

LTL Model Checking for Security Protocols

Roberto Carbone¹

(joint work with **A. Armando¹** and **L. Compagna²**)

¹Artificial Intelligence Laboratory, DIST, University of Genova

²SAP Research, Sophia-Antipolis, France

20th IEEE Computer Security Foundations Symposium
Venice - July 6-8, 2007

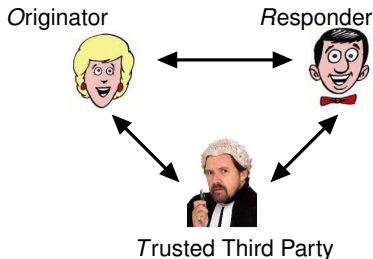


- 1 Introduction
- 2 LTL Model Checking for Security Protocols
- 3 A Motivating Example: the ASW Protocol
 - The Protocol
 - The Assumptions on the Channels and on the Honest Principals
 - Security Properties
- 4 Modelling, Specification and Analysis using Set-rewriting and LTL
 - Specifying the Protocol Scenario and the Intruder
 - Specifying the Assumptions on the Channels and on the Honest Principals
 - Specifying Security Properties
- 5 Conclusions

- Most security protocol analysers **assume** that:
 - ① **channels** controlled by DY intruder
 - ② **honest agents** are **only** asked to react to received messages by sending some other messages.
 - ③ **security properties** are expressed as reachability properties
- In many cases these assumptions can be circumvented by **transforming the problem** (i.e. model+property) at hand.
- In some cases these transformations can be automated.
- In other, more complex cases this must be done manually, but this is in general **difficult** and hence **time consuming** and **error prone**.

- Most security protocol analysers **assume** that:
 - ① **channels** controlled by DY intruder
 - ② **honest agents** are **only** asked to react to received messages by sending some other messages.
 - ③ **security properties** are expressed as reachability properties
- In many cases these assumptions can be circumvented by **transforming the problem** (i.e. model+property) at hand.
- In some cases these transformations can be automated.
- In other, more complex cases this must be done manually, but this is in general **difficult** and hence **time consuming** and **error prone**.

Motivations: a concrete example



The optimistic fair exchange protocol proposed by Asokan, Shoup, and Waidner (ASW) violates all three assumptions:

- **Channels** are assumed to be confidential and/or resilient.
- **Agents** are assumed to make progress during the execution of the protocol (O and R must timeout and T must be always available).
- **Security property** (fair exchange) cannot be directly expressed as a reachability property.

- 1 general framework for automatic security protocol analysis that, by using
 - set-rewriting for specifying the model and
 - LTL for specifying security properties and constraints,allows us to **relieve the above assumptions**
- 2 extension of SATMC, a SAT-based model checker for security protocols, to support the analysis of LTL formulae and
- 3 **effectiveness** of the approach assessed through analysis of the ASW protocol: new attack found and key security property refined.

- 1 Introduction
- 2 LTL Model Checking for Security Protocols**
- 3 A Motivating Example: the ASW Protocol
 - The Protocol
 - The Assumptions on the Channels and on the Honest Principals
 - Security Properties
- 4 Modelling, Specification and Analysis using Set-rewriting and LTL
 - Specifying the Protocol Scenario and the Intruder
 - Specifying the Assumptions on the Channels and on the Honest Principals
 - Specifying Security Properties
- 5 Conclusions

Security protocol analysis boils down to building and solving model checking problems of the form:

$$M \models (C_I \wedge C_H) \supset G$$

where

- M : transition system modelling a superset of the behaviours of the honest agents and of the intruder.
- C_I : the allowed behaviours of the intruder.
- C_H : the allowed behaviours of the honest principals.
- G : encoding the security property.

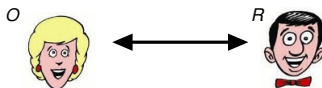
SATMC: SAT-based Model Checking of Security Protocols

- SATMC reduces the problem of determining whether a protocol violates a security property in k steps to SAT.
- By leveraging on state-of-the-art SAT solvers, SATMC can compete with—and in some cases outperform—other state-of-the-art protocol analysers.
- SATMC is a back-end of the AVISPA Tool.
- We have extended SATMC to support model checking of LTL formulae.

- 1 Introduction
- 2 LTL Model Checking for Security Protocols
- 3 A Motivating Example: the ASW Protocol**
 - The Protocol
 - The Assumptions on the Channels and on the Honest Principals
 - Security Properties
- 4 Modelling, Specification and Analysis using Set-rewriting and LTL
 - Specifying the Protocol Scenario and the Intruder
 - Specifying the Assumptions on the Channels and on the Honest Principals
 - Specifying Security Properties
- 5 Conclusions

The ASW Protocol

The ASW protocol consists of three subprotocols.



