A Primer on Zero Knowledge Protocols

Gerardo I. Simari
gis@cs.uns.edu.ar

Departamento de Ciencias e Ingeniería de la Computación
Licenciatura en Ciencas de la Computación
Universidad Nacional del Sur

Abstract

It is a common weakness among traditional communication protocols to be vulnerable to impersonation attacks. Every time this sort of protocol is executed, the system degradates because of the threat of an eavesdropper listening in on the communication. Zero Knowledge Protocols, presented by Goldwasser, Micali, and Rackoff, are an improvement on these situations. The objective is to obtain a system in which it is possible for a prover to convince a verifier of his knowledge of a certain secret without disclosing any information except the validity of his claim. This article will cover the basics of zero knowledge systems, explaining the main properties and characteristics. A series of examples, in growing level of difficulty, will also be presented, to see the main areas of application of such protocols.

1 Introduction and Motivation

Traditional protocols for the identification of parties in a transaction suffer from flaws that are inherent to the process used to achieve the objective. In simple password protocols, a claimant A gives his password to a verifier B. If certain precautions are not taken, an eavesdropper can get hold of the password that was transfered, and from there on he can impersonate A to his liking. Other protocols try to improve on this, as in the case of challenge-response systems. In this sort of protocols, A responds to B's challenge to prove knowledge of a shared secret. Of course, the challenge is changed every time the protocol is used; therefore, an eavesdropper can, in time, gather enough partial information about the shared secret to try an impersonation attack like the one described above.

In this article, we will discuss Zero Knowledge Protocols (abbreviated ZKP from here on), which are designed to defeat the disadvantages described above. In ZKP, a *prover* will try to demonstrate knowledge of a certain secret to a *verifier*. The main idea is to allow the proof to take place without revealing any information whatsoever about the proof itself, except of course for the fact that it is indeed a valid one. Zero Knowledge Proofs can be compared to an answer obtained from a trusted oracle.

It must be noted that the concept of *proof* in ZKP is different from the traditional mathematical concept. Mathematical proofs are strict, using either self evident statements or statements obtained from proofs established beforehand. ZK proofs

are more similar to the dynamic process used by humans to establish the truth of a statement throughout the exchange of information. Furthermore, in a ZKP, instead of presenting a static proof for a statement, the prover involves the verifier in a process in which he tries to convince the verifier of the truth of the statement interactively.

1.1 ZKP Parties

As hinted earlier, there are two parties involved in ZKP:

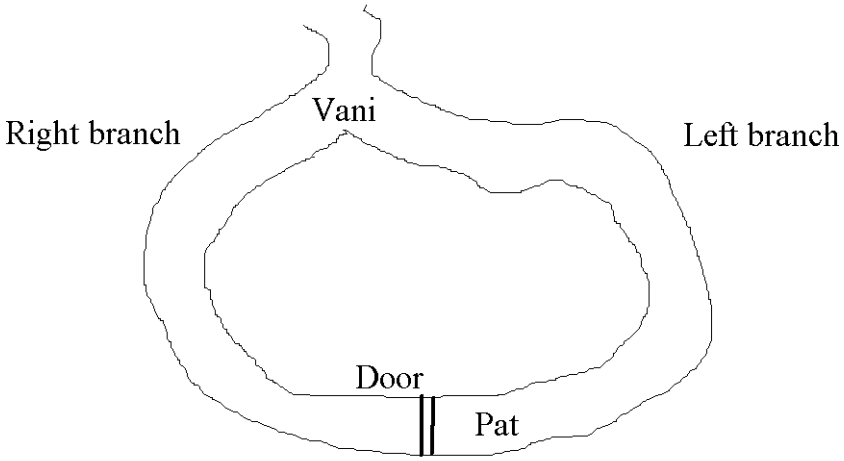
- *Pat, the prover*: Pat wishes to convey a proof of certain knowledge to Vani, but it is not in his wish to let Vani in on his secret.
- *Vani, the verifier*: Vani asks Pat a series of questions, which help her decide if Pat really knows what he claims to know. Vani cannot learn anything from this interaction, even if she were to cheat, or engage in activities outside the protocol itself. We will return to this aspect of ZKPs later on.

