Worcester Polytechnic Institute

Introduction to Game Theory

MA 4891

Interactive Proof Systems

Author:	*Email:*
Michael Giancola	mjgiancola@wpi.edu
Nathan Hughes	nhhughes@wpi.edu

May 2, 2016

Contents

1 Introduction

This report expands upon the concepts of interactive and zero knowledge proof systems to try and provide a clear explanation of the material contained in [11] to a typical undergraduate student with no former knowledge of the subject. None of the material within this report was originally derived by the report's authors. All material was developed by explaining previous works in greater detail; we indicate the reference we used at the beginning of each expanded proof or definition.

This paper is mainly concerned with the following work:

Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on computing.* 18(1):186-208, 1989.

We also found the following textbook to be a good source for examples of interactive proofs:

Douglas R Stinson. *Cryptography: theory and practice.* CRC press, 1st edition. 1995.

Finally, we found several examples in various lectures from Cornell University. They are cited appropriately in the References section at the end of this report.

1.1 The Strange Cave of Ali Baba

A zero-knowledge proof is accomplished when an entity is able to prove a statement to some other entity without giving away any other knowledge besides the truth of said statement. For example, a simple method of proving that some graph is Hamiltonian is by exhibiting a Hamiltonian walk; but this gives additional information (namely, the walk) beyond simply whether the graph is Hamiltonian or not. In zero-knowledge proofs, our goal is to convince a verifier that something is true without giving the verifier the means of proving it themselves.

Quisquater et al. gave one of the simplest examples of a zero knowledge proof [18]. Consider the cave shown in Figure 1. There is a door between the two paths which requires knowledge of a secret phrase to open. Peggy can prove to Vic that she knows the secret phrase without telling Vic the phrase.[1]

To accomplish this, Peggy and Vic perform the following protocol. First, Peggy flips a coin; if the coin lands heads-up, she enters the Left path, and on tails she enters

[1]The names "Peggy" and "Vic" are used to represent the prover and verifier respectfully throughout this report; this notation is borrowed from [19]

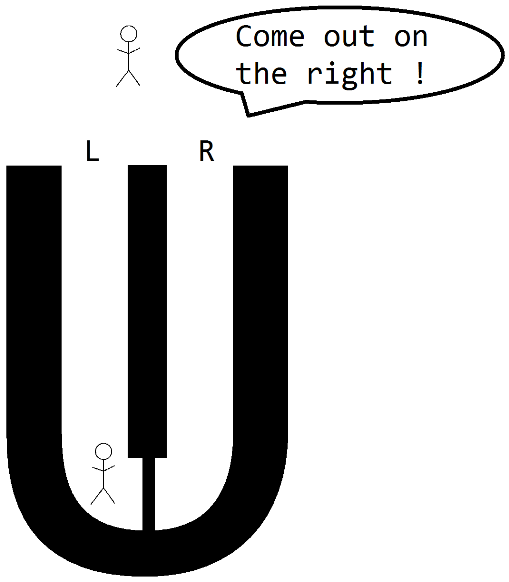

Figure 1: An illustration of Quisquater's (et al.) Zero Knowledge Proof for Children (Figure was created by Michael Giancola, inspired by [18])

the Right path. She then waits inside the cave. Next, Vic flips another coin; if it lands heads-up, he tells Peggy to come out the Left side, and on tails to exit the Right side. If Peggy happened to enter on the same side that Vic requests she leave out of, she doesn't need to know the secret phrase, and can simply exit. However, if Peggy and Vic repeat this game multiple times, there is a very high probability that Peggy will need to use the secret phrase to exit the correct path for Vic (ie. Peggy enters the Left path and Vic requests Right, or vice versa).

Thus, after sufficiently many trials, Vic will be convinced, with very high probability, that Peggy really knows the phrase. If she didn't, she would have to guess which side Vic is going to ask her to exit out of; the probability that she is able to do this successfully after say 50 trials is $\frac{1}{2^{50}}$, which is insignificant. We see a property of zero-knowledge proofs emerging from this simple example. That is, if Peggy knows the secret, Vic can be nearly guaranteed that she does, and if Peggy doesn't know the secret, Vic will almost never believe that she does. This is the essence of a zero-knowledge proof; a more mathematically rigorous definition will be given later.

1.2 Definitions

The following concepts are defined intuitively for now to provide a basis for our discussion of Zero-Knowledge Interactive Proofs. More mathematically rigorous definitions will be given later.

A *Turing Machine* is an abstract model for a computer which utilizes tape for storage of data. Tapes are divided into cells which can each contain a single symbol. For our purposes, we will assume a tape is either read-only or write-only [13].

An *Interactive Turing Machine* (ITM) is a Turing Machine with a read-only input tape, a write-only output tape, and a random input tape. The random tape is the only source of randomness in the system; the tape's output is equivalent to flipping a coin.

An *interactive proof system* consists of two entities, the prover and the verifier, and a statement to be proved. The verifier is able to ask the prover questions, and the prover answers to indicate their possession of some knowledge [11].

A *theorem-proving procedure* is a method of proving a statement to a verifier. Any theorem-proving procedure must have three attributes: first, it should be possible to prove a true theorem. Similarly, it must be impossible to prove a false theorem. Third, the proof should be efficient; in other words, regardless of the work the prover must do, it should be simple for the verifier to confirm the prover's results.

A *zero-knowledge interactive proof* is accomplished when the prover is able to assure the verifier, with high probability, that the prover possesses some knowledge, without giving up any additional information.

Two random variables are *indistinguishable* if to an observer, a sequence of values is equally likely to have been generated from either of the random variables.

A random sequence of values is *approximable* if there is another (probabilistic) Turing Machine that can produce an indistinguishable sequence to the first easily (i.e. in a reasonable amount of time).

1.3 History

The original definition of a zero knowledge proof arose from the seminal work of Goldwasser, Micali and Rackoff [11] in 1989. Throughout the rest of our paper we explore the major concepts of this paper, but first we want to develop a picture of the related work that led to this seminal paper, and some of the work that has been pursued since then.

Goldwasser et al. developed their concept of an interactive proof system (and consequently, the concept of zero knowledge interactions and proofs) in parallel with several other competing ideas. The most prominent of these was the development of

Arthur-Merlin and Merlin-Arthur proof systems in Babai's seminal paper, [2]. The differences between these systems and the zero knowledge interactive proof system is discussed later. Both of these proof systems drew upon several other works in interactive identification systems such as [14], [7], [5], and [3].

