The Clean Paper

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

Ein kryptografischer Beweis kann sein Geheimnis verbergen, indem die Nichtexistenz des Simulators schwer zu beweisen ist

Zero-Knowledge-Beweise erlauben es, eine Aussage zu beweisen, ohne den Zeugen preiszugeben. Die klassische Theorie sagt, dass dies im vollen Sinn nicht gleichzeitig mit nur einer Nachricht, ohne vertrauenswürdiges Setup und mit perfekter Soundness möglich ist. Rahul Ilangos Arbeit lässt diese Unmöglichkeit nicht verschwinden. Sie verändert das Ziel: Statt zu verlangen, dass ein Simulator tatsächlich existiert, verlangt sie, dass kein gewähltes Beweissystem effizient beweisen kann, dass der Simulator nicht existiert. Unter bedeutenden Annahmen aus Beweiskomplexität und Kryptografie genügt dies, um falsifizierbare, spielbasierte Folgen von Zero-Knowledge Eigenschaft für Eigenschaft zurückzugewinnen. Es geht nicht um eine sofort einsetzbare Internet-Primitive, sondern um einen beweistheoretischen Weg, mathematische Unbeweisbarkeit als kryptografische Deckung zu nutzen.

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

Der Trick besteht nicht darin zu beweisen, dass das Geheimnis verborgen ist

Beginnen wir mit der einfachsten Version von Zero-Knowledge.

Alice möchte Bob davon überzeugen, dass ein Sudoku eine Lösung besitzt. Schickt sie ihm die Lösung, ist Bob überzeugt — aber das Rätsel ist ruiniert. Was sie will, ist etwas Seltsameres: ein Beweis dafür, dass eine Lösung existiert, ohne die Lösung selbst preiszugeben.

Genau das verspricht ein Zero-Knowledge-Beweis. Der Prover (Alice) überzeugt den Verifizierer (Bob), dass eine Aussage wahr ist, ohne über die Wahrheit dieser Aussage hinaus etwas zu verraten.

Dieses Versprechen hat allerdings einen Preis. Ein gewöhnlicher mathematischer Beweis besitzt zwei angenehme Eigenschaften. Er besteht aus **einer Nachricht**: Man schreibt ihn auf, übergibt ihn und geht wieder. Und er hat **perfekte Soundness**: Für eine falsche Aussage existiert überhaupt kein gültiger Beweis. Klassische Unmöglichkeitsresultate besagen, dass Zero-Knowledge auf beide Eigenschaften verzichten muss — und nicht nur auf ihre Kombination; jede einzelne ist für sich bereits ausgeschlossen.

Erstens benötigt ein Zero-Knowledge-Beweis Interaktion. Schickt Alice nur eine einzige Nachricht, ohne vorher eingerichtetes vertrauenswürdiges Setup, bricht die Zero-Knowledge-Garantie zusammen — unabhängig davon, wie viel Soundness man im Gegenzug preiszugeben bereit wäre.

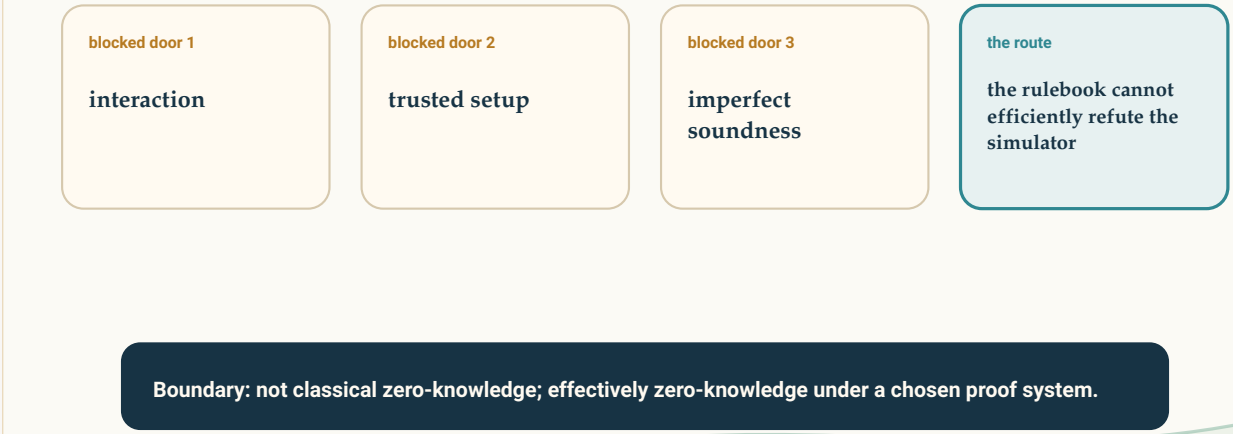
Zweitens braucht ein Zero-Knowledge-Beweis eine kleine Fehlertoleranz. Perfekte Soundness zu fordern zerstört überraschenderweise auch die Interaktion: Ein Verifizierer, der unabhängig von seinen Zufallsentscheidungen niemals getäuscht werden kann, könnte diese Entscheidungen genauso gut im Voraus festlegen. Sobald der Verifizierer vorhersehbar ist, kann Alice alles in einer einzigen Nachricht beantworten — und genau dieser Fall war bereits unmöglich.

Rahul Ilangos Arbeit beschreibt einen Weg um diese doppelte Mauer herum. Nicht indem sie so tut, als gäbe es die Mauer nicht, und auch nicht indem sie im unmöglichen Setting klassisches Zero-Knowledge erzeugt. Der Schritt ist subtiler: Sie schwächt die Bedeutung von „verrät nichts“ ab — aber so, dass die Sicherheitsmerkmale erhalten bleiben, die Kryptografen tatsächlich testen können.

Das Ergebnis heißt **effectively zero-knowledge** — effektiv Zero-Knowledge.

