Zero Knowledge Proofs and Applications to Financial Regulation

Abstract

We consider zero-knowledge proofs, a class of cryptographic protocols by which an agent (a Prover) can prove to another agent (a Verifier) that a statement is true without revealing any additional information. For example, a zero-knowledge proof allows one to prove *knowledge of* a password to somebody at the other end of the communication without actually revealing the password.

We present an introduction to and survey literature on zero-knowledge proofs, covering the history, formal definition, and classical applications of zero-knowledge proofs. In addition, we consider connections to complexity, demonstrating that all problems in the complexity class **NP** have zero-knowledge proofs, and also discuss more exotic applications of zero-knowledge, namely in electronic voting and nuclear disarmament.

We then consider applications of zero-knowledge to financial regulation, specifically in balancing transparency and confidentiality in financial reporting. Namely, we polled professionals in the financial industry to identify three major classes of regulatory problems. We then utilize zero-knowledge proofs to develop and present cryptographic protocols/mechanisms and solutions to these regulatory problems: (1) An employer verifying an employee has no financial holdings on a blacklist without revealing the other (allowed) holdings of the employee, (2) A fund convincing its investors that its holdings subscribe to particular risk constraints, without disclosing the actual holdings, (3) A collection of investors of a fund verifying aggregate information provided by the fund, while preserving pairwise anonymity. Applications (1) and (3) are novel applications developed in this paper, while (2) is drawn from [47].

Acknowledgements

Firstly, I would like to thank my advisor Prof. Boaz Barak for his constant guidance and support over these past two semesters, whether it be suggestions on papers to read, helpful discussions and feedback, or general musings about cryptography.

I would also like to thank Prof. Taubes for being my math advisor for this thesis. Prof. Taubes' feedback throughout the process greatly enhanced the readability and quality of the paper, particularly in Part I.

In addition, I would also like to thank Prof. Salil Vadhan and Prof. Scott Kominers for being my Computer Science department thesis readers. In particular, I would like to thank Scott for introducing me to the fields of market and mechanism design; the spirit of mechanism design greatly informs the protocol design in Part II of the thesis.

I also thank Prof. Nicole Immorlica and Prof. Adam Smith for assisting me in selecting my thesis topic. Prof. Immorlica recommended looking into zero-knowledge proofs and suggested researching exotic applications such as electronic voting. Prof. Adam Smith's original paper on the transparency and confidentiality trade-off served as an inspiration to looking into applications in financial regulation. In particular, his recommendation to collect vignettes and personal testimonies greatly aided me in finding relevant and robust applications.

Also, I would like to thank my current and former colleagues at PJT Partners, Bracebridge Capital, and Ellington Management Group for willing to speak with me about their own views on financial regulatory matters and allowing me to collect their ideas and insights in Chapter 6.

More generally, I would like to thank Harvard's mathematics, computer science, and economics departments that have helped me grow as a student. This thesis is very much a culmination and combination of ideas developed over the years and influenced by all three departments.

I would also like to thank my friends for all their support throughout the entire process, whether it be accepting of my tendency to bail on plans, or simply helping me cool off in times of stress.

Lastly, I would like to thank my parents for everything they have done for me. They instilled in me my original passion for mathematics, science, and technology, and it is to them I dedicate this thesis.

Contents

Chapter 0

Introduction

Imagine your friend is color-blind and you have two balls: one red and one blue, but otherwise indistinguishable. To your friend, the balls look identical and hence he is skeptical they are actually different. You want to prove to your friend that the balls are distinct without revealing which ball is which. How might we do this?

Consider the following procedure. Your friend takes both balls and puts them behind his back. He takes one ball out from behind his back and displays it to you. He returns this ball to behind his back, and then again chooses to reveal one of the two balls, switching to the other ball with probability 50%. He then asks, "did I switch balls?"

If the balls are different colors, then we can say with certainty whether our color-blind friend switched balls. However, if the balls were the same color and hence identical, we can only guess correctly with probability 50%. Therefore, if we repeat this process for a few iterations, and answer correctly every time, our friend will become convinced that the balls are indeed distinguishable.

This classical example (from [22, 53]) illustrates the core idea behind *zero-knowledge proofs*, protocols by which one can prove a statement to be true (i.e. the balls are different color) without revealing any additional information (i.e. friend never learns which ball is red or which ball is blue). Zero-knowledge proofs are naturally of great importance in security and cryptography (i.e. proving/demonstrating knowledge of a password or secret key without directly providing the actual secret key), and have also been shown to have applications in fields as diverse as electronic voting ([6, 26]) and nuclear disarmament ([3]).

In this paper, we consider applications of zero-knowledge to *financial regulation*. Classical consideration of cryptography with respect to finance have focused almost exclusively on maintaining the privacy and security of transactions. In contrast, here we will consider the use of zero-knowledge to better balance the trade-off between *transparency and confidentiality* (for example, a hedge fund accurately and securely proving to its investors that its holdings satisfy certain risk constraints without revealing additional information on the holdings themselves). In this thesis, we will present and develop various zero-knowledge proof protocols for financial regulation. In particular, we utilize case studies and correspondence we collected with various professionals in the financial industry to further motivate these applications to financial regulation.

0.1 Structure of the Thesis

The paper is divided into two parts. Part I (Chapters 1-5) serves as a primer on zero-knowledge proofs. This section is expository and is more of a literature review, covering the history, formal definition, and classical applications of zero-knowledge proofs. Part II (Chapters 6-10) focuses on applications of zero-knowledge to the field of financial regulation. This section contains original research, as we we will motivate existing financial regulatory concerns, and present novel solutions to these problems via zero-knowledge proofs.

NB: As Part I is more expository in nature, it is intended to satisfy the Mathematics department thesis requirements. Part II, focused more on original research, is intended to satisfy the Computer Science department thesis requirements.

Part I

A Primer on Zero Knowledge

Chapter 1

Introduction to Zero Knowledge Proofs

In this chapter we provide an introduction to zero-knowledge proofs for the "zero-knowledge". In other words, assuming limited mathematical or theoretical computer science background, we attempt to provide a gentle introduction to zero knowledge proof protocols. We will cover a few abstract examples of zero knowledge and broadly explain why zero knowledge is useful with respect to cryptography and security applications.

1.1 Illustrative Examples

1.1.1 Ali Baba Cave

The following story is inspired by the tale of "Ali Baba and the Forty Thieves" from [34].

In the land of Arabia, there lived a old man named Ali Baba. One day, while roaming the bazaar, a thief grabbed Ali Baba's purse and dashed away. Ali Baba chased after the thief until the thief entered a cave. Ali Baba followed him into the cave, coming upon a fork in the cave. Ali Baba randomly chose the right fork and followed the path deeper into the cave, until he reached a dead-end, finding no sign of the thief. Inferring then that the thief must have then taken the left fork, Ali Baba retraced his steps back and took the left fork until he reached a dead-end, yet still found no sign of the thief. Assuming the thief slipped out through the entrance while he was exploring the right fork, Ali Baba went home.

However, the same thing happened the next day. Another thief stole his purse and ran into the same cave. Ali Baba randomly chose one of the forks, yet did not find the thief. Yet, this continued to happen; in all, 39 thieves stole his purse and each time Ali Baba selected the "wrong" fork in the road, finding no trace of the thieves.

Now, Ali Baba was suspicious; he suspected there was more at play here given the unlikelihood of being so unlucky and selecting the "wrong" fork every time. Hence, instead of going to the bazaar the next day, Ali Baba hid (under a pile of blankets) at the end of the right fork of the cave. Sometime later, a thief entered the cave and came into the right fork. His victim, audibly angry and yelling, began entering the right fork of the cave as well. To Ali Baba's amazement, the thief then whispered "Open Sesame" and a door appeared, linking the right and left forks of the cave. The thief then slipped through the door, the door closed, and the thief's victim appeared to find no trace of the thief.

But now, Ali Baba had learned the secret of the cave - the two forks of the cave, while appearing to be dead ends, were linked by a door that could only be opened by uttering the secret words "Open Sesame".

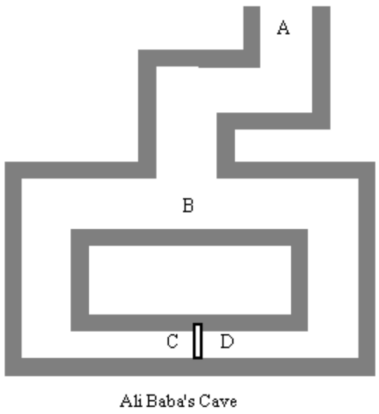


Figure 1.1: Depiction of Ali Baba cave from [30]

While this is an entertaining story, how is this at all related to zero-knowledge proofs? Motivated by the description of the cave above, consider the following scenario (adapted from [45]).

Consider now that Ali Baba wants to convince a reporter of the secret passageway. Understandably, the reporter is suspicious of Ali Baba's claim. However, Ali Baba cannot simply utter "Open Sesame" in front of the reporter since Ali Baba wishes to maintain the secret words. Therefore, how can Ali Baba convince the reporter without revealing the magic words? We can consider the following procedure (using location labels from image above).

- Reporter and Ali Baba both are outside the cave (A).
- Ali Baba enters the cave and chooses one fork of the cave and follows it (C or D).
- Reporter enters the cave and waits at the fork in the cave (B).
- Reporter then randomly announces which fork he wants Ali Baba to take.
- Ali Baba then (passing through the secret door if necessary) comes to point B along the announced path.

Now, if Ali Baba indeed has access to the secret passageway, he will be able to return along the announced path with 100% probability. In contrast, if Ali Baba were bluffing, he would only be able to return along the announced path with 50% probability (likelihood of Reporter selecting the same passage way as Ali Baba at random). Therefore, if the Reporter and Ali Baba repeat this process many times, and each time Ali Baba returns along the announced path, the Reporter can be very confident that Ali Baba indeed knows the secret words.

Note that through this procedure, Ali Baba is simply "proving" to the Reporter that *he knows the secret words* without revealing any additional information. This is the core idea of

a zero-knowledge proof protocol, in which a prover (Ali Baba) provides a proof to a verifier (Reporter) that does not reveal his knowledge (i.e. the password) to the Reporter nor reveal the fact of his knowledge to the world in general. Stated more simply, one verifies only the statement to be proved - nothing more.

1.1.2 Yao's Millionaires' Problem

We may consider another illustrative example, one perhaps more believable in terms of not requiring any mystical powers, and more relevant in terms of its tangential relation to finance (and hence the applications we will later consider).

In [1], the following famous problem was stated

Problem (Yao's Millionaires' Problem, [1])**.** *Alice has a private number a and Bob has a private number b, and the goal of the two parties is to solve the inequality $a \leq b$? without revealing the actual values of a or b, or more stringently, without revealing any information about a or b other than $a \leq b$ or $a > b$.*

This problem is termed the "Millionaires' Problem" in that the original formulation of the problem was to determine which of two millionaires is wealthier, and hence is a specific example of the problem stated above. Here, we consider the following discrete variant of the problem above. Namely,

Problem (Yao's Millionaires' Problem, Discrete Variant)**.** *Alice has a private number a and Bob has a private number b such that a, b are integers in $\{1, 2, \cdots n\}$ for $n \in \mathbb{N}$ where n is "small." They wish to determine whether $a = b$ without revealing any information about a or b other than $a = b$ or $a \neq b$.*

When we say n is small, we mean one could easily obtain say, n boxes or n envelopes. For example, with respect to Alice and Bob comparing their incomes, these numbers may correspond to tax brackets, i.e. Alice and Bob wish to determine if they fall within the same tax bracket (< 10 such buckets).