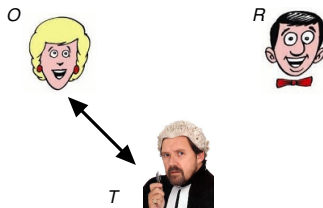
The exchange subprotocol: played by O and R and—if successful—it allows to achieve a mutual commitment on a previously agreed contractual text (**standard contract**).

The abort subprotocol: O can request T to **abort** the previously initiated contract signing procedure.

The resolve subprotocol: O and R can request T to force the **resolution** of the contract, possibly obtaining a **replacement contract**.

The ASW Protocol

The ASW protocol consists of three subprotocols.



The exchange subprotocol: played by O and R and—if successful—it allows to achieve a mutual commitment on a previously agreed contractual text (standard contract).

The abort subprotocol: O can request T to **abort** the previously initiated contract signing procedure.

The resolve subprotocol: O and R can request T to force the **resolution** of the contract, possibly obtaining a **replacement contract**.

The ASW Protocol

The ASW protocol consists of three subprotocols.



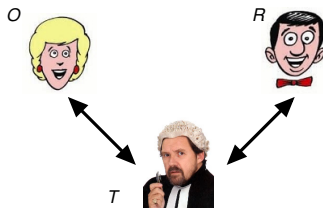
The exchange subprotocol: played by O and R and—if successful—it allows to achieve a mutual commitment on a previously agreed contractual text (**standard contract**).

The abort subprotocol: O can request T to **abort** the previously initiated contract signing procedure.

The resolve subprotocol: O and R can request T to force the **resolution** of the contract, possibly obtaining a **replacement contract**.

The ASW Protocol

The ASW protocol consists of three subprotocols.



The exchange subprotocol: played by O and R and—if successful—it allows to achieve a mutual commitment on a previously agreed contractual text (**standard contract**).

The abort subprotocol: O can request T to **abort** the previously initiated contract signing procedure.

The resolve subprotocol: O and R can request T to force the **resolution** of the contract, possibly obtaining a **replacement contract**.

The Exchange Subprotocol

First round: O and R express their **public commitments** to the contract.

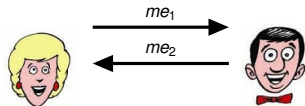
Second round: O and R exchange their **secret commitments**, required for the **standard contract** (i.e. $\{ me_1, N_O, me_2, N_R \}$).

$$E1. \quad O \rightarrow R \quad : \quad me_1 = \text{Sig}_O(V_O, V_R, T, \text{Text}, h(N_O))$$

$$E2. \quad R \rightarrow O \quad : \quad me_2 = \text{Sig}_R(me_1, h(N_R))$$

$$E3. \quad O \rightarrow R \quad : \quad N_O$$

$$E4. \quad R \rightarrow O \quad : \quad N_R$$



The Exchange Subprotocol

First round: O and R express their **public commitments** to the contract.

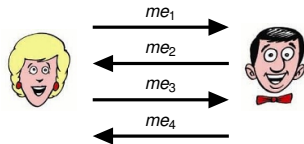
Second round: O and R exchange their **secret commitments**, required for the **standard contract** (i.e. $\{ me_1, N_O, me_2, N_R \}$).

$$E1. \quad O \rightarrow R \quad : \quad me_1 = \text{Sig}_O(V_O, V_R, T, \text{Text}, h(N_O))$$

$$E2. \quad R \rightarrow O \quad : \quad me_2 = \text{Sig}_R(me_1, h(N_R))$$

$$E3. \quad O \rightarrow R \quad : \quad N_O$$

$$E4. \quad R \rightarrow O \quad : \quad N_R$$



The Abort Subprotocol

A1. $O \rightarrow T$: $ma_1 = \text{Sig}_O(\text{aborted}, me_1)$

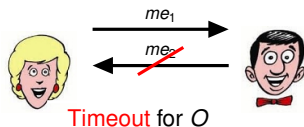
A2. $T \rightarrow O$: **if** $\text{resolved}(me_1, me_2) \in DB$
then $\text{Sig}_T(me_1, me_2)$

// Replacement contract

else $DB := DB \cup \{\text{aborted}(me_1)\};$

$\text{Sig}_T(\text{aborted}, ma_1)$

// Abort token



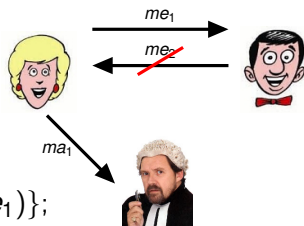
The Abort Subprotocol

A1. $O \rightarrow T$: $ma_1 = \text{Sig}_O(\text{aborted}, me_1)$

A2. $T \rightarrow O$: **if** $\text{resolved}(me_1, me_2) \in DB$
then $\text{Sig}_T(me_1, me_2)$

// Replacement contract
else $DB := DB \cup \{\text{aborted}(me_1)\};$
 $\text{Sig}_T(\text{aborted}, ma_1)$

