# Analysing cryptographic protocols using Tamarin

Stéphanie Delaune

14 june 2021

Univ Rennes, CNRS, IRISA, Spicy team

Security protocol design is **critical** and **error-prone** as illustrated by many **attacks**:

SSL/TLS: FREAK, Logjam, ...

Use formal methods to improve confidence:

- prove the absence of attacks under certain assumptions; or
- identify weaknesses

Many tools already exist:

ProVerif, Tamarin, AKISS, DeepSec, AVISPA, Squirrel, ...



Security protocol design is **critical** and **error-prone** as illustrated by many **attacks**:

SSL/TLS: FREAK, Logjam, ...

Use formal methods to improve confidence:

- prove the absence of attacks under certain assumptions; or
- identify weaknesses

Many tools already exist:

ProVerif, Tamarin, AKISS, DeepSec, AVISPA, Squirrel, ...

Problem: trade-off between automation and completeness



 $\longrightarrow$  mainly developped at ETH Zurich https://tamarin-prover.github.io



- A verification tool for the symbolic model with induction, loops, mutable state
- Successfully used for many large-scale case studies: 5G AKA, TLS 1.3, EMV ...
- Security protocol model based on multiset rewriting
- Constraint-solving algorithm for analysis of **unbounded number of sessions**
- Interactive and automatic modes

## Interaction and automation

Tamarin's **interactive mode** allows the user to inspect and direct proof search

- Gives the **flexibility** required for complex case-studies
- Enables fine-tuning of models and proof strategies

Ranning Torus 1.1.	•				
TAM Tamarin prover		N			$\frac{K_{\mathrm{e}}^{\dagger}}{K_{\mathrm{exp}}^{\dagger}(f)}$
Authors: Simon Me Contributors: Cas C Observational Equi	ier, Benedikt Temers, Cedr valence Auth	Schmidt ic Staub ars: Jannik De	vier, Ralf Saxon		
Tossas was devela ABSOLUTELY NO W UCENSE.	ped at the int ARRANTY. It is	free softwar	e, and you are welcome b	. This program comes w o redistribute it according	ith g to its
More information a the Tousan webper	bout Tamarin M	and technica	i papers describing the u	iderlying theory can be l	bund on
Theory name	Time	Version	Origin		
RestExample	16:48:41	Original	(TirstExample.splby)		
Loading a new You can load a new Planame Onesa E Load new Peery Note: You can save	theory file fr	ern disk in er m fownlowding	der to work with it. the secree.		

## Interaction and automation

Tamarin's **interactive mode** allows the user to inspect and direct proof search

- Gives the **flexibility** required for complex case-studies
- Enables fine-tuning of models and proof strategies

Ranning Torms 1.1.	•				
TAM		N			$\frac{K_{e}^{\dagger}}{K_{exp}^{\dagger}(f)}$
Authors: Simon Me Contributors: Cas C Observational Equi	ier, Benecikt remers, Cedr	Schmidt Ic Staub ars: Jannik De	vier, Ralf Sanse		
Takas was develo ABSOLUTELY NO W	ped at the int ARRANTY. It is	free softwar	e, and you are welcome t	. This program comes w predistribute it according	ith g to its
More information a the Towars webpeg	bout Tamarin M.	and technica	i papers describing the u	iderlying theory can be l	bund on
Theory name	Time	Version	Origin		
<b>FirstExareple</b>	16:48:41	Original	./TirstExample.spthy		
	theory				

On the downside, Tamarin's **automatic mode** often fails (compared to, e.g., ProVerif), even on relatively **simple examples**.  $\longrightarrow$  partial deconstructions.

## Interaction and automation

Tamarin's **interactive mode** allows the user to inspect and direct proof search

- Gives the **flexibility** required for complex case-studies
- Enables fine-tuning of models and proof strategies

Ranning Torono L.L.	•				
TAM Tamarin prover		[N mode			
Authors: Simon Me Contributors: Cas C	ier, Benecikt Yerners, Ced	Schmidt ric Staub	miar Ball Same		
Tawais was develo ABSOLUTELY NO W UCENSE. More information a the Tawais webper	ped at the lay WIRANTY, IL is bout Tamarin IR.	and technics	ourity institute, ETH Zuric e, and you are welcome t al papers describing the u	h. This program comes w to redistribute it according anderlying theory can be to	ith j to its bund on
Theory name	Time	Version	Origin	1	
Bruttkareple	16:48:41	Original	./TirstExample.spthy		
Loading a new You can load a new Pitename Choses I Load new Reny	theory theory file fr	orn disk in or en	der to work with II.		