Consider the following procedure (adapted from [31]) as a solution to the problem stated above.

- Bob gets n lockable boxes and labels each box with a unique number from $\{1, 2, \cdots, n\}$. Note Bob can do this since n is "small".
- Bob keeps the key corresponding to the box with label b and throws away the rest of the keys. (Alice could verify that Bob only holds one key)
- Bob leaves the room and Alice enters the room. Alice then slips a small paper with "Yes" written into the (locked) box with label a. She then slips similar small papers with "No" written into each of the remaining $n - 1$ boxes.
- Alice leaves the room and Bob enters the room. Bob unlocks box b and retrieves the slip of paper in the box. He then removes all the boxes and just leaves the slip of paper.

- Alice enters the room and views the slip of paper as well.

Note if the slip of paper states "Yes" then both have the same value. However, if the paper says "No" then they must have different values. Note this process has the *zero knowledge property* in that neither Alice nor Bob gains any additional information on the other's value. In particular, if $a \neq b$, neither gains information on whether $a < b$ or $a > b$. See image below for an illustration of the protocol for $n = 4, a = 2, b = 3$.

Figure 1.2: Bob throws away keys to all boxes except Box 3 ($b = 3$)

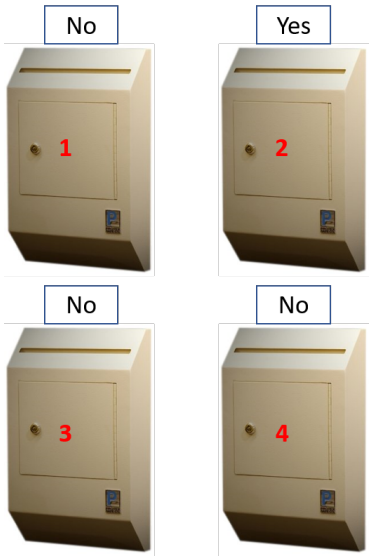


Figure 1.3: Alice slips "Yes" into Box 2 ($a = 2$), "No" into all other boxes

Figure 1.4: Bob opens Box 3, revealing a "No". Bob and Alice therefore conclude $a \neq b$

1.2 Zero Knowledge and Crypto

The previous examples served to provide a window into *zero-knowledge*. But how do these relate to cyryptography? We will review fundamental notions in crypto and explain why Zero Knowledge is useful.

1.2.1 Core Ideas in Cryptography

Cryptography is the study of methods for secure communication; in particular, cryptography is often focused on developing protocols that prevent third parties from reading private messages ([28]).

A familiar illustrative example is that of online passwords. Consider when one visits Gmail, Yahoo Mail, or any other e-mail client. Typically, one must enter a plaintext password, that is then verified by the e-mail client to allow access to one's messages. However, it is important to understand what is going on under the hood. The e-mail clients typically do not store plaintext passwords for its users, as a data breach would then be catastrophic. Rather, they store a *hash* of a client's password. Namely, each client has a hash function that transforms a user's provided plaintext (input) into a unique ciphertext (hash) password, one resembling a string of random characters (see image below).

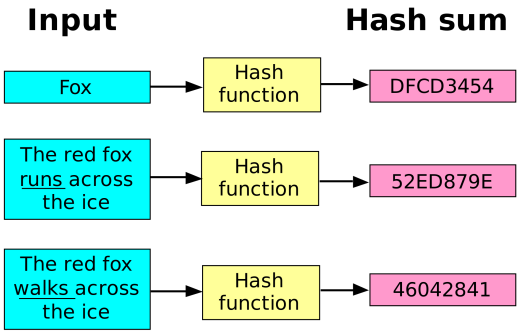


Figure 1.5: Depiction of a hash function from [11]

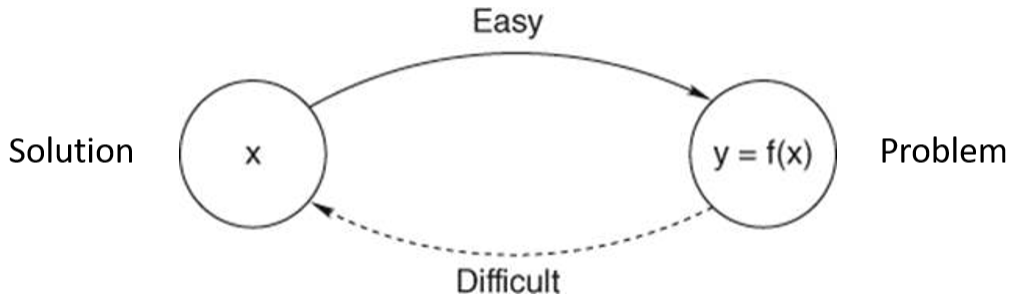
In particular, for security purposes, most clients choose a hash function that is *difficult to reverse* (see [28] for an exposition of such functions), i.e. it is computationally hard to deduce

the plaintext (input) from the ciphertext (hash), even if the specific hash function used is known.

Hence, a user interaction with an e-mail client usually works as follows. A user enters a plaintext password p. The client then applies a hash function f to p to yield a ciphertext c. The client then checks c against the ciphertext (hash) stored for the user. If they match, the user is authenticated.

Now, consider the role of third party adversaries here. Even if they get their hands on the ciphertext (hash) stored for a particular user, if the hash function is hard to reverse, the adversary will be unable to deduce the plaintext password. Hence, the adversary will essentially be stuck, keeping on guessing random plaintext passwords; given the space of possible passwords, the user's information will then likely remain secure.

We can extend these observations to cryptographic primitives and protocols in general. Namely, most crypto protocols are implicitly *one-way functions* i.e. given a (mathematical) problem, it "easy" to check if a particular answer x is a solution to the problem, whereas it is "difficult" to find the actual solution given just the problem. In other words, there exists a function f that given an answer x can easily return $f(x)$ i.e. whether x is a solution to the problem. But such a function f is hard to invert i.e. it is difficult to find x such that $f(x) =$ TRUE. See below for an illustration of the one-way function property (inspired by [22]).



When we say it is "easy" to verify a solution, we can check if the answer is a solution to the problem in *polynomial time*. In contrast, the problem in "difficult" to solve in that it requires *super-polynomial time* i.e. "not polynomial time" to discover a solution to the problem.

Hence, this one-way function property is implicit in the various cryptographic primitives that we will utilize and is the core building block on which many crypto protocols are based on.

1.2.2 How is Zero Knowledge Useful?

Now, the one-way function property shown above is extremely useful. Hence, it may seem like we're done. To transmit a message or prove our identity, we just submit a solution (that only we have) to a crypto problem. After all, the one-way property appears to provide a security guarantee that an adversary will be unable to discover a solution.

However, let us return to the e-mail client example from the previous section. Note that every time we log-in, we are submitting a solution (the plaintext) to the problem (discovering the text whose hash matches the stored ciphertext). Yet, in interacting with the e-mail client, we are directly providing the solution. This is a big problem. An adversary (whether it be a

third party snooping or an executive at the e-mail client who can see the plaintext coming in) who sees this plaintext then can mimic the user; they have all the information they need to login to the e-mail client as the user. Therefore, this protocol isn't as secure as we might hope.

Now, it is important to consider what our real goal here is as a user. Our primary aim is to simply convince the client that we *possess a solution*. In particular then, an optimal mechanism is one that convinces the client that we, as a user, *posses a solution* without revealing any additional information about the solution (i.e. the password).

This is precisely where zero-knowledge adds value in cryptography! The "without revealing any additional information" aligns precisely with the zero-knowledge property discussed in Section 2.1. In particular, zero-knowledge protocols, though not currently implemented in most e-mail clients, are extremely powerful in that they allow one to *demonstrate possessing a solution without revealing any information about the actual solution.*

1.2.3 An Example Protocol: Discrete Log

We now briefly describe an example of such a zero knowledge protocol using the discrete logarithm (example from [52]). This section assumes knowledge of basic number theory mod p.

Definition 1.1 (Discrete Log)**.** *Let p be a large prime and g be a generator of the multiplicative group of integers modulo p. The* discrete log of v with respect to generator g (or $\log_g v$) *is the unique x such that*

$$g^x = v \pmod{p}.$$

Note that since p is a large prime, even if any one is given the triple (g, v, p), it is (known to be) hard to compute the discrete log x. Using this, we may consider the following protocols.

Discrete Log Protocol without Zero Knowledge

Let (g, v, p) be public information. A Prover wishes to convince the Verifier that he possesses $x = \log_g v$. Consider the following protocol that does not utilize zero knowledge.

- Prover provides x to the Verifier.
- Verifier checks that $g^x = v \pmod{p}$, and accepts/rejects accordingly.

Note this protocol suffers from the same issues as discussed with respect to passwords with an e-mail client. Namely, any eavesdropper could catch x and impersonate the Prover later. Hence, we may instead consider a protocol utilizing zero knowledge.

Discrete Log Protocol with Zero Knowledge

Again the setup is identical. Let (g, v, p) be public information. A Prover wishes to convince the Verifier that he possesses $x = \log_g v$. Consider the following protocol that utilizes zero knowledge.

- Prover selects a random number $0 \leq r < p$ and sends the Verifier $h = g^r \pmod{p}$.

- Verifier sends a random bit $b \in \{0, 1\}$ to the Prover.
- Prover sends $a = r + bx \pmod{p}$ to Verifier.
- Verifiers checks that $g^a = hv^b \pmod{p}$, and accepts/rejects accordingly.
- Repeat above steps multiple times (to confirm acceptance).

Note the equality conditions checks out in that

$$g^a = g^{r+bx} = g^r g^{bx} = (g^r)(g^x)^b = hv^b \pmod{p}.$$

Therefore, if a Prover and Verifier engage in a few iterations of this (with at least some $b = 1$), the Verifier will be confident that the Prover indeed knows x. Namely, when $b = 0$ the Verifier is providing the discrete log of h, while if $b = 1$ the Verifier is providing the discrete log of hv.

This protocol is more secure in that it reveals no information about x to a potential eavesdropper. Note an eavesdropper can only possibly see the values h, b, a. Since solving the discrete log problem is difficult, the eavesdropper will be unable to deduce r from h. Therefore, if $b = 0$, an eavesdropper will only learn $r = a$ and will unable to deduce any information on x. Similarly, if $b = 1$, an eavesdropper will only learn $r + x = a$; since r is randomly drawn, they will be unable to deduce any information on x. The end result therefore is that the Verifier provides proof of possessing x without directly revealing x, and is hence an example of a zero-knowledge protocol.

In particular, even if an adversary watches an entire round of (h, b, a) triples being transmitted, he will not be able to mimic the Prover, since over many rounds the Verifier may select a different bit for the same h. Namely, the role of bit b is to prevent a *replay* attack, in which an adversary tries to mimic a Prover using a transcript of a prior communication. Even if the adversary provides the same h that the Prover provided (in the previous iteration), the Verifier may provide a different bit b in this round, and therefore the adversary will be unlikely to fool the Verifier over many iterations.

Therefore, the above protocol using discrete log serves as an illustration of and introduction to the power of zero-knowledge proof protocols. Throughout this paper, we will consider a number of existing zero-knowledge protocols, as well as design a few of our own.