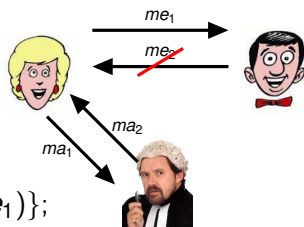
// Abort token



The Abort Subprotocol

A1. $O \rightarrow T$: $ma_1 = \text{Sig}_O(\text{aborted}, me_1)$

A2. $T \rightarrow O$: **if** $\text{resolved}(me_1, me_2) \in DB$
then $\text{Sig}_T(me_1, me_2)$
 // Replacement contract
else $DB := DB \cup \{\text{aborted}(me_1)\};$
 $\text{Sig}_T(\text{aborted}, ma_1)$
 // Abort token



The Resolve Subprotocol (1)

R1. $R \rightarrow T$: $mr_1 = \langle me_1, me_2 \rangle$

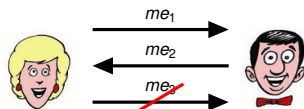
R2. $T \rightarrow R$: **if** $aborted(me_1) \in DB$
then $Sig_T(aborted, ma_1)$

// Abort token

else $DB := DB \cup \{resolved(me_1, me_2)\};$

$Sig_T(me_1, me_2)$

// Replacement contract

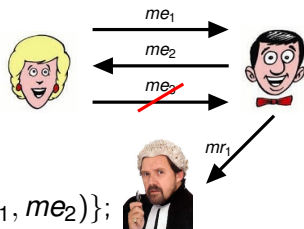


Timeout for R

The Resolve Subprotocol (1)

R1. $R \rightarrow T$: $mr_1 = \langle me_1, me_2 \rangle$

R2. $T \rightarrow R$: **if** $aborted(me_1) \in DB$
then $Sig_T(aborted, ma_1)$
 // Abort token
else $DB := DB \cup \{resolved(me_1, me_2)\};$
 $Sig_T(me_1, me_2)$
 // Replacement contract



The Resolve Subprotocol (1)

R1. $R \rightarrow T$: $mr_1 = \langle me_1, me_2 \rangle$

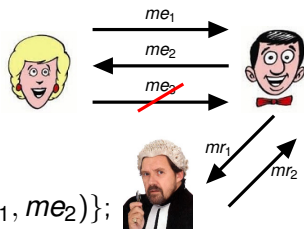
R2. $T \rightarrow R$: **if** $aborted(me_1) \in DB$
then $Sig_T(aborted, ma_1)$

// Abort token

else $DB := DB \cup \{resolved(me_1, me_2)\};$

$Sig_T(me_1, me_2)$

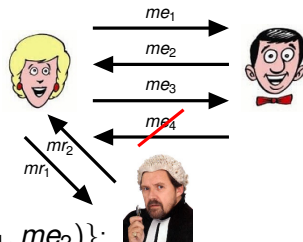
// Replacement contract



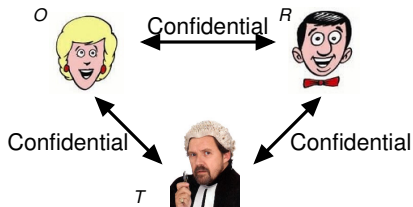
The Resolve Subprotocol (2)

R1. $O \rightarrow T$: $mr_1 = \langle me_1, me_2 \rangle$

R2. $T \rightarrow O$: **if** $aborted(me_1) \in DB$
then $Sig_T(aborted, ma_1)$
 // Abort token
else $DB := DB \cup \{resolved(me_1, me_2)\};$
 $Sig_T(me_1, me_2)$
 // Replacement contract

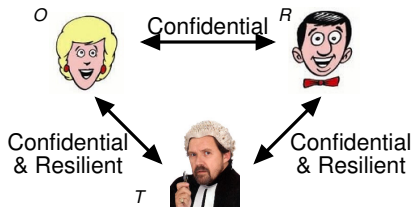


ASW: Assumptions on the Channels



- **Confidential:** eavesdroppers do not have access to the information.
- **Resilient:** any message will be eventually delivered to the intended recipient.

ASW: Assumptions on the Channels



- **Confidential:** eavesdroppers do not have access to the information.
- **Resilient:** any message will be eventually delivered to the intended recipient.

- **Timeout (O and R):** During the exchange sub-protocol, O and R will not indefinitely wait for a reply from the corresponding party and will eventually start either the abort or resolve subprotocol.
- **Availability (T):** T must be always available, i.e. he must eventually process all the messages received by replying to requests by O and R .

Fair exchange can be expressed as the conjunction of the following two properties (slightly adapted from [SM02]):

- (A) A corrupt principal cannot obtain a valid contract without allowing the remaining principal to also obtain a valid contract.
- (B) Once an honest principal obtains an abort token, it is impossible for any other principal to obtain a valid contract.

