FM – biased 14-11-2013

KATHOLIEKE UNIVERSITEIT LEUVEN

Formal verification of cryptographic protocols

Applied to the smart metering use case

KU Leuven: ELECTA – COSIC

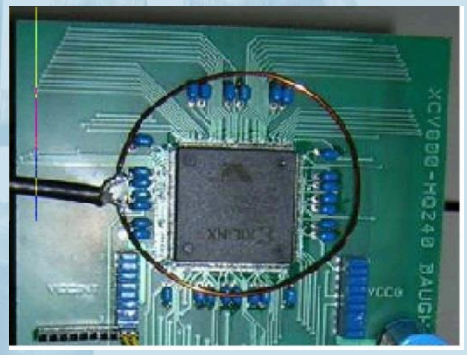
Department Electrical Engineering

ELECTA = Electric systems and Computer Architectures

COSIC = COmputer Security and Industrial Cryptography

COSIC

- Cryptographic algorithms
- Design
- Analysis
- Secure Implementation



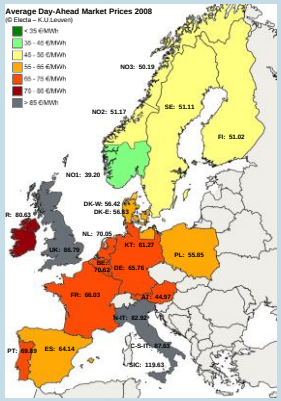
ELECTA

Power systems

Distributed generation

Techno-economic issues

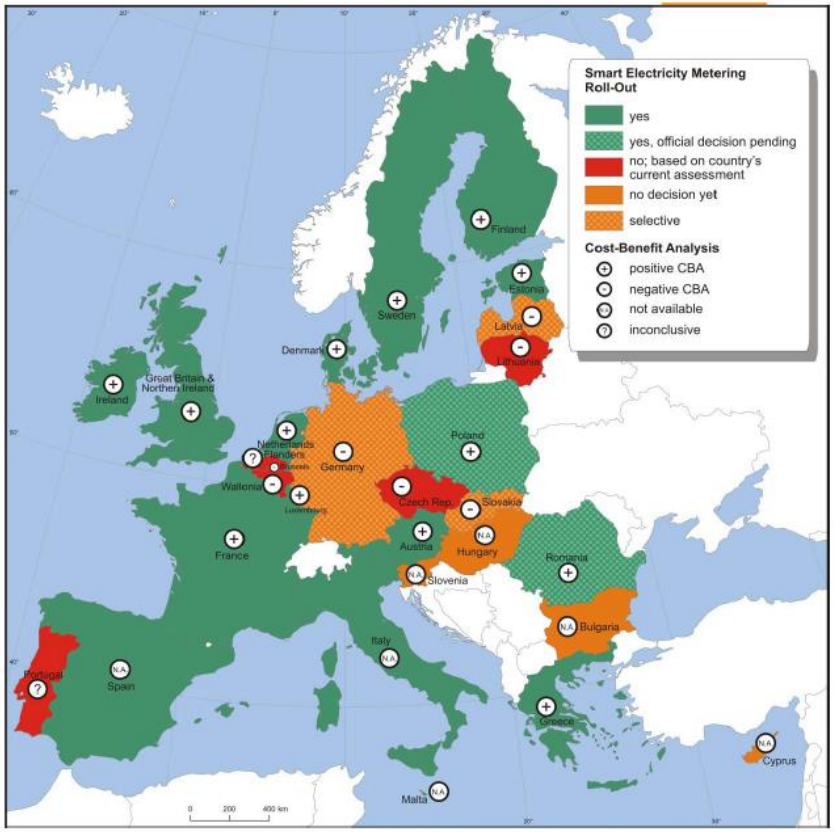
Smart grids



Overview

- Why smart metering security?
- Introduction to security protocols
  - Basic building blocks
  - Capabilities/limitations of attacker
- Formal verification of security protocols
  - History
  - Verification approaches
  - State of the Art
- Use case KU Leuven: Smart metering application
  - Formal methods in smart metering
  - Lab setup
  - Expected outcome