1.3 The Notion of a "Proof"

The preceding sections served to provide general intuition on the term "zero-knowledge" in "zero-knowledge proofs". Yet, the second half of "zero-knowledge proofs", the notion of a "proof", may be unintuitive and require clarification. This discussion is adapted from a similar discussion in [22].

1.3.1 Static Object versus Interactive Process

In mathematics, proofs are traditionally viewed as *static objects*. Namely, a "proof" is often depicted as a fixed sequence consisting of statements that are either axiomatic or are derived

from previous statements via derivation rules. Hence, a proof in mathematics is often viewed as a fundamentally fixed object.

In contrast, the proofs we refer to here are those that are established through an interactive process. For example, consider the protocols that we've considered in this chapter, such as the Ali Baba cave and Discrete Log protocols. Note in each of these cases, a proof was established through the careful interaction of parties. Hence, as opposed to the traditional static nature of mathematical proofs (i.e. those that are "written"), the proofs here are of a dynamic nature (i.e. those that are established via an interaction).

Note, regardless, the ends goals are the same: to show a particular statement to be true. These static and interactive methods are each distinct and valid methods of proof. However, by the nature of cryptographic protocols and specifically zero-knowledge proofs, here we will typically be interested in proofs of a dynamic nature i.e. proofs that are established via an interaction.

1.3.2 Prover and Verifier

Since we are interested here in proofs as interactive processes, it is important to develop some language regarding the parties in such a proof system.

The notion of a prover is implicit in all discussion of proofs, whether mathematical or interactive: the *prover* is the (sometimes hidden or transcendental) entity providing the proof. In contrast, the notion of a *verifier* tends to be more explicit, as in both mathematical and interactive proofs, the proofs are defined in terms of the verification procedure.

In particular, when considering more static mathematical proofs, the verifier is the one actively "writing" the proof and hence verifying the truth of a particular statement. In contrast, the prover can be viewed as some transcendental entity (i.e. the god of mathematics that sets the axioms).

In an interactive proof setting, the proof is determined via an interaction between the prover and verifier. For example, in the Ali Baba cave example, Ali Baba is the prover providing the proof that he knows the secret words, while the Reporter is the verifier trying to establish the correctness of Ali Baba's claim. Hence, it is through their interaction that the prover is able to convince the verifier of his claim. In particular, this interaction between provers and verifiers is the general setup of most cryptographic (including zero-knowledge) protocols.

It is also important to realize an inherently "distrustful attitude" towards the prover (by the verifier) that underlies any proof system. After all, if the verifier trusts the prover, no proof would otherwise be needed. Hence, taking into account the potential for a malignant "prover" is critical and will be central to informing the formal definition of (secure) zero-knowledge proof protocols in the next chapter.

Chapter 2

Formal Definition of Zero Knowledge Proofs

While the previous chapter served to build intuition for zero-knowledge proofs, in this chapter we will present a more formal definition of zero-knowledge. We will also return to the example of the discrete log, to show that the protocol presented in the previous chapter satisfies the formal framework we develop here. Lastly, we present the history of the development and applications of zero-knowledge proofs.

2.1 Formal Definition

Before we present a formal definition with notation, it is helpful to have an informal presentation of the characteristics we wish a zero-knowledge proof to have.

2.1.1 An Informal Definition

Definition 2.1 (Zero-Knowledge Proof, Informal Formulation (see [53]))**.** *A zero-knowledge proof is an interactive proof of a statement involving a Prover and Verifier satisfying the following three properties*

- ***Completeness****: if the statement is true, an honest Verifier will be convinced of the truth of the statement by an honest Prover.*
- ***Soundness****: if the statement is false, no cheating Prover can convince an honest Verifier that the statement is true, except with some small probability.*
- ***Zero-Knowledge****: if the statement is true, no (honest or cheating) Verifier learns anything other than the fact the statement is true.*

Hence, these are the three properties we require for a zero-knowledge proof. Note the first two properties, *completeness* and *soundness* are more characteristics of interactive proof systems in general (as hinted at the close of the previous chapter). It is, unsurprisingly, the zero-knowledge that is key here. We will now develop each of these properties in more detail.

2.1.2 Interactive Proof Systems

In an interactive proof system, a proof arises from the interaction between a Prover and a Verifier. Namely, a Prover seeks to convince the Verifier that their solution is valid.

For a given problem, define $\mathcal{L}$ as the set of true statements (restricted to a particular language). In particular, for a statement $x \in \mathcal{L}$, we would want the interaction between the Prover and Verifier to *Accept* x with high probability, and for $x \notin \mathcal{L}$, we would want the interaction between the Prover and Verifier to *Accept* x with low probability. In particular, define $\langle P, V\rangle(x)$ to denote the output of the interaction between the Prover and Verifier on x. If $\langle P, V\rangle(x) = 1$, this denotes an *Accept*. With this notation, we define an interactive proof system as follows

Definition 2.2 (Interactive Proof System (see [4, 22]))**.** *A pair of interactive algorithms $\langle P, V\rangle$ is an interactive proof system for a language $\mathcal{L}$ if V runs in polynomial time and*

- ***Completeness**: For every $x \in \mathcal{L}$,*

$$Pr[\langle P, V\rangle(x) = 1] \geq \frac{9}{10}$$

- ***Soundness**: For every $x \notin \mathcal{L}$ and any algorithm B,*

$$Pr[\langle B, V\rangle(x) = 1] \leq \frac{1}{10}$$

Note these notions of *completeness* and *soundness* align with those stated earlier. The actual values ($\frac{9}{10}$ and $\frac{1}{10}$) in the definition are arbitrary, as the following remarks will clarify.

Completeness

In this definition, $Pr[\langle P, V\rangle(x) = 1]$ denotes the probability that an interaction between an honest prover P and an honest verifier V correctly *accepts* an $x \in \mathcal{L}$. Note *honesty* here denotes a party that truly follows the protocol correctly. Ideally, we would like $Pr[\langle P, V\rangle(x) = 1] = 1$, but this is not always the case in an interactive proof system. However, almost all the protocols we consider in this paper will be from the subset of interactive proof systems in which $Pr[\langle P, V\rangle(x) = 1] = 1$. Therefore, in general, we will not have to worry about any type of *completeness error* here. To conclude, throughout this paper, completeness will usually reduce to a honest prover and honest verifier confirming a solution $x \in \mathcal{L}$ with 100% probability (*perfect completeness*).

Soundness

For a system to be sound, we want an $x \notin \mathcal{L}$ to be accepted with very low probability. In particular, we take into account any cheating prover. Therefore, we consider any algorithm by the prover, hence *any algorithm* B, to account for this potential nefarious behavior.

In addition, the value of $\frac{1}{10}$ is arbitrary. Note the verifier V is required to be a polynomial-runtime algorithm while we place no such restrictions on the prover B. Therefore we can consider running the above interaction $p(n)$ (a polynomial) number of times, *rejecting* only

if every iteration outputs a *reject*. Since the product of two polynomials is a polynomial, the runtime of the verifier V (in the $p(n)$ iteration interactive proof system) is polynomial as well. Note that the soundness error in this system is then

$$Pr[\langle B, V\rangle(x) = 1] \leq \left(\frac{1}{10}\right)^{p(n)} = \frac{1}{10^{p(n)}}$$

and hence can be made arbitrarily small. (Note we don't worry about changes to completeness error here since the protocols we will consider here all exhibit perfect completeness). Therefore, we have an interactive proof system with an arbitrarily small soundness error.

2.1.3 Zero-Knowledge Property

The previous section provided more details on the completeness and soundness properties. Now we turn to indisputably the most important property of zero-knowledge proofs, the zero-knowledge property. To this point, our descriptions of this property i.e. "not revealing any additional information" have been rather wishy-washy and hence we develop some formalism here.

The Simulation Paradigm

Definition 2.3 (Computational Zero-Knowledge, see [4, 22])**.** *Let $\langle P, V\rangle$ be an interactive proof system for some language $\mathcal{L}$. We say that $\langle P, V\rangle$ is computational zero-knowledge (or just zero-knowledge) if for every efficient (polynomial time) verifier strategy V^*, there exists an efficient (polynomial time) probabilistic algorithm S^* (known as the simulator) such that for every $x \in \mathcal{L}$, the following random variables are computationally indistinguishable:*

- *The output of V^* after interacting with P on input x*
- *The output of S^* on input x.*

Essentially, we have zero-knowledge if we can show the verifier does not gain anything from the interaction; no matter what algorithm V^* uses, whatever he learned as a result of interacting with the prover, he could have just as equally learned by simply running the standalone algorithm S^* on the same input. Note that since S^* does not have access to the solution, this implies that indeed the verifier gained no additional information.

This construct (see [22, 23] for more details) is known as the *simulation paradigm*, which postulates that whatever a party can do by itself cannot be considered a gain from interaction with the outside. Hence, to show zero-knowledge, it is enough to show that for every cheating verifier there is some simulator that, given only the statement to be proved (and no access to the prover), can produce a *transcript* that "appears" like an interaction between the honest prover and the cheating verifier.

While this notion of a simulator may seem a bit abstract, the return to the discrete log example in the next section will hopefully serve to provide more intuition and greater clarity.

***Computational Indistinguishability**

Ax expressed in the previous definition, for practical purposes with respect to zero-knowledge, we will often concern ourselves with *computational indistinguishability*; namely, that two distributions are "very close" as opposed to being identical. More formally,

Definition 2.4 (Computational Indistinguishability, [41, 22])**.** *Let $\{X_n\}_{n\in\mathbb{N}}$ and $\{Y_n\}_{n\in\mathbb{N}}$ be two ensembles (sequences of probability distributions). We say $\{X_n\}_{n\in\mathbb{N}}$ and $\{Y_n\}_{n\in\mathbb{N}}$ are computationally indistinguishable if for any non-uniform probabilistic polynomial time algorithm D, there exists a negligible function ϵ such that for every $n \in \mathbb{N}$,*

$$|Pr_{t\leftarrow X_n}[D(t) = 1] - Pr_{t\leftarrow Y_n}[D(t) = 1]| \leq \epsilon(n).$$

In other words, any non-uniform probabilistic polynomial time decider cannot tell apart a sample from X_n and Y_n. Also, note we can use any indexing set for the ensemble (not only $\mathbb{N}$).

If two ensembles $\{X_n\}_{n\in\mathbb{N}}$ and $\{Y_n\}_{n\in\mathbb{N}}$ are computationally indistinguishable we write,

$$\{X_n\}_{n\in\mathbb{N}} \approx \{Y_n\}_{n\in\mathbb{N}}.$$

In particular, with respect to zero-knowledge, we have the condition

$$\{\langle P, V^*\rangle(x)\}_{x\in\mathcal{L}} \approx \{S^*(x)\}_{x\in\mathcal{L}}.$$

NB: Verifying computational indistinguishability formally is usually quite tedious. Therefore, in this paper, we will usually gloss over checking this result as it is simply an exercise in algebra.

2.2 A Return to the Discrete Log Protocol

We now return to the discrete log protocol from Section 1.2.3 to better illustrate the properties of zero-knowledge proofs detailed above. This serves to hopefully make the definitions from the prior section seem a bit less abstract.

2.2.1 The Setup

We define U as the universe of statements in the language we restrict ourselves to here. Namely any statement in U is of the form:

$$\text{[Prover Name] knows the discrete log of [}v\text{] (mod } p\text{)}$$

where [Prover Name] and $[v]$ are variable.

