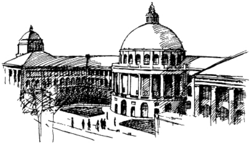
Symbolic verification of cryptographic protocols using Tamarin

Part 3 : Security Properties and Algorithmic Verification

David Basin
ETH Zurich



Summer School on Verification Technology, Systems & Applications
Nancy France
August 2018

Outline

1. Protocol Security Goals
2. Automated Verification
3. Tamarin workflow

1. Protocol Security Goals
2. Automated Verification
3. Tamarin workflow

Goals

What the protocol should achieve, e.g.,

- Authenticate messages, binding them to their originator
- Ensure timeliness of messages (recent, fresh, ...)
- Guarantee secrecy of certain items (e.g., generated keys)

Goals

What the protocol should achieve, e.g.,

- Authenticate messages, binding them to their originator
- Ensure timeliness of messages (recent, fresh, ...)
- Guarantee secrecy of certain items (e.g., generated keys)

Most common goals

- secrecy
- authentication (many different forms)

Other goals

- anonymity, non-repudiation (of receipt, submission, delivery), fairness, availability, sender invariance, ...

Properties

- Semantics of protocol P is a set of traces: $\|P\| = \mathit{traces}(P)$.
  (Traces may be finite or infinite, state- or event-based.)
- Security goal / property ϕ also denotes a set of traces $\|\phi\|$.

Properties

- Semantics of protocol P is a set of traces: $\|P\| = traces(P)$. (Traces may be finite or infinite, state- or event-based.)
- Security goal / property ϕ also denotes a set of traces $\|\phi\|$.

Correctness has an exact meaning

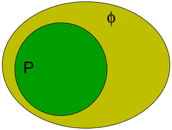
- Protocol P satisfies property ϕ, written $P \models \phi$, iff

$$\|P\| \subseteq \|\phi\|$$

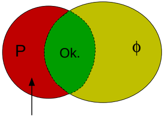
- Attack traces are those in

$$\|P\| - \|\phi\|$$

- Every correctness statement is true or false. Later: how to determine which holds



Ok, no attacks.



Attacks.

Specifying poperties in Tamarin

Property specification language: a fragment of a many-sorted first-order logic with a sort for timepoints (prefixed with #), where quantification is over both messages and timepoints:

- `All` for universal quantification (temporal variables are prefixed with #)
- `Ex` for existential quantification (temporal variables are prefixed with #)
- `==>` for implication, `not` for negation
- `|` for disjunction ("or"), `&` for conjunction ("and")
- `f @ i` for action constraints (the sort prefix for the temporal variable 'i' is optional)
- `i < j` for temporal ordering (the sort prefix for the temporal variables 'i' and 'j' is optional)
- `#i = #j` for an equality between temporal variables 'i' and 'j'
- `x = y` for an equality between message variables 'x' and 'y'

Example

The property that the fresh value ~n is distinct in all applications of a fictitious rule:

```
lemma distinct_nonces:
    "All n #i #j. Act1(n)@i & Act1(n)@j ==> #i=#j"
```

or equivalently

```
lemma distinct_nonces:
  all-traces
    "All n #i #j. Act1(n)@i & Act1(n)@j ==> #i=#j"
```

These lemmas require that the property holds for all traces, we can also express that there exists a trace for which the property holds:

```
lemma distinct_nonces:
  exists-trace
    "not All n #i #j. Act1(n)@i & Act1(n)@j ==> #i=#j"
```

- All action fact symbols may be used in formulas.
- All variables must be guarded.

Guardedness

For universally quantified variables:

- all variables must occur in an action constraint right after the quantifier and
- the outermost logical operator inside the quantifier must be an implication

For existentially quantified variables:

- all variables must occur in an action constraint right after the quantifier and
- the outermost logical operator inside the quantifier must be a conjunction

Direct formulation

- Formulate property ϕ directly in terms of actions occurring in protocol traces, i.e., as a set of (or predicate on) traces.
- Drawback: standard properties like secrecy and authentication become highly protocol-dependent, since they need to refer to the concrete protocol messages.

Direct formulation

- Formulate property ϕ directly in terms of actions occurring in protocol traces, i.e., as a set of (or predicate on) traces.
- Drawback: standard properties like secrecy and authentication become highly protocol-dependent, since they need to refer to the concrete protocol messages.

Protocol instrumentation

- Idea: insert special claim events into the protocol roles:

$$\mathsf{Claim\_}\texttt{claimtype}(R, t),$$

  where R is the executing role, `claimtype` indicates the type of claim, and t is a message term.
- Interface for expressing properties independently of protocol.
- Example: $\mathsf{Claim\_}\texttt{secret}(A, N_A)$ claims that N_A is a secret for role A, i.e., not known to the intruder.

Claim events are part of the protocol rules as actions.

Properties of claim events

- Their only effect is to record facts (claims) in protocol trace.
- Intruder cannot have observed, modified, or generated them.

Claim events are part of the protocol rules as actions.

Properties of claim events

- Their only effect is to record facts (claims) in protocol trace.
- Intruder cannot have observed, modified, or generated them.

Expressing properties using claim events

- Properties of traces *tr* expressed in terms of claim events and other actions (e.g., adversary knowledge K) occuring in *tr*.
- Properties are formulated from the point of view of a given role, thus yielding security guarantees for that role.
- We concentrate on secrecy and variants of authentication, although the approach is not limited to these properties.

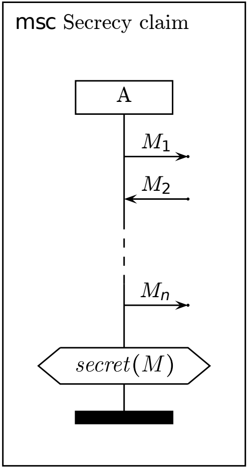
Definition (Secrecy, informally)

The intruder cannot discover data (e.g., key, nonce, etc.) that is intended to be secret.

Role instrumentation

- Insert the claim event Claim_*secret*(A, M) into role A to claim that the message M used in the run remains secret.
- Position: at the end of the role.
- For instance, in NSPK, the nonces *na* and *nb* should remain secret.

Note: In graphs, where the executing role is clear from context, we abbreviate Claim_*claimtype*(A, t) to *claimtype*(t) inside a hexagon.



Definition (Secrecy, first attempt)

The secrecy property consists of all traces *tr* satisfying

$$\forall A, M, i.\ \mathrm{Claim\_}\mathit{secret}(A, M)@i \Rightarrow \neg(\exists j. \mathrm{K}(M)@j)$$

- Let $tr = tr_1; tr_2; \ldots; tr_k$ and $x@k$ is shorthand for $x \in tr_k$.

Definition (Secrecy, first attempt)

The secrecy property consists of all traces tr satisfying

$$\forall A, M, i.\ \mathsf{Claim\_}\mathit{secret}(A, M)@i \Rightarrow \neg(\exists j.\mathsf{K}(M)@j)$$

- Let $tr = tr_1; tr_2; \ldots; tr_k$ and $x@k$ is shorthand for $x \in tr_k$.
- Can only require M to remain secret if A runs the protocol with another honest agent, i.e.,
- Trivially broken if A or B is instantiated with a compromised agent, since then the adversary rightfully knows M.
- This definition is fine for a passive adversary, who observes network traffic, but does not participate in the protocol.

Definition (Compromised Agent)

A compromised agent is under adversary control. It shares all its information with the adversary and can participate in protocols. We model this by having the agent give its initial secret information to the adversary, which can then mimic the agent's actions.

We note the fact that an agent is compromised by a Rev event in the trace, attached to the rule that reveals its initial secrets to the adversary (compare to the creation rule):

$$[!\mathsf{Ltk}(A, skA)] \xrightarrow{\mathsf{Rev(A)}} [\mathsf{Out}(skA)]$$

Exercise: convince yourself that, given the agent's secret, the adversary can perform all of the agent's send and receive steps.

Definition (Honesty)

An agent A is honest in a trace tr when $\mathrm{Rev}(A) \notin tr$.
When making a claim in a rule action, all parties B that are expected to be honest must be listed with a $\mathrm{Honest}(B)$ action in that rule.