A proof without leaking the witness

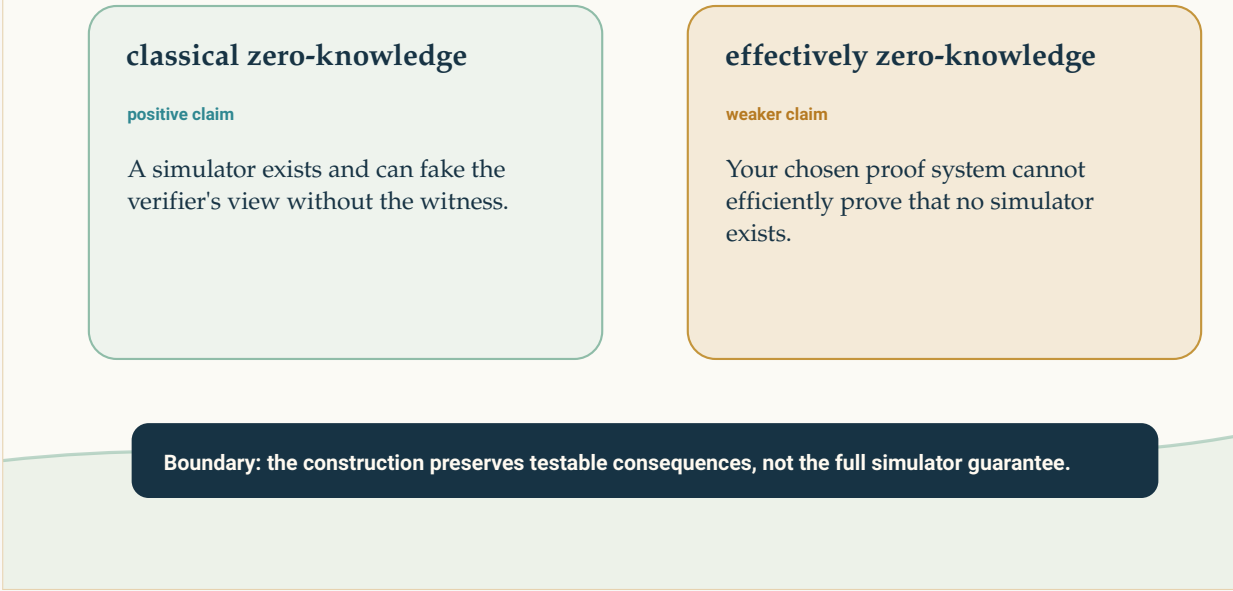
The paper keeps the ordinary-proof shape by weakening what privacy must prove.



Zero-Knowledge steht vor drei verschlossenen Türen: Interaktion, vertrauenswürdiges Setup und nicht-perfekte Soundness. Ilangos Konstruktion nimmt einen anderen Weg: Das Regelwerk kann den Simulator nicht effizient widerlegen.

Classical privacy vs effective privacy

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



Klassisches Zero-Knowledge fragt, ob ein Simulator existiert; „effectively zero-knowledge“ fragt nur, ob das

gewählte Regelwerk effizient beweisen kann, dass keiner existiert. Gerade diese schwächere Frage erlaubt es der Konstruktion, eine Nachricht, kein Setup und perfekte Soundness gleichzeitig zu behalten.

Der alte Test: Ein Simulator existiert

Die klassische Formalisierung von Zero-Knowledge verwendet einen fiktiven Helfer, den **Simulator**.

Die Idee lautet: Stellen wir uns Jane vor, die Alices Geheimnis **nicht** kennt. Wenn Jane ganz allein Beweise erzeugen kann, die genauso aussehen wie jene, die Bob von Alice erhalten hätte, dann haben Alices Beweise Bob nichts Neues beigebracht. Jane konnte das Erlebnis bereits ohne Alices Geheimnis nachbilden.

Klassisches Zero-Knowledge verlangt deshalb einen tatsächlichen Simulator. Es muss einen effizienten Algorithmus geben, der scheinbar echte Beweise erzeugen kann, ohne das Geheimnis zu kennen — im Fachjargon den *Zeugen* (*witness*); beim Sudoku ist der Zeuge einfach das vollständig gelöste Gitter.

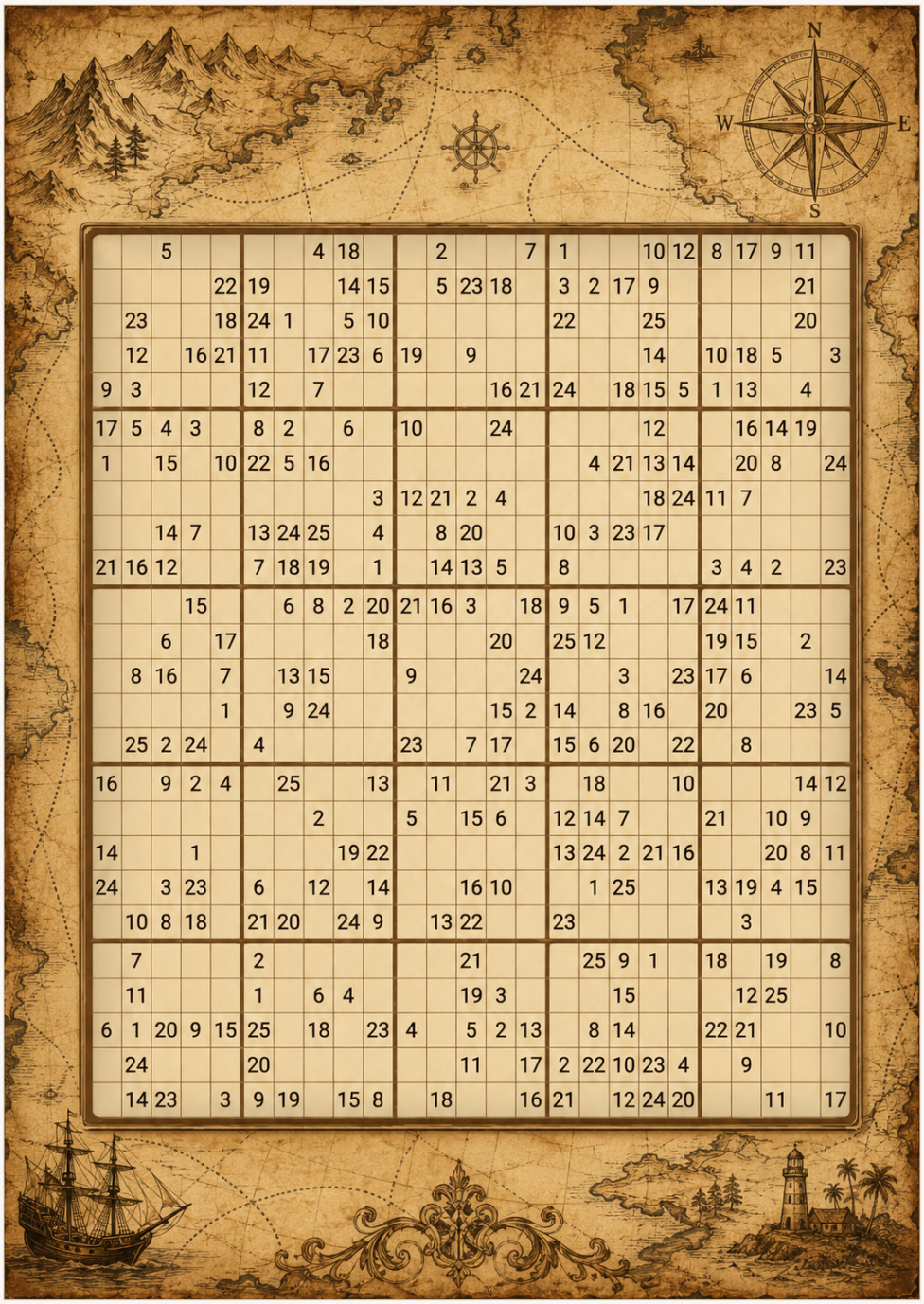
Diese Definition ist mächtig, aber genau hier greifen die alten Unmöglichkeitsresultate. Die Intuition ist einfach: Ein wirklich nicht-interaktiver Beweis ist nur ein String. Sobald Bob diesen String besitzt, kann er ihn jemand anderem zeigen. Er hat damit die Fähigkeit gewonnen, die Aussage gegenüber Dritten zu beweisen — und das klingt bereits nach mehr als „nichts“. Die klassischen Sätze machen aus dieser Intuition die oben genannten Unmöglichkeiten.

