Моделирование и анализ информационных систем. Т. 25, № 6 (2018), с. 589–606
Modeling and Analysis of Information Systems. Vol. 25, No 6 (2018), pp. 589–606

Семантика, спецификация и верификация программ
Program Semantics, Specification and Verification

DOI: 10.18255/1818-1015-2018-6-589-606

УДК 517.9

Даже простые процессы π-исчисления трудны для анализа

Аббас М. М.[1], **Захаров В.А.**[2]

Поступила в редакцию 15 сентября 2018
После доработки 30 октября 2018
Принята к публикации 10 ноября 2018

Аннотация. Математические модели распределенных вычислений, построенные на основе исчисления мобильных процессов (π-исчисления), широко используются для проверки свойств информационной безопасности криптографических протоколов. Поскольку π-исчисление является полной по Тьюрингу моделью вычислений, эта задача в общем случае алгоритмически неразрешима. Поэтому ее исследование проводится лишь для некоторых специальных классов процессов π-исчисления с ограниченными вычислительными возможностями, например, для нерекурсивных процессов, в которых все вычисления имеют ограниченную длину, для процессов с ограниченным числом параллельных компонентов и др. Однако и в этих случаях предложенные разрешающие алгоритмы оказываются весьма трудоемкими. Мы полагаем, что это обусловлено самой природой процессов π-исчисления. Цель данной работы — показать, что даже для наиболее слабой модели пассивного противника и для сравнительно простых протоколов, в которых используются лишь базовые операции π-исчисления, задача проверки свойств информационной безопасности этих протоколов является co-NP-полной.

Ключевые слова: π-исчисление, протокол, безопасность, пассивный противник, верификация, сложность, NP-полнота

Для цитирования: Аббас М. М., Захаров В.А., "Даже простые процессы π-исчисления трудны для анализа", *Моделирование и анализ информационных систем*, **25**:6 (2018), 589–606.

Об авторах:
Аббас Марат Мазен, orcid.org/0000-0001-8019-7090, студент,
Московский государственный университет им. М.В. Ломоносова,
Ленинские горы, 1, г. Москва, 119991 Россия, e-mail: unlock96@gmail.com

Захаров Владимир Анатольевич, orcid.org/0000-0002-3794-9565, доктор физ.-мат. наук, профессор,
Национальный исследовательский университет «Высшая школа экономики»,
ул. Мясницкая, 20, г. Москва, 101000 Россия,
Институт системного программирования им. В.П. Иванникова РАН, ул. А. Солженицына, 25, 109004, г. Москва,
e-mail: zakh@cs.msu.ru

Благодарности:
[1] Работа выполнена при финансовой поддержке гранта РФФИ N 18-01-00854
[2] Работа выполнена при финансовой поддержке гранта РФФИ N 16-01-00714

1. Введение

В стремлении расширить выразительные возможности исчисления взаимодействующих систем (CCS, Calculus of Communicating Systems) Робин Милнер и его коллеги в статье [25] ввели в рассмотрение новую математическую модель распределенных вычислительных систем — исчисление мобильных процессов, или коротко, π-исчисление. Оно имеет две отличительные особенности: 1) механизм порождения новых имен (ν-оператор) и 2) возможность передачи по каналам связи имен каналов связи. Благодаря этим качествам процессы π-исчисления способны изменять коммуникационную среду, вводя по ходу вычисления новые каналы связи, и моделировать, таким образом, миграцию процессов. В статье [26] Р. Милнеру удалось показать, что процессы π-исчисления способны воспроизводить вычисления термов λ-исчисления. Поэтому π-исчисление, в отличие от CCS, является полной по Тьюрингу моделью вычислений. Своей универсальностью π-исчисление обязано сочетанию двух указанных особенностей с оператором репликации, унаследованным от CCS; при отсутствии любого из этих трех факторов π-исчисление перестает быть универсальной вычислительной моделью.

Спустя сравнительно короткое время после публикации работы [25] было обнаружено, что помимо описания поведения систем мобильных процессов π-исчисление можно успешно использовать для формального описания механизмов работы с объектами в объектно-ориентированном программировании [33], моделирования бизнес-процессов [30] и биохимических реакций [28]. Но, вероятно, наибольший интерес вызвала обнаруженная авторами статьи [1] возможность построения на основе π-исчисления формальных моделей криптографических протоколов.

В основополагающей статье [19] Д. Долев и Э. Яо предложили разделить задачу проверки свойств безопасности криптографических протоколов на две подзадачи: 1) доказательство свойств безопасности (конфиденциальности, целостности и др.) для базовых функций (шифрование, хэширование и пр.), используемых в криптографических протоколах, и 2) проверка стойкости криптографических протоколов в предположении о том, что все используемые в них криптографические примитивы удовлетворяют необходимым требованиям стойкости. При этом модель Долева–Яо наделяет злоумышленника широкими возможностями, включая способность перехватывать, формировать и отправлять сообщения по открытым каналам связи. Предложенное в статье [1] исчисление криптографических протоколов — spi-исчисление — оказалось удачной моделью, в рамках которой проверка стойкости криптографических протоколов в модели Долева–Яо может быть сведена (или, если говорить более формально, определена посредством сведения) к задачам проверки различных видов эквивалентности процессов spi-исчисления или к задаче проверки достижимости особо выделенных «состояний уязвимости» процессов. В работе [2] было предложено еще одно расширение π-исчисления — так называемое прикладное π-исчисление, — в котором разрешается строить сложные термы и описывать их алгебраические свойства при помощи уравнений. Прикладное π-исчисление может быть также обогащено вспомогательной памятью с общедоступными и конфиденциальными ячейками [5, 9], при помощи которой удобно описывать протоколы аутентификации, распределения ключей и др.

Задачам анализа поведения процессов в исчислениях криптографических протоколов, построенных на основе π-исчисления, посвящено большое число работ. Поскольку π-исчисление является алгоритмически полной моделью вычислений, все указанные задачи в общем случае алгоритмически неразрешимы. Поэтому их исследование проводится лишь для некоторых специальных классов процессов с ограниченными вычислительными возможностями, например, для нерекурсивных процессов, в которых все вычисления имеют ограниченную длину, для процессов с ограниченным числом параллельных компонентов и др. Наиболее значительные достижения исследований задач анализа поведения процессов spi-исчисления таковы. Было показано, что задача проверки достижимости разрешима для нерекурсивных процессов spi-исчисления [3, 23] и является NP-полной [29]. Среди различных видов эквивалентности процессов наиболее полезными для криптографических приложений оказываются тестовая эквивалентность [1] и наблюдаемая эквивалентность [2]. Разрешимость задачи проверки тестовой эквивалентности нерекурсивных процессов была установлена в статье [20], однако предложенная разрешающая процедура имеет большую вычислительную трудоемкость. Алгоритмы проверки наблюдаемой эквивалентности для некоторых классов процессов spi-исчисления и прикладного π-исчисление были представлены и исследованы в статьях [10, 15, 16, 18]. В статье [16] было также показано, что для нерекурсивных процессов прикладного π-исчисления задача проверки наблюдаемой эквивалентности NP-полна. Наряду с наблюдаемой эквивалентностью процессов исследовались также и более простые отношения бисимуляции [8]. В статье [32] показано, что задача проверки отношения открытой бисимуляции нерекурсивных процессов spi-исчисления разрешима. Для процессов spi-исчисления исследовалась также возможность их верификации при помощи методов статического анализа [7]. Преимущество этого подхода состоит в том, что он применим к любым (в том числе рекурсивным) процессам. Предложенные формальные методы анализа криптографических протоколов на основе π-исчисление применялись на практике: с их помощью удалось, например, обнаружить и устранить уязвимость в сетевом протоколе безопасной маршрутизации ARAN [22].

