Automated Security Analysis of Cryptographic Protocols

Alessandro Armando

Security & Trust Research Unit

Centro per le Tecnologie dell'Informazione

Fondazione Bruno Kessler

SECURITY & TRUST

FONDAZIONE BRUNO KESSLER

Outline

Definitions

- A protocol consists of a set of rules (conventions) that determine the exchange of messages between two or more principals.
  In short, a distributed algorithm with emphasis on communication.
- Cryptographic (or security) protocols use cryptographic mechanisms to achieve security objectives, e.g.
  - entity or message authentication,
  - key establishment,
  - timeliness,
  - non-repudiation,
  - fair exchange, ...
- Small recipes, but nontrivial to design and understand.

Communication

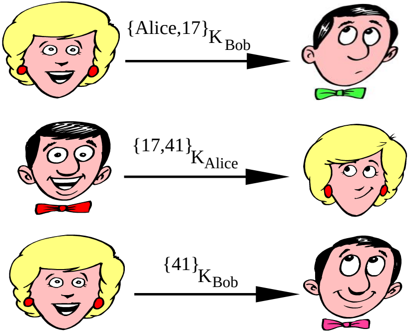
- Fundamental event is communication between principals.

$$A \rightarrow B : \{A, T_A, K_{AB}\}_{K_B}$$

- *A* and *B* name roles.
  Can be instantiated by any principal playing in the role.
- Communication is asynchronous (depending on semantic model).
- Sender/receiver names “$A \rightarrow B$” are not part of the message.
- Protocol specifies actions of principals.
  Equivalently, protocol defines a set of event sequences (traces).

$$
\begin{array}{lll}
1. & A \rightarrow B: & \{A, N_A\}_{K_B} \\
2. & B \rightarrow A: & \{N_A, N_B\}_{K_A} \\
3. & A \rightarrow B: & \{N_B\}_{K_B}
\end{array}
$$

Here is an instance (a protocol run):



How protocol is executed

1. $A \to B: \{A, N_A\}_{K_B}$
2. $B \to A: \{N_A, N_B\}_{K_A}$
3. $A \to B: \{N_B\}_{K_B}$

Each principal executes a "protocol automaton", e.g., Alice in role A.

State s_1:
- Generate nonce N_{Alice}, concatenate to name, and encrypt with K_{Bob}.
- Send $\{Alice, N_{Alice}\}_{K_{Bob}}$ to Bob.
- Goto state s_2.

State s_2:
- Receive message C and decrypt it: $M = \{C\}_{K_{Alice}^{-1}}$.
- If M is not of the form $\{N_{Alice}, X\}$ for some nonce X, then goto reject state else goto state s_3.

State s_3: ...

State reject: terminate with failure.

N.B. principals can be engaged in multiple runs.

Assumptions and Goals

- **Assumptions:** Implicit (or explicit) prerequisites.
  - Principals know their private keys and public keys of others.
  - Principals can generate nonces.
- **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).

Theses:

- A protocol without clear goals (and assumptions) is useless.
- A protocol without a proof of correctness is probably wrong.

- **Assumptions:** Implicit (or explicit) prerequisites.
    - Principals know their private keys and public keys of others.
    - Principals can generate nonces.
- **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).

Theses:

- A protocol without clear goals (and assumptions) is useless.
- A protocol without a proof of correctness is probably wrong.

Assumptions: Attacker

How do we model the attacker? Possibilities:

- He knows the protocol but cannot break crypto. (Standard)
- He is passive but overhears all communications.
- He is active and can intercept and generate messages.

  “Transfer $20 to Bob” $\rightsquigarrow$ “Transfer $10,000 to Charlie”
- He might even be one of the principals running the protocol!

A friend’s just an enemy in disguise. You can’t trust nobody.
(Charles Dickens, Oliver Twist)

Standard Attacker Model (Dolev & Yao)

- The attacker is active. Namely:
  - He can intercept and read all messages.
  - He can decompose messages into their parts.
    But cryptography is secure: decryption requires inverse keys.
  - He can build new messages with the different constructors.
  - He can send messages at any time.
- Sometimes called the Dolev-Yao attacker model.

