Verification of security protocols via constraint solving

Stéphanie Delaune

January 5, 2009

Cryptographic protocols

Cryptographic protocols

- small programs designed to secure communication
- use cryptographic primitives (*e.g.* encryption, hash function, . . .)

Security properties

Secrecy: May an intruder learn some secret message between two honest participants ?

Authentication: Is the agent Alice really talking to Bob ?

Security properties

Secrecy: May an intruder learn some secret message between two honest participants ?

Authentication: Is the agent Alice really talking to Bob ?

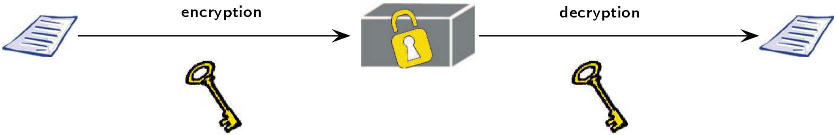
Privacy: Alice participate to an election. May a participant learn something about the vote of Alice ?

Receipt-Freeness: Alice participate to an election. Does Alice gain any information (a receipt) which can be used to prove to a coercer that she voted in a certain way ?

Fairness: ...

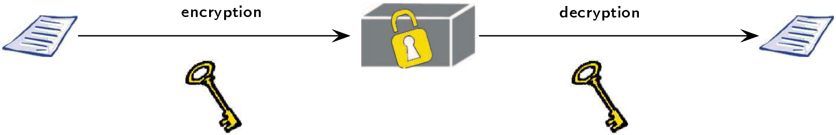
Cryptographic primitives

Symmetric encryption

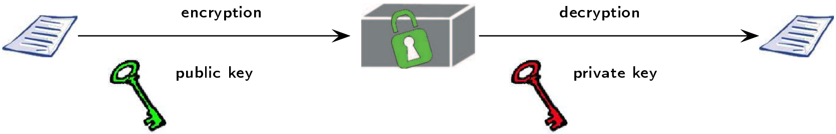


Cryptographic primitives

Symmetric encryption



Asymmetric encryption



Verification of cryptographic protocols

Verification of cryptographic protocols

How cryptographic protocols can be attacked?

Breaking encryption

How cryptographic protocols can be attacked?

Breaking encryption

Logical attack

Logical attack – What is it?

transfer 100 euros on merchant's bank account ⟶

Logical attack – What is it?

transfer 100 euros on
merchant's bank account
⟶

transfer 100 euros on

merchant's bank account
⟶

Logical attack – What is it?

transfer 100 euros on
merchant's bank account
$\longrightarrow$

transfer 100 euros on
merchant's bank account
$\longrightarrow$

transfer 100 euros on
merchant's bank account
$\longrightarrow$

$\vdots$

transfer 100 euros on
merchant's bank account
$\longrightarrow$

Credit Card Payment Protocol

Example: credit card payment

- The client Cl puts his credit card C in the terminal T.
- The merchant enters the amount M of the sale.
- The terminal authenticates the credit card.
- The client enters his PIN.
  If $M \geq$ €100, then in 20% of cases,
  - The terminal contacts the bank B.
  - The banks gives its authorisation.

More details

the Bank B , the Client Cl, the Credit Card C and the Terminal T

More details

the Bank B , the Client Cl, the Credit Card C and the Terminal T

Bank

- a private signature key – $\mathrm{priv}(B)$
- a public key to verify a signature – $\mathrm{pub}(B)$
- a secret key shared with the credit card – K_{CB}

More details

the Bank B , the Client Cl, the Credit Card C and the Terminal T

Bank

- a private signature key – $\mathsf{priv}(B)$
- a public key to verify a signature – $\mathsf{pub}(B)$
- a secret key shared with the credit card – K_{CB}

Credit Card

- some *Data*: name of the cardholder, expiry date ...
- a signature of the *Data* – $\mathsf{sign}(Data, \mathsf{priv}(B))$
- a secret key shared with the bank – K_{CB}

More details

the Bank B , the Client Cl, the Credit Card C and the Terminal T

Bank

- a private signature key – $\text{priv}(B)$
- a public key to verify a signature – $\text{pub}(B)$
- a secret key shared with the credit card – K_{CB}

Credit Card

- some *Data*: name of the cardholder, expiry date ...
- a signature of the *Data* – $\text{sign}(Data, \text{priv}(B))$
- a secret key shared with the bank – K_{CB}

Terminal

- the public key of the bank – $\text{pub}(B)$

Payment protocol

the terminal T reads the credit card C:

$$1. \quad C \quad \rightarrow \quad T: \quad Data, \mathsf{sign}(Data, \mathsf{priv}(B))$$

Payment protocol

the terminal T reads the credit card C:

$$1. \quad C \;\rightarrow\; T : \quad Data, \mathsf{sign}(Data, \mathsf{priv}(B))$$

the terminal T asks the code:

$$\begin{array}{llcll} 2. & T & \rightarrow & Cl : & code? \\ 3. & Cl & \rightarrow & C : & 1234 \\ 4. & C & \rightarrow & T : & ok \end{array}$$

Payment protocol

the terminal T reads the credit card C:

$$1. \quad C \rightarrow T : \quad Data, \mathsf{sign}(Data, \mathsf{priv}(B))$$

the terminal T asks the code:

$$\begin{array}{lllll} 2. & T & \rightarrow & CI: & code? \\ 3. & CI & \rightarrow & C: & 1234 \\ 4. & C & \rightarrow & T: & ok \end{array}$$

the terminal T requests authorisation the bank B:

$$\begin{array}{lllll} 5. & T & \rightarrow & B: & auth? \\ 6. & B & \rightarrow & T: & 4528965874123 \\ 7. & T & \rightarrow & C: & 4528965874123 \\ 8. & C & \rightarrow & T: & \mathsf{enc}(4528965874123, K_{CB}) \\ 9. & T & \rightarrow & B: & \mathsf{enc}(4528965874123, K_{CB}) \\ 10. & B & \rightarrow & T: & ok \end{array}$$

Attacks on the credit card

Security was initially ensured by:

- the cards were difficult to reproduce,
- the protocol and the keys were secret.

