The Clean Paper

WHAT THE PAPER SAYS · WHAT IT DOESN'T · WHY IT MATTERS

A cryptographic proof can hide its secret by making the missing simulator hard to prove

Zero-knowledge proofs let someone prove a statement without revealing the witness. Classical theory says you cannot have that, in the full sense, with one message, no trusted setup and perfect soundness. Rahul Ilango's paper does not make that impossibility vanish. It changes the target: instead of requiring that a simulator really exists, it asks that no chosen proof system can efficiently prove that the simulator does not exist. Under major proof-complexity and cryptographic assumptions, that is enough to recover falsifiable, game-based consequences of zero-knowledge property by property. The point is not a plug-in internet primitive. It is a proof-theoretic way to turn mathematical unprovability into cryptographic cover.

Written by Cinzia Vaglio · Cross-checked by Vera Rubin · Edited by Michele Renda · Figures by Nova Vela · AI-assisted, human-reviewed

Paper: *Gödel in Cryptography: Effectively Zero-Knowledge Proofs for NP with No Interaction, No Setup, and Perfect Soundness*, Rahul Ilango, FOCS 2025 / IACR ePrint 2025/1296 · DOI: 10.1109/FOCS63196.2025.00058

The trick is not to prove the secret is hidden

Start with the simplest version of zero-knowledge.

Alice wants to convince Bob that a Sudoku puzzle has a solution. If she sends the solution, Bob is convinced, but the puzzle is ruined. What she wants is stranger: a proof that a solution exists, without revealing the solution.

That is the promise of a zero-knowledge proof. The prover (Alice) convinces the verifier (Bob) that a statement is true while revealing nothing beyond the truth of the statement.

The problem is that this promise costs something. An ordinary mathematical proof has two comfortable features. It is **one message**: you write it down, hand it over, and walk away. And it is **perfectly sound**: a false statement has no valid proof at all. Classical impossibility results say that zero-knowledge must give up both features — and not merely the two together; each one is off-limits on its own.

First, a zero-knowledge proof needs conversation. If Alice sends a single message, with no trusted setup arranged in advance, the zero-knowledge guarantee collapses — and this holds however much soundness you are willing to trade away in exchange.

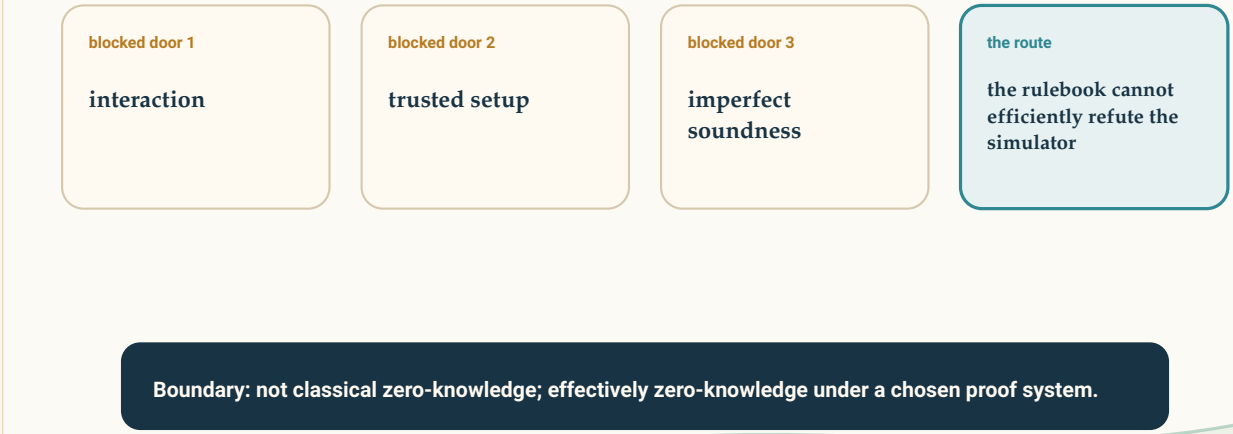
Second, a zero-knowledge proof needs a small tolerance for error. Demanding perfect soundness turns out to quietly destroy interaction too: a verifier that can never be fooled, no matter which random choices it makes, might as well fix those choices in advance — and once the verifier is predictable, Alice can answer everything in a single message, which is exactly the case that already broke.

Rahul Ilango's paper is about a way around that double wall. Not by pretending the wall is not there, and not by producing classical zero-knowledge in the impossible setting. The move is subtler: weaken what "reveals nothing" means, but weaken it in a way that preserves the security properties cryptographers can actually test.