> **Die drei Eigenschaften, auf denen diese Arbeit besteht**
>
> Der Titel der Arbeit nennt drei Anforderungen:
> **Keine Interaktion:** Alice schickt einen einzigen Beweisstring. Es gibt kein Hin und Her im Protokoll.
> **Kein Setup:** Alice und Bob verlassen sich weder auf einen vertrauenswürdig erzeugten gemeinsamen Referenzstring noch auf andere vorab vereinbarte öffentliche Zufallswerte. Viele Systeme, die „nicht-interaktives Zero-Knowledge“ heißen, benötigen dennoch ein Setup; hier ist tatsächlich null Setup gemeint.
> **Perfekte Soundness:** Für eine falsche Aussage existiert kein gültiger Beweis. Nicht „sie wird fast nie akzeptiert“ — es existiert schlicht kein gültiger Beweis.
> Genau diese drei Eigenschaften besitzt gewöhnliche schriftliche Mathematik — und klassisches Zero-Knowledge kann sie, wie oben erklärt, nicht gleichzeitig bewahren.

Der Unterschied als Mega-Sudoku

Hier ist eine absichtlich vereinfachte Analogie, mit der sich der Unterschied greifen lässt.

Für den ernsthaften Teil der Analogie darf es kein gewöhnliches 9×9-Sudoku sein. Das ist zu klein und zu endlich: Ein Computer kann es einfach lösen oder beweisen, dass keine Lösung existiert. Stellen wir uns stattdessen eine Familie von **MegaSudoku(n)**-Rätseln vor. Wir skalieren die üblichen Regeln: Wähle eine Blockgröße n, setze $N = n^2$ und baue ein $N \times N$-Gitter aus $n \times n$-Blöcken mit N verschiedenen Symbolen. Das gewöhnliche Sudoku ist nur der winzige Fall $n = 3$, $N = 9$: ein 9×9-Gitter, 3×3-Blöcke und neun Symbole. Die Geschichte aus der Beweiskomplexität beginnt erst, wenn n wachsen darf und das Gitter zusätzliche „Gadgets“ enthalten kann, die es wie eine SAT-Formel im Sudoku-Kostüm funktionieren lassen. Eine SAT-Formel ist im Kern eine Liste von Ja/Nein-Bedingungen: Kann man den Variablen Wahr/Falsch-Werte zuweisen, sodass alle Bedingungen erfüllt sind?



Ein 25×25-Sudoku: Seine Regeln lassen sich überprüfen, ohne das fertige Gitter zu zeigen — ein visuelles Stellvertreterbild für einen Beweis, der eine verborgene Lösung, den Zeugen, verifiziert.

Sudoku und SAT: dasselbe Rätsel in zwei Kostümen

Die Aussage, ein Sudoku könne sich „wie eine SAT-Formel verhalten“, ist keine Metapher. Die Übersetzung funktioniert in beide Richtungen, und die einfache Richtung lässt sich vollständig hinschreiben.

Von Sudoku zu SAT. SAT kennt nur Wahr/Falsch. Also bekommt jedes Tripel aus (Zeile, Spalte, Wert) eine boolesche Variable: *x(r,c,v)* bedeutet „Die Zelle in Zeile *r*, Spalte *c* enthält den Wert *v*.“ Ein 4×4-Sudoku (2×2-Blöcke, Werte 1–4) braucht 4·4·4 = 64 Variablen; das klassische 9×9-Sudoku braucht 729. Jede Sudoku-Regel wird anschließend zu einer Gruppe von Klauseln. (Eine Klausel ist ein ODER aus Variablen oder ihren Negationen; die gesamte Formel ist das UND aller Klauseln.)

Jede Zelle enthält mindestens einen Wert — eine Klausel pro Zelle:

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

Jede Zelle enthält höchstens einen Wert — eine „nicht beide“-Klausel für jedes Wertepaar:

$$\neg x(1,1,1) \vee \neg x(1,1,2) \quad \neg x(1,1,1) \vee \neg x(1,1,3) \quad \text{… und so weiter für alle sechs Paare.}$$

Jede Zeile enthält jeden Wert — für Zeile 1 und den Wert 3: mindestens einmal,

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

und höchstens einmal: $\neg x(1,1,3) \vee \neg x(1,2,3)$, und so weiter für jedes Zellpaar der Zeile.

Spalten und Blöcke — dieselben Klauselgruppen, nur mit anderen Zellgruppen. Für den Block oben links und den Wert 2:

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

plus die paarweisen „nicht beide“-Klauseln.

Die vorgegebenen Zahlen — der einfachste Teil: Jede Vorgabe ist eine Klausel mit nur einer Variable. Eine gedruckte 3 in der linken oberen Ecke wird zur Klausel

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

Das UND all dieser Klauseln ist genau dann erfüllbar, wenn das Sudoku eine Lösung hat — und eine erfüllende Belegung *ist* die Lösung: Man liest ab, welche x(r,c,v) wahr sind, und trägt sie ins Gitter ein. Beim 9×9-Sudoku ergeben sich 729 Variablen und einige Tausend Klauseln, die ein moderner SAT-Solver in Millisekunden erledigt. Beachte die Vorgaben-Klausel x(1,1,3): Sie sagt „Diese Zelle ist exakt 3“, nicht „Diese Zellen sind alle verschieden“. Genau diese Asymmetrie erzwingt bei den Vorgabenzellen den zusätzlichen Trick im Protokoll weiter unten.