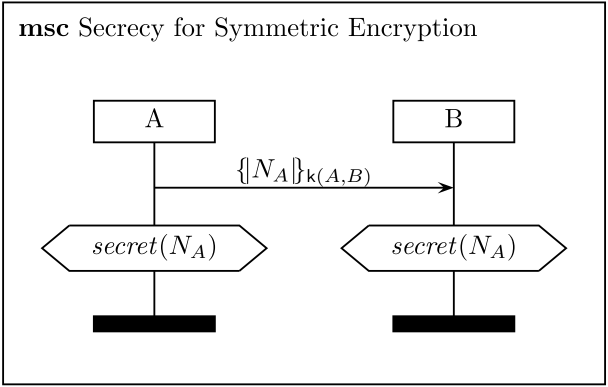
Definition (Honesty)

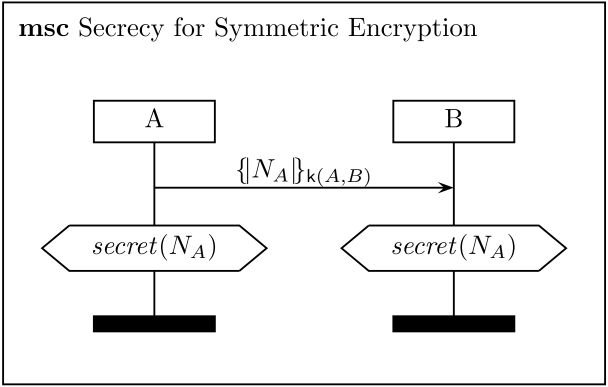
An agent A is honest in a trace tr when $\mathrm{Rev}(A) \notin tr$.
When making a claim in a rule action, all parties B that are expected to be honest must be listed with a $\mathrm{Honest}(B)$ action in that rule.

Definition (Secrecy)

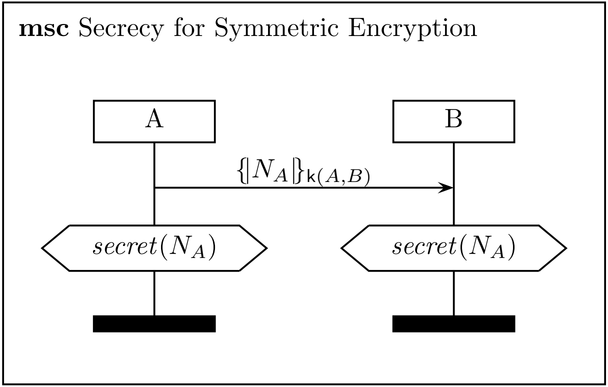
The secrecy property consists of all traces tr satisfying

$$\forall A\ M\ i.\ (\mathrm{Claim\_}\mathit{secret}(A, M)@i)$$
$$\Rightarrow (\neg(\exists j.\mathrm{K}(M)@j) \vee (\exists B\ j.\mathrm{Rev}(B)@j \wedge \mathrm{Honest}(B)@i))$$

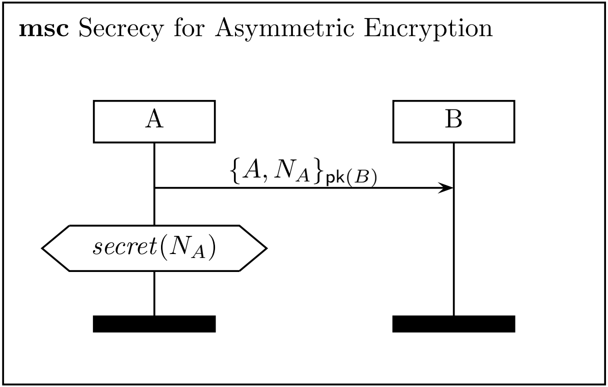

msc Secrecy for Symmetric Encryption
A
B
$\{|N_A|\}_{\mathsf{k}(A,B)}$
$secret(N_A)$
$secret(N_A)$

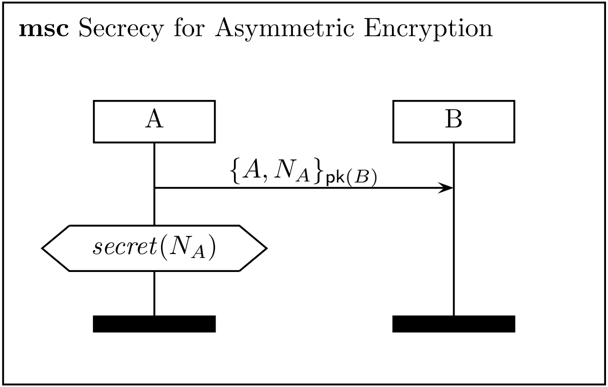


- This is fine: secrecy holds for both A and B.

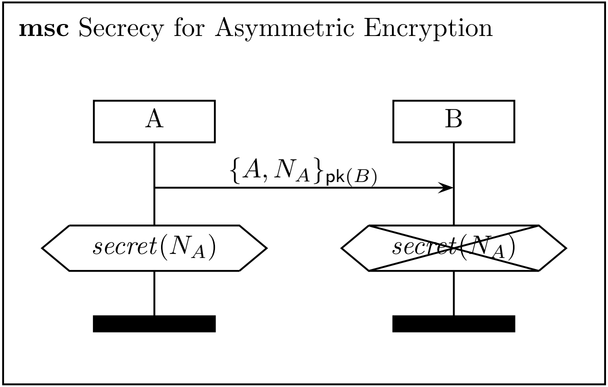


- This is fine: secrecy holds for both A and B.
- We omit the obvious annotations $\mathsf{Honest}(A), \mathsf{Honest}(B)$ in message sequence charts for 2-party protocols.


msc Secrecy for Asymmetric Encryption
A
B
$\{A, N_A\}_{\mathsf{pk}(B)}$
$secret(N_A)$



- Secrecy holds for A: she knows that only B can decrypt message.



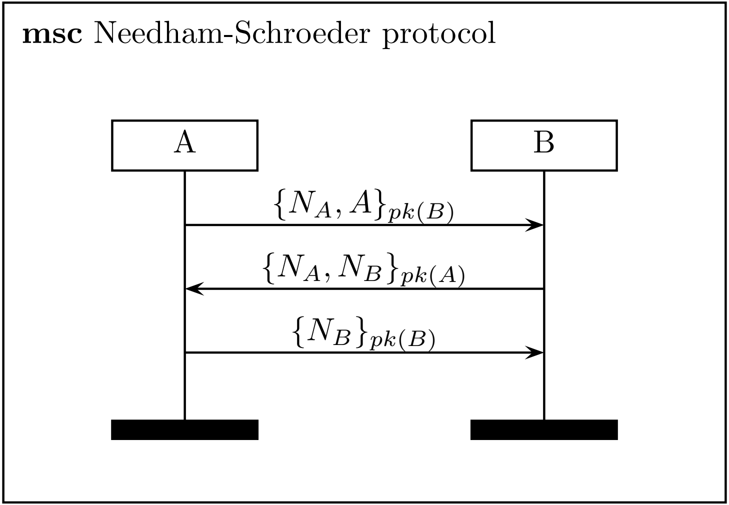
- Secrecy fails for B: he does not know who encrypted message!
  N.B. Intruder can build and send message (without reveal).

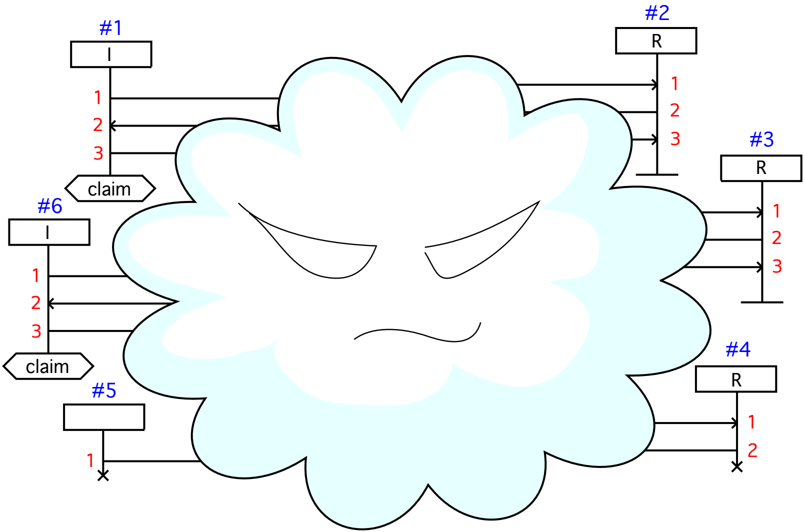
Which authentication are you talking about?

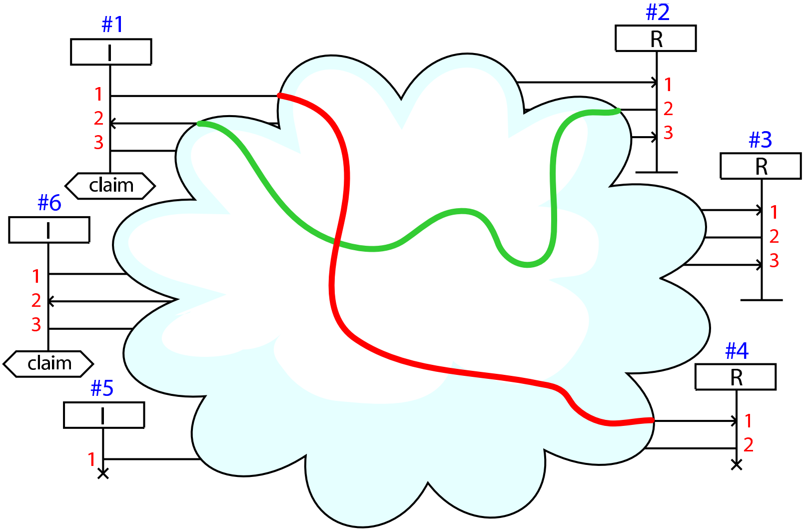
- No unique definition of authentication, but a variety of different forms.
- Considerable effort spent on specifying and classifying, semi-formally or formally, different forms of authentication (e.g., by Cervesato/Syverson, Clark/Jacob, Gollmann, Lowe, Cremers et al.).

