On the definitions and the properties of Zero-Knowledge arguments

by

Rosario Gennaro

Laurea in Matematica, Università di Catania (1989)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 1993

© Rosario Gennaro, 1992. All rights reserved.

The author hereby grants to MIT permission to reproduce and distribute publicly paper and electronic copies of this thesis document in whole or in part, and to grant others the right to do so.

Author .....
Department of Electrical Engineering and Computer Science
December 10, 1992

Certified by .....
Silvio Micali
Professor of Computer Science
Thesis Supervisor

Accepted by .....
Campbell L. Searle
Chairman, Departmental Committee on Graduate Students

On the definitions and the properties of Zero-Knowledge arguments

by

Rosario Gennaro

Submitted to the Department of Electrical Engineering and Computer Science

on December 10, 1992, in partial fulfillment of the

requirements for the degree of

Master of Science

Abstract

The notion of Zero-Knowledge proof was introduced by Goldwasser, Micali and Rackoff in their seminal paper. In their original setting a Prover of unbounded computational power proves to a polynomially bounded Verifier a given theorem in a way that conveys no additional knowledge except the mere fact that the theorem is true.

One of the most interesting variants of Zero-Knowledge protocols was proposed by Brassard, Chaum and Crépeau in various papers and subsequently named *Zero-Knowledge arguments*. In a Zero-Knowledge argument both Prover and Verifier are probabilistic polynomial time interactive Turing machines. The Prover knows a proof of a certain theorem (usually an **NP**-statement) and tries to convince the Verifier that the theorem is true without revealing the proof itself.

Those papers missed a formal definition. Attempts made later of formalizing these concepts have been proven wrong in various ways. We will give a new correct definition for this kind of protocols and argue that it corresponds to our intuition of the notion. Finally we prove a more general theorem about the relation between the zero-knowledge property and the witness indistinguishability property for interactive protocols.

The applications of zero-knowledge arguments are countless: identification schemes, signature schemes, noninteractive proofs, oblivious transfer, etc etc. It was frustrating that such important notions were still lacking of a formalization. The goal of this thesis is to give an unified and correct formal frame to these notions.

Thesis Supervisor: Silvio Micali

Title: Professor of Computer Science

Contents

1 Zero-Knowledge protocols **5**

1.1 Introduction 5

1.2 Interactive Proof Systems 6

1.3 Zero-Knowledge Proof Systems 7

1.4 Examples of Zero-Knowledge Proof Systems 9

1.5 Some results about Zero-Knowledge Proof Systems 14

2 Proofs of knowledge **15**

2.1 Introduction 15

2.2 An example: a zero-knowledge argument for graph 3-colorability 16

2.2.1 A digression on number theory 17

2.2.2 The protocol 18

2.3 A first approach to the definition 19

2.3.1 What exactly are proofs of knowledge? 19

2.3.2 The zero-knowledge property for proofs of knowledge 22

3 The definition of zero-knowledge arguments **24**

3.1 Introduction 24

3.2 The definition 24

3.2.1 Redefining proofs of knowledge 25

3.2.2 Redefining the zero-knowledge condition 26

3.2.3 3-coloring revisited 27

3.3 ZKA and Witness Indistinguishability 28

Acknowledgments

First and foremost I would like to thank my advisor Silvio Micali for his help and support. I could not imagine a better person to work with. His enthusiasm makes research work always exciting. I am looking forward to keep working with him.

The Theory Group here at the Laboratory for Computer Science at MIT has provided me with a supportive and cooperative environment. Special thanks to Shafi Goldwasser for introducing me to cryptography and for having me as her teaching assistant. Thanks to Rafi Ostrovsky and Alex Russell for useful research discussions. I would like to thank my officemates David Gillman, Arthur Lent and Lisa Tucker. And finally the group would not be such an enjoyable place without its staff members, Be Hubbard above all.

Part of this thesis has been written during the summer of 1992 at the Dipartimento di Matematica of the Universitá di Catania in Italy. I would like to thank my undergraduate advisor there Alfredo Ferro for suggesting to me the opportunity to go abroad for graduate school. Very special thanks to my best friends over there Giovanni Gallo and Elvinia Riccobene for their tremendous support in the past two years.

Finally I would like to thank my family: my dad Giuseppe, my mom Maria Teresa, my sister Titti and my brother Salvatore. In these past two years, the hardest thing was how much I missed all of you. I would like to dedicate this thesis to you. I hope I can make you proud of me.

Chapter 1

Zero-Knowledge protocols

1.1 Introduction

In this chapter we will introduce the notion of *interactive proof systems* and the concept of *zero-knowledge* protocols as they were initially defined in the seminal paper by Goldwasser, Micali and Rackoff [19].

Informally an interactive proof system for a given language L is a two-parties protocol in which one of the parties (called the Prover) tries to convince the other one (the Verifier) that a certain input string x is in L. The Prover is given unbounded computational power while the Verifier is a polynomial time machine. A very simple example is when L is in **NP**. In that case the Prover computes a proof that $x \in L$ and the Verifier easily check that the proof is correct. This protocol has no interaction at all. Allowing Prover and Verifier to *interact* i.e. to talk back and forth and *randomizing* them i.e. letting them toss coins, we can extend the class of languages whose membership can efficiently checked.

Once we have defined interactive proof systems it comes natural to ask the following question: how much knowledge is released to the Verifier with the proof of the theorem? Let's try to explain what do we mean with this question with an example. Suppose that the Prover wants to prove to the Verifier that a certain graph G is Hamiltonian. An easy way for the all powerful Prover would be to show to the Verifier an Hamiltonian cycle in G. But it is clear that this kind of proof releases much more knowledge that the simple bit "G is (is not) Hamiltonian", in particular it reveals an Hamiltonian cycle of G that the Verifier would have never been able to compute by herself. Is it possible to convince the Verifier in a way that at the end of the protocol she knows only the bit $x \in (\notin) L$?

As we will see the answer is affirmative and the answer relies on the concept of zero-knowledge proof systems.

1.2 Interactive Proof Systems

Let's try now to be a little bit more formal in the definition of interactive proof systems. In the following with the word "theorem" we will refer to a statement of the kind $x \in L$ where L is a given language and x is an input string common to Prover and Verifier.