Smart metering in Europe



Roll out of smart electricity metering by 2020

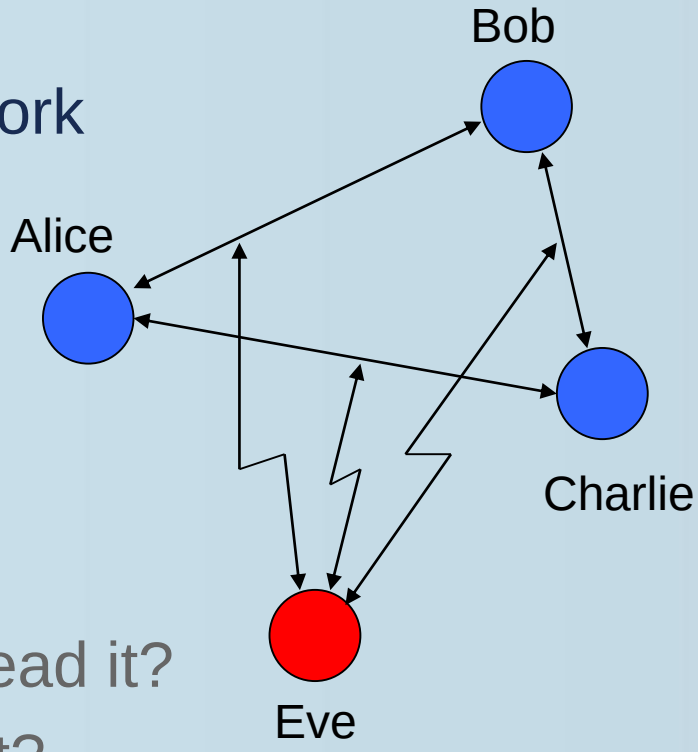
- ❖20 CBA, 16+ MS rolling out
- ❖75% EU consumers
- ❖198 million meters
- ❖ € 35 billion

Dr. Manuel Sánchez Jiménez © European Commission manuel.sanchez-jimenez@ec.europa.eu 9/19

- No standard implementation yet
- Security issues hot topic

What are cryptographic security protocols ?

- Two or more parties
- Communication over insecure network
- Active adversary can
  - Intercept messages
  - Forge messages
  - Replay messages
- Protocol goals
  - Confidentiality — Who can read it?
  - Authentication — Who sent it?
  - Integrity — Has it been altered?
  - Anonymity — Who could've sent it?….



Basic building blocks

- In order to achieve the security goals, crypto protocols are constructed from *cryptographic primitives:*
  - Symmetric Encryption: encryption and decryption with the same key (eg AES Rijndael algorithm, 2001)
  - Public Key Encryption: encryption with a public key, decryption with corresponding private key (eg RSA)
  - Signatures: authenticate data with a publicly verifiable digital signature (eg RSA, DSA)
  - Hash function: one-way reduction function (arbitrary length to fixed length)
  - ...

Capabilities of the adversary

- Total network control:
  - See all exchanged messages
  - Delete, alter, inject and redirect messages
  - Initiate new communications (has a valid identity)
  - Reuse messages from past sessions
  - Try to guess password
- Usually known as “Dolev-Yao” (first authors to propose a formal security model)
- Good for formal automatic verification:
  - Finite participants, infinite messages -> decidable
  - Infinite participants -> undecidable

Limitations of the adversary

- Assume attacker is a Turing machine running in poly-time, so he cannot
  - Break strong cryptography material
  - Break “hard” problems (factor large primes, solve discrete log problem in large groups, …)
  - Guess keys
  - Guess pseudo-random values (eg. nonces)
- Get other identities (identity theft)
- Use side-channel information:
  - Time messages & statistical traffic analysis
  - Power or electro-magnetic analysis

