Lecture 1: Course Introduction

Zero-Knowledge Proofs

EECS 498/598 W26

Primary instructor: Paul Grubbs

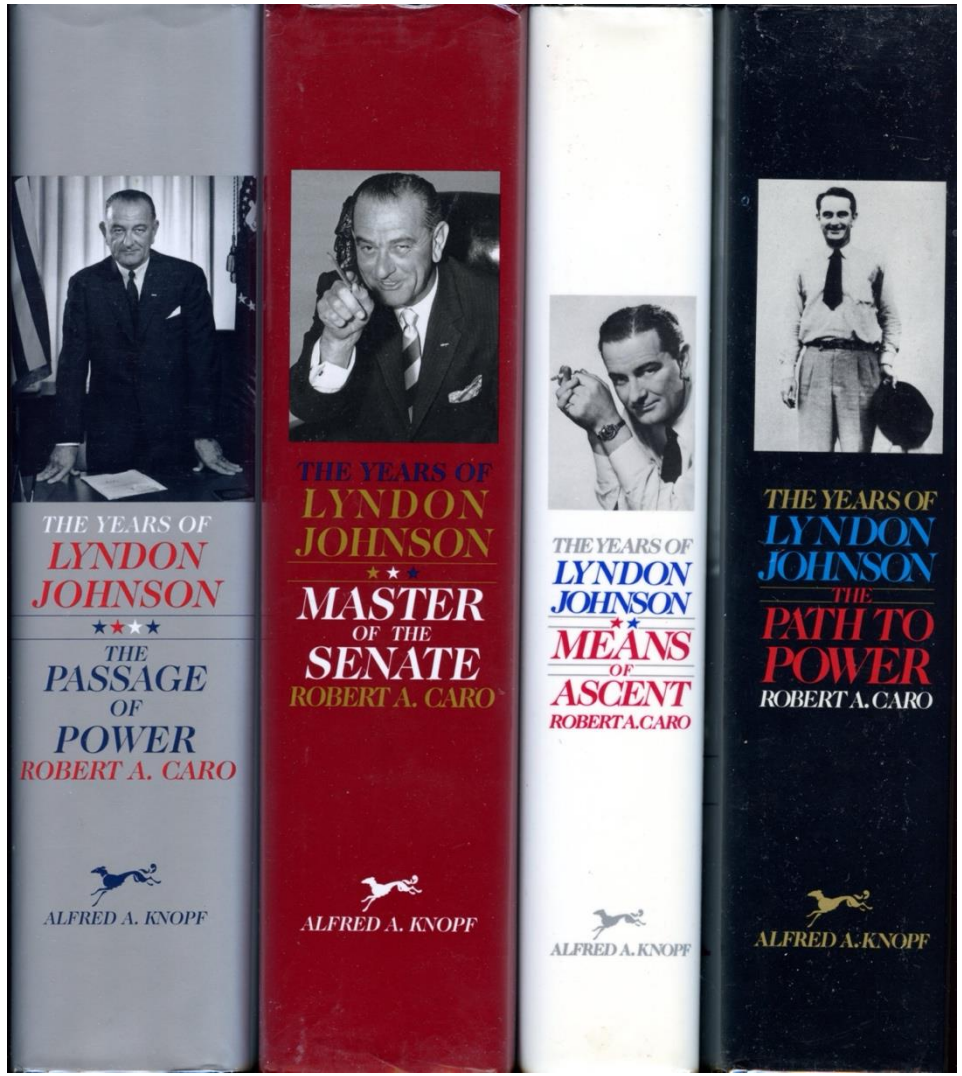
GSI: Chad Sharp

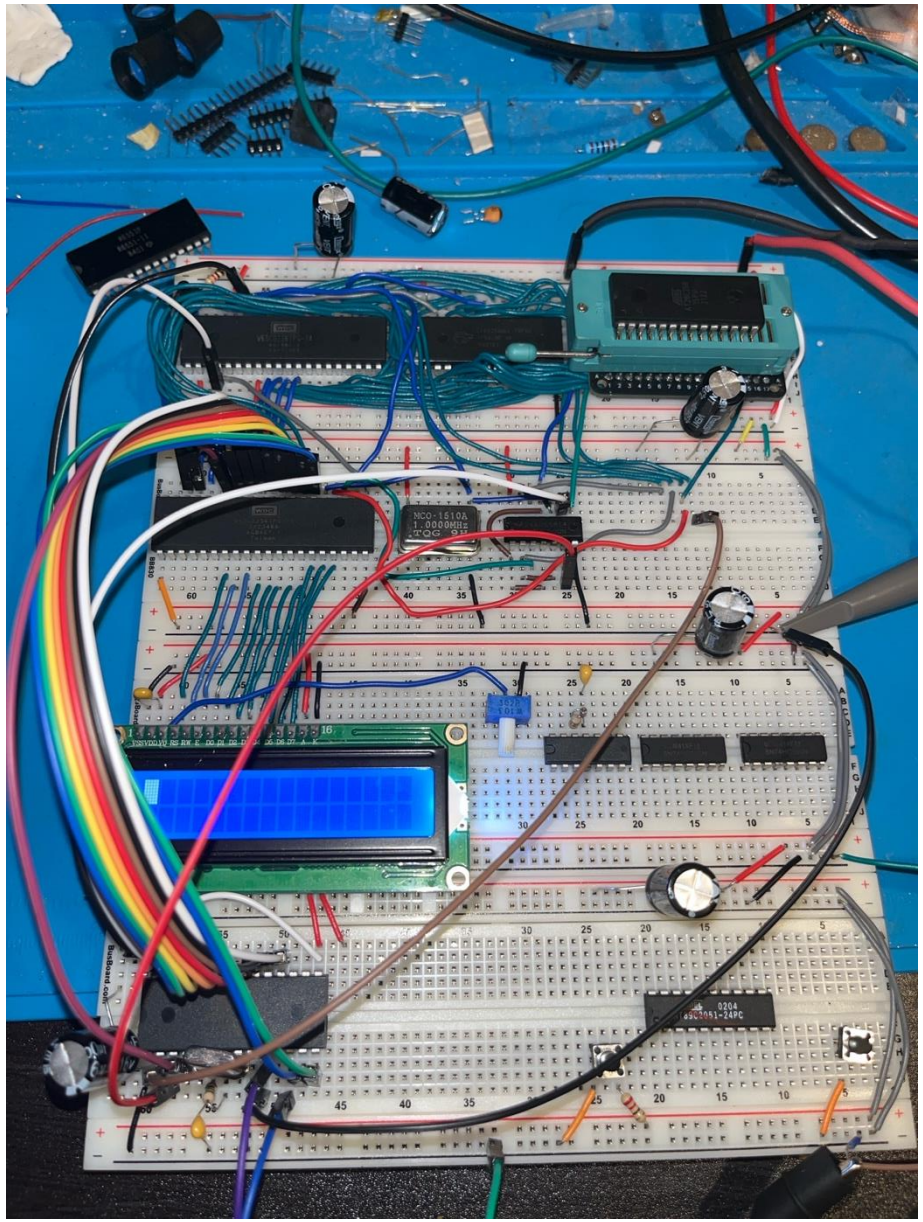
Introducing the course staff

- Midway through fifth year of faculty
- Office: 4709 BBB
- Office hours: 9-10:30am Wednesdays
  - Please come visit…I get lonely in there by myself all day ☹
- Career bio:
  - Did a postdoc at NYU before joining UMich
  - PhD Cornell (2020), undergrad at Indiana
  - Worked as a cryptography engineer

- website: https://web.eecs.umich.edu/~paulgrub/
- Research: applied cryptography
- Outside of work:
  - Reading (mostly nonfiction)
