Towards Automated Computationally Faithful Verification of Cryptoprotocols

Jan Jürjens

Dep. of Computer Science, TU München

Germany

juerjens@in.tum.de

http://www.jurjens.de/jan

Security Analysis a la Dolev-Yao

Specify protocol participants as processes following Dolev, Yao 1982: In addition to expected participants, model attacker who:

- may participate in some protocol runs,
- knows some data in advance,
- may intercept messages on the public network,
- injects messages that it can produce into the public network

Symbolic Analysis: Limitations

Keys are symbols, crypto-algorithms are abstract operations.

- Can only decrypt with right keys.
- Can only compose with available messages.
- Cannot perform statistical attacks.

Crypto assumed perfect, which isn't.

Computationally faithful analysis

Abadi, Rogaway 2000; Abadi, Jürjens 2001: Symbolic equivalence-based analysis faithful wrt. classical complexity-theoretical model (symmetric encryption, passive adversaries).

Problem: Symbolic model from AJ01 does not directly support automated verification.

Here: Ongoing work to extend above work to automated verification using first-order logic atp's (Dolev-Yao style).

Context: „Verisoft" Project

Goal: Practical application of formal methods.

Planned for 8 years from 7/2003; 12 industrial + academic partners.

Full formal verification from application software down to operating system and processor.

Intended result: Verified C-implementation.

One application example: Biometric authentication protocol (T-Systems).

Goal: Mechanical proof of complexity-theoretical security.

Security analysis in first-order logic

Idea: Given set *P* of control flow diagrams (of C-programs), approximate set of possible data values known to adversary from above.

Predicate *knows(E)* meaning that the adversary may get to know *E* during the execution of the protocol.

Say that a data value *s* is secret in *P* if one can not derive *knows(s)*.

Crypto Expressions

Term algebra generated by *Var* ∪ *Keys* ∪ *Data* and

- *_ :: _* (concatenation)
- *(_)*$^{-1}$ (inverse key)
- *{ _ }* *_* (encryption)
- *Sign_( )* (signing)
- *Dec_( )* (decryption)
- *Ext_( )* (extracting from signature)

with appropriate equations.

FOL rules for Crypto Expressions

$$\forall E_1, E_2. \Big(\mathsf{knows}(E_1 :: E_2) \Rightarrow \mathsf{knows}(E_1) \wedge \mathsf{knows}(E_2)\Big)$$
$$\wedge \Big(\mathsf{knows}(\{E_1\}_{E_2}) \wedge \mathsf{knows}(E_2^{-1}) \Rightarrow \mathsf{knows}(E_1)\Big)$$
$$\wedge \Big(\mathsf{knows}(\{E_1\}_{E_2}) \wedge \mathsf{knows}(E_2) \wedge \mathsf{sym}(E_2) \Rightarrow \mathsf{knows}(E_1)\Big)$$
$$\wedge \Big(\mathsf{knows}(\mathcal{S}ign_{E_2^{-1}}(E_1)) \wedge \mathsf{knows}(E_2) \Rightarrow \mathsf{knows}(E_1)\Big)$$
$$\wedge \Big(\mathsf{knows}(E_1) \wedge \mathsf{knows}(E_2) \Rightarrow \mathsf{knows}(E_1 :: E_2) \wedge \mathsf{knows}(\{E_1\}_{E_2}) \wedge \mathsf{knows}(\mathcal{S}ign_{E_2}(E_1))\Big)$$

Model for Security Protocols

State machine (Mealy automaton) with control states, local variables and transitions between states labeled *(in(var_in),cond(vars),out(msg_out))* where *msg_in* is a local variable to which the incoming message is assigned, *msgs* can be variables to which messages have been previously assigned, and *msg_out* is an output expression (each possibly empty).

Generate from protocol specs/code.

Security protocols into 1st order logic

Define *knows(E)* for any *E* initially known to the adversary (protocol-specific).

