Проверка криптографических протоколов при помощи Tamarin Prover

Евгений Винарский [1, 2]
Александр Камкин [1, 2, 3, 4]
Алексей Демаков [1]
Нина Евтушенко [1, 3]

[1] Институт системного программирования РАН им. В.П.Иванникова,

[2] Московский Государственный Университет им. М.В.Ломоносова,

[3] Высшая школа экономики

[4] Московский Физико-технический Институт

МЕЖДУНАРОДНАЯ КОНФЕРЕНЦИЯ “ИВАННИКОВСКИЕ ЧТЕНИЯ”
ОРЁЛ, 25-26 сентября, 2020

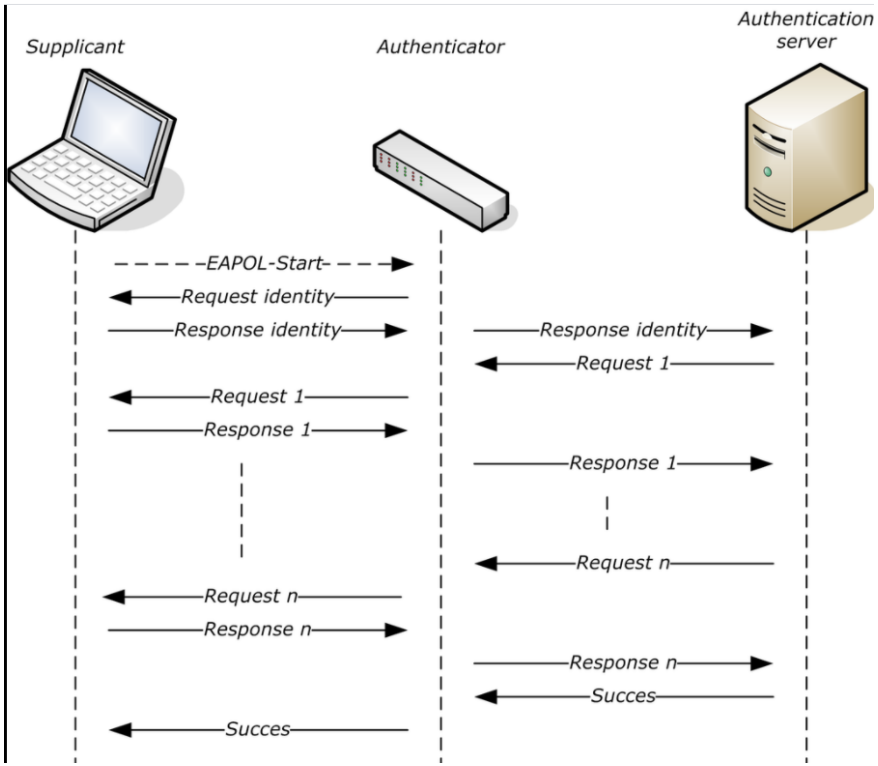
Содержание

Возможные атаки на криптографические протоколы

- Атаки на архитектуру (криптографическая система не может быть надежнее использованных в ней отдельных алгоритмов шифрования)
- Атаки на конкретные реализации
  - Переполнение буферов
  - Не полностью удалённая секретная информация
  - ...
- Атаки на сетевое оборудование
- Атаки на пользователей
- Атаки с использованием побочных каналов
- ...

Для того чтобы преодолеть систему защиты, достаточно взломать любой из ее компонентов

Трасса протокола



Трасса – последовательность действий реализации протокола, с назначением конкретных значений всем символьным параметрам

- долговременным и сессионным ключам
- случайным значениям
- . . .

Трудности формальной верификации криптографического протокола

- В RFC описание на естественном языке
- Протокол допускает множество сценариев поведения (режим PSK / режим ECDHE в TLS)
- “Адекватная” формальная модель слишком сложна для анализа
- ...

