пакеты в текущий момент времени, называется **окном**. Мы рассматриваем совокупность $x_0, \ldots, x_{n-1}$ компонентов массива x как «свёрнутую в кольцо», и окно является связным подмножеством этого кольца. С окном связаны две переменные из $Var(P)$: b (нижняя граница окна) и s (верхняя граница окна), их значениями являются элементы $\mathbf{Z}_n = \{0, 1, \ldots, n-1\}$. Если в текущий момент $b = s$, то

окно является пустым, иначе окно имеет вид $\{x_b, x_{b+1}, \ldots, x_{s-1}\}$, где все арифметические операции в этом и следующем параграфе выполняются по модулю n. Запись $[b, s[$ обозначает множество $\{b, b+1, \ldots, s-1\}$, т.е. совокупность индексов тех компонентов массива x, которые входят в окно. $Var(P)$ содержит переменную w, значение которой равно размеру окна (т.е. числу компонентов массива x, входящих в окно), $b + w = s$.

Каждый пересылаемый кадр f имеет вид $[x, s, r]$, где x – пакет, $s, r \in \mathbf{Z}_n$, s – номер, сопоставленный пакету x и кадру f, r – номер последнего полученного неискажённого кадра. Для извлечения пакетов и номеров из кадров используются функции *info*, *seq* и *ack*, где

$$info([x, s, r]) = x, \quad seq([x, s, r]) = s, \quad ack([x, s, r]) = r.$$

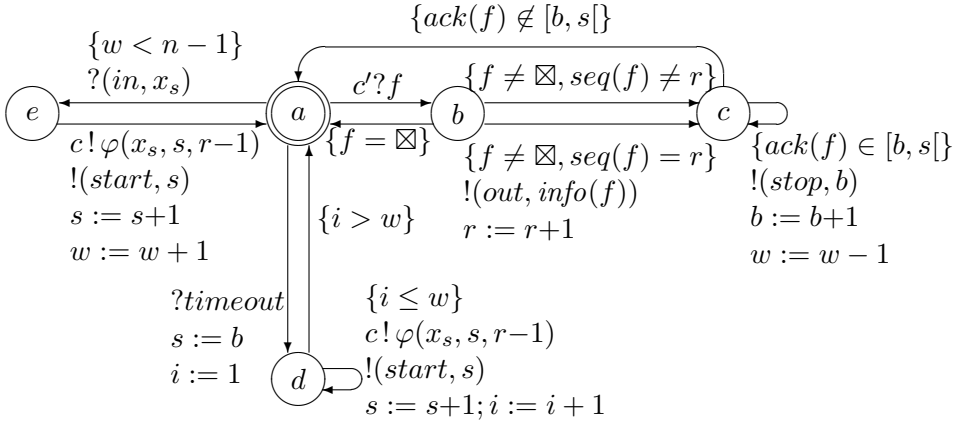
Каждая компонента i массива x связана с соответствующим таймером, который определяет продолжительность ожидания подтверждения от другого агента получения им пакета x_i. Поведение совокупности этих таймеров представляется процессом $Timers$, который имеет вид



(1)

$Timers$ имеет переменную i со значениями в $\mathbf{Z}_n$ и переменную I со значениями в $\mathbf{2}^{\mathbf{Z}_n}$, значением I в текущий момент является множество номеров включенных таймеров в этот момент. $Init(Timers) = \{I = \emptyset\}$.

Поведение каждого из участников этого протокола описывается следующим процессом P:



(2)

Поясним смысл некоторых компонентов процесса P.

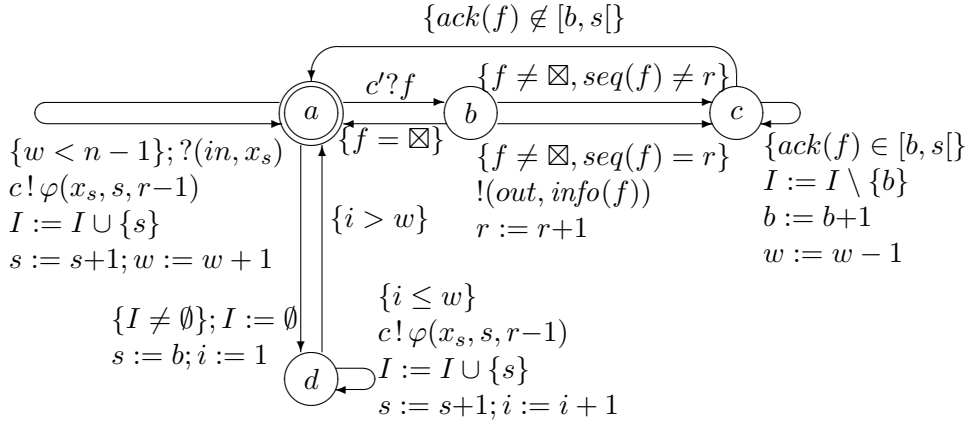
- Значение переменной r равно номеру ожидаемого кадра.

- Переход $a \to e$ означает получение нового пакета от источника данных, который записывается в x_s, s считается номером этого пакета.
- Переход $c \to c$ означает удаление подтверждённого пакета из окна. Если агент получает кадр, третья компонента r которого (т.е. номер подтверждения) такова, что $r \in [b, s[$, то все пакеты с номерами из $[b, r[$ тоже считаются подтверждёнными и удаляются из окна (даже если их подтверждения не дошли).
- Переход $d \to d$ означает повторную передачу всех пакетов из окна, это происходит в том случае, когда заканчивается лимит времени ожидания подтверждения какого-либо пакета из окна.

Процесс *Agent*, описывающий поведение агента данного протокола, имеет вид

$$(P|Timers) \setminus \{start, stop, timeout\}.$$

Нетрудно доказать, что *Agent* наблюдаемо эквивалентен процессу



(3)

Поведение всего протокола описывается процессом

$$\begin{aligned} &Protocol \stackrel{\text{def}}{=} (Agent_1 \,|\, Agent_2), \\ &Init(Protocol) = \{w_i = b_i = s_i = r_i = 0, I_i = \emptyset \ (i = 1, 2), \\ &\qquad\qquad c_{12} = c_{21} = \varepsilon\}, \end{aligned} \tag{4}$$

где процессы $Agent_1$ и $Agent_2$ определяются аналогично соответствующим процессам из предыдущего пункта.

2 Протокол скользящего окна с заданным повтором

В этом пункте рассматривается другой протокол двунаправленной передачи, в котором отправитель тоже может послать в канал несколько

кадров подряд, не дожидаясь подтверждений. Этот протокол называется **протоколом скользящего окна (ПСО) с заданным повтором**. Так же, как и в предыдущем протоколе, каждый из участников этого протокола совмещает функции отправителя и получателя, и номера пересылаемых кадров являются элементами $\mathbf{Z}_n$, где n – четное число. Однако, в отличие от предыдущего протокола, у агента этого протокола имеется не одно, а два окна, называемых **посылающим** и **принимающим** окнами. В посылающем окне содержатся отправленные, но еще не подтвержденные пакеты. Принимающее окно предназначено для размещения пакетов, полученных от другого агента, которые пока не м.б. переданы потребителю данных, т.к. ещё не пришли некоторые пакеты с меньшими номерами. Главное отличие от предыдущего протокола заключается в том, что в случае обнаружения искажения или потери ожидаемого кадра получатель посылает запрос отправителю на повторную передачу этого кадра, данный запрос обозначается **NAK** (**Negative Acknowledgement**), в то время как в предыдущем протоколе в данном случае происходит повторная посылка всех кадров из окна отправителя.

Среди переменных агента протокола имеются массивы x, $timers$, y, $arrived$ размера $m = n/2$, индексированные элементами $\mathbf{Z}_m$. Данные массивы имеют следующий смысл.

- В x размещается посылающее окно. Если агент посылает в канал кадр номер s, то соответствующий этому кадру пакет записывается в компоненту массива x с номером $s\%m$, где $\%m$ обозначает операцию взятия остатка по модулю m. Как и в предыдущем протоколе, мы рассматриваем совокупность $x_0, \ldots, x_{m-1}$ компонентов массива x как «свёрнутую в кольцо», и посылающее окно является связным подмножеством этого кольца. С этим окном связаны две переменные агента протокола: b (нижняя граница окна) и s (верхняя граница окна), их значениями являются элементы $\mathbf{Z}_n$. Если в текущий момент $b = s$, то посылающее окно является пустым, иначе оно имеет вид $\{x_{b\%m}, x_{(b+1)\%m}, \ldots, x_{(s-1)\%m}\}$. Агент протокола содержит переменную w, значение которой равно размеру окна (т.е. числу компонентов массива x, входящих в окно), $b + w = s$.

- В $timers$ размещаются булевы значения, соответствующие таймерам: если для некоторого $i \in \mathbf{Z}_m$ $timers_i = 1$, то таймер номер i включён, иначе он выключен. Каждый из этих таймеров предназначен для оповещения агента о том, что время ожидания подтверждения пакета с соответствующим номером из посылающего окна закончилось и необходимо послать кадр с этим пакетом ещё раз.

- В y размещается принимающее окно, его размер всегда равен m. В это окно помещаются пакеты, извлеченные из принятых кадров. Пакет, извлеченный из кадра номер i, размещается в компоненте $y_{i\%m}$. С этим окном связаны переменные r (нижняя граница окна) и u (верхняя граница окна), их значения – элементы $\mathbf{Z}_n$, $u = r + m$.

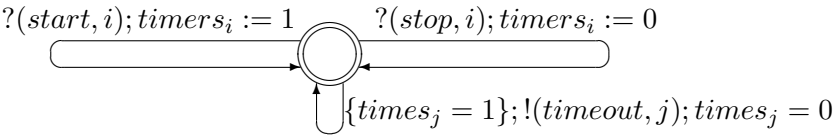
- В $arrived$ размещаются булевы значения, связанные с соответствующими компонентами принимающего окна и имеющие следующий смысл: $\forall\, i = 0, \ldots, m-1$ $arrived_i = 1$, если в i-й компоненте принимающего окна содержится пакет, который агент пока ещё не передал своему потребителю данных.

  Если в принимающем окне присутствует пакет с номером r, то агент может передать этот пакет своему потребителю данных, после чего он увеличивает r и u на 1.

Каждый пересылаемый кадр f имеет вид $[k, x, s, r]$, где k – тип кадра, он может иметь следующие значения: $data$ (информационный кадр), ack (кадр, содержащий лишь подтверждение), nak (кадр с запросом на повторную передачу), x – пакет, $s, r \in \mathbf{Z}_n$, s – номер, сопоставленный пакету x и кадру f, r – номер последнего полученного неискажённого кадра. У кадров типа ack и nak компоненты x и s фиктивны и обозначаются символом $*$. Для извлечения пакетов и номеров из кадров используются функции $kind$, $info$, seq и ack, где

$$kind([k, x, s, r]) = k, \quad info([k, x, s, r]) = x,$$
$$seq([k, x, s, r]) = s, \quad ack([k, x, s, r]) = r.$$

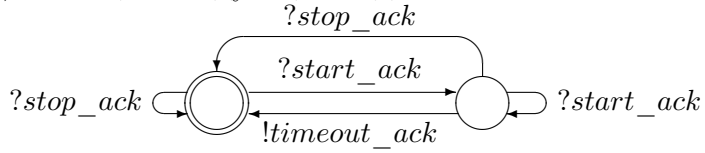
Поведение таймеров представляется процессом $Timers$, который использует переменную i со значениями в $\mathbf{Z}_m$ и вышеупомянутый массив $timers$, представляющий множество номеров включенных таймеров в текущий момент. Процесс $Timers$ имеет вид



(5)

Средняя стрелка в (5) обозначает m переходов, соответствующих значениям $j = 0, \ldots, m-1$. $Init(Timers) = \{timers = (0, \ldots, 0)\}$.

Наряду с вышеупомянутыми таймерами агент протокола использует дополнительный таймер, поведение которого описывается процессом Add_Timer, имеющим следующий вид:

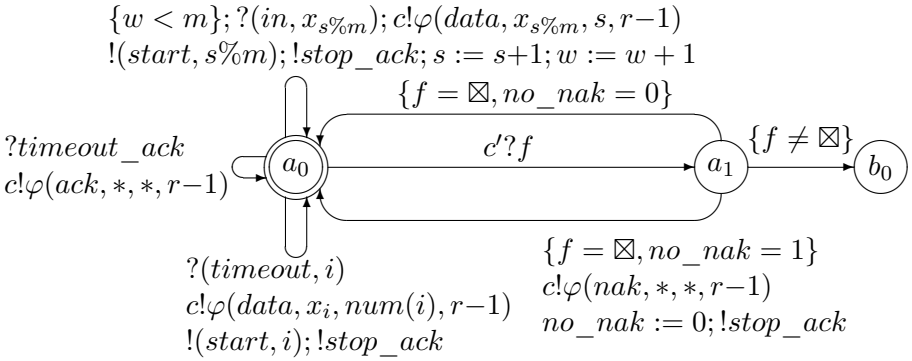


Дополнительный таймер используется в том случае, когда агенту необходимо послать подтверждение a: он включает дополнительный таймер ($!start\_ack$), и если до сигнала $timeout\_ack$ агент получит новый пакет x от своего источника данных, он посылает кадр $[data, x, s, a]$ (т.е. посылает это подтверждение a вместе с пакетом x) и выключает дополнительный таймер, иначе он посылает кадр $[ack, *, *, a]$.

