	Prover (P)	statement Proof	Verifier (V)	Proof properties
legacy, common Proof		E.g.: statement = **theorem** **axioms** ⇒ … ⇒ **hypothesis** ⇒ **conclusion** logical implications by derivation rules		Theorem is TRUE ⇔ V is convinced **Completeness:** Theorem is TRUE ⇒ V is definitely convinced: $\mathcal{P}[\text{V is convinced}] \equiv 1$ **Soundness:** Theorem is FALSE ⇒ V cannot be convinced: $\mathcal{P}[\text{V is convinced}] \equiv 0$
Interactive Proof (IP)		**messages exchange** (several rounds)		**Completeness:** Statement is TRUE ⇒ $\mathcal{P}[\text{V is convinced}] > 1/2$ **Soundness:** Statement is FALSE ⇒ $\mathcal{P}[\text{V is convinced}] < 1/2$ pushing thresholds (see notes below) properties become **statistical** (vs. legacy **perfect** ones)

Notes

Interactive Proofs (**IP**) can be seen as a generalization of legacy ones, where the static nature of the latter is replaced by an active role of both the Prover and the Verifier, which send messages each other and also generate random values to be used in the proof. The 1/2 likelihood threshold could be regarded as "too weak", however probability can be increased in completeness -and reduced in soundness- as wished, repeating the IP many times and deciding by majority if V is convinced or not; this strategy can be justified with various degrees of math rigor and/or layman reasoning:

- analytical proof: using Chebyshev's inequality or Chernoff bound
- numerical check: calculating probabilities with various threshold values and numbers of repetitions
- by intuition: it seems reasonable that majority occurrence of an adversely-biased event is so much improbable as repetitions raise (so complementary favorably-biased event's probability increases)

More: it can be proved that, given an IP for a statement, for the same statement we can get an IP with perfect completeness $\mathcal{P}[\text{V is convinced}] \equiv 1$ and with V sending messages which contain only the random values it generates: the so called **Arthur-Merlin (AM)** -aka **public-coin**- **proof systems**

Proof	Proof of Knowledge (PoK)
proves that a statement is true; this happens thanks to these (already encountered) properties:	proves that the Prover knows something, so it's a proof whose statement is: "the prover has knowledge of …". Asserting that the Prover knows something means it could output an evidence, a **Witness W** about it, even if it's not expected during normal operations; that's why we also need a new special entity called extractor:
Perfect Completeness (remember, we can always make it Perfect) **Soundness**	**Perfect Completeness** (sometimes called **Non-triviality** in this context) existence of a **Knowledge Extractor (KE)** (sometimes called **Validity**) defined as an entity capable -outside the constraints of proof execution if needed- of extracting the **Witness W** of the Prover's knowledge, only **$\forall$P* s.t. $\mathcal{P}$[V is convinced] > err**

- V is also called **Knowledge Verifier**
- Don't be confused by term "knowledge": PoK could be non-ZK (in fact we haven't listed ZKness): trivially, a Prover sending its Witness to the Verifier
- When dealing with ZKPoK, the Knowledge Extractor doesn't break Zero Knowledge in the same way a Simulator doesn't break Soundness: KE obtains W from P using capabilities not available during normal proof execution: it has black-box oracle access to P (e.g. it can rewind it)
- err is the **Knowledge Extractor Error**, in the form of a threshold in KE definition below which it cannot extract W
- Soundness isn't explicitly stated among properties because it's implied by KE existence, so it's also called **Knowledge** (or **Special**) **Soundness**:

KE extract W ⇒ statement is TRUE (because W is an evidence of the statement) $\xrightarrow[\text{contrapositive}]{\text{taking the}}$ **statement is FALSE ⇒ KE never extracts W ⇒ $\underbrace{\forall P^* \ \mathcal{P}[\text{V is convinced}] \le \text{err}}$**

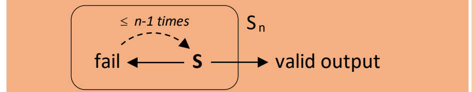
so if **err = 0** we get **perfect soundness** , and **err < 1/2** leads to IP's **statistical soundness** ; when **err ≥ 1/2** we are in the quite common case in which a satisfying PoK is obtained by **n** sequential repetitions of the original one: the resulting protocol will have **KE Error = err ^{n}** , permitting again statistical soundness for a large enough **n** (the Ali Baba Cave is an ELI5 example of this kind of proof by successful repetitions of a base one with too big error)

