

Zero Knowledge Proofs : A Survey

Gaurav Jain

Abstract

A zero knowledge interactive proof system allows one person to convince another person of some fact without revealing the information about the proof. In particular, it does not enable the verifier to later convince anyone else that the prover has a proof of the theorem or even merely that the theorem is true (much less that he himself has a proof). This paper reviews the field of zero knowledge proof systems giving a brief overview of zero knowledge proof systems and the state of current research in this field.

1 Introduction

Is it possible to prove a statement without yielding anything beyond its validity? Zero-knowledge protocols, as their name says, are cryptographic protocols which do not reveal the information or secret itself during the protocol, or to any eavesdropper. These proofs have some very interesting properties, and as the secret itself is not transferred to the verifying party, they cannot try to masquerade as you to any third party.

1.1 What constitutes a proof ?

Goldreich argued in [9] that the notion of a proof in a zero knowledge protocol is somewhat different from concept of a proof in a strict mathematical sense and more similar to a dynamical way of understanding the proof by interaction and interpretation used by humans. Mathematical proofs are more strict; they have a static and formal nature as they are either self-evident or are obtained from earlier rules. However, humans tend to use a more intuitive sense of proofs where the soundness of a statement is established through the process.

In a similar sense, in a zero knowledge proof protocol, instead of presenting a static proof for the claim, the prover tries to convince the verifier by interactively convincing him of the proof. This dynamic interpretation (at least in a weak sense) is essential to the nontriviality of the notion of a zeroknowledge proof.

2 Zero Knowledge Protocol Basics

2.1 The Parties in Zero Knowledge

We'll come across these people very often in zero-knowledge protocols:

- *Peggy, the Prover* - Peggy has some information that she wants to prove to Victor, but she doesn't want to tell the secret itself to Victor.
- *Vic, the Verifier* - Vic asks Peggy a series of questions, trying to find out if Peggy really knows the secret or not. Vic does not learn anything of the secret itself, even if he would cheat or not adhere to the protocol.

2.2 Features of Zero-knowledge Protocols

Zero-knowledge protocols can be described as cryptographic protocols having the following special features summarized in [2] :

- ***The verifier can not learn anything from the protocol*** The Verifier does not learn anything from the protocol, that he could not learn all by himself, without the prover. This is the central zero-knowledge concept, i.e. No knowledge (zero amount of knowledge) is transferred. Without this feature, the protocol would be called a minimum-disclosure protocol, i.e. zero-knowledge protocols require that absolutely no information can be leaked in any case.
- ***The prover can not cheat the verifier*** If Peggy doesn't know the secret, she can only succeed with a great amount of good luck. After several rounds of the protocol, the odds of an impostor passing as legitimate can be made as low as necessary. The protocols are also cut and choose, i.e. the first time the prover fails, Vic knows Peggy is not legitimate. So, with each round of the protocol, the certainty gets better and better. The protocols can be made to work even if the odds of a guess passing are high, you just need more rounds in the protocol. In other words, **validity** captures the verifier ability of protecting itself from being convinced of false statements (no matter what the prover does in order to fool it).
- ***The verifier can't cheat the prover*** Victor can't get any information out of the protocol, even if he does not follow the protocol. The only thing Victor can do is to convince himself that Peggy knows the secret. The Prover will always reveal only one solution of many to any one problem, never all of them which would allow finding out the secret itself.
- ***The verifier can not pretend to be the prover to any third party*** Because no information can leak from Peggy to Victor, Victor can't try to masquerade as Peggy to any outside third party. With some of these protocols, a man-in-the-middle attack is possible, though, meaning that someone can relay the traffic from the true Peggy and try to convince another Victor that he, the perpetrator, is Peggy. Also, if the verifier records

the conversation between him and the prover, that recording can't be used to convince any third party. It looks the same as a faked conversation (e.g. where the verifier and prover agreed beforehand which requests the verifier will choose).

- ***Vic could be convinced of any true statement*** This property is called **completeness** and it essentially captures the ability of some prover to convince the verifier of true statements (belonging to some predetermined set of true statements).

Suppose, Peggy tries to convince Vic that she knows a secret such as the satisfiability of a boolean formula ϕ . She could have done this by sending the satisfying truth assignment for ϕ to Vic. Instead, GMR [8][6] present a probabilistic scenario where Vic is convinced that Peggy indeed has a satisfying assignment for ϕ by exchanging multiple messages with him and asserting that she knows the secret. Because of the validity and completeness property of the zero knowledge proofs, if such a truth assignment does exist, then Peggy can behave in such a way that Vic almost always accepts; but if such a assignment does not exist, then no matter what Peggy does, Vic will certainly reject.