The result is called **effectively zero-knowledge**.

A proof without leaking the witness

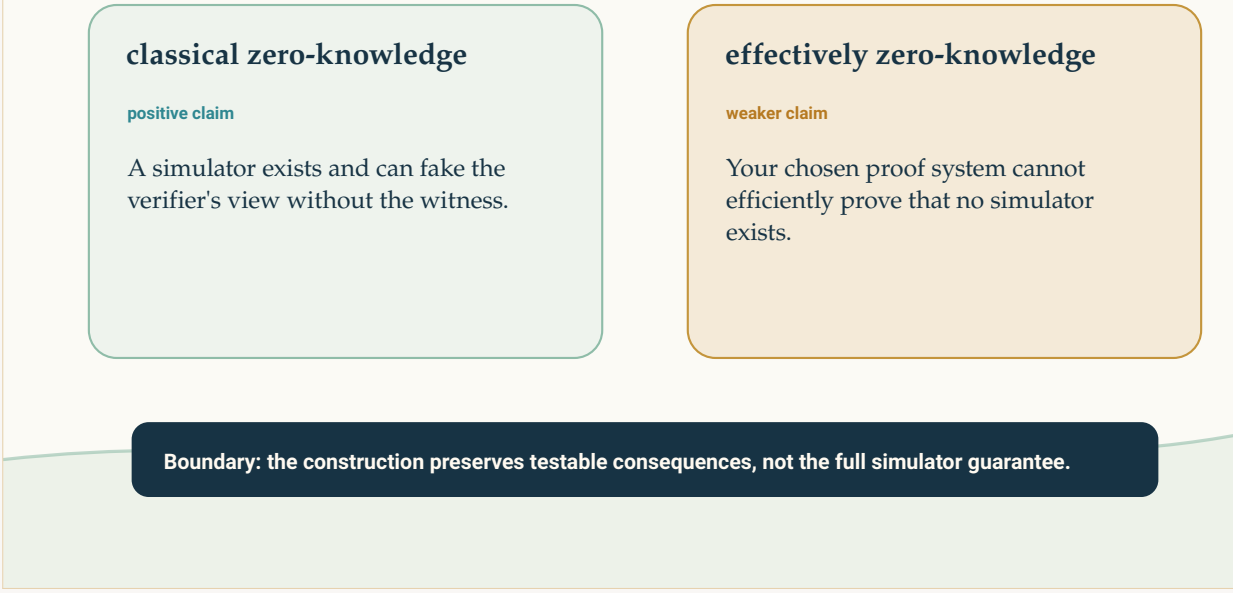
The paper keeps the ordinary-proof shape by weakening what privacy must prove.



Zero-knowledge is blocked at three doors — interaction, trusted setup, and imperfect soundness. Ilango's construction slips through a different one: the rulebook cannot efficiently refute the simulator.

Classical privacy vs effective privacy

The relaxation is the point: the paper changes what the privacy claim has to establish.



Classical zero-knowledge asks whether a simulator exists; "effectively zero-knowledge" asks only whether your chosen rulebook can efficiently prove one cannot. The weaker question is what lets the construction keep

one message, no setup and perfect soundness.

The old test: a simulator exists

The classical way to formalize zero-knowledge uses a fictional helper called a **simulator**.

The idea is this: imagine Jane, who does **not** know Alice's secret. If Jane can generate, entirely by herself, proofs that look just like the proofs Bob would have received from Alice, then Alice's proofs did not teach Bob anything new. Jane could already fake the experience without Alice's secret.

So classical zero-knowledge asks for an actual simulator. There must be an efficient algorithm that can produce fake-looking proofs without knowing the secret — the *witness*, in the jargon; for Sudoku, the witness is simply the solved grid.

That definition is powerful, but it is also exactly where the old impossibility bites. Here is the intuition. A truly non-interactive proof is just a string. Once Bob has that string, he can show it to someone else: he has gained the ability to prove the statement to others, which already sounds like more than "nothing." The classical theorems sharpen that intuition into the impossibilities above.

