DOI: 10.15514/ISPRAS-2021-33(5)-6

Оптимизация программ на ProVerif для криптографических протоколов выработки общего ключа

[1,2] *Е.М. Винарский, ORCID:* 0000-0002-7328-0942 *<vinarskii@ispras.ru>*
[1] *А.В. Демаков, ORCID: 0000-0001-6573-7925 <demakov@ispras.ru>*
[1] *Институт системного программирования им. В.П. Иванникова РАН, 109004, Россия, г. Москва, ул. А. Солженицына, д. 25*
[2] *Национальный исследовательский университет «Высшая школа экономики», 109028, г. Москва, Покровский бульвар, д. 11*

Аннотация. Криптографические протоколы используются для установления безопасного соединения между “честными” агентами, которые общаются строго в соответствии с правилами протокола. Для того, чтобы убедиться, что спроектированный криптографический протокол является криптографически стойким, обычно используются различные программные инструменты. Однако, адекватная спецификация криптографического протокола обычно представляется в виде набора требований к последовательностям передаваемых сообщений, включая формат таких сообщений. Выполнение всех этих требований приводит к тому, что формальная спецификация для реального криптографического протокола становится громоздкой, в следствии чего её трудно анализировать формальными методами. Одним из таких стремительно развивающихся инструментов для формальной верификации криптографических протоколов является ProVerif. Отличительной особенностью инструмента ProVerif является то, что при больших протоколах он часто не справляется с их анализом, т.е. не может ни доказать безопасность протокола, ни опровергнуть её. В таких случаях прибегают либо к апроксимации задачи, либо к эквивалентным преобразованиям модели программы на языке ProVerif, упрощающих ProVerif-модель. В этой статье мы предлагаем способ для упрощения ProVerif-спецификаций для AKE-протоколов, использующих схему шифрования Эль-Гамаля. А именно, мы предлагаем эквивалентные преобразования, позволяющие построить ProVerif-спецификацию, которая упрощает анализ спецификации для инструмента ProVerif. Экспериментальные результаты для криптопротоколов Нидхема-Шрёдера и Yahalom показывают, что такой подход может быть перспективным для автоматической проверки реальных протоколов.

Ключевые слова: криптографические протоколы; эквивалентные преобразования; ProVerif

Для цитирования: Винарский Е.М., Демаков А.В. Оптимизация программ на ProVerif для криптографических протоколов выработки общего ключа. Труды ИСП РАН, том 33, вып. 5, 2021 г., стр. 105-116. DOI: 10.15514/ISPRAS–2021–33(5)–6

Благодарности: Исследование поддержано грантом Минобрнауки РФ №075-15-2020-788

Optimization of ProVerif programs for AKE-protocols

[1,2] *E.M. Vinarskii, ORCID: 0000-0002-7328-0942 <vinaskii@ispras.ru>*
[1] *A.V. Demakov, ORCID: 0000-0001-6573-7925 <demakov@ispras.ru>*
[1] *Ivannikov Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia*
[2] *National Research University Higher School of Economics, 11, Pokrovsky Boulevard, Moscow, 109028, Russia*

Abstract. Cryptographic protocols are used to establish a secure connection between “honest” agents who communicate strictly in accordance with the rules of the protocol. In order to make sure that the designed cryptographic protocol is cryptographically strong, various software tools are usually used. However, an adequate specification of a cryptographic protocol is usually presented as a set of requirements for the sequences of transmitted messages, including the format of such messages. The fulfillment of all these requirements leads to the fact that the formal specification for a real cryptographic protocol becomes cumbersome, as a result of which it is difficult to analyze it by formal methods. One of such rapidly developing tools for formal verification of cryptographic protocols is ProVerif. A distinctive feature of the ProVerif tool is that with large protocols, it often fails to analyze them, i.e. it can neither prove the security of the protocol nor refute it. In such cases, they resort either to the approximation of the problem, or to equivalent transformations of the program model in the ProVerif language, simplifying the ProVerif model. In this article, we propose a way to simplify the ProVerif specifications for AKE protocols using the El Gamal encryption scheme. Namely, we suggest equivalent transformations that allow us to construct a ProVerif specification that simplifies the analysis of the specification for the ProVerif tool. Experimental results for the Needham-Schroeder and Yahalom cryptoprotocols show that such an approach can be promising for automatic verification of real protocols.

Keywords: cryptographic protocols; equivalent transformations; ProVerif

For citation: Vinarskii E.M., Demakov A.V. Optimization of ProVerif programs for AKE-protocols. Trudy ISP RAN/Proc. ISP RAS, vol. 33, issue 5, 2021, pp. 105-116 (in Russian). DOI: 10.15514/ISPRAS–2021–33(5)–6

Acknowledgements. The research is supported by the grant of the Ministry of Education and Science of the Russian Federation No. 075-15-2020-788.