Attacks on the credit card

Security was initially ensured by:

- the cards were difficult to reproduce,
- the protocol and the keys were secret.

But there are some flaws:

- cryptographic flaw: keys of 320 bits are too small,
- logical flaw: no link between the secret code and the authentication of the card;
- fake cards can be easily build.

⟶ "YesCard" built by Serge Humpich (1997).

YesCard: How does it work?

Logical Flaw:

$$
\begin{array}{llll}
1.C & \rightarrow T & : \text{Data}, \text{sign}(\text{Data}, \text{priv}(B)) \\
2.T & \rightarrow CI & : code? \\
3.CI & \rightarrow C & : 1234 \\
4.C & \rightarrow T & : ok
\end{array}
$$

YesCard: How does it work?

Logical Flaw:

$$
\begin{array}{llll}
1. C & \rightarrow & T & : \text{Data}, \text{sign}(\text{Data}, \text{priv}(B)) \\
2. T & \rightarrow & Cl & : code? \\
3. Cl & \rightarrow & C' & : 0000 \\
4. C' & \rightarrow & T & : ok
\end{array}
$$

YesCard: How does it work?

Logical Flaw:

$$
\begin{array}{llll}
1. C & \rightarrow & T & : \text{Data}, \text{sign}(\text{Data}, \text{priv}(B)) \\
2. T & \rightarrow & Cl & : code? \\
3. Cl & \rightarrow & C' & : 0000 \\
4. C' & \rightarrow & T & : ok
\end{array}
$$

$\longrightarrow$ Note that there is someone to debit.

YesCard: How does it work?

Logical Flaw:

$$\begin{array}{llll} 1.C & \to T & : \text{Data}, \text{sign}(\text{Data}, \text{priv}(B)) \\ 2.T & \to CI & : code? \\ 3.CI & \to C' & : 0000 \\ 4.C' & \to T & : ok \end{array}$$

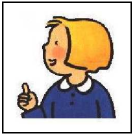
$\longrightarrow$ Note that there is someone to debit.

YesCard (by Serge Humpich)

$$\begin{array}{llll} 1.C' & \to T & : \text{XXX}, \text{sign}(\text{XXX}, \text{priv}(B)) \\ 2.T & \to CI & : code? \\ 3.CI & \to C' & : 0000 \\ 4.C' & \to T & : ok \end{array}$$

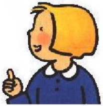
Needham-Schroeder's protocol

Needham-Schroeder's Protocol (1978)



- $$\begin{array}{rcrr} A & \rightarrow & B: & \{A, N_a\}_{\mathsf{pub}(B)} \\ B & \rightarrow & A: & \{N_a, N_b\}_{\mathsf{pub}(A)} \\ A & \rightarrow & B: & \{N_b\}_{\mathsf{pub}(B)} \end{array}$$

Needham-Schroeder's Protocol (1978)

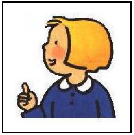


$$A \rightarrow B : \{A, N_a\}_{\mathsf{pub}(B)}$$

- $$B \rightarrow A : \{N_a, N_b\}_{\mathsf{pub}(A)}$$

$$A \rightarrow B : \{N_b\}_{\mathsf{pub}(B)}$$

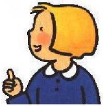
Needham-Schroeder's Protocol (1978)



$$A \rightarrow B : \{A, N_a\}_{\mathsf{pub}(B)}$$
$$B \rightarrow A : \{N_a, N_b\}_{\mathsf{pub}(A)}$$

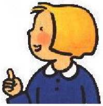
- $A \rightarrow B : \{N_b\}_{\mathsf{pub}(B)}$

Needham-Schroeder's Protocol (1978)



$$
\begin{array}{lcll}
A & \rightarrow & B: & \{A, N_a\}_{\mathsf{pub}(B)} \\
B & \rightarrow & A: & \{N_a, N_b\}_{\mathsf{pub}(A)} \\
A & \rightarrow & B: & \{\color{red}{N_b}\}_{\mathsf{pub}(B)}
\end{array}
$$

Needham-Schroeder's Protocol (1978)

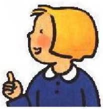


$$\begin{array}{rcll} A & \to & B: & \{A, N_a\}_{\mathsf{pub}(B)} \\ B & \to & A: & \{N_a, N_b\}_{\mathsf{pub}(A)} \\ A & \to & B: & \{N_b\}_{\mathsf{pub}(B)} \end{array}$$

Questions

- Is N_b secret between A and B ?
- When B receives $\{N_b\}_{\mathsf{pub}(B)}$, does this message really comes from A ?

Needham-Schroeder's Protocol (1978)



$$\begin{array}{lcll} A & \rightarrow & B: & \{A, N_a\}_{\mathsf{pub}(B)} \\ B & \rightarrow & A: & \{N_a, N_b\}_{\mathsf{pub}(A)} \\ A & \rightarrow & B: & \{N_b\}_{\mathsf{pub}(B)} \end{array}$$

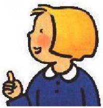
Questions

- Is N_b secret between A and B ?
- When B receives $\{N_b\}_{\mathsf{pub}(B)}$, does this message really comes from A ?

