30 years of Zero Knowledge Proofs

Muthuramakrishnan
Venkitasubramaniam

UNIVERSITY of ROCHESTER
An example

Millennium Problems

Yang–Mills and Mass Gap
Experiment and computer simulations suggest the existence of a "mass gap" in the solution to the quantum versions of the Yang–Mills equations. But no proof of this property is known.

Riemann Hypothesis
The prime number theorem determines the average distribution of the primes. The Riemann hypothesis tells us about the deviation from the average. Formulated in Riemann's 1859 paper, it asserts that all the 'non-obvious' zeros of the zeta function are complex numbers with real part 1/2.

P vs NP Problem
If it is easy to check that a solution to a problem is correct, is it also easy to solve the problem? This is the essence of the P vs NP question. Typical of the NP problems is that of the Hamiltonian Path Problem: given N cities to visit, how can one do this without visiting a city twice? If you give me a solution, I can easily check that it is correct. But I cannot so easily find a solution.
An example

S

I (re)solved “P vs NP?”

How?

Here is the proof $\pi$

Can I convince someone the validity of something without revealing the proof?

Can I reveal “zero-knowledge” about a proof?

Clay Institute

Million $\$
Proof Systems
Proof systems

$L = \{(A, 1^k) : A \text{ is a true mathematical assertion of proof length } k\}$

What is a “proof”?

Insight: meaningless unless can be efficiently verified
Proof systems

Given language L, goal is to prove \( x \in L \)

Proof system for L is a verification algorithm V

– Completeness: \( \forall x \in L, \exists \Pi, V \text{ accepts } (x, \Pi) \)
  “true assertions have proofs”

– Soundness: \( \forall x \notin L, \forall \Pi^*, V \text{ rejects } (x, \Pi^*) \)
  “false assertions have no proofs”

– Efficiency: V runs in polynomial time in |x|
Classical Proofs (a.k.a NP)

Previous definition: “classical” proof system

\[ L \in \text{NP} \text{ iff expressible as} \]
\[ L = \{ x | \exists y \text{ s.t. } |y| < |x|^k \text{ and } (x, y) \in R \} \]

where \( R \) is polynomial time computable

\( \text{NP} \) is the set of languages with classical proof systems
Interactive Proofs [GMR85]
Interactive Proofs [GMR85]

• Two new ingredients:
  – **Randomness**: verifier tosses coins, errs with some small probability
  – **Interaction**: rather than “reading” proof, verifier interacts with prover

• Classical proof systems lie in this framework: prover sends proof, verifier does not use randomness
Interactive Proofs [GMR85]

Interactive proof system for $L$ is an interactive protocol $(P, V)$

- **completeness**: $x \in L$
  \[ \Pr[V \text{ accepts in } (P, V)(x)] = 1 \]

- **soundness**: $x \notin L, \forall P^*$
  \[ \Pr[V \text{ accepts in } (P^*, V)(x)] \leq 1/2 \]

- **efficiency**: $V$ is p.p.t. machine

**Repetition**: can reduce error to any $\varepsilon$

**Interactive Arguments**: Soundness only against PPT machines
Interactive Proof for Graph Isomorphism

Graph $G_0 = (V_0, E_0)$

$V_0 = \{1, 2, \ldots, 8\}$

$E_0 = \{(1, 2), (1.4), \ldots\}$

Graph $G_1 = (V_1, E_1)$

$V_1 = \{a, b, \ldots, j\}$

$E_1 = \{(a, g), (a, h), \ldots\}$

Isomorphic: Exists a mapping $\phi : V_0 \to V_1$ such that

$(\alpha, \beta) \in E_0 \iff (\phi(\alpha), \phi(\beta)) \in E_1$
Interactive Proof for Graph Isomorphism

\[ L = \{ (G_0, G_1) \mid G_0 \cong G_1 \} \]

Prover
Alice

Verifier
Bob

Accept if \( \rho_b(G_b) = H \)

\( G_0 \not\cong G_1 \)

\( b \in [0,1] \)

\( H \)

\( \phi \)

\( \rho_0 \)

\( \rho_1 \)
Zero Knowledge
Interactive Proofs
What is Knowledge?