Мы предлагаем анализировать не весь протокол, а каждую “трассу” по-отдельности

- Фиксируем модель противника
- Вводим отношение “похожести” для трасс относительно свойств безопасности
- Анализируем одну трассу из каждого класса “похожих” трасс

Модель противника

- **Модель атаки** Возможности противника по взаимодействию с системой
- **Ресурсы противника** Предположения о вычислительных и информационных ресурсах противника
- **Угроза** Задача противника по нарушению свойств безопасности

Уязвимости криптосистемы возникают, если неправильно выбраны

1. модель атаки
2. угроза
3. предположения о ресурсах

Обзор литературы по формальной верификации криптопротоколов при помощи Tamarin Prover

Построены модели на языке Tamarin Prover для реальных криптопротоколов

- S. Meier, B. Schmidt, C. Cremers, and D. A. Basin, “The TAMARIN prover for the symbolic analysis of security protocols”
- S. Dunki, “Modelling and Analysis of Web Applications in Tamarin”

Предложены автоматические процедуры трансляции из описания протокола в язык Tamarin Prover

- S. Meier “Advancing automated security protocol verification”
- B. Schmidt “Formal analysis of key exchange protocols and physical protocols”

Мы исследуем подход, основанный на рассмотрении не всего протокола, а семейства его трасс

Наш подход

1. На основе RFC описания криптопротокола мы строим *исполнимую спецификацию*, которая после исполнения выдаёт конкретную трассу
2. Фиксируем модель противника
3. Определяем отношение “похожести” трасс криптопротокола относительно свойства безопасности
4. Для одной трассы tr из каждого класса “похожести” строим формальную модель в виде системы переписывания правил $\mathcal{M}(tr)$
5. Описываем свойство безопасности трассы формулой первого порядка φ
6. Проверяем выполнимость формулы φ для системы $\mathcal{M}(tr)$ $(\mathcal{M}(tr) \models \varphi)$

Tamarin Prover: символьная верификация криптопротокола

- Σ – сигнатура (переменные, функциональные символы, предикаты, утверждения)
- E – уравнения, связывающие функциональные символы
- P – система переписывания правил (протокол)
- Dependency Graph (dg) – граф зависимости, который строится по Σ, E и P
- $paths(dg)$ – множество путей, порождаемых в графе зависимости dg
- Формула φ

Проверяем выполнимость формулы φ на всех путях множества $paths(dg)$ ($paths(dg) \models \varphi$)

Tamarin Prover: состояния и переходы

Текущая конфигурация в системе Tamarin определяется совокупностью состояний агентов

Состояние агента определяется выполненными шагами протокола

- Ключевая пара сгенерирована
- Сообщение отправлено
- ...

Переход – изменение состояния системы в результате выполнения некоторого (возможно пустого) действия

- Соединение установлено
- Общий сессионный ключ выработан
- ...

Состояние, в котором никакой агент ещё не осуществил ни одного шага, объявляется *начальным состоянием*

Tamarin Prover: правила

- **Left part**: правило может быть применено к состоянию, в котором верны соответствующие утверждения
- **Action part**: переход помечается соответствующими действиями *Act1(~n)* и *Act2(x)*
- **Right part**: после применения правила будет переход в состояние, в котором верен набор соответствующих утверждений

Пусть верны утверждения *Pre(x)* и *Fr(~n)*
Тогда, применяя правило *fictitious*, переходим в состояние с утверждением *Out(<x, ~n>)*

```
rule fictitious:
   [ Pre(x), Fr(~n) ]
 --[ Act1(~n), Act2(x) ]-->
   [ Out(<x,~n>) ]
```