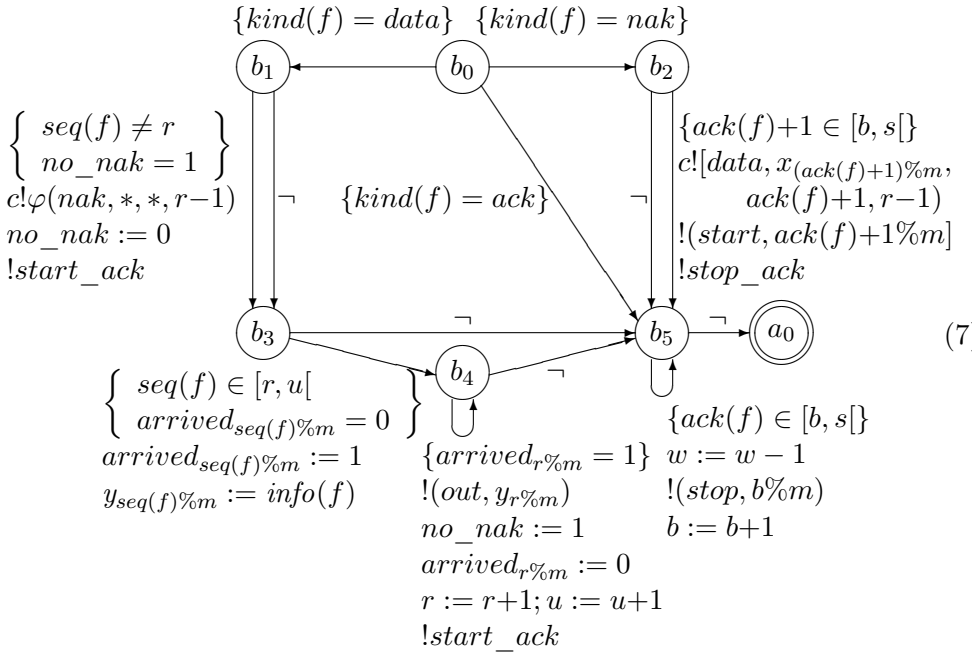
Поведение каждого из участников этого протокола описывается нижеследующим процессом P, который мы для удобства восприятия представляем двумя графами. В этих графах используются следующие обозначения и соглашения:

- вершины с метками a_0 и b_0 в первом графе совпадают с вершинами с соответствующими метками во втором графе,
- если какое-либо ребро $v \to v'$ помечено последовательностью АО A, в которой имеется более одного АО приема или посылки, то данное ребро является сокращенным обозначением пути $v \stackrel{A_1}{\to} \ldots \stackrel{A_n}{\to} v'$, где A является конкатенацией $A_1 \ldots A_n$,
- $num(i)$ обозначает число $j \in [b, s[$, такое, что $j\%m = i$,
- символ $\neg$ на ребре $edge$ какого-либо из данных графов имеет следующий смысл: пусть v – начало данного ребра, тогда существует другое ребро с началом в v, первый АО в его метке имеет вид $\{\varphi\}$, и метка ребра $edge$ имеет вид $\{\neg\varphi\}$,
- P имеет булеву переменную no_nak, предназначенную для предотвращения нескольких запросов на повторную передачу ожидаемого кадра (его номер равен значению переменной r): $no\_nak = 1$, если запрос на повторную передачу кадра номер r ещё не был послан, при получении искаженного кадра $\boxtimes$ или кадра с номером $\neq r$ агент посылает запрос NAK на повторную передачу кадра номер r.

Графы, представляющие процесс P, имеют следующий вид (во втором графе представлены различные варианты обработки полученного неискаженного кадра):



(6)



(7)

Поясним смысл некоторых переходов процесса P.

- Верхний переход $a_0 \to a_0$: получение нового пакета от источника данных, который записывается в $x_{s\%m}$, s считается номером этого пакета и кадра, содержащего этот пакет.
- Переход $b_3 \to b_4$: запись принятого пакета в принимающее окно.
- Переход $b_4 \to b_4$: передача потребителю данных тех принятых пакетов, которые можно передать, и их удаление из принимающего окна.
- Переход $b_5 \to b_5$: удаление подтвержденных пакетов из посылающего окна. Если агент получает кадр, четвертая компонента a которого (номер подтверждения) такова, что $a \in [b, s[$, то все пакеты

с номерами из $[b, a[$ тоже считаются подтверждёнными и удаляются из посылающего окна (даже если их подтверждения не дошли).

Процесс $Agent$, описывающий поведение агента, имеет вид

$$(P|Timers|Add\_Timer) \setminus \left\{ \begin{array}{l} start, stop, timeout \\ start\_ack, stop\_ack, timeout\_ack \end{array} \right\}$$

Поведение всего протокола описывается процессом

$$\begin{array}{l} Protocol \stackrel{\text{def}}{=} (Agent_1 \,|\, Agent_2), \\ Init(Protocol) = \{w_i = b_i = s_i = r_i = 0, u_i = m = n/2, no\_nak_i = 1, \\ \qquad\qquad arrived_i = (0 \ldots 0), timers_i = (0, \ldots, 0)\ (i = 1, 2), \\ \qquad\qquad c_{12} = c_{21} = \varepsilon\}, \end{array} \tag{8}$$

где процессы $Agent_1$ и $Agent_2$ определяются аналогично соответствующим процессам из предыдущего пункта.

Межфакультетский курс «Интеллектуальные методы анализа протоколов безопасности» Лекция 8 Протоколы безопасности на вероятностном уровне

А.М.Миронов

1 Введение

Наряду с изложенными выше моделями протоколов безопасности используются и другие модели, среди которых наибольшую популярность получили **вероятностные системы переходов (ВСП)**.

ВСП являются эффективным средством моделирования таких ПБ, в которых присутствуют ненадежные компоненты, выходящие из строя с некоторой вероятностью, или возникают различные случайные события (например, потеря сообщений в каналах связи), или в алгоритмах явно присутствует рандомизация.

В этой главе мы будем изучать логический язык описания свойств ВСП и методы их анализа, называемые **вероятностным model checking** (**probabilistic model checking, PMC**). В настоящее время PMC является одним из наиболее широко используемых методов моделирования и верификации протоколов безопасности.

Понятие ВСП является обобщением понятия цепи Маркова, которое имеет широкие применения в естественных и гуманитарных науках. Понятие ВСП можно рассматривать также как частный случай понятия вероятностного автомата.

Одной из главных причин актуальности ВСП в задачах верификации ПБ является существенно меньшая (по сравнению с моделями в виде СП) сложность вероятностных моделей анализируемых ПБ. Однако

построение и анализ моделей анализируемых ПБ в виде ВСП является нетривиальной задачей по следующим причинам:

- для получения численных значений вероятностей переходов в модели анализируемой ПБ в виде ВСП необходимо проведение достаточно трудоемких экспериментов с исходной ПБ;
- свойства модели в виде ВСП могут отличаться от свойств исходной ПБ, это приводит к проблеме оценки меры соответствия свойств исходной ПБ и свойств ее модели в виде ВСП.

Первые алгоритмы PMC были предложены в 1980-е годы. Данные алгоритмы были предназначены для верификации качественных вероятностных свойств (то есть таких, которые выполняются с вероятностью 1 или 0). Затем эти алгоритмы были обобщены на случай верификации количественных вероятностных свойств. Первые промышленные системы PMC были разработаны в 2000-х годах. Эти системы успешно применяются во многих областях, в том числе анализе распределенных алгоритмов, телекоммуникационных протоколов, компьютерной безопасности, криптографических протоколах, моделировании биологических процессов. С использованием этих систем PMC были обнаружены уязвимости и аномальные поведения анализируемых систем. При помощи систем PMC могут быть вычислены такие характеристики программных систем, как, например, вероятность вторжения злоумышленника в компьютерную сеть, мат. ожидание времени отклика веб-сервиса, и другие количественные и качественные характеристики.

Наиболее популярной практической системой PMC является система PRISM, разработанная на факультете компьютерных наук Оксфордского университета (Великобритания) в группе Quantitative Analysis and Verification под руководством Марты Квятковской.

2 Вероятностные системы переходов

2.1 Понятие вероятностной системы переходов

Вероятностная система переходов (ВСП) (называемая также **Discrete Time Markov Chain**) – это четверка Σ вида

$$\Sigma = (S, P, L, s^0), \tag{1}$$

компоненты которой имеют следующий смысл:

- S – множество **состояний** ВСП Σ,

- P – функция вида $P : S \times S \to [0, 1]$, называемая **функцией переходов** ВСП Σ и удовлетворяющая условию:

$$\forall\, s \in S \quad \sum_{s' \in S} P(s, s') = 1,$$

- L – функция вида $L : S \times AP \to \{0, 1\}$, называемая **оценкой**,
- $s^0 \in S$ – **начальное состояние** ВСП Σ.

$\forall\, (s_1, s_2) \in S \times S$ число $P(s_1, s_2)$ понимается как вероятность того, что если в текущий момент времени Σ находится в состоянии s_1, то через один такт времени Σ будет находиться в состоянии s_2.

Оценка L имеет следующий смысл: $\forall\, s \in S$, $\forall\, p \in AP$ утверждение p считается **истинным** в s, если $L(s, p) = 1$, и **ложным** иначе.

ВСП $\Sigma = (S, P, L, s^0)$ удобно рассматривать как граф (обозначаемый тем же символом Σ), в котором

- вершинами являются состояния из S и
- для каждой пары $(s_1, s_2) \in S \times S$ такой, что $P(s_1, s_2) > 0$, имеется ребро из s_1 в s_2 с меткой $P(s_1, s_2)$.

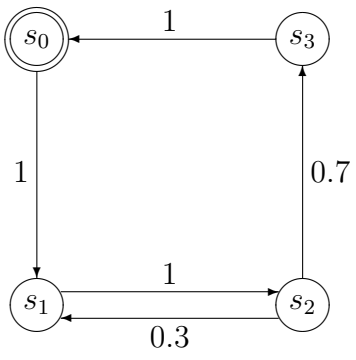
Рёбра данного графа будем называть **переходами** ВСП Σ.

2.2 Примеры вероятностных систем переходов

1. Упрощённая модель протокола передачи сообщений через ненадёжный канал, в котором сообщения могут пропадать. Протокол представляет собой систему, состоящую из

   - двух агентов – отправителя и получателя, а также
   - канала, в который помещаются сообщения, пересылаемые от одного агента другому.

   Граф, представляющий эту ВСП, имеет следующий вид:



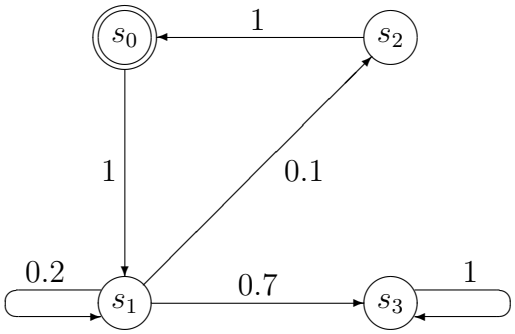
(2)

Переходы этой ВСП имеют следующий смысл.

- Переход $s_0 \xrightarrow{1} s_1$ заключается в получении отправителем от внешнего источника сообщения, которое должно быть передано через канал получателю.
- Переход $s_1 \xrightarrow{1} s_2$ заключается в помещении сообщения в канал отправителем.
- Переход $s_2 \xrightarrow{0.3} s_1$ заключается в потере сообщения в канале.
- Переход $s_2 \xrightarrow{0.7} s_3$ заключается в передаче сообщения из канала получателю.
- Переход $s_3 \xrightarrow{1} s_0$ заключается в получении сообщения получателем и посылке им подтверждения отправителю.

2. Упрощённая модель агента, работа которого представляет собой последовательность сеансов. В каждом из этих сеансов агент пытается выполнить некоторое действие. Если действие было выполнено, то он переходит к следующему сеансу, а если не было выполнено, то он завершает свою работу.

   Граф, представляющий эту ВСП, имеет следующий вид:



   Состояния этой ВСП имеют следующий смысл:

   - В состоянии s_0 агент начинает очередной сеанс.
   - В состоянии s_1 агент предпринимает попытку выполнения действия (переход $s_1 \xrightarrow{0.2} s_1$ означает, что попытка выполнения действия пока не осуществима).

- В состояние s_2 агент попадает тогда, когда действие было выполнено успешно.
- В состоянии s_3 агент оказывается тогда, когда его попытка выполнить действие закончилась неудачей.

