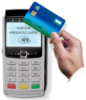
Automatic Verification of Privacy Protection for Unbounded Sessions

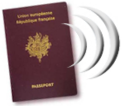
CSF 5’

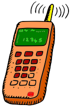
Lucca Hirschi

July 16, 2015

Introduction

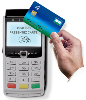


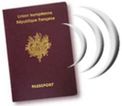


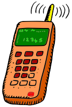


⇝ we need formal verification of crypto protocols covering privacy

Introduction





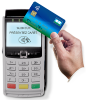


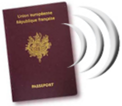
⇝ we need formal verification of crypto protocols covering privacy

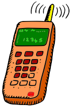
Goal:

- checking privacy (unlinkability and anonymity)
- in the symbolic model
- for unbounded sessions.

Introduction







$\rightsquigarrow$ we need formal verification of crypto protocols covering privacy

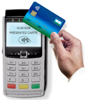
Goal:

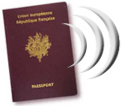
- checking privacy (unlinkability and anonymity)
- in the symbolic model
- for unbounded sessions.

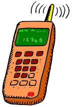
Strong unlinkability [Ryan *et al.* CSF'10]:

$$! \, \nu \vec{k} \; ! \, \nu \vec{n}(T \mid R) \approx ! \, \nu \vec{k} . \nu \vec{n}(T \mid R)$$

Introduction







⇝ we need formal verification of crypto protocols covering privacy

Goal:

- checking privacy (unlinkability and anonymity)
- in the symbolic model
- for unbounded sessions.

Strong unlinkability [Ryan *et al.* CSF'10]:

$$! \, \nu \vec{k} \; ! \, \nu \vec{n} (T \mid R) \approx \, ! \, \nu \vec{k} . \nu \vec{n} (T \mid R)$$

Existing approaches:

- manual: need to exhib huge bisimulations;
- automatic (ProVerif/Maude-NPA): abstractions yield false attacks.

Contribution

We identify:

- 2 conditions implying unlinkability and anonymity
- for a class of 2-agents protocols including some target case studies;

We make sure:

- our conditions can be checked automatically using Proverif;
- they correspond to good design practices.

$\rightsquigarrow$ sound approach to check automatically privacy properties working well in practice

A taste of $C_{\mathrm{rel}} \& C_{\mathrm{honest}}$

	Equivalence?	Active Attacker?
UK/ANO	☑	☑

A taste of C_{rel} & C_{honest}

	Equivalence?	Active Attacker?
UK/ANO	☑	☑

⇑ **implies** ⇑

C_{rel}	all outputs are indistinguishable from "nonces"
C_{honest}	test of A holds $\Rightarrow$ A had an honest interaction

A taste of C_{rel} & C_{honest}

	Equivalence?	Active Attacker?
UK/ANO	☑	☑

⇑ **implies** ⇑

	Equivalence?	Active Attacker?
C_{rel}	☑	☐
C_{honest}	☐	☑

A taste of C_{rel}&C_{honest}

	Equivalence?	Active Attacker?
UK/ANO	☑	☑

⇑ **implies** ⇑

	Equivalence?	Active Attacker?
C_{rel}	☑	☐
C_{honest}	☐	☑

⇑ can be **checked** ⇑

- C_{rel}: automatic check of diff-equivalence using Proverif
- C_{honest}: automatic check of correspondence prop. using Proverif

Applications

New proofs of UK & Ano for:

- BAC+PA+AA (ePassport);
- PACE+PA+AA (ePassport v2);
- (fixed) LAK (RFID auth.);
- Hash-Lock (RFID auth.).

Applications

New proofs of UK & Ano for:

- BAC+PA+AA (ePassport);
- PACE+PA+AA (ePassport v2);
- (fixed) LAK (RFID auth.);
- Hash-Lock (RFID auth.).

When conditions fail to hold: no direct attacks but still...

Flaws/attacks discovered:

- some versions of PACE ($\neg$ UK);
- LAK ($\neg$ UK).

... still looking for other case studies ...

Thank You!