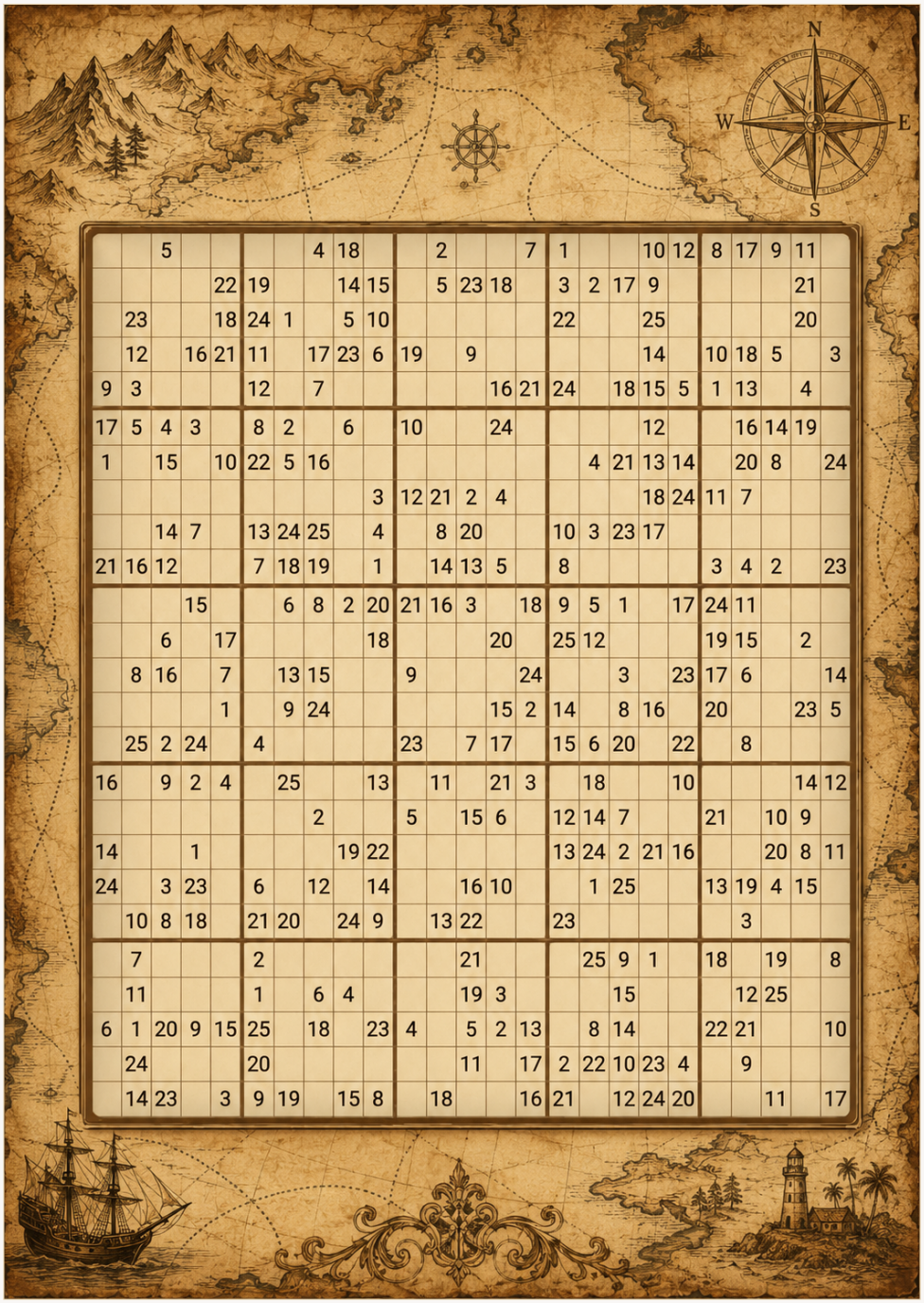
> **The three properties this paper insists on**
>
> The paper's title names three constraints:
> **No interaction:** Alice sends one proof string. There is no back-and-forth protocol.
> **No setup:** Alice and Bob do not rely on a trusted common reference string or other pre-arranged public randomness. Many systems called "non-interactive zero-knowledge" still rely on setup; this paper means zero setup.
> **Perfect soundness:** a false statement has no valid proof. Not "almost never accepted"; no valid proof exists.
> Those three properties are exactly what ordinary written mathematics has — and, as explained above, classical zero-knowledge cannot keep them.

A mega-Sudoku version of the difference

Here is a deliberately simplified way to feel the difference.

