Automated verification of privacy-type properties for security protocols

Ivan Gazeau

LORIA, INRIA

March 16th, 2018

Security protocols

Cryptographic protocols are intensively used on many contexts:

- https
- electronic passport
- contact less credit card
- electronic voting

Threats

Every years, news papers reports attacks on these protocols
We can distinguish between four kinds of attacks:

- Attack on cryptographic primitives

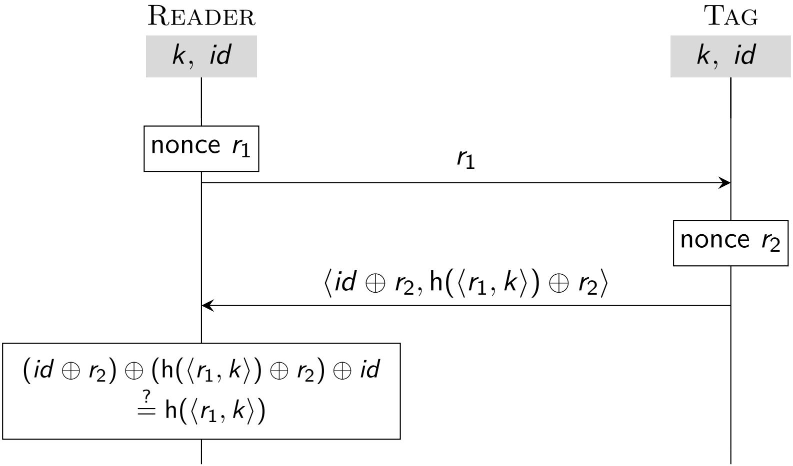
  SHA-1
- Logical attacks on the specification of protocols

  BAC on French passports, Crack on WPA2 Wifi
- Attacks on the implementation of the protocols

  Heart blood
- Side channel attacks

  Spectre

Protocol specification example



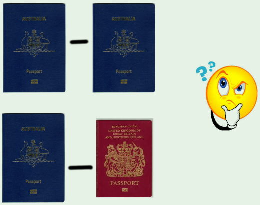
Security properties

We distinguish between two kinds of security properties:

- Trace properties (predicates on system behavior)
  - (weak) secrecy of a key
  - authentication (correspondence properties)
- Equivalence properties (involving two systems)
  - Unlinkability
  - Offline Guessing-attack
  - Non interference

We focus on **equivalence properties**.

Unlinkability



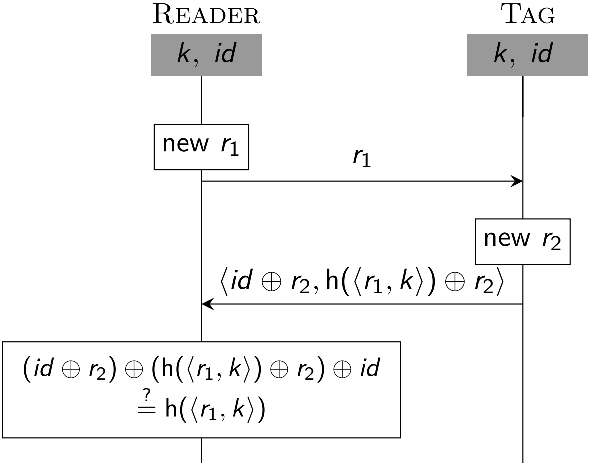
Equivalence property

Testing equivalence ($P \approx Q$)
for all processes A, we have that:

$$A \mid P \Downarrow c \text{ if, and only if, } A \mid Q \Downarrow c$$

$P \Downarrow c$ when P can send a message on the channel c.

Authentication protocol of a RFID tag (KCL)



Is unlinkability satisfied?

$$\mathrm{tag}(id, k) \mid \mathrm{tag}(id, k) \stackrel{?}{\approx} \mathrm{tag}(id, k) \mid \mathrm{tag}(id', k')$$

Modelling the protocol

The interactions between devices

Protocols written in a process calculus, the applied pi calculus

$$
\begin{array}{rcll}
P ::= & & 0 & \\
& \mid & \mathrm{in}(c, x).P & \text{input} \\
& \mid & \mathrm{out}(c, t).P & \text{output} \\
& \mid & \text{if } t_1 = t_2 \text{ then } P \text{ else } Q & \text{conditional} \\
& \mid & P \parallel Q & \text{parallel} \\
& \mid & !P & \text{replication} \\
& \mid & \text{new } n.P & \text{restriction}
\end{array}
$$

The network **is** the attacker

No communication between the parallel processes:

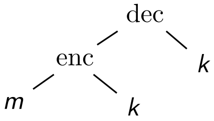
- messages are sent to the attacker
- received messages are the ones chosen by the attacker
- concurrency is determined by the attacker

Modelling the protocol

Cryptographic primitives

Data structure and functions used are represented by terms.