Von SAT zu Sudoku. Die Arbeit braucht die umgekehrte, schwierigere Richtung: Aus einer *beliebigen* SAT-Formel muss ein Mega-Sudoku gebaut werden, das genau dann lösbar ist,

wenn die Formel erfüllbar ist. Die nativen Sudoku-Regeln können nur sagen: „Diese Zellen enthalten alle verschiedene Symbole." Beliebige logische Bedingungen müssen deshalb *gebaut* werden — und genau dafür dienen die Gadgets. Ein Gadget ist eine kleine vorgefertigte Zellgruppe, eine pro Klausel der Formel. Bestimmte Zellen übernehmen darin die Rolle von Variablen (ihr Symbol codiert Wahr oder Falsch), und die internen Bedingungen werden so konstruiert, dass genau jene legalen Ausfüllungen möglich sind, die diese Klausel erfüllen. Das ist Standardhandwerk aus NP-Vollständigkeitsbeweisen; für verallgemeinertes Sudoku wurde es 2003 von Yato und Seta ausgearbeitet.
Zusammen sagen beide Richtungen: N×N-Sudoku und SAT sind dasselbe Problem in unterschiedlichen Kostümen. Genau das erlaubt diesem Artikel — und der Arbeit —, mit Gittern und Symbolen eine Geschichte über ganz NP zu erzählen.

Der Zeuge bleibt leicht vorstellbar. Alice kennt eine vollständige gültige Ausfüllung des Mega-Sudokus. Bob möchte davon überzeugt werden, dass eine solche Ausfüllung existiert, aber Alice will sie nicht preisgeben. Schickt sie das ganze ausgefüllte Gitter, ist Bob überzeugt, doch das Geheimnis ist weg.

In der klassischen Zero-Knowledge-Version interagieren Alice und Bob. Ein altes Anschauungsmodell arbeitet mit verdeckten Plättchen. Alice verbirgt das gelöste Gitter, benennt vor jeder Runde heimlich die Symbole um und lässt Bob genau eine zufällig ausgewählte lokale Bedingung kontrollieren: eine Zeile, eine Spalte, einen Block oder ein Gadget. Zeigen die geöffneten Zellen lauter verschiedene Symbole, gewinnt Bob Vertrauen. Danach wird alles wieder verdeckt, und die Symbole werden erneut zufällig umbenannt. (Eine Feinheit: Die vorgegebenen Zahlen des Rätsels brauchen einen zusätzlichen Trick, weil die Umbenennung auch sie versteckt. Die folgende Anmerkung erklärt, wie klassische Protokolle das lösen; für die weitere Geschichte reicht das vereinfachte Bild.)

Wie klassische Protokolle die Vorgabenzellen tatsächlich behandeln

Der Umbenennungstrick hat einen blinden Fleck. Zeilen-, Spalten- und Blockregeln sagen jeweils: „Diese Zellen sind alle verschieden", und *alle verschieden* bleibt unter jeder Umbenennung der Symbole wahr. Eine Vorgabe sagt dagegen: „Diese Zelle enthält exakt 5." Nach der Umbenennung sieht Bob nur $\sigma(5)$ — irgendein maskiertes Symbol —, ohne die Umbenennung σ zu kennen. Er kann also nichts prüfen. Bliebe das ungelöst, könnte Alice beweisen, dass *irgendein* gültiges Gitter existiert, während sie die gedruckten Vorgaben vollständig ignoriert — und das beweist nichts über *dieses* Rätsel. Die klassische Literatur verwendet zwei Standardlösungen.
Die Palette. Man ergänzt das verborgene Gitter um eine zusätzliche Zeile mit N Zellen — eine Palette, die Alice in einer festen öffentlichen Reihenfolge mit den Symbolen 1…N füllt und dann zusammen mit allem anderen umbenennt, sodass dort $\sigma(1)…\sigma(N)$ stehen. Bobs zufällige Herausforderung erhält nun eine zusätzliche Option. Neben Zeile, Spalte, Block oder Gadget darf er **die Palette plus eine Vorgabenzelle** auswählen. Alice deckt beides auf; die Palette verrät die Umbenennung dieser Runde, und Bob prüft, ob die Vorgabenzelle genau die umbenannte Version der gedruckten Vorgabe zeigt. Das bleibt Zero-Knowledge, weil Bob nur σ

erfährt — das in jeder Runde frisch gezogen wird und allein wertlos ist — sowie den Wert einer Zelle, den er aus dem Rätsel ohnehin schon kannte. Über die geheimen Zellen wird nichts verraten, und ein Simulator kann diese Sicht erzeugen, indem er selbst ein zufälliges σ wählt. Das Protokoll ist sound, weil eine betrügende Alice pro Runde mit fester Wahrscheinlichkeit erwischt wird; die Runden werden so lange wiederholt, bis die verbleibende Unsicherheit vernachlässigbar ist.

Die Vorgaben wegkompilieren. Eine strukturellere Variante entfernt die Sonderprüfung, statt sie hinzuzufügen. Anstatt den Vorgabewert *zu verifizieren*, erzwingt man ihn mit Ungleichheitsbedingungen: Die Vorgabenzelle wird mit jeder Palettenzelle außer jener für den eigenen Wert verknüpft — „verschieden von $\sigma(1)$, verschieden von $\sigma(2)$, …, verschieden von allem außer $\sigma(5)$“. Das einzige Symbol, das die Zelle legal enthalten kann, ist damit die Vorgabe. Nun ist jede Bedingung wieder vom Typ „Diese beiden unterscheiden sich“ — invariant unter Umbenennung und genauso prüfbar wie eine Zeile. Derselbe Kunstgriff wird für vorgefärbte Knoten im klassischen Graphfärbungsprotokoll verwendet; er entspricht auch dem Geist der oben erwähnten *Gadgets*: Im Bild MegaSudoku-als-SAT werden die Vorgaben wie jede andere Bedingung in Ungleichheits-Gadgets kompiliert.

Das physische Protokoll. Das reale Kartenprotokoll für Sudoku (Gradwohl, Naor, Pinkas und Rothblum, 2007) verwendet überhaupt keine Umbenennung und klärt die Vorgaben, bevor das Verbergen beginnt. Für jede Zelle legt Alice drei identische Karten mit dem Zellwert aus — verdeckt für geheime Zellen, aber **offen für Vorgabenzellen**, sodass Bob mit eigenen Augen sieht, dass die Vorgaben eingehalten werden, bevor auch diese Karten umgedreht werden. Anschließend kommt aus jeder Zelle je eine Karte in das Paket ihrer Zeile, eine in das Paket ihrer Spalte und eine in das Paket ihres Blocks. Jedes Paket wird gemischt und aufgedeckt, und Bob prüft, dass alle N Symbole enthalten sind. Das Mischen zerstört die Positionsinformation — darin liegt das Zero-Knowledge —, während die Vorgaben bereits beim Austeilen festgenagelt wurden.