Do not use an ordinary 9-by-9 Sudoku for the serious part of the analogy. It is too small and too finite: a computer can just solve it, or prove that it has no solution. Instead imagine a family of **MegaSudoku(n)** puzzles. Scale the usual rule up: choose a block size n, let $N = n^2$, and build an N by N grid divided into n by n blocks, with N symbols. Ordinary Sudoku is just the tiny $n = 3$, $N = 9$ case: a 9-by-9 grid, 3-by-3 blocks and nine symbols. The proof-complexity story only begins when n is allowed to grow, and when the grid can carry extra gadgets that make it behave like a SAT formula

dressed as a Sudoku puzzle. A SAT formula is just a list of yes/no constraints: can you assign true/false values to the variables so that every constraint is satisfied?



A 25x25 Sudoku: its rules can be checked without revealing the finished grid — a visual stand-in for a proof that verifies a hidden solution, the witness.

Sudoku and SAT: the same puzzle in two costumes

The claim that a Sudoku can “behave like a SAT formula” is not a metaphor. The translation

runs in both directions, and the easy direction can be written down in full.

From Sudoku to SAT. SAT only speaks true/false, so give it one boolean variable per (row, column, value) triple: *x(r,c,v)* means "the cell in row *r*, column *c* contains the value *v*." A 4-by-4 Sudoku (2-by-2 blocks, values 1–4) needs 4·4·4 = 64 variables; the classical 9-by-9 needs 729. Every Sudoku rule then becomes a batch of clauses. (A clause is an OR of variables or their negations; the whole formula is the AND of all its clauses.)

Every cell holds at least one value — one clause per cell:

$$x(1,1,1) \vee x(1,1,2) \vee x(1,1,3) \vee x(1,1,4)$$

Every cell holds at most one value — a "not both" clause for each pair of values:

$$\neg x(1,1,1) \vee \neg x(1,1,2) \quad \neg x(1,1,1) \vee \neg x(1,1,3) \quad \text{... and so on for all six pairs.}$$

Every row contains every value — for row 1 and the value 3: at least once,

$$x(1,1,3) \vee x(1,2,3) \vee x(1,3,3) \vee x(1,4,3)$$

and at most once: $\neg x(1,1,3) \vee \neg x(1,2,3)$, and so on for each pair of cells in the row.

Columns and blocks — identical batches; only the group of cells changes. For the top-left block and the value 2:

$$x(1,1,2) \vee x(1,2,2) \vee x(2,1,2) \vee x(2,2,2)$$

plus the pairwise "not both" clauses.

The printed clues — the simplest part: each clue is a clause with a single variable. A printed 3 in the top-left corner becomes the clause

$$x(1,1,3)$$

The AND of all of this is satisfiable exactly when the Sudoku has a solution — and a satisfying assignment *is* the solution: read off which x(r,c,v) are true and fill in the grid. For a 9-by-9 this comes to 729 variables and a few thousand clauses, which a modern SAT solver dispatches in milliseconds. Notice the clue clause x(1,1,3): it says "this cell equals exactly 3," not "these cells are all different" — the same asymmetry that will force the extra trick for clue cells in the protocol note further down.

From SAT to Sudoku. The paper needs the opposite, harder direction: given an *arbitrary* SAT formula, build a mega-Sudoku that has a solution exactly when the formula does. Sudoku's native rules can only say "these cells are all different," so arbitrary logical constraints have to be *built* — and this is precisely what the gadgets are. A gadget is a small pre-fabricated cluster of cells, one per clause of the formula, in which designated cells play the role of variables (the symbol they hold encodes true or false) and the cluster's internal constraints are engineered so that its only legal fillings correspond to assignments satisfying that clause. This is standard craftsmanship from NP-completeness proofs; for generalized Sudoku it was carried out by Yato and Seta in 2003.

Together, the two directions say that N-by-N Sudoku and SAT are the same problem wearing different costumes. That is what licenses this article — and the paper — to tell a story about all of NP using grids and symbols.

The witness is still easy to picture. Alice knows a complete valid filling of the mega-Sudoku. Bob wants to be convinced that such a filling exists, but Alice does not want to reveal it. If she sends the whole filling, Bob is convinced, but the secret is gone.

In the classical zero-knowledge version, Alice and Bob interact. One old-style mental model uses covered tiles. Alice hides the solved grid, secretly renames the symbols before each round, and lets Bob inspect one randomly chosen local constraint: a row, a column, a box, or a gadget. If the opened cells show all-different symbols, Bob gains confidence. Then everything is covered again and the symbols are freshly renamed. (One wrinkle: the given clues of the puzzle need an extra trick, because renaming the symbols hides them too. The note below explains how the classical protocols solve this; the toy picture is enough for what follows.)