Даже в тех случаях, когда нерекурсивный процесс spi-исчисления P имеет всего лишь конечное множество вычислений, задача анализа его поведения во взаимодействии с внешней средой оказывается непростой. Причина состоит в том, что в модели Долева–Яо внешняя среда (злоумышленник) представляется в виде бесконечного семейства процессов $\mathcal{A}$, и задача проверки свойств информационной безопасности состоит в анализе поведения всех процессов вида $P|R$, где $R \in \mathcal{A}$. Процессы семейства $\mathcal{A}$ способны, вообще говоря, формировать сколь угодно сложные сообщения, и поэтому система процессов $\{P|R \ : \ R \in \mathcal{A}\}$ может иметь бесконечно много состояний и вычислений. Методы, предложенные и развитые в работах [10, 15, 16, 18, 20, 32], позволяют выделить (иногда неявно) такое конечное подмножество процессов $\mathcal{A}'$, $\mathcal{A}' \subseteq \mathcal{A}$, описывающих внешнюю среду, что для проверки стойкости протокола P в модели Долева–Яо достаточно проанализировать поведение лишь конечного числа процессов вида $P|R$, где $R \in \mathcal{A}'$. Размер множества процессов $\mathcal{A}'$ оказывается, по меньшей мере, экспоненциально зависящим от размера процесса P, и поэтому алгоритмы проверки свойств безопасности процессов spi-исчисления или прикладного π-исчисления, предложенные в указанных работах, оказываются трудоемкими.

Впервые об NP-полноте задачи проверки свойств небезопасности криптографических протоколов с ограниченным числом сеансов (нерекурсивных протоколов) было объявлено в работах [3,4]. Однако в них рассматривались лишь модели криптографических протоколов с простыми (атомарными) ключами шифрования. Кроме того, авторы статьи [4] привели лишь общую схему обоснования сложности описанной ими разрешающей процедуры. Полное доказательство теоремы об NP-полноте задачи проверки свойств небезопасности нерекурсивных криптографических протоколов с атомарными ключами шифрования было опубликовано в статьях [21,31]. Этот результат был усилен в работе [29]; в ней было показано, что задача проверки свойства небезопасности остается NP-полной и для нерекурсивных криптографических протоколов, использующих составные ключи шифрования. Доказательство принадлежности этой задачи классу сложностей NP опирается на утверждение о том, что всякий нестойкий в модели Долева–Яо протокол может быть скомпрометирован активным противником, способным порождать сообщения, размер которых полиномиально зависит от размера протокола. Этот подход к построению и обоснованию трудоемкости процедур проверки свойств небезопасности был использован в последующих статьях [11–14] для протоколов с более сложными криптографическими примитивами. Однако, как было показано в статье [24], доказательство утверждения о существовании минимальной модели противника полиномиального размера, приведенное в работе [29], содержит ошибку. Она была исправлена в той же статье [24].

Во всех указанных выше работах авторы исследовали задачу проверки свойств небезопасности для криптографических протоколов со все более расширяющимся разнообразием вычислительных и коммуникационных действий, и основное внимание уделялось доказательству принадлежности этой задачи классу сложностей NP. Доказательство NP-трудности рассматриваемой задачи проводилось путем сведения к ней проблемы выполнимости 3-КНФ. Естественно, что это сведение осуществлялось все проще по мере усложнения модели криптографических протоколов. Для этих целей использовались различные средства, допускаемые в spi-исчислении, включая функции шифрования/расшифрования, функции пары, операторы ветвления и др. Мы полагаем, однако, что трудности анализа стойкости криптографических протоколов, моделируемых при помощи процессов различных расширений π-исчисления, обусловлены, в первую очередь, алгоритмическими трудностями, присущими самому π-исчислению как базовой модели систем распределенных мобильных вычислений. Мы рассматриваем задачу проверки стойкости протоколов, описанных при помощи базовых средств π-исчисления, в модели пассивного противника. Анализируемые протоколы представляют собой параллельную композицию процессов $P_1|P_2|\cdots|P_n$, каждый из которых является последовательной композицией действий отправления и приема сообщений $P_i = act_1.act_2.\cdots.act_m$. Среди имен, используемых в качестве сообщений, только одно является конфиденциальным, а все остальные считаются либо общеизвестными, либо случайными именами, использующимися лишь однократно. Пассивный противник имеет возможность перехватывать (подслушивать) сообщения, передаваемые по тем каналам связи, имена которых ему известны. Подслушанные имена расширяют знания противника и могут быть использованы при последующих перехватах. Протокол считается стойким,

если при любом его выполнении знания пассивного противника не будут включать в себя конфиденциальные имена.

В данной статье мы показываем, что для любой 3-КНФ φ можно построить, используя лишь базовые средства π-исчисления, такой процесс $Proc_\varphi = P_1|P_2|\cdots|P_n$ указанного выше вида, который является стойким относительно пассивного противника в том и только том случае, когда 3-КНФ φ невыполнима. При этом размер процесса $Proc_\varphi$ линейно зависит от размера формулы φ, и все вычисления этого процесса завершаются нормально, не попадая в тупики. В свете результатов об NP-полноте задачи проверки свойств небезопасности криптографических протоколов, полученных в статьях [3, 4, 11–14, 21, 24, 29, 31], основная теорема данной статьи показывает, что главным фактором, определяющим сложность рассматриваемой задачи, является ограниченность длины вычислений криптографических протоколов; влияние других факторов, таких как разнообразие используемых в протоколах криптографических примитивов и средств управления вычислением протоколов, структура и размер передаваемых сообщений и др., оказывается второстепенным.

Статья устроена следующим образом. Во втором разделе приводится описание синтаксиса и семантики π-исчисления. В разделе 3 определяется модель пассивного противника и формулируется задача проверки стойкости процессов π-исчисления в модели пассивного противника. В разделе 4 описано устройство процесса $Proc_\varphi$, соответствующего произвольной 3-КНФ φ, и показано, что этот процесс обладает свойством нормальной завершаемости — все его вычисления завершаются в одном и том же пустом процессе. Также установлено, что все вычисления процесса $Proc_\varphi$ безопасны в том и только том случае, если 3-КНФ φ невыполнима. Отсюда следует, что проблема выполнимости 3-КНФ сводима к задаче проверки свойства небезопасности процессов π-исчисления в модели пассивного противника. В заключении мы обсуждаем значение полученных результатов.

2. Синтаксис и семантика π-исчисления

Мы ограничиваемся рассмотрением базового нерекурсивного фрагмента синхронного монадического исчисления мобильных процессов. Пусть задано некоторое бесконечное множество объектов $\mathcal{N}$, которые будут называться *именами*. Имена служат для обозначения каналов связи, а также данных, передаваемых по этим каналам. Для записи имен будут использоваться строчные латинские буквы $a, b, \ldots, x, y, z$.

Элементарным действием синхронной коммуникации E называется всякое выражение вида $\overline{x}\langle y\rangle$ (отправление сообщения y по каналу связи x) или $x(y)$ (прием сообщения, связывающего имя y, по каналу связи x). *Процессом* π-исчисления называется всякое выражение, которое может быть построено по следующим правилам:

$$
\begin{array}{rcll}
P, Q & ::= & \mathbf{0} & \text{(завершить выполнение процесса)} \\
 & | & E.P & \text{(выполнить действие } E \text{ и перейти к выполнению процесса } P) \\
 & | & P|Q & \text{(выполнять параллельно процессы } P \text{ и } Q) \\
 & | & (\nu x)\ P & \text{(ввести новое имя } x \text{ и перейти к выполнению процесса } P).
\end{array}
$$

Множество всех определенных таким образом процессов обозначим символом $\mathcal{P}$.