1.2 A simple example

Before going on with the presentation, we will present a classic example of ZKP. There exists a series of examples that adequately express the main features of ZKP, some of which can be found in (Koblitz, 1994; Menezes *et al.*, 1996; Schneier, 1995).

Consider, for the sake of example, a cave consisting of a circular tunnel. Diametrically opposite to the entrance of this cave, there is a door which can only be opened by password. Although this situation is probably not a real life scenario, it is quite useful in the display of the basic properties of ZKP. Now Pat knows the password to this door, and he wants to prove this to Vani without actually dislcosing it to her. They set off to complete the task as follows:

- Pat goes into a random branch of the cave (that is, left or right). He does this without Vani knowing which branch he chose.
- Standing at the entrance to the cave, Vani calls out a random branch (again, either left or right), where she wants Pat to come out from.
- If Pat is not lying about his knowledge of the secret password, he can obey Vani every time, using the door if necessary. If he doesn't know the password, he has a 50% of initially fooling Vani.



Now, the key to this protocol lies in the repetition of the above steps. If Vani is happy with a 1 in 256 chance that Pat is cheating, they will repeat the steps eight times. This can be proved using basic probability theory, noting that one series of steps is completely independent from another series.

This example also demonstrates another feature of ZKP; Vani is now (probabilistically) convinced that Pat can indeed open the door to his desire, but he cannot convince anybody else, since she doesn't know the secret. Suppose Vani were to videotape the whole process. Such a recording is useless, because to an outsider it appears identical to a forged videotape, where Pat and Vani agreed in advance about the sequence of chosen branches. Absolutely no information flowed to Vani throughout the execution of the protocol, except for the fact that Pat knows the password.

1.3 Features

Zero Knowledge Protocols have the following properties, some of which were already described above:

- ***The verifier cannot learn anything from the protocol***. The verifier does not learn anything in the process of the proof that he could derive from public information by himself. This is the central concept of zero knowledge, i.e., zero amount of knowledge is transferred. There are similar protocols, called *Minimum Disclosure Protols*, which relax this property trying to maintain the flow of information to a mininum.
- ***The prover cannot cheat the verifier***. If Pat doesn't know the secret, he can only fool Vani with an incredible amount of luck. The odds that an

impostor can cheat the verifier can be made as low as necesary by increasing the number of rounds executed in the protocol.

- ***The verifier cannot cheat the prover***. Vani can't get any information out of the protocol, even if she doesn't stick to the rules. The only thing Vani can do is decide when she accepts that Pat actually knows the secret. The prover will always reveal one solution of many; by doing this he insures that the secret remains intact. This point will become more clear after the presentation of some more complicated systems below.
- ***The verifier cannot pretend to be the prover to a third party***. As stated earlier, no information flows from Pat to Vani. This precludes Vani from trying to masquerade as Pat to a third party. Nevertheless, some ZKP protocols are vulnerable to man-in-the-middle attacks, in which an eavesdropper relays traffic to achieve the desired impersonation effect. OF course, other cryptographic services (outside the scope of this article) can be used to prevent this type of attack. See (Schneier, 1995; Menezes *et al.*, 1996) for more on this topic.
  Like we said above, a recording of the execution of the protocol is worthless in convincing a third party. Such a recording is identical to a faked one, in which Pat and Vani agreed on the steps beforehand.

In our effort to obtain a definition of Zero Knowledge Protocol, we will discuss a few properties that must be satisfied in order to ensure the behavior promised for such systems. These properties include *soundness* and *completeness*, in the context of Interactive Proof Systems.

2 Interactive Proof Systems

Zero Knowledge Protocols are instances of *Interactive Proof Systems*, wherein a prover and a verifier exchange challenges and responses, typically dependent on random numbers (ideally, the outcomes of fair coin tosses) which they are allowed to keep secret. Like we said above, the proofs in this context are *probabilistic* rather than absolute as in the mathematical sense. These proofs need only be correct with a certain bounded probability (although this probability can be made arbitrarily close to 1). Interactive proofs are sometimes called *proofs by protocol*.