The other major contribution of [11] is the definition of zero-knowledge (the first one being the concept of an interactive proof system). This concept of zero-knowledge is derived directly from previous work by Goldwasser and Micali [10] and Yao [21] in encryption and indistinguishability.

Since [11] was published, there have been several significant works that extend this notion of a zero knowledge proof. The first of these are [1] and [8] that explore several other examples of zero knowledge proofs besides quadratic residuosity (most notably graph isomorphism). In addition, [8] and other papers ([2] and [12]) make an effort to define the complexity class of problems associated with zero knowledge proofs. Finally, there has been work undertaken to modify zero knowledge proofs to make them more amendable to computer based implementations, such as [4], which presents a non-interactive version of zero-knowledge proofs.

2 Interactive Proof Systems

Much of the work in [11] is involved with the concept of interactive proof systems. What follows is a description of the structure of interactive proof systems, and a short discussion of the differences between the interactive systems proposed by [11] and the Arthur Merlin proof systems proposed by [2]. In addition, we briefly touch upon the complexity classes of these proof systems.

2.1 Pairs of Interactive Turing Machines

A pair of Interactive Turing Machines consists of two ITMs as defined above, except that they share one read-only input tape. Also, A can read from B's write-only tape, and vice versa. We will denote the write and read-only tapes of A and B the *communication tapes* between them.

A pair (A,B) of ITMs is ordered (i.e. the pair (B,A) is distinct from (A,B)). In their interaction, the machines take turns being active, and B goes first. While a machine is active, it can perform computations and read or write from its various tapes. During any turn of either machine, the active machine may terminate the interaction. While a pair of ITMs need not ever terminate, any pair of ITMs which is

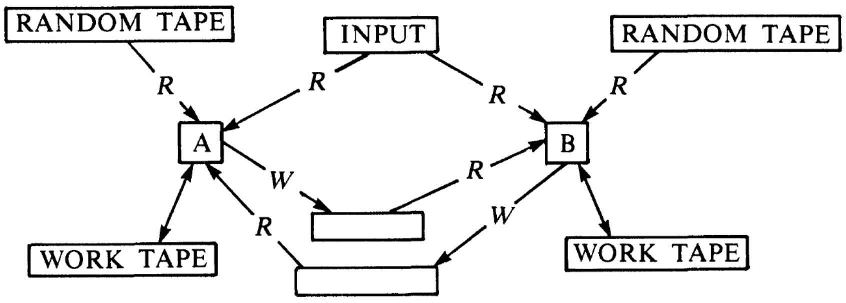

Figure 2: An interactive system between two Turing Machines A and B. — denotes a read/write head, R read-only, and W write-only. (Figure borrowed from [11])

performing an interactive proof must eventually terminate; thus these are the types of ITMs which we will discuss.

Consider an interaction between a pair of ITMs (A,B) on input x. A machine's ith message to the other machine is the string of symbols that the active machine write to its communication tape. We then define a_i to be A's ith message to B, and b_i to be B's ith message to A. Now we can define the *text of computation*, for a communication lasting n turns, to be $\{b_1, a_1, ..., b_n, a_n\}$ (a_n is empty if B terminates the interaction on its nth turn).

We denote (A,B)[x] to be the set of all possible texts between ITMs A and B on input x. Notice that the only element of the system that changes between multiple iterations of communication on the same input x is the random tape. Thus, the set (A,B)[x] forms a probability simplex.

2.2 Interactive Proof Systems

Before we introduce the definition of an interactive proof system, we need to briefly explain formal languages, as much of the work in [11] deals with formal languages. A language (L) (as defined by [13]) is a set of strings from a alphabet. An alphabet is a finite set of symbols (Σ) and a string created from an alphabet is a finite sequence of symbols that are members of such an alphabet. A string can also be denoted as an element of the set (Σ^*), the set of all arbitrary length strings formed by all possible combinations of the alphabet. Languages are typically defined as a subset of (Σ^*). [11] primarily focuses on languages that are subsets of $\{0, 1\}^*$, which correspond to arbitrary length binary representations of numbers.

Next we will use a pair of ITMs, along with certain constraints, to concretely define an Interactive Proof System. Let L$\subseteq$ $\{0.1\}^*$ be a language. Essentially, L is a binary language, containing only the symbols “0” and “1”, and all combinations of those symbols. Also, let (A,B) be a pair of ITMs.

(A,B) is an interactive proof system for L (where A is the prover, B is the verifier), if A has infinite power and B has limited power. In other words, A can perform NP-hard computations in a reasonable amount of time, and B is polynomial time. Additionally, the system must satisfy the following properties:

1. For any input $x \in L$ (with length n) given to (A,B), B accepts A’s proof with probability at least $1 - \frac{1}{n^k}$ for some constant $k > 0$ and sufficiently large n.
2. For any input $x \notin L$ given to (A,B), B accepts A’s proof with probability at most $\frac{1}{n^k}$ for some constant $k > 0$ and sufficiently large n.

Condition 1 essentially states that there must be an easy way to prove any true theorem, giving the verifier overwhelming confidence that the proof is sound. Similarly, Condition 2 states that, no matter what strategy a prover uses to prove a false theorem, the verifier will practically never believe the proof is sound.

2.3 Arthur Merlin Games and Complexity Classes

Much of the work of [11] is concerned with the concept of zero-knowledge: how little information can a prover give away during an interaction with the verifier and still convince the verifier that they know the proof conclusively. Another important class of interactive proofs are Arthur Merlin proofs. In the seminal paper on this topic, [2], Babai mentions several major proofs that are assumed to be correct, but have been hard to verify, such as the proof of the Four Color Theorem, which relies heavily on the the aide of the computer to prove the theorem for a large number of cases. To combat these harder to verify proofs, [2] introduces Arthur-Merlin and Merlin-Arthur interactive proof systems to verify such proofs. Rather than going through the proof exhaustively, Arthur (the verifier) asks Merlin (the prover) a series of challenges, after which Arthur is convinced with some high probability that Merlin does truly know the proof. The setup for these Arthur Merlin Games is exactly the same as the interactive proof protocol proposed by [11] that we described above, except for the fact that Arthur’s coin tosses are public (i.e. A can read from B’s random tape).