In beiden Fällen ist die Lektion dieselbe, zu der dieser Artikel immer wieder zurückkehrt: Ein Zero-Knowledge-Protokoll ist eine sorgfältige Buchführung darüber, *welche* Tatsachen das Verbergen überleben. Umbenennen bewahrt „alle verschieden“ und löscht „gleich 5“ — also muss „gleich 5“ auf einem anderen Weg wieder hineingeschmuggelt werden.

Das ist nicht das Protokoll der Arbeit. Es ist nur das Anschauungsmodell für klassisches Zero-Knowledge:

- Alice und Bob kommunizieren hin und her.
- Bob wählt zufällige Prüfungen.
- Alice offenbart nur lokale Konsistenz, nicht die ganze Lösung.
- Der Datenschutzbeweis funktioniert, indem gezeigt wird, dass Bobs Sicht auch ohne Alices geheime Lösung hätte erzeugt werden können.

Klassisches Zero-Knowledge beruht also auf einer positiven Tatsache:

> Ein Simulator existiert tatsächlich.

Nun entfernen wir die komfortablen Teile. Alice schickt einen einzigen Beweisstring und geht. Es gibt kein vertrauenswürdiges Setup, keinen zuvor vorbereiteten gemeinsamen Zufallsstring, und Bob darf niemals ein falsches Rätsel akzeptieren. Genau in diesem Setting kann klassisches Zero-Knowledge nicht überleben.

Bevor der Trick funktioniert, brauchen wir noch eine Figur. Fixiere ein **Regelwerk**: ein formales Beweissystem im logischen Sinn — eine feste Menge von Axiomen plus mechanische Regeln, mit denen geschriebene mathematische Beweise geprüft werden. ZFC, das Standardsystem von Axiomen für die Mathematik, ist das kanonische Beispiel. Von hier an wird alles relativ zu einem vorab gewählten Regelwerk formuliert, und die Wahl ist flexibel: Die Konstruktion funktioniert für jedes festgelegte Regelwerk, einschließlich ZFC.

(Eine terminologische Anmerkung aus der Arbeit selbst: „Beweissystem“ meint hier immer dieses Regelwerk — also das formale System, das mathematische Beweise prüft — und niemals die Nachrichten, die Alice verschickt. Alices und Bobs kryptografische Mechanik heißt „Prover und Verifizierer“.)

Die Gödel-artige Version behält die Mega-Sudoku-Geschichte, verändert aber den Beweis.

Wähle ein zweites Bedingungssystem derselben dargestellten Größe und nenne es **D**. In der Geschichte sind S und D zwei MegaSudoku(n)-Rätsel im selben Format. Hinter den Kulissen kann D als schwierige logische Formel anderer Größe entstanden sein; falls nötig, lässt sie sich mit harmlosen Dummy-Bedingungen auffüllen, bis sie ins gleiche Gitter passt. D wird aus einer logischen Formel gebaut, die tatsächlich **unerfüllbar** ist: Es gibt keine mögliche Wertebelegung, die alle Bedingungen wahr macht — so wie ein kaputtes Rätsel keine legale Komplettlösung besitzt. Ein Spielzeugbeispiel wäre eine Formel, die gleichzeitig „X ist wahr“ und „X ist falsch“ verlangt. D besitzt also keine gültige Ausfüllung.

D darf aber kein kaputtes Rätsel sein, dessen Fehler *leicht nachzuweisen* ist. Das Spielzeugbeispiel scheitert daran: Jedes vernünftige Regelwerk widerlegt „X und nicht X“ in einer Zeile. D muss auf eine Weise falsch sein, die das gewählte Regelwerk nicht mit einem kurzen Argument zertifizieren kann. Könnte das Regelwerk D mit einem kurzen Beweis widerlegen, würde die nachfolgende Konstruktion zusammenbrechen: Der alternative Weg, der Beweise ohne Alices Geheimnis hätte erzeugen können, ließe sich formal ausschließen — und damit auch die Datenschutzgarantie. D wird deshalb aus einer Familie gewählt, die das festgelegte Regelwerk nicht effizient widerlegen kann: Innerhalb dieses Regelwerks gibt es keinen kurzen Beweis dafür, dass D keine Lösung besitzt.

Alices Ein-Nachrichten-Beweis betrifft dann eine Entweder-oder-Aussage:

> Entweder das echte Mega-Sudoku S besitzt eine Lösung, oder der Köder D besitzt eine Lösung.

Das ist die logische Verbindung. D wird **nicht** auf magische Weise so erzeugt, dass dadurch S wahr wird. Der Beweis argumentiert nicht: „D hat keine Lösung, also hat S eine Lösung.“ Er beweist die Disjunktion **S oder D**. Perfekte Soundness bedeutet, dass eine falsche Disjunktion keinen gültigen Beweis besitzen kann. Da D in Wirklichkeit falsch ist — es hat keine Lösung —, kann die Disjunktion nur dann wahr sein, wenn S wahr ist. Wird der Beweis akzeptiert, muss S also eine Lösung haben.

Der Köder kann ein falsches S nicht wahr machen.

Für den Zero-Knowledge-artigen Teil fragen wir jedoch, was passieren würde, *wenn* D eine Lösung hätte. Diese Köderlösung wäre ein alternativer Zeuge. Mit ihr könnte jemand Beweise erzeugen, ohne Alices echte Mega-Sudoku-Lösung zu kennen — also als Simulator fungieren. In Wirklichkeit besitzt D keine Lösung; dieser Simulationsweg ist daher geschlossen. Entscheidend ist aber: Das Regelwerk kann nicht effizient beweisen, dass er geschlossen ist.

D hat damit zwei Aufgaben. Für die **Soundness** ist D falsch; ein gültiger Beweis von „S oder D“ erzwingt deshalb S. Für **effektives Zero-Knowledge** ist D schwer zu widerlegen; das Regelwerk kann den Köderweg, der Simulation möglich gemacht hätte, nicht schnell ausschließen.

Der Sicherheitstest lautet damit nicht länger:

> Können wir beweisen, dass ein Simulator tatsächlich existiert?

Sondern:

> Kann dein Regelwerk effizient beweisen, dass der Simulator unmöglich ist?

Ist die Antwort nein, folgt etwas überraschend Starkes: Jede Sicherheitsgarantie, die (a) durch Ausführen eines Tests beobachtbar ist und (b) innerhalb dieses Regelwerks nachweislich aus der Existenz eines Simulators folgen würde, gilt tatsächlich. Ein erfolgreicher Angriff auf eine solche Garantie würde selbst auf die fehlende kurze Widerlegung hinauslaufen — und genau diese kurze Widerlegung existiert nicht. Das ist das „effektiv“ in effektivem Zero-Knowledge.