Hence, as described earlier, a zero-knowledge interactive proof system $\langle P, V\rangle$ for the set of true statements $\mathcal{L} \subset U$ is as follows

Definition 2.5 (Zero-Knowledge Discrete Log Protocol, Section 2.2.3)**.** *A Prover wishes to convince the Verifier that he possesses $x = \log_g v$. Let (g, v, p) be public information.*

- *Prover selects a random number* $0 \leq r < p$ *and sends the Verifier* $h = g^r \pmod{p}$.
- *Verifier sends a random bit* $b \in \{0, 1\}$ *to the Prover.*
- *Prover sends* $a = r + bx \pmod{p}$ *to Verifier.*
- *Verifiers checks that* $g^a = hv^b \pmod{p}$*, and accepts/rejects accordingly.*
- *Repeat above steps multiple times (to confirm acceptance).*

We now verify this is a zero-knowledge proof system. (Note Verifier clearly runs in polynomial time.)

2.2.2 Completeness

Completeness is usually quite straight-forward. Here we are dealing with an honest prover (who knows x) and an honest verifier. Therefore, since the prover knows x, the verifier will verify the computation

$$g^a = g^{r+bx} = g^r g^{bx} = g^r (g^x)^b = hv^b$$

with 100% probability in each iteration. Therefore, if we call the statement $\ell \in \mathcal{L}$, we have

$$Pr[\langle P, V \rangle(\ell) = 1] = 1$$

and hence have perfect completeness. In other words, an honest prover and honest verifier confirm a solution with 100% probability.

2.2.3 Soundness

Soundness is a bit tricker. Consider a cheating Prover B (that does not know x) interacting with an honest verifier V. This Prover must submit some h to the Verifier. Hence, the Prover can attempt to cheat via one of two ways:

Case 1: Prover B picks a random r and sends $h = g^r \pmod{p}$ to the Verifier

Note in this case, if the Verifier picks $b = 0$, the Prover B can just send $a = r$ and be accepted. However, if the Verifier picks $b = 1$, the Prover B will be stuck. Namely, the problem is then to find the discrete log of hv. But

$$\log_g(hv) = \log_g(h) + \log_g(v) = r + \log_g(v) = r + x.$$

Since the Prover B already "knows" r, computing the discrete log of hv reduces to the Prover knowing the discrete log of v (i.e. x). Therefore, if $b = 1$, the best a Prover B can do is guess. Since the discrete log of a value is unique (with respect to a particular generator g), probability of success in guessing is at most some $\frac{c}{p}$ for c a constant such that $c << p$. Therefore, in this case, the probability of cheating Prover B being accepted in a round is

$$\frac{1}{2} + \frac{1}{2}\left(\frac{c}{p}\right).$$

Case 2: Prover B picks and sends a random h to the Verifier for which he does not know the discrete log r

Now, in this case, if the Verifier picks $b = 0$, the Prover B is out of luck since he doesn't know the discrete log of h (i.e. r). Therefore, in this case, the best a prover can do is guess, with a probability of success in guessing of some $\frac{c}{p}$. The other possibility is if the Verifier picks $b = 1$. Note the Prover B could have been clever here and picked $h = g^s v^{-1}$ for some random $0 \leq s < p$. Then, if the Verifier picks $b = 1$, the Prover simply submits $a = s$, and the Verifier confirms

$$g^a = g^s = (g^s v^{-1})v = hv.$$

Therefore, the probability of cheating Prover B being accepted in a round is at most (i.e. maximized when Prover picks some $h = g^s v^{-1}$)

$$\frac{1}{2} + \frac{1}{2}\left(\frac{c}{p}\right).$$

Combining the two cases above, we have that a cheating Prover B will trick an honest verifier with probability at most

$$\frac{1}{2} + \frac{1}{2}\left(\frac{c}{p}\right) < 1$$

for $c << p$. Over many iterations, this probability can be made arbitrarily small, i.e

$$\left[\frac{1}{2} + \frac{1}{2}\left(\frac{c}{p}\right)\right]^k \to 0 \text{ as } k \to \infty.$$

Namely, we have

$$Pr[\langle B, V\rangle(\ell) = 1] < 2^{-p(n)}$$

for any polynomial $p(n)$ (adjusting number of iterations accordingly) and $\ell \notin \mathcal{L}$, and therefore have soundness as desired. In other words, a cheating prover will be unable to trick an honest verifier, except with some small probability.

2.2.4 Zero-Knowledge

Let V^* be an arbitrary Verifier strategy (not necessarily honest). We wish to show V^* learns no additional information from an honest prover P.

Per the discrete log protocol described above, we may think of V^* as composed of two functions:

- $V_1(h)$ outputs the bit b that the Verifier V^* chooses after the Prover P submits h
- $V_2(h, a)$ is whatever Verifier V^* outputs after seeing Prover P's response a to the bit b.

Both V_1 and V_2 are clearly efficiently computable. We now need to come up with an efficient simulator S^* that is a standalone algorithm that will output a distribution indistinguishable from the output V^*. The simulator S^* will work as follows (based on [43]):

- Pick a random bit $b' \in \{0, 1\}$. Select a random number $0 \leq r' < p$.

- Send $h = g^{r'}v^{-b'}$.
- Let $b^* = V_1(h)$ (bit output by V^*). If $b' \neq b^*$, go back to Step 1.
- (If $b' = b^*$) Send $a = r'$, and hence have final output of $V_2(h, a)$.

Note this simulator uses V^*. This is necessary since V^* is not necessarily honest and hence we have no guarantees on its strategy. We assert this is a valid simulator with the following claims (inspired and adapted from [4]).

Claim 2.6. *The distribution of h computed by S^* is identical to the distribution of h chosen by the Prover P.*

Proof. In each case h is a random power g^i. Note even in the case, $h = g^{r'}v^{-b'}$, we have

$$h = g^{r'}v^{-b'} = g^{r'-b'x},$$

since r' is randomly drawn and the quantity -$b'x$ is fixed, h is therefore still a random power g^i (i.e. value h computed by S^* is statistically independent of the bit b'). □

Claim 2.7. *With probability at least $\frac{1}{2}$, $b' = b^*$.*

Proof. This follows as a corollary of the previous claim. Since the distributions of h are identical, h gives no information about the choice of b'. In particular, V^* behavior is then unchanged, and therefore if follows $b' = b^*$ with probability $\frac{1}{2}$. □

Together these two claims imply that in expectation S^* only invokes V_1 and V_2 a constant number of times (since every time it goes back to step 1 with probability at most $\frac{1}{2}$), and therefore, S^* is efficient as desired (polynomial-time). It remains to check the following claim.

Claim 2.8. *Conditioning on $b' = b^*$ and the value h computed in Step 2, the value "a" computed by S^* is identical to the value that the Prover P sends when P's first message is h and V^*'s response is b^*.*

Proof. The proof follows from a direct calculation. The value a sent by the Prover P is the discrete log of h if $b^* = 0$ and of hv if $b^* = 1$. Similarly, since

$$g^{r'} = hv^{b'}$$

and $a = r'$, the value a computed by the simulator S^* is the discrete log of h if $b' = 0$ and of hv if $b' = 1$. Since the same respective discrete logs are being computed when $b' = b^*$, this proves the desired claim. □

Hence, this claim implies that the output of S^* is in fact identical (in a distributional sense) to the output of V^* in a true interaction with the Prover P. Namely, Claim 1 states that the simulator S^* gives an identical distribution on h as an honest prover P. Claims 2 and 3 imply that, conditioning on a value h, the simulator and interactive system both yield an identical bit b (or b^*) from the verifier and identical value a.

Consider this from another view. In the interactive system, in one iteration on a given statement $\ell \in \mathcal{L}$, the information transferred (the outputs) are $(h, b, a, \textbf{Accept})$. Analogously, in the simulator, in one interaction on a given statement $\ell \in \mathcal{L}$, the outputs are $(\overline{h}, \overline{b}, \overline{a}, \textbf{Accept})$ (we assume simulator only outputs the last h value). For us to have zero-knowledge, this means that over many iterations, the distributions

$$\{(h, b, a, \text{Accept})\}_{n\in\mathbb{N}} \approx \{(\overline{h}, \overline{b}, \overline{a}, \text{Accept})\}_{n\in\mathbb{N}}$$

(to be equal). The claims above give us this result.

Therefore, we have demonstrated the existence of an efficient simulator S^* such that for every $\ell \in \mathcal{L}$:

- The output of V^* after interacting with P on input ℓ
- The output of S^* on input ℓ

are equal, and hence we have demonstrated the zero-knowledge property.

NB: This section served to verify the properties of a zero-knowledge proof in excruciating detail. Throughout the rest of this paper, we will tend to not be as rigorous in verifying these definitions. More usual than not, the interesting part is the zero-knowledge protocol itself, as the verification of the properties is simply an exercise in algebra.

2.3 History of Zero-Knowledge

We provide a brief overview of the history of the zero-knowledge and motivate our discussion for the remainder of Part I of the paper.

2.3.1 Origin

Zero-knowledge proofs are a relatively recent invention (see [53]). They were first conceived in a 1985 paper by Goldwasser, Micali and Rackoff (see [24] for more details). In particular, the paper introduced a hierarchy of interactive proof systems, and were the first to conceive of the notion of "knowledge complexity", a measurement of the knowledge transferred from the prover to the verifier. Specifically, they were interested in "0 knowledge complexity", hence zero-knowledge, and presented the first known zero-knowledge proof, that of deciding quadratic residues and non-residues mod n.

In particular, the paper also introduced the idea of a simulator to verify zero-knowledge as we did earlier in this section.

2.3.2 Applications in Crypto

Since the original 1985 paper, there has been a flood of various zero-knowledge protocols invented for cryptographic applications. Many of these are identification schemes similar to the discrete log example we've discussed. In Chapter 3, we'll present two of these classical applications. The first, Fiege-Fiat-Shamir, will be of a similar number theory flair as the discrete log protocol. In contrast, the second, Graph Isomorphism, will be of a considerably different flavor as it will be firmly based in graph theory.

2.3.3 Connections to Complexity

Zero-knowledge is of particular interest to theorists in its implications to complexity theory. Namely, Goldreich, Micali, and Widgerson (see [21]) demonstrated a zero-knowledge proof system for the NP-complete graph 3-coloring problem. Since every problem in NP can be reduced to the 3-coloring problem, this yields the surprising result that all problems in NP have zero-knowledge proofs. In Chapter 4, we will demonstrate these complexity results and discuss their significance.

2.3.4 Other Developments

There has also been the development of various variants of the simple zero-knowledge proof protocols described above. These include multi-prover interactive proof systems (see [37]), concurrent zero-knowledge (see [14]), and non-interactive zero knowledge (see [15]). While each of these are important developments and can be described in depth, they are well beyond the scope of this paper.

Of particular interest is a very recent application to cryptocurrencies. For example, Z-cash, a new cryptocurrency launched in 2016 (see [54]), makes use of zero-knowledge proofs as the backbone for the security of transactions involving the cryptocurrency. See below for a schematic.