Interactive proofs used for identification may be formulated as proofs of knowledge. P has a secret s, and he wishes to convince V that he has *knowledge* of s by responding correctly to queries (such queries involve publicly known inputs and agreed upon functions) that require knowledge of s to answer. It is worth noting that it is quite different to prove *knowledge* of s than it is to prove *existence* of s. For example, proving that a certain x is a quadratic residue modulo n differs from proving knowledge of the square root of x modulo n.

An interactive proof is said to be a *proof of knowledge* if it has the properties of *soundness* and *completeness*. These properties are defined next, as in (Menezes *et al.*, 1996).

Definition *2.1* (*Completeness Property*)
An interactive proof protocol is *complete* if, given an honest prover and an honest verifier, the protocol succeeds with overwhelming probability (i.e., the verifier accepts the prover's claim). The definition of *overwhelming*, of course, depends on the application, but generally implies that the probability of failure is not of practical significance.

Definition *2.2* (*Soundness Property*)
An interactive proof protocol is *sound* if there exists an expected polynomial time algorithm M with the following property: if a dishonest prover (impersonating P) can with non-negligible probability execute the protocol with V, then M can be used to extract from this prover the knowledge (essentially equivalent to P's secret) which with overwhelming probability allows subsequent protocol executions.

Since any party capable of impersonating P must in fact have knowledge equivalent to the secret itself, the soundness property guarantees that the protocol is in fact providing a proof of knowledge (in order to succeed, you must count on knowledge equivalent to the secret). This property therefore prevents a dishonest prover from succeeding. A standard method used to prove that a certain protocol is sound is to assume the existence of a dishonest prover who is capable of successfully executing the protocol, and show how this allows to compute the secret in polynomial time.

This "proof of knowledge" idea is the foundation of zero knowledge proofs. However, it is clear that neither of these properties say anything about zero knowledge itself. A ZKP should, in addition, have the property that no amount of knowledge should pass between the prover and the verifier that the verifier couldn't figure out without the help of the prover. This property is simply called the *zero knowledge property*, and will be defined after introducing the concept of *simulator*, as discussed below.

2.1 Simulators

Lets consider our cave example once again. Suppose Vani sends a videotape of the sequence of steps taken with her proof with Pat to her friend, Vincenza. We will call such a tape a *view* (or transcript) of the proof session. Vincenza accuses Vani of faking the tape, and it is clear that she can do nothing to convince her otherwise. Vani posseses no unforgeable, non-repudiable proof that Pat indeed knows the secret password to the door. All she can do is ask Pat to demonstrate it once again for Vincenza, who will pick her own sequence of challenges for Pat (which assures randomness). If there is a way to forge a proof that is indistinguishable from a genuine one (as in the case of the vidotape), we say that there is a *simulator* for the proof in question.

Definition *2.3* (*Simulator*)
A *simulator* is a method or procedure that generates fake (generated without the prover) views that are indistinguishable from a genuine (generated with the prover) view of a proof (Mikucki, 1999).

The concept of simulator is key to the definition of the zero knowledge property mentioned above.

Definition *2.4* (*Zero Knowledge Property*)
A proof of knowledge has the *zero knowedge property* if there exists a simulator for the proof (Mikucki, 1999).

This formalizes what has been said in previous sections. In the context of a ZKP, the verifier does not obtain further information about the secret other than its validity. Furthermore, a much desired property of this type of protocols is that the number of times that the prover participates in them does not vary the chances of success of impersonation attacks (as it might in password or challenge-response protocols). The Zero Knowledge Property allows us to arrive to the definition of Zero Knowledge Proof as follows.

Definition *2.5* (*Zero Knowledge Proof*)
A zero knowledge proof is a proof of knowledge that also has the zero knowledge property (Mikucki, 1999).

2.2 Extensions to the definition of Zero Knowledge Proof

A few extensions to the basic definition of ZKP have been studied. Some of the most important of these extensions are defined next.