Межфакультетский курс «Интеллектуальные методы анализа протоколов безопасности» Лекция 9 Алгоритмы вычисления значений формул вероятностной логики

А.М.Миронов

Свойства протоколов безопасности в вероятностной модели описываются формулами вероятностной темпоральной логики PCTL. Когда известны вероятностная модель протокола безопасности и формула, описывающая свойство этого протокола, то задача определения соответствия протокола этому свойству сводится к вычислению значения формулы на ВСП, моделирующей этот протокол. В этой лекции мы вводим язык вероятностных темпоральных формул и описываем алгоритм вычисления значений этих формул на ВСП.

1 Темпоральная логика PCTL

1.1 Свойства вероятностных систем переходов

Одним из логических языков, предназначенных для формального описания свойств поведения ВСП, является темпоральная логика **PCTL** (**Probabilistic Computation Tree Logic**). Логика PCTL была введена Х. Ханссоном (H. Hansson) и Б. Джонссоном (B. Jonsson).

Формулы логики PCTL могут отражать различные вероятностные аспекты поведения систем, к числу которых относятся, например:

- частота выполнения тех или иных действий или переходов,
- вероятность отказа компонентов систем,

- вероятностный характер взаимодействия системы с её окружением, например частота поступления входных запросов или сообщений, частота получения искажённых сообщений (для протоколов передачи сообщений в компьютерных сетях) и т.п.

Примеры свойств, которые можно описать формулами PCTL:

- вероятность доставки сообщения в заданном временном интервале $[t_1, t_2]$ не меньше 0.999,
- в результате работы алгоритма избрания процесса-лидера избрание такого процесса завершится с вероятностью 1,
- вероятность успешной атаки на криптографический протокол не превышает 0.0001,
- вероятность отклика веб-сервиса в течение 5ms – не менее 0.8,
- ожидаемое время в худшем случае для доставки пакета данных при помощи протокола беспроводной связи – 1 с,
- максимальное ожидаемое энергопотребление за 24 часа для устройства с электропитанием от батареи – не более 190J,
- вероятность того, что система после любого отказа восстановится за $\leq n$ шагов, не менее $1 - \varepsilon$,
- вероятность того что сигнал ready будет получен в течение следующих n единиц времени, больше половины.

1.2 Формулы логики PCTL

Совокупность **формул** логики PCTL, называемых также **PCTL-формулами**, обозначается записью Fm_{PCTL}. Формулы логики PCTL делятся на два класса: Fm^{state}_{PCTL} (**state-формулы**) и Fm^{path}_{PCTL} (**path-формулы**), определяемые следующим образом:

- $AP \subseteq Fm^{state}_{PCTL}$,
- Fm^{state}_{PCTL} замкнут относительно булевых комбинаций,
- $\forall\, A \in Fm_{PCTL}$ запись $\mathbf{X}A$ является path-формулой,
- $\forall\, A, B \in Fm^{state}_{PCTL},\ \forall\, n \geq 0$ записи $\mathbf{U}^{\leq n}(A, B)$ и $\mathbf{U}(A, B)$ являются path-формулами,

- $\forall A \in Fm^{path}_{PCTL}$, $\forall a \in [0,1]$ записи $[\![A]\!]_{\geq a}$, $[\![A]\!]_{>a}$, $[\![A]\!]_{\leq a}$ $[\![A]\!]_{<a}$, являются state-формулами.

$\forall A \in Fm^{state}_{PCTL}$, $\forall n \geq 0$ записи $\mathbf{F}^{\leq n}A$ и $\mathbf{F}A$ обозначают формулы $\mathbf{U}^{\leq n}(\top, A)$ и $\mathbf{U}(\top, A)$ соответственно, и записи $\mathbf{G}^{\leq n}A$ и $\mathbf{G}A$ обозначают формулы $\overline{\mathbf{F}^{\leq n}\overline{A}}$ и $\overline{\mathbf{F}\ \overline{A}}$ соответственно.

1.3 Значения формул логики PCTL в состояниях вероятностных систем переходов

Пусть задана ВСП $\Sigma = (S, P, L, s^0)$.

$\forall s \in S$, $\forall A \in Fm_{PCTL}$ ниже определяется **значение** формулы A в состоянии s, которое обозначается записью A^s, и

- если $A \in Fm^{state}_{PCTL}$, то $A^s \in \{0,1\}$,
- если $A \in Fm^{path}_{PCTL}$, то $A^s \in [0,1]$, A^s интерпретируется как вероятность того, что формула A истинна в состоянии s.

В излагаемом ниже определении значений вида A^s будем использовать следующие обозначения. Пусть S имеет вид $\{s_1, \ldots, s_n\}$.

- $\forall A \in Fm_{PCTL}$ будем обозначать записью A^S столбец $\begin{pmatrix} A^{s_1} \\ \ldots \\ A^{s_n} \end{pmatrix}$.
- Для любых столбцов $U = \begin{pmatrix} u_1 \\ \ldots \\ u_n \end{pmatrix}$, $V = \begin{pmatrix} v_1 \\ \ldots \\ v_n \end{pmatrix}$ из $[0,1]^n$ записи $\max(U,V)$ и UV обозначают столбцы $\begin{pmatrix} \max(u_1, v_1) \\ \ldots \\ \max(u_n, v_n) \end{pmatrix}$ и $\begin{pmatrix} u_1 v_1 \\ \ldots \\ u_n v_n \end{pmatrix}$ соответственно.
- Символ P обозначает матрицу порядка $n \times n$, в которой элемент в i-й строке и j-м столбце равен $P(s_i, s_j)$.
- Для любого столбца $U \in [0,1]^n$ запись $[P^*U]$ обозначает столбец из $\{0,1\}^n$, получаемый заменой всех ненулевых компонентов матрицы P и столбца U на 1 и вычислением $(\sum_{i\geq 0} P^i)U$, где сложение понимается как дизъюнкция (т.е. $1 + 1 = 1$ и сумма $\sum_{i\geq 0} P^i$ конечна).

Значения формул логики PCTL в состояниях ВСП определяются индукцией по структуре формул в соответствии с излагаемыми ниже правилами. В одних из этих правил мы определяем значение A^s, в других определяем столбец A^S целиком.

- $\forall\, s \in S,\, \forall\, p \in AP \quad p^s = L(s,p)$.
- $\forall\, s \in S \;\; \top^s = 1,\, \bot^s = 0$.
- Значения булевых комбинаций определяются стандартным образом: $\forall\, s \in S \;\; (\bar{A})^s = \overline{A^s}, (A \wedge B)^s = A^s \wedge B^s$ и т.д.
- $(\mathbf{X}A)^S = PA^S$ (произведение матрицы P на столбец A^S).
- $(\mathbf{U}^{\leq 0}(A,B))^S = B^S$,
  $\forall\, n > 0 \quad (\mathbf{U}^{\leq n}(A,B))^S = \max\Big(B^S,\; A^S(\mathbf{X}\mathbf{U}^{\leq n-1}(A,B))^S\Big)$.
- $(\mathbf{U}(A,B))^S$ определяется системой линейных уравнений

$$(\mathbf{U}(A,B))^S = \max\Big(B^S,\; [P^*B^S]A^S(P(\mathbf{U}(A,B))^S)\Big).$$

- $\forall\, s \in S \;\; (\llbracket A \rrbracket_{\geq a})^s = \llbracket A^s \geq a \rrbracket$, т.е. $(\llbracket A \rrbracket_{\geq a})^s = \begin{cases} 1, \text{ если } A^s \geq a, \\ 0 \text{ иначе,} \end{cases}$
- аналогичное определение для формул вида $\llbracket A \rrbracket_{>a}$, $\llbracket A \rrbracket_{\leq a}$, $\llbracket A \rrbracket_{<a}$.

1.4 Интерпретация значений формул логики PCTL

Пусть задана ВСП Σ. $\forall\, s \in S_\Sigma,\; \forall\, A \in Fm_{PCTL}$ значение A^s интерпретируется следующим образом:

- если $A \in Fm^{state}_{PCTL}$, то формула A истинна в s при $A^s = 1$ и ложна в s при $A^s = 0$,
- если $A = \mathbf{X}B$, то A^s интерпретируется как математическое ожидание значения формулы B в том состоянии, в которое будет совершён переход из состояния s за один такт времени,
- если $A = \mathbf{U}^{\leq n}(B,C)$, то A^s интерпретируется как вероятность того, что для произвольного пути π в графе Σ, выходящего из s, существует состояние s' на этом пути, такое, что длина отрезка от s до s' пути π не превосходит n, и
  - в каждом состоянии этого отрезка, кроме м.б. s', истинна B,

– в состоянии s' истинна C,

- если $A = \mathbf{U}(B, C)$, то A^s интерпретируется так же, как значение предыдущей формулы, без упоминания того, что длина пути из s в s' не превосходит n.

На основе этой интерпретации можно описывать свойства ВСП формулами логики PCTL. Эти свойства могут выражать динамические аспекты поведения ВСП, т.е. описывать зависимость значения какого-либо утверждения в некотором состоянии рассматриваемой ВСП от значений других утверждений в других состояниях этой ВСП. Например, свойство протокола из пункта **??**, представленного ВСП (**??**):

> каждое сообщение, полученное отправителем от внешнего источника, будет доставлено получателю за не более чем 5 шагов с вероятностью ≥ 0.9,

выражается формулой $\mathbf{G}\Big(p_0 \to [\![\mathbf{F}^{\leq 5} p_3]\!]_{\geq 0.9}\Big)$, где $s_i(p_j) = [\![i = j]\!]$.

Межфакультетский курс «Интеллектуальные методы анализа протоколов безопасности» Лекция 10 Протокол аутентификации Yahalom

А.М.Миронов

В этой лекции мы излагаем модель распределенных процессов, которую применяем для верификации протокола аутентификации Yahalom.

1 Термы и связанные с ними понятия

1.1 Типы, переменные, константы и функциональные символы

Будем предполагать, что заданы следующие множества.

- Множество $Types$, его элементы называются **типами**. Будем понимать типы так же, как понимаются типы данных в языках программирования. Каждому типу τ из $Types$ сопоставлено множество D_τ **значений** типа τ.
- Множества Var и Con, их элементы называются **переменными** и **константами** соответственно. Каждой переменной $x \in Var$ и константе $c \in Con$ сопоставлен тип $\tau(x)$ и $\tau(c) \in Types$ соответственно. Каждая переменная x может принимать **значения** в домене $D_{\tau(x)}$, т.е. в различные моменты времени переменная x может быть связана с различными элементами домена $D_{\tau(x)}$.
- Множество Fun, его элементы называются **функциональными символами (ФС)**. Каждому $f \in Fun$ сопоставлены

– **функциональный тип (ФТ)** $\tau(f)$, который представляет собой запись вида

$$(\tau_1, \ldots, \tau_n) \to \tau, \tag{1}$$

где $\tau_1, \ldots, \tau_n, \tau \in Types$, и

– частичная функция вида $D_{\tau_1} \times \ldots \times D_{\tau_n} \to D_\tau$, где $\tau(f)$ имеет вид (1), данная функция обозначается тем же символом f и называется **интерпретацией** ФС f. (Напомним, что функция $f : D \to D'$ называется **частичной**, если $\forall\, d \in D$ значение $f(d)$ м.б. не определено.)

Будем считать, что $Types$ содержит следующие типы:

- **A**, значения этого типа называются **агентами**,
- **K**, значения этого типа называются **ключами**, они обозначают криптографические ключи, которые агенты могут использовать для шифрования или расшифрования сообщений,
- **M**, значения этого типа называются **сообщениями**, они обозначают сообщения, которые агенты могут пересылать друг другу во время своей работы,
- **B**, значения этого типа – 0 и 1,
- для каждого типа τ множество $Types$ содержит тип $\mathbf{2}^\tau$, значениями этого типа являются подмножества множества D_τ.

Будем предполагать, что Fun содержит следующие ФС.

- ФС *encrypt* типа $(\mathbf{K}, \mathbf{M}) \to \mathbf{M}$.

  Терм вида $encrypt(k, e)$ обозначает сообщение, получаемое шифрованием сообщения e на ключе k. Будем обозначать такой терм записью $k(e)$ и называть его **шифрованным сообщением (ШС)**.

- ФС *shared_key* типа $(\mathbf{2}^{\mathbf{A}}) \to \mathbf{K}$.

  Терм вида $shared\_key(\{A_1, \ldots, A_n\})$ называется **разделяемым ключом** агентов $A_1, \ldots, A_n$ и будет обозначаться записью $k_{A_1 \ldots A_n}$.

1.2 Термы

Термы строятся из переменных, констант и ФС. Множество всех термов обозначается символом Tm. Каждый терм e имеет тип $\tau(e) \in Types$, 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 $e_1, \ldots, e_n \in Tm$, $f \in Fun$, и $\tau(f)$ имеет вид (1), где $\tau_1 = \tau(e_1)$, $\ldots$, $\tau_n = \tau(e_n)$, то запись $f(e_1, \ldots, e_n)$ – терм типа τ.