Attack

An attack was found 17 years after its publication! [Lowe 96]

Example: Man in the Middle Attack



Agent A

Intrus I

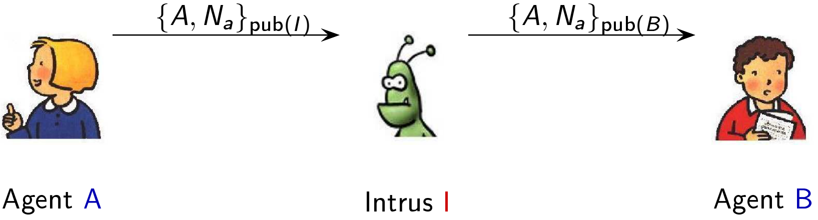
Agent B

Attack

- involving 2 sessions in parallel,
- an honest agent has to initiate a session with I.

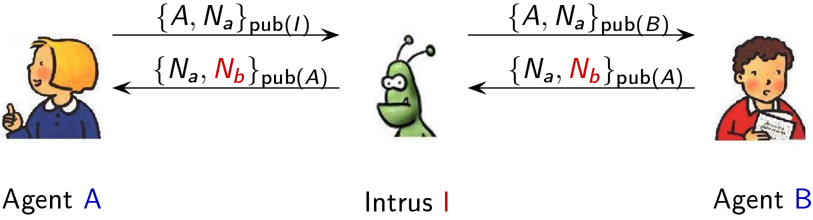
$$
\begin{aligned}
A \rightarrow B &: \{A, N_a\}_{\mathsf{pub}(B)} \\
B \rightarrow A &: \{N_a, N_b\}_{\mathsf{pub}(A)} \\
A \rightarrow B &: \{N_b\}_{\mathsf{pub}(B)}
\end{aligned}
$$

Example: Man in the Middle Attack



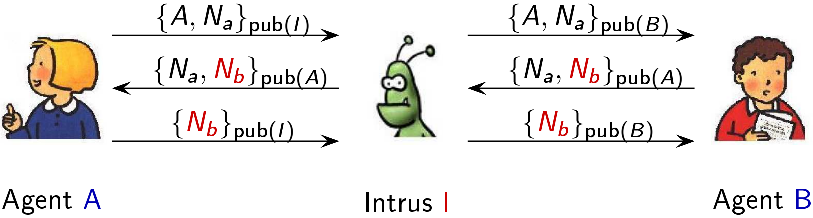
$$
\begin{array}{lll}
A \rightarrow B & : & \{A, N_a\}_{\mathrm{pub}(B)} \\
B \rightarrow A & : & \{N_a, N_b\}_{\mathrm{pub}(A)} \\
A \rightarrow B & : & \{N_b\}_{\mathrm{pub}(B)}
\end{array}
$$

Example: Man in the Middle Attack



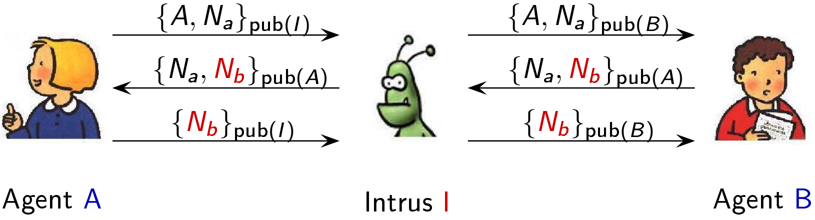
$$
\begin{array}{lcl}
A \rightarrow B & : & \{A, N_a\}_{\mathrm{pub}(B)} \\
B \rightarrow A & : & \{N_a, N_b\}_{\mathrm{pub}(A)} \\
A \rightarrow B & : & \{N_b\}_{\mathrm{pub}(B)}
\end{array}
$$

Example: Man in the Middle Attack



$$
\begin{aligned}
\mathrm{A} \rightarrow \mathrm{B} &: \{A, N_a\}_{\mathrm{pub}(B)} \\
\mathrm{B} \rightarrow \mathrm{A} &: \{N_a, N_b\}_{\mathrm{pub}(A)} \\
\mathrm{A} \rightarrow \mathrm{B} &: \{N_b\}_{\mathrm{pub}(B)}
\end{aligned}
$$

Example: Man in the Middle Attack



Attack

- the intruder knows N_b,
- When B finishes his session (apparently with A), A has never talked with B.

$$
\begin{array}{lll}
A \rightarrow B & : & \{A, N_a\}_{pub(B)} \\
B \rightarrow A & : & \{N_a, N_b\}_{pub(A)} \\
A \rightarrow B & : & \{N_b\}_{pub(B)}
\end{array}
$$

Logical attacks - How to detect them?

Symbolic approach

- messages are represented by terms rather than bit-strings
  $\hookrightarrow \{m\}_k$ encryption of the message m with key k,
  $\hookrightarrow \langle m_1, m_2 \rangle$ pairing of messages m_1 and m_2, ...
- attacker controls the network and can perform specific actions

Logical attacks - How to detect them?

Symbolic approach

- messages are represented by terms rather than bit-strings
  $\hookrightarrow \{m\}_k$ encryption of the message m with key k,
  $\hookrightarrow \langle m_1, m_2 \rangle$ pairing of messages m_1 and m_2, ...
- attacker controls the network and can perform specific actions

Relevance of the approach

- numerous attacks have already been obtained,
- allows us to perform automatic verification, *e.g.* AVISPA, Proverif, ...
- soundness results already exist, *e.g.* [Micciancio & Warinschi'04]

Outline of the talk

1. Introduction

2. How to deal with trace properties (e.g. secrecy, authentication, ...)?

3. How to deal with equivalence based properties (e.g. privacy, ...)?

4. Conclusion

Outline of the talk

1. Introduction
2. How to deal with trace properties (e.g. secrecy, authentication, . . .)?
3. How to deal with equivalence based properties (e.g. privacy, . . .)?
4. Conclusion

Deduction capabilities of the attacker

Composition rules

$$\frac{T \vdash u \quad T \vdash v}{T \vdash \langle u, v \rangle} \qquad \frac{T \vdash u \quad T \vdash v}{T \vdash \mathsf{f}(u, v)} \ \textit{with } \mathsf{f} \in \{\mathsf{enc}, \mathsf{enca}, \mathsf{sign}\}$$

