Верификация криптографических протоколов с использованием Тамарин (Tamarin)

Евтушенко Н.В., Винарский Е.М.
по всем вопросам писать на vinevg2015@gmail.com

Высшая школа экономики (факультет компьютерных наук)

8 ноября 2019 г.

Содержание

1. Синтаксис системы Tamarin
2. Пример описания протокола в системе Tamarin
3. Доказательство Свойства секретности в системе Tamarin
4. Naxos протокол

Криптографичекские сообщения в системе Tamarin

Система Tamarin доступна на tamarin-prover.github.io/

- *Public constants*: идентификаторы и метки агентов (Agent)
- *Fresh constants*: закрытые ключи, nonce

Встроенные функции:

- fst(pair(x,y)) = x
- snd(pair(x,y)) = y
- $< x_1, x_2, \ldots, x_{n-1}, x_n >$
- функции и арность: $f_1/a_1, \ldots, f_n/a_n$

Типы переменных:

- $\sim x$ означает, что x – fresh (секретная информация)
- $\$x$ означает, что x – pub (открытая информация)
- $\#i$ означает, что x – temporal (вспомогательная переменная)

Встроенные теории и примитивы в системе Tamarin

- Теория **hashing**:
  - определяет функцию $h/1$
- Теория **asymmetric-encryption**:
  - определяет функции: $aenc/2$, $adec/2$, $pk/1$
  - определяет аксиомы: $adec(aenc(m, pk(sk)), sk) = m$
- Теория **symmetric-encryption**:
  - определяет функции: $senc/2$, $sdec/2$
  - определяет аксиомы: $sdec(senc(m, k), k) = m$
- Теория **diffie-hellman**:
  - определяет функции: $inv/1$
  - определяет аксиомы:

```
(x^y)^z   = x^(y*z)
x^1       = x
x*y       = y*x
(x*y)*z   = x*(y*z)
x*1       = x
x*inv(x)  = 1
```

Состояния и переходы в системе Tamarin

Текущая конфигурация в системе Tamarin определяется совокупностью состояний агентов и переходами между этими состояниями
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

Правила в системе Tamarin

- **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 $Fr(\sim n)$
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>) ]
```

Утверждение (Facts) в системе Tamarin

Встроенные утверждения в системе Tamarin:

- **In**: Получение сообщения из общего канала (может быть применён только в левой части)
- **Out**: Отправка сообщения в общий канал (может быть применён только в правой части)
- **Fr**: Генерация случайного секрета: закрытый ключ, nonce (может быть применён только в левой части)

persistent facts (!) – никогда не удаляются из состояний

Каждый участник протокола (в том числе и противник) может отправлять и получать все сообщения, находящиеся в общем канале

Архитектура с открытым ключом (Public-key infrastructure) в системе Tamarin

- *Pk* : факт, присваивающий агенту *A* открытый ключ
- *Ltk* : факт, присваивающий агенту *A* долговременный закрытый ключ

```
rule Generate_key_pair:
  [ Fr(~x) ]
  -->
  [ !Pk($A,pk(~x))
  , Out(pk(~x))
  , !Ltk($A,~x)
  ]
```

```
rule Generate_DH_key_pair:
  [ Fr(~x) ]
  -->
  [ !Pk($A,'g'^~x)
  , Out('g'^~x)
  , !Ltk($A,~x)
  ]
```

Леммы в системе Tamarin

Леммы в системе Tamarin записываются на языке логики предикатов первого порядка

- **All**: квантор всеобщности, вспомогательные переменные помечаются префиксом #
- **Ex**: квантор существования, вспомогательные переменные помечаются префиксом #
- ==>: импликация
- &: конъюнкция, |: дизъюнкция, *not*: отрицание
- $f@\,arg$: значение аргумента *arg* при действии на функцию (запись $K(k)@j$ используется для факта раскрытия противником ключа k)
- $a < b$ ($a = b$), если значение переменной a меньше значения (равно значению) переменной b
- $\#a < \#b$ ($\#a = \#b$): если значение вспомогательной переменной a меньше значения (равно значению) вспомогательной переменной b

Протокол аутентификации сервера перед клиентом в системе Tamarin