Терм $e \in Tm$ называется **подтермом** терма $e' \in Tm$, если либо $e = e'$, либо $e' = f(e_1, \ldots, e_n)$, и $\exists\, i \in \{1, \ldots, n\}$: e – подтерм терма e_i. Запись $e \subseteq e'$, где $e, e' \in Tm$, означает, что e является подтермом терма e'. Запись $e \subset e'$, где $e, e' \in Tm$, означает, что $e \subseteq e'$ и $e \neq e'$. Совокупность всех подтермов терма e обозначается $Sub(e)$.

Индукцией по структуре терма $e \in Tm$ нетрудно доказать, что

$$\begin{array}{l}\text{если } e_1 \text{ и } e_2 \text{ – различные подтермы терма } e,\\ \text{то либо } e_1 \subset e_2, \text{ либо } e_2 \subset e_1,\\ \text{либо } e_1 \text{ и } e_2 \text{ не имеют общих компонентов.}\end{array} \qquad (2)$$

Будем использовать следующие обозначения:

- $\forall\, x \in Var, e \in Tm$ запись $x \in e$ означает, что x входит в e,
- $\forall\, e_1, \ldots, e_n \in Tm$

$$Var(e_1, \ldots, e_n) = \{x \in Var \mid \exists\, i \in \{1, \ldots, n\} : x \in e_i\},$$

- $\forall\, E \subseteq Tm$ множество $Tm(E)$ определяется следующим образом:
  - $E \subseteq Tm(E)$, $Con \subseteq Tm(E)$,
  - для каждого терма вида $f(e_1, \ldots, e_n)$, если $e_1, \ldots, e_n \in Tm(E)$, то $f(e_1, \ldots, e_n) \in Tm(E)$.
- $\forall\, \tau \in Types$, $\forall\, E \subseteq Tm$ $E_\tau = \{e \in E \mid \tau(e) = \tau\}$,
- для каждого списка типов $\tau_1, \ldots, \tau_n$ множество $Types$ содержит тип $(\tau_1, \ldots, \tau_n)$, и $D_{(\tau_1, \ldots, \tau_n)} = D_{\tau_1} \times \ldots \times D_{\tau_n}$, Fun содержит ФС $tuple$ типа $(\tau_1, \ldots, \tau_n) \to (\tau_1, \ldots, \tau_n)$, ему соответствует тождественная функция, терм $tuple(e_1, \ldots, e_n)$ обозначается записью $(e_1, \ldots, e_n)$.

Термы типа **B** называются **формулами**. Множество всех формул обозначается Fm. $\forall X \subseteq Var\ Fm(X) = Tm(X) \cap Fm$. При построении формул могут использоваться обычные булевы ФС ($\neg$, $\wedge$, $\vee$, $\to$ и т.д.), которым соответствуют функции отрицания, конъюнкции, дизъюнкции и т.д. Символы 1 и 0 обозначают тождественно истинную и ложную формулу соответственно. Формулы вида $\wedge(e_1, e_2)$ будем записывать в более привычном виде $e_1 \wedge e_2$. Формулы вида $e_1 \wedge \ldots \wedge e_n$ могут также записываться в виде $\left\{ \begin{array}{c} e_1 \\ \vdots \\ e_n \end{array} \right\}$ или $\{e_1, \ldots, e_n\}$. Возможна конъюнкция произвольного семейства формул $\{e_i \mid i \in I\}$, она обозначается $\bigwedge_{i \in I} e_i$.

Терм $e \in Tm$ **замкнут**, если $Var(e) = \emptyset$. Каждому замкнутому терму e соответствует объект $eval(e)$, называемый **значением** данного терма, и либо является элементом $D_{\tau(e)}$, либо не определён. Если e – константа, то $eval(e)$ – интерпретация этой константы, и если e имеет вид $f(e_1, \ldots, e_n)$, то $eval(e)$ определён, только если определено значение функции f на кортеже $(eval(e_1), \ldots, eval(e_n))$, и в этом случае $eval(e)$ равен этому значению. Для каждого замкнутого терма e будем обозначать объект $eval(e)$ той же записью, что и сам терм (т.е. e).

1.3 Подстановки

Подстановкой называется функция $\theta : Var \to Tm$, такая, что для каждой $x \in Var\ \tau(x) = \tau(\theta(x))$. Будем говорить, что подстановка θ заменяет переменную $x \in Var$ на терм $\theta(x)$.

Будем использовать следующие обозначения:

- множество всех подстановок обозначается символом Θ;
- $\forall\, \theta \in \Theta\ Var(\theta) = \{x \in Var \mid \theta(x) \neq x\}$;
- $\forall\, X \subseteq Var \quad \Theta(X) = \{\theta \in \Theta \mid Var(\theta) \subseteq X\}$;
- подстановка $\theta \in \Theta$ может обозначаться записями

$$x \mapsto \theta(x) \quad \text{или} \quad (\theta(x_1)/x_1, \ldots, \theta(x_n)/x_n), \tag{3}$$

  (вторая запись в (3) используется, если $Var(\theta) = \{x_1, \ldots, x_n\}$);
- $\forall\, \theta \in \Theta,\ \forall\, e \in Tm$ запись e^θ обозначает терм, получаемый из e заменой $\forall\, x \in Var(e)$ каждого вхождения x в e на терм $\theta(x)$;
- $\forall\, \theta \in \Theta,\ \forall\, E \subseteq Tm\ E^\theta = \{e^\theta \mid e \in E\}$,
- $\forall\, \theta, \theta' \in \Theta$ запись $\theta\theta'$ обозначает подстановку $x \mapsto (x^\theta)^{\theta'}$.

Подстановка θ' называется **продолжением** подстановки θ, если верно включение $Var(\theta) \subseteq Var(\theta')$ и $\forall\, x \in Var(\theta)\ \ \theta(x) = \theta'(x)$.

Подстановка θ **замкнута**, если $\forall\, x \in Var(\theta)\ Var(x^\theta) = \emptyset$. Множество всех замкнутых подстановок из $\Theta(X)$ обозначается $X^\bullet$.

Термы e_1 и e_2 считаются равными, если $\forall\, \theta \in Var(e_1, e_2)^\bullet\ e_1^\theta = e_2^\theta$.

2 Последовательные и распределенные процессы

2.1 Действия

Действие – это запись одного из следующих видов:

$$\circ!e, \quad \circ?e, \quad e := e', \quad \text{где } e, e' \in Tm,$$

которые называются **выводом** сообщения e в открытый канал $\circ$, **вводом** сообщения e из открытого канала $\circ$, и **присваиванием**, соответственно. Множество всех действий обозначается Act.

2.2 Последовательные процессы

Последовательным процессом (или просто **процессом**) будем называть граф P со следующими свойствами:

- P имеет выделенные вершины $\odot$ и $\otimes$, называемые **начальной** и **терминальной** вершинами соответственно, из $\otimes$ не выходят рёбра,
- каждому ребру графа P сопоставлена метка $a \in Act$, ребро процесса P представляется записью $v \xrightarrow{a} v'$, где v и v' – начало и конец ребра, a – метка ребра.

Процесс является описанием поведения дискретной динамической системы, работа которой заключается в последовательном выполнении действий, связанных с вводом и выводом сообщений и изменением значений переменных. С каждым процессом P связаны

- агент $Agent(P) \in Var_{\mathbf{A}}$ (от имени которого выполняется процесс),
- множество $Var(P)$ переменных процесса P, являющееся дизъюнктным объединением следующих множеств:
  - множество $Input(P)$ **входных переменных**,

– множество $Unique(P)$ **уникальных переменных** (инициализированных уникальными значениями), и обозначают криптографические ключи, или переменные, называемые **нонсами**,

– множество $Private(P)$ **внутренних переменных**,

– $\{at_P\}$, значения at_P – вершины графа P,

– $\{x_P\}$, значения x_P – подмножества $Var(P)$, их элементы называются **инициализированными переменными**,

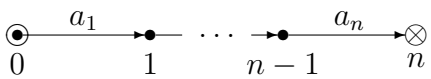
– $\{x_\circ\}$, $\tau(x_\circ) = \mathbf{2^M}$, значения $x_\circ$ интерпретируются как **содержимое открытого канала**.

- **предусловие** $Pre(P) \in Fm$, это конъюнкция формул, содержащая следующие конъюнктивные члены:

$$x_P = Input(P) \cup Unique(P) \cup \{at_P, x_P, x_\circ\},\ at_P = \odot,\ x_\circ = \emptyset$$

(в $Pre(P)$ м.б. и другие конъюнктивные члены).

Процесс называется **линейным**, если он имеет вид



(4)

2.3 Процесс противника

Процесс противника – это процесс $P_\dagger$, обладающий свойствами:

- граф процесса $P_\dagger$ состоит из единственной вершины,
- $\forall\, a \in Act$ граф $P_\dagger$ содержит ребро с меткой a.

Ниже будем предполагать, что $P_\dagger$ – единственный из всех рассматриваемых процессов, граф которого имеет циклы.

2.4 Распределенные процессы

Распределенный процесс (РП) – это семейство $\mathcal{P} = \{P_i \mid i \in I\}$ процессов, таких, что компоненты семейства

$$\{Unique(P_i) \cup Private(P_i) \cup \{at_{P_i}, x_{P_i}\} \mid i \in I\} \tag{5}$$

дизъюнктны (если это условие не выполняется, то соответствующие переменные в процессах P_i переименовываются).

Если $\{\mathcal{P}_i \mid i \in I\}$ – семейство РП, и $\forall\, i \in I \quad \mathcal{P}_i = \{P_{i'} \mid i' \in I_i\}$, то запись $\{\mathcal{P}_i \mid i \in I\}$ обозначает также РП $\{P_{i'} \mid i' \in \bigsqcup_{i\in I} I_i\}$.

Множество $Var(\mathcal{P})$ переменных РП $\mathcal{P}$ определяется как объединение $\bigcup_{i\in I} Var(P_i)$. **Предусловие** $Pre(\mathcal{P})$ РП $\mathcal{P}$ определяется как $\bigwedge_{i\in I} Pre(P_i)$.

Состояние РП $\mathcal{P}$ – это подстановка $\theta \in Var(\mathcal{P})^\bullet$. Состояние θ является частичной функцией, определённой только для тех $x \in Var(\mathcal{P})$, которые входят в $\bigcup_{i\in I} x^\theta_{P_i}$. Состояние θ называется **начальным** (и обозначается θ_0), если $Pre(\mathcal{P})^\theta = 1$.

РП $\mathcal{P} = \{P_i \mid i \in I\}$ обозначается записью P^*, если I – множество натуральных чисел, и все процессы, входящие в $\mathcal{P}$, совпадают с P.

Будем использовать следующее соглашение:

- если в каком-либо рассуждении, связанном с РП вида P^*, некоторый процесс является первым из рассматриваемых процессов, входящих в P^*, то этот процесс и все его переменные обозначаются теми же записями, которые используются в процессе P,
- если кроме этого процесса рассматривается другой процесс, входящий в P^*, то он обозначается $\grave{P}$, и в обозначениях тех его переменных, которые являются дубликатами переменных из $Unique(P) \cup Private(P) \cup \{at_P, x_P\}$, используются обратные штрихи (например, дубликат переменной x в $\grave{P}$ будет обозначаться записью $\grave{x}$), в следующем процессе ($\grave{\grave{P}}$) соответствующие переменные будут обозначаться с двойными обратными штрихами, и т.д.

Для каждого РП $\mathcal{P}$ запись $\mathcal{P}_\dagger$ обозначает РП $\{\mathcal{P}, \{P_\dagger\}\}$.