[SM02] V. Shmatikov and J. C. Mitchell. *Finite-state analysis of two contract signing protocols*. (*Theoretical Computer Science 2002*)

- 1 Introduction
- 2 LTL Model Checking for Security Protocols
- 3 A Motivating Example: the ASW Protocol
 - The Protocol
 - The Assumptions on the Channels and on the Honest Principals
 - Security Properties
- 4 Modelling, Specification and Analysis using Set-rewriting and LTL**
 - Specifying the Protocol Scenario and the Intruder**
 - Specifying the Assumptions on the Channels and on the Honest Principals**
 - Specifying Security Properties**
- 5 Conclusions

$$M \models (C_I \wedge C_H) \supset G$$

Transition system associated with the concurrent execution of a number of sessions of the protocol.

- States: sets of **facts**, i.e. ground atomic formulae
- Transitions: **rewrite rules** that define mappings between sets of facts.

If S is a set of facts, then we interpret the facts in S as the propositions holding in the state represented by S , all other facts being false in that state (closed-world assumption).

The Model: Facts

Fact	Meaning
$\mathbf{state}_{Role}(j, a, es, s)$	Principal a , playing role $Role$, is ready to execute step j in session s of the protocol, and es is a list of expressions representing the internal state of a and thus affecting her future behaviour.
$\mathbf{ak}(a, m)$	Principal a knows message m .
$\mathbf{sent}(rs, b, a, m, c)$	Principal rs has sent message m on channel c to principal a pretending to be principal b .
$\mathbf{rcvd}(a, b, m, c)$	Message m (supposedly sent by principal b) has been received on channel c by principal a , but a has not processed it yet.

Note: $\mathbf{ik}(m)$ abbreviates $\mathbf{ak}(i, m)$.

The Model: Rules for the Honest Agents

Delivery of messages:

sent(RS, B, A, M, C)

$\xrightarrow{\text{receive}(A,B,RS,M,C)}$

rcvd(A, B, M, C)
.ak(A, M)

Processing a previously received message:

rcvd(A, B, M, C)

.state_{Role}(j, A, es, S)

$\xrightarrow{\text{send}_j(A,B,B1,\dots,S)}$

sent(A, A, B1, M1, C1)

.state_{Role}(l, A, es', S)

The Model: Rules for the Honest Agents

Delivery of messages:

sent(RS, B, A, M, C)

$\xrightarrow{\text{receive}(A,B,RS,M,C)}$

rcvd(A, B, M, C)
.ak(A, M)

Processing a previously received message:

rcvd(A, B, M, C)

.state_{Role}(j, A, es, S)

$\xrightarrow{\text{send}_j(A,B,B1,\dots,S)}$

sent(A, A, B1, M1, C1)
.state_{Role}(l, A, es', S)

The Model: Rules for the Intruder


$$\mathbf{sent}(A, A, B, M, C) \xrightarrow{\text{intercept}(A,B,M,C)} \mathbf{rcvd}(i, A, M, C) . \mathbf{ik}(M)$$
$$\mathbf{sent}(A, A, B, M, C) \xrightarrow{\text{overhear}(A,B,M,C)} \mathbf{sent}(A, A, B, M, C) . \mathbf{rcvd}(i, A, M, C) . \mathbf{ik}(M)$$
$$\mathbf{ik}(M) . \mathbf{ik}(A) . \mathbf{ik}(B) \xrightarrow{\text{fake}(A,B,M,C)} \mathbf{sent}(i, A, B, M, C) . \mathbf{ik}(M) . \mathbf{ik}(A) . \mathbf{ik}(B)$$

The Model: Inferential Capabilities of the Agents

$$\begin{aligned} & \mathbf{ak}(A, M) \cdot \mathbf{ak}(A, K) \xrightarrow{\text{encrypt}(A, K, M)} \mathbf{ak}(A, M) \cdot \mathbf{ak}(A, K) \cdot \mathbf{ak}(A, \{M\}_K) \\ \mathbf{ak}(A, \{M\}_K) \cdot \mathbf{ak}(A, K^{-1}) & \xrightarrow{\text{decrypt_puk}(A, K, M)} \mathbf{ak}(A, \{M\}_K) \cdot \mathbf{ak}(A, K^{-1}) \cdot \mathbf{ak}(A, M) \\ \mathbf{ak}(A, \{M\}_{K^{-1}}) \cdot \mathbf{ak}(A, K) & \xrightarrow{\text{decrypt_prk}(A, K, M)} \mathbf{ak}(A, \{M\}_{K^{-1}}) \cdot \mathbf{ak}(A, K) \cdot \mathbf{ak}(A, M) \\ \mathbf{ak}(A, M_1) \cdot \mathbf{ak}(A, M_2) & \xrightarrow{\text{pairing}(A, M_1, M_2)} \mathbf{ak}(A, M_1) \cdot \mathbf{ak}(A, M_2) \cdot \mathbf{ak}(A, \langle M_1, M_2 \rangle) \\ \mathbf{ak}(A, \langle M_1, M_2 \rangle) & \xrightarrow{\text{decompose}(A, M_1, M_2)} \mathbf{ak}(A, \langle M_1, M_2 \rangle) \cdot \mathbf{ak}(A, M_1) \cdot \mathbf{ak}(A, M_2) \end{aligned}$$