In his paper, Babai shows that Arthur-Merlin proof systems are a complexity class slightly larger than the NP complexity class [2]. In addition, this complexity classes was shown to be equivalent the Interactive Proof complexity class in [12] and [2].

3 Indistinguishability, Approximability and Zero Knowledge

An important foundation of any zero knowledge proof is the concept of *indistinguishability* and *approximability*. The authors of [11] use these two concepts to build up a formalization of zero knowledge. Intuitively, they show an interaction of interactive machines is zero knowledge if the interaction between the verifier and prover can be easily (in a reasonable amount of time) be created by a third party without the knowledge the prover possesses about the proof. We will now present the formal definitions taken from [11] of these three concepts: *indistinguishability*, *approximability*, and *zero knowledge*.

3.1 Indistinguishability

What follows is three formal definitions of indistinguishability from [11]. Each successive definition is more restrictive on the observer of the random variables (i.e. more permissive to differences in the random variable's distribution). These three definitions are perfect indistinguishability (the two random variable's distributions are equal), statistical indistinguishability (the two random variables have distributions that appear the same in polynomially bounded sample sizes) and computational indistinguishability (the two random variables appear the same under polynomially bounded computation on a polynomially bounded sample size).

We will first start with the notation of families of random variables, which [11] defines as $U = \{U(x)\}$, the set of members of $0, 1^*$ that result from the parameter x that is a member of some language L.

Expanded from [11], two families of random variables ($U = \{U(X)\}, V = \{V(x)\}$) are perfectly indistinguishable if

$$\forall \alpha \in \{0,1\}^* \, \mathbb{P}(U(x) = \alpha) = \mathbb{P}(Z(x) = \alpha) \tag{1}$$

To explain this conceptually, imagine you are given a random sample that could have come from one of two random number generators. If the two random number generators have the same distributions, no matter how long you spend looking at the sample you wouldn't ever be able to decide which number generator it came from. Specifically, if two families of random variables are indistinguishable on some input x, to a third party it is equally likely that for any output either one of the families could have generated that output.

Taken directly from [11], two families of random variables ($U = \{U(x)\}, V = \{V(x)\}$) are statistically indistinguishable if

$$\sum_{\alpha \in \{0,1\}^*} |\mathbb{P}(U(x) = \alpha) - \mathbb{P}(Z(x) = \alpha)| \leq |x|^{-c} \tag{2}$$

for any $c > 0$ and a sufficiently large x.

Intuitively, this definition is very similar to the last definition. For perfectly indistinguishability, if an observer was given any size sample, U and V would appear to be the same. The modification statistical indistinguishability makes is that if the observer is given any polynomially bounded sample size, they are equally likely that the observer will assign the sample to either one of the random variables. We can see this in the changes between Equations 1 and 2. In Equation 2, the difference between the distributions of random variables U and V is no longer zero, but instead must be in total (over all possible outputs α less that any polynomial function of the size of x, i.e. less than $|x|^c$.

Next follows the formal definition of computational indistinguishability. To formalize this definition, [11] introduces the concept of families of poly-size boolean circuits, denoted by $C = \{C(X)\}$. According to [11], a poly-size boolean circuit is a boolean circuit with a single output and at most $|x|^e$ for any e greater than zero. Colloquially, these circuits are large combinations of different boolean operations (and, or, not) on a given input X that produce a single boolean value, with the restriction that the number of operations is polynomially bounded with the size of x.

The authors of [11] use these families of circuits to define $\mathbb{P}(C, U, x)$ as the probability that the circuit C_x corresponding to a given input x will output 1 on any random sample resulting as an output from the random variable $U(x)$. Using this definition, we can now present the definition of computational indistinguishability.

Taken directly from [11], two families of random variables ($U = \{U(x)\}, V = \{V(x)\}$) are computationally indistinguishable if

$$|\mathbb{P}(U, C, x) - \mathbb{P}(V, C, x)| \leq |x|^c \tag{3}$$

for every poly-size family of circuits C, for all c and a sufficiently large x.

The only difference between this definition and Equation 2 is that the observer is now restricted to polynomial time computations on samples (which is mathematically done through the introduction of the poly-size families of boolean circuits).

3.2 Approximability

Now that we've defined indistinguishability, we can use this definition to formalize the concept of approximability. For the formal definition of approximability we again turn to [11]; what follows is heavily borrowed from their definition of approximability, with minor changes for clarity.

First, define M to be a probabilistic Turing Machine, and denote the output of M for a given input x as $M(x)$. Notice that M is a family of random variables. Some family of random variables U is said to be approximable if there exists a M perfectly indistinguishable from U.

The above definition can be adapted to statistical and computational approximability by substituting statistically indistinguishable and computationally indistinguishable where appropriate.

3.3 Zero Knowledge

The formal definition of zero knowledge comes directly from the definition of indistinguishability and approximability. However, there is a trick associated with how [11] sets up the definition of zero knowledge, relating to how they define a view of an interaction between two Turing Machines.

To define a view of the interaction between a prover and verifier during an interactive proof, [11] first considers an arbitrary B' as the verifier, where B' has some polynomially bounded extra history H (that can be past interactions with the prover or previous computation). The view of an interaction between the prover A and the verifies B' is defined by [11] to be random string ρ generated by the random tape of B' and the sequence of messages from A to B' and from B' to A for n turns of the interactive protocol. The random variable taking on these views is defined by [11] to be $\text{View}_{A,B'}(x, H)$ for some history tape H and input x.

Given this definition, [11] defines a interactive protocol with A as the prover to be perfectly zero knowledge for a language L if for every B', the family of random variables of $\text{View}_{A,B'}(x, H)$ is perfectly approximable on the new language L':

$$L' = \{(x, H) \mid x \in L \, |H| < |x|^c\} \tag{4}$$

for any c greater than 0.

Like approximability, this definition can be adjusted to computational zero-knowledge and statistical zero-knowledge by substituting in computationally approximable and statistically approximable in the appropriate places.

Colloquially, this definition says that an interactive protocol is zero-knowledge if the view of the interaction is easily approximated by some other random process; i.e. that the exchange between A and B can be simulated by a Turing Machine without doing intensive calculation.