Вхождения имен в процесс подразделяются на *свободные* и *связанные*. Множество $fn(P)$ свободных имен процесса P определяется в зависимости от устройства процесса по следующим правилам:

1. $fn(\mathbf{0}) = \emptyset$,
2. $fn(\overline{x}\langle y\rangle.P) = fn(P) \cup \{x, y\}$, $fn(x(y).P) = fn(P) \cup \{x\}) \setminus \{y\}$,
3. $fn((\nu x)\ P) = fn(P) \setminus \{x\}$,
4. $fn(P|Q) = fn(P) \cup fn(Q)$.

Вхождение имени x в процесс P считается свободным, если это вхождение не содержится ни в одном подпроцессе вида $(\nu x)\ P'$ процесса P. Запись вида $P\{x/y\}$ обозначает процесс, полученный из процесса P одновременной заменой всех свободных вхождений имени y именем x. Подстановка $\{x/y\}$ называется *правильной для процесса* P, если каждое свободное вхождение имени y не содержится ни в одном подпроцессе вида $(\nu x)\ P'$ процесса P.

Операционная семантика мобильных процессов определяется отношением структурной конгруэнтности $\equiv_\pi$ и отношением редукции $\to_\pi$. Отношение $\equiv_\pi$ — это наименьшее отношение конгруэнтности на множестве процессов $\mathcal{P}$, удовлетворяющее следующим тождествам:

1. $\mathbf{0}|P \equiv_\pi P$, $P|Q \equiv_\pi Q|P$, $P|(Q|R) \equiv_\pi (P|Q)|R$, т.е. система $(\mathcal{P}, |, \mathbf{0})$ является коммутативной полугруппой;
2. $(\nu y)\ P \equiv_\pi (\nu x)\ P\{x/y\}$ для всякого имени $x, x \notin fn(P)$, и правильной для процесса P подстановки $\{x/y\}$;
3. $((\nu x)\ P)|Q \equiv_\pi (\nu x)\ (P|Q)$ для всякого имени $x, x \notin fn(Q)$;
4. $(\nu x).\mathbf{0} \equiv_\pi \mathbf{0}$, $\quad (\nu x)\ ((\nu y)\ P) \equiv_\pi (\nu y)\ ((\nu x)\ P)$.

Нетрудно заметить, что если $P \equiv_\pi Q$, то $fn(P) = fn(Q)$.

Отношение редукции — это наименьшее четырехместное отношение $\to_\pi \subseteq \mathcal{P} \times \mathcal{N} \times \mathcal{N} \times \mathcal{P}$, удовлетворяющее для любых процессов P, Q, P', Q' и имен x, y, z следующим требованиям (выполнимость отношения $\to_\pi$ для четверки P, x, y, Q традиционно обозначается записью $P \xrightarrow{x(y)}_\pi Q$):

1. $(\overline{x}\langle y\rangle.P)|(x(z).Q) \xrightarrow{x(y)}_\pi P|Q\{y/z\}$;
2. если $P \xrightarrow{x(y)}_\pi P'$, то $P|Q \xrightarrow{x(y)}_\pi P'|Q$;
3. если $P \xrightarrow{x(y)}_\pi P'$, то $(\nu z).\ P \xrightarrow{x(y)}_\pi (\nu z).\ P'$;
4. если $P \equiv_\pi Q$, $P \xrightarrow{x(y)}_\pi P'$ и $P' \equiv_\pi Q'$, то $Q \xrightarrow{x(y)}_\pi Q'$.

Четверки $P \xrightarrow{x(y)}_{\pi} Q$, удовлетворяющие отношению редукции процессов, будем называть *редукциями* (или, иначе, коммуникациями по каналу связи x). Канал связи x, по которому может быть выполнена редукция процесса P, называется *активным в процессе* P.

Вычислением процесса P_0 назовем всякую последовательность редукций вида

$$P_0 \xrightarrow{x_1(y_1)}_{\pi} P_1 \xrightarrow{x_2(y_2)}_{\pi} \cdots \xrightarrow{x_{n-1}(y_{n-1})}_{\pi} P_{n-1} \xrightarrow{x_n(y_n)}_{\pi} P_n. \tag{1}$$

Процесс, не допускающий ни одной редукции, называется *тупиковым*. Вычисление (1) считается *завершенным*, если процесс P_n является тупиковым. Если $P_n \equiv_{\pi} \mathbf{0}$, то завершение вычисления является *нормальным*. Очевидно следующее утверждение, которым мы будем пользоваться при анализе вычислений процессов.

Утверждение 1. *Если в процессе P_0 активен канал связи x, то в любом завершенном вычислении (1) происходит коммуникация по каналу x.*

Иными словами, по каждому активному в процессе P_0 каналу связи рано или поздно обязательно будет передано сообщение.

3. Модель пассивного противника

По отношению к заданному процессу π-исчисления P сторонний наблюдатель (противник) мыслится как некоторый процесс R, который может вступать во взаимосвязь с процессом P. Это взаимодействие определяется параллельной композицией $P|R$. Все имена процесса P, связанные оператором ν, могут быть истолкованы как однократно используемые данные, порождаемые процессом. Поэтому вначале коммуникация процессов P и R может осуществляться только по каналам связи, принадлежащим множеству $fn(P)$. Но затем противник, получая (перехватывая) сообщения, переданные по этим каналам, узнает новые имена, которыми он сможет пользоваться для последующей коммуникации с P. Пассивный противник R способен лишь подслушивать сообщения, отправленные процессом P по тем каналам связи, которые известны противнику в момент выполнения действий отправления и приема сообщения. Такая способность пассивного противника моделируется парой последовательно выполняемых действий $x(y).\overline{x}\langle y\rangle$. Подслушанные таким образом сообщения не теряются и не подменяются. В роли пассивного противника может выступать любой процесс R, представляющий собой последовательную композицию пар действий приема и отправления одного и того же сообщения по одному и тому же каналу связи. В отличие от пассивного противника, активный противник способен не только подслушивать сообщения, но также конструировать новые сообщения и осуществлять подмену перехваченных сообщений; в роли активного противника может выступать любой процесс π-исчисления R.

Свойства информационной безопасности процесса P проявляются в его взаимодействии с произвольным противником того или иного вида. Чтобы избежать необходимости рассматривать поведение всех возможных параллельных композиций вида $P|R$, обычно используется модель абстрактного противника, которая определяется состоянием его знаний K — множеством известных противнику имен. Это

множество имен может изменяться по ходу выполнения действий процессом P. В случае взаимодействия процесса P с пассивным противником, преобразование состояния (P, K) взаимодействующей системы в состояние (P', K') возможно только за счет выполнения некоторого перехода процесса P. Если противник является активным, то преобразование состояния (P, K) в состояние (P', K') может произойти как за счет выполнения некоторой редукции процесса P, так и в результате выполнения некоторого действия $x(y)$ или $\overline{x}\langle y\rangle$ процесса P при условии, что состояние знаний противника K позволяет ему сформировать парное действие вида $\overline{x}\langle z\rangle$ или $x(z)$ соответственно. Целью противника является достижение состояния знаний из некоторого выделенного семейства S, которое составляет угрозу безопасности.