Definition *2.6* (*Perfect Zero Knowledge*)
A protocol is said to be *perfect zero knowledge* if real and simulated transcripts are completely indistinguishable from one another.

Definition *2.7* (*Computational Zero Knowledge*)
A protocol is *computationally zero knowledge* if an observer restricted to probabilistic polynomial time tests cannot distinguish real from simulated transcripts.

Definition *2.8* (*Statistical Zero Knowledge*)
A protocol is *statistical zero knowledge* if there is a negligible difference between the probability distributions of real and simulated transcripts.

It is clear that computational zero knowledge is a relaxation of the basic concept. It is still quite useful, though, because in practice the "enemy" is usually considered in possession of polynomial time capability to perform attacks on our systems. Other extensions have been defined, such as *non-interactive* zero knowledge protocols, in which the prover need not be present in order to convince the verifier of his knowledge. These extensions will not be covered in this article. See (Goldreich, 2001) for a treatment on this subject.

Other types of proofs that are not actually zero knowledge are *"no use" zero knowledge*, and *minimum disclosure* proofs. What varies in these proofs is the amount (and type) of information that is allowed to flow from prover to verifier in the execution of the protocol.

2.3 Remarks and observations

It is clear that the zero knowledge property and the soundness property have no say in the level of security that a system presents. It is of key importance to the security of a given protocol for it to depend on computationally difficult problems. No proofs exist for the most commonly used problems (e.g., integer factorization, knapsack problem, discrete logarithm, etc.), so the security of the systems that use them are directly dependent on future developments in the field of Computational Complexity. This type of system is commonly referred to as *provably secure*.

A few points can be made in the difference between zero knowledge and public key (PK) techniques. These are:

- *No degradation with usage:* Repeated use of a ZK protocol does not present degradation. ZK protocols are also resistant to chosen text attacks. This leads a ZK protocol which *is not* provably secure to be considered against a PKP which *is* provably secure.
- *Efficiency:* ZK protocols are usually less efficient than PK protocols. This is an important factor to consider in certain application environments where (hard or soft) real time computations are to be ensured.
- *Unproven assumptions:* Most ZK and PK protocols depend on the same assumptions (quadratic residuosity, factoring, discrete log, etc).

3 NP $\in$ ZKP

This section includes a brief introduction to the computational complexity classes P and NP. This is necesary in understanding the result that every problem in NP has a zero knowledge proof associated with it. The proof of this result will be outlined at the end of the section by means of a simple protocol for proving knowledge of a solution to an NP-complete problem.

3.1 Introduction to NP-completeness

There are many problems for which there is no known efficient algorithm. Classic examples of these problems include satisfying a boolean formula, the travelling salesman problem, the knapsack problem, etc. It is not yet known if efficient solutions to these problems even *exist*. Maybe we don't have the tools to build the solutions; maybe the solutions don't exist and we don't have the tools to prove that they don't.

One of the most amazing results in computational complexity theory is that an efficient algorithm to solve any of the problems listed above would automatically provide us with efficient solutions for their whole class.

At the heart of this theory lies the idea that there may be problems which are genuinely hard to solve, but the validity of a proposed solution is easily tested. For example, consider the Hamiltonian Cycle Problem: this problem is believed to be hard, but it is very easy to verify if a sequence of nodes is a hamiltonian cycle. For the purpose of the definitions below (extracted from (Brassard & Bratly, 1996)), we

will focus on decision problems, that is, those problems that have a binary (yes/no) answer.

Definition *3.1* (*The Class P*)
P is the class of decision problems that can be solved by a polynomial-time algorithm.

Definition *3.2* (*The Class NP*)
NP is the class of decision problems X that admit a proof system $F \subseteq X \times Q$ s.t. there exists a polynomial p(n) and a polynomial-time algorithm A such that:

- $\forall\ x \in X\ \exists q \in Q$ s.t. $(x, q) \in F$ and moreover, the size of q is at most $p(n)$, where n is the size of x.
- For all pairs (x, q), algorithm A can verify whether or not $(x, q) \in F$.