2.5 Переходы в распределенных процессах

Переход в РП $\mathcal{P} = \{P_i \mid i \in I\}$ – это утверждение, обозначаемое записью $\theta \overset{a_{P_i}}{\to} \theta'$, где $\theta, \theta' \in Var(\mathcal{P})^\bullet$ (θ называется **началом** данного перехода, а θ' – его **концом**) и a – метка некоторого ребра $v \overset{a}{\to} v'$ процесса $P_i \in \mathcal{P}$, причем выполнены следующие условия: $v = at^\theta_{P_i}$ и

- если $a = \circ?e$ или $(e := e')$, и $k(\ldots) \subseteq e$, где $k \in Tm_{\mathbf{K}}$, то

$$\left.\begin{array}{l} \text{либо } k \in x^\theta_P, \\ \text{либо } k = shared\_key(\ldots) \text{ и } Agent(P) \in k \end{array}\right\} \qquad (6)$$

- если $a = \circ!e$, то

$$\left.\begin{array}{l} Var(e) \subseteq x^\theta_P \\ \theta' = (x_\circ \cup \{e^\theta\}/x_\circ, v'/at_{P_i})\theta \end{array}\right\} \qquad (7)$$

- если $a = \circ?e$, то

$$\left.\begin{array}{l} \exists\,\tilde{\theta} \in \Theta(\,Var(e) \setminus x_P^\theta) : e^{\tilde{\theta}\theta} \in x_\circ^\theta \\ \theta' = (x_P^\theta \cup\, Var(e)/x_P, v'/at_{P_i})\tilde{\theta}\theta \end{array}\right\} \tag{8}$$

- если $a = (e := e')$, то

$$\left.\begin{array}{l} Var(e') \subseteq x_P^\theta, \exists\,\tilde{\theta} \in \Theta(\,Var(e) \setminus x_P^\theta) : e^{\tilde{\theta}\theta} = (e')^\theta \\ \theta' = (x_P^\theta \cup\, Var(e)/x_P, v'/at_{P_i})\tilde{\theta}\theta \end{array}\right\}$$

Переход $\theta \stackrel{a_{P_i}}{\to} \theta'$ РП $\mathcal{P}$ интерпретируется как выполнение процессом $P_i \in \mathcal{P}$ действия a, в результате чего $\mathcal{P}$ переходит из состояния θ в состояние θ'. Если в текущий момент $\mathcal{P}$ находится в состоянии θ, и в этот момент некоторый процесс P_i, входящий в $\mathcal{P}$, содержит ребро $at_{P_i}^\theta \stackrel{a}{\to} v'$, причем выполнены условия из вышеприведённого списка, то мы считаем, что РП $\mathcal{P}$, находясь в состоянии θ, может выполнить действие a после чего перейти в состояние θ', при этом

- при выполнении действия $\circ!e$ происходит добавление терма e^θ к содержимому открытого канала $\circ$,

- при выполнении действия $\circ?e$ или $e := e'$ происходит либо чтение некоторого терма из содержимого канала $\circ$, либо присваивание соответственно, путем инициализации неинициализированных в текущий момент переменных из $Var(e)$: терм e рассматривается как шаблон, которому должен соответствовать некоторый терм в $x_\circ^\theta$ или терм $(e')^\theta$ соответственно, и выполняемое действие заключается в определении подходящего продолжения θ' подстановки θ путем определения подходящих значений переменных из $Var(e)$, не вошедших в x_P^θ, с таким расчетом, чтобы значение $e^{\theta'}$ было бы равно некоторому терму из $x_\circ^\theta$ или терму $(e')^\theta$ соответственно.

2.6 Выполнение распределенного процесса

Выполнение РП $\mathcal{P}$ – это последовательность состояний $\pi = (\theta_0, \theta_1, \ldots)$ РП $\mathcal{P}$, такая, что θ_0 – начальное состояние и для каждой пары θ, θ' соседних членов этой последовательности имеется переход $\theta \stackrel{a_P}{\to} \theta'$, где P – какой-либо процесс из $\mathcal{P}$.

Если задано выполнение $\pi = (\theta_0, \theta_1, \ldots)$ и θ, θ' – состояния, входящие в π, то запись $\theta <_\pi \theta'$ означает, что $\theta = \theta_i$ и $\theta' = \theta_j$ для некоторых индексов $i < j$. Запись $\theta \leq_\pi \theta'$ означает, что $\theta <_\pi \theta'$ или $\theta = \theta'$.s Если путь π ясен из контекста, то обозначение π в записях $<_\pi$ и $\leq_\pi$ м.б. опущено.

Состояние θ РП $\mathcal{P}$ называется **достижимым** состоянием, если оно входит в некоторое выполнение РП $\mathcal{P}$. Множество достижимых состояний РП $\mathcal{P}$ обозначается $\Sigma_{\mathcal{P}}$.

Каждое выполнение РП $\mathcal{P}$ можно интерпретировать как путь в графе с множеством вершин $\Sigma_{\mathcal{P}}$, ребра которого соответствуют переходам.

В начальный момент каждого выполнения РП $\mathcal{P} = \{P_i \mid i \in I\}$ переменные из $\bigcup_{i \in I} Unique(P_i)$ инициализированы **уникальными значениями**, т.е. такими значениями, которые никогда не встречались среди всех значений, используемых до начала выполнения $\mathcal{P}$.

3 Свойство защищённости

3.1 Определение свойства защищённости

В рассуждениях, связанных с верификацией РП, будем использовать свойство **защищённости**, обозначаемое записью

$$E \perp P, \text{ где } E \subseteq Tm \text{ и } P \text{ – процесс.} \tag{9}$$

Данное свойство имеет следующий неформальный смысл: переменные из $E \cap Var$ доступны процессу P только в «защищённом» виде, т.е. внутри ШС, которые зашифрованы на ключах, недоступных для P.

Формальное определение данного свойства имеет следующий вид: свойство $E \perp P$ истинно в состоянии $\theta \in \Theta$ (что обозначается $\theta \models E \perp P$), если $\forall\, e \in E^\theta \;\; Agent(P) \notin e$ и

$$\begin{array}{l} \forall\, x \in E^\theta \cap Var,\ \forall\, e \in (x_P^\theta)^\theta \cup x_\circ^\theta \text{ каждое вхождение } x \text{ в } e \\ \text{содержится в подтерме } k(\ldots) \subseteq e, \text{ где } k \in (E^\theta)_{\mathbf{K}}. \end{array} \tag{10}$$

3.2 Теорема о сохранении свойства защищённости при переходах распределённых процессов

В этом параграфе доказывается теорема о сохранении свойства защищённости $E \perp P$ при переходах РП. Данная теорема м.б. интерпретирована как следующее утверждение: если в текущем состоянии θ РП $\mathcal{P}$ верно $E \perp P$, где $P \in \mathcal{P}$, $E \subseteq Tm$, то никакая собственная активность P, начиная с состояния θ, не приведет к тому, что сообщения, представляемые переменными из E, когда-нибудь станут доступны процессу P.

Теорема 1

Пусть заданы РП $\mathcal{P}$, процесс $P \in \mathcal{P}$, и состояния $\theta, \theta' \in \Sigma_{\mathcal{P}}$ таковы, что $\theta \xrightarrow{a_P} \theta'$. Тогда $\forall\, E \subseteq Tm(x_{\mathcal{P}}^{\theta_0})$, где θ_0 – начальное состояние, верна импликация

$$\theta \models E \perp P \Rightarrow \theta' \models E \perp P. \tag{11}$$

Доказательство.

Из $E \subseteq Tm(x_{\mathcal{P}}^{\theta_0})$ следует, что $E = E^{\theta} = E^{\theta'}$.

Пусть верна посылка импликации (11). Докажем, что тогда будет верно её заключение, т.е. $\forall\, x \in E \cap Var$

$$\begin{array}{l}(1)\ \text{каждое вхождение } x \text{ в какой-либо терм } e \in (x_P^{\theta'})^{\theta'} \\ \qquad \text{содержится в подтерме } k(\ldots) \subseteq e,\ \text{где } k \in E_{\mathbf{K}}, \\ (2)\ \text{каждое вхождение } x \text{ в какой-либо терм } e \in x_{\circ}^{\theta'} \\ \qquad \text{содержится в подтерме } k(\ldots) \subseteq e,\ \text{где } k \in E_{\mathbf{K}}.\end{array} \tag{12}$$

1. Если первое утверждение в (12) является неверным, то из (10) следует, что $x_P^{\theta} \subset x_P^{\theta'}$, и имеет место один из двух случаев:

   (a) Первый случай: верно утверждение

   $$\left.\begin{array}{l}\alpha \text{ имеет вид } \circ?e, e^{\theta'} \in x_{\circ}^{\theta}, x_P^{\theta'} = x_P^{\theta} \cup Var(e), \\ \exists\, y \in Var(e) :\ \exists \text{ вхождение } x \text{ в } y^{\theta'}, \text{ которое} \\ \text{не содержится ни в каком подтерме вида} \\ k(\ldots) \subseteq y^{\theta'}, \text{ где } k \in E_{\mathbf{K}}.\end{array}\right\} \tag{13}$$

   Поскольку упомянутое в (13) вхождение x содержится в терме $y^{\theta'} \subseteq e^{\theta'} \in x_{\circ}^{\theta}$, то из (10) следует, что это вхождение x содержится в подтерме $k(\tilde{e}) \subseteq e^{\theta'}$, где $k \in E_{\mathbf{K}}$.

   Из (13) следует, что $k(\tilde{e})$ не м.б. подтермом терма $y^{\theta'}$. Поскольку термы $k(\tilde{e})$ и $y^{\theta'}$ имеют непустое пересечение (оба содержат вышеупомянутое вхождение x), то из (2) следует, что $y^{\theta'} \subset k(\tilde{e})$. Таким образом,

   $$y^{\theta'} \subset k(\tilde{e}) \subseteq e^{\theta'}. \tag{14}$$

   Докажем индукцией по структуре терма e, что из (14) следует

   $$\exists\, z \in Var(e) :\ \ k(\tilde{e}) \subseteq z^{\theta'} \subseteq e^{\theta'}. \tag{15}$$

   Если $e \in Var$, то $z = e$. Если $e \in Con$, то (14) не м.б. верно.

   Если e имеет вид $f(e_1, \ldots, e_n)$, где $f \in Fun$, то

   - если $f = encrypt$, т.е. e имеет вид $k_1(e_1)$, то $k_1 \in Sub(e)_{\mathbf{K}}$, согласно (6), $k_1 \in x_P^{\theta}$, и возможны следующие случаи:

- $k(\tilde{e}) = e^{\theta'} = k_1^{\theta'}(e_1^{\theta'})$, в этом случае $k = k_1^{\theta'} = k_1^{\theta} \in x_P^{\theta}$, однако поскольку $k \in E_{\mathbf{K}}$, то по (10) вхождение k в k должно содержаться в подтерме вида $k'(\ldots) \subseteq k$, что невозможно,
- $k(\tilde{e}) \subseteq k_1^{\theta'}$, данный случай невозможен по определению термов типа $\mathbf{K}$,
- $k(\tilde{e}) \subseteq e_1^{\theta'}$, в данном случае утверждение (15) следует из индуктивного предположения,

• если $f \neq encrypt$, то $\exists\, i \in \{1, \ldots, n\} : k(\tilde{e}) \subseteq e_i^{\theta'}$, и утверждение (15) следует из индуктивного предположения.

Из (14) и (15) следует, что

$$y^{\theta'} \subset k(\tilde{e}) \subseteq z^{\theta'} \subseteq e^{\theta'}. \tag{16}$$

Таким образом, терм e содержит вхождения переменных y и z, обладающие следующим свойством: $y^{\theta'} \subset z^{\theta'}$, откуда для данных вхождений следует включение $y \subset z$, что невозможно.

(b) Второй случай: верно утверждение

$$\left.\begin{array}{l} \alpha \text{ имеет вид } e := e', \text{ где } e' \in Tm(x_P^{\theta}), \\ e^{\theta'} = (e')^{\theta},\ \ x_P^{\theta'} = x_P^{\theta} \cup Var(e), \exists\, y \in Var(e) : \\ \exists \text{ вхождение } x \text{ в } y^{\theta'}, \text{ которое не содержится} \\ \text{ни в каком подтерме вида } k(\ldots) \subseteq y^{\theta'},\ \text{ где } k \in E_{\mathbf{K}}. \end{array}\right\} \tag{17}$$

Поскольку упомянутое в (17) вхождение x содержится в терме $y^{\theta'} \subseteq e^{\theta'} = (e')^{\theta}$, то это вхождение x содержится в подтерме $(z')^{\theta} \subseteq (e')^{\theta}$, где $z' \in Var(e')$.

Так как $e' \in Tm(x_P^{\theta})$, то $Var(e') \subseteq x_P^{\theta}$, следовательно, $z' \in x_P^{\theta}$. Из (10) следует, что упомянутое в (17) вхождение x в $(z')^{\theta}$ содержится в некотором подтерме $k(\tilde{e}) \subseteq (z')^{\theta}$, где $k \in E_{\mathbf{K}}$.

Из (17) следует, что $k(\tilde{e})$ не м.б. подтермом терма $y^{\theta'}$.

Поскольку термы $k(\tilde{e})$ и $y^{\theta'}$ имеют непустое пересечение (оба содержат упомянутое в (17) вхождение x), то из (2) следует, что $y^{\theta'} \subset k(\tilde{e})$.

Из равенства $e' = e^{\theta}$ следует, что $\exists\, z \in Var(e)$: вышеупомянутое вхождение z' в e' входит в подтерм $z^{\theta} \subseteq e^{\theta} = e'$. Следовательно,

$$(z')^{\theta} \subseteq (z^{\theta})^{\theta} = z^{\theta'} \subseteq e^{\theta'}.$$

Таким образом, получаем:

$$y^{\theta'} \subset k(\tilde{e}) \subseteq (z')^{\theta} \subseteq z^{\theta'} \subseteq e^{\theta'}. \tag{18}$$

Как и в предыдущем пункте, на основании (18) заключаем, что терм e содержит вхождения переменных y и z, обладающие следующим свойством: $y^{\theta'} \subset z^{\theta'}$, откуда для данных вхождений следует включение $y \subset z$, что невозможно.