Question as old as Humanity

Mostly studied in Philosophy: Epistemology
(also psychology, neuroscience, economics…)

Today, important in Computer Science
A Computational Approach to Knowledge [Goldwasser Micali 84]

2012 Turing Award Winners

“…for transformative work that laid the complexity-theoretic foundations for the science of cryptography, and in the process pioneered new methods for efficient verification of mathematical proofs in complexity theory”
A Computational Approach to Knowledge [Goldwasser Micali 84]

First in [GM84]: Probabilistic Encryption

Mature in [GMR85]: Zero-Knowledge + Proofs of knowledge

“I only know what I can feasibly compute”

Feasibly compute = \text{PPT}

\text{Probabilistic Polynomial Time Turing Machines}
Zero-Knowledge Proofs [GMR]

Completeness: P can convince V if X is true
Soundness: no (efficient) P* can convince V if X is not true
Zero Knowledge: no efficient V* learns anything more than validity of X

Thank you Alice, I believe X is true. But I don’t know why!
ZK Proof for Graph Isomorphism

Prover
Alice

Verifier
Bob

\[ G_0 \cong G_1 \]

\[ b \in [0,1] \]

Darn! I did not learn a thing
ZK Definition

∀ PPT adversary verifier \( V^* \), \( \exists \) PPT simulator \( S \) such that

\[ S\text{-views} \approx V^*\text{-views} \]

with Prover
ZK Definition

\[ \forall \text{PPT adversary verifier } V^*, \ \exists \text{PPT simulator } S \text{ such that } \frac{S\text{-views}}{V^*\text{-views}} \approx \text{with Prover} \]

ZK Rationale

\[ V^* \text{ learns nothing that cannot be generated by } V^* \text{ itself} \]

\[ V^* \text{ itself } = \text{All Prob. Poly Time} \]
ZK Definition

\[ \forall \text{PPT adversary verifier } V^*, \ \exists \text{PPT simulator } S \text{ such that } \]

\[ S\text{-views} \approx V^*\text{-views} \text{ with Prover} \]
ZK as an instance* of MPC

NP language $L$ with relation $R$

Securely Compute

$f(x, w) = R(x, w)$
ZK Proof for Graph Isomorphism

\[ G_0 \approx G_1 \]

1. Choose \( G_0 \) or \( G_1 \) at random
ZK Proof for Graph Isomorphism

\[ G_0 \approx G_1 \]

1. Choose \( G_0 \) or \( G_1 \) at random
2. Simulator will succeed w.p. \( \frac{1}{2} \)
What can you prove in ZK?

Can prove any classical proof in ZK [GMW86] (a.k.a NP statements)

“Everything provable is provable in ZK” [BGGHKMR90] (a.k.a languages in IP)

IP = PSPACE [S90,LFKN90]
PSPACE contains every language that is solvable with polynomial space
ZK for all of NP

**Step 1:** Construct a ZK Proof for an NP-complete language $L_C$

**Step 2:** Given any NP lang. $L$ and instance $x$, compile* instance $x$ to an instance $x_C$ for $L_C$ and use ZK Proof for $x_C \in L_C$

* compile via Karp reduction

Need Cryptographic Commitments
Commitment Scheme

The “digital analogue” of sealed envelopes.

Sender  Receiver

Commitment phase: Com(v)

Decommitment phase: d

Hiding: The commitment hides the committed value

Binding: The commitment can only open to one value
**Graph 3COL**

**ZERO KNOWLEDGE FOR ALL OF NP**

Prover

\[ x = G(V,E) \]

\[ w = c : V \rightarrow \{1,2,3\} \]

Completeness: Valid 3-Coloring satisfies \( c(i) \neq c(j) \) for every edge \( e(i,j) \)

Soundness: Com() is binding \( \Rightarrow \) prover cannot change colors later

If \( G \) is not 3 colorable, prover caught on at least one edge. Occurs w.p. \( 1/|E| \)

Zero Knowledge: Guess edge \( e(i,j) \) and give different colors for \( c(i) \) and \( c(j) \)

Verifier

\[ x = G(V,E) \]

Accept iff \( c(i) \neq c(j) \)
Constant s-soundness to negligible soundness

Repeat $k \log(1/s)$ times

Prover 1 Caught w.p. s

2 Caught w.p. s

\vdots

$k \log(1/s)$ Caught w.p. s

Each rep. is indep. and soundness is $s^{k \log(1/s)} = 2^{-k}$
What about ZK property?