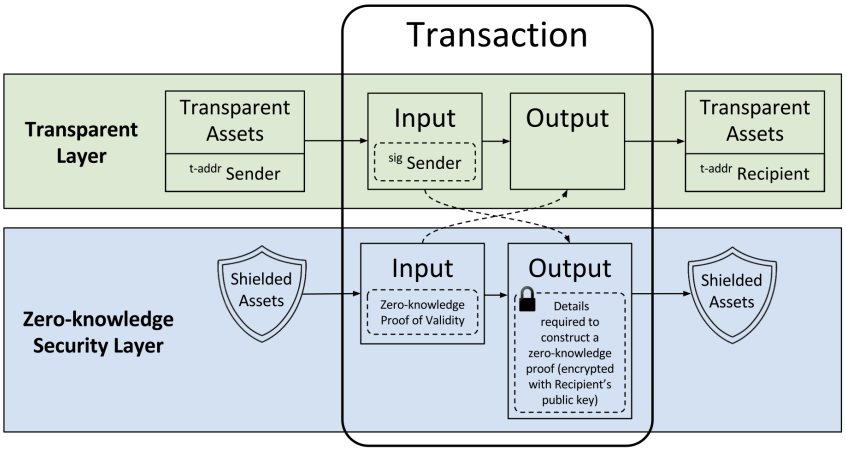


Figure 2.1: Depiction of a Z-cash transaction utilizing Zero-Knowledge proofs from [54]

What is perhaps most intriguing are applications of zero-knowledge that seem to have no apparent relation to classical cryptography. For example, there has been much recent work in the use of zero-knowledge proofs to potentially validate electronic voting. In particular, there has been much discussion in recent years to move voting to an online system due to low turnout. The major arguments against this are potential security issues and greater likelihood of tampering. The use of zero-knowledge proofs may potentially prevent some of these types of vote tampering (see [6] for an overview).

In addition, in perhaps the most surprising of all applications, researchers have demonstrated a zero-knowledge protocol for nuclear warhead verification that may have applicabil-

ity to future nuclear disarmament talks (see [3]).

In Chapter 5, we will discuss these exotic applications of zero-knowledge to voting and nuclear disarmament in more detail.

Chapter 3

Classical Zero-Knowledge Applications

In this chapter we provide two classical examples of zero-knowledge protocols, the *Fiege-Fiat-Shamir Authentication Protocol* and a *Zero-Knowledge Protocol for Graph Isomorphism*. The goal of this chapter is to provide a flavor of the types of zero-knowledge proof protocols that exist.

3.1 Fiege-Fiat-Shamir

The Fiege-Fiat-Shamir protocol is a zero-knowledge authentication scheme. Much like the discrete log example in the prior chapter, the basis for the protocol is based in number theory and modular arithmetic. However, while the discrete log problem utilizes the difficulty in computing the discrete log modulo an arbitrary prime p, the Fiege-Fiat-Shamir protocol utilizes the difficulty in computing square roots modulo a composite that is hard to factor. We present the protocol and verify it is a zero-knowledge proof.

3.1.1 The Protocol

The protocol consists of two stages: a key-generation stage and an authentication stage (see [16] for the original presentation of the protocol).

Definition 3.1 (Fiege-Fiat-Shamir Protocol, Key Generation (see [16]))**.** *A Prover P constructs a private and public key as follows.*

- *A trusted 3^{rd} party provides a Blum Integer $n = p \times q$ [where $p, q = 3 \pmod 4$ and p, q are large primes] and publicizes n.*
- *For each $1 \leq i \leq k$ ($k << 2^k << n$),*
  - *Prover P selects (at random and independent) s_i co-prime to n*
  - *Prover P selects (at random and independent) $u_i \in \{-1, 1\}$*
  - *Prover P computes $v_i = u_i s_i^2 \pmod n$.*
- *Prover P publishes public key and keeps private key secret:*

– *Public:* $\{v_1, v_2, \cdots, v_k\}$
– *Private:* $\{s_1, s_2, \cdots, s_k\}$

Definition 3.2 (Fiege-Fiat-Shamir Protocol, Authentication (see [16]))**.** *A Verifier V authenticates a Prover P as follows.*

- *Prover P selects random r co-prime to n and random sign $s \in \{-1, 1\}$, and computes $X = sr^2 \pmod{n}$ (i.e. $X = r^2$ or $x = -r^2$). Prover P sends "X" to Verifier V.*
- *Verifier V sends a random boolean vector $B = (b_1, b_2, \cdots, b_k)$ to Prover P.*
- *Prover P sends the value*
$$Y = r \prod_{i=i}^{k} s_i^{b_i} \pmod{n}$$
to Verifier V.
- *Verifiers V* accepts *if and only if*
$$Y^2 = \pm X \prod_{i=1}^{k} v_i^{b_i} \pmod{n}.$$
- *Repeat above steps multiple times (until desired confidence level).*

3.1.2 Verifying Zero-Knowledge Properties

We now briefly outline how the protocol satisfies the properties of zero-knowledge proofs.

Completeness

Consider an honest Prover P. Then an honest Verifier V will verify

$$Y^2 = \left(r \prod_{i=i}^{k} s_i^{b_i} \right)^2 = r^2 \prod_{i=i}^{k} (s_i^2)^{b_i} = (s^{-1}X) \prod_{i=i}^{k} (u_i^{-1} v_i)^{b_i} =$$

$$\left(s^{-1} \prod_{i=i}^{k} u_i^{-1} \right) \left(X \prod_{i=1}^{k} v_i^{b_i} \right) = \pm X \prod_{i=1}^{k} v_i^{b_i} \pmod{n}$$

and hence the protocol is perfectly complete.

Soundness

Now, consider the setting of a cheating Prover B^* trying to impersonate (i.e. does not have the private key of) an individual with public key $\{v_1, v_2, \cdots, v_k\}$. To convince an honest Verifier V, cheating Prover B^* must select an X and Y such that

$$X = \pm Y^2 \prod_{i=1}^{k} v_i^{-b_i} \pmod{n}.$$

In particular, since Prover B^* selects X before the Verifier V's choice of $B = (b_1, b_2, \cdots, b_k) \in \{0,1\}^k$ and Y after the selection of B, let us call the conditional choice Y_B. We now proceed by casework.

Case 1: There is at least one $B \in \{0,1\}^k$ for which Y_B fails to satisfy

$$X = \pm Y_B^2 \prod_{i=1}^{k} v_i^{-b_i} \pmod{n}.$$

In this case, the verification will fail with probability at least $\frac{1}{2^k}$.

Case 2: For every $B \in \{0,1\}^k$, Y_B satisfies

$$X = \pm Y_B^2 \prod_{i=1}^{k} v_i^{-b_i} \pmod{n}.$$

Now, consider $B_0 = (0, 0, \cdots, 0)$ (all 0's) and $B_1 = (1, 0, \cdots, 0)$ (only $b_1 = 1$). Then we have

$$X = \pm Y_0^2 = \pm Y_1^2 v_1^{-1} \pmod{n}$$

$$v_1 = \pm (Y_1/Y_0)^2 \pmod{n}$$

which implies Prover B^* then knows a valid s_1 (i.e. if B can compute Y_1 and Y_0, can compute s_1). However, we can repeat this analysis for any $B_0 = (0, 0, \cdots, 0)$ (all 0's) and $B_i = (0, \cdots, 1, \cdots, 0)$ (only $b_i = 1$) to determine that Prover B^* must consequently know a valid s_i. But then, this implies Prover B^* has a valid private key $\{s_1, s_2, \cdots, s_k\}$ for the public key. This contradicts out assumption that the prover does not have a valid private key.

Therefore, combining the cases, verification must fail with probability at least $\frac{1}{2^k}$. Since by assumption $2^k << n$, we may run the authentication $c2^k$ times (c a constant), and hence make the soundness error

$$\left(1 - \frac{1}{2^k}\right)^{c2^k} = \left[\left(1 - \frac{1}{2^k}\right)^{2^k}\right]^c < \left(\frac{1}{2}\right)^c = \frac{1}{2^c}$$

arbitrarily small.

NB: Note the error probability of $\frac{1}{2^k}$ shown here is very loose. See [16] for a tighter bound.

Zero-Knowledge

We present a simulator to demonstrate the zero-knowledge property with an arbitrary Verifier strategy V^*. A Verifier strategy V^* can be thought of as two functions:

- $V_1(X)$ outputs the boolean vector $B \in \{0,1\}^k$ that V^* chooses in response to X from the Prover P
- $V_2(X,Y)$ is whatever Verifier V^* outputs after seeing Prover P's response Y to the boolean vector B.

Both V_1 and V_2 are clearly efficiently computable. We now need to come up with an efficient simulator S^* that is a standalone algorithm that will output a distribution indistinguishable from the output V^*. The simulator S^* will work similarly to the discrete log simulator:

- Pick a random bit vector $B' \in \{0,1\}^k$. Select random r' co-prime to n and random sign $s' \in \{-1,1\}$. Compute $W = \prod_{i=1}^{k} v_i^{b'_i}$.
- Send $X = s'(r')^2 W^{-1}$.
- Let $B^* = V_1(X)$ (bit vector output by V^*). If $B' \neq B^*$, go back to Step 1.
- (If $B' = B^*$) Send $Y = r'$, and hence have final output of $V_2(X,Y)$.

The proof that this simulator yields an indistinguishable output distribution is identical to that of the discrete log example. Hence we omit the proof. Note that when $B' = B^*$, the verification checks out,

$$Y^2 = (r')^2 = (s')^{-1}(X)(W) = (s')^{-1} X \prod_{i=1}^{k} v_i^{b'_i} =$$

$$(s')^{-1} X \prod_{i=1}^{k} v_i^{b^*_i} = \pm X \prod_{i=1}^{k} v_i^{b^*_i} \pmod{n}.$$

In addition, since $B' = B^*$ with probability $\frac{1}{2^k}$, in expectation S^* only invokes V_1 and V_2 a constant number of times (note $2^k << n$, and is hence treated as a constant). Therefore, S^* is an efficient simulator.

3.1.3 Key Points of the Protocol

We cover a few critical aspects of the protocol. The specific aspects we discuss are shared/desired by a number of other cryptographic protocols (not necessarily zero-knowledge).

Public and Private Keys

The concept of distinct public and private keys is a key idea in *public key or asymmetric* cryptography protocols, in which public keys are disseminated widely and private keys are known only to the owner. Most widely used protocols take advantage of the distinction between public and private keys. This Fiege-Fiat-Shamir protocol in particular is a subset of public key cryptography protocols knowns as *identification schemes*, essentially identity verification protocols.

For an exposition of other public key cryptographic protocols, such as digital signatures, see [28].

Blum Integers

Note this protocol made use of a special construct known as a Blum integer.

Definition 3.3 (Blum Integer, [27])**.** *A Blum Integer n is an integer*

$$n = pq,$$

where p, q are very large primes such that $p, q = 3 \pmod 4$.

Blum integers are widely utilized in cryptography. In particular, it is considered extremely difficult to find square roots modulo a Blum integer, hence their usage. Now, consider the following property of the Blum Integers.

Property. *For every Blum integer n, though -1 is not a quadratic residue* (mod n)*, the jacobi symbol*

$$\left(\frac{-1}{n}\right) = 1.$$

This serves to explain why we choose flip signs at random in our key generation protocol. Namely by allowing v_i to be either plus or minus a square integer modulo a Blum integer, we make sure v_i ranges over all the numbers (co-prime to n) with Jacbobi symbol $+1 \pmod n$. Thus, there must an "exist" an s_j for any choice of "reasonable" v_j (i.e. Jacobi symbol $+1$).