Der Kontrast für das Klassenzimmer lautet also:

Klassisches Zero-Knowledge: Die Beweise sind sicher, weil ein Simulator existiert.

Gödel-artiges effektives Zero-Knowledge: Für beobachtbare Sicherheitstests werden die Beweise als sicher behandelt, weil das Regelwerk nicht effizient beweisen kann, dass ein Simulator unmöglich ist.

Die zweite Aussage ist schwächer. Genau deshalb kann die Arbeit die drei Merkmale behalten, an denen die klassische Version scheiterte: eine Nachricht, kein Setup und perfekte Soundness.

Der neue Test: Du kannst nicht beweisen, dass der Simulator fehlt

Ilangos Abschwächung verändert die Frage.

Klassisches Zero-Knowledge fragt:

> Existiert ein Simulator?

Effektives Zero-Knowledge fragt etwas Schwächeres:

> Kann dein gewähltes Regelwerk effizient beweisen, dass kein Simulator existiert?

Das klingt wie ein technischer Ausweichtrick, ist aber die Kernidee. Die Konstruktion befindet sich in einem merkwürdigen Zustand: Ein Simulator existiert tatsächlich **nicht** — die Arbeit sagt das ausdrücklich —, aber das festgelegte Regelwerk kann nicht effizient beweisen, dass er nicht existiert. Wenn jede schlechte Folge, die uns interessiert, eine solche Widerlegung voraussetzen würde, verhält sich das System bezüglich dieser Folgen dennoch wie Zero-Knowledge.

Hier kommt Gödel ins Spiel. Nicht als Dekoration und nicht im Sinn von „Gödel macht Kryptografie sicher“. Die Verbindung ist beweistheoretisch. Ein Regelwerk heißt **optimal**, wenn es in einem präzisen Sinn das bestmögliche ist: Wann immer irgendein Regelwerk eine Formel der relevanten Art mit einem kurzen Beweis widerlegen kann, kann das optimale Regelwerk dies ebenfalls, mit einem höchstens polynomial längeren Beweis. Krajíček und Pudlák vermuteten 1989, dass **kein optimales Beweissystem existiert**: Welches Regelwerk man auch festlegt, es gibt ein anderes, das bestimmte Familien wahrer Aussagen sehr viel kürzer beweisen kann. Das ist eine der zentralen offenen Vermutungen der Beweiskomplexität und gewissermaßen der endliche, komplexitätstheoretische Verwandte von Gödels Unvollständigkeitssatz: Manche wahren Aussagen besitzen im gewählten Regelwerk keinen kurzen Beweis — nicht weil sie prinzipiell unbeweisbar wären, sondern weil jedes feste Regelwerk einige kurze Wahrheiten ohne kurze Beweise zurücklässt.

Die Arbeit nimmt diese Vermutung an, in einer leicht stärkeren „infinitely often“-Variante, wie sie bei kryptografischer Nutzung von Vermutungen üblich ist. Ein Satz von Krajíček und Pudlák liefert dann einen konkreten Gewinn: Für jedes Regelwerk gibt es eine Folge von Formeln, die tatsächlich unerfüllbar sind, für die dieses Regelwerk aber keine kurzen Widerlegungsbeweise besitzt — und die, entscheidend, von einem effizienten Algorithmus *erzeugt* werden können. Diese letzte Eigenschaft, Uniformität, verwandelt die Idee von einer reinen Existenzaussage in einen Algorithmus, den Alice tatsächlich ausführen kann: Ihre Köder D kommen von einem Fließband, nicht aus dem Nichts.

Der kryptografische Kunstgriff besteht darin, genau diesen Mangel an Beweiskraft zu nutzen.

Was die Konstruktion tut

Hier ist die Konstruktion der Arbeit auf ihre Form reduziert.

Fixiere ein Regelwerk — etwa ZFC. Unter der Annahme aus der Beweiskomplexität gibt es eine effizient erzeugbare Folge von Formeln, die tatsächlich unerfüllbar sind, für die das Regelwerk aber keinen kurzen Beweis der Unerfüllbarkeit besitzt.

Nun konstruiere einen Ein-Nachrichten-Beweis der Form:

> Entweder die echte Aussage ist erfüllbar, oder diese spezielle schwierige Formel ist erfüllbar.

Die spezielle schwierige Formel ist nicht erfüllbar. Wenn die zugrunde liegende Beweismechanik perfekte Soundness besitzt, bedeutet die Akzeptanz der Nachricht daher weiterhin, dass die echte Aussage wahr ist. Das liefert perfekte Soundness.

Für die Zero-Knowledge-artige Sicherheit stellen wir uns nun vor, die spezielle schwierige Formel *wäre* erfüllbar. Dann könnte ihr Zeuge verwendet werden, um Beweise zu simulieren, ohne den echten Zeugen zu kennen. In Wirklichkeit ist die Formel nicht erfüllbar — doch das Regelwerk kann dies nicht effizient beweisen. Also kann es auch nicht effizient beweisen, dass der Simulator unmöglich ist.

Das ist das Scharnier der Konstruktion. Das System verbirgt das Geheimnis nicht, indem es einen klassischen Simulator erzeugt. Für eine große Klasse beobachtbarer Sicherheitstests verbirgt es das Geheimnis hinter der Unfähigkeit des Regelwerks, zu zertifizieren, dass der Simulator fehlt.

Was die Arbeit behauptet

Der Hauptsatz besteht aus mehreren Schichten. Das Kernergebnis lautet:

Unter einer Standardannahme der Kryptografie — der Existenz **nicht-interaktiver Witness-Indistinguishable-Beweise**, gut untersuchter Objekte, die aus mehreren etablierten Annahmenpaketen folgen — und unter der Vermutung aus der Beweiskomplexität, dass **kein (infinitely often) optimales Beweissystem existiert**, konstruiert die Arbeit für jedes gewählte Regelwerk einen Prover und Verifizierer für NP/SAT mit einer einzigen Nachricht, **perfekter Soundness** und ohne Setup, der relativ zu diesem Regelwerk effektiv Zero-Knowledge ist. (NP/SAT ist der übliche „härteste gemeinsame Nenner“ rätselartiger Probleme; Mega-Sudoku ist nur eines seiner Kostüme.)

Für die weitergehende Aussage über das Bewahren falsifizierbarer Sicherheitseigenschaften fügt die Arbeit eine weitere Standardannahme hinzu, die Derandomisierungsvermutung **P = BPP** — grob: Zufall gibt Algorithmen keine wesentliche zusätzliche Rechenkraft.