Decomposition rules

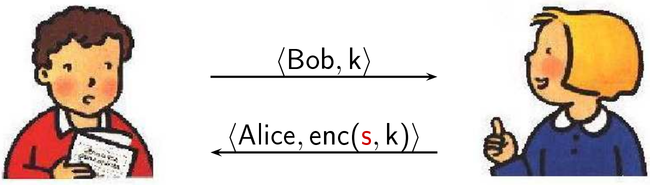
$$\frac{}{T \vdash u}\ u \in T \qquad \frac{T \vdash \langle u, v \rangle}{T \vdash u} \qquad \frac{T \vdash \langle u, v \rangle}{T \vdash v} \qquad \frac{T \vdash \mathsf{enc}(u, v) \quad T \vdash v}{T \vdash u}$$

$$\frac{T \vdash \mathsf{enca}(u, \mathsf{pub}(v)) \quad T \vdash \mathsf{priv}(v)}{T \vdash u} \qquad \frac{T \vdash \mathsf{sign}(u, \mathsf{priv}(v))}{T \vdash u}\ \textit{(optional)}$$

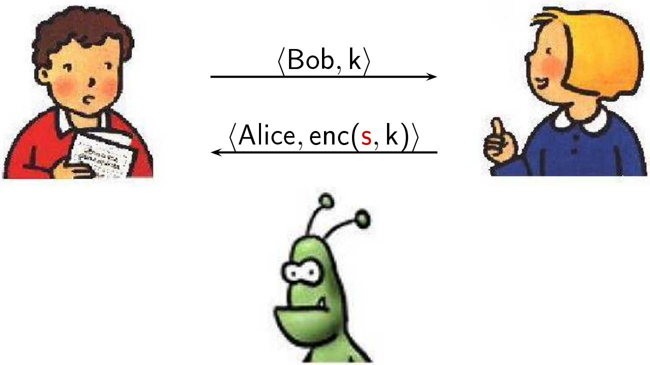
Deducibility relation

A term u is deducible from a set of terms T, denoted by $T \vdash u$, if there exists a prooftree witnessing this fact.

A simple protocol



A simple protocol



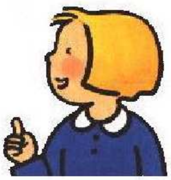
Question?

Can the attacker learn the secret s?

A simple protocol

$$\xrightarrow{\langle \mathsf{Bob}, \mathsf{k} \rangle}$$

$$\xleftarrow{\langle \mathsf{Alice}, \mathsf{enc}(\mathsf{s}, \mathsf{k}) \rangle}$$



Answer: Of course, Yes!

$$\frac{\dfrac{\langle \mathsf{Alice}, \mathsf{enc}(\mathsf{s}, \mathsf{k}) \rangle}{\mathsf{enc}(\mathsf{s}, \mathsf{k})} \qquad \dfrac{\langle \mathsf{Bob}, \mathsf{k} \rangle}{\mathsf{k}}}{\mathsf{s}}$$

Deducibility problem - Some existing results

$\longrightarrow$ depends on the deduction capabilities of the intruder

Dolev-Yao intruder

The deducibility problem is decidable in polynomial time.

Deducibility problem - Some existing results

$\longrightarrow$ depends on the deduction capabilities of the intruder

Dolev-Yao intruder

The deducibility problem is decidable in polynomial time.

Prefix Intruder (*e.g.* Cipher Block Chaining)

$$\frac{T \vdash \{\langle m_1, m_2 \rangle\}_{\mathsf{pub}(A)}}{T \vdash \{m_1\}_{\mathsf{pub}(A)}}$$

Deducibility problem - Some existing results

$\longrightarrow$ depends on the deduction capabilities of the intruder

Dolev-Yao intruder

The deducibility problem is decidable in polynomial time.

Prefix Intruder (*e.g.* Cipher Block Chaining)

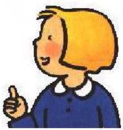
$$\frac{T \vdash \{\langle m_1, m_2 \rangle\}_{\mathsf{pub}(A)}}{T \vdash \{m_1\}_{\mathsf{pub}(A)}}$$

Taking into account algebraic properties of the cryptographic primitives (*e.g.* RSA encrytpion)

$$\mathsf{E} := \left\{ \begin{array}{rcl} \mathsf{dec}(\mathsf{enc}(x, \mathsf{pub}(y)), \mathsf{priv}(y)) & = & x \\ \mathsf{enc}(\mathsf{dec}(x, \mathsf{priv}(y)), \mathsf{pub}(y)) & = & x \end{array} \right.$$

$$\frac{T \vdash m \quad T \vdash k}{T \vdash \mathsf{f}(m, k)} \ \mathsf{f} \in \{\mathsf{dec}, \mathsf{enc}\} \qquad \frac{T \vdash m_1}{T \vdash m_2} \ m_1 =_{\mathsf{E}} m_2$$

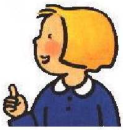
Needham Schroeder protocol:



$$\begin{array}{rcrr} A & \rightarrow & B: & \{N_a, A\}_{\mathsf{pub}(B)} \\ B & \rightarrow & A: & \{N_a, N_b\}_{\mathsf{pub}(A)} \\ A & \rightarrow & B: & \{N_b\}_{\mathsf{pub}(B)} \end{array}$$

Protocol – Example: Needham Schroeder protocol (1978)

Needham Schroeder protocol:



$$\begin{array}{rcrr} A & \rightarrow & B: & \{N_a, A\}_{\mathsf{pub}(B)} \\ B & \rightarrow & A: & \{N_a, N_b\}_{\mathsf{pub}(A)} \\ A & \rightarrow & B: & \{N_b\}_{\mathsf{pub}(B)} \end{array}$$

A protocol is a finite set of roles:

Exemple:

role $\Pi(1)$ corresponding to the 1st participant played by a talking to b:

$$\begin{array}{l} \mathsf{init} \xrightarrow{N} \mathsf{enca}(\langle N, a\rangle, \mathsf{pub}(b)) \\ \mathsf{enca}(\langle N, x\rangle, \mathsf{pub}(a)) \rightarrow \mathsf{enca}(x, \mathsf{pub}(b)). \end{array}$$

Trace properties in presence of an active attacker

Insecurity problem (bounded number of sessions)

Let $\mathcal{I}$ be an inference system modelling the attacker.