Parallel Verification

One intriguing aspect of this protocol in that it appears to verify k values simultaneously, hence making it more difficult to "cheat" in a given round. In particular, think back to the discrete log example. In order to reach a desired threshold of error, we required multiple iterations and hence an excessive number of rounds of interaction. Therefore, a desired property of protocols is often *parallel verification*, essentially the prover and verifier run k independent copies of the basic protocol in parallel, and the verifier accepts only if all of the copies accept. The Fiege-Fiat-Shamir is one of a few *zero-knowledge parallel verification algorithms*; in many other instances, parallel repetition compromises the zero-knowledge property.

3.2 Graph Isomorphism

While the previous examples we considered were all based in Number Theory, the protocol we consider here is of a geometric flavor. We first introduce the notion of graph isomorphism.

3.2.1 Brief Review of Graph Theory

Definition 3.4 (Graph, undirected)**.** *We define a graph $G = (V, E)$ as a set of vertices*

$$V = \{v_1, v_2, \cdots, v_n\}$$

and a set of edges

$$E = \{e_1, e_2, \cdots, e_m\}$$

where every $e \in E$ is of the form (v_i, v_j) i.e. an unordered pair of distinct vertices ($v_i \neq v_j$). (We also assume there can be at most one edge between any given pair of vertices).

Now, let the notation S_n denote the set of permutations of size n (i.e. automorphism on $\{1, 2, \cdots, n\}$). Utilizing this, we have the following notion of isomorphism:

Definition 3.5 (Graph Isomorphism)**.** *Let $G = (V, E)$ and $G' = (V', E')$ be two graphs such that $|V| = |V'| = n$. Label the vertices $V = \{v_1, v_2, \cdots, v_n\}$ and $V' = \{v'_1, v'_2, \cdots, v'_n\}$. We say G and G' are isomorphic (or $G \sim G'$) if there exists a permutation $\sigma \in S_n$ such that*

$$(v_i, v_j) \in E$$

if and only if

$$(v'_{\sigma(i)}, v'_{\sigma(j)}) \in E'.$$

Namely, if there exists a permutation σ of the vertices of G such that the edges of G and G' precisely coincide, we then say $G \sim G'$ and $\sigma(G) = G'$. For an illustrative example, consider the depiction below.

Typically, when confronted with a graph isomorphism problem, we are dealing with extremely large graphs consisting of many vertices and many edges. Therefore, checking all $n!$ permutations of size n is computationally infeasible to check for isomorphism. In addition, the graphs G and G' presented should be feasibly "potentially" isomorphic in the sense that they have identical vertex degree distributions as a precondition (otherwise the problem would be trivial). We now present a zero-knowledge protocol for graph isomorphism and shows it verifies the zero-knowledge proof properties.

3.2.2 The Protocol

In this language $\mathcal{L}$, statements are of the form

$$G_0 \text{ and } G_1 \textbf{ are} \text{ isomorphic.}$$

Say a Prover P possesses a proof, i.e. a permutation $\sigma \in S_n$ that graphs G_0 and G_1 are isomorphic. We have the following zero-knowledge protocol (see [21] for the original presentation).

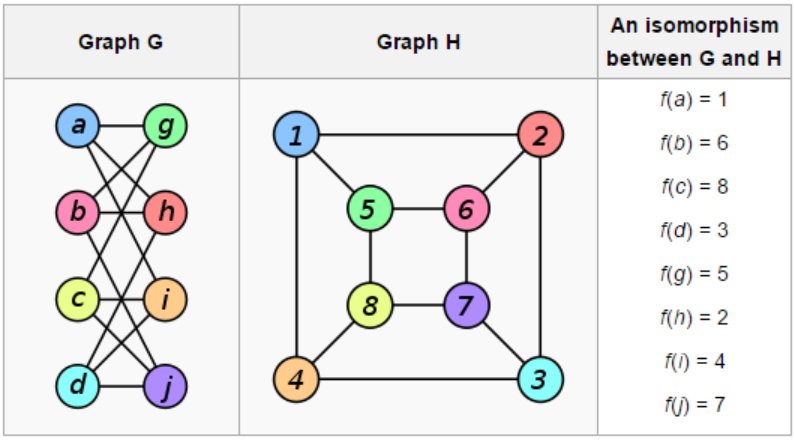


Figure 3.1: Demonstration of isomorphism between two undirected graphs G and H, from [25]

Definition 3.6 (Graph Isomorphism Protocol (see [21]))**.** *A Verifier V authenticates a Prover P as follows. G_0 and G_1 are a common input.*

- *Prover P selects a random permutation $\pi \in S_n$. Prover P sends $H = \pi(G_1)$ to the Verifier V.*
- *Verifier V selects a random bit $b \in \{0, 1\}$ and sends b to the Prover P.*
- *Prover P computes*

$$\psi = \begin{cases} \pi \circ \sigma & \text{if} \quad b = 0 \\ \pi & \text{if} \quad b = 1 \end{cases}$$

  and sends ψ to verifier V.
- *Verifier V checks that $H = \psi(G_b)$ and accepts/rejects accordingly.*
- *Repeat above steps multiple times (until desired confidence level).*

3.2.3 Verifying Zero-Knowledge Properties

We now briefly outline how the protocol satisfies the properties of zero-knowledge proofs.

Completeness

Consider a statement $x \in \mathcal{L}$. Hence the graphs G_0 and G_1 are isomorphic. An honest Prover P wishes to convince an honest Verifier V. Therefore, the verifier will confirm if $b = 0$,

$$\psi(G_0) = (\pi \circ \sigma)(G_0) = \pi(\sigma(G_0)) = \pi(G_1) = H$$

and if $b = 1$,

$$\psi(G_1) = \pi(G_1) = H.$$

Therefore, the protocol is perfectly complete.

Soundness

This is one of the easier checks for soundness. For $x \notin \mathcal{L}$, this implies the graphs in question G_0 and G_1 are not isomorphic. Therefore, with respect to a cheating Prover B attempting to convince an honest Verifier V, if the Verifier V selects $b = 0$, regardless of what the Prover B provides for ψ, the check will always fail since G_0 and G_1 are not isomorphic i.e. there is no σ such that $\sigma(G_0) = G_1$. Hence, with probability at least $\frac{1}{2}$, the check will fail. Hence, with a sufficient number of iterations, the soundness error can be made arbitrarily small.

Zero-Knowledge

We present a simulator to demonstrate the zero-knowledge property with an arbitrary Verifier strategy V^*. A Verifier strategy V^* can be thought of as two functions:

- $V_1(H)$ outputs the bit b that the Verifier V^* chooses after the Prover P submits H
- $V_2(H, \psi)$ is whatever Verifier V^* outputs after seeing Prover P's response ψ to the bit b.

Both V_1 and V_2 are clearly efficiently computable. We now describe an efficient simulator S^* that is a standalone algorithm that will output a distribution indistinguishable from the output V^*. The simulator S^* will work as follows (from [21]):

- Pick a random bit $b' \in \{0, 1\}$. Select a random permutation $\pi' \in S_n$.
- Send $H = \pi'(G_{b'})$.
- Let $b^* = V_1(H)$ (bit output by V^*). If $b' \neq b^*$, go back to Step 1.
- (If $b' = b^*$) Send $\psi = \pi'$, and hence have final output of $V_2(H, \psi)$.

Note the proof of distributional equivalence is literally identical to that of the discrete log example. Namely, the distributional equivalence is guaranteed by the fact that, in the protocol, since $\pi \in S_n$ and is hence invertible, $\pi \circ \sigma$ is effectively a random draw. Note, when $b' = b^*$, the verification checks out,

$$\psi(G_{b^*}) = \pi'(G_{b^*}) = \pi'(G_{b'}) = H,$$

as desired. The remainder of the proof is omitted as it is identical to the discrete log example.

3.2.4 Key Points of the Protocol

We discuss a few interesting aspects of this protocol.

Graph Theory

This protocol made use of graph theory. In particular, graph-related problems are a popular candidate for cryptographic protocols (see [44] for an exposition), problems including: graph isomorphism, vertex cover, Hamiltonian paths, and graph colorings. These graph-related problems on large graphs are popular due to the near impossibility of discovering a solution yet ease of checking a particular solution. This observation in particular has significant complexity implications as these problems are all a part of the class "'NP', an idea we will explore deeper in the following chapter.

Chapter 4

Zero Knowledge and Complexity

Zero-knowledge is of particular interest to computer science theorists because of its connections to complexity. Namely, under reasonable assumptions, there is a zero-knowledge proof protocol for any language in NP. In this chapter, we will present a brief introduction to complexity. We will demonstrate a zero-knowledge protocol for 3-coloring and discuss broader implications of zero-knowledge to complexity.

4.1 Introduction to Complexity

Here, we provide a brief introduction to complexity. For more clarification on any of these introductory complexity topics see [46].

4.1.1 Decision Problems

Consider a universe of statements U and a language $\mathcal{L}$. Say we wish to *decide* whether for a given $x \in U$, we also have $x \in \mathcal{L}$.

Definition 4.1 (Decision Problem)**.** *A decision problem for a language $\mathcal{L}$ is a problem that takes as input $x \in U$ and returns whether $x \in \mathcal{L}$ (yes) or $x \notin \mathcal{L}$ (no).*

For example, a classic decision problem may be to determine whether a given number n is composite ($\mathcal{L}$ here is the set of composite numbers). We define two different kinds of algorithms:

Definition 4.2 (Decider)**.** *We call an algorithm/machine M a decider for $\mathcal{L}$ if it solves a decision problem for a language $\mathcal{L}$ i.e. given an $x \in U$, it returns whether $x \in \mathcal{L}$ (yes) or $x \notin \mathcal{L}$ (no).*

Definition 4.3 (Verifier)**.** *We call an algorithm/machine M a verifier for $\mathcal{L}$ if it verifies a decision problem for a language $\mathcal{L}$ i.e. given an $x \in U$ and sufficient information/witness w, it can confirm that $x \in \mathcal{L}$.*

We may illustrate the contrast between these two types of algorithms by reconsidering the composite number decision problem. Say

$$U = \mathbb{Z}, \quad \mathcal{L} = \{\text{composite numbers}\}$$

An example Decider M for $\mathcal{L}$ may work as follows. Given any $n \in U = \mathbb{Z}$, M will directly decide if n is composite (for example, by checking all $1 < k < n$ for existence of a k such that $k|n$).

In contrast, a Verifier M for $\mathcal{L}$ may work as follows. Given any $n \in U = \mathbb{Z}$ and corresponding witness w (for example, a $w = k$ such that $k|n$), M verifies that n is composite (i.e. by verifying that $k|n$).

Note that Deciders are significantly more powerful in that they solve the problem directly as opposed to Verifiers that can simply verify a given solution (with additional information). Why then do we care at all about Verifiers? The key here is time complexity.

4.1.2 Time Complexity

In practical considerations, we usually care about the run-time of our algorithms. Consider an algorithm M (either a decider or verifier) for a language $\mathcal{L}$. We usually have some natural sense of "size" for an $x \in U$. For example, if $x \in U = \mathbb{Z}$ this notion of size is obvious. If $x \in U = G = (V, E)$ (i.e. graphs), a notion of size may be $|V|$ or $|E|$. We will use a function $s(x)$ to denote the "size" of $x \in U$. We then define

Definition 4.4 (Polynomial Run-Time)**.** *Consider algorithm M on input $x \in U$. Let $T_M(x)$ be the run-time of M on input x. We say M has polynomial run-time if there exists a polynomial $p(n)$ with real coefficients such that for all x,*

$$T_M(x) \leq p(s(x)).$$

Essentially this implies that M has *polynomial run-time* if its run-time on x is bounded above by a polynomial on $s(x)$. Note the choice of "units" for run-time is irrelevant as a scaling factor of a polynomial is still a polynomial.