```
C -> S: aenc(k, pkS)
C <- S: h(k)
```

Протокол аутентификации сервера перед клиентом

```
rule Register_pk:
    [ Fr(~ltk) ]
  -->
    [ !Ltk($A, ~ltk), !Pk($A, pk(~ltk)) ]
```

```
rule Get_pk:
    [ !Pk(A, pubkey) ]
  -->
    [ Out(pubkey) ]
```

```
rule Reveal_ltk:
    [ !Ltk(A, ltk) ]
  --[ LtkReveal(A) ]->
    [ Out(ltk) ]
```

Протокол аутентификации сервера перед клиентом в системе Tamarin (2)

```
// Start a new thread executing the client role, choosing the server
// non-deterministically.
rule Client_1:
    [ Fr(~k)          // choose fresh key
    , !Pk($S, pkS)    // lookup public-key of server
    ]
  -->
    [ Client_1( $S, ~k )     // Store server and key for next step of thread
    , Out( aenc(~k, pkS) )   // Send the encrypted session key to the server
    ]

rule Client_2:
    [ Client_1(S, k)    // Retrieve server and session key from previous step
    , In( h(k) )        // Receive hashed session key from network
    ]
  --[ SessKeyC( S, k ) ]-> // State that the session key 'k'
    []                     // was setup with server 'S'

// A server thread answering in one-step to a session-key setup request from
// some client.
rule Serv_1:
    [ !Ltk($S, ~ltkS)                                  // lookup the private-key
    , In( request )                                    // receive a request
    ]
  --[ AnswerRequest($S, adec(request, ~ltkS)) ]-> // Explanation below
    [ Out( h(adec(request, ~ltkS)) ) ]               // Return the hash of the
                                                       // decrypted request.
```

Примеры лемм в системе Tamarin (секретность выработанного ключа)

Не может быть такого, чтобы клиент установил сессионный ключ k с сервером S, противник узнал k, и при этом долговременный закрытый ключ сервера не скомпрометирован

Примеры лемм в системе Tamarin (секретность выработанного ключа)

Не может быть такого, чтобы клиент установил сессионный ключ k с сервером S, противник узнал k, и при этом долговременный закрытый ключ сервера не скомпрометирован

$$\neg\exists S, k : (SessKeyC(S, k)\ \&\ K(k)@j\ \&\ \neg LtkReveal(S))$$

```
lemma Client_session_key_secrecy:
  " /* It cannot be that a  */
    not(
      Ex S k #i #j.
        /* client has set up a session key 'k' with a server'S' */
        SessKeyC(S, k) @ #i
        /* and the adversary knows 'k' */
      & K(k) @ #j
        /* without having performed a long-term key reveal on 'S'. */
      & not(Ex #r. LtkReveal(S) @ r)
    )
  "
```

Примеры лемм в системе Tamarin (аутентификация клиента)

Для каждого сервера S и ключа k, если с каким-нибудь клиентом C установлено соединение, то либо существует сервер, ответивший клиенту C, либо долговременный ключ сервера скомпрометирован до установления соединения

Примеры лемм в системе Tamarin (аутентификация клиента)

Для каждого сервера S и ключа k, если с каким-нибудь клиентом C установлено соединение, то либо существует сервер, ответивший клиенту C, либо долговременный ключ сервера скомпрометирован до установления соединения

$$\forall S, k : (SessKeyC(S, k) \rightarrow (AnswerRequest(S, k) \mid LtkReveal(S)))$$