2. Если второе утверждение в (12) неверно, то из (10) следует, что

$$\left.\begin{array}{l} \alpha \text{ имеет вид } \circ!e, \text{ где } e \in Tm(x_P^\theta), \\ \exists \text{ вхождение } x \text{ в } e^\theta, \text{ которое не содержится} \\ \text{ни в каком подтерме вида } k(\ldots) \subseteq e^\theta, \text{ где } k \in E_{\mathbf{K}}. \end{array}\right\} \qquad (19)$$

Так как $e \in Tm(x_P^\theta)$, то упомянутое в (19) вхождение x в e^θ содержится в подтерме вида y^θ терма e^θ, где y – некоторая переменная из x_P^θ. Согласно (10), это вхождение x в y^θ содержится в подтерме вида $k(\ldots) \subseteq y^\theta \subseteq e^\theta$, где $k \in E_{\mathbf{K}}$, что противоречит (19). ■

3.3 Теорема для обоснования свойства соответствия

Теорема, излагаемая в этом параграфе, может использоваться для обоснования **свойства соответствия** протоколов аутентификации, которое имеет следующий смысл: если один из участников протокола аутентификации после выполнения этого протокола пришел к выводу, что другой участник этого протокола является подлинным (т.е. объявленные им свое имя и параметры совпадают с его реальными именем и параметрами), то это действительно верно. Доказываемая ниже теорема применяется для обоснования того, что если в некотором состоянии $\theta \in \Sigma_{\mathcal{P}}$ в канале $\circ$ содержится сообщение, содержащее подтерм $k(e)$, где ключ k недоступен в этом состоянии для некоторого процесса P, входящего в $\mathcal{P}$, то в некотором состоянии $\theta' <_\pi \theta$ другой процесс $P' \neq P$ из $\mathcal{P}$ послал в открытый канал $\circ$ сообщение, содержащее этот подтерм $k(e)$.

Теорема 2.

Пусть заданы РП $\mathcal{P}$, процесс $P \in \mathcal{P}$, начальное состояние θ_0 РП $\mathcal{P}$, подмножество E множества $Tm(x_{\mathcal{P}}^{\theta_0})$ и состояние $\theta \in \Sigma_{\mathcal{P}}$, причем

- $\theta \models E \perp P$, и
- $x_\circ^\theta$ содержит терм с подтермом $k(e)$, где $k \in E_{\mathbf{K}}$.

Тогда для каждого пути π из начального состояния θ_0 РП $\mathcal{P}$ в состояние θ существует процесс $P' \in \mathcal{P} \setminus \{P\}$, такой, что π содержит ребро

$$\dot{\theta} \xrightarrow{(!\dot{e})_{P'}} \theta', \quad \text{где } k(e) \subseteq \dot{e}^{\dot{\theta}}. \qquad (20)$$

Доказательство.

Обозначим записью θ' первое состояние на пути π, такое, что $x_\circ^{\theta'}$ содержит терм e' с подтермом $k(e)$. Т.к. $x_\circ^0 = \emptyset$, то $\theta' \neq \theta_0$.

Пусть ребро на пути π с концом в θ' имеет вид $\dot\theta \stackrel{\alpha_{P'}}{\to} \theta'$. Т.к. $e' \notin x_\circ^{\dot\theta}$, то $\alpha = !\dot e$, где $\dot e^{\dot\theta} = e'$. Если $P' \neq P$, то теорема доказана. Докажем, что случай ($P' = P$) невозможен.

Пусть $P' = P$, т.е. $\dot\theta \stackrel{(!\dot e)_P}{\longrightarrow} \theta'$. Докажем, что $k \in E \cap Var$.

Если это неверно, т.е. k – разделяемый ключ, то, согласно определению выполнения процесса, должно быть верно $Agent(P) \in k \in E_{\mathbf{K}}$, что противоречит предположению $\forall\, \tilde e \in E \;\; Agent(P) \notin \tilde e$.

Из $\theta \models E \perp P$ следует, что $\dot\theta \models E \perp P$, откуда получаем $k \notin x_P^{\dot\theta}$, т.к. если $k \in x_P^{\dot\theta}$, то, согласно (10), вхождение k в $k^{\dot\theta} = k$ должно содержаться в подтерме вида $k'(\ldots)$, что невозможно.

Из соотношения $k \notin x_P^{\dot\theta}$ и из условия $\dot e \in Tm(x_P^{\dot\theta})$, которое верно согласно (7), следует, что $k \notin \dot e$.

Аналогично доказательству импликации (14) $\Rightarrow$ (15) в теореме 1, можно доказать, что из свойств $k(e) \subseteq e' = \dot e^{\dot\theta}$ и $k \notin \dot e$ следует, что

$$\exists\, x \in Var(\dot e) \subseteq x_P^{\dot\theta} : k(e) \subseteq x^{\dot\theta} \in x_P^{\dot\theta}. \tag{21}$$

Обозначим записью θ'' первое состояние на пути π, такое, что $(x_P^{\theta''})^{\theta''}$ содержит терм с подтермом $k(e)$, т.е.

$$\exists\, x \in x_P^{\theta''} : k(e) \subseteq x^{\theta''}. \tag{22}$$

Из (21) следует, что θ'' находится на пути π левее θ'. Нетрудно видеть, что θ'' – не начальное состояние, поэтому на пути π существует ребро вида $\ddot\theta \stackrel{\alpha_{P''}}{\longrightarrow} \theta''$. Из выбора состояния θ'' следует, что $x \notin x_P^{\ddot\theta}$, поэтому $P'' = P$, и возможны два случая:

1. $\alpha = \circ?\ddot e, x \in Var(\ddot e), \ddot e^{\theta''} \in x_\circ^{\ddot\theta}$,

   т.к. $k(e) \subseteq x^{\theta''} \subseteq \ddot e^{\theta''} \in x_\circ^{\ddot\theta}$, то получаем противоречие с выбором θ' как самого первого состояния на пути π, такого, что $x_\circ^{\theta'}$ содержит терм e' с подтермом $k(e)$: состояние $\ddot\theta$ имеет аналогичное свойство, и находится левее θ',

2. $\alpha = (\ddot e := \tilde e), x \in Var(\ddot e), \tilde e \in Tm(x_P^{\ddot\theta}), \ddot e^{\theta''} = \tilde e^{\ddot\theta}$,

   поскольку

   - $k(e) \subseteq x^{\theta''} \subseteq \ddot e^{\theta''} = \tilde e^{\ddot\theta}$ и

- $\tilde{e}$ не содержит k, т.к. выше было установлено, что $k \notin x_P^{\dot{\theta}}$, поэтому, учитывая свойство $\ddot{\theta} \leq \dot{\theta}$, из которого следует включение $x_P^{\ddot{\theta}} \subseteq x_P^{\dot{\theta}}$, получаем: $k \notin x_P^{\ddot{\theta}}$, и, следовательно, терм $\tilde{e} \in Tm(x_P^{\ddot{\theta}})$ тоже не содержит k,

то, аналогично доказательству импликации (14) $\Rightarrow$ (15) в теореме 1, можно доказать, что $\exists\, y \in x_P^{\ddot{\theta}} : k(e) \subseteq y^{\ddot{\theta}}$, что противоречит выбору θ'' как самого первого состояния на пути π со свойством (22): $\ddot{\theta}$ имеет аналогичное свойство, и находится левее θ''. ■

4 Верификация протокола Yahalom

В этом параграфе изложенная выше математическая модель РП и связанные с ней теоремы 1 и 2 применяются для верификации свойств соответствия и секретности протокола аутентификации Yahalom.

4.1 Описание протокола Yahalom

Протокол Yahalom предназначен для аутентификации (т.е. проверки подлинности) агентов, взаимодействующих по открытому каналу $\circ$, и передачи сеансовых ключей между этими агентами. Предполагается что

- заданы множество агентов Ag, а также агент J, называемый **доверенным посредником**, данные агенты могут взаимодействовать друг с другом по открытому каналу $\circ$,
- каждый агент $A \in Ag$ имеет разделяемый ключ k_{AJ} с доверенным посредником J, на котором A и J могут шифровать и дешифровать сообщения, используя симметричную систему шифрования.

В каждом сеансе протокола Yahalom принимают участие следующие агенты: **инициатор** $A \in Ag$, доверенный посредник J, и **респондер** $B \in Ag$. Каждый агент из Ag в одних сеансах м.б. инициатором, а в других – респондером. Выполнение сеанса протокола Yahalom с инициатором A, респондером B и доверенным посредником J представляет собой совокупность четырех пересылок сообщений:

$$\begin{array}{lll} 1. & A \to B & : \quad A, n_A \\ 2. & B \to J & : \quad B, k_{BJ}(A, n_A, n_B) \\ 3. & J \to A & : \quad k_{AJ}(B, k, n_A, n_B), k_{BJ}(A, k) \\ 4. & A \to B & : \quad k_{BJ}(A, k), k(n_B) \end{array} \tag{23}$$

Пересылки в (23) имеют следующий смысл.

1. A посылает B запрос на аутентификацию и генерацию сеансового ключа k, этот запрос состоит из имени агента A и нонса n_A.

2. B посылает J запрос на генерацию сеансового ключа k, в свой запрос он включает своё имя, имя агента A, для связи с которым нужен этот ключ, полученный нонс n_A, и свой нонс n_B.

3. J генерирует сеансовый ключ k и посылает A пару сообщений, из первого сообщения A может извлечь сеансовый ключ k, а второе предназначено для того, чтобы A переслал его B.

4. A посылает B пару сообщений,

   - первое из которых было получено им от J, B может извлечь из этого сообщения сеансовый ключ k, и
   - используя ключ k, B дешифрует второе сообщение.

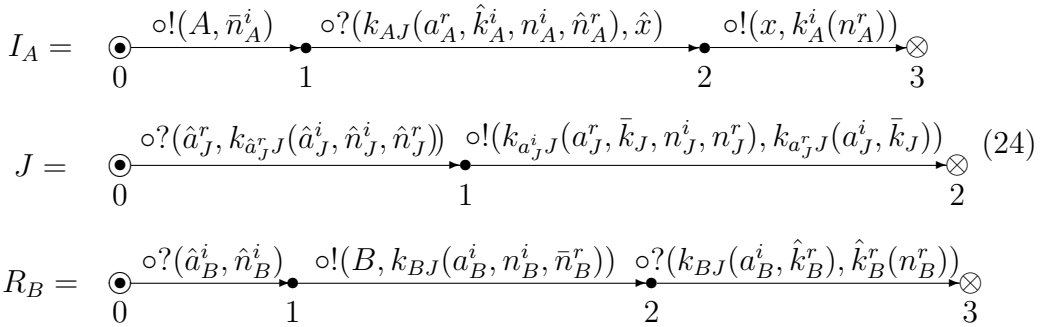
   Если результат дешифрования совпадает с n_B, то это является для B доказательством того, что отправителем этого сообщения был A.

4.2 Формальное описание процессов, входящих в протокол Yahalom

В целях большей наглядности будем использовать следующее соглашение в обозначениях переменных: пусть P – процесс вида (4), и переменная x входит в действие a_i, причём $\forall\, j \in \{1, \ldots, i-1\}$ x не входит в a_j, тогда

- если $x \in Unique(P)$, то будем указывать горизонтальную черту над всеми вхождениями x в a_i (т.е. обозначать их $\bar{x}$) и
- если $x \in Private(P)$, то будем указывать уголок над всеми вхождениями x в a_i (т.е. обозначать их $\hat{x}$).

Описание сеанса протокола Yahalom изображается схемой



(24)

В этой схеме первая и третья диаграммы соответствуют процессам I_A и R_B, описывающим поведение инициатора A и респондера B соответственно, вторая диаграмма соответствует процессу, описывающему поведение посредника J, этот процесс обозначается символом J. Смысл переменных в этих процессах усматривается из сопоставления действий в этих процессах с соответствующими действиями в (23). Верхний индекс i или r при какой-либо переменной означает, что она содержит информацию об инициаторе (i) или респондере (r) данного сеанса. Предполагаем, что $Agent(I_A) = A$, $Agent(R_B) = B$, $Agent(J) = J$.

РП $\mathcal{P}$, соответствующий протоколу Yahalom, имеет вид

$$\mathcal{P} = \{\{I_A^* \mid A \in Ag\}, \{R_B^* \mid B \in Ag\}, J^*\}, \tag{25}$$

т.е. каждый агент может участвовать в неограниченном числе сеансов как в качестве инициатора, так и в качестве респондера.

Ниже будем использовать следующие обозначения:

- если $\mathcal{P}$ – РП, и π – выполнение $\mathcal{P}$, то запись

  $$\pi \ni P^{i,i'} : \theta \xrightarrow{a} \theta'$$

  означает, что π содержит ребро $\theta \xrightarrow{a_P} \theta'$, и $at_P^{\theta} = i$, $at_P^{\theta'} = i'$,

- $\theta \models E \perp e$ означает, что $\forall\, x \in E \cap Var$ каждое вхождение x в e^θ содержится в подтерме $k(\ldots) \subseteq e^\theta$, где $k \in E^\theta_{\mathbf{K}}$.