1. Введение

Криптографические протоколы (КП) широко используются в современных сетях. Они гарантируют аутентификацию участников, секретность сеансовых ключей и т.д. Инструмент для формальной верификации криптопротоколов ProVerif широко используется для проверки свойств безопасности для практически важных протоколов таких, как TLS 1.3, ARINC823 и др. [1, 2, 3]. Поскольку для криптографических протоколов необходимо выполнить множество требований, их адекватные формальные модели очень громоздки, в следствие чего трудно провести их формальную верификацию. Однако, если использовать специфические особенности инструмента ProVerif и проводить анализ только относительно заранее определённых свойств безопасности (например, только секретности общих сессионных ключей), то громоздкость протокола и, соответственно, сложность анализа может быть уменьшена. В частности, для инструмента ProVerif такое упрощение может быть проведено при помощи эквивалентных преобразований (ЭП), то есть таких преобразований, которые не уменьшают адекватность модели, но при этом упрощают анализ криптопротокола.

Одним из основных криптографических примитивов является функция шифрования. Однако, чтобы построить адекватную модель, описывающую операцию шифрования/расшифровния, нам необходимо добавить в ProVerif уравнение, связывающее эти две операции, пример такого уравнения изображён на рис. 1 и 2. Такие уравнения могут сильно усложнить ProVerif-модель, и, следовательно, проверку свойства безопасности для построенной ProVerif-модели.

В этой статье мы предлагаем эквивалентные преобразования, упрощающие ProVerif-модель с операциями шифрования/расшифрования. Мы используем следующие шаги для оптимизации представления криптографического протокола на языке ProVerif, чтобы упростить проверку свойств безопасности.

В настоящее время в формальной верификации криптографических протоколов господствуют две основных модели.

- **Символьная модель.** Криптографические примитивы рассматриваются как идеальные черные ящики, моделируемые функциональными символами в алгебре термов. Все вычисления проводятся в фиксированной теории, образованной функциональными символами и правилами редукции (уравнения) над ними. Сообщения – это термы на этих примитивах. Противник может проводить вычисления, используя только в рамках фиксированной теории. Лучшими символьными солверами на текущий момент являются Tamarin-prover [4, 5, 6] и ProVerif [7, 8, 9].
- **Модель доказуемой стойкости.** В данной модели сообщения – это битовые строки, криптографические примитивы – функции от битовых строк, а противник - любая вероятностная машина Тьюринга. Стойкость протокола выражается через стойкости его примитивов. Самым успешным инструментом на текущий момент является CryptoVerif [10].

В этой статье мы работаем с инструментом ProVerif, который реализует символьный подход к проверке КП. ProVerif принимает на вход модель протокола безопасности, определяя действия честных агентов и спецификацию желаемых свойств протокола. Для описания честных агентов протокола используется алгебра процессов. ProVerif автоматически переводит процессы в систему дизъюнктов Хорна и использует алгоритм, основанный на свободном выборе дизъюнкта из доступного множества, чтобы определить, является ли утверждение верным в заданной системе правил вывода (теории). Если такого логического вывода не существует, то желаемое свойство безопасности считается доказанным. Если вывод существует, то рассматриваемое свойство может быть не стойко. Полной гарантии нестойкости быть не может, так как дизъюнкты Хорна могут применяться неограниченное число раз, что является аппроксимацией криптопротокола, в следствие чего логический вывод может соответствовать ложной атаке [7, 8].

Работы, связанные с оптимизацией представления моделей КП для программных инструментов верификации достаточно развита. Авторы в статье [5] предлагают 2 подхода к упрощению доказательства лемм для модели КП на Tamarin Prover – (i) использование source lemm, которые позволяют ограничить размеры модели, в которой ищутся решения (ii) использование оракулов, помогающих подсказать путь, следуя которому можно доказать/опровергнуть лемму. Мы предлагаем использовать стандартный математический способ (эквивалентные преобразования) для упрощения модели. В статье [7] разбирается метод представления КП дизъюнктами Хорна (правила вывода), получившиеся правила вывода дополняются правилами вывода, определёнными для противника, получившиеся правила вывода формируют теорию. Угрозой считается вывод терма, сигнализирующего, что противник знает секрет. При оптимизации ProVerif-модели для КП мы прежде всего упрощали систему правил вывода. Были сделаны следующие шаги, чтобы оптимизировать ProVerif-модель для КП.

1. Ограничиваем свойства безопасности свойством секретности сессионного ключа.
2. Вводим понятие ProVerif-моделей, эквивалентных относительно формул криптопротоколов.
3. Предлагаем эквивалентные преобразования (ЭП) для функции шифрования и доказываем, что получившийся в результате таких преобразований криптопротокол остаётся эквивалентным исходному относительно свойства секретности сессионного ключа.

4. Проводим эксперименты для протоколов Нидхема-Шрёдера и Yahalom, которые показывают, что предложенные эквивалентные преобразования, действительно, упрощают анализ криптопротокола.

Структура статьи выглядит следующим образом. В разд. 2 представлены основные определения и показано, как ProVerif преобразует криптопротокол в систему дизъюнктов Хорна. Разд. 3 посвящена описанию эквивалентных преобразований. В разд. 4 представлены экспериментальные результаты. В разд. 4 содержится заключение статьи и описаны некоторые направления будущей работы.