INPUT: a finite set $R_1, \ldots, R_m$ of instances of roles,
a finite set T_0 of terms (initial intruder knowledge),
a term s (the secret)

Trace properties in presence of an active attacker

Insecurity problem (bounded number of sessions)

Let $\mathcal{I}$ be an inference system modelling the attacker.

INPUT: a finite set $R_1, \ldots, R_m$ of instances of roles,
a finite set T_0 of terms (initial intruder knowledge),
a term s (the secret)

OUTPUT: Does there exist an interleaving of $R_1, \ldots, R_m$ runnable from T_0 w.r.t. $\mathcal{I}$ at the end of which

- the intruder knowledge is T, and
- s is deducible from T in $\mathcal{I}$?

Trace properties in presence of an active attacker

Insecurity problem (bounded number of sessions)

Let $\mathcal{I}$ be an inference system modelling the attacker.

INPUT: a finite set $R_1, \ldots, R_m$ of instances of roles,
a finite set T_0 of terms (initial intruder knowledge),
a term s (the secret)

OUTPUT: Does there exist an interleaving of $R_1, \ldots, R_m$ runnable from T_0 w.r.t. $\mathcal{I}$ at the end of which

- the intruder knowledge is T, and
- s is deducible from T in $\mathcal{I}$?

Security properties (trace properties): *e.g.* secrecy, some kinds of authentication properties, ...

Secrecy via constraint system

Constraint systems are used to specify secrecy preservation under a particular, finite scenario.

Scenario

$$
\begin{array}{rcl}
\mathrm{rcv}(u_1) & \xrightarrow{N_1} & \mathrm{snd}(v_1) \\
\mathrm{rcv}(u_2) & \xrightarrow{N_2} & \mathrm{snd}(v_2) \\
 & \ldots & \\
\mathrm{rcv}(u_n) & \xrightarrow{N_n} & \mathrm{snd}(v_n)
\end{array}
$$

Secrecy via constraint system

Constraint systems are used to specify secrecy preservation under a particular, finite scenario.

Scenario

$$\begin{array}{ccc} \mathsf{rcv}(u_1) & \stackrel{N_1}{\rightarrow} & \mathsf{snd}(v_1) \\ \mathsf{rcv}(u_2) & \stackrel{N_2}{\rightarrow} & \mathsf{snd}(v_2) \\ & \ldots & \\ \mathsf{rcv}(u_n) & \stackrel{N_n}{\rightarrow} & \mathsf{snd}(v_n) \end{array}$$

Constraint System

$$\mathcal{C} = \left\{ \begin{array}{l} T_0 \Vdash u_1 \\ T_0, v_1 \Vdash u_2 \\ \quad \ldots \\ T_0, v_1, .., v_n \Vdash s \end{array} \right.$$

Secrecy via constraint system

Constraint systems are used to specify secrecy preservation under a particular, finite scenario.

Scenario

$$\begin{array}{c} \mathrm{rcv}(u_1) \xrightarrow{N_1} \mathrm{snd}(v_1) \\ \mathrm{rcv}(u_2) \xrightarrow{N_2} \mathrm{snd}(v_2) \\ \ldots \\ \mathrm{rcv}(u_n) \xrightarrow{N_n} \mathrm{snd}(v_n) \end{array}$$

Constraint System

$$\mathcal{C} = \begin{cases} T_0 \Vdash u_1 \\ T_0, v_1 \Vdash u_2 \\ \quad \ldots \\ T_0, v_1, .., v_n \Vdash s \end{cases}$$

Solution of a constraint system

A substitution σ such that

for every $T \Vdash u \in \mathcal{C}$, $u\sigma$ *is deducible from* $T\sigma$.

Secrecy via constraint system

Constraint systems are used to specify secrecy preservation under a particular, finite scenario.

Scenario

$$\begin{array}{l} \mathrm{rcv}(u_1) \xrightarrow{N_1} \mathrm{snd}(v_1) \\ \mathrm{rcv}(u_2) \xrightarrow{N_2} \mathrm{snd}(v_2) \\ \quad\quad\quad \ldots \\ \mathrm{rcv}(u_n) \xrightarrow{N_n} \mathrm{snd}(v_n) \end{array}$$

Constraint System

$$\mathcal{C} = \begin{cases} T_0 \Vdash u_1 \\ T_0, v_1 \Vdash u_2 \\ \quad \ldots \\ T_0, v_1, .., v_n \Vdash s \end{cases}$$

Well-formed constraint system

- monotonicity: intruder never forgets information
- origination: a variable first appear in a right hand side.