  - amateur radio (K8PAG), hobby electronics, homebrew/vintage computing
  - gaming (platformers, soulslike)
  - Parenting

Picture of Isaac redacted ☺


THE YEARS OF
LYNDON
JOHNSON
THE
PASSAGE
OF
POWER
ROBERT A. CARO
ALFRED A. KNOPF
THE YEARS OF
LYNDON
JOHNSON
MASTER
OF THE
SENATE
ROBERT A. CARO
ALFRED A. KNOPF
THE YEARS OF
LYNDON
JOHNSON
MEANS
OF
ASCENT
ROBERT A. CARO
ALFRED A. KNOPF
THE YEARS OF
LYNDON
JOHNSON
THE
PATH TO
POWER
ROBERT A. CARO
ALFRED A. KNOPF



Introducing our GSI

- Hi, I'm Chad Sharp.
- PhD Student (hopefully not for much longer!)
- Chief Rustacean
- Research interests: Zero-Knowledge Proofs, Post-Quantum Cryptography, and (recently) Formal Verification

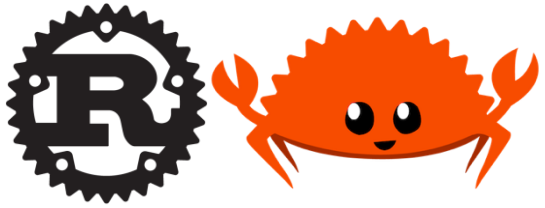
About You!