What we require from an interactive proof system is the following:

- the Prover should always be able to prove a true theorem to the Verifier
- the Verifier should never be convinced of a false theorem, no matter how the Prover tries to cheat her.
- finally the proof must be efficient i.e. the Verifier should be able to finish her share of the protocol in polynomial time (in the size of the input)

First of all we should model these characters of Prover and Verifier as some kind of Turing machines:

Definition 1 ([19]) An *Interactive Turing machine* is a Turing machine with five tapes:

- a read-only input tape
- a work tape
- a random tape containing an infinite sequence of random bits that can be scanned only from left to right. We say that the machine flips a coin when she reads next bit in the random tape.
- a write-only communication tape, to send messages to the other interacting machines.
- a read-only tape, to receive messages.

The task of interactive Turing machines is to carry on *protocols*; for example a two-party protocol is a pair of interactive Turing machines A, B that share the same input tape and B's write-only communication tape is A's read-only communication tape and viceversa.

Work tape and random tape are instead private. The two machines take turns in being active. When (for example) A is active, she performs some internal computation based on the content of her tapes and then writes a message to B in the communication tape and deactivates, then B starts his round. An interactive proof system is a two-party protocol with some particular properties.

Definition 2 ([19]) Let $L \subset \{0,1\}^*$. An *interactive proof system* for L is a two-party protocol (P,V) such that

efficiency P is of unbounded computational power while V is a poly-time interactive Turing machine.

completeness $\forall x \in L$ given as input to (P,V), V will accept with probability 1.

soundness $\forall x \notin L$ sufficiently large, $\forall P'$ interactive Turing machine, on input x to (P',V), V will accept with probability at most $2^{-|x|}$

P is called the *Prover* and V the *Verifier*

It is clear that this definition satisfies the intuitive requirements that we stated at the beginning of this section for interactive proof systems.

The class of languages that have interactive proof systems is called **IP** (*Interactive Polynomial time*). Clearly **NP** $\subseteq$ **IP**. The first example of language in **IP**, but not known to be in **NP** was given by Goldreich, Micali and Wigderson in [17] with an interactive proof system for graph non-isomorphism. Shamir in [24] proved that **IP=PSPACE**. The proof of this fact uses a quite surprising result by Goldwasser and Sipser [20] that states that the secrecy of the coin tosses does not affect the class of languages recognized by interactive proof systems. I.e. an interactive proof system in which the Prover does not "see" the coin tosses of the Verifier can be transformed in one in which the Prover is allowed to see those coin tosses.

1.3 Zero-Knowledge Proof Systems

As we said at the beginning, the informal requirement that we ask to a zero-knowledge protocol is that the Verifier does not have to learn from the Prover more than she is supposed

to. A way to make sure that this requirement is met is to prove that it is possible to reconstruct in polynomial time the conversation between Prover and Verifier when the theorem being proven is actually true. In fact this would mean that whatever the Verifier "sees" during the proof, she could have computed it by herself, so the Prover is not releasing any new knowledge to her.

We put ourselves in an adversarial model i.e. in one in which the Verifier may not follow the protocol, but may deviate from it in an arbitrary way (but still remaining a polynomial time machine). In particular we allow to the Verifier an extra input string (that we will denote by h) to represent any eventual knowledge she may have (for example the history of precedent conversations). A cheating Verifier may try to use h to obtain extra knowledge from the Prover.

Of course we have to formalize these ideas in a satisfactory way. According to [19] we define what the Verifier "sees" as the $View$ of the protocol.

Definition 3 Given an interactive proof system (P, V) for a language L we define the view of V on input x and h as the string $(\sigma, a_1, b_1, \ldots, a_m, b_m)$ where σ is the string of coin tosses of V and a_i (resp. b_i) is the message sent by P (resp. V) at the ith round of the protocol. With $View_{P,V}(x, h)$ we denote the random variable whose value is this view.

Now we want that no matter which cheating Verifier V' interacts with our Prover P the view of the protocol (P, V') has to be efficiently reconstructible. This means that exists a probabilistic algorithm $S_{V'}$ that uses V' as a subroutine which on input $x \in L$ output views of the protocol (P, V') with the "same" distribution of $View_{(P,V')}(x, h)$.

Definition 4 We say that the interactive proof system (P, V) is *zero-knowledge* for a language L if $\forall V'$ verifier, $\exists S_{V'}$ probabilistic polynomial time Turing machine such that $\forall x \in L$ $\forall h$

$$S_{V'}(x) \cong View_{(P,V')}(x, h)$$

$S_{V'}(x)$ is a random value as well, by $\cong$ we mean that the two distributions may be equal or *statistically indistinguishable* or *computationally indistinguishable* as defined in [19]. According to the case we have *perfect* or *statistical* or *computational* zero-knowledge.

Definition 5 Two family of random variables $U(x)$ and $V(x)$ are *statistically indistinguish-*

able on $L \subset \{0,1\}^*$ if for all positive constants c and sufficiently long $x \in L$:

$$\sum_{\alpha \in \{0,1\}^*} |Prob[U(x) = \alpha] - Prob[V(x) = \alpha]| < |x|^{-c}$$

Informally this means that a judge who is given infinite time and poly-size samples of the two distributions is not able to tell which one is which.

Definition 6 Two family of random variables $U(x)$ and $V(x)$ are *computationally indistinguishable* on $L \subset \{0,1\}^*$ if for all poly-size family of circuits C, for all constants c and all sufficiently long $x \in L$:

$$|Prob[C(U(x)) \text{ outputs } 1] - Prob[C(V(x)) \text{ outputs } 1]| < |x|^{-c}$$

In this case a judge is not able to tell apart the two distributions only if he's given poly-size samples and polynomial time to look at them.

A final remark on the definition is that we require the conversation to be simulatable only when $x \in L$; in fact if the simulator were able to output the same views of (P,V) even when $x \in L$ then it would be able to recognize L with small probability of error i.e., since the simulator is a probabilistic polynomial time algorithm, L would be in BPP. In these cases the Verifier would not need the interaction with the Prover to decide if $x \in L$.