2. Представление криптографического протокола на ProVerif и дизъюнкты Хорна

В этой секции мы кратко опишем синтаксические конструкции, включённые в язык ProVerif. На рис. 1 [7] изображена грамматика, определяющая правила построения термов и процессов на языке ProVerif; полное описание можно найти в [7].

$M, N ::=$	terms
x, y, z	variable
a, b, c, k	name
$f(M_1, \ldots, M_n)$	constructor application
$P, Q ::=$	processes
$\overline{M}\langle N\rangle.P$	output
$M(x).P$	input
0	nil
$P \mid Q$	parallel composition
$!P$	replication
$(\nu a)P$	restriction
$let\ x = g(M_1, \ldots, M_n)\ in\ P\ else\ Q$	destructor application
$if\ M = N\ then\ P\ else\ Q$	conditional
$\texttt{event}(M).P$	event

Рис. 1. Синтаксис ProVerif

Термы в ProVerif могут состоять из переменных, идентификаторов, конструкторов и деструкторов. Конструкторы используются для построения термов, деструкторы – частичные функции (определены не над всеми входными данными). Например, деструктор $let\ x = g(M_1, \ldots, M_n)\ in\ P\ else\ Q$ пытается вычислить $g(M_1, \ldots, M_n)$; если это удается, то в переменную x записывается результат вычисления g и выполняется процесс P, иначе выполняется процесс Q. Действие $(\nu a)P$ обозначает, что процесс P обладает случайным секретным значением a.

2.1 Представление шифрования в ProVerif

Мы рассматриваем оптимизацию криптопротоколов, содержащих процедуру симметричного шифрования. При моделировании схемы шифрования в ProVerif обычно используется связка конструктора $Encrypt$ с деструктором $Decrypt$. На рис. 2 и 3 изображено представление на ProVerif схем шифрования.

```
fun Encrypt(bitstring,bitstring):bitstring.
  reduc forall mess:bitstring,key:bitstring
    Decrypt(Encrypt(mess,key),key) = mess
```

Рис. 2. Стандартная функция симметричного шифрования
Fig. 2. Standard symmetric encryption function

```
fun Encrypt(bitstring,bitstring,bitstring):bitstring.
  reduc forall mess:bitstring,key:bitstring,iv:bitstring
    Decrypt(Encrypt(mess,key,iv),key,iv) = mess
```

Рис. 3. Функция симметричного шифрования с вектором инициализации
Fig. *3. Function of symmetric encryption with an initialization vector*

Схему шифрования, изображённую на рис. 2 мы будем называть *стандартной*. Для такой схемы ProVerif порождает следующие дизъюнкты Хорна (ДХ):

- $attacker(mess) \wedge attacker(key) \Rightarrow attacker\big(Encrypt(mess, key)\big)$;
- $attacker\big(Encrypt(mess, key)\big) \wedge attacker(key) \Rightarrow attacker(mess)$.

Схему шифрования, изображённую на рис. 3 мы будем называть *шифрованием с инициализирующим вектором*. Такая схема порождает следующие дизъюнкты Хорна

- $attacker(mess) \wedge attacker(key) \wedge attacker(iv) \Rightarrow attacker\big(Encrypt(mess, key, iv)\big)$;
- $attacker\big(Encrypt(mess, key, iv)\big) \wedge attacker(key) \wedge attacker(key) \Rightarrow attacker(mess)$.

2.2 Схема шифрования Эль-Гамаля

В статье мы предполагаем, что вычисления происходят в группе G, где g – образующий элемент в G [11]. На рис. 4 представлено уравнение $(g^x)^y = (g^y)^x$ ProVerif для группы G. Схема шифрования Эль-Гамаля [11] предлагает способ передать ключ симметричного шифрования другой стороне в зашифрованном виде. Опишем, как схема Эль-Гамаля может быть представлена на языке ProVerif. Предположим, что криптосистема состоит из клиента $Clnt(x, g^x)$ и сервера $Serv(y, g^y)$, где x (y) – долговременный закрытый ключ клиента (сервера) и g^x (g^y) долговременный открытый ключ клиента (сервера). Пусть k – симметричный ключ, сгенерированный на стороне клиента, который необходимо передать серверу защищённым способом. Тогда клиент генерирует случайный $a \in_U G$, вычисляет $s = k \cdot (g^y)^a$ и отправляет серверу сообщение

$$Clnt \overset{(g^a, s)}{\rightarrow} Serv.$$

Сервер принимает s и вычисляет $k = s \cdot (g^a)^{-y} = k \cdot g^{ya} \cdot g^{-ya} = k$. Таким образом сервер получает ключ симметричного шифрования k, отправленный клиентом.

```
equation forall a1:bitstring,a2:bitstring;
  Exp(Exp(g, a1), a2) = Exp(Exp(g, a2), a1).
```

Рис. 4. Уравнение на языке ProVerif
Fig. 4. Equation in ProVerif language

```
fun ModMult(bitstring,bitstring,bitstring):bitstring.
  reduc forall a0:bitstring,a1:bitstring,a2:bitstring;
    ModDiv0(ModMult(a0,a1,a2),a1,a2) = a0.
```