A different notion of provability was independently proposed by Babai[3] called the Arthur-Merlin protocol. It was more limited than the GMR model because in his model, the verifier was required to reveal all the random bits right after the protocol finishes.

3 Interactive Proof Systems

3.1 Basic Protocol

Peggy and Vic are both considered as probabilistic algorithms[11]. They both will perform private computations, and each of them has a private random number generator. Initially, Peggy and Vic both possess an input x . The object of the interactive proof is to convince Vic that x has some specified property.

The interactive proof, which is a challenge-and-response protocol, consists of a specified number of rounds. During each round, Peggy and Vic alternatively do the following:

1. receive a message from the other party
2. perform a private computation
3. send a message to the other party

A typical round consists of a challenge by Vic, and a response by Peggy. At the end of the proof, Vic either accepts or rejects, depending on whether or not Peggy successfully replies to Vic's challenges.

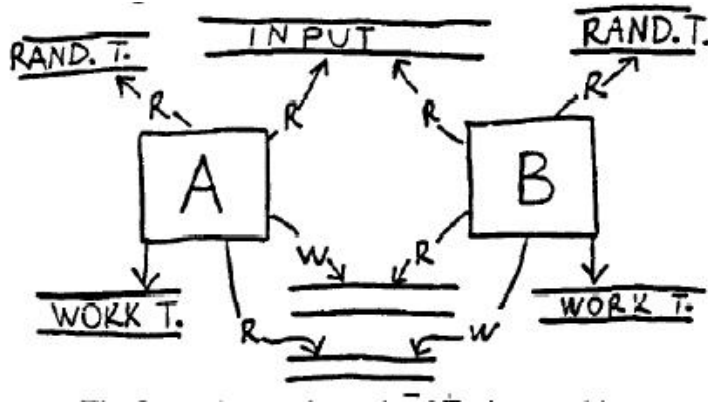


Figure 1: Interactive Turing Machine

3.2 Interactive Turing Machines (ITM)

An Interactive Turing Machine [7] is a six-tape deterministic Turing machine with a read-only input tape, a read-only random tape, a read/write work tape, a read-only communication tape, a write-only communication tape, and a write-only output tape. The string that appears on the input tape is called the input. The (infinite) contents of the random tape can be thought of as the outcomes of an infinite sequence of unbiased coin tosses. The string that appears on the output tape when the machine halts is called the output. The contents of the write-only communication tape can be thought of as messages sent by the machine; while the contents of the read-only communication tape can be thought of as messages received by the machine.

The complexity of an interactive Turing machine is measured in terms of its input (i.e., contents of input tape). Saying, for example, that the interactive Turing machine M is polynomial-time means that there exists a polynomial Q such that the number of steps M performs on input x is at most $Q(|x|)$, no matter what are the contents of its random tape and read-only communication tape. When discussing the expected number of steps of an ITM, the expectation is taken over the contents of the random tape only. (We do not count, of course, the steps made by the interactive machine with which A_1 interacts as defined below.)

3.3 Joint Computation of two ITMs

- Two interactive machines (Fig.1) are said to be linked if they have opposite identities, their inputtapes coincide, their switchtapes coincide, and the readonly communication tape of one machine coincides with the writeonly communicationtape of the other machine, and vice versa. The other tapes of both machines (i.e., the randomtape, the worktape, and the outputtape) are distinct.

- The joint computation of a linked pair of ITMs, on a common input x , is a sequence of pairs. Each pair consists of the local configuration of each of the machines. In each such pair of local configurations, one machine (not necessarily the same one) is active while the other machine is idle.
- If one machine halts while the switchtape still holds its identity the we say that both machines have halted.

3.4 Complexity of an interactive machine

An interactive machine A has time complexity $t : \mathbf{N} \rightarrow \mathbf{N}$ if for every interactive machine B and every string x , it holds that when interacting with machine B , on common input x , machine A always (i.e., regardless of the contents of its randomtape and B 's randomtape) halts within $t(|x|)$ steps.