On the downside, Tamarin's **automatic mode** often fails (compared to, e.g., ProVerif), even on relatively **simple examples**.  $\longrightarrow$  partial deconstructions.



#### 25th European Symposium on Research in Computer Security

Best Paper

#### Our contribution:

automatic handling of partial deconstructions in most cases.

## High-level view of Tamarin

## Modelling part:

- protocol and adversary: multiset rewriting
  - $\longrightarrow$  a transition system which induces a set of traces
- security properties: a fragment of first-order logic
  - $\longrightarrow$  this specifies "good" traces

## Verification part – Tamarin tries to

- construct a counterexample trace, i.e. an attack; or
- provide a proof that all the traces produce by the system are good.

## Terms – messages:

- built using function symbols, e.g. aenc/2, adec/2, pk/1 ...
- interpreted modulo an equational theory.

Example:

 $aenc(\langle req, I, n \rangle, pk(ItkR))$  adec(aenc(x, pk(y), y) = x

## Terms – messages:

- built using function symbols, e.g. aenc/2, adec/2, pk/1 ...
- interpreted modulo an equational theory.

Example:

$$\operatorname{aenc}(\langle req, I, n \rangle, \operatorname{pk}(ItkR))$$
  $\operatorname{adec}(\operatorname{aenc}(x, \operatorname{pk}(y), y) = x$ 

Facts – think "sticky notes on the fridge":

- user defined facts of two kinds: linear or persistent (prefixed with !)
- some special facts: Fr(n), In(t), Out(t), !K(t)

A state of a system is a multiset of facts, and rules specify the possible moves.

Each rule has the following form:  $[I] \rightarrow [r]$  where:

- *l*, *r* are multisets of facts, and
- *a* is a multiset of annotations used for specifying properties

Each rule has the following form:  $[I] \rightarrow [r]$  where:

- *I*, *r* are multisets of facts, and
- *a* is a multiset of annotations used for specifying properties

## Some examples:

- 1.  $[!K(x_1), !K(x_2)] \rightarrow [K(aenc(x_1, x_2))] \rightarrow [!K(aenc(x_1, x_2))]$
- 2.  $[!K(x_1), !K(x_2)] \rightarrow [K(adec(x_1, x_2))] \rightarrow [!K(adec(x_1, x_2))]$
- 3.  $[Out(x)] \rightarrow [!K(x)]$
- 4.  $[!K(x)] \rightarrow [In(x)]$
- 5. []–[] $\rightarrow$ [Fr(*n*)]

Consider the following toy protocol between the initiator 2 and the responder 2:

1. 
$$(req, l, n)_{pk(R)}$$
  
2.  $(req, l, n)_{pk(R)}$   
2.  $(rep, n)_{pk(l)}$ 

Consider the following toy protocol between the initiator  $\widehat{\mathbf{Z}}$  and the responder  $\mathbb{A}$ :

1. 
$$(req, l, n)_{pk(R)}$$
  
2.  $(req, l, n)_{pk(R)}$   
2.  $(rep, n)_{pk(l)}$ 

rule Register\_pk:
 [ Fr(~ltkA) ]
 --> [ !Ltk(\$A, ~ltkA), !Pk(\$A, pk(~ltkA)), Out(pk(~ltkA)) ]

Consider the following toy protocol between the initiator  $\widehat{\mathbf{Z}}$  and the responder  $\mathbb{A}$ :

1. 
$$(req, l, n)_{pk(R)}$$
  
2.  $(req, l, n)_{pk(R)}$   
2.  $(rep, n)_{pk(l)}$ 

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

```
rule Rule_I:
[Fr(n), !Pk(R, pkR), !Ltk(I, ltkI) ]
--[SecretI(I, R, n)]-> [Out(aenc{'req', I, n}pkR)]
```

A set of protocol rules P induces a transition relation between states.

$$S \rightsquigarrow_P^a (S \setminus I) \cup r$$

where  $[I] \rightarrow [r]$  a ground instance of a rule, and  $I \subseteq S$ 

A set of protocol rules *P* induces a transition relation between states.