Control flow diagram: Each transition of form

(in(msg_in),cond(msgs),out(msg_out))

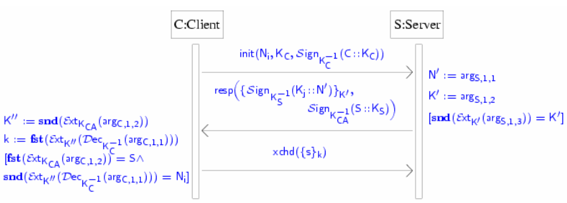
is translated (in a nested way) to:

∀msg_in. [knows(msg_in) ∧ cond(msgs)
⇒ knows(msg_out)]

(where for simplicity we use same names for logical and local variables).

Adversary knowledge approximated from above. Can put in more info, then more exact (+ less efficient).

Example: Proposed Variant of TLS (SSL)



knows(N_i) … ∧
∧ ∀exp… . (knows($arg_{S,1,3}$) ∧ knows($arg_{S,1,2}$) ∧
snd($Ext_{exp_{S,1,2}}$($arg_{S,1,3}$)) = $arg_{S,1,2}$
⇒ knows("arguments of resp method") ∧ …

Analysis

```
E-SETHEO csp03 single processor running on host ...
...
tlsvariant-freshkey-check.tptp
...
time limit information: 300 total (entering statistics module).
problem analysis ...
first-order problem
...
schedule selection: problem is horn with equality (class he).
schedule:605 3 300 597
...
entering next strategy 605 with resource 3 seconds.
...
analyzing results ...
proof found
time limit information: 298 total / 297 strategy (leaving wrapper).
...
```

Computationally faithful ?

Works fine for Dolev-Yao style analysis but: doesn't detect partial violation of secrecy.

Add another clause to each implication: Whenever condition in automaton is reached, all its subterms *relevant* to its validity are added to adversary knowledge.

Again approximation on the „safe“ side which works fine for practical examples.

Comparison to symbolic AJ01

Equivalence-based approach: „extrinsic“. Compute observable traces (somehow) and compare. Close to intuitions (but maybe not immediately clear how to efficiently verify eg with atp's).

Present approach: „intrinsic“. Stay as close to protocol model as possible when trying to detect information flow to enable efficient verification.

The computational view

An encryption scheme consists of algorithms:

$$\begin{aligned} \mathcal{K} &: \text{Parameter} \times \text{Coins} \rightarrow \text{Key} \\ \mathcal{E} &: \text{Key} \times \text{String} \times \text{Coins} \rightarrow \text{Cipher} \\ \mathcal{D} &: \text{Key} \times \text{String} \rightarrow \text{Plain} \end{aligned}$$

where $\text{Parameter} = 1^*$ (numbers in unary), Key, Plain, Cipher $\subseteq$ String.

For all $\eta \in \text{Parameter}$, $k \in \mathcal{K}(\eta)$, and $r \in \text{Coins}$,

- if $m \in \text{Plain}$ then $\mathcal{D}_k(\mathcal{E}_k(m, r)) = m$,
- if $m \notin \text{Plain}$ then $\mathcal{D}_k(\mathcal{E}_k(m, r)) = \perp$ (error message)

Indistinguishable Ensembles

A function $\epsilon : \mathbb{N} \rightarrow \mathbb{R}$ is negligible if for all $c > 0$ there exists N such that $\epsilon(\eta) \leq \eta^{-c}$ for all $\eta \geq N$.

An ensemble (or *probability ensemble*) is a collection of distributions on strings, $D = \{D_\eta\}$, one for each η.

We say that D and D' are indistinguishable and write $D \approx D'$, if

$$\begin{aligned} \epsilon(\eta) \triangleq\ & \Pr[x \stackrel{R}{\leftarrow} D_\eta :\ A(\eta, x) = 1] - \\ & \Pr[x \stackrel{R}{\leftarrow} D'_\eta :\ A(\eta, x) = 1] \end{aligned}$$

is negligible for all polynomial-time adversaries A.