Cryptographic protocols go wrong

- Historically, one keeps finding simple attacks against protocols
  - even carefully-written, widely-deployed protocols, even a long time after their design & deployment
  - simple = no need to break cryptographic primitives
- Why is it so difficult?
  - concurrency + distribution + cryptography
    - Little control on the runtime environment
  - active attackers
    - Hard to test
  - implicit assumptions and goals
    - Authenticity, secrecy

A bit of history

- In 1978 Needham and Schroeder stated that

  "Protocols such as those developed here are prone to extremely subtle errors that are unlikely to be detected in normal operation. The need for techniques to verify the correctness of such protocols is great, and we encourage those interested in such problems to consider this area."

- Dolev and Yao first formalize N&S problem in early 80s
  - They use algebraic, idealized cryptography
    - Shared key decryption: decrypt({M}K,K) = M
    - Public key decryption: decrypt({| M |}KA+,KA-) = M
  - Their work is now widely recognised, but at the time few proof techniques, and little applied

- Since then, this has been an active field (mainly since 1995), and many formal verification tools have been proposed

Verification Approaches

- Cryptographic analysis:

  Protocol security reduced to number-theoretic assumptions

- Formal model checking:

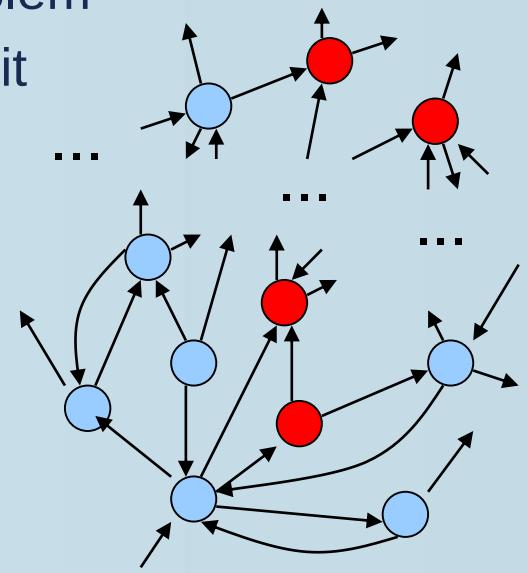
  Build state transition graphs for some system instances and check as well as possible

- Theorem proving:

  Phrase problem as idealized mathematical problem

  (perfect crypto, other simplifications) and prove it

- Others



Current State of the Art

- Back-tracing from “bad” state to an initial state (Scyther)
- Forward state space exploration (AVISPA)
- Automated theorem provers using concurrent process calculus (ProVerif)

Use case description

Formal verification of smart metering security and privacy protection

Smart metering

- Bidirectional communication between meter and DSO
  - Automatic meter reading for billing
  - Grid management
  - DSO can send commands to the meter (e.g. remote switch off)
- Already massive roll-out in Europe
- No standard implementation
- Problem: security and privacy

Security and privacy problems

- Altering meter data
  - Incorrect billing -> economic losses
  - Influence grid management -> might lead to massive black out
- Reading out meter data every 15 min -> leaks sensitive information
- Sending false commands to meters