3.5 Interactive Proof System

Every language in \mathcal{NP} has an interactive proof system [4, 1]. Specifically, let $\mathcal{L} \in \mathcal{NP}$ and let R_L be a witness relation associated with the language L (i.e., R_L is recognizable in polynomialtime and L equals the set $\{x : \exists y \text{ s.t. } |y| = \text{poly}(x) \wedge (x,y) \in R_L\}$). Then, an interactive proof for the language L consists of a prover that on input $x \in L$ sends a witness y (as above), and a verifier that upon receiving y (on common input x) outputs 1 if $|y| = \text{poly}(|x|)$ and $(x,y) \in R_L$ (and 0 otherwise).

Clearly, when interacting with the prescribed prover, this verifier will always accept inputs in the language. On the other hand, no matter what a cheating "prover" does, this verifier will never accept inputs not in the language. In this proof system both parties are deterministic (i.e., make no use of their randomtape). It is easy to see that only languages in \mathcal{NP} have interactive proof systems in which both parties are deterministic.

In other words, \mathcal{NP} can be viewed as a class of interactive proof systems in which the interaction is unidirectional (i.e., from the prover to the verifier) and the verifier is deterministic (and never errs). In general interactive proofs, both restrictions are waived: the interaction is bidirectional and the verifier is probabilistic (and may err with some small probability). Both bidirectional interaction and randomization seem essential to the power of interactive proof systems.

Moreover, Brassard et. al. [5] proved that perfect zero knowledge proofs can even be given in a bounded number of rounds.

3.6 Types of Zero Knowledge Proof Systems

Let (A,B) be an interactive protocol. Let T , the transcript be a random variable denoting the verifier view during the protocol on input x . Namely, for fixed sequence of coin tosses for A and B , T is the sequences of messages exchanged between verifier and prover, in addition to the string of coin tosses that the

verifier used. The string h denotes any private input that the verifier may have with the only restriction that its length is bounded by a polynomial in the length of the common input. (T is distributed over both A's and B's coin tosses).

- (A,B) is *perfect zero-knowledge* for L if there exists a probabilistic, polynomial time Turing machine M s.t. $\forall x \in L$, for all $a > 0$, for all strings h such that $|h| < |x|^a$, the random variable $M(x,h)$ and T are identically distributed. ($M(x, h)$ is distributed over the coin tosses of M on inputs x and h).
- (A,B) is *statistically zero-knowledge* for L if there exists a probabilistic polynomial time Turing machine M s.t. $\forall x \in L$, for all $a > 0$, for all strings h such that $|h| < |x|^a$,

$$\sum_{\alpha} |\text{prob}(M(x, h) = \alpha) - \text{prob}(T = \alpha)| < \frac{1}{|x|^c}$$

for all constants $c > 0$ and sufficiently large $|x|$.

Intuitively the way to think of statistically zero-knowledge protocols, is that an *infinite power examiner* who is given only polynomially large samples of $\{M(x,h) \mid M\text{'s coin tosses}\}$ and $\{T \mid A\text{'s and B's coin tosses}\}$ can't tell the two sets apart.

- (A,B) is *computationally zero-knowledge* for L if \exists probabilistic, polynomial time Turing machine M s.t. \forall polynomial size circuit families $C = C_{|x|}$, \forall constants $a, d > 0$, for all sufficiently large $|x|$ s.t. $x \in L$, and for all strings h such that $|h| < |x|^a$,

$$\begin{aligned} & \text{prob}(C_{|x|}(\alpha) = 1 \mid \alpha \text{ random in } M(x,h)) \\ & - \text{prob}(C_{|x|}(\alpha) = 1 \mid \alpha \text{ random in } T) < \frac{1}{|x|^d} \end{aligned}$$

A protocol (A,B) is computationally zero-knowledge if a *probabilistic polynomial time bounded examiner* given a polynomial number of samples from the above sets can not tell them apart.

4 Zero Knowledge Proof Examples

4.1 Ali Baba's Cave

Consider the example of a circular variety of Ali Baba's cave (Fig.2), with a secret door that can be opened by a password [10]. Peggy knows the password of the door, and wants to convince Vic that she knows it, but doesn't want Victor to know the password itself.

They work as follows:

- Peggy goes into a random branch of the cave, which Victor doesn't know standing outside the cave.
- Vic comes into the cave, and calls out a random branch of the cave (left or right), where Peggy should come out.