В данной статье мы ограничиваемся рассмотрением взаимодействия процессов π-исчисления с пассивным противником. Пассивный противник характеризуется списком записей об известных ему именах. Для формирования записей введем новый оператор κ и будем использовать выражение (κx), где $x \in \mathcal{N}$, для обозначения записи имени x в базе данных противника. *Модель пассивного противника* представляет собой всякий список записей (базу данных) A, который может быть построен по следующим правилам:

$$\begin{array}{rcll} A & ::= & \mathbf{0} & \text{(пустая база данных)} \\ & | & (\kappa x).A & \text{(база данных, содержащая запись имени } x\text{)}. \end{array}$$

В отличие от оператора ν оператор записи κ не осуществляет связывание имен, и поэтому множество $fn(A)$ свободных имен противника A состоит из всех имен, входящих в состав выражения A.

Пассивный противник может наблюдать за вычислениями процесса π-исчисления; для формального описания этого явления будем использовать оператор параллельной композиции. *Мониторингом* процесса P противником A будем называть всякое выражение M, которое может быть построено по следующим правилам:

$$\begin{array}{rcll} M & ::= & P|A & \text{(противник } A \text{ наблюдает за процессом } P\text{)} \\ & | & (\nu x).M & \text{(мониторинг } M \text{ проводится в области определения} \\ & & & \text{однократно используемого имени } x\text{)}. \end{array}$$

Множество всех возможных мониторингов обозначим символом $\mathcal{M}$. Множество свободных имен мониторинга M обозначим $fn(M)$. Во множестве $fn(M)$ свободных имен мониторинга M нас будут особо интересовать свободные имена, входящие в состав записей противника вида (κx). Подмножество всех свободных имен указанного вида обозначим выражением $open(M)$.

Операционная семантика мониторингов, так же как семантика процессов π-исчисления, определяется отношением структурной конгруэнтности $\equiv$ и отношением переходов $\rightarrow$. Отношение $\equiv$ — это наименьшее отношение конгруэнтности на множестве мониторингов $\mathcal{M}$, которое включает отношение структурной конгруэнтности $\equiv_\pi$ на множестве процессов и удовлетворяет следующим тождествам:

1. $(\nu y).\ M \equiv (\nu x).\ M\{x/y\}$ для всякого имени $x, x \notin fn(M)$, и правильной для мониторинга M подстановки $\{x/y\}$;

2. $(\kappa x).(\kappa y).A \equiv (\kappa y).(\kappa x).A$ для всякой пары имен x, y и противника A;

3. $(\kappa x).(\kappa x).A \equiv (\kappa x).A$ для всякого имени x и противника A

Первое из этих тождеств означает, что все сведения противника об однократно используемых именах действительны только в контексте наблюдаемого процесса, а два других тождества позволяют рассматривать базу знаний противника как неупорядоченное множество записей. Учитывая последнее обстоятельство, условимся для произвольного множества имен $X = \{x_1, x_2, \ldots, x_n\}$ обозначать выражением $A(X)$ противника $(\kappa x_1).(\kappa x_2).\ldots.(\kappa x_n).\mathbf{0}$.

Отношение переходов $\to$ — это наименьшее бинарное отношение на множестве мониторингов $\mathcal{M}$, удовлетворяющее следующим требованиям для любых процессов P, Q, P', Q', модели противника A, мониторингов M, N, M', N' и имен x, y:

1. если $P \xrightarrow{x(y)}_{\pi} P'$, то $P|A \to P'|A$;
2. если $P \xrightarrow{x(y)}_{\pi} P'$, то $P|(\kappa x).A \to P'|(\kappa y).(\kappa x).A$;
3. если $M \to M'$, то $(\nu x).\ M \to (\nu x).\ M'$;
4. если $M \equiv N$, $M \to M'$ и $M' \equiv N'$, то $N \to N'$.

Второе из приведенных здесь правил гласит: если наблюдаемый процесс выполняет коммуникацию некоторых данных по каналу связи, имя которого известно противнику, то передаваемые данные тоже становятся известны противнику. Рефлексивно-транзитивное замыкание отношения переходов обозначим $\to^*$. Мониторинг M' считается *достижимым* из мониторинга M, если выполняется отношение $M \to^* M'$.

Пример 1. Пусть имеется процесс

$$P = ((\nu x).(\nu y).\ \overline{ch}\langle x\rangle.x(y).\mathbf{0})|((\nu z).\ ch(z).\overline{z}\langle secret\rangle.\mathbf{0})$$

и модель пассивного противника $A = (\kappa\ ch).\mathbf{0}$. Тогда мониторинг $M = P|A$ порождает последовательность переходов

$$\begin{array}{c} M \\ \downarrow \\ (\nu x).(\nu z).\ \big(((\nu y).\ x(y).\mathbf{0})|(\overline{x}\langle secret\rangle.\mathbf{0})|(\kappa x).(\kappa\ ch).\mathbf{0}\big) \\ \downarrow \\ (\nu x).(\nu y).(\nu z).\ \big((\kappa\ secret).(\kappa x).(\kappa\ ch).\mathbf{0}\big)\ . \end{array}$$

Цель противника, располагающего определенными априорными сведениями о процессе, состоит в выведывании некоторых конфиденциальных данных, с которыми оперирует процесс. Поэтому и атакой, и угрозой, относительно которых формулируется требование стойкости процессов, являются конечные множества имен X и Y. Будем говорить, что процесс P является *стойким относительно угрозы Y при осуществлении атаки X* (коротко, (X, Y)-*стойким*), если для любого мониторинга M, достижимого из начального мониторинга $P|A(X)$, неверно, что $Y \subseteq \mathit{open}(M)$, т.е. ни в одном вычислении процесса P пассивный противник, знающий вначале лишь множество имен X, не может подслушивать коммуникации процесса так, чтобы сформировать из подслушанных имен множество Y. Задача проверки стойкости

процессов в модели пассивного противника состоит в том, чтобы для произвольного заданного процесса P, атаки X и угрозы Y выяснить, является ли (X, Y)-стойким процесс P. В приведенном выше примере процесс P не является стойким относительно угрозы $\{secret\}$ при осуществлении атаки $\{ch\}$.

4. Сложность задачи проверки стойкости процессов

Теорема 1. *Задача проверки стойкости процессов множества $\mathcal{P}$ является co-NP-полной.*

Доказательство. Поскольку процессы множества $\mathcal{P}$ не содержат оператора репликации ! или иных средств рекурсивного описания вычислений, длина каждого вычисления процесса P из множества $\mathcal{P}$ не превосходит размера процесса P. Кроме того, как можно видеть из определения отношения переходов $\to$, у каждого мониторинга M число его попарно неконгруэнтных наследников (образов) M' по отношению $\to$ не превосходит квадрата от размера M. Отсюда следует, что задача проверки стойкости процессов множества $\mathcal{P}$ принадлежит классу сложностей co-NP.

Для обоснования co-NP-трудности рассматриваемой задачи покажем, что к ней *log-space* сводима задача проверки невыполнимости 3-КНФ. Для произвольной заданной 3-КНФ φ мы построим процесс $Proc_\varphi$, который способен моделировать вычисление значения формулы φ на всех наборах значений переменных и позволяет противнику подслушать конфиденциальное имя $secret$, переданное по открытому каналу связи ch в том и только том случае, когда φ выполнима.