Нетрудно доказать, что

$$\theta \models E \perp (e, e') \Leftrightarrow \theta \models E \perp e \quad \text{и} \quad \theta \models E \perp e'. \tag{26}$$

4.3 Свойства протокола Yahalom

В этом пункте приводится формальное описание и верификация трех свойств протокола (25): секретность ключей k_J и нонсов n_B^r, аутентификация инициатора перед респондером и аутентификация респондера перед инициатором. В доказательствах теорем 3, 4, 5 при каждом применении теоремы 2 имеется единственный вариант обоснования существования ребра (20) в графе $\Sigma_{\mathcal{P}_\dagger}$, и мы будем сразу будем излагать это обоснование, без упоминания о единственности варианта такого обоснования.

Теорема 3 (секретность ключей k_J и нонсов n_B^r)

РП (25) обладает следующим свойством:

$$\forall\,\theta \in \Sigma_{\mathcal{P}_\dagger} \quad \theta \models E \perp P_\dagger, \text{ где } E = \{k_{BJ}, k_J, n^r_B \mid B \in Ag\}. \tag{27}$$

Доказательство.

Докажем (27) от противного.

Предположим, что $S = \{\theta \in \Sigma_{\mathcal{P}_\dagger} \mid \theta \not\models E \perp P_\dagger\} \neq \emptyset$.

$\forall\,\theta \in S$ обозначим π_θ путь минимальной длины из θ_0 в θ. Пусть θ – состояние из S с наименьшей длиной π_θ. Т.к. $\theta_0 \models E \perp P_\dagger$, то $\theta \neq \theta_0$.

Пусть $\theta' \stackrel{a_P}{\to} \theta$ – ребро из π_θ с концом в θ.

Из определения θ следует, что $\theta' \models E \perp P_\dagger$, $\theta \not\models E \perp P_\dagger$. Если бы было верно $P = P_\dagger$, то из теоремы 1 следует, что $\theta \models E \perp P_\dagger$, т.е. имеем противоречие. Поэтому $P \in \{I_A, R_B, J \mid A, B \in Ag\}$, и

$$a_P = \circ!e,\ x^\theta_\circ = x^{\theta'}_\circ \cup \{e\},\ \exists\, y \in E \cap Var : \neg(y \perp_E e^\theta), \tag{28}$$

где $y \perp_E e^\theta$, означает, что

$$\begin{array}{l}\text{каждое вхождение } y \text{ в } e^\theta \text{ содержится}\\ \text{в подтерме } k(\ldots) \subseteq e^\theta, \text{ где } k \in (E^\theta)_{\mathbf{K}}\end{array} \tag{29}$$

Перебором всех вариантов обоснования существования ребра $\theta' \stackrel{a_P}{\to} \theta$ со свойствами (28) находим единственное возможное обоснование:

$$\pi_\theta \ni I^{2,3}_A : \theta' \stackrel{\circ!e}{\to} \theta, \text{ где } e = (x, k^i_A(n^r_A)). \tag{30}$$

Т.к. $\theta' \models at_{I_A} = 2$, то $\exists\,\theta_1 \leq_{\pi_\theta} \theta'$:

$$\pi_\theta \ni I^{1,2}_A : \theta'_1 \stackrel{\circ?e_1}{\to} \theta_1, \text{ где } e_1 = (k_{AJ}(a^r_A, \hat{k}^i_A, n^i_A, \hat{n}^r_A), \hat{x}). \tag{31}$$

Т.к. $\theta_1 \leq_{\pi_\theta} \theta'$ и $\theta' \models E \perp P_\dagger$, то $\theta_1 \models E \perp P_\dagger$. В частности, $\theta_1 \models E \perp e_1$. По (26), отсюда следует, что $\theta_1 \models E \perp x$.

По теореме 2, из $\theta_1 \models E \perp P_\dagger$, $e^\theta_1 \in x^{\theta_1}_\circ$ и $k_{AJ} \in E$ следует, что $\exists\,\theta_2 \leq_{\pi_\theta} \theta'_1$: π_θ содержит ребро $\theta'_2 \stackrel{(\circ!e_2)_P}{\longrightarrow} \theta_2$, где $P \in \mathcal{P}$ и первая компонента $k_{AJ}(\ldots)$ терма e^θ_1 входит в e^θ_2. Перебором всех вариантов обоснования существования ребра с такими свойствами находим единственное обоснование:

$$\left\{\begin{array}{l}\pi_\theta \ni J^{1,2} : \theta'_2 \stackrel{\circ!e_2}{\to} \theta_2, \text{ где } e_2 = (k_{a^i_J J}(a^r_J, \bar{k}_J, n^i_J, n^r_J), \ldots)\\ k_{(a^i_J)^\theta J}((a^r_J)^\theta, \bar{k}_J, \ldots) = k_{AJ}(a^r_A, (k^i_A)^\theta, \ldots)\end{array}\right. \tag{32}$$

(многоточие в (32) и ниже обозначает компоненту пары, не представляющую интерес для рассмотрения). Из (32) следует, что $(k^i_A)^\theta = \bar{k}_J$, поэтому

$\theta \models E \perp k_A^i(n_A^r)$. Учитывая установленное выше свойство $\theta_1 \models E \perp x$, на основании (26) получаем: $\theta \models E \perp (x, k_A^i(n_A^r))$, т.е. $\theta \models E \perp P$, что противоречит предположению $\theta \not\models E \perp P$. ■

Доказанное свойство $\forall\, \theta \in \Sigma_{\mathcal{P}_\dagger}\ \ \theta \models E \perp P_\dagger$ используется ниже.

Теорема 4 (аутентификация инициатора перед респондером)
Процесс (25) обладает следующим свойством: $\forall\, R_B \in \mathcal{P}$, $\theta \in \Sigma_{\mathcal{P}_\dagger}$, если $\theta \models at_{R_B} = 3$, то $\exists\ I_A \in \mathcal{P}$:

$$\theta \models \left\{ \begin{array}{l} at_{I_A} = 3 \\ a_A^r = B,\ a_B^i = A \\ n_A^i = n_B^i,\ n_A^r = n_B^r \\ k_A^i = k_B^r \end{array} \right\} \tag{33}$$

Доказательство.
Пусть процесс $R_B \in \mathcal{P}$ и состояние $\theta \in \Sigma_{\mathcal{P}_\dagger}$ таковы, что $\theta \models at_{R_B} = 3$. Докажем, что $\exists\, I_A \in \mathcal{P}$: выполнено (33).
Пусть π – путь из θ_0 в θ. Из $\theta \models at_{R_B} = 3$ следует: $\exists\, \theta_1 \leq_\pi \theta$:

$$\pi \ni R_B^{2,3} : \theta_1' \overset{\circ ? e_1}{\to} \theta_1,\ \text{где}\ e_1 = (k_{BJ}(a_B^i, \hat{k}_B^r), \hat{k}_B^r(n_B^r)).$$

По теореме 2, из $\theta_1 \models E \perp P_\dagger$, $e_1^\theta \in x_\circ^{\theta_1}$ и $k_{BJ} \in E$ следует: $\exists\, \theta_2 \leq_\pi \theta_1'$:

$$\left\{ \begin{array}{l} \pi \ni J^{1,2} : \theta_2' \overset{\circ ! e_2}{\to} \theta_2,\ \ \text{где}\ e_2 = (\ldots, k_{a_J^r J}(a_J^i, \bar{k}_J)) \\ k_{(a_J^r)^\theta J}((a_J^i)^\theta, \bar{k}_J) = k_{BJ}((a_B^i)^\theta, (k_B^r)^\theta) \end{array} \right. \tag{34}$$

Из второго равенства в (34) следует, что

$$(a_J^r)^\theta = B, (a_J^i)^\theta = (a_B^i)^\theta, \bar{k}_J = (k_B^r)^\theta. \tag{35}$$

Из $\theta_2' \models at_J = 1$ следует, что $\exists\, \theta_3 \leq_\pi \theta_2'$:

$$\pi \ni J^{0,1} : \theta_3' \overset{\circ ? e_3}{\to} \theta_3,\ \text{где}\ e_3 = (\ldots, k_{\hat{a}_J^r J}(\hat{a}_J^i, \hat{n}_J^i, \hat{n}_J^r)). \tag{36}$$

Из (35) и (36) следует, что $k_{BJ}(*, *, *) \subseteq e_3^\theta \in x_\circ^{\theta_3}$ (где звёздочки обозначают некоторые термы), откуда по теореме 2, с учетом соотношения $\theta_3 \models E \perp P_\dagger$ и $k_{BJ} \in E$ получаем: $\exists\, \theta_4 \leq_\pi \theta_3'$:

$$\left\{ \begin{array}{l} \pi \ni R_{\grave{B}}^{1,2} : \theta_4' \overset{\circ ! e_4}{\to} \theta_4,\ \text{где}\ e_4 = (\ldots, k_{\grave{B}J}(a_{\grave{B}}^i, n_{\grave{B}}^i, \bar{n}_{\grave{B}}^r)) \\ k_{\grave{B}J}((a_{\grave{B}}^i)^\theta, (n_{\grave{B}}^i)^\theta, \bar{n}_{\grave{B}}^r) = k_{BJ}((a_B^i)^\theta, (n_J^i)^\theta, (n_J^r)^\theta) \end{array} \right. \tag{37}$$

Из второго равенства в (37) следует, что

$$\grave{B} = B, (n_B^i)^\theta = (n_J^i)^\theta, \bar{n}_B^r = (n_J^r)^\theta. \tag{38}$$

По теореме 2, из $\theta_1 \models E \perp P_\dagger$, $(k_B^r(n_B^r))^\theta \subseteq e_1^\theta \in x_\circ^{\theta_1}$, и $(k_B^r)^\theta = \bar{k}_J \in E$ следует, что $\exists\, \theta_5 \leq_\pi \theta_1'$:

$$\left\{ \begin{array}{l} \pi \ni I_A^{2,3} : \theta_5' \stackrel{\circ!e_5}{\rightarrow} \theta_5, \text{ где } e_5 = (\ldots, k_A^i(n_A^r)) \\ (k_A^i(n_A^r))^\theta = \bar{k}_J(n_B^r) \end{array} \right. \tag{39}$$

Из второго равенства в (39) следует, что

$$(k_A^i)^\theta = \bar{k}_J, (n_A^r)^\theta = n_B^r. \tag{40}$$

Из $\theta_5 \models at_{I_A} = 2$ следует, что $\exists\, \theta_6 \leq_\pi \theta_5'$:

$$\pi \ni I_A^{1,2} : \theta_6' \stackrel{\circ?e_6}{\rightarrow} \theta_6, \text{ где } e_6 = (k_{AJ}(a_A^r, \hat{k}_A^i, n_A^i, \hat{n}_A^r), \ldots). \tag{41}$$

Из (40) и (41) следует, что

$$k_{AJ}(a_A^r, (k_A^i)^\theta, n_A^i, (n_A^r)^\theta) = k_{AJ}(a_A^r, \bar{k}_J, n_A^i, n_B^r) \subseteq e_6^\theta \in x_\circ^{\theta_6}. \tag{42}$$

По теореме 2, из $\theta_6 \models E \perp P_\dagger$, $k_{AJ} \in E$, и (42) следует, что $\exists\, \theta_7 \leq_\pi \theta_6'$:

$$\left\{ \begin{array}{l} \pi \ni \grave{J}^{1,2} : \theta_7' \stackrel{\circ!e_7}{\rightarrow} \theta_7, \text{ где } e_7 = (k_{a_{\grave{J}}^i \grave{J}}(a_{\grave{J}}^r, \bar{k}_{\grave{J}}, n_{\grave{J}}^i, n_{\grave{J}}^r), \ldots) \\ k_{(a_{\grave{J}}^i)^\theta J}((a_{\grave{J}}^r)^\theta, \bar{k}_{\grave{J}}, (n_{\grave{J}}^i)^\theta, (n_{\grave{J}}^r)^\theta) = k_{AJ}(a_A^r, \bar{k}_J, n_A^i, n_B^r) \end{array} \right. \tag{43}$$

Из второго равенства в (43) следует, что

$$(a_{\grave{J}}^i)^\theta = A, (a_{\grave{J}}^r)^\theta = a_A^r, \grave{J} = J, (n_J^i)^\theta = n_A^i, (n_J^r)^\theta = \bar{n}_B^r. \tag{44}$$

Свойство (33) следует из (35), (38), (40), (44). ■

Теорема 5 (аутентификация респондера перед инициатором)
Процесс (25) обладает следующим свойством: $\forall\, I_A \in \mathcal{P}$, $\theta \in \Sigma_{\mathcal{P}_\dagger}$, если $\theta \models at_{I_A} = 2$, то $\exists\, R_B \in \mathcal{P}$:

$$\theta \models \left\{ \begin{array}{l} at_{R_B} = 2, \\ a_A^r = B, a_B^i = A, \\ n_A^i = n_B^i, n_A^r = n_B^r \end{array} \right\}. \tag{45}$$

Доказательство.
Пусть процесс $I_A \in \mathcal{P}$ и состояние $\theta \in \Sigma_{\mathcal{P}_\dagger}$ таковы, что $\theta \models at_{I_A} = 2$.