1.4 Examples of Zero-Knowledge Proof Systems

The first examples of zero-knowledge proof systems were given in [19] with two protocols for the languages[1]

$$QR_n = \{x \in Z_n^* \text{ and } x \text{ is a square } \}$$

and

$$QNR_n = \{x \in Z_n^*, J_n(x) = 1 \text{ and } x \text{ is not a square } \}$$

Those problems are well known to be in **NP** $\cap$ **co-NP**, but no probabilistic polynomial

[1] Readers not familiar with the basic notions needed to understand the definitions of these languages should just give a quick reading to Section 2.2.1 for a little digression on number theory

time algorithm has been found to recognize them. In [19] a perfect zero-knowledge proof system is given for QR and a statistical one for QNR.

A big breakthrough came with the paper by Goldreich, Micali and Wigderson [17] where zero-knowledge proofs are given for the following problems:

- Graph isomorphism: a perfect ZK protocol is given for the following language

  $$\{(G, H) : G \equiv H\}$$

  This was the first example of ZK proof for something not known to be in **NP** $\cap$ **co-NP** outside from the domain of number theory.

- Graph non-isomorphism: a statistical ZK protocol is given for the following language:

  $$\{(G, H) : G \not\equiv H\}$$

  As noted in Section 1.2 this was the first example of interactive proof system for something not even known to be in **NP**.

- Graph 3-coloring: a computational ZK protocol is given for

  $$\{G \text{ 3-colorable graph }\}$$

  under the assumption that secure encryption schemes exists. Given the **NP**-completeness of this problem [14] this result implies that all the languages in **NP** have a zero-knowledge proof system.

Eventually in [3] they prove that all **IP** (i.e. all **PSPACE**) has computational zero-knowledge proof system.

In this section we will review the ZK proof for graph 3-coloring since we will need it in the following.

Consider the following informal description of the protocol: on input $G \in 3COL$, let n be the number of vertex of G and m the number of edges.

- the Prover P computes a 3-coloring and choose a random permutation of the 3 colors. Then he sends to the Verifier n locked boxes each one containing the color of a vertex.

- the Verifier V selects randomly an edge $e = (u, v)$ in G, and sends it to the Prover asking him to show that this edge is well colored.
- the Prover sends to V the keys of the boxes containing the colors of u and v.
- the Verifier opens the boxes and checks that e is well colored; if so she asks to continue otherwise she rejects and stop. The Verifier accepts after repeating this basic protocol m^2 times.

Of course in a digital implementation of this protocol we cannot use boxes and keys as means to hide information, but using the result of Goldwasser and Micali in [18] we can simulate them by using *secure encryption schemes*

Definition 7 An *encryption function* is a polynomial time computable function

$$f \ : \ M \times \{0,1\}^* \to \{0,1\}^*$$

(M is the message space) such that

$$\forall x \neq y \in M, \forall r, s \in \{0,1\}^* \ f(x,r) \neq f(y,s)$$

A *probabilistic encryption* of a message x with security parameter n is the random variable $f_n(x) = f(x,r)$, where f is an encryption function and r is randomly chosen in $\{0,1\}^n$

An encryption function is *nonuniformly secure* if $\forall x \neq y \in M$ the two family of random variables $\{f_n(x)\}_n$ and $\{f_n(y)\}_n$ are computationally indistinguishable (as defined in Section 1.2)

Now we can describe the actual implementation of the protocol for $3COL$ using encryption functions instead of boxes. In fact we assume that both Prover and Verifier are able to evaluate an encryption function f:

- the Prover P selects a random permutation of the 3 colors of the nodes and then sends them encrypted to the Verifier. This means that the Prover sends to the Verifier a string $z_1, \ldots, z_n$ with $z_i = f(\pi(c_i), r_i)$ where $c_i \in \{1,2,3\}$ is the color of the i-th node of G, $\pi : \{1,2,3\} \to \{1,2,3\}$ is the permutation, and r_i are random numbers.

- the Verifier V chooses at random an edge $e = (u, v)$ and sends it to the Prover.
- the Prover sends the colors of u and v to the Verifier by sending to her $(\pi(c_u), r_u)$ and $(\pi(c_v), r_v)$
- the Verifier accepts if the colors are actually different and they are really the decryption of what P sent precedently. She does it by checking that $z_u = f(\pi(c_u, r_u)$ and $z_v = f(\pi(c_v, r_v)$.

The basic protocol is again repeated m^2 times and the Verifier at the end accepts if she accepted at each iteration of it.

Theorem 1 *If f is a nonuniformly secure encryption function then this protocol is a computational zero-knowledge proof system for $3COL$.*

Proof

First of all we have to show that the protocol is an interactive proof system for $3COL$ and then we construct the simulator to prove the zero-knowledge property of it.

If G is really 3-colorable then it is clear that the Prover can always complete his share of the protocol and the Verifier will accept with probability 1.

If G is not 3-colorable then in what the Prover will send encrypted to the Verifier there will be at least one edge not well colored. So with probability $\frac{1}{m}$ the Verifier will ask this bad edge. Since each encrypted message has an unique decryption the Prover will not be able to change the colors when he reveals them. The basic protocol is repeated m^2 times so at the end the probability that the Verifier will believe a cheating Prover is $(1 - m^{-1})^2 \cong e^{-m}$.

This proves that the protocol is actually an interactive proof system for $3COL$.

To prove that the protocol is zero-knowledge we have to exhibit a machine $S_{V'}$ that run in expected polynomial time and such that $S_{V'}(G)$ is computationally indistinguishable from $View_{(P,V')}(G)$ for any possible Verifier V'. In the simulation we will try to guess which edge V' is going to ask, we encrypt an incorrect coloring of the graph such that that edge is well colored. If we are lucky V' will actually ask that edge, otherwise the case will be ignored and we will repeat it.