```
lemma Client_auth:
  " /* For all session keys 'k' setup by clients with a server 'S' */
    ( All S k #i.  SessKeyC(S, k) @ #i
       ==>
         /* there is a server that answered the request */
       ( (Ex #a. AnswerRequest(S, k) @ a)
         /* or the adversary performed a long-term key reveal on 'S'
            before the key was setup. */
       | (Ex #r. LtkReveal(S) @ r & r < i)
       )
```

Примеры лемм в системе Tamarin (явная аутентификация клиента)

Для каждого сервера S и ключа k, если с каким-нибудь клиентом C установлено соединение, то либо существует сервер, ответивший клиенту C и при этом не существует другой пары клиент-сервер с тем же сессионным ключом, либо долговременный ключ сервера скомпрометирован до установления соединения

Примеры лемм в системе Tamarin (явная аутентификация клиента)

Для каждого сервера S и ключа k, если с каким-нибудь клиентом C установлено соединение, то либо существует сервер, ответивший клиенту C и при этом не существует другой пары клиент-сервер с тем же сессионным ключом, либо долговременный ключ сервера скомпрометирован до установления соединения

```
" /* For all session keys 'k' setup by clients with a server 'S' */
  ( All S k #i.  SessKeyC(S, k) @ #i
    ==>
      /* there is a server that answered the request */
    ( (Ex #a. AnswerRequest(S, k) @ a
        /* and there is no other client that had the same request */
        & (All #j. SessKeyC(S, k) @ #j ==> #i = #j)
    )
      /* or the adversary performed a long-term key reveal on 'S'
         before the key was setup. */
    | (Ex #r. LtkReveal(S) @ r & r < i)
    )
  )
```

Примеры лемм в системе Tamarin (корректность протокола)

Существует сервер и сессионный ключ такие, что клиент смог с ними установить соединение и при этом долговременный ключ сервера скомпрометирован

Примеры лемм в системе Tamarin (корректность протокола)

Существует сервер и сессионный ключ такие, что клиент смог с ними установить соединение и при этом долговременный ключ сервера скомпрометирован

$$\exists S, k_{CS} : (SessKeyC(S, k_{CS}) \ \& \ \neg LtkReveal(S))$$

```
lemma Client_session_key_honest_setup:
  exists-trace
  " Ex S k #i.
      SessKeyC(S, k) @ #i
    & not(Ex #r. LtkReveal(S) @ r)
  "
```

Описание протокола Ассимитричного шифрования

$$1. A \rightarrow B :< A, na >_{pk_B}$$

Архитектура шифрования с открытым ключом

```
// Public key infrastructure
rule Register_pk:
  [ Fr(~ltkX) ]
  -->
  [ !Ltk($X, ~ltkX)
  , !Pk($X, pk(~ltkX))
  , Out(pk(~ltkX))
  ]
```

```
// Compromising an agent's long-term key
rule Reveal_ltk:
  [ !Ltk($X, ltkX) ] --[ Reveal($X) ]-> [ Out(ltkX) ]
```

Описание протокола Ассимитричного шифрования (2)

```
// Role A sends first message
rule A_1_send:
  [ Fr(~na)
  , !Ltk($A, ltkA)
  , !Pk($B, pkB)
  ]
--[ Send($A, aenc(<$A, ~na>, pkB))
  , Secret(~na), Honest($A), Honest($B), Role('A')
  ]->
  [ St_A_1($A, ltkA, pkB, $B, ~na)
  , Out(aenc(<$A, ~na>, pkB))
  ]


// Role B receives first message
rule B_1_receive:
  [ !Ltk($B, ltkB)
  , In(aenc(<$A, na>,pk(ltkB)))
  ]
--[ Recv($B, aenc(<$A, na>, pk(ltkB)))
  , Secret(na), Honest($B), Honest($A), Role('B')
  ]->
  [ St_B_1($B, ltkB, $A, na)
```

Примеры лемм в системе Tamarin (секретность *nonce* при отправке сообщения агентом A)

Для всех возможных *nonce*, если A отправил секретный *nonce*, то либо не существует противника, знающего *nonce*, либо долговременный ключ честного агента B скомпрометирован

$$\forall n : ((Secret(n) \ \& \ Role('A')) \rightarrow ((\neg K(n)@j) \mid (Reveal(B) \ \& \ Honest(B))))$$

