Formal Methods for Security
Why? – How?

Roberto Segala
University of Verona
Formal Methods and Formal Verification

Why?
Why Formal Analysis?

• 1994: The pentium processor computes wrong divisions
  - INTEL forced to replace most processors
  - Economic damage of 450 million US Dollars

• 1995: The software MacInTax spreads the secrets of US tax payers
  - Error in the debug code distributed with MacInTax
  - Users can use it to access the server of Intuit
  - Everybody can read and modify any tax form
Why Formal Analysis?

• 1995: Problems in Denver Airport
  - The fully automated baggage system fails
  - Scheduled to open in 1993
  - The system looses or tears apart luggage
  - Considerable congestion
  - Considerable lack of design
  - In 2005 the system is still not working
  - The system is too complex
  - Extensive research activity is necessary
Why Formal Analysis?

• 1996: Vector Ariane 5 explodes during take-off
  - The control software assigns a 64 bit number to a 16 bit variable
  - The code was recycled from Ariane 4
  - Ariane 5 is fast and its lateral speed does not fit in 16 bits
  - Result: overflow - the system shuts down
  - The back up computer is started
  - ... but the software is the same
  - Result: again overflow - the system shuts down
  - Ariane, without guidance, self destroys
  - Damage: 1 billion Euros
Why Formal Analysis?

• 1982 Mutual exclusion solved with small shared variables
  - Rabin proposes a randomized distributed algorithm
  - The proof is semi-formal but credible

• 1990 Some problems appear
  - Nancy Lynch gives a lecture on Rabin’s algorithm
  - Roberto Segala is the scribe and tries to formalize the proof
  - Problem in an informally obvious step
    • Two events are compared but they belong to different probability spaces
  - Nondeterminism is the cause of the problem

• 1991 An attack is found

• Later many other algorithms turned out to be bogus
Why Formal Analysis?

- 1978: Needham and Schroeder
  - Propose an authentication protocol
  - The correctness proof is semi-formal

- 1981: Problems with freshness
  - Replay attacks are possible

- 1995: An attack found
  - Parallel sessions may lead to attack

- Needham: you changed my definitions

- Later: many protocols have been attacked
Lessons that we can Learn

• Formal methods are useful (necessary)
  - Need to define what we want
    • Objectives should be clear and accepted
    • We should communicate with others
  - Need to prove properties rigorously
    • We may miss pieces otherwise
    • We need techniques
  - Need modular verification techniques
    • We want to reuse existing proofs
  - Need ways to automate the analysis
    • Large systems require considerable effort

- **Provable security** [GM84]
  - Based on Turing Machines (computational model)
  - Proofs by reduction to known difficult problems

- **Dolev-Yao model** [DY83]
  - Based on automata theory
  - Perfect cryptography
  - Need to know relationship to computational model

- **Universally composable security** [Can01]
  - Based on Interactive Turing Machines
  - Specification includes accepted attacks

- **Reactive Simulatibility** [PW01]
  - Based on Probabilistic I/O Automata
  - Similar to UC framework
Provable Security

- Let $h$ be a computationally hard function
- Let $C$ be a cryptographic primitive
  - Collection of PPT algorithms that compute some functions
- State correctness of $C$ as follows
  - There is no PPT algorithm $A$ that computes some function $f$
- Prove correctness of $C$ as follows
  - Suppose for the sake of contradiction that $A$ exists
  - Build a PPT algorithm for $h$ that uses $A$ as a black box
  - This contradicts the hardness of $h$

- Correctness of $C$ relies on hardness of $h$
Dolev-Yao Model

- Agents communicate through adversarial network
  - Network remembers everything
  - Network may block or reroute messages
  - Network may cast new messages
Dolev-Yao Model: Assumptions

• Computational
  - Messages are bit strings
  - Adversary governed by PPT functions

• Symbolic (typical use of the model)
  - Messages are symbols
  - Cryptography is perfect
  - Adversary power limited by a deduction system
    • Nonces are always fresh
    • No ability to decrypt without decryption key
  - Adversary is nondeterministic
Symbolic Dolev-Yao Model

- **Analysis is simple**
  - The system is described by an automaton
  - Show that no path leads to failure or attack
  - Plenty of techniques from concurrency theory
    - Invariants
    - Compositional analysis
    - Language properties
    - Model checking

- **Sound with respect to computational [AR00]**
  - Attack in computational model yields attack in symbolic model
  - Need some assumptions on underlying cryptoprimitives
    - Non malleability
Symbolic Dolev-Yao Deductions

- $A \vdash X$, $A \vdash Y \Rightarrow A \vdash (X,Y)$
- $A \vdash (X,Y) \Rightarrow A \vdash X$
- $A \vdash (X,Y) \Rightarrow A \vdash Y$
- $A \vdash X$, $A \vdash k \Rightarrow A \vdash \{X\}_k$
- $A \vdash \{X\}_k$, $A \vdash k \Rightarrow A \vdash X$

- **Automaton transitions**
  - Agents add messages to adversary
  - Adversary casts messages according to deductions

- **Invariants**
  - Signature deducible only if it exists already

- **Property**
  - Answers always generated by correct agents
UC [Can01] and RSim [PW01]

Motivation

Secrecy | Authentication
--- | ---

Secrecy and Authentication

Implemented by

A

B

Implemented by

G

F

A

B

F

G

?
UC-Framework [Canetti]

Ideal functionality

Simulator

∃

Environment

∀

Real protocol

Adversary

∀

∀
Reactive Simulatability
[Pfitzmann Waidner]

- Similar to UC Framework
- Based on PIOAs rather than ITMs
- More elaborated on verification techniques
- Large collection of definitions
  - Crypto library [BPW03]
Fine, but how do we prove Facts?