Argument (ARG)	Argument of Knowledge (ARK)
A proof with **computational soundness**, hence a relaxed soundness required to hold in a computationally-bounded context, where all involved entities are bounded: so ANY adversary P* and prescribed P as well (which is not generally required for proofs, even if it's implicit every time we consider a real-world implementation).	A proof of knowledge with **computational soundness**, maybe derived from a computational Knowledge Extractor, e.g.: **DLP is hard ⇒ computational KE ⇒ computational Soundness** (given that logical implication is transitive, reduction is as well; DLP again just an example)

from IP to ZKP: Zero-Knowledgeness property via Simulation paradigm

A **Zero-Knowledge Proof** (**ZKP**) is an IP holding one additional property: **Zero-Knowledgeness.** Roughly speaking, it states that verifier V learns nothing from the proof apart from the statement being true. Intuitively only the messages exchange with prover P can be the intermediary of this learning (if any), so a way to formalize ZKness is to show the existence of an entity - called simulator S - whose ONLY capability is to produce, together with V, a transcript of messages exchange indistinguishable from the original one: if the transcripts are indistinguishable, the learnings will be as well… but nothing can be learnt from a transcript produced by S because it has no capabilities apart from merely producing that transcript, so the same (= nothing) can be learnt from the IP.
Given that transcripts are random variables characterized by distributions (due to parties' capability to "toss dice"), we have 3 indistinguishability flavors:

Perfect Zero-Knowledge (PZK) ⊆



A sometimes failing S is invoked -at most *n times*- up to a valid output, whose distribution has to be **EQUAL** to original IP transcript distribution (so the upper bound of overall S_n failing ratio can be lowered as wished by increasing *n*)

Statistical Zero-Knowledge (SZK) ⊆

- No transcript instance *ts* can appear with too much different probabilities in original IP and S,
- if many transcript probabilities differ between original IP and S, differences must be tiny:

$$\Sigma_{ts} \left| \mathcal{P}[IP \rightarrow ts] - \mathcal{P}[S \rightarrow ts] \right| \text{ is "small"}$$

Computational Zero-Knowledge (CZK)

Transcripts distributions are practically indistinguishable when compared by ANY computationally-bounded entity.

Capturing explicitly the observing entity "class" in a proof isn't a simple task, so often a reduction to a widely accepted computationally-hard problem is used (because hardness is assumed when ALL entities are computationally-bounded):

DLP is hard ⇒ CZK or equivalently: **not CZK ⇒ DLP is broken**

(where DLP is the Discrete Logarithm Problem, and this is just an example)

To avoid breaking IP soundness (the simulator can produce a valid transcript, so it could impersonate a cheating prover claiming a false statement), S has and uses some power not available during a normal IP execution, e.g. **rewinding** of the verifier: imagine V having reached a certain point in the interaction, being wound back and resuming from a previous point. This is possible because S has **black-box oracle access** to V, basically meaning that it can call V's "next message" subroutine whenever it needs. (All of this can also be seen as V alone being the author of the simulation, leveraging full availability of its resources)

Black-box access, blind to V's internals, is known not being the most general usage of V by S; but it's a choice which also allows ZKPs closed under sequential composition (useful to preserve ZKness when we repeat IP for stronger soundness) and permits their embedding into outer protocols. More, simulation itself is a sufficient (⇒) but not necessary (⇐) condition for ZKness, so employing this paradigm already means missing any more comprehensive assumptions.

a taste of Non-Interactive Zero-Knowledge (NIZK)

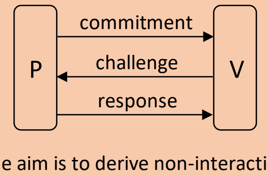
Exchange of messages between P and V seems unavoidable: given that S can produce a fake transcript, we cannot trust an exhibited-only transcript as really coming from an execution of the protocol: a ZKP is **non-transferable** to third-parties (not taking part in the proof) and so it's **deniable** to them. Still 1 round "exchanges" (just P making a proof available for a later check by V) are of huge practical interest because they don't require parties to be online at the same time; to make them possible the common **standard/plain model** (considered till now) is augmented by further assumptions:

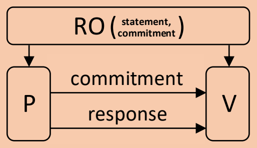
CRS

The enabling factor here is the existence of **Common Reference**(/**Random**) **String** drawn from some(/uniform) probability distribution and known by both P and V. Original inefficiencies of this approach have been partially solved by the quite recent *pairing-based cryptography*, however the common string is just assumed as available, needing a de-facto unspecified **trusted-setup** protocol producing it before NIZK proof execution

Fiat–Shamir (FS) heuristic

This strategy applies to **Sigma** (Σ) **protocols**, which are public-coin proofs with 3-stages structure: a P's random commitment followed by a V's random challenge (this part of the exchange can be repeated multiple times), and a final P's response. The trick is to substitute V's challenge with a **Random Oracle** (**RO**) output, available to both P and V:





A **RO** is an IDEAL function returning a random uniformly-distributed output (but always the same) for a given input. In FS, input includes all transcript's messages up to RO call, because miming the Σ protocol requires the challenge to PROVABLY (to V's benefit) come after its commitment: just a P's random toss wouldn't be enough. Soundness for FS also requires all public data into input, e.g. proof's statement.

The aim is to derive non-interactive proof's properties from their Σ protocol's counterparts. Completeness follows trivially and Soundness can also be derived. Note: FS always results in an Argument, because unbounded P* could "overcome" RO thanks to unlimited queries; and anyway: Σ Soundness threshold must be lowered to balance FS P* precomputing advantage.

ZKness: being its output random, RO acts like a public-coin Honest Verifier: if original proof is HVZK, its simulator can be employed to also forge a transcript for NIZK. Simulations often play with messages out of order, so the extra power to "**program**" **RO**'s outputs as wished (preserving uniformity to respect the prescribed distribution) is granted to S, to reverse the challenge's dependency on commitment. The same idea applies to Knowledge Extractability.

Random Oracle by pseudocode

```
output ← RO(inputs) :=
{
   if permanent_array[inputs] not exist {
      permanent_array[inputs] := new random value
   }
   output ← permanent_array[inputs]
}
```

Heuristic side: implementations use convenient (so not ideal) **Hash functions** as ROs. The security of this choice is commonly accepted but really still matter of research.

Properties' "scope" recap	Completeness	Soundness	Zero-Knowledgeness ⊆	Honest-Verifier Zero-Knowledgeness
	regards the Prover/Verifier couple, both acting honestly (aka following the protocol prescribed by the proof): it holds for **(P,V)**	is the property of the honest Verifier not being fooled by ANY strategy of a Prover pretending a false statement: for **(∀P*,V)**	is the prescribed Prover capability to not leak knowledge to ANY Verifier (another merit of "blind" black-box access to V by S): for **(P,∀V*)**	Typically an exposed Prover has to be leak-resistant against any adversarial Verifier strategy, and this is a weak form of ZKness holding by definition only for the HONEST Verifier, so for **(P,V)**. Nevertheless it's relevant because it sometimes implies IP for the same statement but with the stronger ZKness flavors: • HV **S**ZK → HV **S**ZK for Arthur-Merlin IPs → **S**ZK for Arthur-Merlin IPs • HV **C**ZK for Arthur-Merlin IPs → **C**ZK for Arthur-Merlin IPs

Sources and much more

- The Princeton Companion to Mathematics – Princeton University Press - Timothy Gowers & others (section IV.20 "Computational Complexity")
- Foundation of Cryptography – Cambridge University Press - Oded Goldreich (Volume I, chapters 1 and 4; all companion web pages stuff; errata)
- Blog posts by Matthew Green (here and here – BTW, my first meet with ZKPs), Jeremy Kun (here, here and here) and Yannik Goldgräbe (on Medium)
- Many Q&A on crypto.stackexchange.com, especially answers by Yehuda Lindell and Geoffroy Couteau (some organized per-topic on his web pages)
- Tutorials on the Foundations of Cryptography – Springer - Yehuda Lindell & others (chapter 6 "How to Simulate It")
- Geoffroy Couteau's PhD thesis containing a very affordable overview of the field in introductory chapters 2 and 3
- A Survey of Noninteractive Zero Knowledge Proof System and Its Applications – Hindawi - Huixin Wu & Feng Wang
- The 9th Bar-Ilan University (BIU) Winter School on Cryptography – February 18-21, 2019 (lectures' slides and videos)