```
lemma secret_A:
  all-traces
   "All n #i. Secret(n) @i & Role('A') @i ==> (not (Ex #j. K(n)@j)) | (Ex B #j. Reveal(B)@j & Honest(B)@i)"
```

Примеры лемм в системе Tamarin (секретность *nonce* при получении сообщения агентом B)

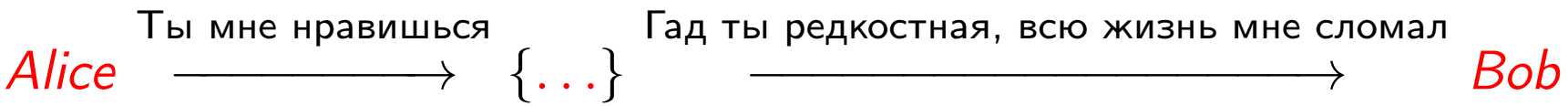
Для всех возможных *nonce*, если B получил *nonce* от A, то либо не существует противника, знающего *nonce*, либо долговременный ключ честного агента B скомпрометирован

$$\forall n : ((Secret(n)\ \&\ Role('B')) \rightarrow ((\neg K(n)@j) \mid (Reveal(B)\ \&\ Honest(B))))$$

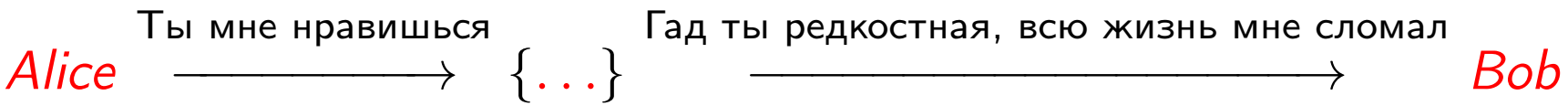
```
lemma secret_B:
  all-traces
   "All n #i. Secret(n) @i & Role('B') @i ==> (not (Ex #j. K(n)@j)) | (Ex B #j. Reveal(B)@j & Honest(B)@i)"
```

Свойство нарушается, почему?

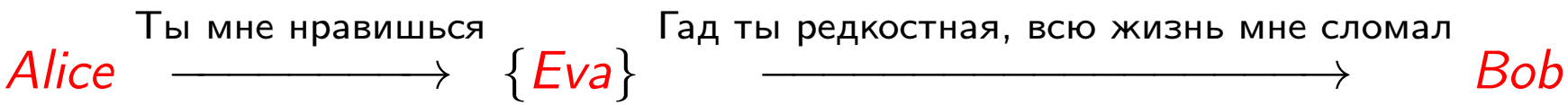
Нежелательные последствия данной уязвимости...



Нежелательные последствия данной уязвимости...



В канале противник!



Как защититься от нежелательных последствий?

$\Downarrow$

Нужна явная аутентификация (механизм электронной подписи)

Naxos протокол

$pk_{\mathcal{I}} = g^{\ell k_I}$, $pk_{\mathcal{R}} = g^{\ell k_R}$

$$\begin{array}{ccc} \mathcal{I} & & \mathcal{R} \\ \text{create fresh } esk_{\mathcal{I}} & \xrightarrow{g^{h_1(esk_{\mathcal{I}}, lk_{\mathcal{I}})}} & \text{receive } X \\ \text{receive } Y & \xleftarrow{g^{h_1(esk_{\mathcal{R}}, lk_{\mathcal{R}})}} & \text{create fresh } esk_{\mathcal{R}} \end{array}$$

$$\begin{aligned} k_{\mathcal{I}} = \; & h_2(\, Y^{lk_{\mathcal{I}}}, \quad x, \quad Y^{h_1(esk_{\mathcal{I}}, lk_{\mathcal{I}})}, \quad \mathcal{I}, \quad \mathcal{R}\,) \\ k_{\mathcal{R}} = \; & h_2(\, y, \quad X^{lk_{\mathcal{R}}}, \quad X^{h_1(esk_{\mathcal{R}}, lk_{\mathcal{R}})}, \quad \mathcal{I}, \quad \mathcal{R}\,) \\ & \text{for } y = (pk_{\mathcal{I}})^{h_1(esk_{\mathcal{R}}, lk_{\mathcal{R}})} \text{ and } x = (pk_{\mathcal{R}})^{h_1(esk_{\mathcal{I}}, lk_{\mathcal{I}})} \end{aligned}$$

$$k_{\mathcal{I}} = h_2(g^{h_1(esk_R, \ell k_R) * \ell k_I}, g^{\ell k_R * h_1(esk_I, \ell k_I)}, g^{h_1(esk_R, \ell k_R) * h_1(esk_I, \ell k_I)}, \mathcal{I}, \mathcal{R})$$
$$k_{\mathcal{R}} = h_2(g^{\ell k_I * h_1(esk_R, \ell k_R)}, g^{h_1(esk_I, \ell k_I) * \ell k_R}, g^{h_1(esk_I, \ell k_I) * h_1(esk_R, \ell k_R)}, \mathcal{I}, \mathcal{R})$$

Моделирование роли получающей стороны Naxos протокола

Каждый раз, когда $R получает сообщение X, он генерирует новое значение $\sim esk_R$, отправляет ответное сообщение и вычисляет ключ lkR