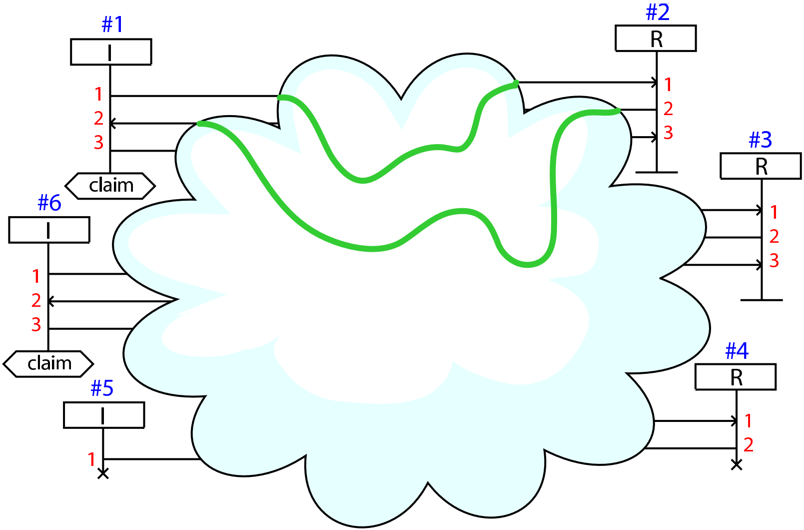
Examples

- ping authentication, aliveness, weak agreement, non-injective agreement, injective agreement, weak and strong authentication, synchronization, and matching histories.


msc Needham-Schroeder protocol
A
B
$\{N_A, A\}_{pk(B)}$
$\{N_A, N_B\}_{pk(A)}$
$\{N_B\}_{pk(B)}$


#1
I
1
2
3
claim
#2
R
1
2
3
#3
R
1
2
3
#6
I
1
2
3
claim
#5
1
#4
R
1
2


#1
I
1
2
3
claim
#2
R
1
2
3
#3
R
1
2
3
#4
R
1
2
#5
I
1
#6
I
1
2
3
claim



#1
I
1
2
3
claim
#6
I
1
2
3
claim
#5
I
1
#2
R
1
2
3
#3
R
1
2
3
#4
R
1
2

A Hierarchy of Authentication Specifications (1)

[Gavin Lowe, 1997]

Gavin Lowe has defined the following hierarchy of increasingly stronger authentication properties[1]:

Aliveness A protocol guarantees to an agent a in role A aliveness of another agent b if, whenever a completes a run of the protocol, apparently with b in role B, then b has previously been running the protocol.

Weak agreement A protocol guarantees to an agent a in role A weak agreement with another agent b if, whenever agent a completes a run of the protocol, apparently with b in role B, then b has previously been running the protocol, apparently with a.

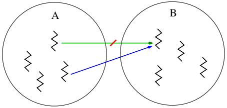
[1] Terminology and notation slightly adapted to our setting. Note that if either a or b is not honest, then the properties are said to trivially hold.

A Hierarchy of Authentication Specifications (2)

[Gavin Lowe, 1997]

Non-injective agreement A protocol guarantees to an agent a in role A non-injective agreement with an agent b in role B on a message M if, whenever a completes a run of the protocol, apparently with b in role B, then b has previously been running the protocol, apparently with a, and b was acting in role B in his run, and the two principals agreed on the message M.

Injective agreement is non-injective agreement where additionally each run of agent a in role A corresponds to a unique run of agent b.



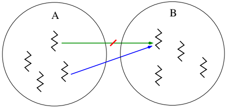
Variants may include recentness: insist that B's run was, e.g., within t time units.

A Hierarchy of Authentication Specifications (2)

[Gavin Lowe, 1997]

Non-injective agreement A protocol guarantees to an agent a in role A non-injective agreement with an agent b in role B on a message M if, whenever a completes a run of the protocol, apparently with b in role B, then b has previously been running the protocol, apparently with a, and b was acting in role B in his run, and the two principals agreed on the message M.

Injective agreement is non-injective agreement where additionally each run of agent a in role A corresponds to a unique run of agent b.



Variants may include recentness: insist that B's run was, e.g., within t time units.

How can we formalize these nontrivial properties?

Role Instrumentation for Authentication

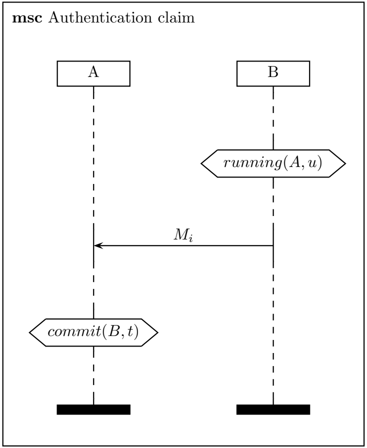
We use two claims to express that role A authenticates role B on t:

In role A:

- Insert a commit claim event $\mathtt{Claim\_commit}(A, B, t)$,$Honest(A)$, $Honest(B)$.
- Position: after A can construct t. Typically, at end of A's role.

In role B:

- Insert a running claim event $\mathtt{Claim\_running}(B, A, u)$.
- Term u is B's view of t.
- Position: after B can construct u and preceding $\mathtt{Claim\_commit}(A, B, t)$.



Definition (Non-injective agreement)

The property $Agreement_{NI}(A, B, t)$ consists of all traces satisfying

$$\begin{array}{ll} \forall a\ b\ t\ i. & \mathrm{Claim\_}\mathtt{commit}(a, b, \langle A, B, t\rangle)@i \\ & \Rightarrow (\exists j. \mathrm{Claim\_}\mathtt{running}(b, a, \langle A, B, t\rangle)@j) \\ & \quad \vee(\exists X\ r. Rev(X)@r \wedge Honest(X)@i) \end{array}$$

- Whenever a commit claim is made with honest agents a and b, then the peer b must be running with the same parameter t, or the adversary has compromised at least one of the two agents.

Definition (Non-injective agreement)

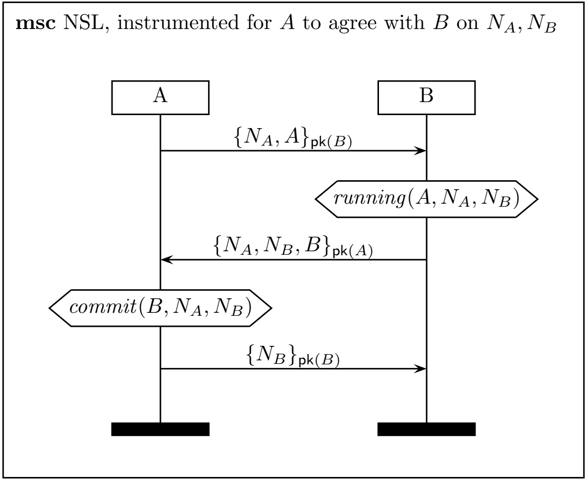
The property $Agreement_{NI}(A, B, t)$ consists of all traces satisfying

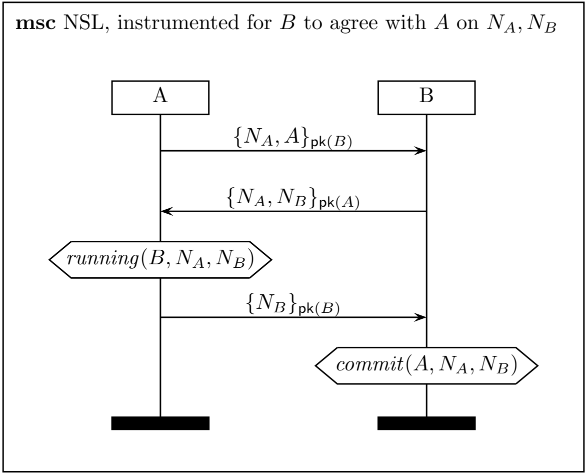
$$\begin{aligned}\forall a\ b\ t\ i.\quad & \mathrm{Claim\_}\mathtt{commit}(a, b, \langle A, B, t\rangle)@i \\ & \Rightarrow (\exists j.\mathrm{Claim\_}\mathtt{running}(b, a, \langle A, B, t\rangle)@j) \\ & \quad \vee(\exists X\ r.Rev(X)@r \wedge Honest(X)@i)\end{aligned}$$

- Whenever a commit claim is made with honest agents a and b, then the peer b must be running with the same parameter t, or the adversary has compromised at least one of the two agents.

Exercise Does ordering of i and j matter.
(Hint: set of traces is prefix closed.)

