How to Verify Privacy Automatically

Laouen Fernet[1, *], Sebastian Mödersheim[1] and Luca Viganò[2]

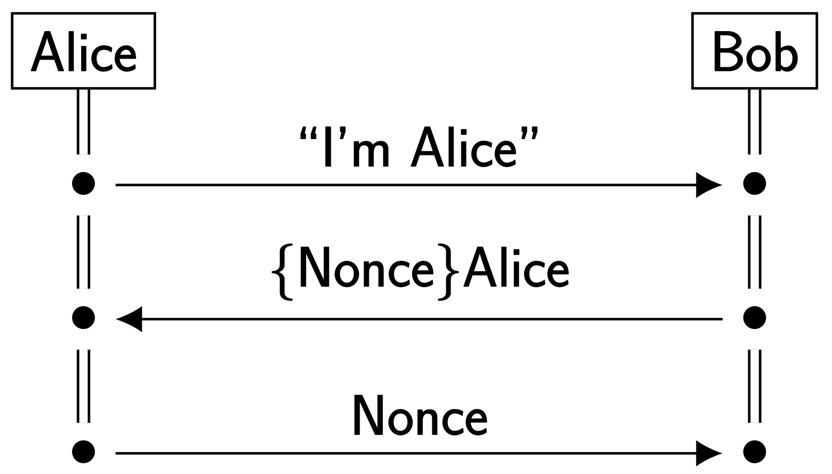
[1] DTU Compute, Kgs. Lyngby, Danmark   * lpkf@dtu.dk
[2] KCL Informatics, London, United Kingdom

1. Problem: Privacy in Security Protocols

With the current trend of **increasing digitalization**, more and more applications use private information to provide various services.

Health   Payment   Transport   Voting

We need **strong guarantees** that digital applications respect privacy. We focus on applications written as **security protocols**: participants exchange messages, often using cryptography.



Example of a simple security protocol

We use **(α, β)-privacy** to characterize privacy with logical formulas.

α is the **payload**: information intentionally disclosed.

β is the **technical information**: intruder knowledge.

Example: $\alpha \equiv x_1, \ldots, x_n \in$ Agent $\rightarrow$ unlinkability goal
If $\beta \Rightarrow x_1 =$ Alice or $x_2 = x_3$, then it is a **violation of privacy**: the intruder has learned more than allowed.

Input

```
...
* x in {a,b,i}.  # Pick an agent
* y in {yes,no}. # Flip a coin
receive M.
try N = dcrypt(inv(pk(s)),M) in
  if y = yes then
    new R. send crypt(pk(x),pair(yes,N),R)
  else
    new R. send crypt(pk(x),no,R)
...
```

2. Objective: Automated Verification

The specification of a protocol defines several **atomic transactions**. **Transition system**: executing a transaction leads to the next state. In each state, a pair (α, β) defines the privacy goals and intruder knowledge.

Our objective: **decide privacy expressed as a reachability property**.