Пусть задана произвольная 3-КНФ $\varphi = D_1 \wedge D_2 \wedge \cdots \wedge D_N$, зависящая от переменных $x_1, x_2, \ldots, x_n$, в которой каждый дизъюнкт $D_i,\ 1 \le i \le N$, имеет вид $\ell_{1i} \vee \ell_{2i} \vee \ell_{3i}$, где $\ell_{1i}, \ell_{2i}, \ell_{3i}$ — литеры, являющиеся переменными или их отрицаниями. Мы будем различать имя литеры ℓ и имя той переменной, на основе которой определяется эта литера. Условимся для каждой литеры ℓ использовать запись x^σ, где $\sigma = 1$, если $\ell = x$, и $\sigma = 0$, если $\ell = \neg x$. Без ограничения общности можно считать, что каждая литера $x_i^\sigma,\ 1 \le i \le n,\ \sigma \in \{0, 1\}$, входит в состав 3-КНФ φ. Кроме того, для литеры ℓ мы будем обозначать записью ℓ^* контрарную литеру противоположной полярности, а записью m_ℓ общее число вхождений литеры ℓ в формулу φ.

Опишем устройство процесса $Proc_\varphi$. В нем есть только два свободных имени ch и $secret$. Имя ch обозначает открытый (доступный для подслушивания) канал связи; это имя составляет атаку. Имя $secret$ обозначает конфиденциальные данные, и оно составляет угрозу. Все остальные имена, фигурирующие в процессе $Proc_\varphi$, связаны операторами ν. Это множество имен состоит из имен переменных $x_1, \ldots, x_n$, имен положительных и отрицательных литер, соответствующих этим переменным, $\ell_1, \ldots, \ell_n, \ell_1^*, \ldots, \ell_n^*$, имен дизъюнктов $d_1, \ldots, d_N$, а также три особых имени g, h и r. Процесс $Proc_\varphi$ представляет собой композицию параллельных подпроцессов, которым отведены следующие роли.

1. Подпроцесс $Init$ призван активизировать для каждой переменной $x_i,\ 1 \le i \le n$, в точности одну из литер x_i или $\neg x_i$, отправив для каждой переменной x_i сообщение, которое может принять только один из двух процессов S_{ℓ_i} или $S_{\ell_i^*}$:

$$Init = \overline{x_1}\langle z\rangle.\overline{x_2}\langle z\rangle.\ldots.\overline{x_n}\langle z\rangle.\mathbf{0}\ .$$

2. Для каждой из $2n$ литер $\ell = x_i^\sigma$, $1 \le i \le n$, $\sigma \in \{0,1\}$, подпроцесс S_ℓ призван активизировать все каналы связи с именем ℓ, используемые в процессе $Proc_\varphi$:

$$S_\ell = x_i(z).\underbrace{\overline{\ell}\langle z\rangle.\overline{\ell}\langle z\rangle.\ldots.\overline{\ell}\langle z\rangle}_{m_\ell + m_{\ell^*}\ \text{раз}}.\mathbf{0}\ .$$

3. Подпроцесс $Check_{\varphi=0}$ предназначен для проверки того, что формула φ принимает значение 0 на том наборе значений переменных, который соответствует активизированным литерам:

$$Check_{\varphi=0} = Check_{D_1=0}|Check_{D_2=0}|\ldots|Check_{D_N=0}\ ,$$

где для каждого дизъюнкта $D_i = \ell_{1i} \vee \ell_{2i} \vee \ell_{3i}$ подпроцесс $Check_{D_i=0}$ имеет вид

$$Check_{D_i=0} = \ell^*_{1i}(z).\ell^*_{2i}(z).\ell^*_{3i}(z).\overline{r}\langle ch\rangle.\mathbf{0}\ .$$

Таким образом, каждый процесс $Check_{D_i=0}$ в том случае, если активизированы литеры, контрарные по отношению к литерам $\ell_{1i}, \ell_{2i}, \ell_{3i}$, отправляет по каналу связи r имя открытого канала ch.

4. Подпроцесс $Check_{\varphi=1}$ предназначен для проверки того, что формула φ принимает значение 1 на том наборе значений переменных, который соответствует активизированным литерам:

$$Check_{\varphi=1} = Check_{D_1=1}|Check_{D_2=1}|\ldots|Check_{D_N=1}|CheckAll\ ,$$

где для каждого дизъюнкта $D_i = \ell_{1i} \vee \ell_{2i} \vee \ell_{3i}$ подпроцесс $Check_{D_i=1}$ имеет вид

$$Check_{D_i=1} = (\ell_{1i}(z).\overline{d_i}\langle z\rangle.\mathbf{0})|(\ell_{2i}(z).\overline{d_i}\langle z\rangle.\mathbf{0})|(\ell_{3i}(z).\overline{d_i}\langle z\rangle.\mathbf{0})\ ,$$

а подпроцесс $CheckAll$ имеет вид

$$CheckAll = d_1(z).d_2(z).\ldots.d_N(z).\overline{r}\langle secret\rangle.\mathbf{0}\ .$$

Таким образом, процесс $Check_{\varphi=1}$ отправляет по каналу связи r секретное имя $secret$, если для каждого дизъюнкта $D_i = \ell_{1i} \vee \ell_{2i} \vee \ell_{3i}$, $1 \le i \le N$, была активизирована хотя бы одна входящая в него литера ℓ_{ji}, $1 \le j \le 3$.

5. Подпроцесс $OpenCh$ призван осуществить коммуникацию по открытому каналу связи ch после того, как было проверено значение формулы φ, и запустить подпроцесс «сборки мусора», который позволит выполнить все незавершенные действия коммуникации процесса $Proc_\varphi$:

$$OpenCh = (r(y).\overline{ch}\langle y\rangle.\overline{g}\langle z\rangle.\overline{h}\langle z\rangle.\mathbf{0})|(ch(x).\mathbf{0})\ .$$

Запуск подпроцесса «сборки мусора» осуществляют действия отправления сообщений $\overline{g}\langle z\rangle$ и $\overline{h}\langle z\rangle$.

6. Подпроцесс «сборки мусора» $Garbage$ предназначен для того, чтобы активизировать все те литеры, которые не были активизированы подпроцессом $Init$, и заставить выполниться все действия подпроцессов $Check_{\varphi=0}$ и $Check_{\varphi=1}$, оставшиеся невыполненными после первой активизации литер процессом $Init$:

$$Garbage = Final \mid Collect \ ,$$

где

$$Final = g(z).\overline{x_1}\langle z\rangle.\overline{x_2}\langle z\rangle.\ldots.\overline{x_n}\langle z\rangle.\mathbf{0} \ ,$$

$$Collect = h(z).d_1(z).d_1(z).d_2(z).d_2(z).\ldots.d_N(z).d_N(z).r(y_1).r(y_2).\ldots.r(y_N).\mathbf{0} \ .$$

Обозначим записью $names$ список всех имен, за исключением ch и $secret$, встречающихся в указанных выше подпроцессах. Процесс $Proc_\varphi$ представляет собой параллельную композицию всех этих подпроцессов, в которой все имена из списка $names$ связаны оператором ν:

$$Proc_\varphi = (\nu\ names).(Init|S_{\ell_1}|S_{\ell_1^*}|\ldots|S_{\ell_n}|S_{\ell_n^*}|Check_{\varphi=0}|Check_{\varphi=1}|OpenCh|Garbage) \ .$$

Легко видеть, что, располагая КНФ φ, процесс $Proc_\varphi$ можно построить, используя лишь логарифмическую память.