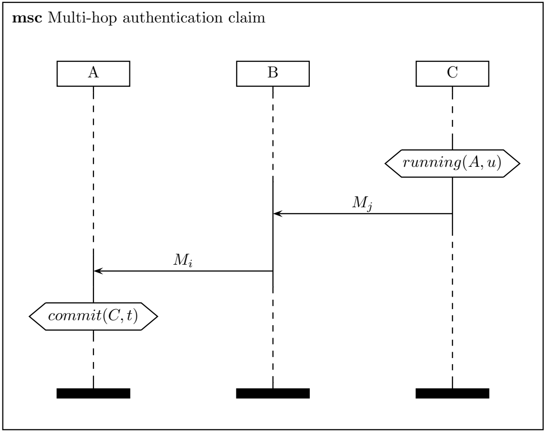
Example: NSL Protocol (1/2)



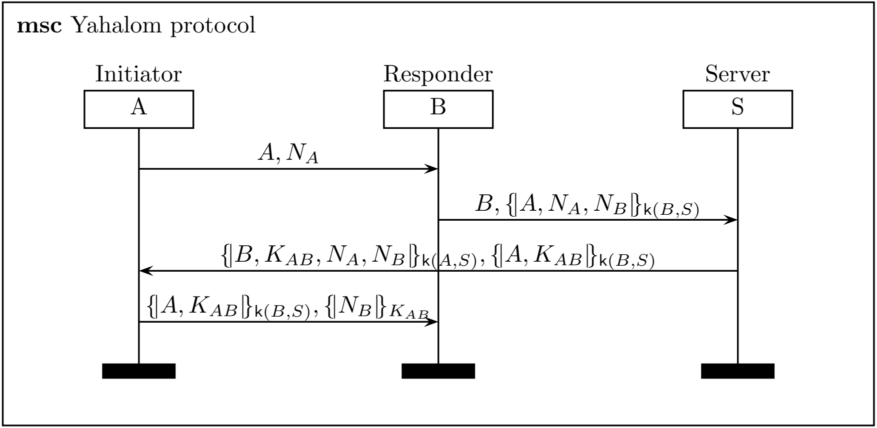


msc NSL, instrumented for B to agree with A on N_A, N_B
A
B
$\{N_A, A\}_{\mathsf{pk}(B)}$
$\{N_A, N_B\}_{\mathsf{pk}(A)}$
$running(B, N_A, N_B)$
$\{N_B\}_{\mathsf{pk}(B)}$
$commit(A, N_A, N_B)$

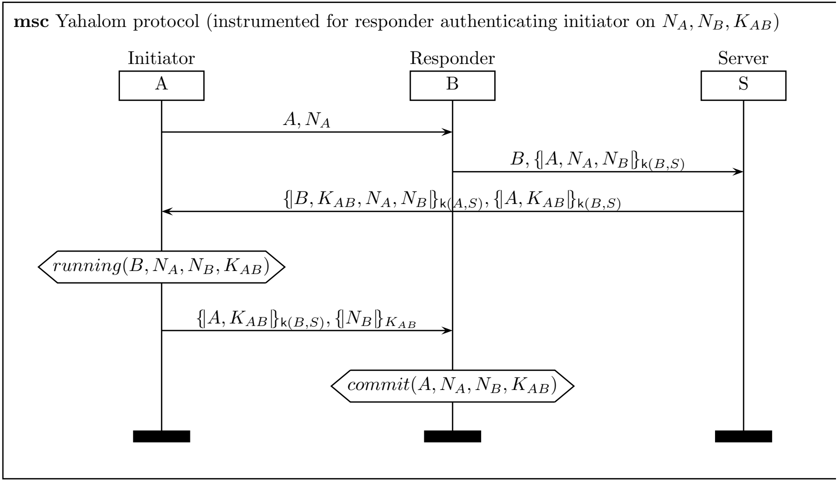
Role Instrumentation for Authentication (cont.)

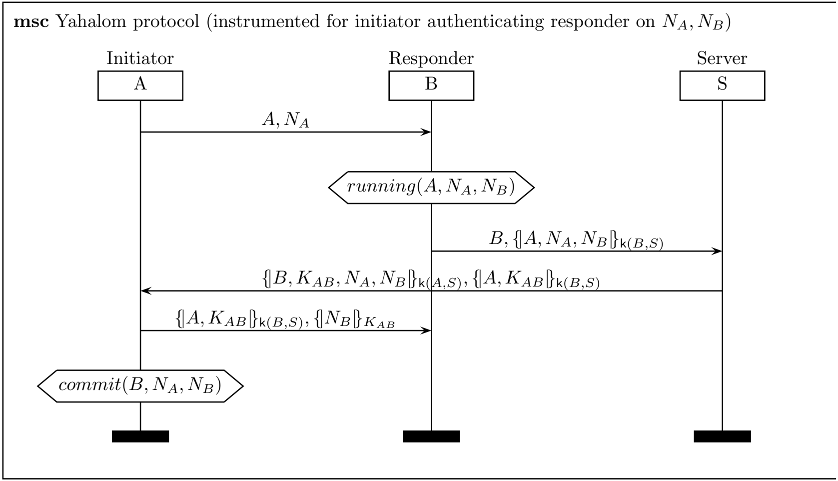


Event causality in multi-hop authentication claims: The *running* event must causally precede the *commit* event and the messages t and u must be known at the position of the claim event in the respective role.


msc Yahalom protocol
Initiator
Responder
Server
A
B
S
A, N_A
$B, \{|A, N_A, N_B|\}_{\mathsf{k}(B,S)}$
$\{|B, K_{AB}, N_A, N_B|\}_{\mathsf{k}(A,S)}, \{|A, K_{AB}|\}_{\mathsf{k}(B,S)}$
$\{|A, K_{AB}|\}_{\mathsf{k}(B,S)}, \{|N_B|\}_{K_{AB}}$

Example: Yahalom Protocol (2/3)





Note: agreement for A on K_{AB} is not possible, since B gets K_{AB} after A.

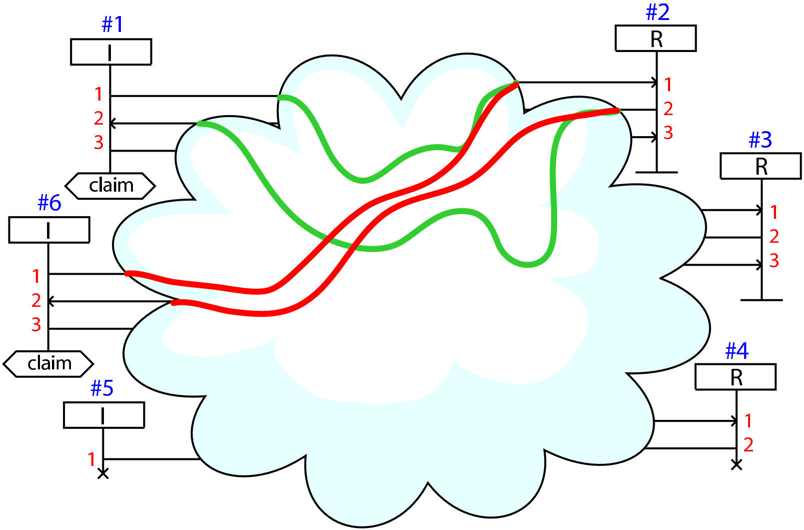
Definition (Injective agreement)

The property $Agreement(A, B, t)$ consists of all traces satisfying:

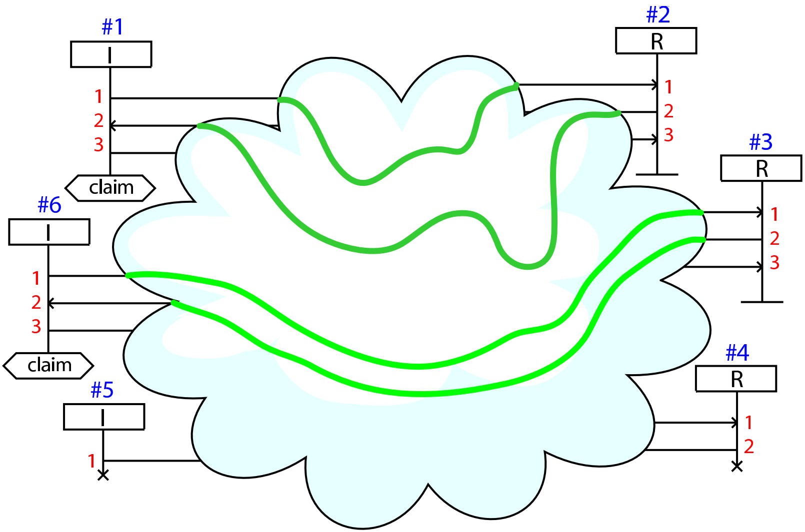
$$
\begin{aligned}
\forall a\ b\ t\ i.\quad & \mathrm{Claim\_commit}(a, b, \langle A, B, t\rangle)@i \\
& \Rightarrow (\exists j.\mathrm{Claim\_running}(b, a, \langle A, B, t\rangle)@j \land j < i \\
& \qquad \land \neg(\exists a_2 b_2 i_2.\mathrm{Claim\_commit}(a_2, b_2, \langle A, B, t\rangle)@i_2 \\
& \qquad \land \neg(i_2 = i))) \\
& \quad \lor (\exists X\ r.Rev(X)@r \land Honest(X)@i)
\end{aligned}
$$

Remarks

- For each commit by a in role A on the trace there is a unique matching b executing role B.