$\Downarrow$

correct protocols function in the largest range of environments.

Kinds of attack

- Replay (or freshness) attack: reuse parts of previous messages.
- Man-in-the-middle (or parallel sessions) attack: $A \leftrightarrow \mathcal{M} \leftrightarrow B$.
- Masquerading attack: pretend to be another principal, e.g.
  - $\mathcal{M}$ forges source address (e.g., present in network protocols), or
  - $\mathcal{M}$ convinces other principals that A's public key is $K_{\mathcal{M}}$.
- Type flaw attack: substitute a different type of message field. Example: use a name (or a key or ...) as a nonce.
- Reflection attack send transmitted information back to originator.

Problem with NSPK

1. $A \to B : \{A, N_A\}_{K_B}$
2. $B \to A : \{N_A, N_B\}_{K_A}$
3. $A \to B : \{N_B\}_{K_B}$

- **Goal:** mutual (entity) authentication.
- Recall principals can be involved in multiple runs.
  Goal should hold in all interleaved protocol runs.
- Correctness argument (informal).
  1. This is Alice and I have chosen a nonce N_{Alice}.
  2. Here is your Nonce N_{Alice}. Since I could read it, I must be Bob. I also have a challenge N_{Bob} for you.
  3. You sent me N_{Bob}. Since only Alice can read this and I sent it back, I must be Alice.

Protocol proposed in 1970s and used for decades.

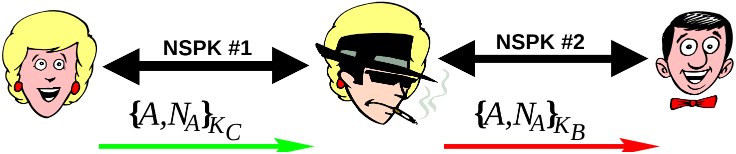
Attack on NSPK (details)

B believes he is speaking with *A*!

Attack on NSPK (details)

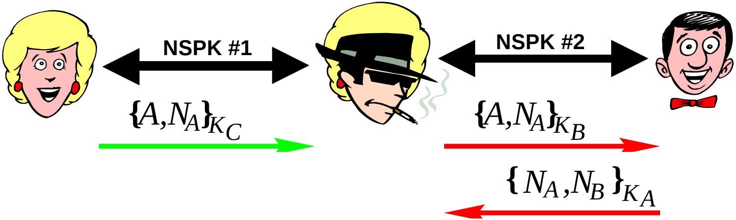
B believes he is speaking with *A*!

Attack on NSPK (details)



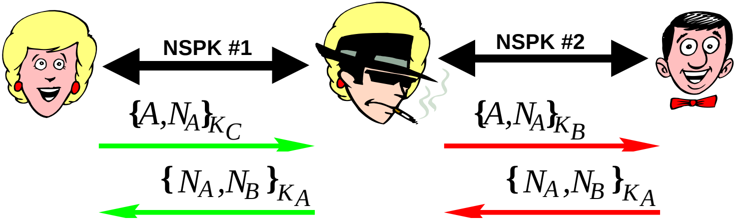
B believes he is speaking with *A*!

Attack on NSPK (details)

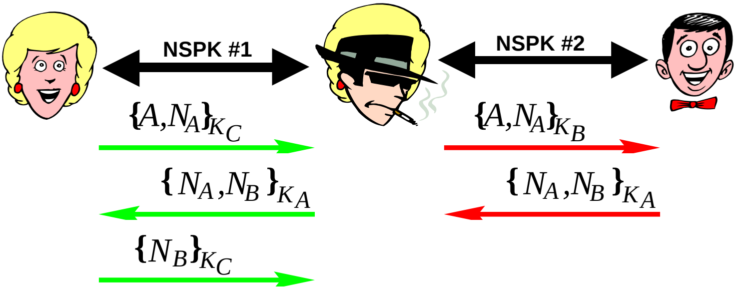


B believes he is speaking with *A*!