How the classical protocols really handle the clue cells

The renaming trick has a blind spot. The row, column and box rules all say "these cells are all different," and *all different* survives any renaming of the symbols. But a clue says "this cell contains exactly 5," and after renaming Bob only sees $\sigma(5)$ — some masked symbol — without knowing the renaming σ. He cannot check anything. Left unfixed, Alice could prove that *some* valid grid exists while ignoring the printed clues entirely, which proves nothing about *this* puzzle. The classical literature has two standard repairs.

The palette. Add one extra row of N cells to the hidden grid — a palette that Alice fills with the symbols 1…N in a fixed public order, and then renames along with everything else, so it contains $\sigma(1)$…$\sigma(N)$. Bob's random challenge now has one extra option. Besides picking a row, column, box or gadget to open, he may pick **the palette plus one clue cell**. Alice uncovers both; the palette reveals that round's renaming, and Bob checks that the clue cell shows exactly the renamed version of the printed clue. This stays zero-knowledge because Bob learns only σ — which is freshly drawn every round and worthless on its own — and the value of a cell he already knew from the puzzle. Nothing about the secret cells leaks, and a simulator can fake the view by drawing a random σ. It is sound because a cheating Alice is caught with fixed probability per round, and rounds are repeated until the doubt is negligible.

Compiling the clues away. A more structural variant removes the special challenge instead of adding it. Rather than *verifying* the clue value, force it with difference constraints: link the clue cell to every palette cell except the one carrying its own value — "different from $\sigma(1)$, different from $\sigma(2)$, …, different from everything but $\sigma(5)$." The only symbol the cell can legally hold is the clue's. Every constraint is now of the "these two differ" kind again — invariant under renaming, checkable exactly like a row. This is the same manoeuvre used for pre-colored vertices in the classical graph-coloring protocol, and it is the spirit of the word *gadgets* above: in the MegaSudoku-as-SAT picture, the clues are compiled into inequality gadgets like every other constraint.

The physical protocol. The real-world card protocol for Sudoku (Gradwohl, Naor, Pinkas and Rothblum, 2007) uses no renaming at all and settles the clues before the hiding even starts. For every cell, Alice lays down three identical cards with the cell's value — face-down for secret cells, but **face-up for clue cells**, so Bob sees with his own eyes that the clues are respected before the cards are flipped. Then one card from each cell goes into its row's packet, one into its column's, one into its box's; each packet is shuffled and revealed, and Bob checks it contains

> all N symbols. The shuffling destroys the position information (that is the zero-knowledge), but the clues were already nailed down at dealing time.
> Either way, the lesson is the same one this article keeps returning to: a zero-knowledge protocol is a careful bookkeeping of *which* facts survive the hiding. Renaming preserves "all different" and erases "equals 5" — so "equals 5" must be smuggled back in by other means.

That is not the protocol in the paper. It is the mental model for classical zero-knowledge:

- Alice and Bob go back and forth.
- Bob chooses random checks.
- Alice reveals only local consistency, not the whole solution.
- The proof of privacy works by showing that Bob's view could have been generated without Alice's secret solution.

So classical zero-knowledge is built around a positive fact:

> A simulator really exists.

Now remove the comfortable parts. Alice sends one proof string and walks away. There is no trusted setup, no shared random string prepared in advance, and Bob must never accept a false puzzle. That is the setting classical zero-knowledge cannot survive.

One more character is needed before the trick. Fix a **rulebook**: a formal proof system, in the logician's sense — a fixed set of axioms plus mechanical rules for checking written mathematical proofs. ZFC, the standard axioms of mathematics, is the canonical example. Everything from here on is stated relative to a rulebook chosen in advance, and the choice is flexible: the construction works for any rulebook you fix, ZFC included.

(A note on words, borrowed from the paper itself: "proof system" here always means this rulebook — the formal system that checks mathematical proofs — never the messages Alice sends. Alice's and Bob's machinery is called "the prover and the verifier.")

The Gödel-style version keeps the mega-Sudoku story but changes the proof.

Pick a second constraint system of the same displayed size, call it **D**. For the story, S and D are two MegaSudoku(n) puzzles in the same format. Behind the scenes, D may have started as a hard logical formula of a different size; if needed, it can be padded with harmless dummy constraints so it fits the same grid. D is built from a logical formula that is actually **unsatisfiable**: there is no possible assignment of values that makes all its constraints true, just as a broken puzzle has no legal completed grid. A toy example would be a formula that demands both "X is true" and "X is false." So D has no valid filling.