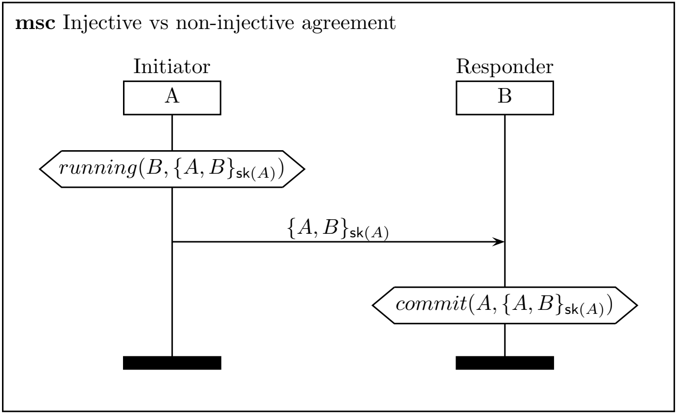

#1
I
1
2
3
claim
#2
R
1
2
3
#3
R
1
2
3
#6
I
1
2
3
claim
#5
I
1
#4
R
1
2

Successful Injective Authentication



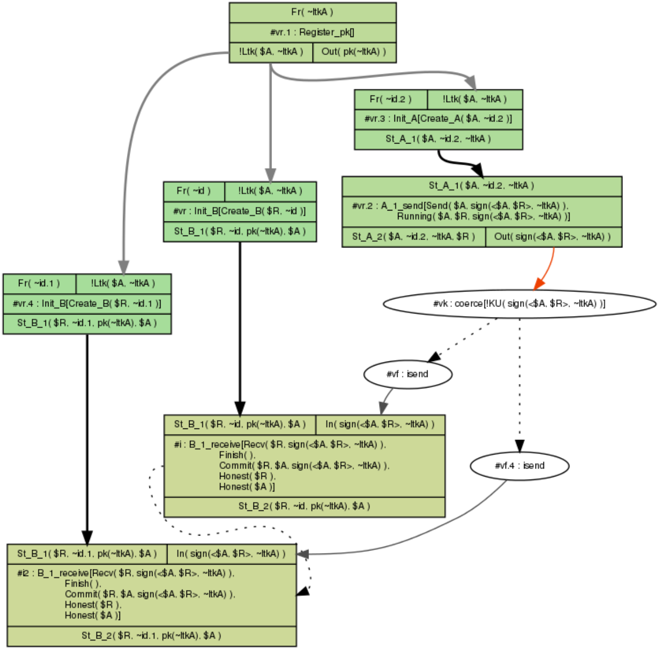
Injective vs Non-injective Agreement

They are not equivalent



- Non-injective agreement holds.
- Injective agreement fails, since the adversary can replay message to several threads in responder role B.

Injective Agreement counter-example



Definition (Weak agreement)

A trace *tr* satisfies the property $WeakAgreement(A, B)$ iff

$$\begin{array}{ll} \forall a\ b\ i. & \mathrm{Claim\_}\mathtt{commit}(a, b, \langle\rangle)@i \\ & \Rightarrow (\exists j.\mathrm{Claim\_}\mathtt{running}(b, a, \langle\rangle)@j) \\ & \quad \vee(\exists X\ r.Rev(X)@r \wedge Honest(X)@i) \end{array}$$

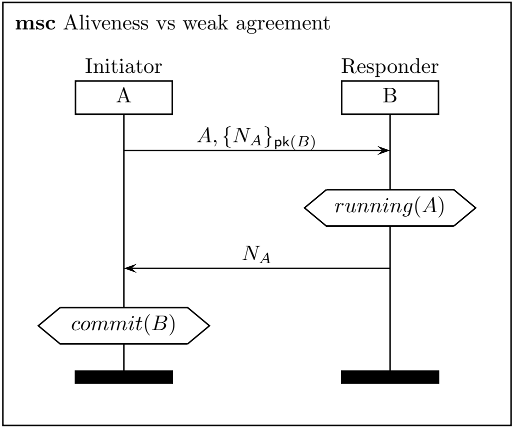
It is sufficient that the agents agree they are communicating, it is not required that they play the right roles. Note also the empty list of data $\langle\rangle$ that is agreed upon, i.e., none.

Definition (Aliveness)

A trace *tr* satisfies the property $Alive(A, B)$ iff

$$
\begin{array}{ll}
\forall a\ b\ i. & \text{Claim\_}\mathtt{commit}(a, b, \langle\rangle)@i \\
& \Rightarrow (\exists j\ id. Create\_\mathtt{B}(b, id)@j \lor Create\_\mathtt{A}(b, id)@j) \\
& \quad \lor(\exists X\ r. Rev(X)@r \land Honest(X)@i)
\end{array}
$$

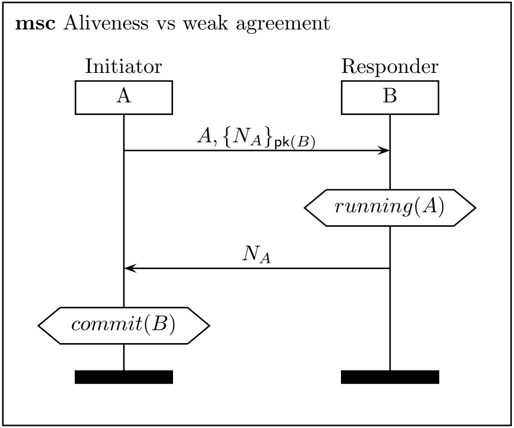
It is neither required that the agent b, believed to instantiate role B by agent a, really plays role B, nor that he believes to be talking to a.



msc Aliveness vs weak agreement
Initiator
Responder
A
B
$A, \{N_A\}_{\mathsf{pk}(B)}$
$running(A)$
N_A
$commit(B)$

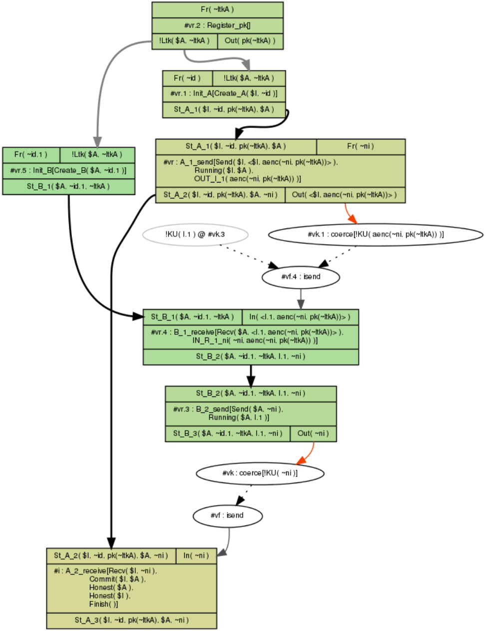
Aliveness vs Weak Agreement

They are not equivalent

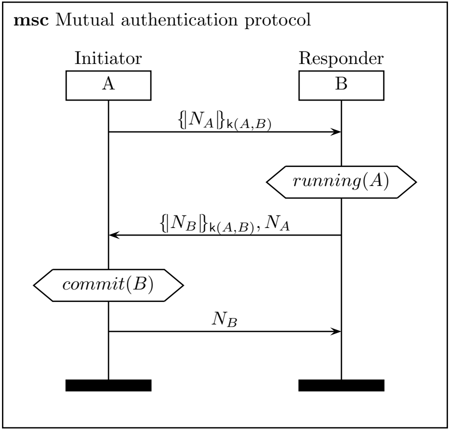


- Aliveness holds: only B can decrypt the fresh nonce N_A.
- Weak agreement fails, since adversary may modify unprotected identity A to C in first message so that B thinks he is talking to C.