Рис. 5. Уравнение на языке ProVerif
Fig. 5. Equation in ProVerif language

Схема шифрования Эль-Гамаля на языке ProVerif может быть реализована с помощью конструктора $ModMult$ и деструктора $ModDiv0$ (рис. 5). Такая схема представляется следующими ДХ, где $vCurveN$ – модуль эллиптической кривой:

- $attacker(key) \wedge attacker(g^y) \wedge attacker(vCurveN) \Rightarrow attacker\big(ModMult(key, g^y, vCurveN)\big)$;
- $attacker\big(ModMult(a_0, a_1, a_2)\big) \wedge attacker(a_1) \wedge attacker(a_2) \Rightarrow attacker(a_0)$.

2.3 Свойства безопасности криптографического протокола

В данном подразделе мы рассматриваем свойство секретности ключа (свойство достижимости [7]), в ProVerif свойство секретности ключа key задаётся формулой $\varphi_{sec} = attacker(key)$.

Чтобы проверить свойство секретности $\varphi_{sec} = attacker(key)$, ProVerif строит систему дизъюнктов Хорна и ищет способ вывести терм $attacker(key)$. Если противник смог вывести терм $attacker(key)$, то это следует понимать, как "противник может знать key". С другой стороны, если противник не может вывести событие $attacker(key)$, то "противник НЕ знает key". Такая аппроксимация восходит к той особенности, что каждый дизъюнкт Хорна может быть применён сколь угодно много раз, что может порождать "ложные" атаки. Тем не менее, задача проверки выводимости терма $attacker(key)$ алгоритмически разрешима (см. [7]), и предложенный алгоритм [7] показал свою эффективность при проверке практически важных протоколов (см. [4, 5, 6]).

Пусть π – криптопротокол, тогда модель π на языке ProVerif мы будем обозначать через $\mathcal{M}_\pi$, $P(\mathcal{M}_\pi)$ – система дизъюнктов Хорна (ДХ), построенных по модели $\mathcal{M}_\pi$. Обозначим через $\mathcal{N}_{pub}(\mathcal{N}_{priv})$ множество открытых (приватных) имён. Таким образом, систему ДХ, построенных по протоколу π и возможностям противника обозначим $\mathcal{R}_\pi = \mathcal{R}_{\pi,\mathcal{N}_{pub},\mathcal{N}_{priv}} = P(\mathcal{M}_\pi) \cup [\{attacker(a[]) | a \in \mathcal{N}_{pub}\} \cup \{(Rn), (Rh), (Rl), (Rs)\}$, где правила вывода $\{(Rn), (Rh), (Rl), (Rs)\}$ представлены на рис. 6 и терм $message(x, y)$ означает, что сообщение y было отправлено в канал x. Построенную таким образом $\mathcal{R}_\pi$ мы называем правилами вывода, порождаемые моделью протокола π или просто *теорией* π.

Мы говорим, что криптопротокол π *удовлетворяет свойству секретности* $\varphi_{sec} = attacker(key)$ тогда и только тогда, когда $\mathcal{R}_{\pi,\mathcal{N}_{pub},\mathcal{N}_{priv}} \vDash \varphi_{sec}$, то есть событие $attacker(key)$ НЕ выводимо в теории $\mathcal{R}_{\pi,\mathcal{N}_{pub},\mathcal{N}_{priv}}$. Дадим определение теорий криптопротоколов, эквивалентных относительно формул.

Определение. Пусть φ – формула, π_1 и π_2 – криптопротоколы, тогда мы говорим, что $\pi_1 \sim_\varphi \pi_2$ если $\mathcal{R}_{\pi_1,\mathcal{N}_{pub},\mathcal{N}_{priv}} \vDash \varphi$ тогда и только тогда, когда $\mathcal{R}_{\pi_2,\mathcal{N}_{pub},\mathcal{N}_{priv}} \vDash \varphi$.

$\text{attacker}(b_0[x])$	(Rn)
For each public constructor f of arity n, $\text{attacker}(x_1) \wedge \ldots \wedge \text{attacker}(x_n) \Rightarrow \text{attacker}(f(x_1, \ldots, x_n))$	(Rf)
For each public destructor g, for each rewrite rule $g(M_1, \ldots, M_n) \rightarrow M$ in $\text{def}(g)$, $\text{attacker}(M_1) \wedge \ldots \wedge \text{attacker}(M_n) \Rightarrow \text{attacker}(M)$	(Rg)
$\text{message}(x, y) \wedge \text{attacker}(x) \Rightarrow \text{attacker}(y)$	(Rl)
$\text{attacker}(x) \wedge \text{attacker}(y) \Rightarrow \text{message}(x, y)$	(Rs)

Рис. 6. Дизъюнкты Хорна для противника
Fig. 6. Horn's disjuncts for an enemy

3. Эквивалентные преобразования

В этом разделе мы описываем некоторые техники, позволяющие провести эквивалентные преобразования над ProVerif-моделью, которые позволяют упростить модель протокола π, оперирующий над открытыми именами $\mathcal{N}_{pub}$ и закрытыми именами $\mathcal{N}_{priv}$ ($\mathcal{R}_{\pi,\mathcal{N}_{pub},\mathcal{N}_{priv}}$). Как показывают наши эксперименты (разд. 4), для оптимизированной ProVerif-модели приведённые здесь эквивалентные преобразования упрощают проверку свойства секретности ключей. Для простоты изложения мы пишем $\mathcal{R}_\pi$ вместо $\mathcal{R}_{\pi,\mathcal{N}_{pub},\mathcal{N}_{priv}}$.