More specifically the simulation of the j^{th} of the m^2 rounds will be as following

- $S_{V'}$ chooses a random string r long enough to cover all the need of coin tosses from V' during the simulation and put it in the random tape of V'
- $S_{V'}$ chooses randomly an edge $e = (u, v) \in E$ and random color $c_i \in \{1, 2, 3\}$ $i = 1, \ldots, n$ for the nodes of the graph with the only condition that $c_u \neq c_v$. Then it sets $z_i = f(c_i, r_i)$
- $S_{V'}$ puts $z_1, \ldots, z_n$ on the communication tape of V' and sets $e' = V'(G, r, H, z_i)$ (H is the transcript of the precedent rounds) i.e the edge that V' would ask at the j^{th} round if feeded with those z_i's.
-
  - if $e = e'$ then $S_{V'}$ simulates the Prover by revealing (c_u, r_u) and (c_v, r_v) appends this round of conversation to H and goes to the next round.
  - if $e \neq e'$ $S_{V'}$ will pretend nothing happened and repeat the simulation of the current round.

$S_{V'}$ will halt after all the m^2 rounds are completed.

To prove that this is a valid simulation we have to prove the following two claims

Claim 1 *$S_{V'}$ ends in expected polynomial time*

Claim 2 *The family of random variables $\{S_{V'}(G)\}_{G \in 3COL}$ and $\{View_{(P,V')}(G)\}_{G \in 3COL}$ are computationally indistinguishable.*

The first claim is proven observing that the probability that V' asks edge $e = (u, v)$ seeing the encryption of a false coloring of the graph must be approximately $\frac{1}{m}$ or otherwise she could be able to break the encryption function i.e. to distinguish between the encryption of a false coloring where $c_u \neq c_v$ from one where $c_r \neq c_s$ for any other edge $e' = (r, s) \in E$ (a detailed proof can be found in [17]). So each round will be completed successfully after expected m trials, so $S_{V'}$ will complete after expected m^3 rounds. Each round is polynomial time since V' is.

The second claim can be proven observing that if a circuit can distinguish between the distribution of $S_{V'}(G)$ and $View_{(P,V')}(G)$ then it can be modified to one that distinguish between the encryption of a true coloring of the graph and the encryption of a false one (in fact this is the difference between the two random variables) contradicting the assumption on the encryption function. Again a complete proof can be found in [17].

■

1.5 Some results about Zero-Knowledge Proof Systems

In the description of the protocol for $3COL$ in Section 1.4 we note that for the Prover it suffices to have a *bit commitment scheme* to be sure of releasing no knowledge to the Verifier.

Definition 8 A *bit commitment* protocol consists of two stages:

the commit stage party A has a bit b to which he wishes to commit exchanging messages with party B

the revealing stage after which B knows the value of b

The protocol must satisfy the following conditions for any polynomial p and security parameter k.

1. After the commit stage B is not able to guess b with probability greater than $\frac{1}{2} + \frac{1}{p(k)}$
2. At the revealing stage A can only reveal one possible value. If she tries to cheat and reveal a different value than the one she committed to, she will be caught with probability $1 - \frac{1}{p(k)}$

Using recent results by Naor [22] and Impagliazzo, Levin and Luby [21] the existence of bit commitment schemes is equivalent to the existence of one-way functions. So Theorem 1 can be restated with the simpler assumption of the existence of one-way function.

We said that in [19] they give a statistical zero-knowledge proof system for the problem of quadratic non-residuosity. Similarly in [17] they give a statistical zero-knowledge proof system for the problem of graph non-isomorphism. In both papers a way to transform these protocols in *perfect* zero-knowledge ones is indicated. In fact it is still unknown if the class of languages that have perfect zero-knowledge proof systems is a strict subset of the class of languages that have statistical ones or if they are equal.

Combining results by Boppana, Hastad and Zachos [4] and Fortnow [11] it has been proven that if an **NP**-complete language had perfect or statistical zero-knowledge proof system then the polynomial time hierarchy would collapse.

Chapter 2

Proofs of knowledge

2.1 Introduction

If we look back at the zero-knowledge proof system for 3-coloring in the previous chapter we can notice that if the Prover knew in advance a 3-coloring of the graph he would not need unbounded computational power, but could terminate his share of the protocol even being a polynomial time machine.

This point suggested to some researchers a possible modification to the original model of Goldwasser, Micali and Rackoff in the following way: the Prover is not infinite power, but he is a polynomial time machine that "knows" a particular witness of the **NP**-statement he wants to prove to the Verifier [6, 7, 10].

Using these ideas Brassard and Crépeau [8] devised a zero-knowledge protocol for SAT (and then for all **NP**) that has the property of not releasing any knowledge to the Verifier in a strong information-theoretic sense i.e. using the terminology of [19] is a *perfect* zero-knowledge protocol. The correctness of the protocol relies though on the limited power of the Prover i.e. a Prover that could, for example, factor would be able to cheat on the Verifier.

At the beginning the differences between the two models were not entirely noticed and the name of zero-knowledge proof system was used to denote even this kind of protocols. The publication of the paper by Fortnow [11] that states that if an **NP**-complete problem had a perfect zero-knowledge proof system then the polynomial time hierarchy would collapse posed people in front of this question: or the hierarchy really collapsed or instead the protocols a la Brassard were not *zero-knowledge proof systems* (not at least as defined by

Goldwasser, Micali and Rackoff).

Fortnow in his paper uses this last model and uses heavily the property of the Prover of being infinite powerful. It is clear that his result does not apply to the other model and some confusion was raising from the misuse of terminology. Brassard and Crépeau "sorted out" the situation proposing in [9] the name of *zero-knowledge arguments* for their protocols. Another name proposed in [12] was *zero-knowledge proofs of knowledge* that makes evident how in this kind of protocols the Prover is engaged in proving to the Verifier that indeed he knows a witness.[1] Still another name was proposed by Chaum and Brassard [5]: *zero-knowledge computationally convincing proofs*. This name instead refers to the fact that these kind of proofs are valid only in a computational sense i.e. a powerful Prover would disrupt the validity of the proof (as we will see in the following). The question was finally settled down and the name *proof systems* now refers to the Goldwasser, Micali, Rackoff model, while the names *arguments, proofs of knowledge, computationally convincing proofs* to the model a la Brassard, Crépeau or Chaum. In this thesis we will follow this convention.

The question that was not settled down instead was the one of finding a good definition for this kind of protocol. The firsts that tried to formalize the properties of *proofs of knowledge* were Feige, Fiat and Shamir in [12]. Unfortunately as we will see at the end of this chapter their definition is not satisfactory in many aspects.