Running example: Needham-Schroeder's protocol

$R_A(a, I)$ and $R_B(b)$ (running in parallel)

$$\begin{array}{rcl} \text{init} & \rightarrow & \{a, n_a\}_{\mathrm{pub}(I)} \\ \{n_a, x_{n_b}\}_{\mathrm{pub}(a)} & \rightarrow & \{x_{n_b}\}_{\mathrm{pub}(I)} \\ & & \\ \{y_a, y_{n_a}\}_{\mathrm{pub}(b)} & \rightarrow & \{y_{n_a}, n_b\}_{\mathrm{pub}(y_a)} \end{array}$$

Running example: Needham-Schroeder's protocol

$R_A(a, I)$ and $R_B(b)$ (running in parallel)

1	init	$\rightarrow$	$\{a, n_a\}_{\mathsf{pub}(I)}$
3	$\{n_a, x_{n_b}\}_{\mathsf{pub}(a)}$	$\rightarrow$	$\{x_{n_b}\}_{\mathsf{pub}(I)}$
2	$\{y_a, y_{n_a}\}_{\mathsf{pub}(b)}$	$\rightarrow$	$\{y_{n_a}, n_b\}_{\mathsf{pub}(y_a)}$

Running example: Needham-Schroeder's protocol

$R_A(a, I)$ and $R_B(b)$ (running in parallel)

1	init	$\rightarrow$	$\{a, n_a\}_{\mathsf{pub}(I)}$
3	$\{n_a, x_{n_b}\}_{\mathsf{pub}(a)}$	$\rightarrow$	$\{x_{n_b}\}_{\mathsf{pub}(I)}$
2	$\{y_a, y_{n_a}\}_{\mathsf{pub}(b)}$	$\rightarrow$	$\{y_{n_a}, n_b\}_{\mathsf{pub}(y_a)}$

Constraints System

Running example: Needham-Schroeder's protocol

$R_A(a, I)$ and $R_B(b)$ (running in parallel)

1	init	$\rightarrow$	$\{a, n_a\}_{\mathrm{pub}(I)}$
3	$\{n_a, x_{n_b}\}_{\mathrm{pub}(a)}$	$\rightarrow$	$\{x_{n_b}\}_{\mathrm{pub}(I)}$
2	$\{y_a, y_{n_a}\}_{\mathrm{pub}(b)}$	$\rightarrow$	$\{y_{n_a}, n_b\}_{\mathrm{pub}(y_a)}$

Constraints System

$$T_0,\ \{a, n_a\}_{\mathrm{pub}(I)}$$

Running example: Needham-Schroeder's protocol

$R_A(a, I)$ and $R_B(b)$ (running in parallel)

1	init	$\rightarrow$	$\{a, n_a\}_{\mathrm{pub}(I)}$
3	$\{n_a, x_{n_b}\}_{\mathrm{pub}(a)}$	$\rightarrow$	$\{x_{n_b}\}_{\mathrm{pub}(I)}$
2	$\{y_a, y_{n_a}\}_{\mathrm{pub}(b)}$	$\rightarrow$	$\{y_{n_a}, n_b\}_{\mathrm{pub}(y_a)}$

Constraints System

$$T_0,\ \{a, n_a\}_{\mathrm{pub}(I)} \ \Vdash\ \{y_a, y_{n_a}\}_{\mathrm{pub}(b)}$$

Running example: Needham-Schroeder's protocol

$R_A(a, I)$ and $R_B(b)$ (running in parallel)

$$
\begin{array}{lrcl}
\mathbf{1} & \text{init} & \rightarrow & \{a, n_a\}_{\text{pub}(I)} \\
\mathbf{3} & \{n_a, x_{n_b}\}_{\text{pub}(a)} & \rightarrow & \{x_{n_b}\}_{\text{pub}(I)} \\
 & & & \\
\mathbf{2} & \{y_a, y_{n_a}\}_{\text{pub}(b)} & \rightarrow & \{y_{n_a}, n_b\}_{\text{pub}(y_a)}
\end{array}
$$

Constraints System

$$
\begin{array}{l}
T_0, \ \{a, n_a\}_{\text{pub}(I)} \ \Vdash \ \{y_a, y_{n_a}\}_{\text{pub}(b)} \\
T_0, \ \{a, n_a\}_{\text{pub}(I)}, \ \{y_{n_a}, n_b\}_{\text{pub}(y_a)}
\end{array}
$$

Running example: Needham-Schroeder's protocol

$R_A(a, I)$ and $R_B(b)$ (running in parallel)

$$
\begin{array}{llcl}
\mathbf{1} & \text{init} & \rightarrow & \{a, n_a\}_{\mathsf{pub}(I)} \\
\mathbf{3} & \{n_a, x_{n_b}\}_{\mathsf{pub}(a)} & \rightarrow & \{x_{n_b}\}_{\mathsf{pub}(I)} \\
 & & & \\
\mathbf{2} & \{y_a, y_{n_a}\}_{\mathsf{pub}(b)} & \rightarrow & \{y_{n_a}, n_b\}_{\mathsf{pub}(y_a)}
\end{array}
$$

Constraints System

$$
\begin{array}{l}
T_0,\ \{a, n_a\}_{\mathsf{pub}(I)} \Vdash \{y_a, y_{n_a}\}_{\mathsf{pub}(b)} \\
T_0,\ \{a, n_a\}_{\mathsf{pub}(I)},\ \{y_{n_a}, n_b\}_{\mathsf{pub}(y_a)} \Vdash \{n_a, x_{n_b}\}_{\mathsf{pub}(a)}
\end{array}
$$

Running example: Needham-Schroeder's protocol

$R_A(a, I)$ and $R_B(b)$ (running in parallel)

$$\begin{array}{lrcl} \mathbf{1} & \text{init} & \rightarrow & \{a, n_a\}_{\text{pub}(I)} \\ \mathbf{3} & \{n_a, x_{n_b}\}_{\text{pub}(a)} & \rightarrow & \{x_{n_b}\}_{\text{pub}(I)} \\ & & & \\ \mathbf{2} & \{y_a, y_{n_a}\}_{\text{pub}(b)} & \rightarrow & \{y_{n_a}, n_b\}_{\text{pub}(y_a)} \end{array}$$