Repeat $k \log(1/s)$ times

Caught w.p. $s$

Caught w.p. $s$

Caught w.p. $s$

$\cdots$

Each rep. is indep. and soundness is $s^{k \log(1/s)} = 2^{-k}$
Can we repeat it in parallel?

Verifier

Prover

Caught w.p. s

Caught w.p. s

Caught w.p. s

Each rep. is indep. and soundness is $s^{\log(1/s)} = 2^{-k}$
Can we repeat it in parallel?

Simulator’s guess for all rep. are correct simultaneously only with probability $2^{-k}$

Expected number of rewidings is $2^k$
ZK for NP

ZK proof for Graph 3 Coloring [GMW86]
ZK proof for Hamiltonicity [Blum86]
ZK proof for SAT [BC87]

**Theorem [BG+90]:** Assume the existence of one-way functions. There exists a ZK proof for all of IP

ZK proof for any NP relation without using Karp reductions [IKOS07]

...more on Wednesday
Numerous Applications

• Boosting passive to active security
• Identification/ Authentication
• CCA secure encryption
• Resettable Security
• Bitcoins
Main Application: Active secure MPC
Compiling passive to active security when majority are dishonest

Passive adversaries (a.k.a. honest-but-curious) follow protocol instructions to-the-word

Passive-secure MPC protocol

Coin Tossing  Zero Knowledge

Active-secure MPC protocol

Active adversaries (a.k.a malicious) arbitrarily deviate from protocol
Passive $\rightarrow$ Active: Enforce honest behavior

1. Force adversary to use a **fixed** input

2. Force adversary to use a **uniform** random tape

3. Force adversary to **follow protocol instructions** exactly
Coin Tossing
Goal: Fix random tape of every party

Com(r₁)

Output: r₁ ⊕ r₂

Open r₁

r₂

Output: r₁ ⊕ r₂
Augmented Coin Tossing: Fix Alice’s tape

Goal: Alice’s random tape is uniform. Bob receives commitment to tape.

\[
\text{Com}(r_1) \quad \rightarrow \quad r_2 \\
\quad \rightarrow \quad \text{Open } r_1
\]

Random tape = \(r_1 \oplus r_2\)

Commitment to coin toss = \(\text{Com}(r_1), r_2\)

Output: \(r_1 \oplus r_2\)
Forcing good behavior

Preamble Phase:

\[
\text{Com}(x), \text{Com}(r_1, A) \quad \text{Com}(y), \text{Com}(r_1, B)
\]

\[
r_2, A \quad r_2, B
\]

\[
\text{Open } r_1, A \quad \text{Open } r_1, B
\]

After this stage, each party holds a commitment to the other party’s input and random tape.

**Main Insight:** A protocol is a deterministic function of a party’s input, random tape and series of incoming messages.
Forcing good behavior

Preamble Phase:

Execute passive protocol
Prove correctness of message every step
Forcing good behavior

Preamble Phase:

"Correct": According to protocol specifications with input $x$ and random tape $r_{1,A} \oplus r_{2,A}$

Expressible as an NP statement

Statement: Transcript
Witness: $x$, $r_{1,A}$ and random for $\text{Com}(x), \text{Com}(r_{1,A})$

Polytime Relation:
1. Check commitments correct w.r.t $x$, $r_{1,A}$
2. Check all messages generated according to honest Alice algorithm with input $x$ and random tape $r_{1,A} \oplus r_{2,A}$

Caveat: Should not reveal witness!

Use ZK

NxtMsg$_i$
Final Compilation (a.k.a. GMW Paradigm)

Commit inputs and gen. rand tape

X

\[ a_1 \]

\[ ZK \text{ Proof that } a_1 \text{ is correct} \]

\[ b_1 \]

\[ ZK \text{ Proof that } b_1 \text{ is correct} \]

Y

Execute passive secure protocol and give ZK Proof every step
State-of-the-art for Active MPC

In theory, ZK Proofs allows compilation of passive to active security

In practice, use other techniques, eg, (cut-and-choose, MPC-in-the-head)

In fact, these other techniques have ZK implicit
Concurrency

Standard ZK is not secure in a concurrent setting
Zero Knowledge Proofs [GMR85]

Cornerstone of modern definitions of security

Techniques for arguing security

Fundamental cryptographic building block
Thank You