Tamarin Prover: леммы

Если

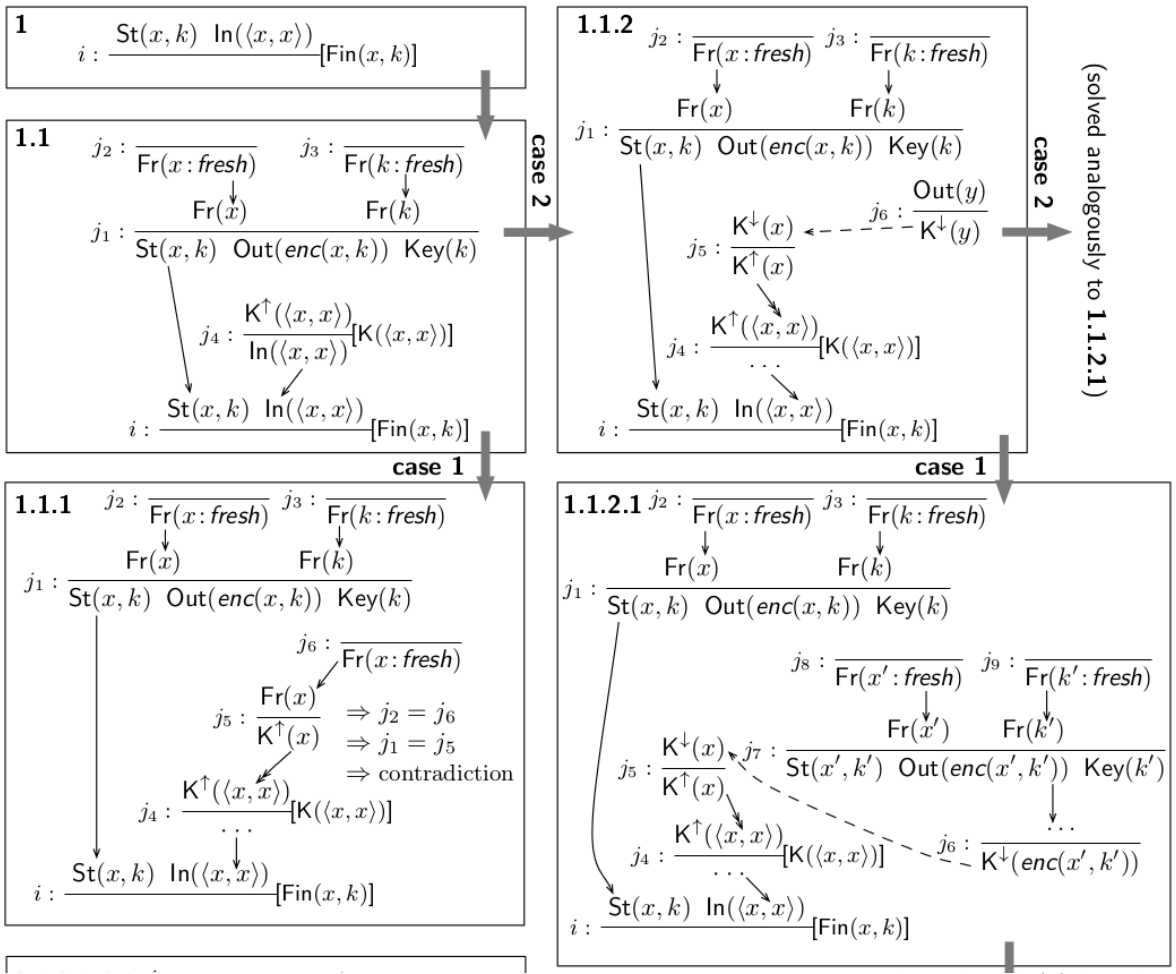
$$\begin{cases} \text{сессионный ключ } k \text{ установлен на стороне клиента } \$C \\ \text{сессионный ключ } k \text{ установлен на стороне сервера } \$S \end{cases}$$

тогда противник A не знает ключа k

```
lemma EAP_code_secrecy:
  all-traces
  " All eapcode #j_1 #j_2.
  (
    EAPcode('Server', eapcode)@ #j_1 &
    EAPcode('Client', eapcode)@ #j_2
  )
  ==>
  (
    not
      (
        Ex #l_1. K(eapcode)@ #l_1
      )
  )
  "
```

Tamarin Prover: пример построения системы ограничений

$$P = \{\ [\mathsf{Fr}(x),\mathsf{Fr}(k)]\!-\![]\!\rightarrow\![\mathsf{St}(x,k),\mathsf{Out}(\mathsf{enc}(x,k)),\mathsf{Key}(k)]$$
$$,\ [\mathsf{St}(x,k),\mathsf{In}(\langle x,x\rangle)]\!-\![\ \mathsf{Fin}(x,k)\ ]\!\rightarrow\![]$$
$$,\ [\mathsf{Key}(k)]\!-\![\ \mathsf{Rev}(k)\ ]\!\rightarrow\![\mathsf{Out}(k)]\ \}\ .$$