Constraints System

$$\begin{array}{l} T_0,\ \{a, n_a\}_{\text{pub}(I)} \Vdash \{y_a, y_{n_a}\}_{\text{pub}(b)} \\ T_0,\ \{a, n_a\}_{\text{pub}(I)},\ \{y_{n_a}, n_b\}_{\text{pub}(y_a)} \Vdash \{n_a, x_{n_b}\}_{\text{pub}(a)} \\ T_0,\ \{a, n_a\}_{\text{pub}(I)},\ \{y_{n_a}, n_b\}_{\text{pub}(y_a)},\ \{x_{n_b}\}_{\text{pub}(I)} \end{array}$$

Running example: Needham-Schroeder's protocol

$R_A(a, I)$ and $R_B(b)$ (running in parallel)

$$\begin{array}{lrcl}
\mathbf{1} & \text{init} & \rightarrow & \{a, n_a\}_{\mathrm{pub}(I)} \\
\mathbf{3} & \{n_a, x_{n_b}\}_{\mathrm{pub}(a)} & \rightarrow & \{x_{n_b}\}_{\mathrm{pub}(I)} \\
\\
\mathbf{2} & \{y_a, y_{n_a}\}_{\mathrm{pub}(b)} & \rightarrow & \{y_{n_a}, n_b\}_{\mathrm{pub}(y_a)}
\end{array}$$

Constraints System

$$\begin{array}{l}
T_0,\ \{a, n_a\}_{\mathrm{pub}(I)} \ \Vdash\ \{y_a, y_{n_a}\}_{\mathrm{pub}(b)} \\
T_0,\ \{a, n_a\}_{\mathrm{pub}(I)},\ \{y_{n_a}, n_b\}_{\mathrm{pub}(y_a)} \ \Vdash\ \{n_a, x_{n_b}\}_{\mathrm{pub}(a)} \\
T_0,\ \{a, n_a\}_{\mathrm{pub}(I)},\ \{y_{n_a}, n_b\}_{\mathrm{pub}(y_a)},\ \{x_{n_b}\}_{\mathrm{pub}(I)} \ \Vdash\ n_b
\end{array}$$

Running example: Needham-Schroeder's protocol

$R_A(a, I)$ and $R_B(b)$ (running in parallel)

$$
\begin{array}{llcl}
\mathbf{1} & \text{init} & \rightarrow & \{a, n_a\}_{\mathsf{pub}(I)} \\
\mathbf{3} & \{n_a, x_{n_b}\}_{\mathsf{pub}(a)} & \rightarrow & \{x_{n_b}\}_{\mathsf{pub}(I)} \\
 & & & \\
\mathbf{2} & \{y_a, y_{n_a}\}_{\mathsf{pub}(b)} & \rightarrow & \{y_{n_a}, n_b\}_{\mathsf{pub}(y_a)}
\end{array}
$$

Constraints System

$$
\begin{array}{l}
T_0,\ \{a, n_a\}_{\mathsf{pub}(I)} \Vdash \{y_a, y_{n_a}\}_{\mathsf{pub}(b)} \\
T_0,\ \{a, n_a\}_{\mathsf{pub}(I)},\ \{y_{n_a}, n_b\}_{\mathsf{pub}(y_a)} \Vdash \{n_a, x_{n_b}\}_{\mathsf{pub}(a)} \\
T_0,\ \{a, n_a\}_{\mathsf{pub}(I)},\ \{y_{n_a}, n_b\}_{\mathsf{pub}(y_a)},\ \{x_{n_b}\}_{\mathsf{pub}(I)} \Vdash n_b
\end{array}
$$

Solution $\quad \sigma = \{y_a \mapsto \quad , \quad y_{n_a} \mapsto \quad , \quad x_{n_b} \mapsto \quad \}$

Running example: Needham-Schroeder's protocol

$R_A(a, I)$ and $R_B(b)$ (running in parallel)

1	init	$\rightarrow$	$\{a, n_a\}_{\mathrm{pub}(I)}$
3	$\{n_a, x_{n_b}\}_{\mathrm{pub}(a)}$	$\rightarrow$	$\{x_{n_b}\}_{\mathrm{pub}(I)}$
2	$\{y_a, y_{n_a}\}_{\mathrm{pub}(b)}$	$\rightarrow$	$\{y_{n_a}, n_b\}_{\mathrm{pub}(y_a)}$

Constraints System

$$T_0,\ \{a, n_a\}_{\mathrm{pub}(I)} \Vdash \{y_a, y_{n_a}\}_{\mathrm{pub}(b)}$$
$$T_0,\ \{a, n_a\}_{\mathrm{pub}(I)},\ \{y_{n_a}, n_b\}_{\mathrm{pub}(y_a)} \Vdash \{n_a, x_{n_b}\}_{\mathrm{pub}(a)}$$
$$T_0,\ \{a, n_a\}_{\mathrm{pub}(I)},\ \{y_{n_a}, n_b\}_{\mathrm{pub}(y_a)},\ \{x_{n_b}\}_{\mathrm{pub}(I)} \Vdash n_b$$

Solution $\quad \sigma = \{y_a \mapsto a,\ \ y_{n_a} \mapsto n_a,\ \ x_{n_b} \mapsto \quad \}$

Running example: Needham-Schroeder's protocol

$R_A(a, I)$ and $R_B(b)$ (running in parallel)

$$\begin{array}{lrcl}
\mathbf{1} & \text{init} & \rightarrow & \{a, n_a\}_{\text{pub}(I)} \\
\mathbf{3} & \{n_a, x_{n_b}\}_{\text{pub}(a)} & \rightarrow & \{x_{n_b}\}_{\text{pub}(I)} \\
 & & & \\
\mathbf{2} & \{y_a, y_{n_a}\}_{\text{pub}(b)} & \rightarrow & \{y_{n_a}, n_b\}_{\text{pub}(y_a)}
\end{array}$$