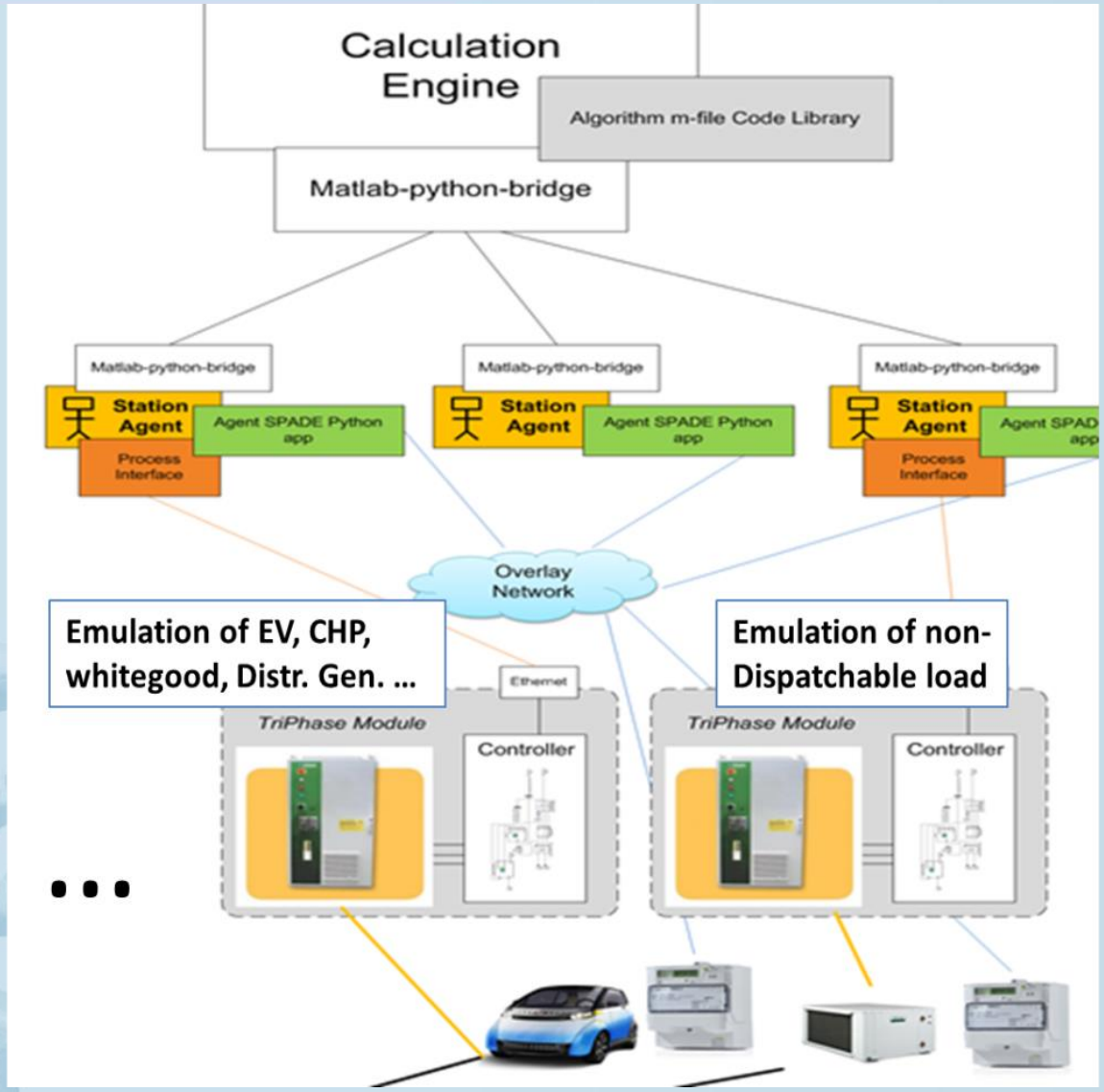
Our approach

- Formal analysis of some of the key protocols in the security architecture
- Demonstrating attack feasibility and impact in laboratory set-up
- Overall report of the results

Formal verification of security protocols

- Not very common, requires extensive knowledge of the tool, its functionality and its limitations
- Not feasible for complex systems, only for specific protocols
- Three different tools
  - SCYTHER: efficient, underlying algorithm publicly available
  - AVISPA: extensive documentations, tools to help the user
  - ProVerif: has been used to verify commonly used protocols (e.g. TLS)

Laboratory set-up



Station agent

Raspberry Pi

Expected outcome

- **Formal tool useful to verify smart meter security**
  - ❑ Which are most efficient and user-friendly?
  - ❑ Is it possible to detect security threats in the system?
  - ❑ If yes, can the threat be reproduced in a laboratory environment?
  - ❑ If no: How can they be adapted to be more practically useful for smart metering systems?

Thank you!