ZKP

ZERO KNOWLEDGE PROOFS

Slides written by William T. Doan

DEFINITION (INFORMAL)

A proof is said to be of zero-knowledge if it can successfully convince a verifier that a statement is true without revealing any other information beyond the fact that the statement is true. It must be:

(1) Complete
(2) Sound
(3) Zero-knowledge

Example: Where's Waldo?

Let there be a man named Waldo amongst a sea of other passersby. A photographer takes a picture of the scene with Waldo in the frame. The question here is simple: where's Waldo? More crucially, how can we prove to a verifier that we know where Waldo is without exposing his location to the verifier?

DEFINITION (FORMAL)

Let there be Turing machines *P*, *V*, and *S*. An interactive proof system with (*P*, *V*) for a language *L* is zero-knowledge if for any probabilistic polynomial time verifier (PPT) *V*^, there exists a PPT simulator *S* such that

$$\forall x \in L, z \in \{0,1\}^*, \text{View}_{\hat{V}}\left[P(x) \leftrightarrow \hat{V}(x,z)\right] = S(x,z)$$

where View_{V^} is a record of the inteeractions between *P*(*x*) and *V*(*x*, *z*)--the prover and the verifier respectively. That means the transcript and V^'s internal randomness, its "view", is indistinguishable from the output of *S* given the same public input and auxiliary information *z*,

Cont'd

P is the prover. It is often idealised as unbounded and it seeks to prove that $x \in L$.

The verifier, V^, is the probabilisitic polynomial time machine that may act maliciousy.

We have a simulator S that is a PPT machine that must reproduce the verifier's view without interacting with P.

The trio P, $V^$ and S are cast as Turing machines and the language L as the mathematical predicate under scrutiny; the discourse assumes the accustomed finitude of the verifier's computational powers and the plenitude of the prover's.

PPT Machines

A probabilistic polynomial-time (PPT) machine is an abstract computational model used to represent an efficient randomized algorithm. It is a probabilistic Turing machine (or equivalent RAM model) that, besides deterministic computation, may read random bits (its "coin flips") during execution. Its running time on any input of length n is bounded by some polynomial $p(n)$; i.e., it halts within $p(n)$ steps for all inputs. It outputs a random variable.

PPT Machines in Cybersecurity

In cryptographic definitions, a PPT machine models any feasible (efficient) adversary or verifier: anything that runs in polynomial time and may use randomness. Security claims are measured against all PPT machines.

Cont'd

Probabilities that are negligible (smaller than 1/q(n) for every polynomial q and sufficiently large n) are treated as effectively zero in this model; likewise, two distributions are considered indistinguishable if no PPT machine can tell them apart with non-negligible advantage.

The Power of the Simulator

The very essence of zero-knowledge is encapsulated in the simulator's potency: if S can forge the verifier's entire experience absent any communion with P, then the verifier demonstrably acquires naught from the prover save the bare fact of membership, and any attempt at clandestine extraction of knowledge is thereby rendered vacuous.

Cont'd

Thus, when we require a simulator S to be a PPT machine, we mean S must efficiently (in polynomial time) and probabilistically produce outputs whose distribution cannot be told apart from the real verifier's view by any other efficient randomized algorithm.

COMMON PROTOCOL PATTERNS

Zero-knowledge proofs are cryptographic protocols whereby a prover convinces a verifier of the truth of a statement without disclosing any information beyond that truth; they formalize privacy-preserving verification within the interactive proof model, employ probabilistic polynomial-time verifiers and simulators to capture feasible adversaries, and underpin a wide array of practical constructions—ranging from simple sigma-protocols to modern succinct arguments—used to achieve authentication, privacy, and verifiable computation in contemporary systems.

SIGMA-PROTOCOLS

Sigma-protocols are three-move interactive proof schemes (commitment— challenge— response) by which a prover convinces a verifier of knowledge of a witness for a statement while revealing nothing beyond the validity of the statement; they furnish simple, composable proofs of knowledge with honest-verifier zero-knowledge and efficient rewinding-based simulators.

(CONT'D)

In a Sigma-Protocol, the prover sends a commitment that hides the witness; the verifier replies with a random challenge; the prover responds with an answer that ties the commitment to the challenge. Zero-knowledge is achieved because a simulator can, by choosing the challenge first and then forging a consistent commitment and response (or by rewinding the verifier), produce an indistinguishable transcript without knowing the witness. The protocol's structure (commitment–challenge–response) and the randomness of the challenge prevent the verifier from learning anything beyond that the prover knows a valid witness.

The Fiat-Shamir Heuristic

The Fiat–Shamir heuristic is a method to convert an interactive sigma-protocol into a non-interactive proof by replacing the verifier's random challenge with the output of a cryptographic hash applied to the prover's commitment (and public input), thereby yielding a single-message proof in the random-oracle or practical-hash model while trading interaction for a heuristic trust in the hash behaviour.

(CONT'D)

The interactive challenge of a sigma-protocol is replaced by the output of a hash function applied to the commitment and public input, producing a non-interactive proof. In the random-oracle model the hash acts like a verifier's random challenge, so a simulator can program the oracle (or simulate its outputs) to produce transcripts indistinguishable from real ones, preserving zero-knowledge; in practice, security relies on modeling the hash as ideal and on how the simulator can simulate or program that oracle.

SNARKs and STARKs

SNARKs and STARKs are families of succinct, non-interactive proofs that allow a prover to convince a verifier of the correctness of large computations with very short proofs and fast verification; SNARKs (Succinct Non-interactive ARguments of Knowledge) typically rely on cryptographic assumptions and may require a trusted setup, whereas STARKs (Scalable Transparent ARguments of Knowledge) attain transparency and post-quantum resilience by relying on publicly verifiable randomness and stronger algebraic/time-efficient encodings at the cost of larger proof sizes or different performance tradeoffs.

(CONT'D): SNARKS

The prover encodes the computation and witness as algebraic objects (e.g., arithmetic circuits or R1CS), and produces a short proof that these encodings satisfy required relations. Zero-knowledge is provided by hiding the witness via randomness injected into the proof (e.g., blinding factors, polynomial masking, or randomized commitments) so that the verifier learns only that the relation holds, not the witness. Many SNARK constructions rely on cryptographic assumptions (pairings, knowledge-of-exponent) and may require a trusted setup; the simulator's ability to produce indistinguishable proofs depends on those setup parameters and assumed hardness.

(CONT'D): STARKS

Like SNARKs, STARKs transform computations into polynomial/trace checks and use probabilistic proofs (via PCP/FRI style arithmetization) to produce succinct, efficiently verifiable proofs. Zero-knowledge is attained by masking or randomizing witness-related polynomials and by using commitment schemes and low-degree tests that reveal only that the masked polynomials satisfy relations. STARKs emphasize transparency (no trusted setup) and use hash-based randomness; their simulator argument uses the public randomness plus witness-masking to produce transcripts indistinguishable from real proofs under the relevant soundness and binding properties.

Common Protocols: KEY

In all these families, zero-knowledge requires explicit witness-hiding mechanisms (commitments, blinding, masking) and a simulator argument showing transcripts can be produced without the witness.

CONCLUSION

Zero-knowledge proofs are principally instantiated by the three protocols we have just described. Properly applied, zero-knowledge furnishes powerful privacy and integrity guarantees across authentication, confidential ledgers, verifiable computation, and secure multiparty protocols; yet these benefits hinge upon rigorous design, careful parameter choices, and secure engineering.