Mechanized analysis of cryptographic protocols using causality based abstraction

Tool
Motivation
Outline

- Motivation
  - Analysis by hand

- Causality-based Abstraction
  - Graph
  - Traces

- Security Analysis

- Concrete Implementation
  - Problems

- Summary
Is this protocol secure?
Is it secure if run one hundred times in parallel?
The Attack

\[
\begin{align*}
\{B, n_b, m\}_{k^+_A} & \quad \quad \{n_b, n_a\}_{k^+_B} \\
\{n_b, n_a\}_{k^+_B} & \quad \quad \{n_a\}_{k^+_E}
\end{align*}
\]
The Fix

\[ \{ B, n_b, m \}_{k_A^+} \]

\[ \{ A, n_b, n_a \}_{k_B^+} \]

\[ \{ n_a \}_{k_A^+} \]

Proved secure by formal methods
Kerberos v5

Is this real-life example secure?
Problems with the Analysis

➢ Analysis by hand highly error prone
➢ Problem in general not decidable

Two approaches:

• Model Checking
• Static Analysis Techniques
Model Checking

- dynamic
- errors are really errors
- no way of proving security
- typically exponential running time

used e.g. for hardware verification
Static Analysis Techniques

- static...
- errors not necessarily errors
- may show a possible attack
- actual proofs of security
- typically poly runtime

used e.g. to verify java byte code
Outline

- Motivation
  - Analysis by hand
  - Causality-based Abstraction
    - Graph
    - Traces
- Security Analysis
- Concrete Implementation
  - Problems
- Summary
Causality-based Abstraction

- Dolev-Yao Intruder Model
- represent protocol in \( \rho \)-spi process calculus
- transform possibly infinitely many protocol sessions into a finite graph

\[
\text{out}\{\{n, m\}_{kA^+}\}.\text{in}(n).\text{end}(A, \ldots) \mid \text{in}\{\{x, y\}_{kA^-}\}.\text{begin}(B, A, \ldots)
\]
newkey(A).
  (new (m).new(n).out({B, n, m}_{kA^+}.in(n).0 | in({B, x, y}_{kA^-}).out(n).0)
Structural Reduction

➢ straightforward

➢ "x is necessarily preceded by y or z"

```
out (n1).end(A, B, m).0
```

```
out (n2).end(A, B, m).0
```

```
end(A, B, m).0
```

```
0
```
Interprocess Reduction

➢ rather complicated

➢ "x is necessarily preceded by y and z"

\[
in \ (x, \ y). \ \text{begin}(B, \ A, \ x).0 \\
\text{Com}(m_x, \ n_y) \\
\text{begin}(A, \ B, \ m).0 \\
\text{out} \ (m, \ n).0
\]
Capabilities of the Attacker $\mathcal{E}$

- pretty much everything (Dolev Yao)

```
• out (n).0
• Com (n_x, n_y)
• Com (\mathcal{E}_x, n_y)
• Com (n_x, \mathcal{E}_y)
• Com (\mathcal{E}_x, \mathcal{E}_y)
```

0
About the Graph

- A protocol has a unique graph.
- Causal net is sound abstraction.
  - Characterizes every possible reduction.
- Worst-case size: $N^{x^2}$.
Outline

- Motivation
  - Analysis by hand
- Causality-based Abstraction
  - Graph
  - Traces
  - Security Analysis
- Concrete Implementation
  - Problems
- Summary
from Graph to Analysis - Traces

new(n).(out(n).0 || in(x).0)

in(x).0

out(n).0

com(E_x)

com(n_x)

new(n).(out(n).0 || in(x).0)

0
new(n).(out(n).0 || in(x).0)

traces(0):
out(n)

com(E_x)
com(n_x)

0
from Graph to Analysis - Traces

traces(0):

\[
\begin{cases}
\text{new}(n) :: \text{out}(n)
\end{cases}
\]
from Graph to Analysis - Traces

new(n).(out(n).0 || in(x).0)

traces(0):
outcom(n_x, n_x) :: in(n_x)
in(n_x)
from Graph to Analysis - Traces

traces(0):

\[
\begin{align*}
\text{new}(n) &:: \text{in}(n_x) \\
\text{outcom}(n_x, n_x) &:: \text{in}(n_x)
\end{align*}
\]

\[
\text{new}(n) . (\text{out}(n)_0 \parallel \text{in}(x)_0)
\]
from Graph to Analysis -
Traces

new(n).(out(n).0 || in(x).0)  

traces(0):
in(E_x)

com(E_x)  com(n_x)  
in(x).0  out(n).0  
n  
0
from Graph to Analysis - Traces

new(n).(out(n).0 || in(x).0)

traces(0):
\{ 
  new(n) :: in(E_x) 
\}

com(n_x)  
com(E_x)  
in(x).0 
out(n).0

0
from Graph to Analysis - Traces

new(n). (out(n).0 || in(x).0)

in(x).0  out(n).0

com(E_x)  com(n_x)

0

traces(0):