2.2 An example: a zero-knowledge argument for graph 3-colorability

In order to understand better the concept of zero-knowledge argument, we are going to give an example for an **NP**-complete problem, to which we will refer later to exemplify the properties of our definition. The first example of a zero-knowledge argument for an **NP**-complete problem is a protocol for SAT described in [6]; here instead we present a simpler one for graph 3-colorability based on the zero-knowledge proof system in [17] that we described in Section 1.4. As we showed before that protocol is only computationally zero-knowledge since a powerful verifier could decrypt the coloring of the graph and gain extra information.

[1] Indeed the protocol can be modified in a way such that at the end the Verifier does not know if $x \in L$ or $x \notin L$ but only that P knows the staus of x with respect to L

To convert this protocol in a zero-knowledge argument we suppose that the Prover already knows a particular color assignment to the nodes of G; at each repetition of the protocol he commits to a random permutation of the colors but using a different form of commitment based on the problem of quadratic residuosity. As we will show later this commitment conveys no information about the coloring in a strong information-theoretic sense; the trade-off is that if the Prover were not polynomially bounded he could open the commitment in different ways.

We divide this section in two parts: in the first we digress on number theory to understand better this new commitment scheme; in the second we describe the protocol itself and prove informally its correctness (in fact to prove it in a formal way we have to wait for a correct formal definition).

2.2.1 A digression on number theory

Let n be an integer, with Z_n^* we denote the multiplicative group modulo n i.e. the set of numbers relatively prime to n between 1 and $n-1$. An integer $x \in Z_n^*$ is called a *quadratic residue* modulo n ($x \in QR_n$) if there exists a $y \in Z_n^*$ such that $x = y^2 \bmod n$.

If $n = p$ is a prime, testing quadratic residuosity is easy [1], and with the *Legendre symbol* we define the number

$$(\frac{x}{n}) = 1 \text{ if } x \text{ is a quadratic residue}$$

$$(\frac{x}{n}) = -1 \text{ otherwise}$$

If $n = \Pi_{i=1}^{k} p_i$ then for all $x \in Z_n^*$ we define the *Jacobi symbol* of x as the number

$$J_n(x) = \Pi_{i=1}^{k} (\frac{x}{p_i})$$

The following facts are well known in number theory [1]:

Fact 1 *Given $n \in N$ and $x \in Z_n^*$, $J_n(x)$ can be computed in time polynomial in $|x|$.*

Fact 2 *If $J_n(x) = -1$ then x is a quadratic nonresidue modulo n. However if $J_n(x) = 1$ there is no polynomial time algorithm known to tell if x is or not a quadratic residue modulo n.*

If $n = pq$ is the product of two odd primes then extracting square roots is computationally equivalent to factoring [23]. We assume that when the primes are large, factoring n is computationally infeasible, and then even extracting square roots modulo n.

Now we can describe the bit commitment scheme based on the problem of quadratic residuosity: party A has a bit b and she wants to commit to it. The protocol goes this way:

commit stage

- B chooses randomly two primes $p_1, p_2 \equiv 3 \bmod 4$, sets $n = p_1 p_2$ and s a random square mod n. She sends n, s to A.
- B proves to A in zero-knowledge that s is indeed a square mod n using the protocol of [19].
- A chooses a random $y \in Z_n^*$ and set $m = y^2 s^b$ and sends m to B

reveal stage

- A just sends y to B

This is a valid bit commitment scheme, in fact

1. B is not able after the commit stage to realize what bit A is hiding. Note that even if B were all powerful he could not discover b, in fact the only thing he sees are random squares. The only chance for B to discover something is to send to A as s a non-square with Jacobi symbol 1. In this case only being lucky he could pass the second step, the one in which he proves that s is indeed a square.
2. At the reveal stage the only way for A to reveal a different bit of the one she had in mind is to compute a square root of s. But if we assume that A is polynomial time and that factoring is infeasible then this is impossible.

2.2.2 The protocol

In more details the protocol is as following:

- V chooses randomly two primes $p_1, p_2 \equiv 3 \bmod 4$, sets $n = p_1 p_2$ and s a random square mod n. She sends n, s to P.
- V proves to P in zero-knowledge that s is indeed a square mod n using the protocol of [19].

- P selects a random permutation of the colors of the node of the graph. He then commits to this coloring of the graph in the following way:
  - for each node v in the graph he chooses two random elements of Z_n^*: $R_{v,0}$ and $R_{v,1}$.
  - if $c_v = (c_{v,0}, c_{v,1})$ is the permuted color of node v encoded by two bits he sets $S_{v,b} = s^{c_{v,b}} R_{v,b}^2$
  - he sends to V $(v, S_{v,0}, S_{v,1})$ for each node v
- V chooses an edge (u, v) at random and sends it to P
- P discloses the colors of that edge by sending to V $R_{u,0}, R_{u,1}, R_{v,0}, R_{v,1}$
- V checks that the colors of u and v are different, if so accepts otherwise rejects.

The last three steps of the protocol are repeated $|E|^2$ times, where E is the set of edges of G. The Verifier accepts iff she accepts in all the iterations of the protocol.

As noted before, the condition of P being limited in computational power is needed for the protocol to be sound. In fact an infinite power Prover could factor n and disclose whichever color he likes in the last step.

2.3 A first approach to the definition

In this section we will describe the definition of zero-knowledge proofs of knowledge proposed by Feige, Fiat and Shamir in [12] and we will point out the bugs contained in their formalization. In next chapter we will try to fix these bugs.

2.3.1 What exactly are proofs of knowledge?

If we look back at the definition of *interactive proof systems* we can see that the Prover's job is to convince the Verifier that a certain theorem $x \in L$ is true. For this reason we ask the following properties to the protocol

completeness if the theorem is true then the Prover must always convince the Verifier of this fact.

soundness if the theorem is false no Prover is able to convince the Verifier if not with exponentially small probability.

The formalization of these requirements had been described in Section 1.2.

Now we are dealing with proofs of knowledge and not anymore with proofs of membership. First of all we have to modify our computation model. Prover and Verifier are both probabilistic **polynomial time** interactive Turing machine that shares a common input tape, and their communication tapes in the same fashion as before (i.e the read-only communication tape of the Prover is the write-only communication tape of the Verifier and viceversa). Moreover each of them has a private random tape, work tape and *knowledge* tape. This is the first difference: a knowledge tape for the Prover. He is going to use it to store the witness of the **NP**-statement he wants to prove to the Verifier.