Constraining the Behaviour of the Intruder

$$M \models (C_I \wedge C_H) \supset G$$

- The **confidentiality** of channel c to a set of principals PS is formalised by:

$$\mathbf{G} \forall (\mathbf{rcvd}(B, A, M, c) \supset \bigvee_{p \in PS} B = p)$$

- The **resilience** of channel c is formalised by:

$$\mathbf{G} \forall (\mathbf{sent}(A', A, B, M, c) \supset \mathbf{F} \mathbf{rcvd}(B, A, M, c))$$

- ...

Constraining the Behaviour of Honest Principals

$$M \models (C_I \wedge C_H) \supset G$$

- O and R should not indefinitely wait for an answer:

$$\mathbf{G} \forall (\text{state}_{\text{Role}}(j, \dots) \supset \mathbf{F} \neg \text{state}_{\text{Role}}(j, \dots))$$

where j identifies the states from which a time-out can occur.

- Messages received will be eventually processed by T

$$\mathbf{G} \forall (\text{rcvd}(t, P, M, C) \supset \mathbf{F} \neg \text{rcvd}(t, P, M, C))$$

$$M \models (C_I \wedge C_H) \supset G$$

Fair Exchange

- (A) A corrupt principal cannot obtain a valid contract without allowing the remaining principal to also obtain a valid contract.
- (B) Once an honest principal obtains an abort token, it is impossible for any other principal to obtain a valid contract.

$$M \models (C_I \wedge C_H) \supset G_A$$

Fair Exchange

- (A) A corrupt principal cannot obtain a valid contract without allowing the remaining principal to also obtain a valid contract.
- (B) Once an honest principal obtains an abort token, it is impossible for any other principal to obtain a valid contract.

$$M \models (C_I \wedge C_H) \supset G_B$$

Fair Exchange

- (A) A corrupt principal cannot obtain a valid contract without allowing the remaining principal to also obtain a valid contract.
- (B) Once an honest principal obtains an abort token, it is impossible for any other principal to obtain a valid contract.

Specifying and Analysing Fair Exchange (A)

Fair Exchange (A): Specification

$$M \models (C_I \wedge C_H) \supset G_A$$

(A) A corrupt principal cannot obtain a valid contract without allowing the remaining principal to also obtain a valid contract.

G_A is the conjunction, for each session considered, of all formulae of the form:

$$\mathbf{G} \forall (hasvc(o, o, r, txt, N_O, N_R, t) \supset \mathbf{F} hasvc(r, o, r, txt, N_O, N_R, t))$$

If o has a valid contract, binding r to txt using N_O and N_R as secret commitments and t as T , then eventually r will have a corresponding valid contract.

Fair Exchange (A): Specification

$$M \models (C_I \wedge C_H) \supset G_A$$

(A) A corrupt **principal** cannot obtain a valid contract without allowing the remaining principal to also obtain a valid contract.

G_A is the conjunction, for each session considered, of all formulae of the form:

$$\mathbf{G} \forall (\text{hasvc}(o, o, r, \text{txt}, N_O, N_R, t) \supset \mathbf{F} \text{hasvc}(r, o, r, \text{txt}, N_O, N_R, t))$$

If o has a valid contract, binding r to txt using N_O and N_R as secret commitments and t as T , then eventually r will have a corresponding valid contract.

Fair Exchange (A): Specification

$$M \models (C_I \wedge C_H) \supset G_A$$

(A) A corrupt principal cannot obtain a valid contract without allowing the **remaining principal to also obtain a valid contract**.

G_A is the conjunction, for each session considered, of all formulae of the form:

$$\mathbf{G} \forall (hasvc(o, o, r, txt, N_O, N_R, t) \supset \mathbf{F} hasvc(r, o, r, txt, N_O, N_R, t))$$

If o has a valid contract, binding r to txt using N_O and N_R as secret commitments and t as T , then **eventually r will have a corresponding valid contract**.