But D must not be a broken puzzle that is *easy to expose*. The toy example above fails this: any rulebook refutes "X and not-X" in one line. D has to be false in a way the chosen rulebook cannot certify with a short argument. If the rulebook could refute D with a short proof, the story below would collapse:

the alternative route that might have produced proofs without Alice's secret could be formally ruled out, and with it the privacy guarantee. So D is chosen from a family that the fixed rulebook cannot efficiently refute: there is no short proof, inside that rulebook, that D has no solution.

Alice's one-message proof is then about an either/or statement:

> either the real mega-Sudoku S has a solution, or the decoy D has a solution.

This is the logical link. D is **not** generated in some magical way that makes S true. The proof is not arguing "D has no solution, therefore S has a solution." It is proving the disjunction **S or D**. Perfect soundness says a false disjunction cannot have a valid proof. Since D is false in reality — it has no solution — the only way the disjunction can be true is for S to be true. So if the proof is accepted, S must have a solution. The decoy cannot make a false S become true.

But for the zero-knowledge-style part, ask what would happen if D *did* have a solution. That decoy solution would act as an alternative witness. It would let someone produce proofs without knowing Alice's real mega-Sudoku solution — a simulator, in other words. In reality D has no solution, so this simulator route is closed. The point is that the rulebook cannot efficiently prove it is closed.

So D has two jobs. For **soundness**, D is false, so a valid proof of "S or D" forces S. For **effective zero-knowledge**, D is hard to refute, so the rulebook cannot quickly rule out the decoy route that would have made simulation possible.

So the security test is no longer:

> Can we prove that a simulator really exists?

It becomes:

> Can your rulebook efficiently prove that the simulator is impossible?

If the answer is no, something surprisingly strong follows: every security guarantee that (a) can be observed by running a test, and (b) provably follows — inside that rulebook — from the existence of a simulator, actually holds. A successful attack on any of them would itself amount to the missing short refutation, and the missing short refutation does not exist. That is the "effective" part of effectively zero-knowledge.

So the classroom contrast is:

Classical zero-knowledge: the proofs are safe because a simulator exists.

Gödel-style effective zero-knowledge: the proofs are treated as safe for observable security tests because the rulebook cannot efficiently prove that the simulator is impossible.

The second claim is weaker. It is also why the paper can keep the three features that broke the classical version: one message, no setup and perfect soundness.

The new test: you cannot prove the simulator is absent

Ilango's relaxation changes the question.

Classical zero-knowledge asks:

> Does a simulator exist?

Effectively zero-knowledge asks something weaker:

> Can your chosen rulebook efficiently prove that no simulator exists?

That sounds like a technical dodge, but it is the core idea. The construction lives in a strange state: a simulator does not actually exist — the paper is explicit about this — but the rulebook you fixed cannot efficiently prove that it doesn't. If every bad consequence you care about would require such a refutation, the system still behaves like zero-knowledge for those consequences.

This is where Gödel enters. Not as decoration, and not as "Gödel makes crypto secure." The connection is proof-theoretic. A rulebook is called **optimal** if it is, in a precise sense, the best possible one: whenever any rulebook can refute a formula of the relevant kind with a short proof, the optimal rulebook can too, with a proof at most polynomially longer. Krajíček and Pudlák conjectured in 1989 that **no optimal proof system exists**: whichever rulebook you fix, some other rulebook proves some family of true statements far more succinctly. This is one of the central open conjectures of proof complexity, and it is the finite, complexity-theoretic cousin of Gödel's incompleteness theorem: some true statements have no short proof in the rulebook you fixed — not because they are unprovable in principle, but because every fixed rulebook leaves some short truths without short proofs.

The paper assumes this conjecture (in a mildly stronger "infinitely often" form, standard when conjectures are used cryptographically). The payoff, by a theorem of Krajíček and Pudlák, is concrete: for every rulebook there is a sequence of formulas that are genuinely unsatisfiable, that the rulebook cannot refute with short proofs — and, crucially, that an efficient algorithm can *generate*. That last property, uniformity, is what turns the whole idea from an existence claim into an actual algorithm Alice can run: her decoys D come off an assembly line, not out of thin air.

The cryptographic move is to put that shortage of proof power to work.

What the construction is doing

Here is the paper's construction, stripped to its shape.

Fix a rulebook — ZFC, say. Under the proof-complexity assumption, there is an efficiently generatable sequence of formulas that are actually unsatisfiable, but the rulebook has no short proof that they are unsatisfiable.