Due to practical considerations of modern computing resources, we usually say an algorithm is *practically computable* if its run-time is polynomial. Therefore, for a particular problem $\mathcal{L}$ there may exist a practically computable Verifier M but no practically computable Decider M'.

4.1.3 Complexity Classes

We may combine these discussions on Decision Problems and Time Complexity to discuss complexity classes. We have the following two definitions.

Definition 4.5 (The Class **P**)**.** *We say $\mathcal{L} \in \boldsymbol{P}$ if there exists a polynomial-time decider M for $\mathcal{L}$.*

Definition 4.6 (The Class **NP**)**.** *We say $\mathcal{L} \in \boldsymbol{NP}$ if there exists a polynomial-time verifier M for $\mathcal{L}$.*

Clearly $\mathbf{P} \subseteq \mathbf{NP}$. Note if a $\mathcal{L} \in \mathbf{NP}$ but $\mathcal{L} \notin \mathbf{P}$, this would be a natural candidate for a cryptographic protocol. Namely, $\mathcal{L} \notin \mathbf{P}$ would imply that it computationally difficult for a cheating prover to find a solution $x \in \mathcal{L}$. In contrast, an honest prover could provide the witness w for a Verifier to verify a solution $x \in \mathcal{L}$ in computationally reasonable time.

For example, we can consider the graph isomorphism example from the previous chapter. To determine if graphs G_0 and G_1 are isomorphic, a cheating prover needs to check all $n!$ permutations of vertices of G_0 (assume $G_0 = (V_0, E_0), |V_0| = n$), which is non-polynomial and computationally infeasible for a large graph. In contrast, an honest prover simply needs to provide a witness, permutation σ, so that verifier can verify (in polynomial time) that $G_1 = \pi(G_0)$. Hence, graph isomorphism has a polynomial verifier and is hence in **NP**.

Therefore, much of modern cryptography rests on the assumption $\mathbf{P} \neq \mathbf{NP}$. Unfortunately, the following is still an open problem:

Problem (**P** versus **NP** Problem, Unsolved)**.** *Is $\boldsymbol{P \neq NP}$ or is $\boldsymbol{P = NP}$?*

Equality would have disastrous implications for cryptography, as it would essentially compromise the security of most cryptographic systems. However, most theorists believe $\mathbf{P} \neq \mathbf{NP}$.

4.1.4 NP-Completeness

Related to the above, we have the following additional definition.

Definition 4.7 (The Class **NP-Complete**)**.** *We say $\mathcal{L} \in \boldsymbol{NP\text{-}Complete}$ if*

- $\mathcal{L} \in \boldsymbol{NP}$
- *every $\mathcal{L}' \in \boldsymbol{NP}$ is reducible to (an instance of) $\mathcal{L}$ in polynomial time.*

In saying $\mathcal{L}'$ is "reducible to $\mathcal{L}$, we mean that a problem in $\mathcal{L}'$ can be transformed into an instance of a problem $\mathcal{L} \in \mathbf{NP}$. Namely, because of this reductive property, if some $\mathcal{L} \in \mathbf{NP\text{-}Complete}$ is also in **P**, then $\mathbf{P} = \mathbf{NP}$, since

- Take a problem in $\mathcal{L}' \in \mathbf{NP}$
- Problem in $\mathcal{L}'$ reduces to a problem in $\mathcal{L}$ in polynomial time
- Problem in $\mathcal{L}$ can be decided in polynomial time (since $\mathcal{L} \in \mathbf{P}$)
- Above steps are a polynomial-time decider for $\mathcal{L}'$

Therefore, much research on the **P** versus **NP** Problem focuses on **NP-Complete** problems. As it turns out, this observation has critical implications for zero-knowledge. Namely, if we can show a zero-knowledge protocol for an **NP-Complete** problem, this implies the existence of a zero-knowledge protocol for every problem in **NP**!

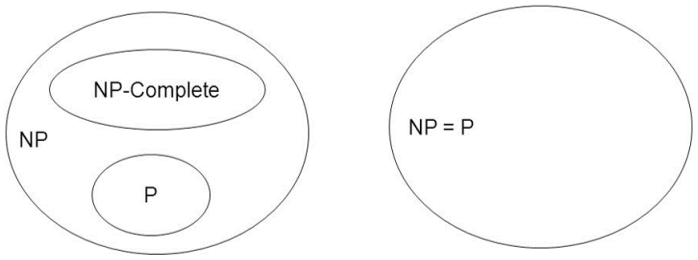


Figure 4.1: Left: If $\mathbf{P} \neq \mathbf{NP}$; Right: If $\mathbf{P} = \mathbf{NP}$

4.2 Zero-Knowledge Graph 3-Coloring Protocol

In particular, the **NP-Complete** problem for which we will show a zero-knowledge proof protocol will be the Graph 3-Coloring Problem. We first introduce the problem and present the protocol.

4.2.1 3-Coloring

We first present the 3-Coloring problem (refer to [28] for greater detail).

Definition 4.8 (3-Coloring)**.** *Let $G = (V, E)$ be a graph. We say G has a 3-Coloring if there exists a function*

$$c : V \to \{R, G, B\}$$

such that for every edge $(v_1, v_2) \in E$,

$$c(v_1) \neq c(v_2).$$

In other words, a 3-Coloring is simply an assignment of vertices to the colors Red, Green, and Blue such that no two adjacent vertices (i.e. along an edge) share the same color (see below image for an example of a 3-Coloring).

The 3-Coloring problem is known to be in the class **NP-Complete**. In other words, given a witness i.e. a coloring, we can verify it is a valid coloring by checking every edge. In contrast, there are no known efficient algorithms to determine if a (large) graph has a 3-Coloring.

4.2.2 Commitment Schemes

Before we present the protocol, we present a cryptographic tool that will be utilized in the protocol.

Definition 4.9 (Commitment Schemes, (see [4]))**.** *A commitment scheme is a function*

$$C : \{0,1\}^{\ell} \times \{0,1\}^{n} \to \{0,1\}^{k}$$

satisfying

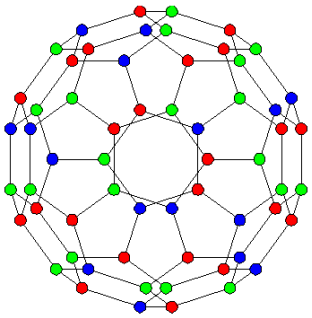


Figure 4.2: Depiction of a valid 3-coloring, from [10]

- ***Secrecy***: *for every* $x, x' \in \{0,1\}^\ell$ *and* U_n *the uniform distribution,*

$$C(x, U_n) \approx C(x', U_n)$$

(i.e. the distributions are indistinguishable).

- ***Binding***: *for every* $y \in \{0,1\}^k$ *there exists at most a single* x *such that*

$$y = C(x, r)$$

for some r*. In other words, there does* ***not*** *exist two distinct pairs* (x, r) *and* (x', r') *such that (for* $x \neq x'$*),* $y = C(x, r) = C(x', r')$*.*

The use of a cryptographic commitment is twofold. It allows an agent to (1) commit to a certain value (i.e. providing y) without revealing the value (secrecy) and (2) binds the party to the particular committed value (binding). This construct/tool will reappear as we will utilize these in our applications later.

4.2.3 The Protocol

Now that we've introduced commitment schemes, we now present a zero-knowledge protocol for 3-Coloring. In this language $\mathcal{L}$, statements are of the form

$$G \text{ admits a 3-Coloring.}$$

Say a Prover P possesses a proof, i.e. a 3-Coloring c of G. We have the following zero-knowledge protocol (see [21] for the original presentation).

Definition 4.10 (Graph 3-Coloring Protocol (see [21]))**.** *We let graph* G *be a common input, and also have a known commitment scheme*

$$C : \{0,1\}^2 \times \{0,1\}^n \to \{0,1\}^k$$

(note two bits is sufficient two encode 3 colors). A Verifier V *authenticates a Prover* P *as follows.*

- *Prover P selects a random bijective function*

$$\pi : \{R, G, B\} \to \{1, 2, 3\}$$

and defines

$$c' = \pi \circ c.$$

Prover P then computes commitments y_i such that for every vertex v_i, Prover P selects random $r_i \in U_n$, and calculates

$$y_i = C(c'(v_i), r_i).$$

Prover P then sends all commitments y_i to Verifier V.

- *Verifier V selects a random edge $e = (v_i, v_j) \in E$ and sends edge e to the Prover P.*
- *Prover P then opens the commitments, i.e. he computes*

$$x_i = c'(v_i) \quad x_j = c'(v_j)$$

and sends (x_i, y_i) and (x_j, y_j) to the Verifier V.

- *Verifier V checks that the commitments are correct:*

$$y_i = C(x_i, r_i) \quad y_j = C(x_j, r_j),$$

checks that the colors are different:

$$x_i \neq x_j,$$

and accepts/rejects accordingly.

- *Repeat above steps multiple times (until desired confidence level).*

Essentially, the function π serves to hide the actual coloring and the commitments y_i serve to ensure the prover sends the committed colors for the requested edge. See the following diagram (from lecture of Prof. Yevgeniy Dodis of NYU, [13]) for a pictorial representation of the protocol.

4.2.4 Verifying Zero-Knowledge Properties

We briefly verify the protocol above is indeed a zero-knowledge proof.

Completeness

Consider an honest Prover P wishing to convince an honest Verifier V of a statement $s \in \mathcal{L}$. Hence, since the graph G admits a valid 3-Coloring c (that the prover is aware of), the honest verifier will clearly confirm the prover's commitments (the prover is honest), and conclude $x_i \neq x_j$ since

$$\begin{aligned} c(v_i) \neq c(v_j) &\implies (\pi \circ c)(v_i) \neq (\pi \circ c)(v_j) \\ &\implies c'(v_i) \neq c'(v_j) \implies x_i \neq x_j. \end{aligned}$$

Therefore, we have perfect completeness.

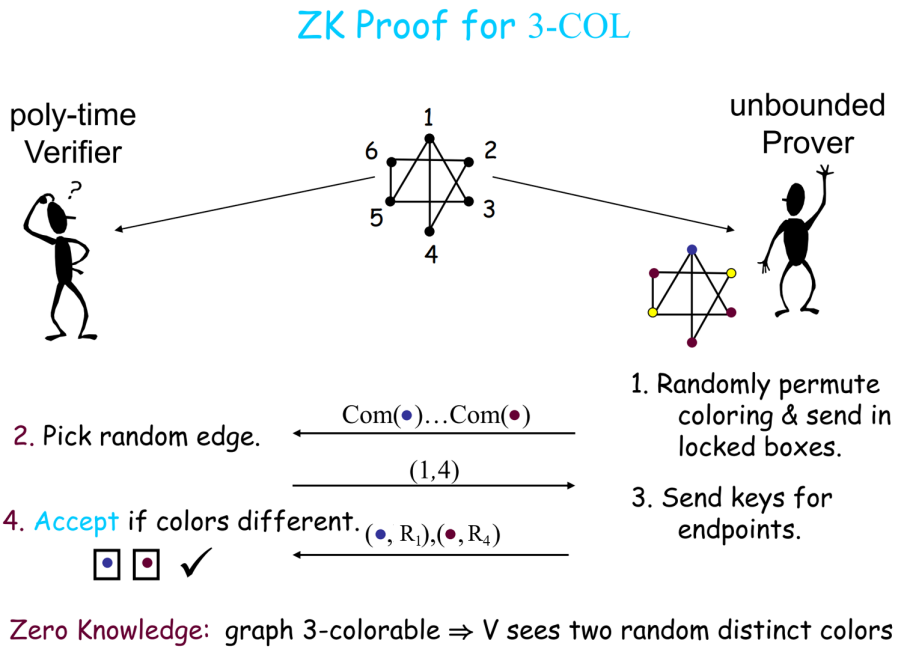


Figure 4.3: Protocol Diagram from lecture of Prof. Yevgeniy Dodis of NYU, [13]

Soundness

Now, consider a cheating Prover B trying to cheat an honest Verifier V with a statement $s \notin \mathcal{L}$. Therefore, the graph G admits no proper 3-Coloring. Now, note that Prover B must provide commitments y_i to Verifier V before the choice of edge e. In addition, from the binding property of commitments, the y_i is determined by a unique pair (x_i, r_i) and hence the Prover B is locked in to a priori choice of x_i.