3.1 Оптимизация в представлении схемы шифрования

Мы докажем, что, если мы проверяем безопасность криптопротокола на ProVerif относительно свойств секретности ключа φ_{sec}, определённых в подразд. 2.3, то схема шифрования с инициализирующем значением эквивалентна стандартной схеме шифрования.

Пусть $\pi^{e_{iv}}$ – криптопротокол, использующий схему шифрования с вектором инициализации iv, тогда $\mathcal{M}_{\pi}^{e_{iv}}$ содержит конструкторы/деструкторы, изображённые на рис. 2. Обозначим соответствующую систему ДХ криптопротокола π через $P^{e_{iv}}(\mathcal{M}_{\pi})$, соответственно, теорию π обозначим через $\mathcal{R}_{\pi}^{e_{iv}}$.

Обозначим через $\mathcal{M}_{\pi}^{\overline{e}}$ модель криптоpотокола π, в которой все вхождения конструктора $Encrypt_{iv}$ (рис. 3) заменены на конструктор $Encrypt$ (рис. 2) и все вхождения деструктора $Decrypt_{iv}$ заменены на деструктор $Decrypt$. Соответствующую теорию обозначим через $\mathcal{R}_{\pi}^{\overline{e}}$.

Тогда справедлива следующее утверждение, в которой утверждается, что теории $\mathcal{R}_{\pi}^{e_{iv}}$ и $\mathcal{R}_{\pi}^{\overline{e}}$ эквивалентны относительно формулы $\varphi_{sec} = attacker(key)$.

Утверждение 1. Пусть $iv \in \mathcal{N}_{pub}$, тогда $\mathcal{R}_{\pi}^{e_{iv}} \sim_{\varphi_{sec}} \mathcal{R}_{\pi}^{\overline{e}}$, т.е. $\mathcal{R}_{\pi}^{e_{iv}} \vDash \varphi_{sec}$ тогда и только тогда, когда $\mathcal{R}_{\pi}^{\overline{e}} \vDash \varphi_{sec}$.

Доказательство.

$\Rightarrow$ Пусть $\mathcal{R}_{\pi}^{e_{iv}} \vDash \varphi_{sec}$, но при этом неверно, что $\mathcal{R}_{\pi}^{\overline{e}} \vDash \varphi_{sec}$. Тогда терм $attacker(key)$ выводим в теории $\mathcal{R}_{\pi}^{\overline{e}}$, рассмотрим такой вывод ρ (последовательное применение ДХ) в теории $\mathcal{R}_{\pi}^{\overline{e}}$, $\rho = r_1, \dots, r_n$. Такой логический вывод ρ обязательно содержит ДХ, порождаемые конструкторами $Encrypt$ и деструкторами $Decrypt$, так как иначе такой же логический вывод ρ существовал бы и в теории $\mathcal{R}_{\pi}^{e_{iv}}$.

Рассмотрим такой вывод $\rho = r_1, \dots, r_n$, и пусть r_j – применение ДХ $attacker(mess) \wedge attacker(key) \Rightarrow attacker\big(Encrypt(mess, key)\big)$. Из того, что $iv \in \mathcal{N}_{pub}$, следует, что противнику доступен терм $attacker(iv)$, следовательно, логический вывод ρ может содержать ДХ $attacker(mess) \wedge attacker(key) \wedge attacker(iv) \Rightarrow attacker\big(Encrypt(mess, key, iv)\big)$. Аналогично, если логический вывод ρ содержит применение ДХ $attacker\big(Encrypt(mess, key)\big) \wedge attacker(key) \Rightarrow attacker(mess)$, то может быть применён и ДХ $attacker\big(Encrypt(mess, key, iv)\big) \wedge attacker(iv) \wedge attacker(key) \Rightarrow attacker(mess)$.

Таким образом, логический вывод $\overline{\rho}$ существует и в теории $\mathcal{R}_{\pi}^{e_{iv}}$, то есть неверно, что $\mathcal{R}_{\pi}^{e_{iv}} \vDash \varphi_{sec}$ – противоречие.

$\Leftarrow$ пусть теперь $\mathcal{R}_{\pi}^{\overline{e}} \vDash \varphi_{sec}$, тогда факт, что $\mathcal{R}_{\pi}^{e_{iv}} \vDash \varphi_{sec}$, доказывается аналогично.

Аналогичное утверждение может быть доказано и для других схем шифрования. Таким образом, далее мы будем рассматривать только теории, в которых используются только стандартные функции шифрования.