Go around the room and introduce yourself to us:

- Name, preferred pronouns
- one thing you want to get out of this class, or a topic you're excited about
- an interesting fact about yourself

About this class

- Accelerated introduction to modern zero-knowledge proofs
- Focus less on theory and more on design, implementation, and applications/systems
  - This class is brand-new; its approach to the material is novel.
  - Some course content has no extant pedagogy at all.
  - Everybody should be ready to be flexible ☺
- Secondary goals:
  - Learn principles of cryptography engineering:
    - "how do we take this nice algorithm on paper, turn it into secure software, and use that software to solve real problems?"
  - Learn Rust programming language



Course Setup

- Lectures on course content; I will give lectures.
  - Monday and Wednesday, 10:30am-noon, EECS 3427
- One discussion per week, led by Chad.
  - Friday, noon-1:30pm, DOW 2150
  - Main place to learn Rust concepts
- Office hours throughout the week (see syllabus)
- If you need to email the course staff, include [EECS498W26] or [CSE598W26] in the subject line
- Lecture attendance strongly encouraged. Participation is part of final grade

Course Materials

- Lecture notes: https://github.com/pag-crypto/EECS498598-W26-ZKP
- Piazza: https://piazza.com/umich/winter2026/eecs498017
- Autograder.io: https://autograder.io/web/course/365
- No required textbook, but you'll likely find the optional textbooks useful (see syllabus)

Lectures

- All lectures and discussions will be recorded and made available online.
- PDFs of slides with markups will also be available on the Github.
- With a few exceptions, the lectures will *not* directly teach you the specific algorithms or protocols you'll implement in the project: it will be up to you to study and understand them outside of class.

The course project

- Course is organized around a semester-long project
  - Implement a complete ZKP system* in Rust (given skeleton code and utils)
  - Five milestones, each builds on previous ones and will take 2-3 weeks
  - Work in teams of at most three
- For students more interested in theory, a “theory track” is available: written HWs instead of later projects
  - Students interested in research on ZKPs should do both tracks. (This will be intense but very rewarding.)
- **Completing the project will be intellectually demanding and time-consuming!**
  - Start the project the day it’s released. Try to work on it a bit every day.
- Programming projects will be graded via autograder.io. More details in P1 handout.
- Written HWs graded by hand

Course policy on LLMs:

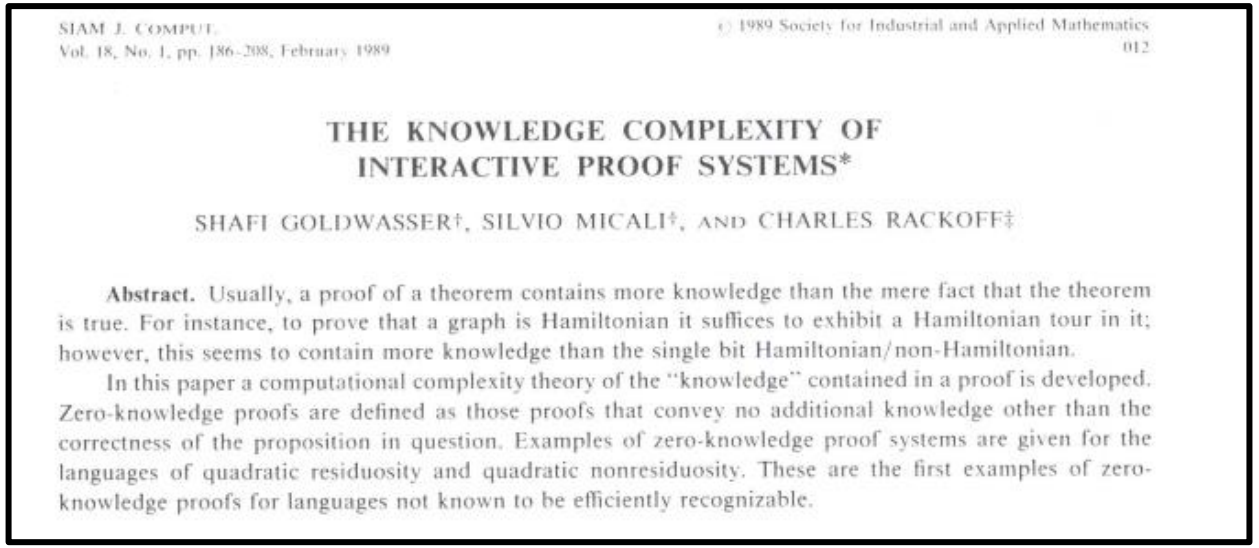
Use LLMs for whatever you want; however, if we ask you to explain a submitted project or homework solution, and you can’t give a satisfactory explanation, you will fail the project.

* You don’t need to know what exactly this means yet; it will be explained later in today’s lecture.

Grading

- Your final grade will have two* components:
  - 80% project and homework grades
  - 20% participation
  - * Some aspects of the course for those enrolled in 598 are still in flux, primarily to make sure everybody in 598 can get the right kind of credit. We will end early so I can discuss with 598 students who are in this position.
- All written assignments *must* be typeset in LaTeX.
- Collaboration and external sources are allowed for homeworks, with some caveats (see syllabus). Must list collaborators on HWs
- Lectures are not yet typeset. If you feel you need extra credit, you may be able to get it for writing good scribe notes for these lectures. Contact me for info.
- This is a brand-new class and will not be graded harshly. If you make an honest effort, you will probably get a good grade.