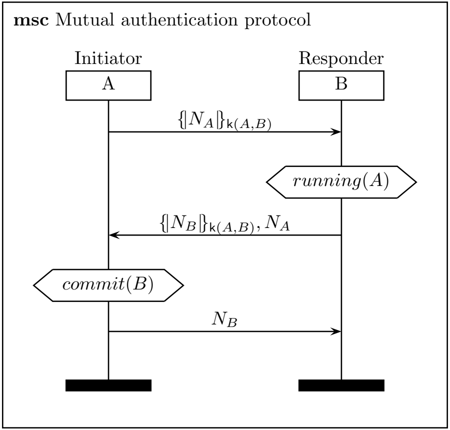
Weak Agreement counter-example



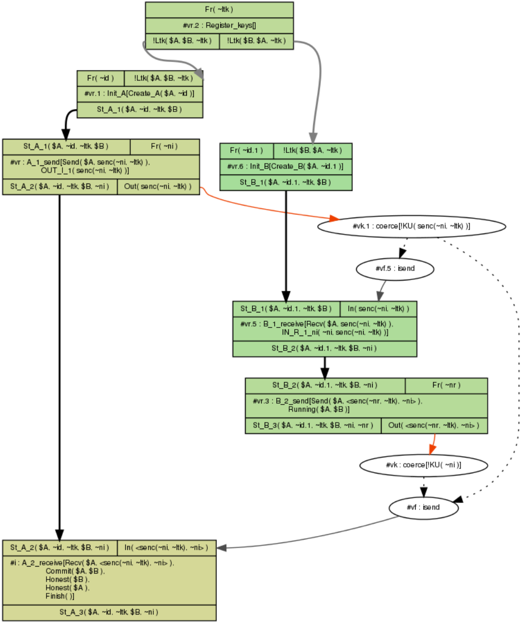
When Even Aliveness Fails ...



When Even Aliveness Fails ...



- Reflection attack: A may complete run without B's participation.
- Hence, aliveness fails.


Fr(~ltk)
#vr.2 : Register_keys[]
!Ltk($A, $B, ~ltk)
!Ltk($B, $A, ~ltk)
Fr(~id)
!Ltk($A, $B, ~ltk)
#vr.1 : Init_A[Create_A($A, ~id)]
St_A_1($A, ~id, ~ltk, $B)
St_A_1($A, ~id, ~ltk, $B)
Fr(~ni)
#vr : A_1_send[Send($A, senc(~ni, ~ltk)), OUT_I_1(senc(~ni, ~ltk))]
St_A_2($A, ~id, ~ltk, $B, ~ni)
Out(senc(~ni, ~ltk))
Fr(~id.1)
!Ltk($B, $A, ~ltk)
#vr.6 : Init_B[Create_B($A, ~id.1)]
St_B_1($A, ~id.1, ~ltk, $B)
#vk.1 : coerce[!KU(senc(~ni, ~ltk))]
#vf.5 : isend
St_B_1($A, ~id.1, ~ltk, $B)
In(senc(~ni, ~ltk))
#vr.5 : B_1_receive[Recv($A, senc(~ni, ~ltk)), IN_R_1_ni(~ni, senc(~ni, ~ltk))]
St_B_2($A, ~id.1, ~ltk, $B, ~ni)
St_B_2($A, ~id.1, ~ltk, $B, ~ni)
Fr(~nr)
#vr.3 : B_2_send[Send($A, <senc(~nr, ~ltk), ~ni>), Running($A, $B)]
St_B_3($A, ~id.1, ~ltk, $B, ~ni, ~nr)
Out(<senc(~nr, ~ltk), ~ni>)
#vk : coerce[!KU(~ni)]
#vf : isend
St_A_2($A, ~id, ~ltk, $B, ~ni)
In(<senc(~ni, ~ltk), ~ni>)
#i : A_2_receive[Recv($A, <senc(~ni, ~ltk), ~ni>), Commit($A, $B), Honest($B), Honest($A), Finish( )]
St_A_3($A, ~id, ~ltk, $B, ~ni)

Basic key-oriented goals

- key freshness
- (implicit) key authentication: a key is only known to the communicating agents A and B and mutually trusted parties
- key confirmation of A to B is provided if B has assurance that agent A has possession of key K
- explicit key authentication $=$ key authentication $+$ key confirmation $\Rightarrow$ expressible in terms of secrecy and agreement

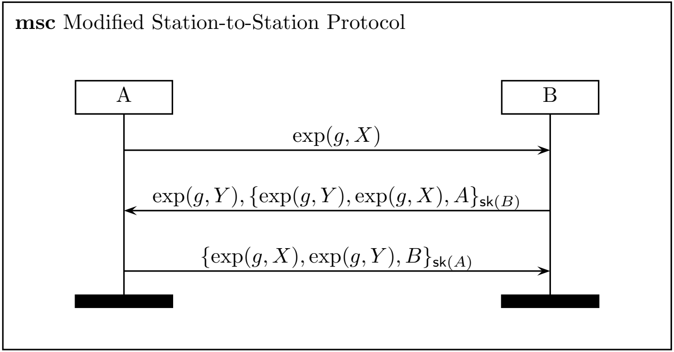
Basic key-oriented goals

- key freshness
- (implicit) key authentication: a key is only known to the communicating agents A and B and mutually trusted parties
- key confirmation of A to B is provided if B has assurance that agent A has possession of key K
- explicit key authentication $=$ key authentication $+$ key confirmation $\Rightarrow$ expressible in terms of secrecy and agreement

Goals concerning compromised keys

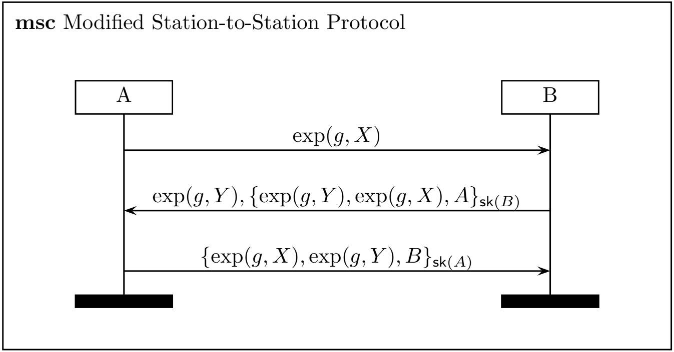
- (perfect) forward secrecy: compromise of long-term keys of a set of principals does not compromise the session keys established in previous protocol runs involving those principals
- resistance to key-compromise impersonation: compromise of long-term key of an agent A does not allow the adversary to masquerade to A as a different principal.

Forward Secrecy: Example 1

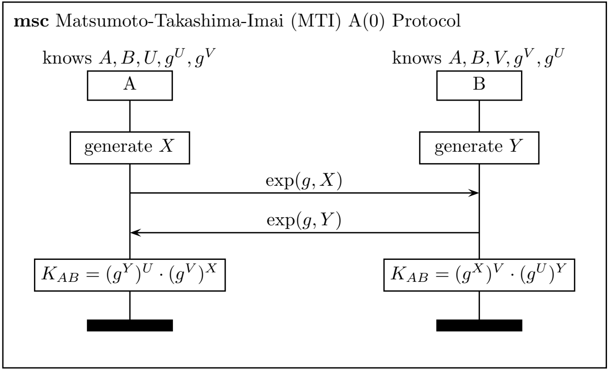


- Signatures are used to authenticate the Diffie-Hellman public keys $\exp(g, X)$ and $\exp(g, Y)$.

Forward Secrecy: Example 1

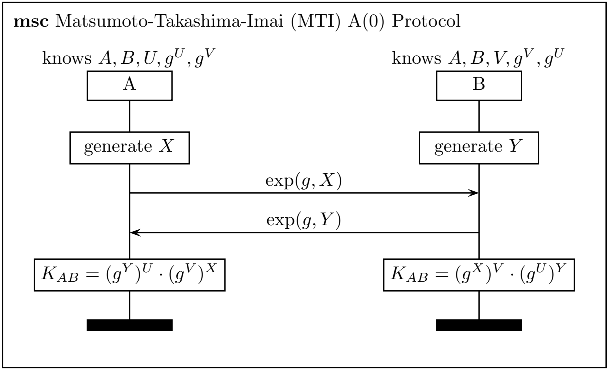


- Signatures are used to authenticate the Diffie-Hellman public keys $\exp(g, X)$ and $\exp(g, Y)$.
- Protocol provides forward secrecy: The adversary cannot derive session key $K_{AB} = \exp(\exp(g, X), Y)$ by compromise of signing keys.

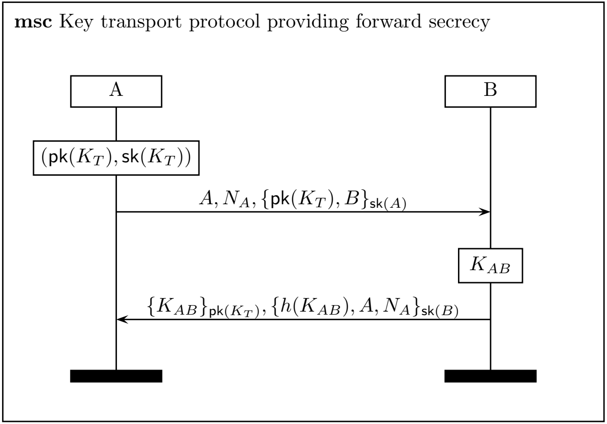


- Message exchange as in basic DH; protocol combines long-term and ephemeral DH keys to authenticate exchanged DH public keys.