Constraints System

$$\begin{array}{l}
T_0,\ \{a, n_a\}_{\text{pub}(I)} \Vdash \{y_a, y_{n_a}\}_{\text{pub}(b)} \\
T_0,\ \{a, n_a\}_{\text{pub}(I)},\ \{y_{n_a}, n_b\}_{\text{pub}(y_a)} \Vdash \{n_a, x_{n_b}\}_{\text{pub}(a)} \\
T_0,\ \{a, n_a\}_{\text{pub}(I)},\ \{y_{n_a}, n_b\}_{\text{pub}(y_a)},\ \{x_{n_b}\}_{\text{pub}(I)} \Vdash n_b
\end{array}$$

Solution $\sigma = \{y_a \mapsto a,\ \ y_{n_a} \mapsto n_a,\ \ x_{n_b} \mapsto n_b\}$

Some existing results

Many theoretical results for different intruder models

- to take into account algebraic properties of cryptographic primitives (exclusive or, cipher block chaining, ...)
- to take into account the fact that some data are poorly-chosen (*e.g.* passwords)

Some existing results

Many theoretical results for different intruder models

- to take into account algebraic properties of cryptographic primitives (exclusive or, cipher block chaining, ...)
- to take into account the fact that some data are poorly-chosen (*e.g.* passwords)

Few generic results

- procedure to solve constraint systems for a class of intruder
  $\hookrightarrow$ *e.g.* any intruder who can be described by a subterm convergent rewiting system
- combination result for disjoint intruder models.

Some existing results

Many theoretical results for different intruder models

- to take into account algebraic properties of cryptographic primitives (exclusive or, cipher block chaining, ...)
- to take into account the fact that some data are poorly-chosen (*e.g.* passwords)

Few generic results

- procedure to solve constraint systems for a class of intruder
  $\hookrightarrow$ *e.g.* any intruder who can be described by a subterm convergent rewiting system
- combination result for disjoint intruder models.

Some tools

- AVISPA tool (Atse, OFMC)

Outline of the talk

Motivation: Electronic voting

Advantages:

- Convenient,
- Efficient facilities for tallying votes.

Drawbacks:

- Risk of large-scale and undetectable fraud,
- Such protocols are extremely error-prone.

> *"A 15-year-old in a garage could manufacture smart cards and sell them on the Internet that would allow for multiple votes"* Avi Rubin

Possible issue: formal methods

abstract analysis of the protocol against formally-stated properties

Expected properties

Privacy: the fact that a particular voter voted in a particular way is not revealed to anyone

Receipt-freeness: a voter cannot prove that she voted in a certain way (this is important to protect voters from coercion)

Coercion-resistance: same as receipt-freeness, but the coercer interacts with the voter during the protocol, e.g. by preparing messages

How to model such security properties?

Formalisation of Privacy
$\hookrightarrow$ consider 2 honest voters and swap their votes

Privacy

A voting protocol respects privacy if

$$S[V_A\{^a/_v\} \mid V_B\{^b/_v\}] \approx S[V_A\{^b/_v\} \mid V_B\{^a/_v\}].$$

How to model such security properties?

Formalisation of Privacy
$\hookrightarrow$ consider 2 honest voters and swap their votes

Privacy

A voting protocol respects privacy if

$$S[V_A\{^a/_v\} \mid V_B\{^b/_v\}] \approx S[V_A\{^b/_v\} \mid V_B\{^a/_v\}].$$

Formalisation of Receipt-freeness and Coercion-resistance in term of equivalence.

Constraint solving

In terms of constraint system, the main ingredient to decide $\approx$:

$\mathcal{C}_1 \sim \mathcal{C}_2$: equivalence of (well-formed) constraint systems

What does it mean?

1. this does not mean that $\mathcal{C}_1$ and $\mathcal{C}_2$ have the same set of (first-order) solutions.
2. Given a solution σ, let $\Lambda_\sigma = \{\lambda_\sigma^1, \ldots, \lambda_\sigma^k\}$ be the witnesses of the fact that σ is a solution of

$$\mathcal{C} := \begin{cases} T_1 \Vdash u_1 \\ \vdots \\ T_\ell \Vdash u_\ell \end{cases}$$

$$\mathcal{C}_1 \sim \mathcal{C}_2 \quad \text{iff} \quad \{\Lambda_\sigma \mid \sigma \in Sol(\mathcal{C}_1)\} = \{\Lambda_\sigma \mid \sigma \in Sol(\mathcal{C}_2)\}.$$

Existing results

A lot of results in the passive case

- to take into account algebraic properties (exclusive or, . . .)
- combination result for disjoint equational theories,
- YAPA tool that works for subterm convergent theories and more

Active case: very few results

- decision procedure for subterm convergent theories (not implemented)
- ProVerif tool

Ongoing work

Motivation: verification of privacy type proeprties in e-voting protocols

Passive case:
$\longrightarrow$ to deal with more complex cryptographic primitives, those that are frequently used in e-voting protocols

- blind signature (already done in the passive case)
- trapdoor bit commitment
- reencryption mechanism

Active case:
design a procedure to decide equivalence of constraint systems in presence of blind signature.

$\longrightarrow$ this will allow us to decide privacy in e-voting protocols, e.g. protocol due to Fujioka, Okamoto and Ohta.

Conclusion

Verification via constraint solving
$\longrightarrow$ a useful approach to verify security protocols

- can be adapted to other cryptographic primitives;
- useful for trace properties but also equivalence based properties;
- can be adapted to deal with regular constraints, e.g. $u \in L$;
- limits: only a bounded number of sessions

Conclusion

Verification via constraint solving
$\longrightarrow$ a useful approach to verify security protocols

- can be adapted to other cryptographic primitives;
- useful for trace properties but also equivalence based properties;
- can be adapted to deal with regular constraints, e.g. $u \in L$;
- limits: only a bounded number of sessions

Questions?