 $S \rightsquigarrow_P^a (S \setminus I) \cup r$ 

where  $[I] \rightarrow [r]$  a ground instance of a rule, and  $I \subseteq S$ 

Executions

 $Exec(P) = \{\{\} \rightsquigarrow_P^{a_1} \ldots \rightsquigarrow_P^{a_n} S_n \mid \forall n. Fr(n) \text{ apprears only once} \\ \text{ on rhs of rules} \}$ 

Traces

$$Traces(P) = \{ [a_1, \ldots, a_n] \mid \{\} \rightsquigarrow_P^{a_1} \ldots \rightsquigarrow_P^{a_n} S_n \in Exec(P) \}$$

## Property specification

First-order logic interpreted over traces  $a_1, \ldots, a_n$ :

- message equality:  $t_1 = t_2$
- action at a particular timepoint: A@#i
- timepoint ordering: #i < #j</p>
- timepoint equality: #i = #j

First-order logic interpreted over traces  $a_1, \ldots, a_n$ :

- message equality:  $t_1 = t_2$
- action at a particular timepoint: A@#i
- timepoint ordering: #i < #j</p>
- timepoint equality: #i = #j

```
Example: Secrecy for the nonce n.
```

## A backward search algorithm starting form the conclusion.

Running TAMARIN 1.7.0	Inc
Proof scripts	Visualization display
theory runningV1 begin Message theory Multiset rewriting rules (5) Raw sources (8 cases, 6 partial deconstructions left)	Applicable Proof Methods: Goals sorted according to the 'sm         1. solve( IPk( B, pkR ) ▶₁ #i ) // nr. 3 (from rule Rule_I)         2. solve( ILtk(\$I, Itkl ) ▶₂ #i ) // nr. 4 (from rule Rule_I)         3. solve( IKU( ~n ) @ #vk ) // nr. 6
<pre>Retined sources (8 cases, 6 partial deconstructions left) lemma nonce_secrecy: all-traces "~(3 A B s #i #j,</pre>	a. autoprove (A. for all solutions) b. autoprove (B. for all solutions) with proof-depth bound 5 Constraint system Image: system           Image: system           Image

#i : isend[K( ~n )]

A backward search algorithm that relies on some precomputations: the sources. Sources are a combination of rules yiedling a particular fact as part of the result.

Example:

```
Sources of "!Ltk( t.1, t.2 ) \triangleright_0 #i" (1 cases)
```

Source 1 of 1 / named "Register\_pk"



Computation of raw sources can stop in an incomplete stage (**partial deconstruction**) if TAMARIN lacks sufficient information about the origins of some fact.

## Algorithm intuition (3/3)

```
Running TAMARIN 1.7.0
                                                                                                                                     Index
                                                                                                                                            Download
                                                                                                                                                      Actions »
                                                                     Visualization display
Proof scripts
theory runningV1 begin
                                                                     Applicable Proof Methods: Goals sorted according to the 'smart' heuristic (loop breakers delayed
Message theory
                                                                     1. solve( !KU( ~n ) @ #vk ) // nr. 6
Multiset rewriting rules (5)
                                                                     a. autoprove (A. for all solutions)
                                                                     b. autoprove (B. for all solutions) with proof-depth bound 5
Raw sources (8 cases, 6 partial deconstructions left)
                                                                     Constraint system
Refined sources (8 cases, 6 partial deconstructions
left)
                                                                                        Fr(~ltkA)
                                                                                                                                        Fr(~ltkA.1.)
lemma nonce_secrecy:
                                                                                      #vr : Register_pk[]
                                                                                                                                      #vr.1 : Register_pk[]
  all-traces
                                                                                     IPk( $A, pk(~ltkA) ) Out( pk(~ltkA) )
                                                                                                                                     (Pk(SL pk(~ltkA.1)) Out( pk(~ltkA.1))
                                                                       ILtk( SA, ~ItkA )
                                                                                                                       ILtk(SL ~ItkA.1.)
  "-(a A B s #i #i.
            (SecretI( A, B, s ) ♥ #i) ∧ (K( s ) ♥
#i))"
                                                                                      Fr(~n) IPk(SA, pk(~ltkA)) ILtk(SI, ~ltkA.1
simplify
                                                                                             #i : Bule I[Secret]($I $A ~n )]
solve( !Pk( B, pkR ) ▶1 #i )
  case Register_pk
                                                                                           Out( aenc(<'reg', $I, ~n>, pk(~ltkA)) )
  solve( !Ltk( $I, ltkI ) ▶₂ #i )
    case Register_pk
    by sorry /* removed */
                                                                                                          !KU( ~n ) @ #vk
  aed
aed
                                                                                                  #i : isend[K( ~n )]
end
                                                                     last: none
                                                                     formulas:
```