Attack on NSPK (details)

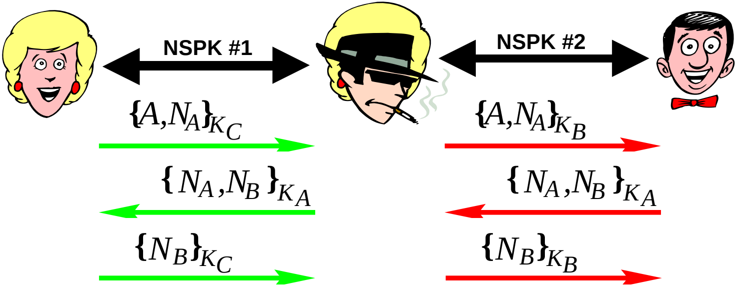


B believes he is speaking with *A*!



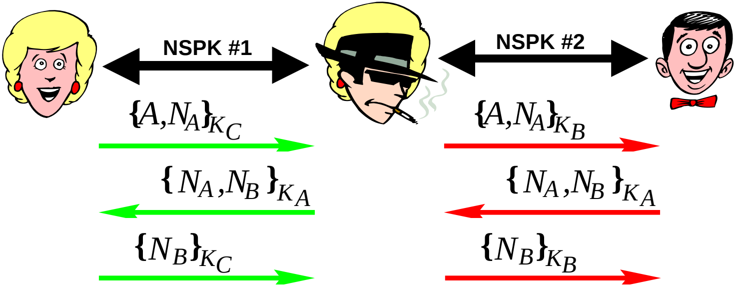
B believes he is speaking with *A*!

Attack on NSPK (details)



B believes he is speaking with *A*!

Attack on NSPK (details)



B believes he is speaking with *A*!

Outline

We consider model-checking problems of the form:

$$(P_1 \| \cdots \| P_n \| I) \models (C \Rightarrow G)$$

- $P_1, \ldots, P_n$: the honest participants.
- I: the DY intruder.
- C: LTL formula constraining the behaviours of the DY intruder on the communication channels.
- G: LTL formula encoding the expected security properties.

By LTL we mean propositional LTL with future (i.e. **G**, **F**, **X**) and past (i.e. **H**, **O**, **Y**) operators.

Formalisation of the SAML SSO

We consider model-checking problems of the form:

$$(P_1 \| \cdots \| P_n \| I) \models (C \Rightarrow G)$$

- $P_1, \ldots, P_n$: the honest participants.
- I: the DY intruder.
- C: LTL formula constraining the behaviours of the DY intruder on the communication channels.
- G: LTL formula encoding the expected security properties.

By LTL we mean propositional LTL with future (i.e. **G**, **F**, **X**) and past (i.e. **H**, **O**, **Y**) operators.

We consider model-checking problems of the form:

$$(P_1 \| \cdots \| P_n \| I) \models (C \Rightarrow G)$$

- $P_1, \ldots, P_n$: the honest participants.
- I: the DY intruder.
- C: LTL formula constraining the behaviours of the DY intruder on the communication channels.
- G: LTL formula encoding the expected security properties.

By LTL we mean propositional LTL with future (i.e. **G**, **F**, **X**) and past (i.e. **H**, **O**, **Y**) operators.

Formalisation of the SAML SSO

We consider model-checking problems of the form:

$$(P_1 \| \cdots \| P_n \| I) \models (C \Rightarrow G)$$

- $P_1, \ldots, P_n$: the honest participants.
- I: the DY intruder.
- C: LTL formula constraining the behaviours of the DY intruder on the communication channels.
- G: LTL formula encoding the expected security properties.

By LTL we mean propositional LTL with future (i.e. **G**, **F**, **X**) and past (i.e. **H**, **O**, **Y**) operators.

We consider model-checking problems of the form:

$$(P_1 \| \cdots \| P_n \| I) \models (C \Rightarrow G)$$

- $P_1, \ldots, P_n$: the honest participants.
- I: the DY intruder.
- C: LTL formula constraining the behaviours of the DY intruder on the communication channels.
- G: LTL formula encoding the expected security properties.