What is a zero-knowledge proof?



SIAM J. COMPUT.
Vol. 18, No. 1, pp. 186–208, February 1989

012

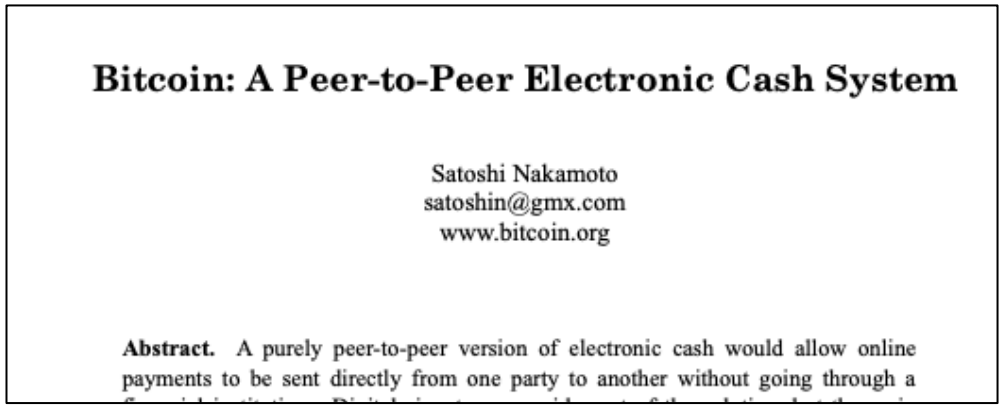
THE KNOWLEDGE COMPLEXITY OF INTERACTIVE PROOF SYSTEMS*

SHAFI GOLDWASSER†, SILVIO MICALI‡, AND CHARLES RACKOFF‡

Abstract. Usually, a proof of a theorem contains more knowledge than the mere fact that the theorem is true. For instance, to prove that a graph is Hamiltonian it suffices to exhibit a Hamiltonian tour in it; however, this seems to contain more knowledge than the single bit Hamiltonian/non-Hamiltonian.

In this paper a computational complexity theory of the "knowledge" contained in a proof is developed. Zero-knowledge proofs are defined as those proofs that convey no additional knowledge other than the correctness of the proposition in question. Examples of zero-knowledge proof systems are given for the languages of quadratic residuosity and quadratic nonresiduosity. These are the first examples of zero-knowledge proofs for languages not known to be efficiently recognizable.

Mostly theory until about ten years ago….