The next definition is concerned with the transformation of solutions of one problem in solutions to other problems. This is called *reduction*, and if the transformation can be done in polynomial time, then we are talking about *polynomial reduction*.

Definition *3.3* (*Polynomial Reduction*)
Let A and B be two problems. We say that A is polynomially Turing reducible to B if there exists an algorithm for solving A in a time that would be polynomial if we could solve arbitrary instances of problem B at unit cost. This is denoted $A \leq_T^p B$ (or simply $A \leq B$).

Now, the concept of reducibility is central in reaching the notions necesary to comprehend the result that we want to show. The notion of NP-complete problem is the last definition we need to establish this section's main result.

Definition *3.4* (*NP-complete problem*)
A decision problem X is NP-complete if

- $X \in NP$; and
- $Y \leq_T^p X$ for every problem $Y \in NP$.

It was shown in (Goldreich *et al.*, 1991; Goldreich, 1995) that if there exists a non-uniform polynomial time encryption scheme then every NP language has a computational zero knowledge interactive proof system. This can essentially be proven by showing a zero knowledge proof system for any NP-complete language, such as graph *three colorability* (G3C). Since any problem in NP can be reduced to G3C (from the above definition), the proof below ensures us that any language in NP has associated with it a zero knowledge proof system.

3.2 G3C ∈ ZKP

Suppose the prover wishes to convince a verifier that he knows a three coloring for a certain graph, without revealing such coloring. The prover could do this in a sequence of $|E^2|$ stages (where E is the set of edges), each of which involves the following steps, as shown in (Jain, n.d.; Koblitz, 1994):

- Pat permutes the three colors at random. This allows him to conceal the true coloring throughout the repetition of the steps.
- Pat "hides" the coloring from Vani (perhaps using encryption schemes).
- Vani selects an edge of the graph at random.
- Pat reveals the colors of the two nodes for which the selected edge is incident.
- Vani confirms that the two colors are valid, i.e., different.

If everything is in order (that is, the graph is three-colorable and Pat knows a coloring), then Vani can never choose an edge that is not labeled correctly. If the graph is not three-colorable (or Pat is lying about his claim), then there is a $\frac{1}{|E|}$ chance on each stage that Pat will be caught in his attempt to fool Vani. This chance is exponentially small after $|E^2|$ stages.

The history of the communication between Pat and Vani is the concatenation of the messages that were exchanged. Based on the assumption that secure encryption is possible, it is possible to prove that the probability distribution defined over these valid histories is indistinguishable in polynomial time from a distribution that Vani could create by herself without the help of Pat. This allows us to conclude that Vani gains zero additional knowledge from the execution of the above protocol, other than the fact that the graph is three-colorable. The fact that the distributions are indistinguishable in polynomial time means (as defined above) that the system is computationally zero knowledge.

This outlines the proof that every language in NP has a zero knowledge proof system.

4 Applications

In this section we will see actual ZK systems at work. One of the main applications of ZKP is in obtaining the cryptographic objective of *authentication*. ZKPs can be a good solution to security problems in financial or other security critical applications, where systems such as smart cards are not secure enough. Smart cards are vulnerable to reverse engineering to extract vital information. A zero knowledge proof system can be implemented to withstand such attacks. Furthermore, as we have seen, ZKPs can be adapted in various degrees of relaxation to fit the application.

In this section we will examine two examples: the first is related to authentication, while the second is a system that can be used to prove (in the context of ZKP) knowledge of graph isomorphisms.

4.1 Feige-Fiat-Shamir Proof of Identity

One of the classic authentication zero knowledge schemes is the Feige-Fiat-Shamir scheme (Feige *et al.*, 1987). The version presented below (adapted from (Jain, n.d.)) is a basic one, and would not be applied in practice for reasons of efficiency. Nevertheless, the modifications introduced to make it efficient are not relevant to the details that are of interest to us. The main difference with the systems actually

used are the amount of challenges issued per round; in order to obtain higher performance rates, a greater level of parallelism is required.