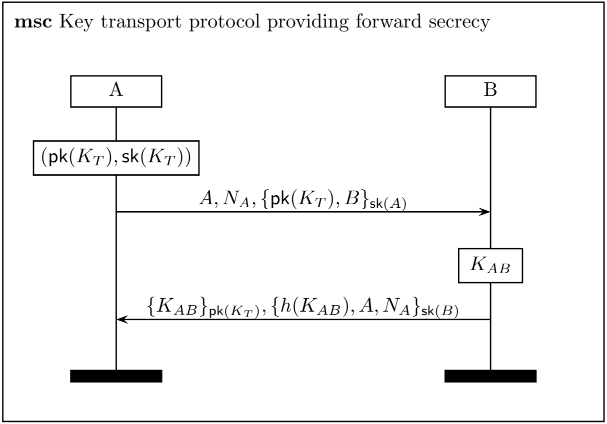
Forward Secrecy: Example 2



- Message exchange as in basic DH; protocol combines long-term and ephemeral DH keys to authenticate exchanged DH public keys.
- Protocol does not provide forward secrecy: The adversary can construct the session key $K_{AB} = g^{VX+UY}$ as $(g^X)^V \cdot (g^Y)^U$ from observed messages and long-term private keys U and V.



- A generates an ephemeral asymmetric key pair $(\mathsf{pk}(K_T), \mathsf{sk}(K_T))$.



- A generates an ephemeral asymmetric key pair $(\mathsf{pk}(K_T), \mathsf{sk}(K_T))$.
- Protocol provides forward secrecy without using Diffie-Hellman keys: Adversary cannot learn session key by compromise of signing keys.

We would like to have a program V with ...

- Input:
  - some description of a program P
  - some description of a functional specification S
- Output: *Yes* if P satisfies S, and *No* otherwise.
- Optional extra: in the *No* case, give a counter-example, i.e. an input on which P violates the specification.

We would like to have a program V with ...

- Input:
  - some description of a program P
  - some description of a functional specification S
- Output: *Yes* if P satisfies S, and *No* otherwise.
- Optional extra: in the *No* case, give a counter-example, i.e. an input on which P violates the specification.

Forget it:

Theorem (Rice)

Let S be any non-empty, proper subset of the computable functions. Then the verification problem for S (the set of programs P that compute a function in S) is undecidable.

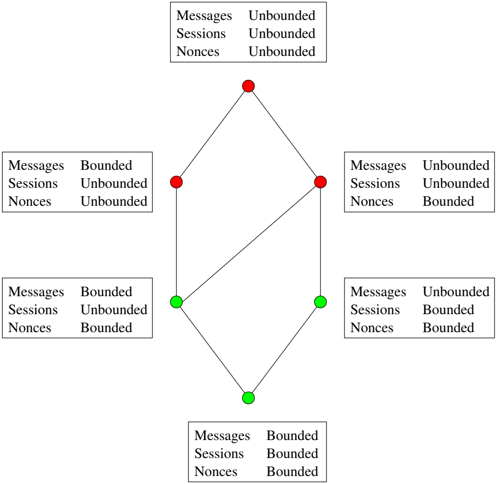
For security protocols, the state space can be infinite for (at least) the following reasons:

Messages The intruder can compose arbitrarily complex messages from his knowledge, e.g., $i, h(i), h(h(i)), \ldots$.

Sessions Any number of sessions (or threads) may be executed.

Nonces Unbounded number of fresh nonces generated.

(Un)decidability: Complete picture



Bottom line: need at least two bounded parameters for decidability.

- Use multiset rewriting to represent protocol
- Adversary message deduction rules given as multiset rewriting rules
- Properties specified in first-order logic
  - Allows quantification over messages and timepoints
- Verification algorithm is proven sound and complete

Tamarin overview

- Use multiset rewriting to represent protocol
- Adversary message deduction rules given as multiset rewriting rules
- Properties specified in first-order logic
  - Allows quantification over messages and timepoints
- Verification algorithm is proven sound and complete

- Backwards reachability analysis – searching for insecure states
  - Negate security property, search for *solutions*
- Constraint solving
  - Uses dependency graphs
  - Normal dependency graphs for state-space reduction – efficiency despite undecidability

$$1 : \frac{}{\mathsf{Fr}(\mathsf{a})} \qquad 2 : \frac{}{\mathsf{Fr}(\mathsf{k})}$$

$$3 : \frac{\mathsf{Fr}(\mathsf{a}) \quad \mathsf{Fr}(\mathsf{k})}{\mathsf{St}(\mathsf{a},\mathsf{k}) \quad \mathsf{Out}(\mathit{enc}(\mathsf{a},\mathsf{k}))) \quad \mathsf{Key}(\mathsf{k})}$$

$$4 : \frac{\mathsf{Key}(\mathsf{k})}{\mathsf{Out}(\mathsf{k})}[\mathsf{Rev}(\mathsf{k})]]$$

$$5 : \frac{\mathsf{Out}(\mathit{enc}(\mathsf{a},\mathsf{k}))}{\mathsf{K}(\mathit{enc}(\mathsf{a},\mathsf{k}))} \qquad 6 : \frac{\mathsf{Out}(\mathsf{k})}{\mathsf{K}(\mathsf{k})}$$

$$7 : \frac{\mathsf{K}(\mathit{enc}(\mathsf{a},\mathsf{k})) \quad \mathsf{K}(\mathsf{k})}{\mathsf{K}(\mathit{dec}(\mathit{enc}(\mathsf{a},\mathsf{k}),\mathsf{k}))}$$

$$8 : \frac{\mathsf{K}(\mathsf{a}) \quad \mathsf{K}(\mathsf{a})}{\mathsf{K}(\langle \mathsf{a},\mathsf{a}\rangle)}$$

$$9 : \frac{\mathsf{K}(\langle \mathsf{a},\mathsf{a}\rangle)}{\mathsf{In}(\langle \mathsf{a},\mathsf{a}\rangle)}[\mathsf{K}(\langle \mathsf{a},\mathsf{a}\rangle)]$$

$$10 : \frac{\mathsf{St}(\mathsf{a},\mathsf{k}) \quad \mathsf{In}(\langle \mathsf{a},\mathsf{a}\rangle)}{}[\mathsf{Fin}(\mathsf{a},\mathsf{k})]$$

Tamarin's constraint solving algorithm

function $\textsc{Solve}(P \vDash_{E_{DH}} \varphi)$
  $\hat{\varphi} \leftarrow \neg\varphi$ rewritten into negation normal form
  $\boldsymbol{\Omega} \leftarrow \{\{\hat{\varphi}\}\}$
  while $\boldsymbol{\Omega} \neq \emptyset$ and $solved(\boldsymbol{\Omega}) = \emptyset$ **do**
    choose $\Gamma \rightsquigarrow_P \{\Gamma_1, \ldots, \Gamma_k\}$ such that $\Gamma \in \boldsymbol{\Omega}$
    $\boldsymbol{\Omega} \leftarrow (\boldsymbol{\Omega} \setminus \{\Gamma\}) \cup \{\Gamma_1, \ldots, \Gamma_k\}$
  if $solved(\boldsymbol{\Omega}) \neq \emptyset$
    then return "attack(s) found: ", $solved(\boldsymbol{\Omega})$
    else return "verification successful"

Trace formula reduction rules:

$\mathbf{S}_{\approx}:$	$\Gamma \rightsquigarrow_P \ \Vert_{\sigma \in unify_{AC}(t_1,t_2)}(\Gamma\sigma)$	if $(t_1 \approx t_2) \in \Gamma$ and $t_1 \neq_{AC} t_2$
$\mathbf{S}_{\doteq}:$	$\Gamma \rightsquigarrow_P \Gamma\{i/j\}$	if $(i \doteq j) \in \Gamma$ and $i \neq j$
$\mathbf{S}_{@}:$	$\Gamma \rightsquigarrow_P \ \Vert_{ri \in [P]^{DH} \cup \{\text{ISEND}\}} \Vert_{f' \in acts(ri)}(i : ri\,,\ f \approx f', \Gamma)$	if $(f@i) \in \Gamma$ and $(f@i) \notin_{AC} as(\Gamma)$
$\mathbf{S}_{\perp}:$	$\Gamma \rightsquigarrow_P \perp$	if $\perp \in \Gamma$
$\mathbf{S}_{\neg,\approx}:$	$\Gamma \rightsquigarrow_P \perp$	if $\neg(t \approx t) \in_{AC} \Gamma$
$\mathbf{S}_{\neg,\doteq}:$	$\Gamma \rightsquigarrow_P \perp$	if $\neg(i \doteq i) \in \Gamma$
$\mathbf{S}_{\neg,@}:$	$\Gamma \rightsquigarrow_P \perp$	if $\neg(f@i) \in \Gamma$ and $(f@i) \in as(\Gamma)$
$\mathbf{S}_{\neg,\lessdot}:$	$\Gamma \rightsquigarrow_P (i \lessdot j, \Gamma) \Vert (\Gamma\{i/j\})$	if $\neg(j \lessdot i) \in \Gamma$ and neither $i \lessdot_\Gamma j$ nor $i = j$
$\mathbf{S}_{\vee}:$	$\Gamma \rightsquigarrow_P (\phi_1, \Gamma) \Vert (\phi_2, \Gamma)$	if $(\phi_1 \vee \phi_2) \in_{AC} \Gamma$ and $\{\phi_1, \phi_2\} \cap_{AC} \Gamma = \varnothing$
$\mathbf{S}_{\wedge}:$	$\Gamma \rightsquigarrow_P (\phi_1, \phi_2, \Gamma)$	if $(\phi_1 \wedge \phi_2) \in_{AC} \Gamma$ and not $\{\phi_1, \phi_2\} \subseteq_{AC} \Gamma$
$\mathbf{S}_{\exists}:$	$\Gamma \rightsquigarrow_P (\phi\{y/x\}, \Gamma)$	if $(\exists x{:}s.\ \phi) \in \Gamma$, $\phi\{w/x\} \notin_{AC} \Gamma$ for every term w of sort s, and $y{:}s$ fresh
$\mathbf{S}_{\forall}:$	$\Gamma \rightsquigarrow_P (\psi\sigma, \Gamma)$	if $(\forall \vec{x}.\ \neg(f@i) \vee \psi) \in \Gamma$, $dom(\sigma) = set(\vec{x})$, $(f@i)\sigma \in_{AC} as(\Gamma)$, and $\psi\sigma \notin_{AC} \Gamma$