## Algorithm intuition (3/3)

Proof scripts     Visualization display       theory runningV1 begin     Applicable Proof Methods: Goals sorted according to the 'smart' heuristic (loop breaked is sorted according to the 'smart' heuristic (loop breaked	
theory running/1 begin     Applicable Proof Methods: Goals sorted according to the 'smart' heuristic (loop break Message theory       Message theory     1. solve( IKU( ~n ) @ #vk ) // nr. 6       Multiset rewriting rules (5)     a. autoprove (A. for all solutions) b. autoprove (B. for all solutions) b. autoprove (B. for all solutions) with proof-depth bound 5       Refined sources (8 cases, 6 partial deconstructions become     Constraint system	
Message theory     1. solve(IKU(-n) @ #vk) // nr. 6       Multiset rewriting rules (5)     a. autoprove (A. for all solutions)       Raw sources (8 cases, 6 partial deconstructions left)     b. autoprove (B. for all solutions) with proof-depth bound 5       Refined sources (8 cases, 6 partial deconstructions     Constraint system	akers delaye
Multiset rewriting rules (5)     a. autoprove (A. for all solutions)       Raw sources (8 cases, 6 partial deconstructions left)     b. autoprove (B. for all solutions) with proof-depth bound 5       Refined sources (8 cases, 6 partial deconstructions     Constraint system	
Raw sources (& cases, 6 partial deconstructions left)     b. autoprove (b. for all solutions) with proof-depth bound 5       Refined sources (& cases, 6 partial deconstructions     Constraint system	
Refined sources (8 cases, 6 partial deconstructions	
lemma nonce_secrecy:       fr(=hbA,1)         all-traces       fr(=hbA,1)         "-(a A B s s if #j.       (SecretI(A, B, s ) @ #i) ∧ (K(s ) @         (SecretI(A, B, s ) @ #i) ∧ (K(s ) @       fr(=hbA,1)         Dut(bit(A, ap(=hbA))       Ls((a,=hbA,1)         Solve(1kk(S1, 1kk() ) > 1 #i)       case Register_pk         solve(1kk(S1, 1kk() ) > 1 #i)       case Register_pk         by Sorry /* removed */       ged         end       fi:loongk(-n1)         last: none       frcmular:	(1)

 $\longrightarrow$  the **proof** of this lemma **does not terminate** due to partial deconstructions.

## **Partial deconstructions**

## **Example: Partial deconstruction**



## **Example: Partial deconstruction**



To **resolve** these partial deconstructions, one has to write **sources lemma** detailing the possible origins of the problematic fact.

To **resolve** these partial deconstructions, one has to write **sources lemma** detailing the possible origins of the problematic fact.

Considering our running example: the input is either the message sent by the initiator, or a message constructed by the intruder.

- $\longrightarrow$  the previous raw source will lead to **two refined sources**:
  - 1. either the variable is actually a **nonce** generated by the initiator;
  - 2. or it a term already known by the attacker (such a detour is not useful).

To **resolve** these partial deconstructions, one has to write **sources lemma** detailing the possible origins of the problematic fact.

Considering our running example: the input is either the message sent by the initiator, or a message constructed by the intruder.

- $\longrightarrow$  the previous raw source will lead to **two refined sources**:
  - 1. either the variable is actually a **nonce** generated by the initiator;
  - 2. or it a term already known by the attacker (such a detour is not useful).

Sources lemmas are used to **refine** the sources, but they also need to be **proven correct**.  $\longrightarrow$  this can be done using Tamarin.

## Source lemma on our example

```
First, we annotate the protocol rules:
 rule Rule I:
      [ Fr(n), !Pk(R, pkR), !Ltk(I, ltkI)]
   --[ I(aenc{'req', I, n}pkR), SecretI(I, R, n) ]->
      [ Out(aenc{'reg', I, n}pkR) ]
 rule Rule R:
   [ In(aenc{'req', I, x}pk(ltkR)),
     !Ltk(R, ltkR), !Pk(I, pkI) ]
  --[R(aenc{'req', I, x}pk(ltkR), x)]->
   [ Out(aenc{'rep', x}pkI) ]
```

## Source lemma on our example

```
First, we annotate the protocol rules:
 rule Rule I:
      [ Fr(n), !Pk(R, pkR), !Ltk(I, ltkI)]
   --[ I(aenc{'req', I, n}pkR), SecretI(I, R, n) ]->
      [ Out(aenc{'reg', I, n}pkR) ]
 rule Rule R:
   [ In(aenc{'req', I, x}pk(ltkR)),
     !Ltk(R, ltkR), !Pk(I, pkI) ]
  --[ R(aenc{'req', I, x}pk(ltkR), x) ]->
   [ Out(aenc{'rep', x}pkI) ]
lemma typing [sources]:
"All x m #i. R(m,x)@#i ==>((Ex #j. I(m)@#j & #j < #i)
                            (Ex #j. KU(x)@#j & #j < #i))"
```