3.2 Использование одного ключа шифрования при схеме шифрования Эль-Гамаля

В данном разделе мы предлагаем эквивалентные преобразования, которые позволяют упростить модели криптопротоколов, использующих симметричное шифрование с использованием схемы Эль-Гамаля. Пусть одна сторона криптопротокола π передаёт другой зашифрованные сообщения $mess_1$ на симметричном ключе k_1 и $mess_2$ на симметричном ключе k_2. Мы рассматриваем свойство секретности криптопротокола π: $\varphi_{sec_1} = attacker(k_1)$.

Мы покажем, что схема с двумя сессионными симметричными ключами k_1, k_2, изображённая на рис. 7, эквивалентна схеме с одним сессионным симметричным ключом k_1, изображённой

на рис. 8. Обозначим теорию, построенную по $\mathcal{M}^{k_1,k_2}(\pi)$, через $\mathcal{R}_{\pi}^{k_1,k_2}$, в то время, как построенную по $\mathcal{M}^{k_1}(\pi)$ через $\mathcal{R}_{\pi}^{k_1}$. Мы доказываем, что теории $\mathcal{R}_{\pi}^{k_1,k_2}$ и $\mathcal{R}_{\pi}^{k}$ эквивалентны относительно формулы $\varphi_{sec} = attacker(k)$.

```
processC(skC, pkC) =
...
new k1:bitstring;
new a1:bitstring;
let g_pkS_a1 = Exp(pkS, a1) in
let cms_k1 = ModMult(k1, g_a1) in
let g_a1 = Exp(vBasePoint, a1) in
let enc_mess1 = Encrypt(mess1, k1) in
out(c, (g_a1, cms_k1, enc_mess1));
...
new k2:bitstring;
new a2:bitstring;
let g_pkS_a2 = Exp(pkS, a2) in
let cms_k2 = ModMult(k2, g_a2) in
let g_a2 = Exp(vBasePoint, a2) in
let enc_mess2 = Encrypt(mess2, k2) in
out(c, (g_a2, cms_k2, enc_mess2));
...

processS(skS, pkS) =
...
in(c,(g_a1:bitstring,cms_k1:bitstring,enc_mess1:bitstring));
let k1 = ModDiv0(cms_k1,Exp(g_a1,skS)) in
let mess1 = Decrypt(enc_mess1, k1) in
...
in(c,(g_a2:bitstring,cms_k2:bitstring,enc_mess2:bitstring));
let k2 = ModDiv0(cms_k2,Exp(g_a2,skS)) in
let mess2 = Decrypt(enc_mess2, k2) in
...
```

Рис. 7. Схема протокола с двумя сессионными ключами симметричного шифрования
Fig. 7. Scheme of the protocol with two session keys of symmetric encryption

```
processC(skC, pkC) =
...
new k:bitstring;
new a:bitstring;
let g_pkS_a = Exp(pkS, a) in
let cms_k = ModMult(k, g_a) in
let g_a = Exp(vBasePoint, a) in
let enc_mess = Encrypt(mess1, k) in
out(c, (g_a, cms_k, enc_mess1));
...
new k2:bitstring;
new a2:bitstring;
let g_pkS_a2 = Exp(pkS, a2) in
let cms_k2 = ModMult(k2, g_a2) in
let enc_mess2 = Encrypt(mess2, k) in
let g_a2 = Exp(vBasePoint, a2) in
out(c, enc_mess2);
...

processS(skS, pkS) =
...
in(c,(g_a:bitstring,cms_k:bitstring,enc_mess:bitstring));
let k = ModDiv0(cms_k,Exp(g_a,skS)) in
let mess1 = Decrypt(enc_mess1, k) in
...
in(c,enc_mess2:bitstring);
let mess2 = Decrypt(enc_mess2, k) in
...
```

Рис. 8. Схема протокола с одним сессионным ключом симметричного шифрования
Fig. 8. Scheme of the protocol with one session symmetric encryption key

Утверждение 2. $\mathcal{R}_{\pi}^{k_1,k_2} \sim_{\varphi_{sec_1}} \mathcal{R}_{\pi}^{k}$.

Доказательство. Отметим, что обе теории определены над одним и тем же пространством имён: $\mathcal{N}_{pub} \cup \mathcal{N}_{priv}$.

Пусть верно $\mathcal{R}_{\pi}^{k_1} \vDash \varphi_{sec}$, но при этом неверно $\mathcal{R}_{\pi}^{k_1,k_2} \vDash \varphi_{sec_1}$. Тогда терм $attacker(k_1)$ выводим в теории $\mathcal{R}_{\pi}^{k_1,k_2}$, рассмотрим такой вывод $\rho = r_1, \dots, r_n$ (последовательное применение ДХ) в теории $\mathcal{R}_{\pi}^{k}$. ДХ, порождаемые теориями $\mathcal{R}^{k}(\pi)$ и $\mathcal{R}^{k_1,k_2}(\pi)$ отличаются тем, что теория $\mathcal{R}^{k_1,k_2}(\pi)$ содержит термы $attacker\big(Exp(g,a_2)\big)$, $attacker\Big(ModMult\big(k_2, Exp(g,a_2)\big)\Big)$, $attacker\big(Encrypt(mess2,k_2)\big)$, которых не содержит теория $\mathcal{R}^{k_1}(\pi)$.