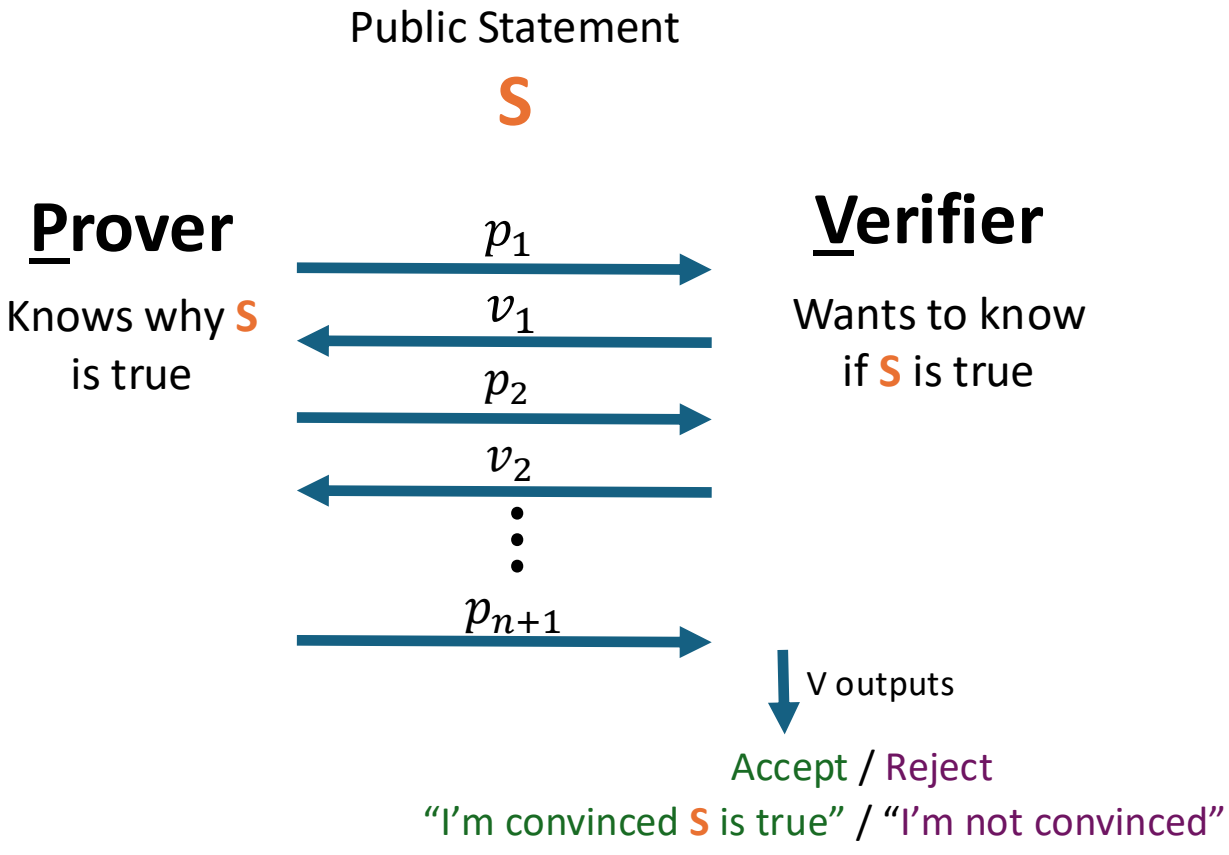


Bitcoin: A Peer-to-Peer Electronic Cash System

Satoshi Nakamoto
satoshin@gmx.com
www.bitcoin.org

Abstract. A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a

Motivating our topic of study



Security guarantees:

1. Soundness:
   - If **V** accepts, **S** is very likely to be true.
2. Zero-knowledge
   - The messages only reveal that **S** is true – not why. **P** discloses as little as possible.

ZKPs useful for balancing privacy and verifiability.
Many examples of tensions between these goals…

Can you think of some examples?

- private digital payments
- online age verification
- network management
- attested photo editing
- Spam/bot prevention
- Machine learning (lots)
- *And more…*

Motivating our topic of study

PRODUCTS › GOOGLE PAY

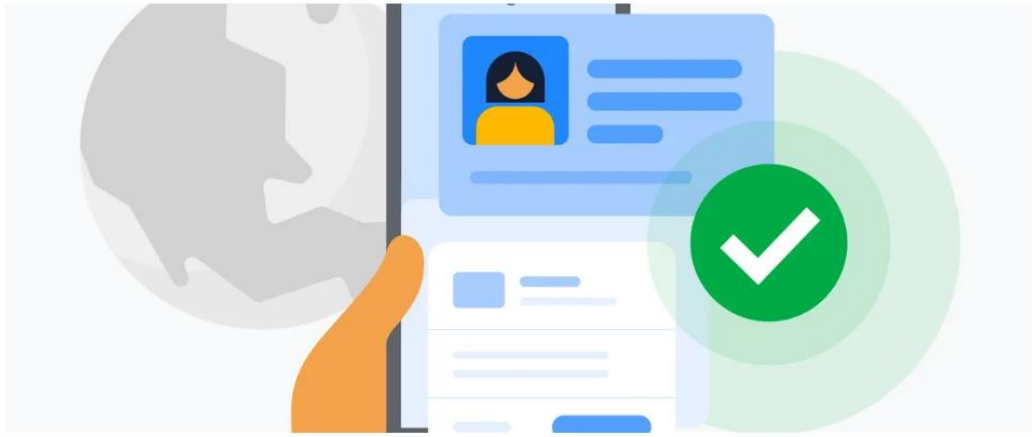
It's now easier to prove age and identity with Google Wallet

Apr 29, 2025
2 min read

ID passes are coming to the U.K. and we are expanding mobile IDs to more U.S. states with more places you can use them.

Alan Stapelberg
Group Product Manager, Google Wallet

Share



Listen to article 3 minutes

Today we're introducing updates that help you prove your age and identity in a safe and secure way, right from your phone.