### Generalize idea & automate the approach:

- 1. Inspect the **raw sources** computed by TAMARIN
- 2. For each partial deconstruction:
  - $2.1\,$  Identify the variables and facts causing the partial deconstruction
  - 2.2 Identify rules producing matching conclusions
  - 2.3 Add necessary **annotations** to the concerned rules
- 3. Generate a sources lemma using all annotations and add it to the input file

### Generalize idea & automate the approach:

- 1. Inspect the **raw sources** computed by TAMARIN
- 2. For each partial deconstruction:
  - 2.1 Identify the variables and facts causing the partial deconstruction
  - 2.2 Identify rules producing matching conclusions
  - 2.3 Add necessary **annotations** to the concerned rules
- 3. Generate a sources lemma using all annotations and add it to the input file

Note that TAMARIN will verify the correctness of the generated lemma.

But we actually **proved** that the lemmas we generate are **correct** under some assumptions (well-formed rules, subterm-convergent equational theory).

We **implemented** the algorithm in TAMARIN (available in version 1.6.0).

To **enable** automatic source lemma generation, run TAMARIN with --auto-sources:

- If partial deconstructions are present and there is no sources lemma, the algorithm generates a lemma and adds it to the theory.
- If there is already a lemma, or there are no partial deconstructions, TAMARIN runs as usual.

## Case studies: SPORE

#### We tried numerous examples from the **SPORE library**:

Protocol Name	Partial Dec.	Resolved	Automatic	Time
Andrew Secure RPC	14	1	1	42.8s
Modified Andrew Secure RPC	21	1	1	134.3s
BAN Concrete Andrew Secure RPC	0	-	1	10.6s
Lowe modified BAN Andrew Secure RPC	0	-	1	29.8s
CCITT 1	0	-	1	0.8s
CCITT 1c	0	-	1	1.2s
CCITT 3	0	-	1	186.1s
CCITT 3 BAN	0	-	1	3.7s
Denning Sacco Secret Key	5	1	1	0.8s
Denning Sacco Secret Key - Lowe	6	1	1	2.7s
Needham Schroeder Secret Key	14	1	1	3.6s
Amended Needham Schroeder Secret Key	21	1	1	7.1s
Otway Rees	10	1	1	7.7s
SpliceAS	10	1	1	5.9s
SpliceAS 2	10	1	1	7.3s
SpliceAS 3	10	1	1	8.7s
Wide Mouthed Frog	5	1	1	0.6s
Wide Mouthed Frog Lowe	14	1	1	3.5s
WooLam Pi f	5	1	1	0.6s
Yahalom	15	1	1	3.1s
Yahalom - BAN	5	1	1	0.9s
Yahalom - Lowe	21	1	1	2.2s

19/22

## Case studies: Tamarin repository

### We also tested all examples from the **Tamarin repository**:

Name	Partial Dec.	Resolved	Automatic	Time (new)	Time (previous)
Feldhofer (Equivalence)	5	1	1	3.8s	3.5s
NSLPK3	12	1	1	1.8s	1.8s
NSLPK3 untagged	12	1	×	-	-
NSPK3	12	1	1	2.4s	2.2s
JCS12 Typing Example	7	1	×	0.3s	0.2s
Minimal Typing Example	6	1	1	0.1s	0.1s
Simple RFID Protocol	24	1	×	0.7s	0.5s
StatVerif Security Device	12	1	1	0.3s	0.4s
Envelope Protocol	9	1	×	25.7s	25.3s
TPM Exclusive Secrets	9	1	×	1.8s	1.8s
NSL untagged (SAPIC)	18	1	1	4.3s	19.9s
StatVerif Left-Right (SAPIC)	18	1	1	28.8s	29.6s
TPM Envelope (Equivalence)	9	×	-	-	-
5G AKA	240	×	-	-	-
Alethea	30	*	-	-	-
PKCS11-templates	68	×	-	-	-
NSLPK3XOR	24	×	-	-	-
Chaum Offline Anonymity	128	×	-	-	-
FOO Eligibility	70	×	-	-	-
Okamoto Eligibility	66	×	-	-	-

- Automation in TAMARIN often fails because of partial deconstructions
- Developed & implemented a new algorithm to automatically generate sources lemmas
- Proved correctness of the generated lemmas
- Algorithm works well in practice, many examples become fully or at least partly automatic
- Available in TAMARIN 1.6.0
- Future work:
  - Handle more general equational theories
  - Handle partial deconstructions stemming from state facts (currenly under submission at JCS)

## **Questions?**