Secure Encryption (variant)

Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme,
let $\eta \in \text{Parameter}$ be a security parameter.

Define

$$\begin{aligned} \text{Adv}_{\Pi,\eta}(A) \triangleq\ & \Pr\left[k, k' \stackrel{R}{\leftarrow} \mathcal{K}(\eta) :\ A^{\mathcal{E}_k(\cdot),\, \mathcal{E}_{k'}(\cdot)}(\eta) = 1\right] - \\ & \Pr\left[k \stackrel{R}{\leftarrow} \mathcal{K}(\eta) :\ A^{\mathcal{E}_k(\mathbf{0}),\, \mathcal{E}_k(\mathbf{0})}(\eta) = 1\right] \end{aligned}$$

Encryption scheme Π is secure if $\text{Adv}_{\Pi,\eta}(A)$ is negligible for all polynomial-time adversaries A.
(Goldwasser & Micali, Bellare et al.; Abadi & Rogaway)

Repetition concealing, message-length concealing, which-key concealing

Wrong key ?

In formal models, decrypting a message with the "wrong" key is a noticeable error. Computational counterpart:

Encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is confusion-free if for all $m \in \text{String}$ the probability

$$\Pr[k, k' \stackrel{R}{\leftarrow} \mathcal{K}(\eta), x \stackrel{R}{\leftarrow} \mathcal{E}_k(m) : \mathcal{D}_{k'}(x) \neq \perp]$$

is negligible.

Related: committing encryption (M. Fischlin)

Computational interpretation

To any set P of control flow graphs assign distribution $[[P]]_{\Pi,\eta}$ on input-/output histories (given an encryption scheme Π and a security parameter η):

Given an initial probability event τ, map each key symbol K to a bitstring $\tau(K)$, using $K(\eta)$. Mark all occurrences of encryptions $\{E\}_K$ with a different coin symbol r: $\{E\}^r_K$. Map each coin symbol r to a bit string $\tau(r)$. Then for expressions:

- $[[b]]^\tau_{\Pi,\eta} = b$
- $[[K]]^\tau_{\Pi,\eta} = \tau(K)$
- $[[M::N]]^\tau_{\Pi,\eta} = ([[M]]^\tau_{\Pi,\eta}, [[N]]^\tau_{\Pi,\eta})$

Computational interpretation II

Define $[[P]]^\tau_{\Pi,\eta}([])=[]$.

If $[[P]]^\tau_{\Pi,\eta}(ins)=outs \wedge p \rightarrow_{(in,gd,out)} p' \wedge gd(in)$

then $[[P[p' \leftarrow p]]]^\tau_{\Pi,\eta}(ins.in)=outs.out$.

(Assume messages to include address and guards to be mutually exclusive for each p.)

Define: data value s in P remains computationally secret if any two substitutions of s by other values are mutually indistinguishable.

Computational soundness

Let P be a set of state machines that does not generate encryption cycles and Π a secure and confusion-free encryption scheme.

If a data value s in P is secret then s is computationally secret.

(Still for symmetric encryption against passive adversaries; extension in progress.)

Conclusion

Work towards automated verification of security-critical software using first-order logic theorem provers which aims to be

- efficient, powerful
- intuitive, simple
- computationally faithful
- practically applicable

Limitations:

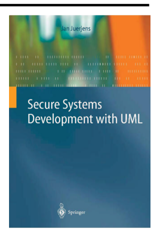
- give up (theoretical) completeness
- complexity theory is also „just“ a theoretical model

(Advertisement block)

Use verification in industrial projects with HypoVereinsbank, T-Systems, BMW, … Hide logic behind industrial notation UML:

Book: Jan Jürjens, Secure Systems Development with UML, Springer-Verlag, 2004

Summer School “Foundation of Security Analysis and Design”, Bertinoro (6-11/9)



More information (slides, tool etc.): http://www.jurjens.de/jan