By LTL we mean propositional LTL with future (i.e. **G**, **F**, **X**) and past (i.e. **H**, **O**, **Y**) operators.

Formalisation of the SAML SSO

We consider model-checking problems of the form:

$$(P_1 \| \cdots \| P_n \| I) \models (C \Rightarrow G)$$

- $P_1, \ldots, P_n$: the honest participants.
- I: the DY intruder.
- C: LTL formula constraining the behaviours of the DY intruder on the communication channels.
- G: LTL formula encoding the expected security properties.

By LTL we mean propositional LTL with future (i.e. **G**, **F**, **X**) and past (i.e. **H**, **O**, **Y**) operators.

Modeling the Honest Participants

$$(P_1 \| \cdots \| P_n \| I) \models (C \Rightarrow G)$$

Processes associated concurrently executing a number of sessions of the protocol.

- **States:** sets of facts, i.e. ground atomic formulae
- **Transitions:** rewrite rules defining the allowed behaviours.

Modeling the Honest Participants

Fact	Meaning
$\mathtt{state}_{Role}(j, a, es, s)$	Principal a, playing role *Role*, is ready to execute step j in session s of the protocol.
$\mathtt{ik}(m)$	The intruder knows message m.
$\mathtt{sent}(rs, b, a, m, c)$	Principal rs has sent message m on channel c to principal a pretending to be principal b.
$\mathtt{rcvd}(a, b, m, c)$	Message m (supposedly sent by principal b) has been received on channel c by principal a

Example (State):

$$\mathtt{state}_{Init}(\mathtt{2}, \mathtt{a}, [\mathtt{ka}, \mathtt{ka}^{-1}, \mathtt{kb}], \mathtt{1}) \,.\, \mathtt{sent}(\mathtt{a}, \mathtt{a}, \mathtt{i}, \{\langle \mathtt{a}, \mathtt{na}\rangle\}_{\mathtt{ki}}, \mathtt{c}) \\ .\, \mathtt{state}_{Resp}(\mathtt{1}, \mathtt{b}, [\mathtt{kb}, \mathtt{kb}^{-1}, \mathtt{ka}], \mathtt{1}) \,.\, \mathtt{ik}(\mathtt{ka}) \,.\, \mathtt{ik}(\mathtt{kb})$$

Modeling the Intruder

$$(P_1 \| \cdots \| P_n \| I) \models (C \Rightarrow G)$$

Interception

$$\texttt{sent}(\texttt{A},\texttt{A},\texttt{B},\texttt{M},\texttt{C}) \xrightarrow{\textbf{intercept}(\texttt{A,B,M,C})} \texttt{rcvd}(\texttt{i},\texttt{A},\texttt{M},\texttt{C}).\texttt{ik}(\texttt{M})$$

Overhearing

$$\texttt{sent}(\texttt{A},\texttt{A},\texttt{B},\texttt{M},\texttt{C}) \xrightarrow{\textbf{overhear}(\texttt{A,B,M,C})} \texttt{sent}(\texttt{A},\texttt{A},\texttt{B},\texttt{M},\texttt{C}).\texttt{rcvd}(\texttt{i},\texttt{A},\texttt{M},\texttt{C}).\texttt{ik}(\texttt{M})$$

Faking

$$\texttt{ik}(\texttt{M}).\texttt{ik}(\texttt{A}).\texttt{ik}(\texttt{B}) \xrightarrow{\textbf{fake}(\texttt{A,B,M,C})} \texttt{sent}(\texttt{i},\texttt{A},\texttt{B},\texttt{M},\texttt{C}).\texttt{ik}(\texttt{M}).\texttt{ik}(\texttt{A}).\texttt{ik}(\texttt{B})$$