4 Zero Knowledge Languages

[11] mentions several languages that can be used as a basis for a interactive zero knowledge proof systems, and describes two in detail: Quadratic Residuosity and Quadratic Non-Residuosity. We present the formal definitions of these two languages, as well as the formal definition of several other examples we found from [19].

4.1 Quadratic Residuosity and Non-Residuosity Languages

The seminal paper by Goldwasser, Micali and Rackoff [11] is primarily concerned about two specific languages: quadratic non-residue and quadratic residue. What follows is an interpretation of these languages according to [11], adapted to the notation of [19].

Let $\mathbb{N}$ be the natural numbers. Consider n, an element of the natural numbers $\mathbb{N}$. We can then define Z_n^* as the set of all numbers between zero and one that do not share any factors with n except for 1 as

$$Z_n^* = \{x \mid 1 \leq x < n, \gcd(x, n) = 1\} \tag{5}$$

Some element x of Z_n^* is defined to be quadratic mod residue if there exists some u element of the real numbers such that $u^2 = x \mod n$. We will discuss different properties and related facts of the quadratic residue later, but it is important to first define the Quadratic Residue (QR) language according to [11]:

$$QR = \{(n, x) \mid n \in \mathbb{N}, x \in Z_n^*, Q_n(x) = 0\} \tag{6}$$

where $Q_n(x)$ is the quadratic residue predicate, defined to be 0 when x is quadratic residue mod n.

The problem of determining whether some number is a quadratic residue is considered to be hard by [11], provided you do not know the prime factorization of that number. [11] contends that this is because factoring a given number is also considered to be hard.

4.2 Graph Isomorphism

Two graphs G_1 and G_2 (with vertex sets V_1 and V_2 respectively) are *isomorphic* if there exists a bijection $\pi : V_1 \to V_2$ such that whenever two vertices $i, j \in V_1$ are adjacent, $\pi(i)$ and $\pi(j)$ are adjacent [20, p.18]. Also, if $i, j \in V_1$ are *not* adjacent, then $\pi(i)$ and $\pi(j)$ are not adjacent.

It will be important later to note that graph isomorphism is a transitive property. That is, if graphs G_1 and G_2 are isomorphic, and graphs G_2 and G_3 are isomorphic, then G_1 and G_3 are isomorphic.

To verify this, let

$$\pi : G_1 \to G_2$$

$$\sigma : G_2 \to G_3$$

Since π and σ are bijections, we have

$$\sigma\pi : G_1 \to G_3$$

4.3 Map 3-Coloring

In general terms, a *coloring* of a graph G is a map from a vertex in G to a value in the set $\{1, 2, ..., n\}$ [20, p.76]. A *proper coloring* is a coloring in which all sets of adjacent vertices have different colors [20, p.76].

For this report, we will discuss 3-colorings of Graphs (so $n = 3$ in the above definition), which are assumed to be proper.

5 Proofs

We use this section to explain the proofs that the provided protocols for a zero-knowledge quadratic residuosity and graph isomorphism proof in [11].

5.1 Proposed Quadratic Residuosity Protocol

The following protocol we present is adapted from [11] to use wording closer to [19]. We assume a prover Peggy trying to convince Vic that she knows a quadratic residue for a given number. Both Peggy and Vic share x, the number Peggy is trying to convince Vic is quadratic residue, and n, the modulus being used.

For each round, the following interaction occurs:

- Peggy sends Vic the quadratic residue $y = v^2 \mod n$
- Vic sends Peggy $b \in \{0, 1\}$ using his random tape
- Peggy sends Vic $z = u^b v \mod n$ where $x = u^2 \mod n$
- Vic checks to see if $z^2 \equiv xy \mod n$

The protocol terminates after m rounds if Peggy provides correct responses each round, or after Peggy provides an incorrect response.

Before [11] proves that the protocol above is zero-knowledge, they first show that it is indeed an interactive proof system. What follows is an adaptation of their argument.

Theorem: (A, B) is an interactive proof system. (note that A refers to Peggy and B refers to Vic).

Proof: Remember that an interactive proof system for a system of interactive Turing Machines needs the following two properties:

- For any input x in the language L given to the proof system (A, B), B will accept A's proof with likelihood $1 - \frac{1}{|x|^k}$ for any $k > 0$
- For any input x not in the language L given to the proof system (A, B), B will accept A's proof with likelihood $\frac{1}{|x|^k}$ for any $k > 0$

The first case is relatively simple. If A has the proof that x is in fact quadratic residue, then at every challenge from B where $b = 0$, then by definition A will provide a pair (v, y) that will pass the verification test by B. If $b = 1$, then A will also provide the correct response by definition (A knows the a correct square root of x, u), yielding a likelihood of 1 that B will accept A over m rounds, fulfilling the first condition for an interactive proof system.

The second case is also relatively simple. Considering some A' that has a pair (u', x) where $u'^2 \neq x \mod n$, then the only way that A' can provide verifiable response to B when $b = 1$ is to also provide a y that is not actually a quadratic residue. This would then fail the other case, $b = 0$, which means that A' can pass one case, but not both. The probability that A' passes all these cases successfully is $\frac{1}{2^m}$ (as the coin flips of B are not known a priori to A'), fulfilling the second condition for an interactive proof system. From these two cases, [11] concludes that the above protocol is in fact an interactive proof system. □

5.2 Proof of Zero Knowledge for Quadratic Residuosity

What follows is an expansion of the proof provided by [11] that the interactive proof system defined in the last subsection is zero-knowledge. Note that we change some of the notation to reflect the examples in [19] and the later examples of this protocol that we provide.

Theorem: (A, B) is a perfectly zero-knowledge interactive proof system

Proof: We first start with the assumptions and definitions. Assume some arbitrary interactive Turing Machine B' that runs in polynomial time, and assume B' has some history H that is polynomially bounded is size with respect to the size of x. The common input to A and B' is (x, n), where A is trying to show that $\exists u^2 \equiv x \mod n$. The view between (A, B') is defined to be the sequence

$$R, Y_1, B_1, Z_1, Y_2, B_2, Z_2, \dots Y_m, B_m, Z_m \tag{7}$$

where R is value of the random tape of B', Y_i the random variable representing A's initial message to B at turn i, B_i is the random variable representing B's message to A at turn i and Z_i is the random variable representing A's response to B at turn i. The family of views seen by B' of these random variables is defined as $\text{View}_{A,B'}((x, n), H)$.