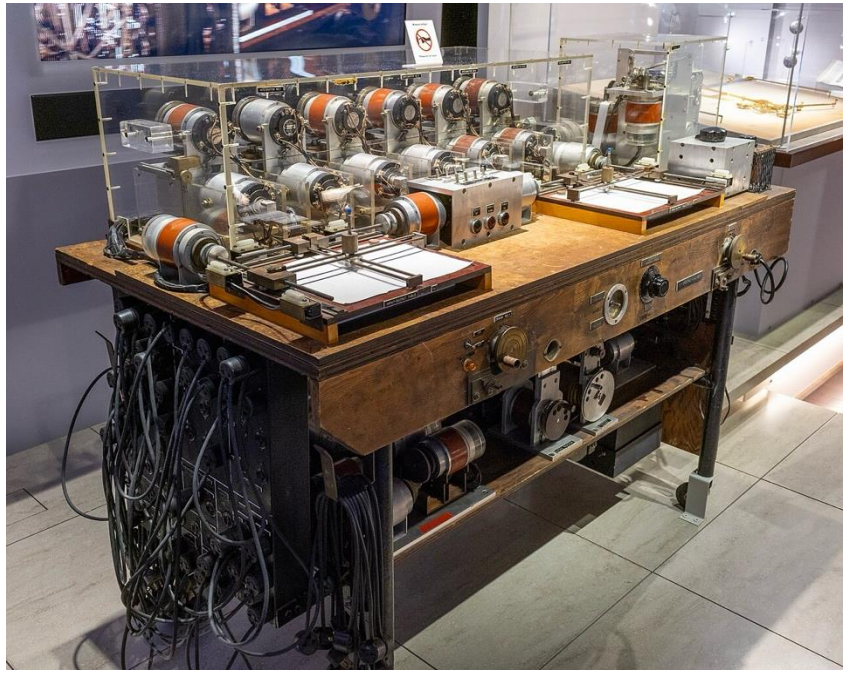
Fast and private age verification

Given many sites and services require age verification, we wanted to develop a system that not only verifies age, but does it in a way that protects your privacy. That's why we are integrating Zero Knowledge Proof (ZKP) technology into Google Wallet, further ensuring there is no way to link the age back to your identity. This implementation allows us to provide speedy age verification across a wide range of mobile devices, apps and websites that use our Digital Credential API.

Motivating our topic of study

- Course won't be an overview of all of ZKPs. Very "opinionated" focus on modern constructions that can prove all of NP with nearly practical efficiency.
- "Prove all of NP" => programmability. Turns hardware problems into software problems. Usually a very good trade; modern computer science is built on it.

A hardware solution for solving differential equations



Differential analyser built by Arnold Nordsieck, at the Computer History Museum
By The wub - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=158377009

A software solution

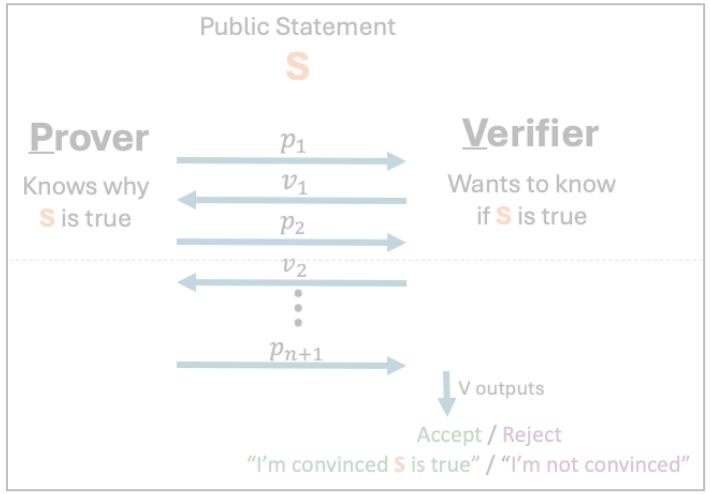
scipy.integrate.

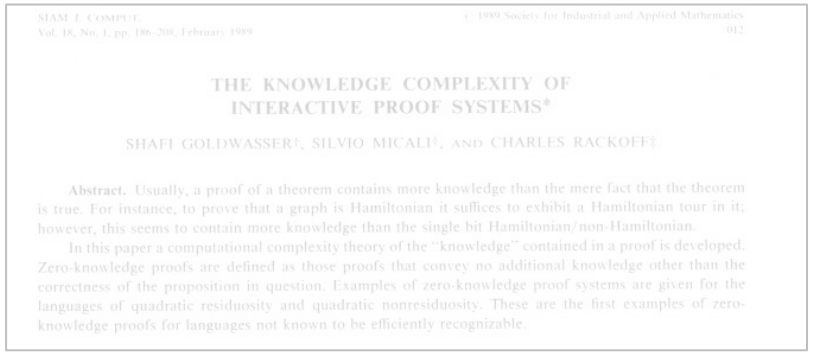
solve_ivp

```
solve_ivp(fun, t_span, y0, method='RK45', t_eval=None, dense_output=False,
events=None, vectorized=False, args=None, **options)                [source]
```

Solve an initial value problem for a system of ODEs.