- **Provable security**
  - Semi-formal arguments
  - A lot of wording

- **Dolev-Yao**
  - Semi-formal arguments
  - ... or typical arguments from concurrency theory

- **UC Framework**
  - Semi-formal arguments

- **Reactive simulatability**
  - Semi-formal arguments
  - “Simulation” up to “error sets”
  - Negligible probability of error sets
Can we be More Rigorous?

- Use Dolev-Yao and Soundness
  - Concurrency theory has plenty of techniques

- Use Process Algebraic formalisms [MRST06 and earlier]
  - Expressions denote PPT computable functions
  - Equivalence denotes undistinguishability
  - Axiomatic reasoning

- Use game transformations [Sho04,Bla05]
  - Correctness in provable security expressed as a game
  - Transform games preserving correctness

- Use Automata Theory [CCKLLPS06,ST07]
  - Add computational assumptions
  - Extend known techniques (simulation method)
Game Transformations
Example: El Gamal

- $x \leftarrow \mathbb{Z}_q, \alpha \leftarrow \gamma^x$
- $r \leftarrow R, (m_0, m_1) \leftarrow A(r, \alpha)$
- $b \leftarrow \{0, 1\}, y \leftarrow \mathbb{Z}_q, \beta \leftarrow \gamma^y, \delta \leftarrow \alpha^v, \xi \leftarrow \delta m_b$
- $b \leftarrow A(r, \alpha, \beta, \xi)$

- $x \leftarrow \mathbb{Z}_q, \alpha \leftarrow \gamma^x$
- $r \leftarrow R, (m_0, m_1) \leftarrow A(r, \alpha)$
- $b \leftarrow \{0, 1\}, y \leftarrow \mathbb{Z}_q, \beta \leftarrow \gamma^y, z \leftarrow \mathbb{Z}_q, \delta \leftarrow \alpha^z, \xi \leftarrow \delta m_b$
- $b \leftarrow A(r, \alpha, \beta, \xi)$

$D(\alpha, \beta, \delta)$

- $x \leftarrow \mathbb{Z}_q, \alpha \leftarrow \gamma^x$
- $r \leftarrow R, (m_0, m_1) \leftarrow A(r, \alpha)$
- $b \leftarrow \{0, 1\}, \xi \leftarrow \delta m_b$
- $b \leftarrow A(r, \alpha, \beta, \xi)$
UC-Framework [Canetti]

- Ideal functionality
- Simulator
- Real protocol
- Adversary
- Environment

∀ ∈ ℝ

∃ ∈ ℝ

∀ ∈ ℝ
UC-Security with PIOAs
[Canetti, Cheung, Kaynar, Liskov, Lynch, Pereira, Segala, Vaandrager]
Nondeterminism: why There?

• If we have several components
  - Who moves first (nondeterminism)?
  - Can the order of operations reveal secrets?

• If we expect input
  - What input do we receive?

• If we have partial specification
  - How do we implement (nondeterminism)?

• Nondeterminism resolved by a “scheduler”
  - Not all resolutions are safe
Example of Nondeterminism

• Order of messages may reveal one bit of $s$ to $E$
Example of Nondeterminism
MAP1 Protocol [BR93]

- Authentication protocol
  - Symmetric key signature schema
  - Computational Dolev-Yao
  - Adversary queries agents

- Potential problems
  - Let $s$ be the shared key
  - Adversary queries $k$ agents
  - Agent $i$ replies if $i^{th}$ bit of $s$ is 1
  - The adversary knows the shared key
Approaches to Nondeterminism

- Reduce the power of the scheduler
  - Process Algebras
    - Scheduler sees only enabled action type
  - UC framework
    - ITMs have a token passing mechanism
    - No nondeterminism
  - Reactive simulatibility
    - Again token passing mechanism
    - Nondeterminism based on local information only
  - Task PIOAs
    - Define equivalence classes of states and actions
    - Scheduler sees only equivalence classes, not elements

- Careful specifications
  - Avoid dangerous nondeterminism in the specification
  - Is it always possible?
Example of Nondeterminism
MAP1 Protocol [BR93]

- Authentication protocol
  - Symmetric key signature schema
  - Computational Dolev-Yao
  - Adversary queries agents

- Potential problems
  - Let $s$ be the shared key
  - Adversary queries $k$ agents
  - Agent $i$ replies if $i$th bit of $s$ is 1
  - The adversary knows the shared key

- Solution
  - One query at a time
  - Wait for the answer (oracle model)
Concluding Remarks

• Formal methods are useful
  - Semi-formal proofs may be wrong
  - Semi-formal proofs require attention

• There are several approaches
  - Computational, symbolic, compositional
  - Suitable for cryptoprimitives or protocols

• We need proof techniques
  - Algebraic, symbolic, automata theoretic

• Nondeterminism arises and gives problems
  - Restrict resolution of nondeterminism
  - Avoid dangerous nondeterminism
What Else?

• Need more techniques
  - Can we have a uniform view
  - Can we relate better computational and symbolic approaches?
  - Any crucial differences between cryptoprimitives and protocols?
  - How about cross migration of techniques?

• Need more automation
  - ... but we need to understand what we automate