The Model: Inferential Capabilities of the Intruder

$$\mathtt{ak}(\mathtt{M}).\mathtt{ak}(\mathtt{K}) \xrightarrow{\mathbf{encrypt}(\mathtt{A},\mathtt{K},\mathtt{M})} \mathtt{ak}(\mathtt{M}).\mathtt{ak}(\mathtt{K}).\mathtt{ak}(\{\mathtt{M}\}_{\mathtt{K}})$$

$$\mathtt{ak}(\{\mathtt{M}\}_{\mathtt{K}}).\mathtt{ak}(\mathtt{K}^{-1}) \xrightarrow{\mathbf{decrypt\_puk}(\mathtt{A},\mathtt{K},\mathtt{M})} \mathtt{ak}(\{\mathtt{M}\}_{\mathtt{K}}).\mathtt{ak}(\mathtt{K}^{-1}).\mathtt{ak}(\mathtt{M})$$

$$\mathtt{ak}(\{\mathtt{M}\}_{\mathtt{K}^{-1}}).\mathtt{ak}(\mathtt{K}) \xrightarrow{\mathbf{decrypt\_prk}(\mathtt{A},\mathtt{K},\mathtt{M})} \mathtt{ak}(\{\mathtt{M}\}_{\mathtt{K}^{-1}}).\mathtt{ak}(\mathtt{K}).\mathtt{ak}(\mathtt{M})$$

$$\mathtt{ak}(\mathtt{M}_1).\mathtt{ak}(\mathtt{M}_2) \xrightarrow{\mathbf{pairing}(\mathtt{A},\mathtt{M}_1,\mathtt{M}_2)} \mathtt{ak}(\mathtt{M}_1).\mathtt{ak}(\mathtt{M}_2).\mathtt{ak}(\langle \mathtt{M}_1, \mathtt{M}_2 \rangle)$$

$$\mathtt{ak}(\langle \mathtt{M}_1, \mathtt{M}_2 \rangle) \xrightarrow{\mathbf{decompose}(\mathtt{A},\mathtt{M}_1,\mathtt{M}_2)} \mathtt{ak}(\langle \mathtt{M}_1, \mathtt{M}_2 \rangle).\mathtt{ak}(\mathtt{M}_1).\mathtt{ak}(\mathtt{M}_2)$$

Modeling Secure Channels

$$(P_1 \| \cdots \| P_n \| I) \models (C \Rightarrow G)$$

Confidential Channel

A channel *ch* is confidential to principal *p* iff its **output** is exclusively accessible to a given receiver *p*:

$$confidential(ch, p) := \mathbf{G}\,\forall(\mathtt{rcvd}(A, B, M, ch) \Rightarrow A = p)$$

Authentic Channel

A channel *ch* is authentic for principal *p* iff its **input** is exclusively accessible to a given sender *p*:

$$authentic(ch, p) := \mathbf{G}\,\forall(\mathtt{sent}(RS, A, B, M, ch) \Rightarrow (A = p \wedge RS = p))$$

- Capital letters denote variables.
- $\forall(\alpha)$ abbreviates the universal closure of α.
- Quantifiers are over finite domains (bounded analysis).

Modeling Secure Channels

Weakly Confidential Channel

A channel *ch* is weakly confidential iff its **output** is exclusively accessible to a single, yet unknown, receiver:

$$weakly\_confidential(ch) :=$$
$$\mathbf{G}\,\forall((\mathtt{rcvd}(A, B, M, ch) \wedge \mathbf{F}\,\mathtt{rcvd}(A', B', M', ch)) \Rightarrow A = A')$$

Weakly Authentic Channel

A channel *ch* is weakly authentic iff its **input** is exclusively accessible to a single, yet unknown, sender:

$$weakly\_authentic(ch) :=$$
$$\mathbf{G}\,\forall((\mathtt{sent}(RS, A, B, M, ch) \wedge \mathbf{F}\,\mathtt{sent}(RS', A', B', M', ch)) \Rightarrow (A = A' \wedge RS = RS'))$$

Unilateral SSL Channel

A run of SSL/TLS in which principal y has a valid certificate but principal x does not, is modelled by a pair of channels $x2y$ and $y2x$:

$$\begin{aligned} &unilateral\_confidential\_authentic(x, y, x2y, y2x) := \\ &\quad (confidential(x2y, y) \wedge weakly\_authentic(x2y) \wedge \\ &\quad weakly\_confidential(y2x) \wedge authentic(y2x, y) \wedge \\ &\quad \mathbf{G}\,\forall(\mathbf{F}\,\mathtt{sent}(RS, x, y, M, x2y) \wedge \mathbf{F}\,\mathtt{rcvd}(R, y, M', y2x)) \\ &\qquad \Rightarrow RS = R)) \end{aligned}$$

With the additional requirement that the principal sending messages on $x2y$ is the same principal that receives messages from $y2x$.