Tamarin Prover: эвристики

Задача проверки формулы логики первого порядка сводится к задаче нахождения пути, опровергающего систему ограничений

$\Downarrow$

Даже проверка формулы для конкретной трассы протокола – алгоритмически НЕразрешимая задача

Для автоматической верификации трассы

- Вводим отношение “похожести” для трасс, позволяющее сократить перебор пользовательских функций в трассе
- Дополняем трассу специальными леммами, сокращающими перебор путей в системе ограничений
- Применяем оракулы – скрипты, “подсказывающие” противнику правильное направление поиска атаки

Модель противника

Согласно Tamarin Prover, противник обладает следующими свойствами

1. противник активный
2. ему доступно правило

$$[K(x_1), \ldots, K(x_k)] \rightarrow K(f(x_1, \ldots, x_k)),$$

   где f – любая функция, определённая пользователем
3. может выполнять следующее упрощение: если $f()$ и $g()$ – взаимно обратные функции, то $K(f(g(x))) = K(x)$

“Похожие” трассы

Определение 1

Пусть

- $F_1 = \{f_1, \ldots, f_n\}$ *и* $F_2 = \{f^1, \ldots, f^n\}$ *множества необратимых функций,* $F_1 \cap F_2 = \emptyset$
- *Трасса* tr_1*, включающая функции из множества* F_1
- *Трасса* tr_2 *получена из трассы* tr_1 *заменой функций из множества* F_1 *на функции из множества* F_2

Трассы tr_1 *и* tr_2 *называются изоморфными*

Определение 2

Трассы tr_1 *и* tr_2 *– “похожие” трассы относительно формулы* φ*, если* $\mathcal{M}(tr_1) \models \varphi \iff \mathcal{M}(tr_2) \models \varphi$

“Похожие” трассы (2)