This family of random variables represents all the possible interactions that some verifier B' could have with A. Something important to note is that each of b_i isn't necessarily calculated via the proposed protocol, and instead we need to show that for any method of computing b_{i+1}, i.e. the next message B' sends to A, the protocol still remains zero-knowledge. Assume we have $(R, Y_1, B_1, Z_1, \dots Y_i, B_i, Z_i)$ denoted as V_i taking on specific values v_i. v_i represents a partial view of the interaction between

A and B', while V_i represent all the possible interactions that A and B' can have up to turn i. B' can then compute b_{i+1} through some function $f(x, n, H, v_i, y_{i+1})$, or that B' can only compute the next message based on the information it has seen thus far.

We want to show that this family of random variables $\text{View}_{A,B'}((x, n), H)$ is perfectly approximable on the language of Quadratic Residuosity, i.e. we must show that there exists some polynomially time probabilistic Turing Machine M that produces the exact same distribution for $\text{View}_{A,B'}((x, n), H)$ as A. Given some partial view v_i, the proposed Turing Machine M by [11] repeatedly computes the following values at round $i + 1$:

- b_{i+1} is chosen randomly from $\{0, 1\}$
- z_{i+1} is chosen randomly from Z_n^*
- y_{i+1} is chosen to be z_{i+1}^2 if $b_{i+1} = 0$, $Z_{i+1}^2 x^{-1}$ otherwise

This continues until the termination condition $b_{i+1} = f(x, n, H, v_i, y_{i+1})$ is reached; at this point the Turing Machine outputs $y_{i+1}, b_{i+1}, z_{i+1}$ as its simulated result of the interaction between A and B'.

There are two important facts about this protocol: M chooses numbers randomly less than n until it arrives at a v relatively prime to n, and M chooses $x^{-1} \in Z_n^*$ such that when x^{-1} is multiplied by $x \mod n$, the result is 1. The goal now is to show that M can produce the same distribution for turn $i + 1$ in the proof as (A, B').

Consider the view produced by M, defined as $R', Y_1', B_1', Z_1', Y_2', B_2', Z_2', \dots Y_m', B_m', Z_m'$, which should have the same distribution as $\text{View}_{A,B'}((x, n), H)$. We know that R' is the same as R by definition (the distribution of different random tapes for different Turing Machines is the same).

In addition, the transcript of interaction between A and B' should have the same distribution as the simulated transcript produced by M. To show this, we need to define V_i', the random variable capturing all partial views of the simulated interaction of M up to round i. $V_i' = (R', Y_1', B_1', Z_1', \dots Y_i', B_i', Z_i'$. We want to show that if V_i and V_i' are assumed to have the same distribution and both take on the same value v_i that $y_{i+1}, b_{i+1}, z_{i+1}$ has the same distribution for both V_i and V_i'.

The protocol of M is such that for round $i + 1$, b_{i+1}' is randomly selected until it takes on the value $f(x, n, H, v_i, y_{i+1}')$, i.e. that for any y_{i+1}' that M might generate, B_{i+1}' will have the same distribution as B_{i+1}, regardless of the distribution of Y_{i+1}'.

Along the same lines, Y_{i+1}' will also follow the same distribution as Y_{i+1}. If $b_{i+1}' = 0$ then y_{i+1}' will take on the value of a quadratic residue mod n (with the same distribution as Y_{i+1} taking on values of quadratic residues mod n). If $b'_{i+1} = 1$, then Y'_{i+1} will still also be a quadratic residue, following the same distribution as before.

Finally, Z'_{i+1} will have the same distribution as Z_{i+1}. If $b'_{i+1} = 0$, then z'_{i+1} will be a square root of y'_{i+1} (and therefore Z'_{i+1} has the same distribution as Z_{i+1}). If $b'_{i+1} = 1$, then $z'_{i=1}$ will again be a square root of y'_{i+1} (and therefore Z'_{i+1} has the same distribution as Z_{i+1}).

Notice that this holds for any previous view (even empty ones), so it holds that the distribution of V_{i+1} and V_{i+1} are identical for any i. Therefore the entire view V_i and V'_i have the exact same distributions (i.e. they are perfectly indistinguishable) and are perfectly approximable by M. This satisfies the definition of a perfectly zero-knowledge proof system, and therefore the protocol between A and B is a zero knowledge proof system. $\square$

Essentially, this Turing Machine M can "fake" knowing a square root of x at each time step by fabricating a y that makes it look like they passed each challenge. However, this fabricated y doesn't allow M to produce a verifiable response to a real challenge from B (as shown by the last proof). Because the protocol between A and B was shown to be an interactive proof system, and because we have just shown that the previously described M can produce the exact same distribution of interactions, the protocol between A and B is a zero-knowledge proof.

5.3 Proposed Graph Isomorphism Protocol

The following protocol we present is adapted from [9] to use wording closer to [19]. We assume a prover Peggy trying to convince Vic that she knows an isomorphism between two graphs. Both Peggy and Vic are aware of the graphs $G_1(V, E_1)$ and $G_2(V, E_2)$. Peggy also knows (but Vic doesn't) an isomorphism ϕ between $G_1(V, E_1)$ and $G_2(V, E_2)$.

For each round, the following interaction occurs:

- Peggy sends Vic a graph isomorphic G_1, H
- Vic sends Peggy $b \in \{0, 1\}$ using his random tape
- Peggy sends Vic π, the permutation from H to G_1 if $b = 0$ and $\pi\phi$ is $b = 1$
- Vic verifies the isomorphism sent from Peggy (ψ) is correct by computing $H = \psi G_1$ if $b = 1$ and $H = \psi G_2$ if $b = 0$

The protocol terminates after m rounds if Peggy provides correct responses each round, or after Peggy provides an incorrect response.

Unlike [11], [9] simply states that the above is a interactive proof system. We demonstrate later in our examples section (See Section 6.2) that an alternative, but similar protocol is also an interactive proof.