Unilateral SSL Channel

A run of SSL/TLS in which principal y has a valid certificate but principal x does not, is modelled by a pair of channels $x2y$ and $y2x$:

$$
\begin{aligned}
&\mathit{unilateral\_confidential\_authentic}(x, y, x2y, y2x) := \\
&\qquad (\mathit{confidential}(x2y, y) \wedge \mathit{weakly\_authentic}(x2y) \wedge \\
&\qquad \mathit{weakly\_confidential}(y2x) \wedge \mathit{authentic}(y2x, y) \wedge \\
&\quad \mathbf{G}\,\forall (\mathbf{F}\,\mathtt{sent}(RS, x, y, M, x2y) \wedge \mathbf{F}\,\mathtt{rcvd}(R, y, M', y2x)) \\
&\qquad\qquad \Rightarrow RS = R))
\end{aligned}
$$

With the additional requirement that the principal sending messages on $x2y$ is the same principal that receives messages from $y2x$.

Specifying Security Properties

$$(P_1 \| \cdots \| P_n \| I) \models (C \Rightarrow G)$$

Authentication

b authenticates *a* on *m* in session *s* iff

$$authentication(b, a, m, s) :=$$
$$\mathbf{G}\,\forall(\texttt{state}_{r_b}(final\_step, b, [a, \ldots, m, \ldots], s) \Rightarrow$$
$$\exists\,\mathbf{O}\,\texttt{state}_{r_a}(initial\_step, a, [b, \ldots, m, \ldots], s))$$

Secrecy

Secrecy of *m* holds iff the intruder cannot possibly know it:

$$secret(m) := \mathbf{G}\,\neg\,\texttt{ik}(m)$$

Outline

SATMC: a Bounded Model Checker for Cryptographic Protocols

- SATMC is a bounded model checker for cryptographic protocols.
- Back-end of the AVISPA Tool and of the AVANTSSAR Platform.
- SATMC automatically generates a propositional formula whose satisfying assignments (if any) correspond to counterexamples (i.e. execution traces of $P_1 \| \cdots \| P_n \| I$ that satisfy C, and falsify G) of length bounded by some integer k.
- Successful combination of
  - SAT-reduction techniques developed for AI-planning
  - Bounded model-checking techniques developed for reactive systems.
- Finding attacks (of length k) on the protocol therefore boils down to solving propositional satisfiability problems.

Outline

- **[2007]** Flaw detected “patched” version of optimistic fair exhange protocol proposed by Asokan, Shoup, and Waidner (ASW).
- **[2008]** Man-in-the-middle attack discovered in SAML-based Single Sign-On for Google Apps
- **[2010]** Authentication flaw detected in SAML 2.0 Web Browser SSO Profile.
- **[2012]** Authentication flaw detected in use case of commercial 2-factors authentication protocol.

Outline

1. Cryptographic Protocols: a Gentle Introduction
2. Formal Modeling of Cryptographic Protocols
3. Model Checking of Cryptographic Protocols
4. Results
5. Next Steps

From Model Checking to Automatic Security Testing of Web-based Applications

- **Problem:** Checking the feasibility of attack traces returned by a model checker is a difficult and a labour-intensive activity.
- **Goal:** bind specifications of cryptographic protocols to actual implementations and use the model checker to automatically drive the security testing of implementations.

SPaCIoS

Conclusions

- Security protocols
  - crucial in securing distributed applications.
  - ubiquitous (can be found in all 7 layers of the ISO-OSI stack)
  - deceptively simple.
- Formal modeling forces designers to spell out assumptions and security goals.
- Model checkers effective in unveiling most subtle flaws.