Instead of talking about **NP**-languages it is preferable to talk about *polynomially bounded relationships* i.e. a relationship $R \subset \{0,1\}^* \times \{0,1\}^*$ that is

1. $R \in \mathbf{P}$
2. $\exists p$ polynomial such that if $(x,y) \in R$ then $|y| \leq |x|$

The **NP**-language defined by such a relationship R is

$$L_R = \{x : \exists y \text{ such that } (x,y) \in R\}$$

If $x \in L_R$ with R_x we denote the set of *witnesses* of x i.e.

$$R_x = \{y : (x,y) \in R\}$$

x will be put in the common input tape, while y (resp h) will go on the private knowledge tape of the Prover (resp Verifier). To stress this fact we will denote the final output of the protocol (i.e. accept or reject iff the Verifier accepts or rejects) with $(P(y), V(h))(x)$.

The informal properties that we require from a proof of knowledge are

completeness if the theorem is true and the Prover knows a proof of it then he must always convince the Verifier of this fact

soundness no Prover that does not know a proof of the theorem can convince the Verifier if not with exponentially small probability.

The main problem in formalizing these concepts is to find a good definition of a "machine that knows something". This problem was solved in [12] with the following idea: if a machine A "knows" something then there is some other machine that will be able to extract this knowledge from A. This machine was called *knowledge extractor*. More precisely:

A knows a string s if there is some polynomial time Turing machine M with complete control over A which outputs s as a result of its interaction with A.

In [12] they reformulated the soundness condition in the following way

soundness whenever a Prover convinces the Verifier with non negligible probability then he must know a proof of the theorem and then it is possible to extract this knowledge from him.

Writing everything in a more formal language the definition in [12] becomes:

Definition 9 A pair of interacting polynomial time Turing machine (P, V) is an *interactive proof system of knowledge* for the polynomially bounded relationship R if

completeness $\forall x, y$ such that $(x, y) \in R$, $\forall h$

$$Prob[(P(y), V(h))(x) \text{ accepts }] = 1$$

soundness $\forall k_1$, $\exists M$ prob. poly-time Turing Machine, $\forall k_2$, $\forall P'$ provers, $\exists k_3$, $\forall x$ with $|x| > k_3$, $\forall y$, $\forall h$

if

$$Prob[(P'(y), V(h))(x) \text{ accepts }] > \frac{1}{|x|^{k_1}}$$

then

$$Prob[M(P', x, y) \text{ outputs } y' \in R_x] > 1 - \frac{1}{|x|^{k_2}}$$

Goldreich in [16] considers this definition not general enough. In fact he notices that the above scheme requires nothing in the case in which the Prover does not convince the Verifier with non negligible probability. This is not the same of saying that this Prover convinces the Verifier with negligible probability, in which case we should not worry about it.

An example of this subtle point is the following (still due to Goldreich in [16]):

Example 1: Consider a Prover P and a Verifier V on an infinite sequence of CNF formulas $\{\phi_n : n \in N\}$. P convinces V that on common input ϕ_n he knows a satisfying assignment but only with probability $\frac{1}{n^k}$, where $n = |\phi_n|$ and k is the length of the longest clause of ϕ_n. This sequence of probabilities is neither negligible (i.e. smaller than $\frac{1}{n^c}$ for any $c > 0$) nor non-negligible (i.e. bigger than $\frac{1}{n^c}$ for some $c > 0$). So the previous definition requires nothing for such a Prover. This would even disrupt the possibility of proving that a given protocol is zero-knowledge: in fact suppose to invert the roles of P and V, P sends a satisfying assignment to V only after she convinces him that she already knows it. This protocol should be zero-knowledge according to our intuition, but we are not able with this definition to prove that V actually "knows" the assignment and then that the protocol is zero-knowledge.

As noted by Goldreich and how we will see in the next chapter this is not just a theoretical worry, since the problem appears even in concrete applications of zero-knowledge proofs of knowledge to encryption schemes.

In chapter 3 we will discuss again this and other problems of this definition and will propose possible solutions.

2.3.2 The zero-knowledge property for proofs of knowledge

As we noticed in section 1.3 when we talk about zero-knowledge we have to take into account even the extra knowledge tape of the Verifier, we will denote with h this string of knowledge already in possess of the Verifier.

We will define the View of the protocol in the same way as in [19] (see section 1.3) and denote it with $View_{(P(y),V(h))}(x)$ to stress the fact that y is private to P and h to V.

According to [12] (see also [13]) the zero-knowledge condition goes as following:

zero-knowledge $\forall V'$ probabilistic polynomial time interactive Turing machine, $\forall h$, $\exists S_{V'}$ probabilistic polynomial time Turing machine such that $\forall (x, y) \in R$, we have

$$S_{V'}(x, h) \cong View_{(P(y),V'(h))}(x)$$

where $\cong$ means computationally indistinguishable

Notice that the simulation is required only when the Prover has on his private tape a correct witness of the **NP**-statement that he wants to prove.
This point of the definition was criticized by the author in [15]. To understand the problems with this formulation consider the following:

Example 2: Let G be a 3-colorable graph and P_G a Prover for 3-colorability that has hardwired a 3-coloring of G. P_G follows the protocol we described before except when G is the input graph in which case if he has a correct 3-coloring on his private tape then he follows the protocol otherwise he gives away the coloring he has hardwired and convinces V that G is indeed 3-colorable. This protocol (P_G, V) is obviously not zero-knowledge according to our intuition since it reveals the coloring on certain inputs. It satisfies the conditions in [12] though. In fact the particular conversation in which the coloring is revealed need not to be simulated according to their definition since the Prover does not have a witness in his private tape.

Next chapter will be devoted to the solution of these problems.