- messages = **terms**



- **perfect** cryptography (equational theories)

$$\mathrm{dec}(\mathrm{enc}(x, y), y) = x \quad \mathsf{fst}(\mathsf{pair}(x, y)) = x \quad \mathsf{snd}(\mathsf{pair}(x, y)) = y$$

Specificities:

- messages are **terms**
- equality in conditionals interpreted modulo an **equational theory**

Reasoning about attacker knowledge

Outputs

The execution of the process is formalized by an operational semantics.

$$\textsc{Send}\quad (\mathbf{out}(c,t).P,\varphi) \xrightarrow{\mathbf{out}(c)} (P, \varphi \cup \{\mathsf{w}_{|\varphi|+1} \mapsto t{\downarrow}\})$$

There is no attacker process, the semantics only remember the messages received by the attacker in a **frame**.

$$P := \mathbf{out}(c,t_1).\mathbf{out}(c,t_2).\mathbf{out}(c,t_3)$$

$$(P,\emptyset) \xrightarrow{\mathbf{out}(c)}{}^{3} (0, \mathsf{new}\ \bar{n}.\ \{^{t_1}/_{w_1}, ^{t_2}/_{w_2}, ^{t_3}/_{w_3}\})$$

Reasoning about attacker knowledge

Inputs

Recipe: A term which contains elements of the frame.

It represents the computations performed by the attacker.

The attacker uses these recipes to:

- provide inputs to the process
- test equalities

Semantics for input:

$$\textsc{Recv} \quad (\mathbf{in}(c,x).P,\varphi) \xrightarrow{\mathbf{in}(c,R)} (P\{x \mapsto t{\downarrow}\},\varphi)$$

if t can be obtained from φ with recipe R.

The label $\mathbf{in}(c,R)$ keeps track of which recipe has been chosen by the attacker. It is used to compare with traces of the other process.

Reasoning about attacker knowledge

Deducibility

A recipe provides a term to the process.
Formally, $\phi \vdash^R t$ if R is a public term and $R\phi =_E t$

$$\varphi = \mathsf{new}\ n_1, n_2, k_1, k_2.\ \{^{\mathrm{enc}(n_1,k_1)}/_{w_1}, ^{\mathrm{enc}(n_2,k_2)}/_{w_2}, ^{k_1}/_{w_3}\}$$

$$\varphi \vdash^{\mathrm{dec}(w_1,w_3)} n_1 \qquad \varphi \nvdash n_2 \qquad \varphi \vdash^{\mathbf{0}} \mathbf{0}$$

Reasoning about attacker knowledge

Static equivalence:

The received messages look like random nonce for the attacker except if some equalities hold.

$$\text{new } n, k.\ \{^{\mathrm{enc}(0,k)}/_{w_1}, ^{k}/_{w_2}\}$$

$(dec(w_1, w_2)) = 0$

If some equality hold in one scenario but not in another one, it can distinguish the two scenarios.

$$\text{new } n_1, n_2.\ \{^{n_1}/_{w_1}, ^{n_2}/_{w_2}\} \not\sim_s \text{new } n_1.\ \{^{n_1}/_{w_1}, ^{n_1}/_{w_2}\}$$

Check $(w_1 \stackrel{?}{=} w_2)$

Trace equivalence (for processes)

Static equivalence: two frames where the same equalities hold.

$\phi_1 \sim_s \phi_2$ if $\forall$ public terms R, R'.

$$R\phi_1 = R'\phi_1 \Leftrightarrow R\phi_2 = R'\phi_2$$

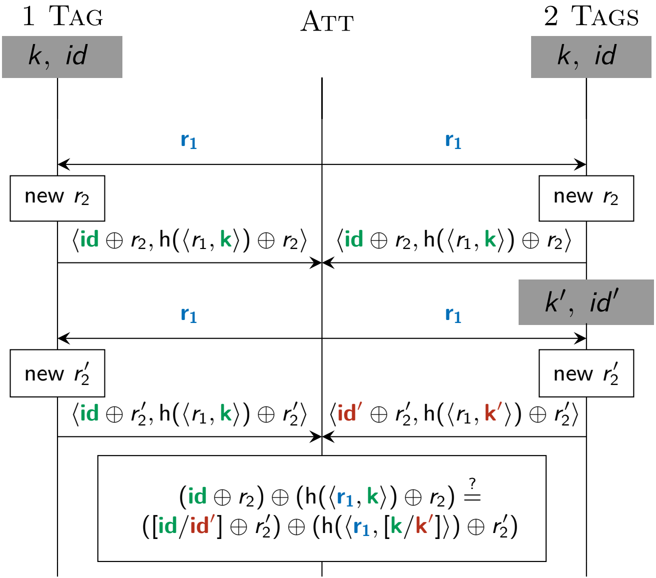
$$\text{new } k.\ \{^{\mathrm{enc}(\mathbf{0},k)}/_{x_1}, ^{\mathrm{enc}(\mathbf{0},k)}/_{x_2}\} \sim_s \text{new } k.\ \{^{\mathrm{enc}(\mathbf{1},k)}/_{x_1}, ^{\mathrm{enc}(\mathbf{1},k)}/_{x_2}\}$$

