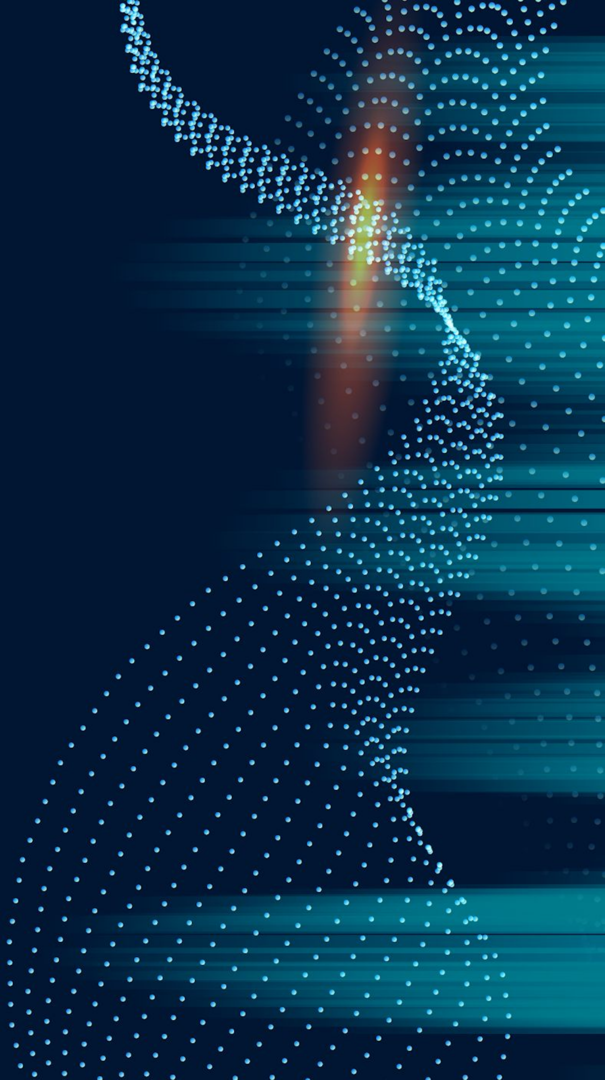
Deducibility constraint systems. Verification of cryptographic properties

Special Topics in Security and Applied Logics I

Mocanu Alexandru
Group 510 - Security and Applied Logics

Motivation

- **Verification of communication protocols between parties**
- **Cryptography verification**
  - Abstractizations can be used (assuming a certain standard for the cryptographic functions)
  - Certain properties still require to be analyzed
- **Formal verification**
  - Requires a logic to model the design of protocols and of security properties
  - Example of models: transition system, deducibility constraint system



Deduction system

- **Similar to natural deduction**
- **Deduction rules**
  - Symmetric encryption
  - Asymmetric encryption
  - Pairing
  - Digital signature
- **Proof can be organized as trees**
  - Deduce an axiom
  - Apply a deduction rule

$$\dfrac{\dfrac{\dfrac{M \vdash enca(k,b) \qquad M \vdash priv(b)}{M \vdash k} \qquad M \vdash enc(m,k)}{M \vdash m} \qquad M \vdash enc(h(m),m)}{M \vdash h(m)}$$

Deducibility constraint system

- **A constraint:**
  - The intruder knowledge
  - One message sent by a corrupted agent
- **The knowledge of the attacker increases**
- **Variables** = changeable parts from a message
- **Solution** = a possible instantiation of the protocol

$$T_0 = \{a, b, i, priv(i)\}$$

$$T_1 = \{a, b, i, priv(i), enca((a, n_a), i)\} \Vdash enca((a, x), b)$$

$$T_2 = \{a, b, i, priv(i), enca((a, n_a), i), enca((x, n_b), a)\} \Vdash enca((n_a, y), a)$$

$$T_3 = \{a, b, i, priv(i), enca((a, n_a), i), enca((x, n_b), a), enca(y, i)\} \Vdash enca(n_b, b)$$

Simplifying constraint systems

- Offers a solution to solve constraint systems
- *Solved* form
- Rules of simplification
- Transformations conserve solutions
- Guarantees termination