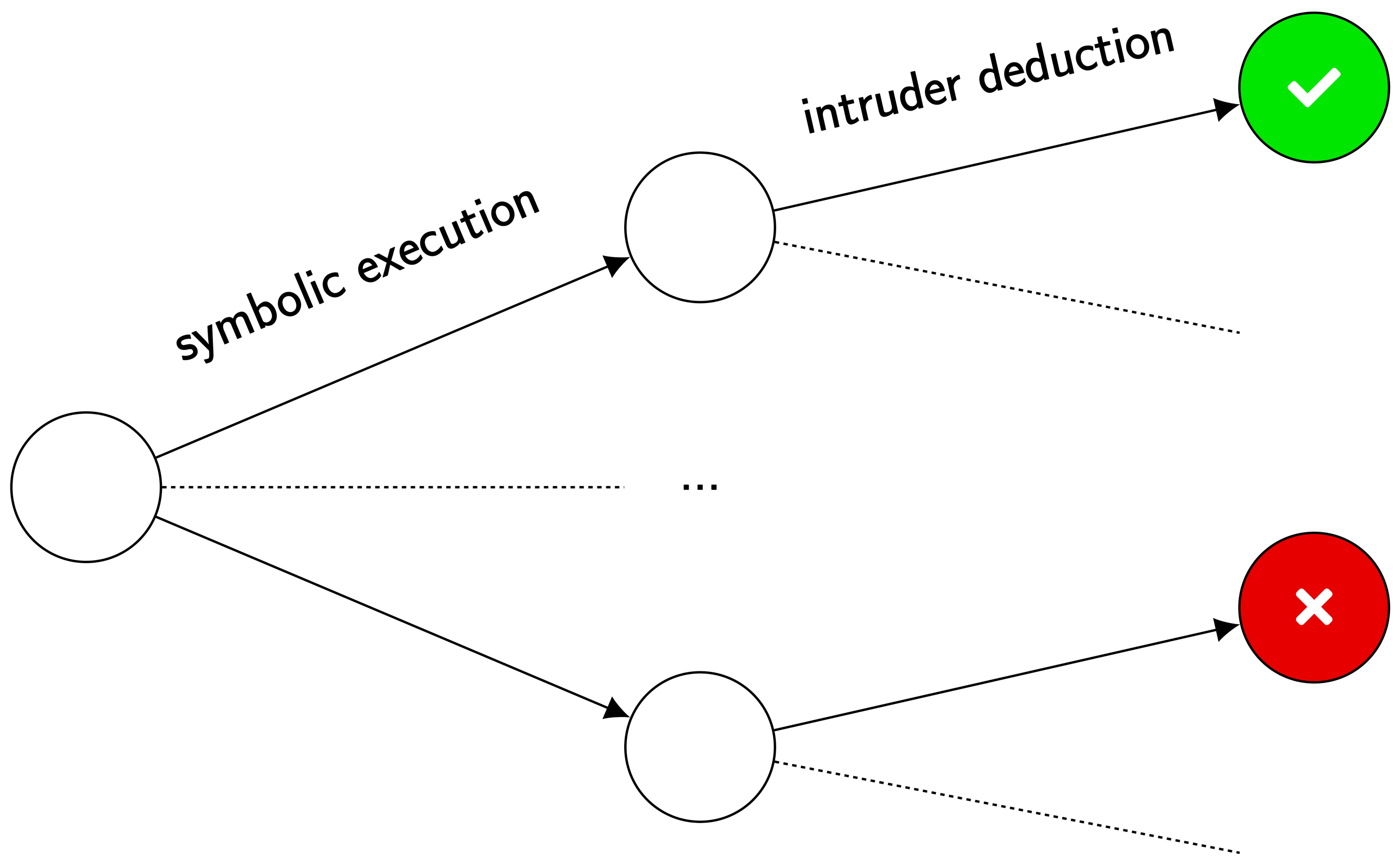
Main challenge: verify an **infinite state space**.

1. The intruder has infinitely many choices when sending messages.
   $\rightarrow$ We use a **symbolic representation** with constraint systems.
2. Some transaction can always be executed.
   $\rightarrow$ We only look at a **bounded number** of transactions.

Our **decision procedure**:

1. Execute a transaction.
2. Saturate the intruder knowledge by decrypting and comparing messages.
3. Verify (α, β)-privacy in the symbolic states reached.
4. Repeat until the protocol execution meets the bound specified.

Computation



3. Results: Tool Support

Main result: decision procedure along with proofs of **correctness**. Implementation: we now have a **prototype tool**.

Input: specification of the protocol with a bound.
Output:

- either **attack trace**: reachable state with a violation of privacy.
- or confirmation that **the privacy goals are achieved**.

Case studies: models for several protocols and analyses of privacy guarantees.

- Basic Hash: unlinkability holds but no forward privacy.
- OSK: known attacks on unlinkability.
- ICAO 9303 BAC: known attacks on unlinkability in some variants + unlinkability holds in the corrected variants.
- Private Authentication: we found the **strongest privacy goal**, which is subtle and goes beyond unlinkability.

Conclusion: (α, β)-privacy allows for **declarative and intuitive** specification of privacy and automated verification is **practical**.

Output

```
Privacy violation found after 2 transactions.
alpha: x in {a,b,i} and y in {yes,no}
beta implies: x = i and y = no
(alpha, beta)-privacy does not hold for the
state where the intruder has sent
crypt(pk(s),R1,R2) and has successfully
decrypted the reply from the server.
...
```