Chapter 3

The definition of zero-knowledge arguments

3.1 Introduction

In this chapter we will propose our version of the definition for zero-knowledge arguments. We will state formally our definition and then we will dedicate a couple of sections to an informal explanation of it and of the motivations that bring us to believe that this is the correct one. We will show how this definition correctly handles the examples given in the previous chapter that instead raised problems with the definition of [12]. Finally, as an example of what kind of mistakes can spring from dealing with incorrect formalizations, we will talk briefly about the relationship between zero-knowledge arguments and witness indistinguishable protocols. This topic was first brought up by Feige and Shamir in [13]. Unfortunately their result is in some way affected by the wrong definitions they refer to. We will revise their results at the light of our new formal frame.

3.2 The definition

Recall from Section 2.3.1 that Prover and Verifier are probabilistic polynomial time interactive Turing machine with a shared public input tape and a private input tape. With $(P(y), V(h))(x)$ we denote the output of the interaction on public input x and y (resp. h) private input for the Prover (resp. Verifier) and with $View_{(P(y),V'(h))}(x)$ the view (as defined in [19]) of the protocol.

Definition 10 Given a polynomially bounded relationship R (see section 2.3.1) we say that (P,V) is a *Zero-Knowledge argument* (ZKA) for R if the following conditions are met

honesty $\forall x, y'$ such that $(x, y') \notin R$ then $P(x, y') = \text{STOP}$

completeness $\forall x, y$ such that $(x, y) \in R$, $\forall h$

$$Prob[(P(y), V(h))(x) \text{ accepts }] = 1$$

soundness $\forall x$, $\forall P'$ prover, $\forall h$, $\forall y'$, $\exists M_{P'}$ probabilistic Turing machine, such that if

$$p(x) = Prob[(P'(y'), V(h))(x) \text{ accepts }] > 2^{-|x|}$$

then $M_{P'}$ on input x outputs $y \in R_x$ within an expected number of steps bounded by $|x|^c \cdot |y| \cdot \frac{1}{p(x)}$ for some positive constant c.

zero-knowledge $\forall V'$ verifier, $\exists S_{V'}$ probabilistic polynomial time Turing machine such that $\forall (x, y) \in R$ $\forall h$

$$S_{V'}(x) \simeq View_{(P(y), V'(h))}(x)$$

where $\simeq$ means that the two distribution are equal or statistically or computationally indistinguishable.

$M_{P'}$ is called *universal knowledge extractor* and $S_{V'}$ the *simulator*.

This series of requirements may look cryptic at first sight, let's try to explain in informal words what do we really mean with them.

3.2.1 Redefining proofs of knowledge

Let's leave aside for a moment the zero-knowledge question and try to focus on how we are redefining the concept of proof of knowledge. Next section we will return to the zero-knowledge part.

As we saw in Section 2.3.1 Goldreich in [16] suggested that the problem with the previous formalization of [12] was that it requires nothing in the case in which the Prover does not convince the Verifier with non-negligible probability. The problem was in the formalization of the soundness condition in [12]; the way we reformulated it is still due to Goldreich.

Definition 11 Let $f : N \to R$ be a function.

We say that f is *negligible* if $\forall c > 0$ and all sufficiently large n we have $f(n) < \frac{1}{n^c}$.

We say that f is *non-negligible* if $\exists c > 0$ such that for all sufficiently large n we have $f(n) \geq \frac{1}{n^c}$.

Now non-negligible is not the negation of negligible, but a strong negation of it. There exists functions that are neither negligible nor non-negligible. An example is the function describing the probability of acceptance in Example 1 in Section 2.3.1.

As stated above the soundness condition implies that if $x \notin L_R$ then the probability that V accepts is at most $2^{-|x|}$, otherwise the knowledge extractor $M_{P'}$ would output $y \in R_x$ while such an y does not exist. In other words a false theorem can be proven only with negligible probability.

But as we saw in Section 2.3.1 the soundness condition requires more. It requires a relation between the probability of acceptance of the Verifier and the expected time needed by the knowledge extractor to find the witness.

If this probability is non-negligible then the knowledge extractor runs in expected polynomial time and in this case the definition of Goldreich agrees with the one in [12]. This soundness condition however is more general as it is valid in the cases in which the previous one does not suffice.

Interested readers may refer to Goldreich paper [16] for a very interesting discussion about these topics.

3.2.2 Redefining the zero-knowledge condition

Besides the soundness condition, the other difference between our definition and the definition in [12] is in the additional "honesty" condition that we require to our Prover.

If we look at the zero-knowledge condition we see that requesting that the conversation is simulatable we basically ask that whatever the Verifier sees she could have computed it by herself and then she learns nothing from the conversation. This is the same notion as in [19], but notice that we here ask the conversation to be simulatable when $(x, y) \in R$. In fact the main difference with the standard *proof systems* is the presence of the private input of the Prover.

Now as we noticed in Example 2 in Section 2.3.2, this may bring to a leak of knowledge in the cases in which the Prover does not have a witness in his private tape. In that case we require nothing so we could have $x \in L_R$, $y \notin R_x$ and the Prover able in some way to convince the Verifier. But since in this case we do not require the conversation to be simulatable the Prover could release all the knowledge he wants to the Verifier and still pass as a zero-knowledge Prover according to the definition.

This is the reason why we add the "honesty" condition. Basically what we are saying is that a Prover who does not want to risk to release knowledge to the Verifier should be "honest" and shut up as soon as he realizes that he does not have a witness in his private tape.

This is analogous to the case of zero-knowledge proof systems where a Prover should not attempt to prove that $x \in L$ when actually $x \notin L$ if he wants to be sure to not release any knowledge to the Verifier. In fact in [19] the simulation of the protocol is required only when $x \in L$.

The Prover in Example 2 would not pass the honesty condition and then he would not be zero-knowledge according to our definition. As we noticed before he would instead satisfy the definition in [12].

3.2.3 3-coloring revisited

The protocol for 3-colorability described in Section 2.2 is actually a ZKA once we add at the beginning an extra step in which the Prover would stop if he does not have a 3-coloring of the graph G in his private tape. Let's see why it is so going through the required conditions one by one:

honesty It's clearly satisfied.

completeness If the graph G is 3-colorable and the Prover knows a 3-coloring of the nodes then he can always perform his part of the protocol and convince the Verifier with probability 1.

soundness If the graph is not 3-colorable or if the Prover does not know the 3-coloring then an analysis similar to the one in [17] shows that any Prover who cannot change the coloring to which he has committed in the second step, succeeds in convincing the Verifier only with probability $e^{-|E|}$. In our case these are all the probabilistic

polynomial time provers who cannot factor n. So in this case the probability that the Prover convinces the Verifier is always 1 or negligible so we do not have to worry about the running time of the knowledge extractor.

zero-knowledge The proof of zero-knowledge is very similar to the one in [17] with the only difference that in this case the commitment of the colors by the Prover does not convey any information in a strong sense and so we can achieve perfect zero-knowledge.