Formal definition of the equivalence property:

Definition (Trace equivalence: $P \approx Q$)

if $(P, \emptyset) \xrightarrow{\mathsf{tr}} (P', \varphi)$
then $\exists Q', \varphi'.\ (Q, \emptyset) \xrightarrow{\mathsf{tr}} (Q', \varphi') \wedge \varphi \sim_s \varphi'$ and reciprocally.

Linkability attack



Verification tools

- Good support to verify traces properties (which rely on one process)
- Equivalence properties (which rely on two processes): Maude-NPA, APTE, SPEC, ProVerif, Tamarin, Deepsec, Akiss

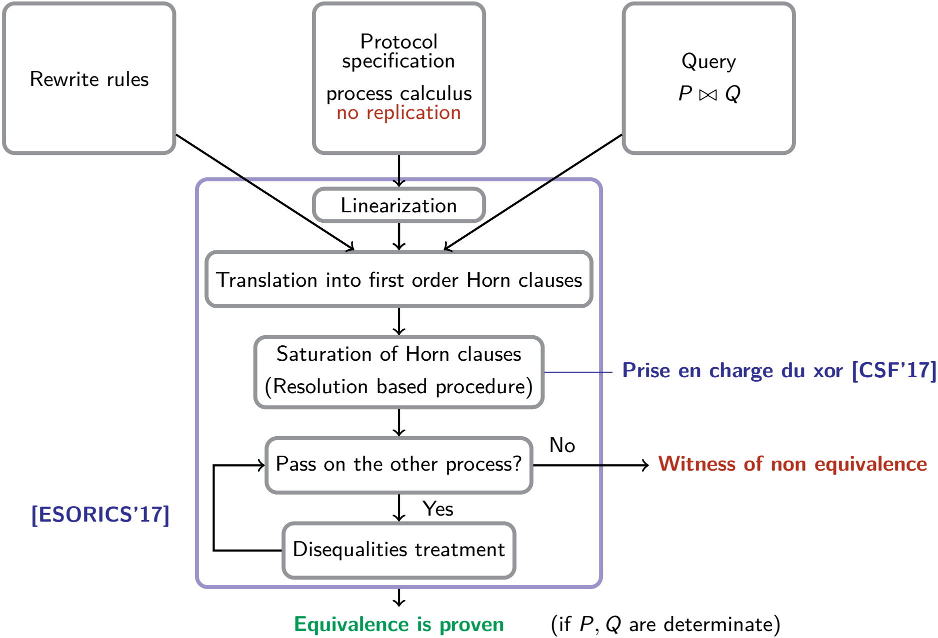
But

- Except for APTE and DeepSec, no support for else branches or support for simple cases (same control flow in both scenarios).
- No tool for equivalence provides Xor primitive support and else branches.

My personal contribution to Akiss

- Add xor support [CSF'17]
- Add else branches support [ESORICS'17]

AKISS: overview



Predicates

Three kinds of predicate:

r_w there is a trace $L_1, \ldots, L_n$ whose abstraction is w.

$\mathsf{k}_w(R, t)$ after a trace w, the attacker can build t with recipe R.

$\mathsf{i}_w(R, R')$ after a trace w, recipes R and R' provide the same term.

where w represents the trace where the predicate is valid.

Modelling protocols in Horn clauses

Direct translation from the protocol and the theory to Horn Clauses

The seed of a process

$$\mathcal{R} = \{dec(enc(x, y), y) \to x\}$$

$$T = \mathbf{in}(c, x).\text{if } (\mathrm{dec}(x, k) = a) \text{ then } \mathbf{out}(c, s)$$

$$\mathsf{k}_w(\mathrm{dec}(X, Y), z) \;\Leftarrow\; \mathsf{k}_w(X, \mathrm{enc}(z, y)), \mathsf{k}_w(Y, y)$$

$$\mathsf{k}_{\mathbf{receive}(c,\mathrm{enc}(a,k)),\mathbf{test},\mathbf{send}(c)}(w_1, s) \;\Leftarrow\; \mathsf{k}(X, \mathrm{enc}(a, k))$$

Saturation

Resolution procedure from the seed statements

$$\frac{H \Leftarrow B_1, B_2 \qquad B_1 \Leftarrow B_3, B_4}{H \Leftarrow B_2, B_3, B_4}$$

Saturation until we get a saturated set where premisses does not carry information

Theorem

For any identity that the attacker can get on P, there is a more general identity in the saturated set.