Now build a one-message proof of this form:

> either the real statement is satisfiable, or this special hard formula is satisfiable.

The special hard formula is not satisfiable. So if the underlying proof machinery is perfectly sound, accepting the message still means the real statement is true. That gives perfect soundness.

But for the zero-knowledge-like security, imagine the special hard formula *were* satisfiable. Then its witness could be used to simulate proofs without knowing the real witness. The formula is not satisfiable in reality — but the rulebook cannot efficiently prove that. So it cannot efficiently prove that the simulator is impossible.

That is the hinge. The system does not hide the secret by producing a classical simulator. It hides the secret, for a large class of observable security tests, behind the rulebook's inability to certify that the simulator is absent.

What the paper claims

The main theorem comes in layers. The core result is this:

Under a standard cryptographic assumption — the existence of **non-interactive witness indistinguishable proofs**, well-studied objects that follow from several established assumption packages — and under the proof-complexity conjecture that **no (infinitely often) optimal proof system exists**, the paper constructs, for every choice of rulebook, a one-message prover and verifier for NP/SAT with **perfect soundness** and no setup that is effectively zero-knowledge relative to that rulebook. (NP/SAT is the standard "hardest common denominator" of puzzle-like problems; mega-Sudoku is one costume it wears.)

For the broader claim about preserving falsifiable security properties, the paper adds one more standard assumption, the derandomization belief **P = BPP** (roughly: randomness gives algorithms no essential extra power).

Translated out of theorem language:

- The proof is one message.
- There is no trusted setup.
- False statements cannot be proved.
- The prover is not classical zero-knowledge — it has no simulator.
- But every falsifiable, game-based security consequence of classical zero-knowledge can be achieved in this setting.

"Falsifiable" matters. It means a security failure can be tested by running an adversary in a game. Many cryptographic security definitions have this form: can the adversary distinguish two encryptions, invert a function, recover a witness, or win some specified experiment? The theorem gives

a prover for each falsifiable property, one at a time. A single prover enjoying *every* falsifiable property at once is likely impossible — the old reusability attack ("Bob can show the proof to others") is itself a falsifiable property, and it genuinely fails here. The paper's proposal is that a single prover can plausibly cover all *natural* falsifiable properties — the ones that actually occur in cryptographic practice — but that part is a conditional theorem resting on an informal notion of "natural," plus an explicit conjecture. The guarantee is aimed at observable failures, not at every philosophical or simulation-based meaning of secrecy.

One concrete corollary is worth naming: the construction yields the first non-interactive *witness hiding* proofs with a uniform prover — "a proof of a puzzle does not help you find its solution," with no interaction and no setup — a modest-sounding object that had resisted construction for decades.

What this does not say

This is the section that keeps the piece honest.

It does **not** say the old impossibility theorems were wrong. The construction avoids them by changing the definition.

It does **not** give ordinary, classical zero-knowledge with no interaction, no setup and perfect soundness. The paper explicitly says the constructed prover does not have a simulator.

It does **not** mean the proof cannot be reused. A one-message proof can still be shown to someone else; the paper does not preserve deniability-style properties. (Non-interactive zero-knowledge with trusted setup has the same limitation.)

It does **not** mean this is a practical protocol ready for deployment. This is complexity theory and cryptographic foundations. The result depends on major assumptions from proof complexity and cryptography, and the construction is about what is possible in principle.

It does **not** make "Gödel" a magic security primitive. The Gödel connection is through proof systems, optimal proof systems and finite analogues of incompleteness. The usable intuition is not "incompleteness protects your password." It is: if a rulebook cannot efficiently prove that a simulator is impossible, then attacks that would require that proof can be blocked at the level of security definitions.

Why it is interesting anyway

Cryptography often turns hardness into safety. Factoring is hard, so RSA-style assumptions become useful. Lattice problems are hard, so lattice cryptography becomes useful. Here the hardness is stranger: not "hard to compute a secret," but "hard to prove that a certain proof object cannot exist."

That is why the paper feels unusual. It treats axioms and rulebooks almost like cryptographic re-

sources. The usual impossibility says there is a tension between soundness and simulation. Ilango's move is to place the tension behind a proof-theoretic curtain: the simulator is absent, but the formal system cannot efficiently expose that absence.

For a reader, the surprising part is not that this will replace today's zero-knowledge systems. It probably will not, at least not directly. The surprising part is that a limitation from mathematical logic can be used constructively: not just as a wall, but as a kind of cover.

How strong is the evidence?