Fair Exchange (A): Specification

$hasvc(P, O, R, Txt, N_O, N_R, T) :=$

// P has a standard contract, i.e. he knows me_1 , me_2 , N_O , and N_R

$ak(P, me_1(O, R, Txt, N_O, T)) \wedge$

$ak(P, me_2(O, R, Txt, N_O, N_R, T)) \wedge$

$ak(P, N_O) \wedge$

$ak(P, N_R)$

\vee

// P has a replacement contract, i.e. he knows $Sig_T(me_1, me_2)$

$ak(P, Sig_T(me_1(O, R, Txt, N_O, T), me_2(O, R, Txt, N_O, N_R, T)))$

Fair Exchange (A): Specification

$hasvc(P, O, R, Txt, N_O, N_R, T) :=$

// P has a standard contract, i.e. he knows me_1 , me_2 , N_O , and N_R

$ak(P, me_1(O, R, Txt, N_O, T)) \wedge$

$ak(P, me_2(O, R, Txt, N_O, N_R, T)) \wedge$

$ak(P, N_O) \wedge$

$ak(P, N_R)$

\vee

// P has a replacement contract, i.e. he knows $Sig_T(me_1, me_2)$

$ak(P, Sig_T(me_1(O, R, Txt, N_O, T), me_2(O, R, Txt, N_O, N_R, T)))$

Fair Exchange (A): Specification

$hasvc(P, O, R, Txt, N_O, N_R, T) :=$

// P has a standard contract, i.e. he knows me_1 , me_2 , N_O , and N_R

[$ak(P, me_1(O, R, Txt, N_O, T)) \wedge$
 $ak(P, me_2(O, R, Txt, N_O, N_R, T)) \wedge$
 $ak(P, N_O) \wedge$
 $ak(P, N_R)$]

\vee

// P has a replacement contract, i.e. he knows $Sig_T(me_1, me_2)$
 $ak(P, Sig_T(me_1(O, R, Txt, N_O, T), me_2(O, R, Txt, N_O, N_R, T)))$

Fair Exchange (A): Analysis

- SATMC finds the following attack:

$E1. O \rightarrow I : me_1$

$E2. I \rightarrow O : me_2$

I computes new random N'_I and then me'_2

$E3. O \rightarrow I : N_O$

$E4. I \rightarrow O : N_I$

- At the end of the protocol, the intruder owns a standard contract $\{me_1, me'_2, N_O, N'_I\}$ while O owns only the standard contract $\{me_1, me_2, N_O, N_I\}$.
- This attack is similar in spirit to the attack in [SM02], but it is simpler as T is not involved.

Fair Exchange (A): Analysis

- SATMC finds the following attack:

$E1.$ $O \rightarrow I$: me_1
 $E2.$ $I \rightarrow O$: me_2
 I computes new random N'_I and then me'_2
 $E3.$ $O \rightarrow I$: N_O
 $E4.$ $I \rightarrow O$: N_I

- At the end of the protocol, the intruder owns a standard contract $\{me_1, me'_2, N_O, N'_I\}$ while O owns only the standard contract $\{me_1, me_2, N_O, N_I\}$.
- This attack is similar in spirit to the attack in [SM02], but it is simpler as T is not involved.

Fair Exchange (A): Analysis of the Patched Version

[SM02] propose to repair ASW by replacing steps $E3$ and $E4$

$E3. O \rightarrow R : N_O$

$E4. R \rightarrow O : N_R$

SATMC confirms that the improved version of the protocol does not suffer from the previous attack, but it detects a new (i.e. previously unknown) attack:

$E1. O \rightarrow I : me_1$

$E2. I \rightarrow O : me_2$

I computes new random N'_I and then me'_2

$E3'. O \rightarrow I : Sig_O(N_O, h(N_I))$

$E4'. I \rightarrow O : Sig_I(N_I, h(N_O))$

$R1. I \rightarrow T : mr_1 = \langle me_1, me'_2 \rangle$

$R2. T \rightarrow I : mr_2 = Sig_T(me_1, me'_2)$

The **intruder**, here playing the responder, obtains a replacement contract relative to the secret commitments N_O and N'_I , while O obtains only a valid contract relative to the secret commitments N_O and N_I .

Fair Exchange (A): Analysis of the Patched Version

[SM02] propose to repair ASW by replacing steps $E3$ and $E4$ with:

$$E3'. O \rightarrow R : \text{Sig}_O(N_O, h(N_R))$$

$$E4'. R \rightarrow O : \text{Sig}_R(N_R, h(N_O))$$

SATMC confirms that the improved version of the protocol does not suffer from the previous attack, but it detects a new (i.e. previously unknown) attack:

$$E1. O \rightarrow I : me_1$$

$$E2. I \rightarrow O : me_2$$

I computes new random N'_I and then me'_2

$$E3'. O \rightarrow I : \text{Sig}_O(N_O, h(N_I))$$

$$E4'. I \rightarrow O : \text{Sig}_I(N_I, h(N_O))$$

$$R1. I \rightarrow T : mr_1 = \langle me_1, me'_2 \rangle$$

$$R2. T \rightarrow I : mr_2 = \text{Sig}_T(me_1, me'_2)$$

The **intruder**, here playing the responder, obtains a replacement contract relative to the secret commitments N_O and N'_I , while O obtains only a valid contract relative to the secret commitments N_O and N_I .