5.4 Proof of Zero Knowledge for Graph Isomorphism

What follows is an expansion and re-organization of the proof provided by [9] with notation similar to [11] that the interactive proof system of graph isomorphism defined in the last subsection is zero-knowledge. This proof follows a similar strategy as the earlier proof that the proposed quadratic residue system was also a zero knowledge interactive proof system, i.e. showing that a polynomial time probabilistic Turing Machine can simulate the interaction between (A, B') for any B' (where B' is defined the same as in the earlier section).

The proof offered by [9] that the above graph-isomorphism interactive proof depends on showing that their constructed Turing Machine M has two properties: it halts in expected polynomial time, and $\{M(G_1, G_2)\}$ is indistinguishable from $\text{View}_A, B'(G_1, G_2, H)$ (borrowing terminology from the last proof).

Before we provide an expansion of their proofs of the above two properties, we first need to introduce the constructed probabilistic Turing Machine M. The protocol of this Turing Machine is relatively simple: M privately interacts with B' for a large number of trials, throwing away interactions that where B' rejects M and building up a transcript of interaction with m turns where B' accepts the messages M sends. It is important to note that M builds up the sequence of random bits on the random tape of B' as it interacts with B', and gives B' that same random tape every subsequent time it interacts with B.

Formally, [9] defines the protocol for M as follows. M fixes the random tape of B' with some sequence of random bits (R) and uses this for every interaction between M and B'. On every turn i, M interacts with B' in the following manner:

- M generates β, the desired coin toss from B' and H, a random permutation of G_1 if $\beta = 1$ and G_2 if $\beta = 2$.
- M simulates B' with all available history up to this round i, i.e. $(G1, G2, R, \psi_1, H_1, \psi_2, H_2, \ldots \psi_{i-1} H_{i-1})$, and gets the response of B', b_i.
- If $b_i = \beta_i$, M saves the interaction of the current round. Otherwise M tries the first two steps again

Essentially, M cheats and creates a isomorphism only between the graph it constructs H and one of the two graphs (G_1 or G_2) by clever construction of H. If B'

requests the wrong isomorphism M keeps trying again until the right isomorphism is requested. While it isn't necessarily clear from the protocol, [9] shows that this generates the same distribution as the interaction between (A, B') and it terminates in expected polynomial time.

What follows is an expansion and clarification of the proof from [9] that the proposed M above terminates in expected polynomial time

Theorem: M terminates in expected polynomial time

Proof: At any round, the produce H and β by M is statistically independent. If this is the case, then the probability that $\beta_i = b_i$ (i.e. that the desired coin flip that M wants is the same as the coin flip produced by B' given every simulated interaction that occurred before. Therefore, there's a probability of $\frac{1}{2}$ that the desired coin flip of M and the output of B' at that time step line up, and therefore it is expected that M will take 2 rounds of simulation for every turn simulated between M and B'. Because B' is guaranteed to terminate in expected polynomial time, M is also guaranteed to terminate in expected polynomial time. $\square$

What follows is an expansion and clarification of the proof from [9] that the proposed M above produces interactions with the same distribution as some interactive pair (A, B').

Theorem: The probability distribution of $M(G_1, G_2)$ is that same as the distribution of $\text{View}_A, B'(G_1, G_2, H)$.

Proof: This is shown by induction, similar to the last proof. We can define two views: V_i, the interaction between A and B up to turn i (defined similarly to the proof that the quadratic residuosity protocol is zero knowledge) and V_i', the simulated interaction between A and B produced by M up until turn i. We assume that V_i and V_i' both produce some value v_i for this transcript of interaction. We want to show that the distribution of the interaction simulated by M for turn $i+1$ is the same as the distribution for interaction between A and B for turn $i+1$ is the same. Notice that the base case, an empty history trivially has the same distribution, so we only need to demonstrate that the inductive step holds.

For the inductive case, we need to show that the distribution for $(H'_{i+1}, b'_{i+1}, \psi'_{i+1})$ is the same as the distribution for $(H_{i+1}, b_{i+1}, \psi_{i+1})$. Notice that H'_{i+1} is chosen from the same distribution as H_{i+1}, so these distributions are the same. M is defined so that the simulated b'_{i+1} is chosen from the distribution of b_{i+1}, so these distributions must also be the same. Finally, there is a one-to-one mapping from H_{i+1} (and H'_{i+1}) to ψ'_{i+1}, i.e. there is only one possible isomorphic mapping from H_{i+1} to G_1 and G_2, so therefore the distribution of ψ'_{i+1} and ψ_{i+1} is the same. $\square$

Finally, we present the proof that this graph isomorphism protocol will be a zero knowledge proof from [9]

Theorem: The interactive protocol (A, B) for proving graph isomorphism is a zero-knowledge proof system.

Proof: We define a Turing Machine M as above that can approximate the view of the interaction between A and B' for any interactive expected polynomial time Turing Machine B'. As show above, M completes in expected polynomial time, and M has the same distribution as the view of A and B'. Therefore, the interaction (A, B') is perfectly approximable for any B', so the interactive protocol for A and B described above is a zero knowledge proof system by definition. $\square$

6 Examples

In the following subsections, we present several protocols and examples for zero knowledge proofs.

6.1 Graph Non-Isomorphism

Several sentences and ideas in this section are borrowed from [16].

Say Peggy and Vic receive graphs G_1 and G_2 as input. Peggy wants to prove to Vic that the two graphs are non-isomorphic without giving Vic any additional information, including the means of proving it himself.

First, Vic chooses a random value $i \in \{1, 2\}$ and a random bijection π where $H = \pi(G_i)$. If G_1 and G_2 are truly non-isomorphic, H is isomorphic to exactly one of them; precisely, H is isomorphic to G_i, which is dependent on Vic's random choice. Vic sends H to Peggy.

Next, Peggy finds j such that G_j is isomorphic to H, and sends j to Vic. Peggy is able to do this as we generally assume that the prover is of infinite power. Vic accepts the proof if $i = j$.

If G_1 and G_2 are non-isomorphic, it is always true that $i = j$. If it were possible that $i \neq j$, then both G_1 and G_2 would be isomorphic to H, meaning G_1 and G_2 are isomorphic to each other.

Thus if Peggy were trying to fool Vic, and G_1 and G_2 really were isomorphic, she would have to guess Vic's choice of i. The probability that she could do this successfully given n trials is $\frac{1}{2^n}$, which is probabilistically insignificant.

