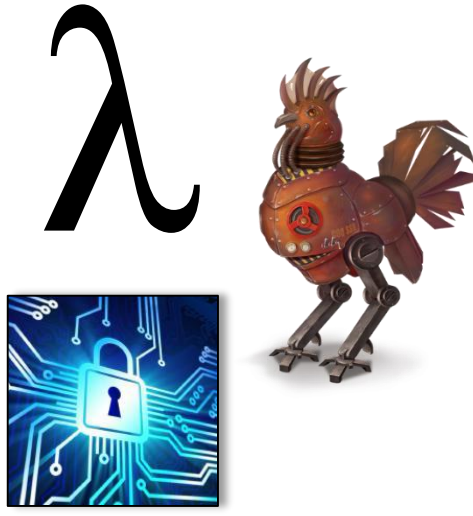
Foundations of Programming Languages, Verification, and Security



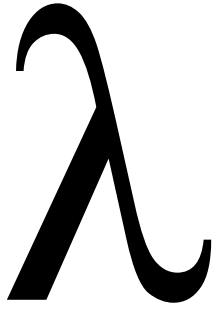
Winter 2023/24 @ Ruhr Uni Bochum

Lecturers: Rob Blanco and Cătălin Hriţcu

Teaching Assistant: Jérémy Thibault

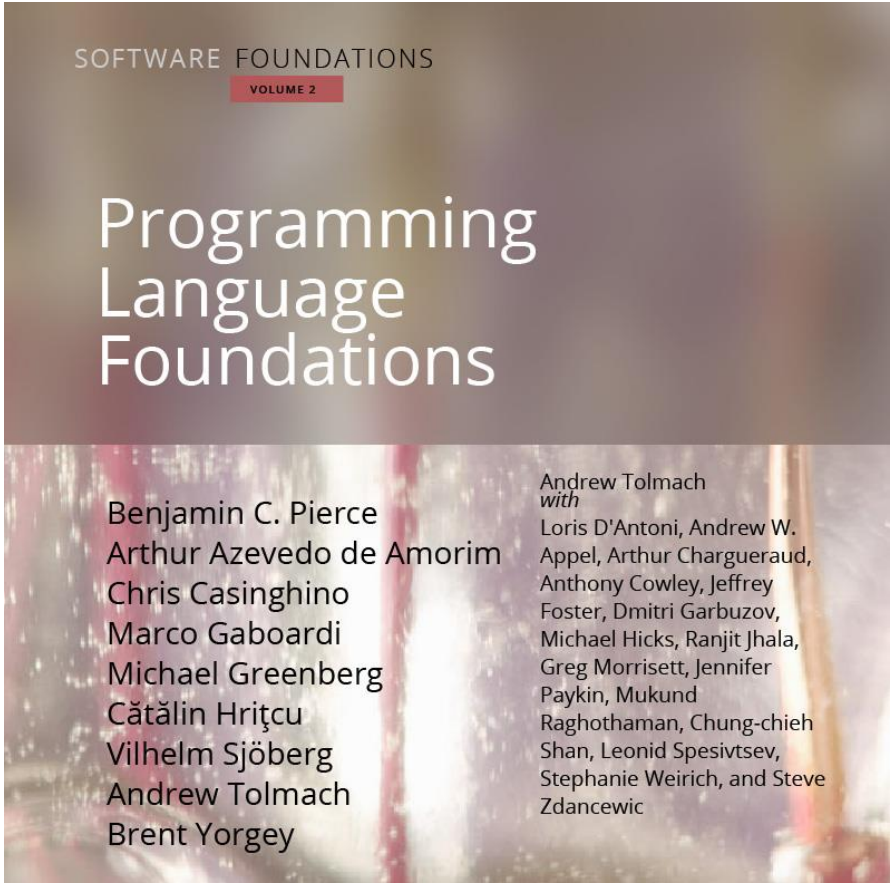
Max Planck Institute for Security and Privacy (MPI-SP)

Foundations of Everything



- **Programming Languages**
  - Imp and Simply Typed Lambda-Calculus (functional)
  - type systems, program equivalence, semantics, metatheory
- **Verification**
  - Hoare Logic: verify Imp programs
  - Relational Hoare Logic (RHL)

- **Security**
  - Information Flow Control: enforcing noninterference
    - Static enforcement: RHL, types, cryptographic constant time
    - Dynamic enforcement: Secure Multi-Execution, ...