This is a theorem paper, so "evidence" means something different from a biology or astronomy paper. The question is not whether an experiment replicated. The question is whether the definitions, assumptions and proof chain support the claim.

The proof is formal, and the paper is explicit about its assumptions. The assumptions are not casual. Non-interactive witness indistinguishable proofs are standard objects in cryptography and follow from several established assumption packages. The no-optimal-proof-system conjecture is a central conjecture in proof complexity. P = BPP is a standard derandomization belief used only for the broader falsifiable-property theorem.

The paper also argues the assumptions are the right price, not an arbitrary scaffold: it proves a converse showing they are essentially necessary — if constructions like this exist at all, then non-interactive witness indistinguishable proofs must exist, and (granting standard one-way functions) no optimal proof system can exist. And the assumptions are "win-win": refuting any of them would itself be a landmark discovery in proof complexity, cryptography or complexity theory.

But because the result is conditional, its confidence is conditional too. If those assumptions fail, the theorem's interpretation changes. And even if the assumptions hold, the guarantee is not full classical zero-knowledge; it is the paper's relaxed, proof-theoretic version.

So the right confidence is high that the paper establishes a coherent conditional possibility result; moderate that its assumptions describe the cryptographic world we actually live in; and low for any immediate practical consequence.

Why it matters

The paper opens a route that was supposed to be closed.

Classical theory says: full zero-knowledge cannot be one message without setup, and cannot be perfectly sound. Ilango's paper says: if we ask for the consequences of zero-knowledge that can be tested in security games, and if we allow the security definition to depend on what a rulebook can or cannot efficiently refute, then much of the useful behaviour can be recovered — with one message, no setup and perfect soundness.

That is not a small definitional tweak. It is a different way to think about cryptographic guarantees. Instead of asking only what exists, ask what your rulebook can rule out. Instead of treating unprovability as a philosophical nuisance, use it as structure.

The practical world may not change tomorrow. But the conceptual map does. There is now a formal sense in which "no one can efficiently prove that the secret leaked" can be strong enough to recover many of the game-based protections we wanted from "the secret did not leak."

That is why Gödel belongs in the title.

Clean summary

Zero-knowledge proofs let a prover convince a verifier that a statement is true without revealing the witness. Classical impossibility results say zero-knowledge cannot be squeezed into one message without setup, and cannot have perfect soundness. Rahul Ilango's paper does not refute those impossibilities. It defines a weaker notion, effectively zero-knowledge: instead of requiring that a simulator really exists, it requires that a chosen proof system — a formal rulebook like ZFC — cannot efficiently prove that no simulator exists. Under major assumptions from cryptography (non-interactive witness indistinguishable proofs) and proof complexity (no optimal proof system exists), the paper constructs one-message provers for NP/SAT with no setup and perfect soundness that achieve the falsifiable, game-based consequences of zero-knowledge property by property. A single prover covering all "natural" such properties is a further, partly conjectural extension — and covering literally every falsifiable property is likely impossible, because proofs remain reusable. The result is theoretical and conditional, not a deployed primitive, but it shows a new way to use proof-theoretic unprovability as a cryptographic resource.

No-BS check

What the paper shows: Under stated assumptions, one can build one-message, no-setup, perfectly sound provers for NP/SAT that are effectively zero-knowledge relative to any chosen proof system, and that achieve each falsifiable game-based consequence of classical zero-knowledge.

What is plausible but not proven unconditionally: That the needed proof-complexity and cryptographic assumptions hold. They are serious, well-studied assumptions — and the paper shows they are essentially necessary as well as sufficient — but still assumptions.

What it does not show: Classical zero-knowledge with no interaction, no setup and perfect soundness; a practical system ready for deployment; deniability or non-reusability of proofs; or that Gödel's incompleteness theorem by itself secures cryptography.

Main limitations: The guarantee is a relaxation of zero-knowledge; the broadest version depends on multiple assumptions; the single-universal-prover claims remain partly conjectural; and the result is primarily foundational.

How much confidence should a general reader have? High that this is an important conditional theory result if the definitions are accepted. Moderate that the assumptions capture reality. Low for immediate practical deployment. The safe takeaway is: the paper does not break the zero-knowledge impossibilities; it finds a new proof-theoretic way around the parts of them that matter for many security games.

Sources

Ilango, IACR ePrint 2025/1296
https://eprint.iacr.org/2025/1296

FOCS 2025, DOI 10.1109/FOCS63196.2025.00058
https://doi.org/10.1109/FOCS63196.2025.00058