Fair Exchange (A): Analysis of the Patched Version

[SM02] propose to repair ASW by replacing steps $E3$ and $E4$ with:

$$E3'. O \rightarrow R : \text{Sig}_O(N_O, h(N_R))$$

$$E4'. R \rightarrow O : \text{Sig}_R(N_R, h(N_O))$$

SATMC confirms that the improved version of the protocol does not suffer from the previous attack, but it detects a new (i.e. previously unknown) attack:

$$E1. O \rightarrow I : me_1$$

$$E2. I \rightarrow O : me_2$$

I computes new random N'_I and then me'_2

$$E3'. O \rightarrow I : \text{Sig}_O(N_O, h(N_I))$$

$$E4'. I \rightarrow O : \text{Sig}_I(N_I, h(N_O))$$

$$R1. I \rightarrow T : mr_1 = \langle me_1, me'_2 \rangle$$

$$R2. T \rightarrow I : mr_2 = \text{Sig}_T(me_1, me'_2)$$

The **intruder**, here playing the responder, obtains a replacement contract relative to the secret commitments N_O and N'_I , while O obtains only a valid contract relative to the secret commitments N_O and N_I .

Fair Exchange (A): Analysis of the Patched Version

[SM02] propose to repair ASW by replacing steps $E3$ and $E4$ with:

$$E3'. O \rightarrow R : \text{Sig}_O(N_O, h(N_R))$$

$$E4'. R \rightarrow O : \text{Sig}_R(N_R, h(N_O))$$

SATMC confirms that the improved version of the protocol does not suffer from the previous attack, but it detects a new (i.e. previously unknown) attack:

$$E1. O \rightarrow I : me_1$$

$$E2. I \rightarrow O : me_2$$

I computes new random N'_I and then me'_2

$$E3'. O \rightarrow I : \text{Sig}_O(N_O, h(N_I))$$

$$E4'. I \rightarrow O : \text{Sig}_I(N_I, h(N_O))$$

$$R1. I \rightarrow T : mr_1 = \langle me_1, me'_2 \rangle$$

$$R2. T \rightarrow I : mr_2 = \text{Sig}_T(me_1, me'_2)$$

The **intruder**, here playing the responder, **obtains a replacement contract relative to the secret commitments N_O and N'_I** , while O obtains only a valid contract relative to the secret commitments N_O and N_I .

Fair Exchange (A): Analysis of the Patched Version

[SM02] propose to repair ASW by replacing steps $E3$ and $E4$ with:

$$E3'. O \rightarrow R : \text{Sig}_O(N_O, h(N_R))$$

$$E4'. R \rightarrow O : \text{Sig}_R(N_R, h(N_O))$$

SATMC confirms that the improved version of the protocol does not suffer from the previous attack, but it detects a new (i.e. previously unknown) attack:

$$E1. O \rightarrow I : me_1$$

$$E2. I \rightarrow O : me_2$$

I computes new random N'_I and then me'_2

$$E3'. O \rightarrow I : \text{Sig}_O(N_O, h(N_I))$$

$$E4'. I \rightarrow O : \text{Sig}_I(N_I, h(N_O))$$

$$R1. I \rightarrow T : mr_1 = \langle me_1, me'_2 \rangle$$

$$R2. T \rightarrow I : mr_2 = \text{Sig}_T(me_1, me'_2)$$

The **intruder**, here playing the responder, obtains a replacement contract relative to the secret commitments N_O and N'_I , while **O obtains only a valid contract relative to the secret commitments N_O and N_I .**

Fair Exchange (A): Analysis of the Patched Version

- Also this new attack **does not have serious consequences**: O and R do not have the same contract, but they have a contract for the same text.
- Yet the ability to detect this attack, which has eluded previous formal analyses of the protocol, gives evidence of the **effectiveness** of the approach.
- We have come to the conclusion that the problem is not in the protocol but in the property, which requires the contract to be relative to the same secret commitment N_R .
- We have therefore weakened the property accordingly and successfully checked it with SATMC.

Specifying and Analysing Fair Exchange (B)

Fair Exchange (B): Specification

$$M \models (C_I \wedge C_H) \supset G_B$$

(B) Once an honest principal obtains an abort token, it is impossible for any other principal to obtain a valid contract.

We rephrase the property and express it in LTL:

(B') If an honest principal obtains an abort token, then any other principal has already a corresponding valid contract or will not be able to obtain one in the future.

Fair Exchange (B): Analysis

SATMC finds the following attack:

$E1. I \rightarrow R : me_1$
 $E2. R \rightarrow I : me_2$
 $A1. I \rightarrow T : ma_1 = Sig_I(aborted, me_1)$
 $A2. T \rightarrow I : ma_2 = Sig_T(aborted, ma_1)$
 $R1. R \rightarrow T : mr_1 = \langle me_1, me_2 \rangle$
 $R2. T \rightarrow R : mr_2 = Sig_T(aborted, ma_1)$
 $E1. I \rightarrow R : me_1$
 $E2. R \rightarrow I : me'_2$
 $E3. I \rightarrow R : N_I$
 $E4. R \rightarrow I : N'_R$

R obtains an **abort token** relative to the secret commitment N_I and eventually the **intruder**, playing the originator, obtains a standard contract relative to the same contractual text and secret commitment.