Textbook written in Coq; our extended version at:

https://mpi-sp-foe-2023-24.github.io

Prerequisite: Proofs are Programs

1. Logic and proofs
2. Functional programming
3. Program verification

- **Using the Coq proof assistant**

- **Prerequisite:**
  Having attended the Proofs are Programs course last semester or having (self-)studied the Logical Foundations book:
  https://mpi-sp-pap-2023.github.io

Why it's helpful to have foundations for Programming Languages?

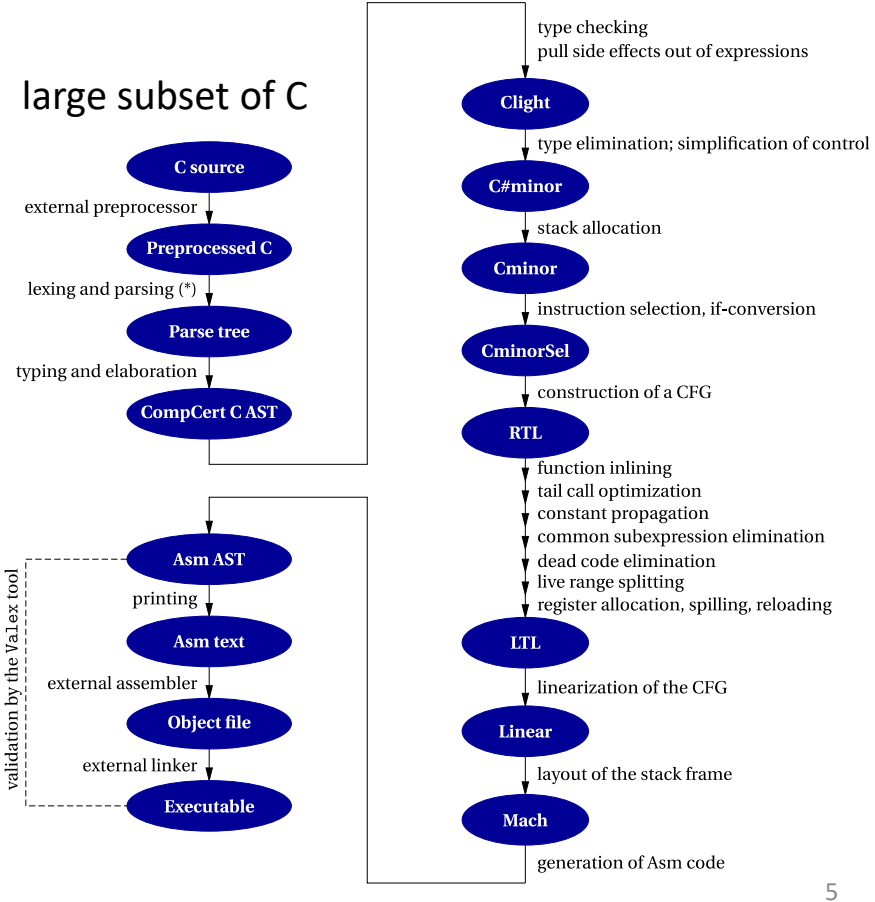
CompCert C compiler

verified in Coq to correctly compile,
unless program has "undefined behavior"

Rob, Jérémy, Catalin, et al:
Formally Secure Compilation of Compartmentalized C Programs

Extended CompCert with compartments
(mutually distrustful, with clearly specified interfaces, interacting via procedure calls)

Proved in Coq that the impact of undefined behavior is restricted



Motivation for verifying imperative programs

=proving mathematically that a **program** satisfies a **specification,**
for instance in Hoare Logic (introduced in this course)

- **the seL4 OS microkernel** (Isabelle/HOL), **the CertiKOS hypervisor** (Coq)
- **EverCrypt cryptographic library** (F*) -- shipping in Firefox and the Linux kernel
- **EverParse3D verified binary parsers** (F*) -- shipping in Windows kernel
  - Cătălin et al working on design of F* proof assistant, which combines dependent types (à la Coq) with Hoare-style verification (Dijkstra monads)
- **Libjade high-assurance post-quantum crypto library** (EasyCrypt, MPI-SP et al)
- **Rob working on verifying security of masking countermeasures** (EasyCrypt)
  - EasyCrypt proof assistant for Probabilistic Relational Hoare Logic (MPI-SP)

Motivation for secure information flow

- **Cryptographic constant time (aka secret independent timing)**
  - **widely-used programming discipline for writing cryptographic code** without leaking secrets via obvious cache side channels:
    - no secret-dependent branches and no secret-dependent memory accesses
  - can be checked by a very simple type system
  - the obtained guarantees formalized as a variant of noninterference
- **Speculative constant time**
  - Spectre: constant time code can still leak because of speculative execution
  - Stronger noninterference variant that prevents leaks in speculative executions