Докажем, что $\exists\, R_B \in \mathcal{P}$: выполнено (45).

Пусть π – путь из θ_0 в θ. Из $\theta \models at_{I_A} = 2$ следует, что $\exists\, \theta_1 \leq_\pi \theta$:

$$\pi \ni I_A^{1,2} : \theta_1' \overset{\circ?e_1}{\to} \theta_1, \text{ где } e_1 = (k_{AJ}(a_A^r, \hat{k}_A^i, n_A^i, \hat{n}_A^r), \ldots). \tag{46}$$

По теореме 2, из $\theta_1 \models E \perp P_\dagger$, $k_{AJ}(*,*,*,*) \subseteq e_1^\theta \in x_\circ^{\theta_1}$ (где звёздочки обозначают некоторые термы) и $k_{AJ} \in E$, следует, что $\exists\, \theta_2 \leq_\pi \theta_1'$:

$$\begin{cases} \pi \ni J^{1,2} : \theta_2' \overset{\circ!e_2}{\to} \theta_2, \text{ где } e_2 = (k_{a_J^i J}(a_J^r, \bar{k}_J, n_J^i, n_J^r), \ldots) \\ k_{(a_J^i)^\theta J}((a_J^r)^\theta, \bar{k}_J, (n_J^i)^\theta, (n_J^r)^\theta) = k_{AJ}(a_A^r, (k_A^i)^\theta, n_A^i, (n_A^r)^\theta) \end{cases} \tag{47}$$

Из второго равенства в (47) следует, что

$$(a_J^i)^\theta = A, (a_J^r)^\theta = a_A^r, \bar{k}_J = (k_A^i)^\theta, (n_J^i)^\theta = n_A^i, (n_J^r)^\theta = (n_A^r)^\theta. \tag{48}$$

Из $\theta_2 \models at_J = 1$ следует, что $\exists\, \theta_3 \leq_\pi \theta_2'$:

$$\pi \ni J^{0,1} : \theta_3' \overset{\circ?e_3}{\to} \theta_3, \text{ где } e_3 = (\ldots, k_{\hat{a}_J^r J}(\hat{a}_J^i, \hat{n}_J^i, \hat{n}_J^r)). \tag{49}$$

Из (48) и (49) следует, что $k_{a_A^r J}(A, n_A^i, (n_A^r)^\theta) \subseteq e_3^\theta \in x_\circ^{\theta_3}$, откуда по теореме 2, учитывая $\theta_3 \models E \perp P_\dagger$, и $k_{a_A^r J} \in E$, получаем: $\exists\, \theta_4 \leq_\pi \theta_3'$:

$$\begin{cases} \pi \ni R_B^{1,2} : \theta_4' \overset{\circ!e_4}{\to} \theta_4, \text{ где } e_4 = (\ldots, k_{BJ}(a_B^i, n_B^i, \bar{n}_B^r)) \\ k_{BJ}((a_B^i)^\theta, (n_B^i)^\theta, \bar{n}_B^r) = k_{a_A^r J}(A, n_A^i, (n_A^r)^\theta). \end{cases} \tag{50}$$

Второе равенство в (50) влечёт равенства, из которых следует (45): $B = a_A^r, (a_B^i)^\theta = A, (n_B^i)^\theta = n_A^i, \bar{n}_B^r = (n_A^r)^\theta$. ■

Межфакультетский курс «Интеллектуальные методы анализа протоколов безопасности» Лекция 11 Задача двух миллионеров, протоколы с нулевым разглашением (изоморфизм графов, доказательство знания гамильтонова цикла в графе), протокол подписания контракта

А.М.Миронов

В данной лекции мы рассматриваем протоколы безопасности, предназначенные для решения некоторых прикладных задач.

Ниже будем использовать следующее обозначение: выражение $[\![\varphi]\!]u : v$, где φ – некоторое логическое условие, и u, v – значения, имеет значение u, если φ верно, и v, если φ неверно.

1 Сравнение двух чисел («задача двух миллионеров»)

Излагаемый ниже протокол безопасности предназначен для решения следующей задачи (называемой «задачей двух миллионеров»): агенты A и B («миллионеры») имеют числа x и y соответственно (предполагается, что $x, y \in \{1, \ldots, 100\}$), они хотят узнать без разглашения чисел x и y, верно ли, что $x \leq y$.

Один из протоколов для решения данной задачи имеет следующий вид:

- $A \to B : z - x$, где $z = B^+(r)$, $r \underset{r}{\in} \mathbf{Z}$,

- B вычисляет $\{z_i := B^-(z - x + i) \mid i = 1, \ldots, 100\}$, и проверяет условие
$$\forall\, i \neq j \quad |z_i - z_j| \geq 2$$
если это условие не выполняется, то B просит A начать выполнение протокола заново (выбрать новое r, и т.д.)

- $B \to A:\ z_1, \ldots, z_y, z_{y+1} + 1, \ldots, z_{100} + 1,$

- $A \to B$: ответ $= (x \leq y)$, если $r = x$-й член этой последовательности.

Одна из уязвимостей данного протокола связана с тем, что в нём нет контроля честности участников.

2 Доказательства с нулевым разглашением

2.1 Понятие доказательства с нулевым разглашением

Иногда какой-либо агент (A) хочет путем диалога с другим агентом (B) убедить его в том, что он знает некоторый секрет s, которым может быть, например,

- криптографический ключ, или

- знание решения вычислительно сложной задачи,

таким образом, чтобы агент B

- убедился в том, что A знает секрет,

- но при этом не смог бы извлечь из сообщений, которые он получает от A в процессе данного диалога, никакой информации об этом секрете.

Протокол взаимодействия между агентами A и B, обладающий указанными выше свойствами, называется **доказательством с нулевым разглашением (Д0Р)** знания секрета s.

Как правило, Д0Р представляет собой распределённый вероятностный алгоритм, и состоит из нескольких раундов обмена сообщениями между A и B, где каждый раунд имеет вид

- $B \to A$: вопрос

- $A \to B$: ответ
- B проверяет правильность полученного ответа,

и если A даёт правильный ответ на каждый вопрос, который задаёт ему B, то B принимает решение, что A знает секрет.

Вопросы, которые B задаёт A, как правило, представляют собой выбранные случайным образом элементы некоторого множества. Задача поиска ответов на вопросы должна обладать следующими свойствами:

- если A действительно знает секрет, то он может за полиномиальное время найти правильный ответ на каждый вопрос, который он получит от B, и
- если A не знает секрета, то вероятность того, что ему удастся найти правильный ответ на произвольный вопрос от B за полиномиальное время, должна быть ограничена некоторым числом q, где $0 \leq q < 1$.

Из второго условия следует, что A сможет найти правильные ответы во всех n раундах, в каждом из которых B генерирует свой вопрос независимо от других раундов, с вероятностью, не превышающей q^n. Путём подходящего выбора числа n раундов можно добиться того, чтобы A, не зная секрета, мог убедить B в том, что он знает секрет (т.е. смог найти правильные ответы на все заданные ему вопросы), со сколь угодно малой вероятностью.

2.2 Д0Р знания изоморфизма графов

Пусть задан граф G, и N – множество его вершин. Для каждой перестановки ξ множества N (т.е. биективного отображения $\xi : N \to N$) запись $\xi(G)$ обозначает граф с множеством вершин N, множество рёбер которого определяется следующим образом: для любых $i, j \in N$

$$\begin{array}{c} G \text{ содержит ребро } i \to j \text{ тогда и только тогда, когда} \\ \xi(G) \text{ содержит ребро } \xi(i) \to \xi(j). \end{array}$$

Будем называть графы G_1 и G_2 **изоморфными**, если $G_1 = \xi(G_2)$ для некоторой перестановки ξ. Запись $G_1 \sim G_2$ обозначает утверждение о том, что G_1 и G_2 изоморфны.

Если агент A знает доказательство того, что $G_1 \sim G_2$ (т.е. A знает перестановку σ, такую, что $G_1 = \sigma(G_2)$), то A может доказать B то, что он знает такую перестановку σ при помощи Д0Р, раунд которого имеет следующий вид:

- $A \to B : \xi(G_1)$, где ξ – случайным образом порождённая перестановка на множестве вершин G_1,
- $B \to A : i \underset{r}{\in} \{1, 2\}$,
- $A \to B$: перестановка, доказывающая $\xi(G_1) \sim G_i$
  т.е. $A \to B : [\![i = 1]\!]\ \xi : \xi \circ \sigma$.

2.3 Д0Р знания существования гамильтонова цикла в графе

Гамильтонов цикл (ГЦ) в графе – это цикл в этом графе (т.е. путь, начало которого совпадает с его концом), который проходит через каждую вершину ровно один раз.

Если A знает некоторый ГЦ в графе G, то A может убедить в этом B при помощи Д0Р, раунд которого имеет следующий вид:

- A случайным образом порождает перестановку ξ на множестве вершин графа G,
- $A \to B : H$, где H – матрица, каждый элемент которой представляет собой результат вероятностного шифрования соответствующего элемента матрицы инцидентности графа $\xi(G)$ (который равен 0 или 1),
- $B \to A : i \underset{r}{\in} \{1, 2\}$,
- $A \to B : [\![i = 1]\!]\ \xi : \sigma$, где σ – список элементов матрицы H, результат расшифровки которых представляет собой ГЦ в графе $\xi(G)$.

2.4 Общая схема Д0Р

Приведённые выше Д0Р можно рассматривать как частные случаи общей схемы, позволяющей доказывать с нулевым разглашением знание решения некоторого класса задач.

Предполагается, что на множестве задач $\mathcal{P}$ из этого класса задана некоторая совокупность Ξ преобразований, обладающих следующими свойствами.

- Каждое преобразование $\xi : \mathcal{P} \to \mathcal{P}$ из множества Ξ сопоставляет каждой задаче $p \in \mathcal{P}$ задачу $\xi(p) \in \mathcal{P}$, изоморфную (в некотором смысле) задаче p.

- Если какой-либо агент знает решение σ задачи $p \in \mathcal{P}$, то $\forall \xi \in \Xi$ данный агент знает решение задачи $\xi(p)$. Будем обозначать это решение записью $\xi(\sigma)$.

Если агент A знает решение σ задачи p, то A может доказать агенту B то, что он знает σ, при помощи Д0Р, раунд которого имеет следующий вид:

- $A \to B : (\xi(p), h(\xi(\sigma)))$, где ξ – выбранное случайным образом преобразование из Ξ, h – ХФ,
- $B \to A : i \underset{r}{\in} \{1, 2\}$,
- $A \to B : [\![i = 1]\!]\, \xi : \xi(\sigma)$.

2.5 Д0Р знания дискретного логарифма

Пусть p – простое число, и $g, h \in \mathbf{Z}_p$. Ниже все вычисления и сравнения производятся по mod p.

Если агент A знает число $x \in \mathbf{Z}^*_{p-1}$, такое, что $g^x = h$, то он может доказать другому агенту B знание такого x при помощи следующего протокола:

- $A \to B : \{h_i := g^{x_i} \mid i = 1, \ldots, n\}$, где $x_1, \ldots, x_n \underset{r}{\in} \mathbf{Z}^*_{p-1}$,
- A и B совместно генерируют случайные биты $e_1, \ldots, e_n$,
- $i_1 :=$ min число из $\{1, \ldots, n\}$ со свойством $e_{i_1} = 1$,
- $A \to B : \{[\![e_i = 0]\!]\, x_i : y_i \mid i = 1, \ldots, n\}$, где

$$\forall\, i = 1, \ldots, n \quad y_i := (x_i - x_{i_1})_{p-1},$$

- $\forall\, i = 1, \ldots, n$ агент B проверяет, что

$$[\![e_i = 0]\!]\, (g^{x_i} = h_i) : (g^{y_i} = h_i h_{i_1}^{-1})$$

- $A \to B : z := (x - x_{i_1})_{p-1}$
- B принимает ответ, если $g^z = h h_{i_1}^{-1}$.

Можно доказать, что данный протокол является Д0Р, и если A не знает x со свойством $g^x = h$, то вероятность того, что B примет ответ, не превосходит 2^{-n}.

3 Протокол подписания контракта

Протокол подписания контракта (ППК) используется в тех случаях, когда участники A и B, которые не доверяют друг другу, хотят совместно подписать некоторое сообщение (называемое **контрактом**). Простое решение этой задачи, заключающееся в том, что

- один из участников подписывает контракт, после чего
- подписанный контракт пересылается другому участнику, который тоже подписывает этот контракт

не устраивает обоих участников, так как они не исключают возможности того, что партнёр может не подписать полученный контракт.

Если для решения данной задачи можно привлечь доверенного посредника J, то соответствующий протокол имеет простой вид:

1. $A \to J : \langle m \rangle_A$ (m – это контракт)
2. $J \to B$: извещение, что он имеет $\langle m \rangle_A$
3. $B \to A : \langle m \rangle_B$
4. $A \to J$: извещение, что он имеет $\langle m \rangle_B$
5. $J \to B : \langle m \rangle_A$