Fair Exchange (B): Analysis

SATMC finds the following attack:

$E1. I \rightarrow R : me_1$
 $E2. R \rightarrow I : me_2$
 $A1. I \rightarrow T : ma_1 = Sig_I(aborted, me_1)$
 $A2. T \rightarrow I : ma_2 = Sig_T(aborted, ma_1)$
 $R1. R \rightarrow T : mr_1 = \langle me_1, me_2 \rangle$
 $R2. T \rightarrow R : mr_2 = Sig_T(aborted, ma_1)$
 $E1. I \rightarrow R : me_1$
 $E2. R \rightarrow I : me'_2$
 $E3. I \rightarrow R : N_I$
 $E4. R \rightarrow I : N'_R$

R obtains an abort token relative to the secret commitment N_I and eventually the intruder, playing the originator, obtains a **standard contract** relative to the same contractual text and secret commitment.

Fair Exchange (B): Analysis

SATMC finds the following attack:

$E1. I \rightarrow R : me_1$
 $E2. R \rightarrow I : me_2$
 $A1. I \rightarrow T : ma_1 = Sig_I(aborted, me_1)$
 $A2. T \rightarrow I : ma_2 = Sig_T(aborted, ma_1)$
 $R1. R \rightarrow T : mr_1 = \langle me_1, me_2 \rangle$
 $R2. T \rightarrow R : mr_2 = Sig_T(aborted, ma_1)$
 $E1. I \rightarrow R : me_1 //$
 $E2. R \rightarrow I : me'_2 //$ **R has the**
 $E3. I \rightarrow R : N_I //$ **same contract**
 $E4. R \rightarrow I : N'_R //$

R obtains an abort token relative to the secret commitment N_I and eventually the intruder, playing the originator, obtains a standard contract relative to the same contractual text and secret commitment.

Fair Exchange (B): Specification and Analysis

- Again the problem lies in the formulation of the security property and not in the protocol.
- In fact R eventually obtains the same standard contract obtained by the intruder.
- We rectify the problem by reformulating the goal as follows:

(B'') If an honest principal, say A , obtains an abort token, then any other principal has already a corresponding valid contract or will not be able to obtain one in the future, **or A already owns or eventually will obtain a corresponding valid contract.**

No attack is found by SATMC while checking the protocol w.r.t. (B'').

- 1 Introduction
- 2 LTL Model Checking for Security Protocols
- 3 A Motivating Example: the ASW Protocol
 - The Protocol
 - The Assumptions on the Channels and on the Honest Principals
 - Security Properties
- 4 Modelling, Specification and Analysis using Set-rewriting and LTL
 - Specifying the Protocol Scenario and the Intruder
 - Specifying the Assumptions on the Channels and on the Honest Principals
 - Specifying Security Properties
- 5 **Conclusions**

Conclusions

- We have presented a **general framework for security protocols** (set-rewriting + LTL) that allows for the specification of:
 - assumptions on principals and channels
 - complex security propertiesthat are normally not handled by state-of-the-art analysers.
- We have demonstrated the **effectiveness** of the approach. By analysing the **ASW protocol** we have:
 - found all known attacks + a new attack on the “patched” version
 - refined/revised the specification of fair exchange.
- Our analysis shows that the use of LTL
 - is **not only** useful for specifying and supporting the mechanical verification of security protocols,
 - **but also** to support the understanding and the specification of their security requirements.

Conclusions

- We have presented a **general framework for security protocols** (set-rewriting + LTL) that allows for the specification of:
 - assumptions on principals and channels
 - complex security propertiesthat are normally not handled by state-of-the-art analysers.
- We have demonstrated the **effectiveness** of the approach. By analysing the **ASW protocol** we have:
 - found all known attacks + a new attack on the “patched” version
 - refined/revised the specification of fair exchange.
- Our analysis shows that the use of LTL
 - is **not only** useful for specifying and supporting the mechanical verification of security protocols,
 - **but also** to support the understanding and the specification of their security requirements.

Conclusions

- We have presented a **general framework for security protocols** (set-rewriting + LTL) that allows for the specification of:
 - assumptions on principals and channels
 - complex security propertiesthat are normally not handled by state-of-the-art analysers.
- We have demonstrated the **effectiveness** of the approach. By analysing the **ASW protocol** we have:
 - found all known attacks + a new attack on the “patched” version
 - refined/revised the specification of fair exchange.
- Our analysis shows that the use of LTL
 - is **not only** useful for specifying and supporting the mechanical verification of security protocols,
 - **but also** to support the understanding and the specification of their security requirements.