```
rule NaxosR_attempt1:
        [
          In(X),
          Fr(~eskR),
          !Ltk($R, lkR)
        ]
        -->
        [
          Out( 'g'^h1(< ~eskR, lkR >) )
        ]
```

Моделирование роли получающей стороны Naxos протокола (2)

НО! Получатель также вычисляет сессионный ключ kR

```
rule NaxosR_attempt2:
        [
          In(X),
          Fr(~eskR),
          !Ltk($R, lkR)
        ]
        --[ SessionKey($R, kR ) ]->
        [
          Out( 'g'^h1(< ~eskR, lkR >) )
        ]
```

Моделирование роли получающей стороны Naxos протокола (2)

НО! Получатель также вычисляет сессионный ключ kR

```
rule NaxosR_attempt2:
        [
          In(X),
          Fr(~eskR),
          !Ltk($R, lkR)
        ]
        --[ SessionKey($R, kR ) ]->
        [
          Out( 'g'^h1(< ~eskR, lkR >) )
        ]
```

Моделирование роли получающей стороны Naxos протокола (3)

НО! Для вычисления ключа нам нужен открытый ключ партнёра $I, который будет связан с идентификатором ~tid

```
rule NaxosR_attempt3:
  let
      exR = h1(< ~eskR, lkR >)
      hkr = 'g'^exR
      kR  = h2(< pkI^exR, X^lkR, X^exR, $I, $R >)
  in
   [
       In(X),
       Fr( ~eskR ),
       Fr( ~tid ),
       !Ltk($R, lkR),
       !Pk($I, pkI)
   ]
   --[ SessionKey( ~tid, $R, $I, kR ) ]->
   [
       Out( hkr )
   ]
```

Моделирование роли отправляющей стороны Naxos протокола

Цель отправляющей стороны: отправка сообщения и ожидание ответа

```
rule NaxosI_1:
  let exI = h1(<~eskI, ~lkI >)
      hkI = 'g'^exI
  in
   [   Fr( ~eskI ),
       Fr( ~tid ),
       !Ltk( $I, ~lkI ) ]
   -->
   [   Init_1( ~tid, $I, $R, ~lkI, ~eskI ),
       Out( hkI ) ]
```

```
rule NaxosI_2:
  let
      exI = h1(< ~eskI, ~lkI >)
      kI  = h2(< Y^~lkI, pkR^exI, Y^exI, $I, $R >)
  in
   [   Init_1( ~tid, $I, $R, ~lkI , ~eskI),
       !Pk( $R, pkR ),
       In( Y ) ]
   --[ SessionKey( ~tid, $I, $R, kI ) ]->
   []
```

Свойства секретности для протокола Naxos

Для всех сессионных ключей kI при установлении соединение верно, что противник не знает kI

```
lemma secret_kI:
    all-traces
        "All kI #keyI. ((Secret(kI) @keyI) ==> (not (Ex #j. K(kI)@j)))"
```

```
lemma secret_kR:
    all-traces
        "All kR #keyR. ((Secret(kR) @keyR) ==> (not (Ex #j. K(kR)@j)))"
```