Motivating our approach





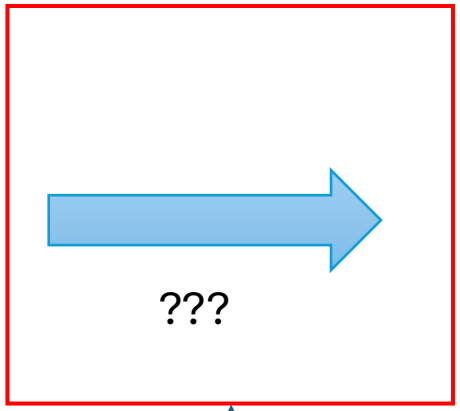
SIAM J. COMPUT.
Vol. 18, No. 1, pp. 186–208, February 1989
© 1989 Society for Industrial and Applied Mathematics
012

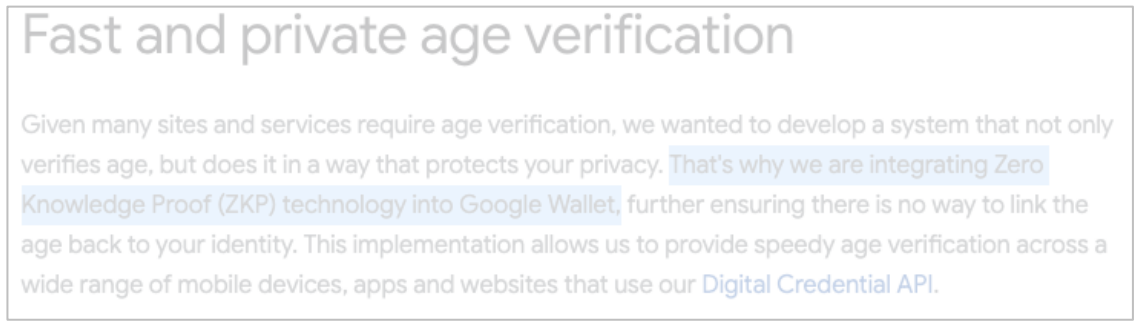
THE KNOWLEDGE COMPLEXITY OF INTERACTIVE PROOF SYSTEMS*

SHAFI GOLDWASSER†, SILVIO MICALI‡, AND CHARLES RACKOFF‡

Abstract. Usually, a proof of a theorem contains more knowledge than the mere fact that the theorem is true. For instance, to prove that a graph is Hamiltonian it suffices to exhibit a Hamiltonian tour in it; however, this seems to contain more knowledge than the single bit Hamiltonian/non-Hamiltonian.

In this paper a computational complexity theory of the "knowledge" contained in a proof is developed. Zero-knowledge proofs are defined as those proofs that convey no additional knowledge other than the correctness of the proposition in question. Examples of zero-knowledge proof systems are given for the languages of quadratic residuosity and quadratic nonresiduosity. These are the first examples of zero-knowledge proofs for languages not known to be efficiently recognizable.





Fast and private age verification

Given many sites and services require age verification, we wanted to develop a system that not only verifies age, but does it in a way that protects your privacy. That's why we are integrating Zero Knowledge Proof (ZKP) technology into Google Wallet, further ensuring there is no way to link the age back to your identity. This implementation allows us to provide speedy age verification across a wide range of mobile devices, apps and websites that use our Digital Credential API.