Вначале покажем, что любое завершенное вычисление процесса $Proc_\varphi$

$$Proc_\varphi = P_0 \longrightarrow_\pi P_1 \longrightarrow_\pi \cdots \longrightarrow_\pi P_{m-1} \longrightarrow_\pi P_m \tag{2}$$

оканчивается нормально, т.е. $P_m = \mathbf{0}$.

Прежде всего, заметим, что ни одно имя u, являющееся аргументом какого-либо действия приема сообщения $v(u)$, не является именем никакого канала связи. Это означает, что имена каналов связи по ходу вычисления (2) не изменяются (возможно лишь переименование, допустимое по отношению конгруэнтности $\equiv_\pi$). Кроме того, в процессе $Proc_\varphi$ для каждого имени канала связи u число действий отправления сообщений по каналу u равно числу действий приема сообщений по этому же каналу. Поэтому для доказательства нормального завершения вычисления (2) достаточно показать, что в нем выполнены все действия отправления сообщений процесса $Proc_\varphi$.

Заметим, что в процессе $Proc_\varphi$ активен канал x_1, а процесс P_m является тупиковым. Значит, согласно утверждению 1, по ходу вычисления (2) была выполнена коммуникация по каналу x_1. Действия приема сообщения по этому каналу связи есть только в подпроцессах активизации литер S_{x_1} и $S_{\neg x_1}$. Рассмотрим тот из этих двух подпроцессов, в котором коммуникация по каналу связи x_1 выполнилась раньше всего в вычислении (2); пусть этот подпроцесс предназначен для активизации литеры $\ell_1 = x_1^{\sigma_1}$. Тогда после выполнения первой коммуникации по каналу x_1 активным становится канал связи ℓ_1. Согласно описанию подпроцесса S_ℓ активность канала ℓ_1 будет поддерживаться по ходу вычисления до тех пор, пока не сработают все возможные действия приема сообщений по этому каналу.

После выполнения коммуникации по каналу x_1 (в любом из подпроцессов $Init$ или $Final$) активным становится также канал связи x_2, для которого будут справедливы рассуждения подобные тем, что были приведены выше для канала x_1. Проводя

эти рассуждения для всех каналов связи $x_1, x_2, \ldots, x_n$, приходим к заключению о том, что в вычислении (2) для некоторого двоичного набора $\alpha = (\sigma_1, \sigma_1, \ldots, \sigma_n)$ оказываются активизированными каналы связи $\ell_1 = x_1^{\sigma_1}, \ell_2 = x_2^{\sigma_2}, \ldots, \ell_n = x_n^{\sigma_n}$, и их активность будет поддерживаться по ходу вычисления до тех пор, пока не сработают все возможные действия приема сообщений по этим каналам.

Далее необходимо рассмотреть два случая, в зависимости от того, какое значение принимает формула φ на наборе α значений переменных.

Если $\varphi(\alpha) = 0$, то для некоторой тройки выделенных выше литер $\ell_{i_1}, \ell_{i_2}, \ell_{i_3}$ КНФ φ содержит дизъюнкт $D_j = \neg\ell_{i_1} \vee \neg\ell_{i_2} \vee \neg\ell_{i_3}$. Согласно описанию подпроцесса $Check_{\varphi=0}$, одним из компонентов его параллельной композиции является подпроцесс

$$Check_{D_j=0} = \ell_{i_1}(z).\ell_{i_2}(z).\ell_{i_3}(z).\overline{r}\langle ch\rangle.\mathbf{0}\ .$$

Поскольку, как было установлено выше, подпроцессы вида S_ℓ поддерживают активность каналов $\ell_{i_1}, \ell_{i_2}, \ell_{i_3}$, пока не выполнятся все возможные действия приема сообщений по этим каналам, в вычислении (2) осуществляются коммуникации по каналам $\ell_{i_1}, \ell_{i_2}, \ell_{i_3}$, в которых принимают участие действия приема сообщений подпроцесса $Check_{D_j=0}$. После выполнения последней из этих трех редукций активным становится канал связи r, поскольку действием приема сообщения по этому каналу начинается один из компонентов параллельной композиции подпроцесса $OpenCh$.

Если $\varphi(\alpha) = 1$, то каждый дизъюнкт D_j, $1 \le j \le N$, содержит одну из выделенных выше литер $\ell_1, \ell_2, \ldots, \ell_n$. Значит, в соответствии с описанием подпроцесса $Check_{\varphi=1}$ для каждого j, $1 \le j \le N$, параллельная композиция этого подпроцесса содержит компонент $\ell_i(z).\overline{d_j}\langle z\rangle.\mathbf{0}$, где ℓ_i — одна из выделенных литер, являющаяся именем активизированного канала связи. Ввиду того, что активность «литерных» каналов поддерживается, пока не сработают все возможные действия приема сообщений по этим каналам, в вычислении (2) осуществляются редукции, в которых принимают участие все действия приема сообщений, стоящие в начале указанных компонентов. После выполнения этих редукций поочередно активизируются каналы связи с именами $d_1, d_2, \ldots, d_N$. Активизация этих каналов обусловлена тем, что действия приема сообщений по этим каналам стоят в начале последовательной композиции действий, образующей подпроцесс $CheckAll$. Тогда, согласно утверждению 1, в вычислении (2) происходят коммуникации по каналам связи $d_1, d_2, \ldots, d_N$.

Если все упомянутые выше коммуникации по каналам связи $d_1, d_2, \ldots, d_N$ происходят с привлечением действий приема сообщений из подпроцесса $CheckAll$, то после их выполнения активизируется канал связи r. А в том случае, если хотя бы одна из упомянутых коммуникаций происходит с привлечением действия приема сообщения из подпроцесса $Collect$, то это оказывается возможным согласно описанию этого подпроцесса только после коммуникации по каналу связи h. Однако передача сообщения по этому каналу, как видно из описания подпроцесса $OpenCh$, возможна только после коммуникации по каналу связи r.

Таким образом, независимо от значения $\varphi(\alpha)$ в вычислении (2) активизируется канал r. Значит, согласно утверждению 1, в вычислении (2) происходит коммуникация по каналу связи r. Рассмотрим самую первую из таких редукций. Она не может проводиться с использованием действий приема сообщений из подпроцесса $Collect$: как было отмечено выше, эти действия приема сообщений могут быть использованы только после коммуникации по каналу связи h, а этот канал может быть активи-

зирован лишь после проведения хотя бы одной коммуникации по каналу связи r. Следовательно, первая коммуникация по каналу связи r в вычислении (2) проводится с привлечением действия приема сообщения по этому каналу из подпроцесса $OpenCh$.

Как видно из описания этого подпроцесса, после выполнения первого содержащегося в нем действия приема сообщения по каналу r последовательно активизируются каналы ch, g и h. Поэтому, согласно утверждению 1, в вычислении (2) происходят коммуникации по указанным каналам связи. После выполнения этих редукций оказываются вновь активизированы все каналы связи $x_1, x_2, \ldots, x_n$. После проведения коммуникаций по этим каналам с использованием действий приема сообщений из подпроцессов вида S_ℓ оказываются активизированы все каналы связи, соответствующие всем литерам КНФ φ. После проведения коммуникации по всем активизированным «литерным» каналам связи с привлечением соответствующих действий приема сообщений из подпроцессов $Check_{\varphi=0}$ и $Check_{\varphi=1}$ активными оказываются все каналы связи $d_1, d_2, \ldots, d_N$, а также канал связи r. Выполнение коммуникаций по этим каналам связи проводится с использованием действий приема сообщений из подпроцесса $Collect$, а также подпроцесса $CheckAll$ в случае $\varphi(\alpha) = 0$. В результате проведения этих редукций в вычислении (2) образуется процесс P_m, в котором нет ни одного действия, т.е. $P_m \equiv \mathbf{0}$.