As before, the objective here is for Pat to prove to Vani his identity by demonstrating his knowledge of a certain secret s. This secret s is associated with Pat through authentified public data, as we shall see below. The security of the Fiege-Fiat-Shamir system rests on the supposed difficulty of extracting square roots modulo large composite integers of unknown factorization.

Feige-Fiat-Shamir proof of identity:

- Precalculation (setup): An arbitrator (a *trusted*, independent entity) generates a random number n to be used as modulus. This modulus, a product of two large primes, will be in practice a 512-1024 bit number. The arbitrator then generates a public and private key pair for Pat. He does this by choosing a number v, which is a quadratic residue modulo n (i.e., $x^2 \equiv v \pmod{n}$ has a solution, and $v^{-1} \bmod n$ exists). The public key will then be v, and the private key is the smallest s for which $s = \sqrt{\frac{1}{v}} \bmod n$.
- The two parties then proceed: Pat picks a random number r s.t. $\gcd(r, n) = 1$. He then computes $x = r^2 \bmod n$ and sends it to Vani.
- Vani sends a random bit, b, to Pat.
- If $b = 0$, Pat sends Vani r. If $b = 1$, he sends her $y = r * s \bmod n$.
- If $b = 0$, Vani verifies that $x \equiv r^2 \bmod n$, proving that Pat knows $\sqrt{x} = r$. If the bit was 1, she verifies that $x \equiv y^2 * v \bmod n$, proving that Pat knows $\sqrt{\frac{x}{v}}$.

This protocol ensures that if an impostor is trying to pose as Pat, he can prepare (in his choice of r) to answer one (and only one) of the challenges. This gives the impostor a 50% chance per round of succeding in his masquerade. Like in the case of the cave discussed above, Vani can challenge Pat any number of times until she is convinced that he knows the secret s.

It is worth noting that Pat should always choose a new r in each round. If he did not do this, Vani could gather (by manipulating the "random" bits) enough pairs of responses to try and mount an impersonation attack on the protocol with a third party. Another point that should be noted is that this protocol does in fact reveal a bit of information: in the case that the answer is $y = r * s$, this is supporting evidence that v is indeed a square modulo n, and because this is a sound protocol, after a certain amount of iterations we can conclude that this is true.

4.2 Graph Isomorphism

The classic problem of Graph Isomorphism in mathematics is basically the question: Given two graphs G_1 and G_2, is there a bijection between their sets of nodes that preserves edges? In other words, can we rename G_1's nodes and arrive at G_2?

Following the general lines presented so far, we will be interested in showing a proof system that allows us to convince a verifier that we know a certain isomor-

phism without revealing it. Finally, we will look into an algorithm that produces forged transcripts of our proofs (Mikucki, 1999).

Suppose we have two graphs: G_1 and G_2. Both sets of nodes are labeled from 1 to 5. Suppose Pat knows the secret permutation $\sigma = \{5, 4, 3, 2, 1\}$ that yields the isomorphism, that is, σ maps G_1's nodes {5,4,3,2,1} into G_2's nodes $\{1, 2, 3, 4, 5\}$. The protocol works as follows:

1. Pat selects a random permutation π of the set $\{1, \cdots, 5\}$. He obtains a new graph, H, that is the image of G_1 under π, and sends it to Vani.
2. Vani sends a random integer $i \in \{1, 2\}$ to Pat.
3. Pat computes a permutation ρ of the set $\{1, \cdots, 5\}$ such that H is the image of G_i under ρ. If $i = 1$, Pat uses $\rho = \pi$. If $i = 2$, Pat uses $\rho = \sigma \circ \pi$, where σ is the fixed permutation such that $G_1 = \sigma(G_2)$ (that is, the secret isomorphism).
4. Vani checks that H is the image of G_i.