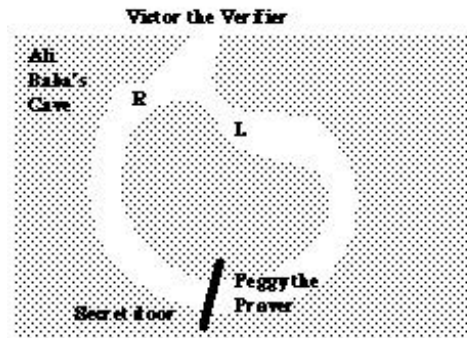


Figure 2: Ali Baba's Cave

- If Peggy knows the secret password, she can come out the right way every time, opening and passing the secret door with the password if necessary. If Peggy doesn't know the password, she has a 50% chance of initially going into the wrong branch, and as she is not able to pass the secret door, Victor can call her bluff.

They repeat this as many times as needed to convince Victor. If Victor will be happy with a 1 in 1024 chance of Peggy not knowing the password, they need 10 repetitions ($2^{10} = 1024$). This example also demonstrates another feature of zero-knowledge protocols : Now Victor is convinced that Peggy knows the secret password, but he cannot convince anyone else himself, as he doesn't know the secret!

Let's say that Victor would videotape the operation. But that recording can't be used to convince anyone else, as it looks just the same as a faked videotape, where a mischievous verifier and the prover agreed in advance which passage the prover should come out each time. So, Victor can't even convince others, just himself, about Peggy knowing the secret. Absolutely no information flowed to Victor in the protocol.

4.2 Rubik's Cube

A more complex protocols can be illustrated by a non-computer example about solving Rubik's cube. Suppose that Peggy knows how to solve the cube and wants to convince Victor about her skill, but doesn't want to show Victor how to actually solve the cube. Victor shows Peggy a scrambled Rubik's cube and asks for a proof of her skill to solve it. Peggy shows Victor a new, scrambled Rubik's cube. Victor can then choose to ask either

- for Peggy to show how to move from the new scrambled position to the original scrambled position, or
- for Peggy to show how to move from the new scrambled position to the solved position.

If Peggy knows how to go from the original position to the new scrambled position to the solution, she can answer either of these requests. If she does not know how to solve the cube, she could choose to be able to present one of the two choices that Victor might ask for, but not both.

Note that knowing both parts of the solution means that you can solve the cube, i.e. you know the secret. This is why Peggy must always present a differently scrambled cube in each round, because if Victor gets both answers to a single problem, he gains the knowledge of how to solve the cube in that case.

Both the problems, Ali Baba's cave and Rubik's Cube are problems which give *Proof of Knowledge* once they are proved by the prover. On the other hand, there also exist *Proof of Identity* problems whose protocol ensures that no third party could masquerade as Peggy or Vic.

4.3 Feige-Fiat-Shamir Proof of Identity

- *Precalculation*: An arbitrator generates a random modulus n (512-1024 bits) which is the product of two large primes. The arbitrator generates a public and private key pair for Peggy, by choosing a number, v , which is a quadratic residue mod n (i.e. $x^2 \equiv v \pmod{n}$ has a solution, and $v^{-1} \pmod{n}$ exists). This number, v , is the public key. The private key is then the smallest s for which $s = \sqrt{\frac{1}{v}} \pmod{n}$.
- The identification protocol proceeds then as follows: Peggy picks a random number r where $r \perp n$. She then computes $x = r^2 \pmod{n}$ and sends it to Victor.
- Victor sends Peggy a random bit, b .
- If the bit is 0, Peggy sends Victor r . If the bit is 1, she sends $y = r * s \pmod{n}$.
- If the bit was 0, Victor verifies that $x \equiv r^2 \pmod{n}$, proving that Peggy knows \sqrt{x} . If the bit was 1, he verifies that $x \equiv y^2 * v \pmod{n}$, proving that Peggy knows $\sqrt{\frac{x}{v}}$.

If an impersonator tries to pass for Peggy, she can either pick r such that she can reply if Victor sends a 0 or 1 bit, but she cannot prepare for both cases, so she will be found out with a 50% chance each round. Victor can't try to masquerade as Peggy to another verifier, as the bit Victor randomly sent to Peggy earlier has only a 50% chance of being the same as the second verifier will ask for.

In this protocol, Peggy should never reuse an r . If she did, Victor could send her the other random bit, and collect a set of both responses. Then, if he had enough of these, he could try to impersonate Peggy to an outsider.