Таким образом, всякое вычисление (2) процесса $Proc_\varphi$ завершается нормально.

Далее заметим, что процесс $Proc_\varphi$ допускает передачу по каналу связи r только имен ch и $secret$. Покажем, что в любом вычислении (2) процесса $Proc_\varphi$ выполнение редукции $P_i \overset{ch(secret)}{\longrightarrow_\pi} P_{i+1}$ возможно в том и только том случае, когда 3-КНФ φ выполнима.

Единственное действие передачи сообщения по открытому каналу связи ch содержится в подпроцессе $OpenCh$. Он устроен так, что редукции $P_i \overset{ch(secret)}{\longrightarrow_\pi} P_{i+1}$ в вычислении (2)

- должна предшествовать редукция $P_j \overset{r(secret)}{\longrightarrow_\pi} P_{j+1}$,
- не может предшествовать ни одна коммуникация с привлечением действий из подпроцесса $Garbage$.

Все действия отправления сообщений по каналу r содержатся только в подпроцессах $Check_{\varphi=0}$ и $Check_{\varphi=1}$. Однако в подпроцессе $Check_{\varphi=0}$ эти действия призваны отправлять по каналу r имя ch, и лишь в подпроцессе $CheckAll$ имеется единственное действие отправления имени $secret$ по каналу связи r. Этому действию в подпроцессе $CheckAll$ предшествуют действия приема сообщений по каналам связи $d_1, d_2, \ldots, d_N$. Значит, выполнение редукции $P_j \overset{r(secret)}{\longrightarrow_\pi} P_{j+1}$ в вычислении (2) возможно в том и только том случае, если этому выполнению предшествовали выполнения коммуникаций по каналам связи $d_1, d_2, \ldots, d_N$.

Действия отправления сообщений по указанным каналам связи содержатся только в подпроцессах $Check_{D_k=1}$, $1 \le k \le N$. Согласно описанию каждого из подпроцессов $Check_{D_k=1}$ активизации канала связи d_k предшествует выполнение коммуникации по одному из каналов $\ell_{1k}, \ell_{2k}, \ell_{3k}$. Обозначим записью ℓ_{i_k} имя одного из тех каналов $\ell_{1k}, \ell_{2k}, \ell_{3k}$, выполнение коммуникации по которому предшествовало в вычислении (2) активизации канала связи d_k.

Рассмотрим множество выделенных литер $L = \{\ell_{i_k} \ : \ 1 \leq k \leq N\}$. Согласно описанию подпроцессов $Check_{D_k=1},\ 1 \leq k \leq N$, в каждом дизъюнкте D_k КНФ ϕ содержится одна из литер рассматриваемого множества. Кроме того, можно заметить, что в множестве литер L нет контрарных пар. Действительно, если бы в L содержалась контрарная пара литер $\ell = x^0_{i_k}$ и $\ell^* = x^1_{i_k}$, то это означало бы, что в вычислении (2) оба канала связи ℓ и ℓ^* были активизированы ранее, чем была выполнена редукция $P_i \overset{ch(secret)}{\longrightarrow_\pi} P_{i+1}$. Согласно описанию подпроцессов S_ℓ и S_{ℓ^*} активизация обоих указанных каналов может осуществиться только после выполнения двух коммуникаций по каналу x_m. Действия отправления сообщения по каналу x_m содержатся в подпроцессах $Garbage$ и $Init$, однако, на отрезке вычисления (2), предшествующем выполнению редукции $P_i \overset{ch(secret)}{\longrightarrow_\pi} P_{i+1}$, все действия подпроцесса $Garbage$ еще заблокированы, а подпроцесс $Init$ содержит лишь одно действие отправления сообщения по каналу x_m. Следовательно, только один из двух каналов связи ℓ и ℓ^* может быть активизирован до выполненния редукции $P_i \overset{ch(secret)}{\longrightarrow_\pi} P_{i+1}$ в вычислении (2).

Существование непротиворечивого множества литер, которое имеет хотя бы одну общую литеру с каждым дизъюнктом КНФ $Proc_\varphi$, является признаком выполнимости КНФ φ. Таким образом, если в каком-либо вычислении процесса $Proc_\varphi$ проводится редукция $P_i \overset{ch(secret)}{\longrightarrow_\pi} P_{i+1}$, то КНФ φ выполнима. И, напротив, если КНФ φ выполнима, то нетрудно построить вычисление процесса $Proc_\varphi$, в котором по открытому каналу связи ch передается имя $secret$. Значит, процесс $Proc_\varphi$ является стойким относительно угрозы $\{secret\}$ при осуществлении атаки $\{ch\}$ в том и только том случае, если КНФ φ невыполнима. Следовательно, проблема невыполнимости 3-КНФ *log-space* сводима к задаче проверки стойкости процессов множества $\mathcal{P}$ в модели пассивного противника. □

5. Заключение

Авторы еще раз отмечают, что представленный в статье результат о co-NP-полноте задачи проверки стойкости моделей нерекурсивных криптографических протоколов не является принципиально новым. Новизна этого результата состоит в том, что он был получен для, вероятно, самой простой модели вычислений, в которой можно сформулировать содержательную задачу проверки свойств информационной безопасности. Теорема 1 свидетельствует о том, что данная задача является вычислительно трудной даже в самой простой постановке, когда в проверяемых протоколах отсутствуют какие-либо криптографические примитивы, а противник является пассивным.

Особого внимания заслуживает модель пассивного противника и новое понятие мониторинга, расширяющее выразительные возможности исчислений мобильных процессов как средства описания криптографических протоколов. Ранее, насколько нам известно, моделирование противника и его взаимодействия с протоколом осуществлялось за рамками строгой модели π-исчисления. Мы убеждены, что с привлечением понятия мониторинга удастся разработать и общую модель активного противника, соответствующую концепции Долева–Яо. Создание такой модели

и получение для нее результатов о сложности задачи проверки стойкости протоколов, подобных тем, которые были установлены в статьях [4,11,21,24,29,31], является целью наших дальнейших исследований.

Поскольку задача проверки стойкости нерекурсивных процессов π-исчисления оказалась вычислительно трудной, представляет интерес вопрос о том, для каких классов процессов эта задача может быть решена за полиномиальное время. Как показывает доказательство теоремы 1, эта задача тесно связана с задачей проверки нормальной завершаемости процессов π-исчисления, которая является актуальной задачей проверки правильности поведения систем взаимодействующих процессов. Установление эффективно проверяемых достаточных условий нормальной завершаемости процессов π-исчисления также рассматривается нами как одна из тем дальнейших исследований.

Список литературы / References

[1] Abadi M., Gordon A. D., "A Calculus for Cryptographic Protocols: The Spi Calculus", *Information and Computation*, **148**:1 (1999), 1–70.

[2] Abadi M., Fournet C., "Mobile Values, New Names, and Secure Communication", *Proceedings of the 28-th ACM Symposium on Principles of Programming Languages*, 2001, 104–115.

[3] Amadio M. R., Lugiez D., "On the Reachability Problem for Cryptographic Protocols", *Proceedings of the 11-th International Conference on Concurrency Theory*, 2000, 380–394.

[4] Amadio M. R., Lugiez D., Vanackere V., "On the symbolic reduction of processes with cryptographic functions", *Theoretical Computer Science*, **290**:1 (2003), 695–740.