6.2 Graph Isomorphism

What follows is an alternative protocol for proving Graph Isomorphism which is presented in [15]. Several sentences and ideas in this section are borrowed from [15].

Say Peggy and Vic receive graphs G_1 and G_2 as input. Peggy knows a bijection $\pi : G_1 \to G_2$, proving that G_1 and G_2 are isomorphic. Peggy wants to prove to Vic that the graphs are isomorphic without revealing anything about the (secret) bijection π.

First, Peggy chooses a random value $i \in \{1, 2\}$ and a random bijection σ such that $H = \sigma(G_i)$. Peggy sends H to Vic.

Next, Vic chooses a random value $j \in \{1, 2\}$ and sends it to Peggy. Finally, Peggy sends Vic the bijection τ, where

$$\tau = \begin{cases} \sigma, & \text{if } i = j \\ \sigma\pi^{-1}, & \text{if } i = 1, j = 2 \\ \sigma\pi, & \text{if } i = 2, j = 1 \end{cases}$$

At this point, Vic accepts the proof if $H = \tau(G_j)$. It isn't trivial as to why this protocol works, so we will investigate each case individually.

First, say $i = j = 1$ (the case is identical if $i = j = 2$). Then

$$\pi : G_1 \to G_2$$

$$\sigma : G_1 \to H$$

Since $\tau = \sigma$,

$$\tau(G_j) = \tau(G_1) = \sigma(G_1) = H$$

Next, say $i = 1, j = 2$. Then

$$\pi : G_1 \to G_2$$

$$\sigma : G_1 \to H$$

Since $\tau = \sigma\pi^{-1}$,

$$\tau(G_j) = \tau(G_2) = \sigma\pi^{-1}(G_2) = \sigma(G_1) = H$$

Finally, say $i = 2, j = 1$. Then

$$\pi : G_1 \to G_2$$

$$\sigma : G_2 \to H$$

Since $\tau = \sigma\pi$,

$$\tau(G_j) = \tau(G_1) = \sigma\pi(G_1) = \sigma(G_2) = H$$

Notice that π must be a bijection for Vic to accept Peggy's proof, unless $i = j$. If Peggy were trying to fool Vic by guessing his choice of j, the probability that she is able to guess correctly in n consecutive trials is $\frac{1}{2^n}$.

Also notice that, using this protocol, it isn't important that Peggy's choice of H changes between rounds of the interaction. Even if Vic had all three possible τ's for a particular H, he still couldn't recreate π.

Another neat comment about this example is that it creates a path for proving other properties of graphs in zero knowledge. For example, if Peggy wanted to prove that G_1 or G_2 was Hamiltonian, she could exhibit a Hamiltonian walk in one of the Temporary Graphs.

6.3 Graph 3-Coloring

Several sentences and ideas in this section are borrowed from [17]

Say Peggy and Vic receive a graph G as input. Peggy knows a 3-coloring for G, $\sigma : G \to \{1, 2, 3\}$, and wants to prove that she truly does know a 3-coloring for G without giving Vic any information about it.

To start, Peggy chooses a permutation π of set $S = \{1, 2, 3\}$, which she "commits to". In this particular Zero Knowledge Proof, it is required that Vic trusts Peggy to not change her choice of π. So Peggy "committing to" π is to say that she won't change the permutation in an attempt to deceive Vic.

Vic chooses a random pair of vertices $(i, j) \in G$ with the condition that an edge directly joins the two vertices. Vic sends (i, j) to Peggy. Peggy then sends Vic $\pi\sigma(i)$ and $\pi\sigma(j)$, the permuted colors of vertices i and j.

Since vertices i and j are connected by an edge, Vic verifies that the colors Peggy sends back are indeed different; otherwise, Peggy didn't really possess a 3-coloring.

This example takes many more trials than the other examples for Vic to be assured that Peggy truly possesses a 3-coloring. Say Peggy has a 3-colored a graph *except* that one set of adjacent vertices share the same color. The only way for Vic to notice this is for him to request the correct pair (i,j); his probability for doing so is $\frac{1}{|E|}$, where $|E|$ is the number of edges in graph G.

To be sufficiently convinced of Peggy's proof, Peggy and Vic would have to perform $n|E|$ interactions, at which point the chance of Vic *not* noticing Peggy's deception is $(1 - \frac{1}{|E|})^{n|E|} \approx e^{-n}$, which is insignificant, so Vic is convinced.

6.4 Quadratic Residuosity

Next we present an example of a possible interaction between Peggy and Vic where Peggy proves that she knows a quadratic residue mod 14. The protocol used in this example was borrowed from [19]; the example itself was created by the report's authors.

N mod 14	0	**1**	2	3	4	**5**	6	7	8	**9**	10	**11**	12	13
N^2 mod 14	0	**1**	4	9	2	**11**	8	7	8	**11**	2	**5**	4	1
Relative Prime to 14?	N	**Y**	N	Y	N	**Y**	N	N	N	**Y**	N	**Y**	N	Y
Quadratic Residue?	N	**Y**	N	N	N	**Y**	N	N	N	**Y**	N	**Y**	N	N

For this example, say Peggy and Vic receive the following input:

- $n = 14$ (Peggy knows factorization $14 = 2 * 7$, Vic does not)[2]
- $x = 11$ (Chosen arbitrarily from the set of quadratic residues)

Peggy wants to prove that x is quadratic residue mod 14. $x = 11$ has two square roots mod 14; namely, 5 and 9. Arbitrarily, let's say that Peggy wants to prove

[2]In a real-world application of this protocol, n would be a product of two large primes, so it would not be trivial for Vic to determine the prime factorization of n.

$u = 5$ is a square root of 11 mod 14. Following the protocol described in Section 5.1:

- Peggy chooses a random $v \in Z_n^*$, computes $y = v^2$ mod n, and sends y to Vic

Notice $Z_n^* = \{1, 5, 9, 11\}$. Peggy randomly chooses $v = 5$, so she sends $y = 11$ to Vic.

- Vic chooses a random $i \in \{0, 1\}$ and sends it to Peggy

Vic randomly chooses $i = 0$, and sends it to Peggy.