Aus der Sprache des Satzes übersetzt:

- Der Beweis besteht aus einer Nachricht.
- Es gibt kein vertrauenswürdiges Setup.
- Falsche Aussagen können nicht bewiesen werden.
- Der Prover ist nicht klassisch Zero-Knowledge — er besitzt keinen Simulator.
- Aber jede falsifizierbare, spielbasierte Sicherheitsfolge des klassischen Zero-Knowledge kann in diesem Setting erreicht werden.

„Falsifizierbar“ ist wichtig. Es bedeutet, dass ein Sicherheitsversagen getestet werden kann, indem man einen Angreifer in einem Spiel laufen lässt. Viele kryptografische Sicherheitsdefinitionen haben genau diese Form: Kann der Angreifer zwei Verschlüsselungen unterscheiden, eine Funktion invertieren, einen Zeugen rekonstruieren oder ein bestimmtes Experiment gewinnen? Der Satz liefert für jede falsifizierbare Eigenschaft jeweils einen passenden Prover. Ein einzelner Prover, der *alle* falsifizierbaren Eigenschaften gleichzeitig besitzt, ist wahrscheinlich unmöglich — der alte Wiederver-

wendungsangriff („Bob kann den Beweis anderen zeigen“) ist selbst eine falsifizierbare Eigenschaft, und genau dort scheitert die Konstruktion tatsächlich. Die Arbeit schlägt vor, dass ein einzelner Prover plausibel alle *natürlichen* falsifizierbaren Eigenschaften abdecken könnte — also jene, die in der kryptografischen Praxis tatsächlich auftreten. Dieser Teil ist jedoch ein bedingter Satz, der auf einem informellen Begriff von „natürlich“ und zusätzlich auf einer expliziten Vermutung beruht. Die Garantie zielt auf beobachtbare Sicherheitsverletzungen, nicht auf jede philosophische oder simulationsbasierte Bedeutung von Geheimhaltung.

Eine konkrete Folgerung verdient einen Namen: Die Konstruktion liefert die ersten nicht-interaktiven **Witness-Hiding-Beweise** mit uniformem Prover — „Ein Beweis dafür, dass ein Rätsel lösbar ist, hilft dir nicht dabei, die Lösung zu finden“, ohne Interaktion und ohne Setup. Dieses bescheiden klingende Objekt hatte sich jahrzehntelang einer Konstruktion entzogen.

Was dies nicht sagt

Dieser Abschnitt hält den Artikel ehrlich.

Die Arbeit sagt **nicht**, dass die alten Unmöglichkeitssätze falsch waren. Die Konstruktion umgeht sie, indem sie die Definition verändert.

Sie liefert **kein** gewöhnliches, klassisches Zero-Knowledge mit null Interaktion, null Setup und perfekter Soundness. Die Arbeit sagt ausdrücklich, dass der konstruierte Prover keinen Simulator besitzt.

Sie bedeutet **nicht**, dass der Beweis nicht wiederverwendet werden kann. Ein Ein-Nachrichten-Beweis kann weiterhin jemand anderem gezeigt werden; Eigenschaften wie Abstreitbarkeit (*deniability*) werden nicht bewahrt. (Nicht-interaktives Zero-Knowledge mit vertrauenswürdigem Setup besitzt dieselbe Einschränkung.)

Sie bedeutet **nicht**, dass hier ein praktisches, einsatzbereites Protokoll vorliegt. Das ist Komplexitätstheorie und kryptografische Grundlagenforschung. Das Ergebnis hängt von bedeutenden Annahmen aus Beweiskomplexität und Kryptografie ab und handelt davon, was prinzipiell möglich ist.

Und sie macht aus „Gödel“ **keine** magische Sicherheitsprimitive. Die Gödel-Verbindung verläuft über Beweissysteme, optimale Beweissysteme und endliche Analoga der Unvollständigkeit. Die brauchbare Intuition lautet nicht: „Unvollständigkeit schützt dein Passwort.“ Sondern: Wenn ein Regelwerk nicht effizient beweisen kann, dass ein Simulator unmöglich ist, können Angriffe, die genau einen solchen Beweis erfordern würden, bereits auf Ebene der Sicherheitsdefinition blockiert werden.

Warum das trotzdem interessant ist

Kryptografie verwandelt Härte häufig in Sicherheit. Faktorisieren ist schwer, also werden RSA-artige Annahmen nützlich. Gitterprobleme sind schwer, also wird gitterbasierte Kryptografie nützlich. Hier

ist die Härte ungewöhnlicher: nicht „Es ist schwer, ein Geheimnis zu berechnen“, sondern „Es ist schwer zu beweisen, dass ein bestimmtes Beweisobjekt nicht existieren kann“.

Deshalb fühlt sich die Arbeit so ungewöhnlich an. Sie behandelt Axiome und Regelwerke fast wie kryptografische Ressourcen. Die klassische Unmöglichkeit sagt, dass zwischen Soundness und Simulation eine Spannung besteht. Ilangos Schritt legt diese Spannung hinter einen beweistheoretischen Vorhang: Der Simulator fehlt, aber das formale System kann dieses Fehlen nicht effizient offenlegen.

Für Leser liegt das Überraschende nicht darin, dass dieses Verfahren heutige Zero-Knowledge-Systeme ersetzen wird. Das wird es wahrscheinlich nicht, zumindest nicht direkt. Überraschend ist vielmehr, dass eine Begrenzung der mathematischen Logik konstruktiv genutzt werden kann: nicht nur als Mauer, sondern als eine Art Deckung.

Wie belastbar sind die Belege?

Dies ist eine Theorem-Arbeit, daher bedeutet „Evidenz“ hier etwas anderes als in Biologie oder Astronomie. Die Frage ist nicht, ob ein Experiment repliziert wurde. Die Frage lautet, ob Definitionen, Annahmen und Beweiskette die Behauptung tragen.

Der Beweis ist formal, und die Arbeit legt ihre Annahmen offen. Diese Annahmen sind nicht beiläufig. Nicht-interaktive Witness-Indistinguishable-Beweise sind Standardobjekte der Kryptografie und folgen aus mehreren etablierten Annahmenpaketen. Die Vermutung, dass kein optimales Beweissystem existiert, ist eine zentrale offene Vermutung der Beweiskomplexität. P = BPP ist eine Standardannahme zur Derandomisierung und wird nur für den weitergehenden Satz über falsifizierbare Eigenschaften benötigt.