Lecture logistics

- 14 lectures: first 6.5 by Rob, last 6.5 by Cătălin, 1 by Jérémy
- Vacation 21 Dec to 8 Jan, so no lecture, no tutorials
- Public holiday on 1 November, so no tutorial
- We hope for a mostly in-person course
  - **So please attend physically whenever possible!**
  - When you **really cannot** attend physically
    you can use Zoom or watch the recording

Exercises (same as last semester)

- **This is a very hands on course**
- **New exercise sheet will be released on Moodle after most courses**
  - 1(-2) Coq file(s) with holes you have to fill (basically 1-2 book chapters)
  - so there will be 10-12 exercise sheets in total
- You have to **turn in your solution on Moodle before next course**
  - up to Wednesday at 23:59
- **Exercises count for 40% of final grade** (+5% in bonus points)
  - not required to do the optional exercises; they don't count for grade
- **Exercises are individual, please don't share your solutions in any way!**
- **Exercises highly recommended even if you're not taking this for credit**

Tutorials: Q&A about the exercise sheets

- **TA:** Jérémy Thibault <jeremy.thibault@mpi-sp.org>
- **New time: Wednesday at 14:15-15:45**
- Same room as the course (MB, flour 5, room SM-MO-506)
- You can come ask questions
- You can also come work on your own during tutorials
  - and ask questions as they arise
- **New**: you can also join tutorial via Zoom if you prefer
- **New**: you can also write emails to Jérémy to ask questions
  - We still have Moodle forum; but don't post (partial) solutions there

Exams

- **Midterm exam** (optional)
  - practice for the final exam
  - also written, on paper
  - duration: 60 minutes
  - bonus points: up to 10%
  - date: 29 November at 14:00
- **Final exam**
  - written, on paper
    - so we will also teach you how to write down proofs informally
  - duration: 120 minutes
  - 60% of the grade
  - date: 21 February
  - re-exam: 22 March

Early final exam registration?

- A week ago the CS examination office informed us that a new policy will apply to us, since our course has exercises during the semester:
  - Students who want to get credit points would need to register by Monday, 23 October (so soon!)
- To me this seems like a bad idea
  - Giving students less freedom in choosing courses
  - Making our courses less appealing to students who are not sure from the beginning whether this course is for them or not?
- Discussing with examination office later this week
- Please let me know what you think

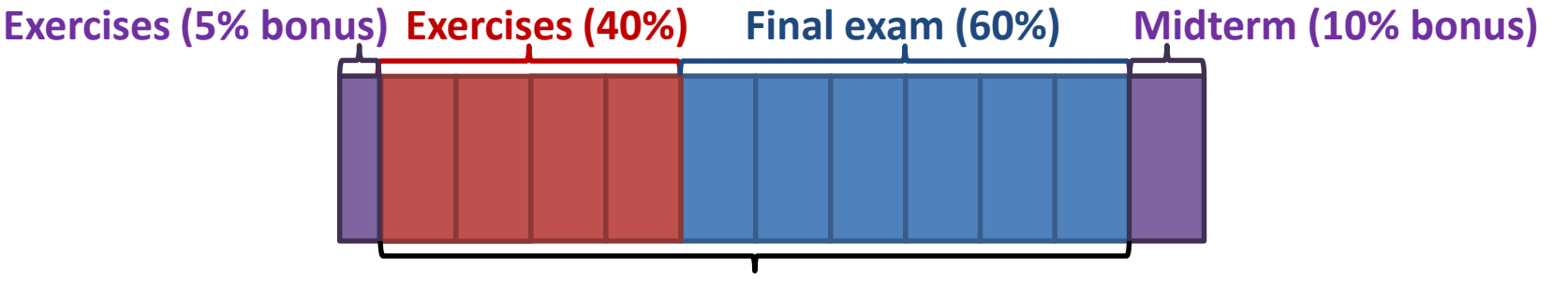
Final grade

CASA PhD students

– attendance requirement is to receive 50% of the exercises score

You can get credit if you study

- Computer Science MSc
- IT Security MSc
- Mathematics MSc



You need 50% of this to pass and get credit + attending final exam(!)

You need close to 100% of this to get maximum grade

Considering prequel course (next semester?)

Functional Programming

OCaml

Probably using the OCaml programming language

Potential lecturers: Rob Blanco, Cătălin Hriţcu, Clara Schneidewind