- Peggy computes $z = u^i v$ mod n, where u is a square root of x, and sends z to Vic.

Peggy computes $z = 5^0 * 5 mod 14 = 5$, and sends it to Vic.

- Vic checks to see if $z^2 = x^i y$ mod n

$z^2 = 11^0 * 11$ mod n $= 11$ and $z^2 = 5^2$ mod $14 = 11$, so Vic accepts.

To perform another round, follow the same four steps above:

First, Peggy randomly chooses $v = 11$, and sends $y = 5$ to Vic.
Vic randomly chooses $i = 1$ and sends i to Peggy.
Peggy computes $z = 5^1 * 11$ mod $14 = 1$ mod $14 = 1$ and sends z to Vic.
Vic checks that $z^2 = 11^1 * 5$ mod 14, and $z^2 = 1$ so Vic accepts.

To verify the proof sufficiently (for the protocol given by [19]), Peggy and Vic would need to perform $log_2 n \approx 4$ rounds.

7 Further Work

After the writing of Goldwasser, Micali, and Rackoff's seminal paper on Interactive Proof Systems [11], several applications have emerged for zero-knowledge interactive proofs. Two of them are discussed below.

7.1 Identification Schemes

The field of cryptography is one of the most common for applications of zero-knowledge proofs. In [6], Fiat and Shamir discuss protocols for identification that allow users to identify themselves in zero-knowledge. In an identification scheme, this report is concerned with allowing Peggy to prove her identity to Vic without allowing Vic to prove to anyone else that he is Peggy. To develop their protocol, they rely on the zero-knowledge protocol for quadratic residuosity established in [11].

7.2 Non-Interactive Zero-Knowledge Proof Systems

The authors of [4] built upon the results of [11] to establish and prove a protocol for zero-knowledge proofs that don't require interaction between the prover and verifier. In particular, their set of constraints is as follows:

1. The prover can only talk and the verifier can only listen.
2. Instead of having separate, private random inputs, it is sufficient for the prover and verifier to share a random input.

[4] expands upon [11]'s protocol for proving quadratic residuosity in zero-knowledge, by doing so with the above, more strict constraints.

8 Conclusion

In this paper, we have discussed interactive proof systems in an effort to establish and prove protocols for zero-knowledge proof. The majority of the work discussed in this report is based on [11]; supporting materials were used to further the explanation provided by [11], which are cited appropriately. If time had allowed to expand on this report we would have also explored [2], the relationship between the Arthur Merlin and Interactive Proof complexity classes, and where Arthur Merlin fit into the complexity class hierarchy. Unfortunately, the scope of this report would not have allowed the fullness of discussion that this topic deserved.

References

[1] Lászío Babai and Endre Szemerédi. On the complexity of matrix group problems i. In *Foundations of Computer Science, 1984. 25th Annual Symposium on*, pages 229–240. IEEE, 1984.

[2] László Babai and Shlomo Moran. Arthur-merlin games: a randomized proof system, and a hierarchy of complexity classes. *Journal of Computer and System Sciences*, 36(2):254–276, 1988.

[3] Manuel Blum. Coin flipping by telephone a protocol for solving impossible problems. *ACM SIGACT News*, 15(1):23–27, 1983.

[4] Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Non-interactive zero-knowledge proof systems. In *Advances in Cryptology—Crypto'87*, pages 52–72. Springer, 1987.

[5] Uriel Feige, Amos Fiat, and Adi Shamir. Zero-knowledge proofs of identity. *Journal of cryptology*, 1(2):77–94, 1988.

[6] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology—CRYPTO'86*, pages 186–194. Springer, 1986.

[7] Michael J. Fischer, Silvio Micali, and Charles Rackoff. A secure protocol for the oblivious transfer. *Journal of Cryptology*, 9(3):191–195, 1996.

[8] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design. In *— 27th Annual Symposium on Foundations of Computer Science*, pages 174–187. IEEE, 1986.

[9] 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 (JACM)*, 38(3):690–728, 1991.

[10] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of computer and system sciences*, 28(2):270–299, 1984.

[11] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on computing*, 18(1):186–208, 1989.

[12] Shafi Goldwasser and Michael Sipser. Private coins versus public coins in interactive proof systems. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 59–68. ACM, 1986.

[13] Peter Linz. *An introduction to formal languages and automata.* Jones & Bartlett Publishers, 2011.

[14] Michael Luby, Silvio Micali, and Charles Rackof. How to simultaneously exchange a secret bit by flipping a symmetrically-biased coin. In *Foundations of Computer Science, 1983., 24th Annual Symposium on*, pages 11–22. IEEE, 1983.

[15] Rafael Pass and Igor Gorodezky. Lecture 18: Zero-knowledge proofs. `http://www.cs.cornell.edu/courses/cs6810/2009sp/scribe/lecture18.pdf`, 2009. "Section 2 was used in the example of proving Graph Isomorphism".

[16] Rafael Pass and J. Aaron Lenfestey. Lecture 14: Interactive proofs. `http://www.cs.cornell.edu/courses/cs6810/2009sp/scribe/lecture14.pdf`, 2009. "Section 2 was used in the example of proving Graph Non-Isomorphism".

[17] Rafael Pass and Karn Seth. Lecture 18: Zero-knowledge proofs - part 3. `http://www.cs.cornell.edu/courses/cs6830/2011fa/scribes/lecture18.pdf`, 2011. "Section 1 was used in the example of proving that a Graph has a 3-coloring".

[18] Jean-Jacques Quisquater, Myriam Quisquater, Muriel Quisquater, Michaël Quisquater, Louis Guillou, Marie Guillou, Gaïd Guillou, Anna Guillou, Gwenolé Guillou, and Soazig Guillou. How to explain zero-knowledge protocols to your children. In *Advances in Cryptology—CRYPTO'89 Proceedings*, pages 628–631. Springer, 1990.

[19] Douglas R Stinson. *Cryptography: theory and practice.* CRC press, 1st edition, 1995.

[20] Vitaly I. Voloshin and Ebrary Academic Complete. *Introduction to graph theory.* Nova Science Publishers, New York, 2009.

[21] Andrew C Yao. Theory and application of trapdoor functions. In *Foundations of Computer Science, 1982. SFCS'08. 23rd Annual Symposium on*, pages 80–91. IEEE, 1982.