Die Arbeit argumentiert außerdem, dass diese Annahmen der angemessene Preis und kein willkürliches Gerüst sind: Sie beweist eine Umkehrung, nach der sie im Wesentlichen notwendig sind. Wenn Konstruktionen dieser Art überhaupt existieren, müssen nicht-interaktive Witness-Indistinguishable-Beweise existieren; und unter der Standardannahme von Einwegfunktionen kann kein optimales Beweissystem existieren. Die Annahmen sind zudem „Win-win“: Würde eine von ihnen widerlegt, wäre das selbst eine bedeutende Entdeckung in Beweiskomplexität, Kryptografie oder Komplexitätstheorie.

Da das Ergebnis bedingt ist, ist auch das Vertrauen darin bedingt. Falls diese Annahmen falsch sind, verändert sich die Interpretation des Satzes. Und selbst wenn sie stimmen, ist die Garantie kein volles klassisches Zero-Knowledge, sondern die abgeschwächte, beweistheoretische Variante der Arbeit.

Angemessen ist daher hohes Vertrauen darin, dass die Arbeit ein kohärentes bedingtes Möglichkeitsresultat etabliert; mittleres Vertrauen darin, dass ihre Annahmen die kryptografische Welt beschreiben, in der wir tatsächlich leben; und geringes Vertrauen in unmittelbare praktische Konsequenzen.

Warum das wichtig ist

Die Arbeit öffnet einen Weg, der als verschlossen galt.

Die klassische Theorie sagt: Volles Zero-Knowledge kann ohne Setup nicht aus nur einer Nachricht bestehen und kann keine perfekte Soundness besitzen. Ilangos Arbeit sagt: Wenn wir nach den Folgen von Zero-Knowledge fragen, die sich in Sicherheitsspielen testen lassen, und wenn die Sicherheitsdefinition davon abhängen darf, was ein Regelwerk effizient widerlegen kann oder nicht, dann lässt sich ein großer Teil des nützlichen Verhaltens zurückgewinnen — mit einer Nachricht, ohne Setup und mit perfekter Soundness.

Das ist keine kleine Änderung an einer Definition. Es ist eine andere Art, über kryptografische Garantien nachzudenken. Statt nur zu fragen, was existiert, fragt man, was das eigene Regelwerk ausschließen kann. Statt Unbeweisbarkeit als philosophisches Ärgernis zu behandeln, nutzt man sie als Struktur.

Die praktische Welt wird sich dadurch vielleicht morgen nicht verändern. Die begriffliche Landkarte aber schon. Es gibt nun einen formalen Sinn, in dem „Niemand kann effizient beweisen, dass das Geheimnis geleakt ist“ stark genug sein kann, um viele jener spielbasierten Schutzwirkungen zurückzugewinnen, die wir von „Das Geheimnis ist nicht geleakt“ wollten.

Deshalb gehört Gödel in den Titel.

Kurz zusammengefasst

Zero-Knowledge-Beweise erlauben einem Prover, einen Verifizierer von der Wahrheit einer Aussage zu überzeugen, ohne den Zeugen preiszugeben. Klassische Unmöglichkeitsresultate besagen, dass Zero-Knowledge ohne Setup nicht in eine einzige Nachricht gepresst werden kann und keine perfekte Soundness besitzen kann. Rahul Ilangos Arbeit widerlegt diese Unmöglichkeiten nicht. Sie definiert einen schwächeren Begriff, **effectively zero-knowledge**: Statt zu verlangen, dass ein Simulator tatsächlich existiert, verlangt sie, dass ein gewähltes Beweissystem — ein formales Regelwerk wie ZFC — nicht effizient beweisen kann, dass kein Simulator existiert. Unter bedeutenden Annahmen aus der Kryptografie (nicht-interaktive Witness-Indistinguishable-Beweise) und der Beweiskomplexität (kein optimales Beweissystem existiert) konstruiert die Arbeit Ein-Nachrichten-Prover für NP/SAT ohne Setup und mit perfekter Soundness, die die falsifizierbaren, spielbasierten Folgen von klassischem Zero-Knowledge Eigenschaft für Eigenschaft erreichen. Ein einzelner Prover, der alle „natürlichen“ solchen Eigenschaften abdeckt, ist eine weitergehende, teilweise vermutungsabhängige Erweiterung — und buchstäblich jede falsifizierbare Eigenschaft gleichzeitig abzudecken ist wahrscheinlich unmöglich, weil Beweise wiederverwendbar bleiben. Das Resultat ist theoretisch und bedingt, keine einsatzbereite Primitive; aber es zeigt einen neuen Weg, beweistheoretische Unbeweisbarkeit als kryptografische Ressource zu nutzen.

Nüchtern geprüft

Was die Arbeit zeigt: Unter den angegebenen Annahmen lassen sich Ein-Nachrichten-Prover für NP/SAT ohne Setup und mit perfekter Soundness konstruieren, die relativ zu jedem gewählten Beweissystem effektiv Zero-Knowledge sind und jeweils die falsifizierbaren, spielbasierten Sicherheitsfolgen des klassischen Zero-Knowledge erreichen.

Was plausibel, aber nicht unbedingt bewiesen ist: Dass die benötigten Annahmen aus Beweiskomplexität und Kryptografie tatsächlich gelten. Es sind ernsthafte, gut untersuchte Annahmen — und die Arbeit zeigt, dass sie im Wesentlichen nicht nur hinreichend, sondern auch notwendig sind —, aber es bleiben Annahmen.

Was die Arbeit nicht zeigt: Klassisches Zero-Knowledge ohne Interaktion, ohne Setup und mit perfekter Soundness; ein praktisches, einsatzbereites System; Abstreitbarkeit oder Nicht-Wiederverwendbarkeit der Beweise; oder dass Gödels Unvollständigkeitssatz für sich allein Kryptografie absichert.

Wichtigste Einschränkungen: Die Garantie ist eine Abschwächung von Zero-Knowledge; die breiteste Variante hängt von mehreren Annahmen ab; die Aussagen über einen einzigen universellen Prover bleiben teilweise vermutungsabhängig; und das Resultat ist in erster Linie grundlagentheoretisch.

Wie viel Vertrauen sollte ein allgemeiner Leser haben? Hohes Vertrauen darin, dass dies — wenn man die Definitionen akzeptiert — ein wichtiges bedingtes Resultat der Theorie ist. Mittleres Vertrauen darin, dass die Annahmen die Realität erfassen. Geringes Vertrauen in eine unmittelbare praktische Nutzung. Die sichere Kernaussage lautet: Die Arbeit bricht die Unmöglichkeitsresultate für Zero-Knowledge nicht; sie findet einen neuen beweistheoretischen Weg um jene Teile herum, die für viele Sicherheitsspiele relevant sind.

Quellen

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