The protocol might proceed between Pat and Vani as follows:

- Round 1:
  1. Pat secretly selects $\pi = \{1, 4, 3, 2, 5\}$. He then sends $H = \pi \circ G_1 = \{1, 4, 3, 2, 5\}$ to Vani.
  2. Vani selects $i = 1$ and sends it to Pat.
  3. Pat sends Vani $\rho = \pi$.
  4. Vani verifies that $H = \{1, 4, 3, 2, 5\} = \rho \circ G_1 = \{1, 4, 3, 2, 5\}$.
- Round 2
  1. Pat secretly selects $\pi = \{2, 4, 1, 3, 5\}$. He sends $H = \rho \circ G_1 = \{2, 4, 1, 3, 5\}$ to Vani.
  2. Vani chooses $i = 2$ and sends it to Pat.
  3. Pat computes $\rho = \pi \circ \sigma = \{2, 4, 1, 3, 5\} \circ \{5, 4, 3, 2, 1\} = \{4, 2, 5, 3, 1\}$ and sends it to Vani.
  4. Vani verifies that $H = \{2, 4, 1, 3, 5\} = \rho \circ G_2 = \{4, 2, 5, 3, 1\} \circ \{1, 2, 3, 4, 5\} = \{4, 2, 5, 3, 1\}$

The rounds will continue until Vani is satisfied that the chance of a successful cheat is small enough. We can point out that when $i = 1$, Pat is showing that $H = \pi \circ G_1$, that is, H is isomorphic to G_1. When $i = 2$, Pat shows that $H = \pi \circ G_1$ is isomorphic to $(\pi \circ \sigma) \circ G_2$, i.e., a graph that is isomorphic to G_1 (H) is isomorphic to G_2.

Now we must show that this protocol is indeed zero knowledge. An earlier result established that if a simulator for a protocol exists, then that protocol satisfies the zero knowledge property. We will now see a forgery algorithm that generates false views of proofs that never took place. Such views shall be indistinguishable from real ones, if we want to guarantee the zero knowledge property.

Forgery Algorithm

1. $T_0 = (G_1, G_2)$
2. Randomly select $i_b \in \{1, 2\}$.
3. Create a random permutation $H_b = \rho_b \circ G_b$.
4. Add (H_b, i_b, ρ_b) to T (the transcript).
5. Repeat until the desired transcript length has been reached.

Following this algorithm, we can see that the transcripts that it generates are perfectly possible ones, with probabilities identical to the occurrence of a real transcript. This does not allow us to prove anything to a third party, however, since we haven't learned anything from the protocol. The chances of responding correctly to the challenges are very small, as we have seen in other examples. These facts allow us to conclude that this is indeed a zero knowledge proof system (Goldreich, 2001).

References

Brassard, Gilles, & Bratly, Paul. (1996). *Fundamentals of algorithmics*. Prentice Hall.

Cormen, Thomas H., Leiserson, Charles E., & Rivest, Ronald L. (1990). *Introduction to algorithms*. MIT Press.

Feige, U., Fiat, A., & Shamir, A. (1987). Zero knowledge proofs of identity. *Proceedings of the nineteenth anual acm symposium on theory of computing, new york, n.y.*, 210–217.

Goldreich, Oded. (1995). *Foundations of cryptography (fragments of a book)*. Weizmann Intitute of Science.

Goldreich, Oded. (2001). *Foundations of cryptography: Basic tools*. Cambridge University Press, Cambridge, England.

Goldreich, Oded, Micali, Silvio, & Wigderson, Avi. (1991). 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.

Goldwasser, S., Micali, S., & Rackoff, C. (1985). The knowledge complexity of interactive proof systems. *Proceedings of the seventeenth anual acm symposium on theory of computing, rhode island*, 291–304.

Jain, Gaurav. *Zero knowledge proofs: A survey*. Tech. rept. University of Pennsylvania, http://www.cis.upenn.edu/ jaing/papers/znp.pdf.

Koblitz, Neal. (1994). *A course in number theory and cryptography*. Springer.

Menezes, Alfred J., van Oorschot, Paul C., & Vanstone, Scott. A. (1996). *Handbook of applied cryptography*. CRC Press.

Mikucki, John J. (1999). *Zero knowledge proofs: A survey*. Tech. rept. Rochester Institute of Technology, http://www.cs.rit.edu/ jjm7570/crypto/ZKP.dvi.

Schneier, Bruce. (1995). *Applied cryptography*. John Wiley and Sons, Inc.