Interest of Xor ?

- Used in many low device protocols
  - RFID tags
  - Mobile telephony (AKA)
- Used as a ciphering method but weaker than encryption
  - Some attacks exploit the algebraic properties of xor

There was no tool to effectively analyze equivalence properties for such protocols

Xor: a complex operator

Lot of algebraic properties:

- Associativity: $x \oplus (y \oplus z) = (x \oplus y) \oplus z$
- Commutativity: $x \oplus y = y \oplus x$
- Nilpotent: $x \oplus x = 0$
- Unit: $x \oplus 0 = x$

Requires reasoning modulo AC

Unifiers of $h(x) \oplus y = h(x') \oplus y'$?

Several solutions (no most general unifier)

- $x \mapsto x' \qquad y \mapsto y'$
- $y \mapsto h(x') \qquad y' \mapsto h(x)$
- $y \mapsto h(x') \oplus z \qquad y' \mapsto h(x) \oplus z$

Dealing with Xor

Partial correctness: no issue

Termination: saturation never terminates

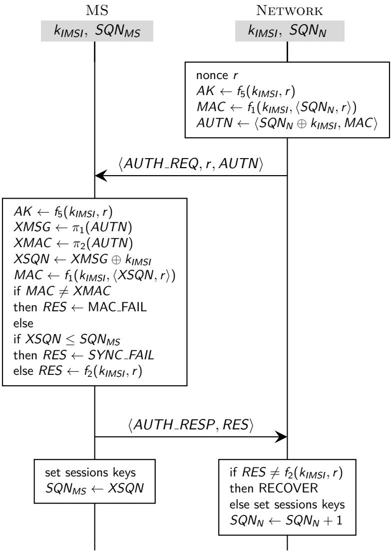
- Restrict the use of some saturation rules
- Show it is equivalent to enforce some parenthesizing

Protocols with Else branches: a large class of protocols

Else branches are used for:

- sending error messages
- avoid replay attacks
- restart processes when failure

The AKA protocol used in 3G telephony.



Why are Else branches difficult to analyze?

Easy when checking trace property:
Only need to check that disequalities can be satisfied.

Issue when considering equivalence of P and Q:

- A single action on P can be simulated in Q by different actions

Example

$$P = \mathbf{in}(x)\mathbf{out}(x)$$

$$Q = \mathbf{in}(x)\ \mathbf{if}\ x = a\ \mathbf{then}\ \mathbf{out}(x)\ \mathbf{else}\ \mathbf{out}(x)$$

- Not stable by substitution

Example

$x = w_1$ does not imply $a = w_1$

Disequalities: our approach

If an identity exists on P but not on Q due to a disequality test then a test pass on Q where the disequality has been replaced by an equality and on P.

Symbolic run:	$\mathbf{receive}(c, h(x)).\mathbf{send}(c)$
On process Q:	$\mathbf{in}(c, x).[x \neq h(0)].\mathbf{out}(c, 0)$
Resulting trace:	$\mathbf{in}(c_{\mathrm{art}}, y).\mathbf{in}(c, x).[x = h(y)].[x = h(0)].\mathbf{out}(c, 0)$

This approach uses the procedure for positive processes as a black box.

Case studies

Akiss is implemented in Ocaml and is available on Github.
`https://github.com/akiss/akiss`

We test several protocols:

- AKA (attack in 3m)
- BAC (attack in 1m30)
- PAP (safe in 4s)
- RFID protocols: KCL, LAK, LD, MD, NSL xor, OTYT, SLK.
- Guessing attack: Nonce, Direct authentication, Gong

Current work: abstract structure to deal with interleaving

Issue

Currently all interleaving are verified independently: exponential blow up

$$P = \mathbf{in}(c, x).\mathbf{out}(c, f(x)) \mid \mathbf{in}(c, y).\mathbf{out}(c, g(y)) \mid \mathbf{in}(c, z).\mathbf{out}(c, h(z))$$

240 interleavings

Idea to solve the problem

- $\mathrm{k}_w(R, t)$: make w abstract to represent a set of interleavings

Conclusion

A procedure to analyze equivalence of processes.

- bounded number of sessions
- support for xor
- support for disequalities

Properties:

- Sound and complete
- Termination granted without xor
- An efficient tool