\{
  \ \text{new}(n) :: \ \text{outcom}(n_x, n_x) :: \ \text{in}(n_x)
\}

\{
  \ \text{new}(n) :: \ \text{in}(n_x)
\}

\{
  \ \text{new}(n) :: \ \text{out}(n)
\}

\{
  \ \text{new}(n) :: \ \text{in}(E_x)
\}
Analysis

Secrecy (of n)

new(n).(out(n).0 || in(n).0)

in(n).0  out(n).0

com(n)

0
Analysis

Secrecy (of n)

\[
\text{new}(n). (\text{out}(n).0 \parallel \text{in}(n).0)
\]

\[
\text{in}(n).0 \quad \text{out}(n).0
\]

\[
\text{com}(n)
\]

\[
0
\]
Analysis

Secrecy

```
new(n).(out({n}_{k+}).0 || in({n}_{k-}).0)
```

0

```
in({n}_{k-}).0
out({n}_{k+}).0
com({n})
```
Analysis

Authenticity

\begin{align*}
\text{begin}^1_n(A,B,m) & \quad \text{end}^1_n(B,A,m) \\
\{B,n,m\}_{kA+} & \\
\end{align*}
Analysis

Authenticity

newkey(A) :: new(m) :: new(n) ::
out({(B, n, m)}_{kA+}) :: in(n) :: end^n(A, B, m)

newkey(A) :: in({(B, n_x, m_z)}_{kA-}) ::
begin^{nx}(B, A, m_z) :: outcom(n, n) :: in(n) :: end^n(A, B, m)

newkey(A) :: new(m) :: new(n) ::
outcom({(B, n_x, m_z)}_{kA+}, {(B, n_x, m_z)}_{kA+}) ::
in({(B, n_x, m_z)}_{kA-}) :: begin^n(B, A, m_z) ::
outcom(n, n) :: in(n) :: end^n(A, B, m)
Outline

- Motivation
  - Analysis by hand
- Causality-based Abstraction
  - Graph
  - Traces
- Security Analysis
  - Concrete Implementation
    - Problems
- Summary
What did we do?

moved from stack overflow to endless loop. great result.
parser now in my opinion ugly but gives more expressive error messages than just parser error
found out about precedence and went back to old version, so no more nice error messages so far

at some point we get now information about the place where it occurs in the parser or lexer or .. somewhere else
did something

nicer output lexer

r4864: parser takes extremely long
r4871: parser restored
Implementational Tricks and Features

- dynamic programming using hashtables
- static environment
- trees
- different outputs
- light-weight gui
Problems

Needham-Schroeder-Lowe

\[ \{ B, n_b, m \}_A^+ \]

\[ \{ A, n_b, n_a \}_B^+ \]

\[ \{ n_a \}_A^+ \]
Problems

Needham-Schroeder-Lowe

auth check fails

new(na)...

new(na)...

in({na})...

in({na})...

in({na})...

out({na})...

com({na})...

end$_{na}$(A,B,E)...

end$_{na}$(A,B,m)...

18 : out(((A^11, B_xm^12), na^13)}_{k+B^14}).in({na})_{k-A}.end^2_{na}(A,B,E_xm).0

13 : out(((A^5, nb_xn^6), na^7)}_{k+B^8}).in({na})_{k-A}.end^2_{na}(A,B,m_xm).0

22 : Com(((A^5, nb^6), na_xa^7)}_{k+B^8})

5 : end$_{A1}$nb(B,A,m).begin$_2$na_xa(B,A,m).out({na_xa})_{k-A}.0

4 : begin$_2$na_xa(B,A,m).out({na_xa})_{k-A}.0

{n^9}_{k-A^10}
auth check succeeds

new(na). ...

19: out({(A, B \_xn), na})_{k+ B}. in({na} \_k - A). end^2_{na}(A, B, B \_xm).0

new(na1). ...

13: out({(A^5, nb \_xn^6), na1^7})_{k+ B^8}. in({na1} \_k - A). end^2_{na1}(A, B, m \_xm).0

in({na1}). ...

22: Com({(A^5, nb^6), na1 \_xa^7})_{k+ B^8}

in({na}). ...

5: end^1_{na} (B, A, m). begin^2_{na1 \_xa}(B, A, m). out({na1} \_k - A).

out({na1}). ...

4: begin^2_{na1 \_xa}(B, A, m). out({na1} \_k - A).

end_{na1}(A,B,m)
Alpha-renaming as solution?

\[
\text{in}(x).\text{new}(n).\text{out}(n).0
\]

\[
\text{com}(E)
\]

\[
\text{new}(n).\text{out}(n).0
\]

\[
\text{out}(n).0
\]

\[
0
\]
Alpha-renaming as solution?
Alpha-renaming as solution?
Alpha-renaming as solution?
Finally the solution!
Finally the solution!
Finally the solution!
Summary

The protocol is cycle invariant.
The protocol does not guarantee secrecy for the following names:
  * n leaked by nodes: 3
The protocol guarantees authenticity.
Outlook and further work

- testing the tool
- refactoring and optimization
- integration in larger project (avispa)
- support for unified description language
References


Thank you!