Таким образом, так как логический вывод ρ существует в теории $\mathcal{R}^{k_1,k_2}(\pi)$, но не существует в теории $\mathcal{R}^{k_1}(\pi)$, то ρ содержит термы $attacker\big(Exp(g,a_2)\big)$, $attacker\Big(ModMult\big(k_2, Exp(g,a_2)\big)\Big)$, $attacker\big(Encrypt(mess_2,k_2)\big)$, так как иначе такой вывод существовал бы и в теории $\mathcal{R}^{k_1}(\pi)$.

При этом термы $attacker\big(Exp(g,a_2)\big)$, $attacker\Big(ModMult\big(k_2, Exp(g,a_2)\big)\Big)$ могут быть выведены противником согласно правилу вывода (Rh) с рис. 6. Аналогично, если в выводе ρ, приводящий к обладанию противником терма $attacker(k_1)$, то есть терм $attacker\big(Encrypt(mess_2,k_2)\big)$ и существует вывод $\overline{\rho} = \rho[k_1/k,\ k_2/k]$, следовательно, неверно, что $\mathcal{R}^{k_1}(\pi) \vDash attacker(k_1)$ – противоречие.

Теперь рассмотрим, как ещё может быть упрощена схема шифрования Эль-Гамаля. Пусть клиент использует симметричный ключ шифрования k_C, переданный серверу с использованием схемы Эль-Гамаля. В то время, как сервер использует симметричный ключ шифрования k_S, переданный клиенту с использованием схемы Эль-Гамаля. Мы предлагаем эквивалентные преобразования, которые позволяют перейти к модели, в которой используется только один ключ симметричного шифрования. Обозначим теорию, построенную по $\mathcal{M}^{k_C,k_S}(\pi)$ через $\mathcal{R}_{\pi}^{k_C,k_S}$, в то время, как построенную по $\mathcal{M}^{k_C}(\pi)$ через $\mathcal{R}_{\pi}^{k_C}$. Тогда справедливо утверждение 3, которое доказывается аналогично утверждению 2.

Утверждение 3. $\mathcal{R}_{\pi}^{k_C,k_S} \sim_{\varphi_{sec_C}} \mathcal{R}_{\pi}^{k_C}$.

Таким образом, мы показали, что при проверке свойства секретности сессионного ключа в протоколе со схемой шифрования Эль-Гамаля, в которой используются несколько сессионных ключей можно перейти к проверке свойства секретности сессионного ключа в протоколе, в котором используется всего один сессионный ключ. Также мы показали, что схемы шифрования с вектором инициализации эквивалентны стандартным схемам шифрования для ProVerif модели. В разд. 4 мы покажем, как использовать эти результаты для оптимизации ProVerif-кода.

4. Экспериментальные результаты

В данном разделе представлены эксперименты с протоколами Нидхема-Шрёдера и Yahalom, которые показывают эффективность предложенных эквивалентных преобразований для ProVerif-кода криптопротокола.

Мы работали с модификацией протокола Нидхема-Шрёдера, изображённой на рис. 9. В оригинале в этой схеме используется 3 сессионных ключа. Мы проверяли свойство секретности сессионного ключа $\varphi_{sec} = attacker(key)$. Мы модифицировали эту схему, сначала согласно Теореме 1 перешли к стандартной функции шифрования, а затем применили теоремы 2 и 3, чтобы перейти к эквивалентным ProVerif-моделям с меньшим числом правил вывода. Далее мы запустили Proverif, чтобы проверить 3 версии ProVerif-моделей: с тремя сессионными ключами, с двумя сессионными ключами и с одним

сессионным ключом. Все исходные коды экспериментов доступны по ссылке [12]. В качестве метрики мы использовали количество правил, которые использовал ProVerif для доказательства формулы секретности сессионного ключа $\varphi_{sec} = attacker(key)$.

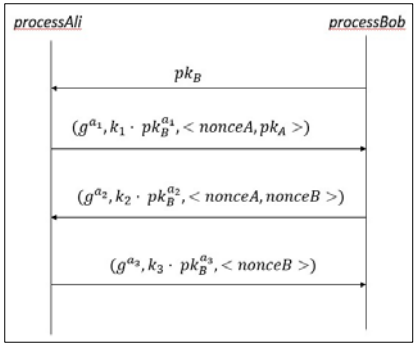

Рис. 9. Схема протокола Нидхема-Шрёдера с тремя сессионным ключами симметричного шифрования
Fig. 9. Scheme of the Needham-Schroeder protocol with three session keys of symmetric encryption

В результате, если используется 3 сессионных ключа, то ProVerif использует 45800 правил, если 2 сессионных ключей, то ProVerif использует 3000 правил и если 1 сессионных ключа, то ProVerif использует 200 правил. Таким образом, мы видим существенный выигрыш при использовании оптимизаций, предложенных в статье.