Some more constraint solving rules

Graph constraint reduction rules:

$\mathbf{U}_{lbl}$: $\Gamma \rightsquigarrow_P (ri \approx ri', \Gamma)$ if $\{i : ri, i : ri'\} \subseteq \Gamma$ and $ri \neq_{AC} ri'$

$\mathbf{DG1}_1$: $\Gamma \rightsquigarrow_P \bot$ if $i \lessdot_\Gamma i$

$\mathbf{DG1}_2$: $\Gamma \rightsquigarrow_P (f \approx f', \Gamma)$ if $c \rightarrowtail p \in \Gamma$, $(c, f) \in cs(\Gamma)$, $(p, f') \in ps(\Gamma)$, and $f \neq_{AC} f'$

$\mathbf{DG2}_1$: $\Gamma \rightsquigarrow_P (\text{if } u = v \text{ then } \Gamma\{i/j\} \text{ else } \bot)$ if $\{(i,v) \rightarrowtail p, (j,u) \rightarrowtail p\} \subseteq \Gamma$ and $i \neq j$

$\mathbf{DG2}_{2,P}$: $\Gamma \rightsquigarrow_P \|_{ri \in [P]^{DH} \cup \{\text{ISEND}, \text{FRESH}\}} \|_{u \in idx(concs(ri))} (i : ri, (i,u) \rightarrowtail p, \Gamma)$
if p is an open f-premise in Γ, f is not a $\mathsf{K}^\uparrow$- or $\mathsf{K}^\downarrow$-fact, and i fresh

$\mathbf{DG3}$: $\Gamma \rightsquigarrow_P (\text{if } u = v \text{ then } \Gamma\{i/j\} \text{ else } \bot)$ if $\{c \rightarrowtail (i,v), c \rightarrowtail (j,u)\} \subseteq \Gamma$, c linear in Γ, and $i \neq j$,

$\mathbf{DG4}$: $\Gamma \rightsquigarrow_P \Gamma\{i/j\}$ if $\{i : -[]\!\!\rightarrow \mathsf{Fr}(m), j : -[]\!\!\rightarrow \mathsf{Fr}(m)\} \subseteq_{AC} \Gamma$ and $i \neq j$

$\mathbf{N1}$: $\Gamma \rightsquigarrow_P \bot$ if $(i : ri) \in \Gamma$ and ri not $\downarrow_{DH}$-normal

$\mathbf{N5,6}$: $\Gamma \rightsquigarrow_P \Gamma\{i/j\}$ if $\{((i,1), \mathsf{K}^d_e(t)), ((j,1), \mathsf{K}^{d'}_{e'}(t))\} \subseteq_{AC} cs(\Gamma)$, $i \neq j$, *and*
$d = d'$ or $\{i,j\} \cap \{k \mid \exists ri \in insts(\{\text{PAIR}\uparrow, \text{INV}\uparrow, \text{COERCE}\}).\ (k : ri) \in \Gamma\} = \emptyset$

$\mathbf{N6}$: $\Gamma \rightsquigarrow_P (i \lessdot j, \Gamma)$ if $((j,v), \mathsf{K}^\uparrow_{e'}(t)) \in ps(\Gamma)$, $m \in_{AC} inp(t)$, $((i,u), \mathsf{K}^\downarrow_e(m)) \in cs(\Gamma)$, and not $i \lessdot_\Gamma j$

$\mathbf{N7}$: $\Gamma \rightsquigarrow_P \bot$ if $(i : \mathsf{K}^\downarrow_{\mathsf{exp}}(s_1), \mathsf{K}^\uparrow_e(t_1) -[]\!\!\rightarrow \mathsf{K}^\downarrow_{\mathsf{noexp}}(s_2 \hat{}\ t_2)) \in \Gamma$, s_2 is of sort *pub*, and $inp(t_2) \subseteq inp(t_1)$

Message deduction constraint reduction rules:

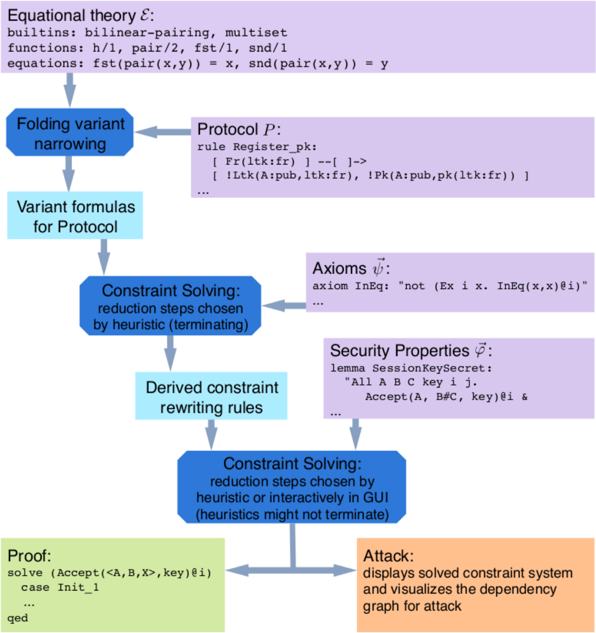
$\mathbf{DG2}_{2,\uparrow i}$: $\Gamma \rightsquigarrow_P \|_{(l -[]\rightarrow \mathsf{K}^\uparrow_e(t)) \in ND^{c\text{-}expl}} (i : (l -[]\!\!\rightarrow \mathsf{K}^\uparrow_e(t)), t \approx m, (i,1) \twoheadrightarrow p, \Gamma)$
if p is an open implicit m-construction in Γ, m non-trivial, and i fresh

$\mathbf{DG2}_{2,\uparrow e}$: $\Gamma \rightsquigarrow_P \|_{ri \in ND^{c\text{-}expl}} (i : ri, (i,1) \rightarrowtail p, \Gamma)$
if p is an open $\mathsf{K}^\uparrow_e(m)$-premise in Γ, $\{m\} = inp(m)$, m non-trivial, and i fresh

$\mathbf{DG2}_{2,\downarrow}$: $\Gamma \rightsquigarrow_P (i : \mathsf{Out}(y) -[]\!\!\rightarrow \mathsf{K}^\downarrow_{\mathsf{exp}}(y), (i,1) \dashrightarrow p, \Gamma)$ if p is an open $\mathsf{K}^\downarrow_e(m)$-premise in Γ and y, i fresh

$\mathbf{DG2}_{\dashrightarrow}$: $(c \dashrightarrow p, \Gamma) \rightsquigarrow_P (c \rightarrowtail p, \Gamma) \parallel \|_{ri \in ND^{destr}} (i : ri, c \rightarrowtail (i,1), (i,1) \dashrightarrow p, \Gamma)$
if $(c, \mathsf{K}^\downarrow_e(m)) \in cs(\Gamma)$, $m \notin \mathcal{V}_{msg}$, and i fresh

Tamarin workflow



- David Basin, Cas Cremers, Jannik Dreier, and Ralf Sasse. *Symbolically Analyzing Security Protocols using TAMARIN*, SIGLOG News, 2017.
- Benedikt Schmidt, Simon Meier, Cas Cremers, and David Basin. *Automated analysis of Diffie-Hellman Protocols and Advanced Security Properties*, Proceedings of the 25th IEEE Computer Security Foundations Symposium (CSF) 2012.
- Shimon Even and Oded Goldreich. *On the security of multi-party ping-pong protocols*, Symposium on Foundations of Computer Science, IEEE Computer Society, 1983.
- N. Durgin, P. Lincoln, J. Mitchell, and A. Scedrov. *Undecidability of bounded security protocols.* In Workshop on Formal Methods and Security Protocols (FMSP '99), 1999.
- Michaël Rusinowitch and Mathieu Turuani. *Protocol Insecurity with Finite Number of Sessions is NP-complete.* CSFW, 2001.