This class

Motivating our approach

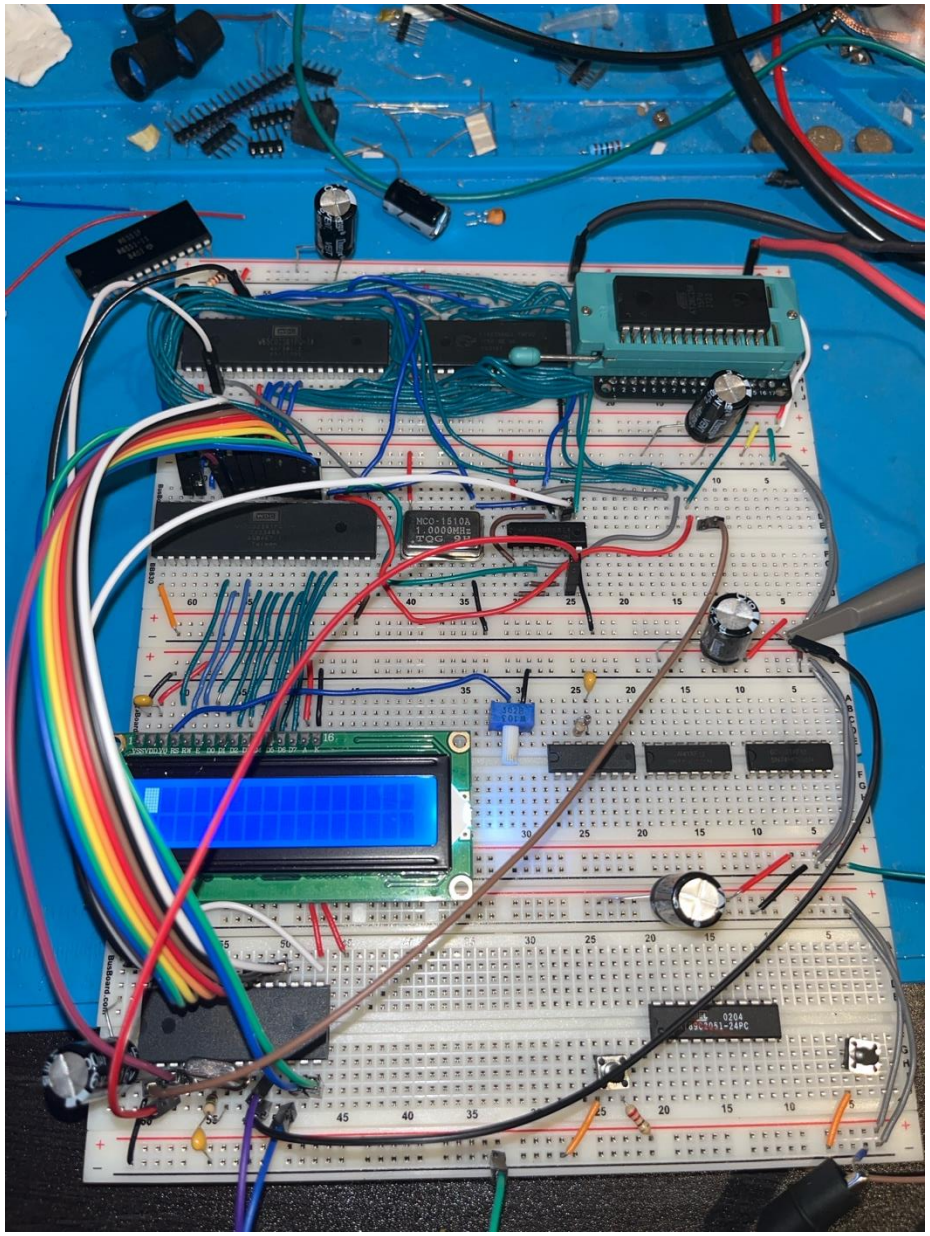
This class studies the entire ZKP stack. Three key pieces:

1. Frontend (transforms application-level statements into ZK-friendly representation)
2. Backend (the ZKP protocol implementation itself)
3. Common to both: library of mathematical primitives

Why does the class cover all of these pieces?
Why is it important for you to *build* all these pieces?

- Many people (including me) learn best by actually building things
- ZKPs as an area have *lots* of abstraction boundaries. Good for design but bad for engineering: gaps between abstractions cause performance and security problems. Building the whole stack gives cross-boundary understanding.
- Fixing performance bottlenecks requires working across boundaries
- Case study for key challenges of modern cryptography engineering: write *correct, fast,* and *safe* cryptography code, and integrate it correctly into broader system.

Motivating our approach



Tentative project schedule

Public google doc link:
https://docs.google.com/document/d/13Dvt-kfJ61lh1g24a4P4x3B58SkgeAAFDSfzclZqNGw/

- Project 1: library for mathematical primitives
  - Modular arithmetic, elliptic curves, polynomials
  - Available **today**, due **Jan. 28**
- Project 2: the Delphian argument of knowledge
  - First milestone for the course protocol
  - Available **Jan. 28**, due **Feb. 11**
- Project 3: transforming Delphian into a zkSNARK*
  - Implementing Fiat-Shamir, making Delphian zero-knowledge
  - Available **Feb. 11**, due **Feb. 25**
- Project 4: the frontend
  - System for representing computations as constraints
  - Available **Mar. 9**, due **Mar. 23**
- Project 5: a simple application
  - Use the result of P4 to implement a hash-preimage proof
  - Available **Mar. 23**, due **Apr. 6**

* Ripping a bandaid off here: what you end up with won't be a "true" zkSNARK according to some stricter definitions of the term. Turning the P3 deliverable into a "true" zkSNARK is possible and not terribly difficult, but ultimately not worth the time investment for us.

Tentative lecture topics

Public google doc link:
https://docs.google.com/document/d/13Dvt-kfJ61lh1g24a4P4x3B58SkgeAAFDSfzclZqNGw/

- Unit 1: foundations
  - Mathematical tools used in ZKPs. (More emphasis on utility than comprehensiveness.)
  - Groups, rings, fields, polynomials
- Unit 2: backends
  - How are modern zero-knowledge proofs built?
  - Polynomial commitments, the Fiat-Shamir transform, interactive oracle proofs
- Unit 3: frontends, or “ZK programming”
  - How do we transform a computation into a ZK-friendly representation?
  - Random-access memory, proving CPU executions in ZK
- Unit 4: advanced topics and applications
  - Incrementally verifiable computation, lattice-based ZKPs

Other questions

What questions do you have about the class?

Conclusion

(If you’re in 598, hang back so we can talk about course requirements…)