3.3 ZKA and Witness Indistinguishability

In [2, 13] the concept of *witness indistinguishability* was independently introduced with the following definition:

Definition 12 *A proof system (P,V) is witness indistinguishable (WI) for R if for any V', any large input $x \in L_R$ and any $y_1, y_2 \in R_x$ the distributions $View_{(P(y_1),V'(h))}(x)$ and $View_{(P(y_2),V'(h))}(x)$ are statistically indistinguishable.*

in the same paper the following theorem was proved:

Theorem 2 *Let (P,V) be any zero-knowledge protocol. Then the protocol is WI.*

The proof of this theorem is wrong for two reasons: one is that computational zero-knowledge cannot imply a statistical indistinguishability result. The second reason is a consequence of their wrong definition of the zero-knowledge property.

The theorem is still true though once it is reformulated in the following way:

Theorem 3 *Let (P,V) a statistical zero-knowledge argument for R. Then (P,V) is a witness indistinguishable protocol for R.*

Sketch of Proof The idea of the proof is the same as in [13]. Let J be an unlimited power judge that outputs 1 with probability p on views of the ZKA (P,V) when P is using y_1 as witness. Then J outputs 1 with the same probability if he has been given the simulated view output by S. But this view is independent of the witness P is using then J has the same probability of outputting 1 whatever witness P is using. ■

Considering that we gave a perfect zero-knowledge argument for $3COL$ and that $3COL$ is an **NP**-complete language we can state the following theorem:

Theorem 4 *Any language in* **NP** *has a perfect zero-knowledge argument and then a perfect WI protocol.*

Bibliography

[1] D.Angluin, *Lecture notes on the complexity of some problems in number theory*, Department of Computer Science, Yale University, Tech. Rep. 243, 1982

[2] M.Bellare, S.Micali, R.Ostrovsky, *The (true) complexity of statistical zero-knowledge*, in Proceedings of STOC 1990, pp.482-493

[3] M.Ben-Or, O.Goldreich, S.Goldwasser, J.Hastad, J.Kilian, S.Micali, P. Rogaway, *Everything provable is provable in Zero-knowledge* in Proceedings of CRYPTO 1988, LNCS Vol.403, pp. 37-56

[4] R.Boppana, J.Hastad, S.Zachos, *Does co-NP have short interactive proofs?*, Inf. Proc. Lett. Vol.37, No.2, pp.127-132, 1987

[5] G.Brassard, *The Cryptology Column*, SIGACT News, Vol.22, No.1, pp.5-12, 1991

[6] G.Brassard, D.Chaum, C.Crépeau, *Minimum Disclosure Proofs of Knowledge*, J. of Computer and System Sciences, Vol.37, No.2, pp.156-189, 1988

[7] G.Brassard, C.Crépeau, *Zero-knowledge simulation of Boolean circuits*, in Proceedings of CRYPTO 1986, LNCS Vol.263, pp.223-233

[8] G.Brassard, C.Crépeau, *Non-transitive transfer of confidence: a perfect zero-knowledge interactive protocol for SAT and beyond*, in Proceedings of FOCS 1986, pp.188-195

[9] G.Brassard, C.Crépeau, *Sorting out zero-knowledge*, in Proceedings of EUROCRYPT 1989, LNCS Vol.434

[10] D.Chaum, *Demonstrating that a public predicate can be satisfied without revealing any information about how*, in Proceedings of CRYPTO 1986, LNCS Vol.263, pp.195-199

[11] L.Fortnow, *The complexity of perfect zero-knowledge*, in Proceedings of STOC 1987, pp.204-209

[12] U.Feige, A.Fiat, A.Shamir, *Zero-Knowledge proofs of identity*, J. of Cryptology, Vol.1, pp.77-94, 1988

[13] U.Feige, A.Shamir, *Witness Indistinguishable and Witness Hiding Protocols*, in Proceedings of STOC 1990, pp.416-426

[14] M.Garey, D.Johnson, *Computers and Intractability: A guide to the theory of NP-completeness*, Freeman, 1979

[15] R.Gennaro, *A note on the definitions and the properties of zero-knowledge arguments and witness indistinguishable protocols*, manuscript, 1992

[16] O.Goldreich, *On defining proofs of knowledge*, manuscript, 1991

[17] O.Goldreich, S.Micali, A.Wigderson, *Proofs that yield nothing but their validity and a methodology of cryptographic protocol design*, in Proceedings of FOCS 1986, pp.174-187
final version: *Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems*, J. of the ACM, Vol.38, No.1, pp. 691-729, 1991

[18] S.Goldwasser, S.Micali, *Probabilistic Encryption*, J. of Computer and System Sciences, Vol.18, No.1, pp.186-208

[19] S.Goldwasser, S.Micali, C.Rackoff, *The knowledge complexity of interactive proof systems*, in Proceedings of STOC 1985, pp. 291-304
final version: *The knowledge complexity of interactive proof systems*, in SIAM J. Comput. Vol.18 No.1, pp 186-208, 1989.

[20] S.Goldwasser, M.Sipser, *Private coins versus public coins in interactive proof systems*, in Proceedings of STOC 1986, pp. 59-68.

[21] R.Impagliazzo, L.Levin, M.Luby, *Pseudorandom generation from one-way functions*, in Proceedings of STOC 1989, pp.12-24.

[22] M.Naor, *Bit commitment using pseudorandomness*, In Proceedings of CRYPTO 1989, LNCS Vol.435, pp.128-137.

[23] M.Rabin, *Digitalized signatures and public-key functions as intractable as factorization*, MIT-LCS-TR212, Cambridge MA, 1979.

[24] A.Shamir, $IP = PSPACE$, in Proceedings of FOCS 1990, pp.11-15.