This protocol can be implemented in a parallel fashion, making the public and private keys be a set of quadratic residues mod n , etc. Then we can do as many rounds in parallel as you have keys in the set, speeding up the protocol (but with larger memory requirements) and needing fewer messages.

5 Is NP ϵ Zero Knowledge Proof

It was shown in [7, 9] that if there exist (non-uniform) polynomial-time indistinguishable encryption scheme then every NP language has a computational zero-knowledge interactive proof-system.

The proof outline is to show a zero-knowledge proof system for an NP-complete language, graph three colorability (G3C). Suppose the prover wish to convince the verifier that a certain input graph is three-colorable, without revealing to the verifier the coloring that the prover knows. The prover can do so in a sequence of $|E|^2$ stages, each of which goes as follows.

- The prover switches the three colors at random (e.g., switching all red nodes to blue, all blue nodes to yellow, and all yellow nodes to red).
- The prover encrypts the color of each node, using a different probabilistic encryption scheme for each node, and show the verifier all these encryptions, together with the correspondence indicating which ciphertext goes with which vertex.
- The verifier selects an edge of the graph at random.
- The prover reveals the decryptions of the colors of the two nodes that are incident to this edge by revealing the corresponding decryption keys.
- The verifier confirms that the decryptions are proper, and that the two endpoints of the edge are colored with two different but legal colors.

If the graph is indeed three-colorable (and the prover know the coloring), then the verifier will never detect any edge being incorrectly labelled. However, if the graph is not three-colorable, then there is a chance of at least $\frac{1}{|E|}$ on each stage that the prover will be caught trying to fool the verifier. The chance that the prover could fool the verifier for $|E|^2$ stages without being caught is exponentially small.

in the case that the graph is three-colorable, the history of our communications consists of the concatenation of the messages sent during each stage. It is possible to prove (on the assumption that secure encryption is possible) that the probability distribution defined over these histories by our set of possible interactions is indistinguishable in polynomial time from a distribution that the verifier can create on these histories by itself, without the provers participation. This fact means that the verifier gains zero (additional) knowledge from the protocol, other than the fact that the graph is three-colorable.

The proof that graph three-colorability has such a zero-knowledge interactive proof system can be used to prove that every language in NP has such a zero-knowledge proof system.

References

- [1] Paul van Oorschot Alfred Menezes and Scot Vanstone. Handbook of applied cryptography, 1996.
- [2] H. Aronsson. Zero knowledge protocols and small systems.
- [3] L Babai. Trading group theory for randomness. In *ACM Symposium on Theory of Computing*, pages 421–429, 1985.
- [4] Michael Ben-Or, Oded Goldreich, Shafi Goldwasser, Johan Håstad, Joe Kilian, Silvio Micali, and Phillip Rogaway. Everything provable is provable in zero-knowledge. In S. Goldwasser, editor, *Advances in Cryptology—CRYPTO '88*, volume 403 of *Lecture Notes in Computer Science*, pages 37–56. Springer-Verlag, 1990, 21–25 August 1988.
- [5] Gilles Brassard, Claude Crepeau, and Moti Yung. Everything in NP can be argued in perfect zero-knowledge in a bounded number of rounds (extended abstract). In *Theory and Application of Cryptographic Techniques*, pages 192–195, 1989.
- [6] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to prove all NP statements in zero-knowledge and a methodology of cryptographic protocol design (extended abstract). In A. M. Odlyzko, editor, *Advances in Cryptology—CRYPTO '86*, volume 263 of *Lecture Notes in Computer Science*, pages 171–185. Springer-Verlag, 1987, 11–15 August 1986.
- [7] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(3):691–729, 1991.
- [8] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *ACM Symposium on Theory of Computing*, pages 291–304, 1985.
- [9] Oded Goldreich. Foundations of Cryptography (Fragments of a Book). Weizmann institute of science, 1995.
- [10] Jean-Jacques Quisquater, Myriam Quisquater, Muriel Quisquater, Michaël Quisquater, Louis Guillou, Marie Annick Guillou, Gaïd Guillou, Anna Guillou, Gwenolé Guillou, Soazig Guillou, and Tom Berson. How to explain zero-knowledge protocols to your children. In G. Brassard, editor, *Advances in Cryptology—CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 628–631. Springer-Verlag, 1990, 20–24 August 1989.
- [11] D. Stinson. Cryptography: Theory and practice, 1994.