Аналогичные эксперименты мы провели с ProVerif-моделью протокола Yahalom (мхема изображена на рис. 10). Мы проверяли свойство секретности сессионного ключа $\varphi_{sec} = attacker(k)$, все исходные коды экспериментов доступны по ссылке [12]. Мы модифицировали эту схему, сначала согласно утверждениям 2 и 3, перейдя к схеме шифрования Эль-Гамаля с одним сессионным ключом. В результате, если используется 2 сессионных ключа, то ProVerif использует 2600 правил, а если 1 сессионный ключ, то 200 правил. Таким образом, снова, мы видим существенный выигрыш при использовании оптимизаций, предложенных в статье для доказательства свойства секретности сессионного ключа с использованием ProVerif.

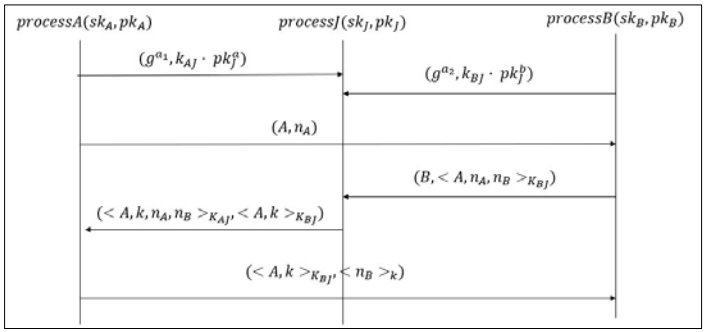

Рис. 10. Схема протокола Yahalom с двумя сессионным ключами симметричного шифрования
Fig. 10. Scheme of the Yahalom protocol with two session keys of symmetric encryption

5. Заключение

В этой статье мы предложили техники для оптимизации ProVerif-модели криптографического протокола. Мы предложили несколько преобразований ProVerif-модели и доказали, что они являются эквивалентными относительно формулы секретности сессионного ключа. Эффективность данных преобразований была проверена на реализации протоколов Нидхема-Шрёдера и Yahalom.

Далее мы планируем предложить различные техники оптимизации для других конструкций языка ProVerif и провести эксперименты с практически важными протоколами.

Список литературы / References

[1]. B. Blanchet. Symbolic and Computational Mechanized Verification of the ARINC823 Avionic Protocols. In Proc. of the 30th IEEE Computer Security Foundations Symposium (CSF'17), 2017, pp. 68-82.
[2]. K. Bhargavan, B. Blanchet, and N. Kobeissi. Verified models and reference implementations for the TLS 1.3 standard candidate. Research Report RR-9040, Inria, 2017.
[3]. K. Bhargavan, B. Blanchet, and N. Kobeissi. Verified Models and Reference Implementations for the TLS 1.3 Standard Candidate. In Proc. of the IEEE Symposium on Security and Privacy (S&P'17), pp. 483-503.
[4]. S. Meier, B. Schmidt et al. The TAMARIN prover for the symbolic analysis of security protocols. In Proc. of the 25th International Conference on Computer Aided Verification, 2013. pp. 696-701.
[5]. S. Meier. Advancing automated security protocol verification. Doctoral Thesis. ETH Zurich, 2013, 214 p.
[6]. B. Schmidt. Formal analysis of key exchange protocols and physical protocols. Doctoral Thesis. ETH Zurich, 2012, 199 p.
[7]. B. Blanchet. Modeling and Verifying Security Protocols with the Applied Pi Calculus and ProVerif. Foundations and Trends in Privacy and Security, vol. 1, no. 1-2, 2016, pp 1-135.
[8]. B. Blanchet. Automatic Verification of Correspondences for Security Protocols. Journal of Computer Security, vol. 17, issue 4, 2009, pp 363-434.
[9]. B. Blanchet. Automatic Verification of Security Protocols in the Symbolic Model: the Verifier ProVerif. Lecture Notes in Computer Science, vol. 8604, 2012, pp. 54-87.
[10]. B. Blanchet. CryptoVerif: A Computationally Sound Mechanized Prover for Cryptographic Protocols. In Proc. of the Dagstuhl Seminar on Formal Protocol Verification Applied, 2007.
[11]. T. Elgamal. A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. IEEE Transactions on Information Theory, vol. 31, no. 4, 1985, pp. 469-472, 1985.
[12]. E. Vinarskii. URL: https://github.com/vinevg1996/proverif_code_optimisation, accessed 24.10.2021.

Информация об авторах / Information about authors

Евгений Максимович ВИНАРСКИЙ – аспирант факультета ФКН ВШЭ, старший лаборант ИСП РАН. Научные интересы: конечные автоматы, верификация.

Evgenii Maksimovich VINARSKII – PhD student, Department of Computer Science, Higher School of Economics. Research interests: finite state machines, software testing and verification.

Алексей Васильевич ДЕМАКОВ – кандидат физико-математических наук, ведущий научный сотрудник. Научные интересы: графы программ, тестирование.

Alexey Vasilyevich DEMAKOV – PhD in Physics and Mathematics, Leading Researcher. Research interests: program graphs, software testing and verification.