Теорема 1

Если

- *трассы* tr_1 *и* tr_2 *изоморфны*
- φ *– формула логики первого порядка*

то трассы tr_1 *и* tr_2 *– “похожие” трассы относительно формулы* φ

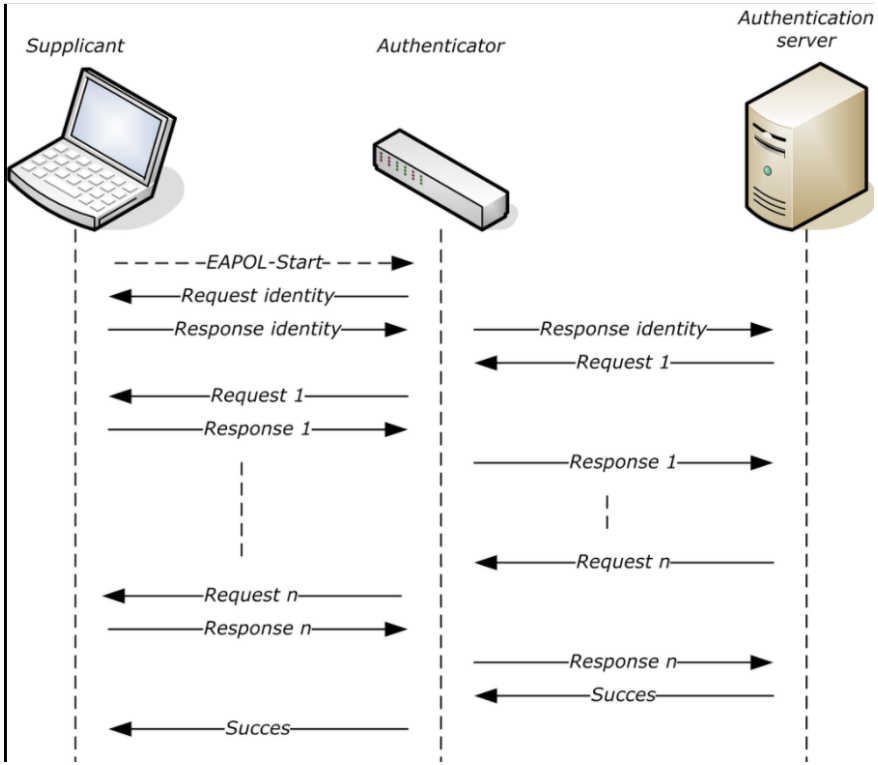
Теорема 2

Пусть tr_1 *и* tr_2 *– трассы и*

- $f_1, f_2, \ldots, f_m$ *– унарные необратимые функции такие, что* $Arg(f_i) \cap Arg(f_j) = \emptyset$, $1 \leq i < j \leq m$
- h *– унарная необратимая функция*
- φ *– формула логики первого порядка*

Если tr_2 *получена из* tr_1 *заменой функций* $f_1, f_2, \ldots, f_m$ *на функцию* h, *то трассы* tr_1 *и* tr_2 *– “похожие” трассы относительно формулы* φ

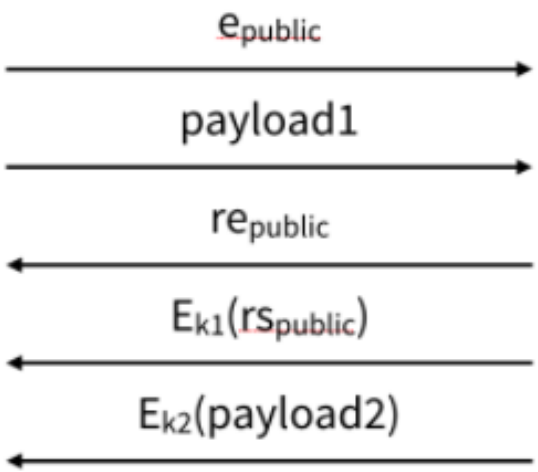
Экспериментальные результаты: *EAP-MD5* протокол




```
lemma EAP_vsharedSecret:
  all-traces
  " All vsharedSecret msk_secret #i0 #i1 #i2
      #j0 #j1.
  (
    MSK('Client', msk_secret)@ #i0 &
    PSK('Client', '0', vsharedSecret)@ #i0 &
    PSK('Client', '1', vsharedSecret)@ #i1 &
    PSK('Client', '2', vsharedSecret)@ #i2 &
    MSK('Server', msk_secret)@ #j0 &
    PSK('Server', '0', vsharedSecret)@ #j0 &
    PSK('Server', '1', vsharedSecret)@ #j1 &
    not
      (
        Ex #j.
        (
          KU(vsharedSecret)@ #j
        )
      )
  )
  ==>
  (
    not
    (
      Ex #k.
      (
        K(msk_secret)@ #k
      )
    )
  )
  "
```

Выбранная трасса протокола *EAP-MD5* является стойкой относительно данной модели противника

Экспериментальные результаты: *Noise-PSK* протокол




```
lemma Noise_vsharedSecret:
  all-traces
  " All vpskvalue mess #k1 #k2 #i1 #j1.
  (
    PSK('Client', '3', vpskvalue)@ #i1 &
    PSK('Server', '3', vpskvalue)@ #j1 &
    AEAD('Client', '4', mess)@ #k1 &
    AEAD('Server', '4', mess)@ #k2 &
    not
    (
      Ex #k_adv1.
      (
        KU(vpskvalue)@ #k_adv1
      )
    )
  )
  ==>
  (
    not
    (
      Ex #k_adv2.
        (
          K(mess)@ #k_adv2
        )
    )
  )
  "
```

Выбранная трасса протокола *Noise-PSK* является стойкой относительно данной модели противника

Заключение

1. На основе RFC построены *исполнимые спецификации* для криптопротоколов *EAP-MD5* и *Noise-PSK*
2. Предложен алгоритм трансляции трассы в систему переписывания правил для инструмента Tamarin Prover
3. Свойства безопасности криптопротоколов описаны формулами логики первого порядка, и проверка стойкости трассы криптопротокола сведена к проверке выполнимости формулы логики первого порядка
4. Введено отношение “похожести” трасс, что позволяет сократить количество проверяемых трасс
5. Получены экспериментальные результаты