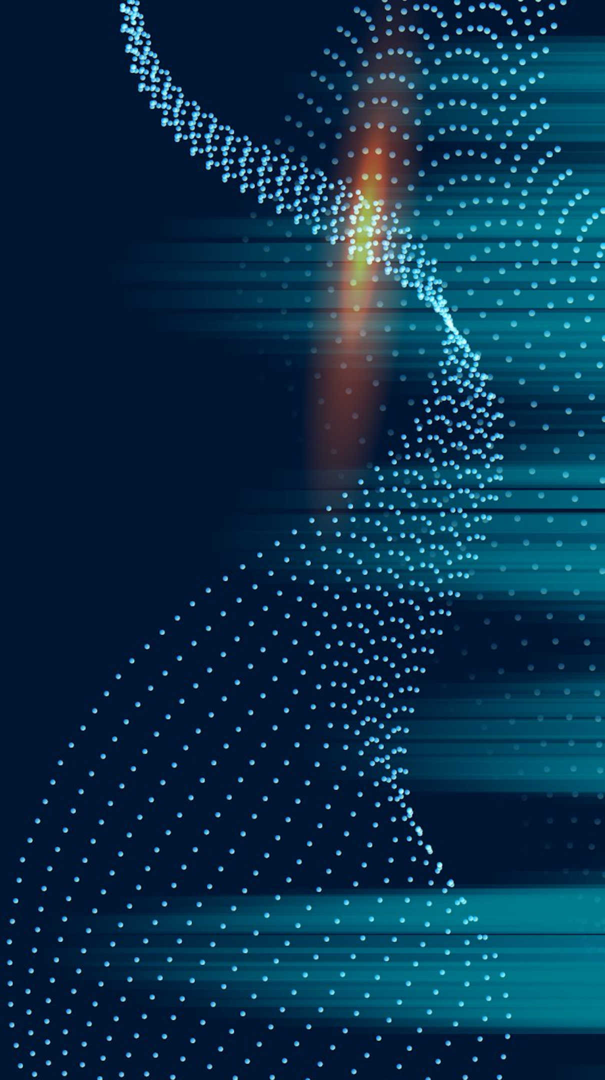
$$\begin{cases} T_1 \Vdash \langle \mathrm{enca}(x,a), \mathrm{enca}(y,a) \rangle \\ T_2 \Vdash k_1 \end{cases} \overset{R_{\langle\rangle}}{\rightsquigarrow} \begin{cases} T_1 \Vdash \mathrm{enca}(x,a) \\ T_1 \Vdash \mathrm{enca}(y,a) \\ T_2 \Vdash k_1 \end{cases} \overset{R_{\mathrm{enca}}}{\rightsquigarrow} \begin{cases} T_1 \Vdash x \\ T_1 \Vdash a \\ T_1 \Vdash \mathrm{enca}(y,a) \\ T_2 \Vdash k_1 \end{cases} \overset{R_1}{\rightsquigarrow} \begin{cases} T_1 \Vdash x \\ T_1 \Vdash \mathrm{enca}(y,a) \\ T_2 \Vdash k_1 \end{cases}$$

Example from: Hubert Comon-Lundh, Veronique Cortier and Eugen Zalinescu. “Deciding security properties for cryptographic protocols. Application to key cycles”

Memorization

- **Downside to simplification:**
  - Complexity can grow exponentially
- **Solution:**
  - Memorize transformed constraints

- **Correct and complete**
- **The complexity is polynomially bounded**



Security properties

- **Constraint systems**
  - Do not describe security properties
  - Need to express what to check (similar with claims)
- **Security property** = formula
- **Attack** = solution to both constraint system and security property

Cryptographic properties

- **Can be extended for cryptographic properties analysis**
- **Encryption viewed as a relation between keys**

- **Key cycles**
  - A general weakness of encryption functions
  - Do not want to have cycles in the encryption relation

- **Key ordering**
  - Impose a rule on the encryption relation
  - Example of utility: forward secrecy

Other security properties

- **Small logic**
  - Basic connectives and equality
  - Example: authentication
- **Time constraints**
  - Each message is labelled with a timestamp
  - Extend the constraint system to allow inequations on timestamps
  - Example: fresh values

Thank you!