[5] Arapinis M., Liu J., Ritter E., Ryan M., "Stateful Applied Pi Calculus", *Proceedings of the Principles of Security and Trust—Third International Conference*, 2014, 22–41.

[6] Blanchet B., Smith B., "Automated reasoning for equivalences in the applied pi calculus with barriers", *Proceedings of the 29-th IEEE Computer Security Foundations Symposium*, 2014, 310–324.

[7] Bodei C., Degano P., Nielson F., Nielson H. R., "Static Analysis for the pi-Calculus with Applications to Security", *Information and Computation*, **168**:1 (2001), 68–92.

[8] Borgstrom J., Nestmann U., "On bisimulations for the spi calculus", *Mathematical Structures in Computer Science*, **15**:3 (2005), 487–552.

[9] Bruni A., Modersheim S., Nielson F., Nielson H. R., "Set-Pi: Set Membership Pi-Calculus", *Proceedings of the 28-th IEEE Computer Security Foundations Symposium*, 2015, 185–198.

[10] Chadha R., Cheval V., Ciobaca S., Kremer S., "Automated Verification of Equivalence Properties of Cryptographic Protocols", *ACM Transactions on Computational Logic*, **17**:4 (2016), 1–32.

[11] Chevalier Y., Kusters R., Rusinowitch M., Turuani M., "Deciding the security of protocols with Diffie-Hellman exponentiation and products in exponents", *Proceedings of the 23-rd Annual Conference on the Foundations of Software Technology and Theoretical Computer Science*, 2003, 124–135.

[12] Chevalier Y., Kusters R., Rusinowitch M., Turuani M., "An NP decision procedure for protocol insecurity with XOR", *Theoretical Computer Science*, **338**:1–3 (2005), 247–274.

[13] Chevalier Y., Kusters R., Rusinowitch M., Turuani M., "Deciding the security of protocols with commuting public key encryption", *Electronic Notes in Theoretical Computer Science*, **125**:1 (2005), 55–66.

[14] Chevalier Y., Kusters R., Rusinowitch M., Turuani M., "Complexity results for security protocols with Diffie-Hellman exponentiation and commuting public key encryption", *ACM Transactions on Computational Logic*, **9**:4 (2008), 1–52.

[15] Chretien R., Cortier V., Delaune S., "Decidability of trace equivalence for protocols with nonces", *Proceedings of the 28-th IEEE Computer Security Foundations Symposium*, 2015, 170–184.

[16] Cortier V., Delaune S., "A method for proving observational equivalence", *Proceedings of the 2009 22nd IEEE Computer Security Foundations Symposium*, 2009, 266–276.

[17] Curti M., Degano P., Priami C., Balardi C. T., "Modelling biochemical pathways through enhanced pi-calculus", *Theoretical Computer Science*, **325**:1 (2004), 111–140.

[18] Delaune S., Ryan M., Smyth B., "Automatic verification of privacy properties in the applied pi calculus", *Trust Management II*, IFIP International Federation for Information Processing, **263**, Springer, Boston, 2008, 263–278.

[19] Dolev D., Yao A., "On the security of public key protocols", *IEEE Transactoions on Information Theory*, **29**:2 (1983), 198–208.

[20] Durante L., Sisto R., Valenzano A., "Automatic Testing Equivalence Verification of Spi Calculus Specifications", *ACM Transactions on Software Engineering and Methodology*, **12**:2 (2003), 222–284.

[21] Durgin N. A., Lincoln P., Mitchell J. C., "Multiset rewriting and the complexity of bounded security protocols", *Journal of Computer Security*, **12**:2 (2004), 247–311.

[22] Godskesen J. C., "Formal Verification of the ARAN Protocol Using the Applied Pi-calculus", *Proceedings of the Sixth International IFIP WG 1.7 Workshop on Issues in the Theory of Security*, 2006, 99–113.

[23] Huima A., "Efficient infinite state analysis of security protocols", *Proceedings of the Workshop on Formal Methods and Security Protocols*, 1999.

[24] Liang Z., Verma R. M., "Correcting and Improving the NP Proof for Cryptographic Protocol Insecurity", *Proceedings of the 5-th International Conference on Information Systems Security*, 2009, 101–116.

[25] Milner R., Parrow J., Walker D., "A calculus of mobile processes, I and II", *Information and Computation*, **100**:1 (1992), 1–40 and 41–77.

[26] Milner R., "Functions as Processes", *Mathematical Structures in Computer Science*, **2** (1992), 119–141.

[27] Milner R., *Communicating and mobile systems — the Pi-calculus*, MIT Press, 1999.

[28] Regev A., "Representation and Simulation of Biochemical Processes Using the pi-Calculus Process Algebra", *Proceedings of the 6-th Pacific Symposium on Biocomputing*, 2001, 459–470.

[29] Rusinowitch M., Turuani M., "Protocol Insecurity with Finite Number of Sessions is NP-complete", *Theoretical Computer Science*, **299**:1–3 (2003), 451–475.

[30] Smith H., Fingar P., *Business Process Management: The Third Wave*, Meghan-Kiffer Press Tampa, 2003.

[31] Tiplea F. L., Enea C., Birjoveanu C. V., "Decidability and complexity results for security protocols", *Verification of Infinite-State Systems with Applications to Security*, IOS Press, Amsterdam, 2006, 185–211.

[32] Tiu A., Dawson J., "Automating Open Bisimulation Checking for the Spi Calculus", *Proceedings of the 23rd IEEE Computer Security Foundations Symposium*, 2010, 307–321.

[33] Walker D., "Objects in the π-calculus", *Information and Computation*, **116**:4 (1995), 253–271.

Abbas M. M., Zakharov V.A., "Even Simple Processes of π-calculus are Hard for Analysis", *Modeling and Analysis of Information Systems*, **25**:6 (2018), 589–606.

DOI: 10.18255/1818-1015-2018-6-589-606

Abstract. Mathematical models of distributed computations, based on the calculus of mobile processes (π-calculus) are widely used for checking the information security properties of cryptographic protocols. Since π-calculus is Turing-complete, this problem is undecidable in general case. Therefore, the study is carried out only for some special classes of π-calculus processes with restricted computational capabilities, for example, for non-recursive processes, in which all runs have a bounded length, for processes with a bounded number of parallel components, etc. However, even in these cases, the proposed checking procedures are time consuming. We assume that this is due to the very nature of the π -calculus processes. The goal of this paper is to show that even for the weakest model of passive adversary and for relatively simple protocols that use only the basic π-calculus operations, the task of checking the information security properties of these protocols is co-NP-complete.

Keywords: π-calculus, protocol, security, passive adversary, verification, complexity, NP-completeness

On the authors:
Marat M. Abbas, orcid.org/0000-0001-8019-7090, student,
Lomonosov Moscow State University,
GSP-1, Leninskie Gory, Moscow, 119991, Russia, e-mail: unlock96@gmail.com

Vladimir A. Zakharov, orcid.org/0000-0002-3794-9565, Dr. of Science, professor,
National Research University Higher School of Economics (HSE),
20 Myasnitskaya st., Moscow, 101000 Russia,
Institute for system programming of the Russian Academy of Science (ISP RAS),
25 Alexander Solzhenitsyn st., Moscow, 109004 Russia,
e-mail: zakh@cs.msu.ru

Acknowledgments:
[1] This work was supported by the Russian Foundation for Basic Research, Grant N 18-01-00854
[2] This work was supported by the Russian Foundation for Basic Research, Grant N 16-01-00714