Assume for the sake of contradiction, for any choice of edge $e = (v_i, v_j)$, the Verifier will find $x_i \neq x_j$. However, note all $x_i \in \{1, 2, 3\}$. Then, we can choose an arbitrary bijective function

$$\pi' : \{1, 2, 3\} \to \{R, G, B\}.$$

But then, since $x_i \neq x_j$,

$$\pi'(x_i) \neq \pi'(x_j)$$

and hence, π' applied to the x_i yields a valid coloring of G. But, this is clearly a contradiction since G admits no proper 3-Coloring. Therefore, there must exist some edge $e \in E$ such that the Verifier will find $x_i = x_j$ and reject. Therefore, if $|E| = m$, the verification will fail with probability at least $\frac{1}{m}$. The soundness error can be made arbitrarily small, as running the protocol cm times yields an error of

$$\left(1 - \frac{1}{m}\right)^{cm} = \left[\left(1 - \frac{1}{m}\right)^{m}\right]^{c} < \left(\frac{1}{2}\right)^{c} = \frac{1}{2^c}$$

as desired.

Zero-Knowledge

We present a simulator to demonstrate the zero-knowledge property with an arbitrary Verifier strategy V^*. A Verifier strategy V^* can be thought of as two functions:

- $V_1[\{y_i\}_i]$ outputs the edge $e = (v_i, v_j)$ that the Verifier V^* chooses after the Prover P submits commitments $\{y_i\}_i$.
- $V_2[\{y_i\}_i, (x_i, r_i), (x_j, r_j)]$ is whatever Verifier V^* outputs after seeing Prover P's response $(x_i, r_i), (x_j, r_j)$ to the edge e.

Both V_1 and V_2 are clearly efficiently computable. We now need to come up with an efficient simulator S^* that is a standalone algorithm that will output a distribution indistinguishable from the output V^*. The simulator S^* will work as follows (from [21]):

- Pick a random edge $e' = (v_{i'}, v_{j'}) \in E$. Select random $x_{i'} \in \{1, 2, 3\}$ and random $x_{j'} \in \{1, 2, 3\} \setminus x_{i'}$.

  Compute commitments (for all i) y_i as follows. Select random $r_i \in U_n$. Then

$$y_i = \begin{cases} C(1, r_i) & \text{if} \quad i \notin \{i', j'\} \\ C(x_{i'}, r_i) & \text{if} \quad i = i' \\ C(x_{j'}, r_i) & \text{if} \quad i = j' \end{cases}$$

- Send commitments $\{y_i\}_i$.
- Let $V_1[\{y_i\}_i] = e^* = (v_{i^*}, v_{j^*})$. If $e' \neq e^*$, go back to Step 1.
- (If $e' = e^*$) Send $(x_{i'}, r_{i'})$ and $(x_{j'}, r_{j'})$ and hence have final output of $V_2[\{y_i\}_i, (x_{i^*}, r_{i^*}), (x_{j^*}, r_{j^*})]$.

The proof that this is an valid simulator is obvious and follows similarly as the proof for the previous protocols discussed. In particular, the equivalence of the distribution of commitments y_i follows directly from the secrecy property of commitment schemes.

Hence we have verified that the protocol described above is indeed a zero-knowledge proof for the **NP-Complete** 3-Coloring problem.

4.3 Broader Implications to NP

Now that we've demonstrated a zero-knowledge protocol for an **NP-Complete** problem, and since there then exists a reduction from any problem in **NP** to this problem, we have the following beautiful result (from [21]):

Theorem 4.11 (All Problems in **NP** Have Zero-Knowledge Proofs, [21])**.** *Let $\mathcal{L} \in \boldsymbol{NP}$. Then, using commitment schemes, there then exists a zero-knowledge proof protocol for showing $x \in \mathcal{L}$.*

This is a very interesting theoretical result. However, on a more practical level, as observed in [4], the zero knowledge protocol for **NP** is a dividing line between practical and theoretical cryptography. Namely, the reduction between **NP** statements people want to prove in practice and the language of graph 3-Coloring is extremely complicated and inefficient (in the sense that it requires a great deal of information to encode a 3-Coloring for a graph).

Hence, while using the zero knowledge protocol presented above shows a polynomial-time protocol for any **NP** problem, it will not yield a very practical protocol. Therefore, much energy is expended at making tailor-made zero-knowledge proofs that are simpler and more efficient for specific interesting classes of **NP** statements. In particular, as [4] notes, many of the more interesting and sophisticated cryptographic protocols and schemes follow this paradigm.

Regardless, these results are of interest to theorists. Additional similar kinds of complexity results involving zero-knowledge have been researched within other complexity results, notably the stronger result (shown in [5]):

Theorem 4.12 (from [5])**.** *(Under certain assumptions) Anything that can be proved by an interactive proof system can also be proved via a zero-knowledge protocol.*

Chapter 5

Exotic Applications of Zero-Knowledge

Up to this point, the applications of zero-knowledge that we've discussed have been classically cryptographic i.e. often as identification protocols or password verifiers. However, one of the most intriguing aspects of zero-knowledge are applications beyond these traditional applications. In this chapter, we will discuss two more exotic applications of zero-knowledge proofs: *electronic voting (or e-voting)* and *nuclear disarmament*. Note that, in contrast to previous chapters, here we will simply provide a cursory overview of the protocols in question; our goal here is to simply provide a sense of how zero-knowledge proofs are utilized in these unique instances.

5.1 E-Voting

We provide a surface overview of applications of zero-knowledge to E-voting.

5.1.1 The Fundamental Problem

In our modern age of democracy, voter turnout within many developed countries and in particular the United States is often abysmally low. A common reason offered for this low voter participation is the required effort to physically vote at a polling station. Hence, a common solution offered is to allow electronic or online voting.

While e-voting albeit has its obvious downsides (i.e. requiring a base level of technological literacy, etc), a significant advantage of e-voting is the potential for completely transparent vote counting, where every citizen could verify if the votes are tallied correctly. In particular, such a measure is critical when the government itself is the agent doing the vote counting and hence has incentive to cheat/lie about the results. Therefore, a secure e-voting system may be viewed as an agent to ensure fair democracy.

Much research has been done on developing secure e-voting protocols. While the methodologies vary, the general goals are the same (see [6]):

- **Eligibility**: only legitimate voters can cast a ballot
- **Verifiability**: outcome is valid

  - **Individual**: verify their vote was cast and recorded correctly
  - **Universal**: any one can verify that the tally is correct

- **Privacy**: individual voter's choice cannot be ascertained
- **Coercion-Resistance**: no party can be forced to vote (or abstain) in a certain way

5.1.2 Sigma Protocols

Many protocols have been designed that satisfy many of the security goals stated above (see [26, 6] for these protocols). Since these protocols require many steps and require the introduction of many other cryptographic primitives, here we will simply introduce the class of zero-knowledge proofs utilized by e-voting protocols: *sigma protocols*.

Formal Definition

We present the following generalized class of protocols:

Definition 5.1 (Σ-Protocol, [6])**.** *Let $\mathbb{F}$ be a field. Let W, V be $\mathbb{F}$-vector spaces and let $\phi : W \to V$ be a $\mathbb{F}$-linear map. We then have the following Σ-Protocol, Σ_ϕ as an interactive proof protocol between a Prover P of knowledge of a pre-image x,*

$$y = \phi(x),$$

for a value y to a Verifier V (i.e. V already knows y):

- *Prover P selects random $r \in W$ and sends $a = \phi(r)$ to Verifier V.*
- *Verifier V selects random $c \in \mathbb{F}$ and sends to Prover P.*
- *Prover P computes $d = r + cx$ (in field $\mathbb{F}$) and sends to Verifier V.*
- *Verifier V checks that $\phi(d) = a + cy$ and accepts/rejects accordingly.*
- *Repeat above steps (until desired confidence level).*

Theorem 5.2. *For $\phi : W \to V$ a $\mathbb{F}$-linear map, Σ_ϕ is an "honest-verifier" zero-knowledge proof protocol.*

Proof. The proof this protocol is zero-knowledge is straightforward, hence we just provide a sketch. Completeness follows from linearity of ϕ:

$$\phi(d) = \phi(r + cx) = \phi(r) + \phi(cx) = \phi(r) + c\phi(x) = a + cy.$$

Soundness and "Honest-Verifier Zero-Knowledge" (i.e. zero-knowledge only in the case of an honest verifier) follow from the random choice of r and c. □

For example, when working with the discrete log example, we were essentially working with the protocol Σ_ϕ where

$$\phi : \mathbb{F}_p \to G_p, \quad \phi(x) = g^x$$

where G_p is a group of order p with generator $g \in G_p$. (However as we showed, the discrete log protocol also exhibits stronger "malicious-verifier" zero-knowledge).

Usage

This class of protocols is extremely powerful due to the following key properties ([22]):

- They can be transformed into non-interactive variants (i.e. the Prover doesn't need to run the interactive proof separately with every verifier and hence is more practical).
- They can be repeated in parallel i.e. parallel verification (i.e. can run multiple iterations at once to reduce number of interactions yet maintain zero-knowledge).
- They can be easily chained together to create proofs for *AND*, *OR* statements.

In particular, through these protocols, e-voting protocols have been developed that utilize Zero-Knowledge Σ-Protocols in the following way for security guarantees ([6]):

- Voters provide a ZK-Proof that their vote is valid without revealing their vote.
- Authorities provide a ZK-Proof that they have tallied the votes correctly.

Hence, the use of these Σ-Protocols in conjunction with other cryptographic primitives form the basis for theoretical secure e-voting systems.

Non-Interactive Variant

As mentioned above, a key aspect of the strength of Σ-protocols lies in their ability to be transformed into non-interactive proofs. In the case of an election, this is critical: if all voters were to ask the authorities to verify that the votes were tallied correctly, participating in an interactive proof with every voter would be totally impractical. Here, we will present this transformation to a non-interactive variant (courtesy of [15]).

Definition 5.3 (Σ-Protocol: Non-Interactive Variant, [15])**.** *Let Σ_ϕ^H be a Σ-Protocol for $\phi : W \to V$ a $\mathbb{F}$-linear map and $H : V \times V \to \mathbb{F}$ an external, public (i.e. known) hash function whose output is essentially random. We then have the following protocol, $\Sigma_{\phi,\text{Non-Inter}}$ as a non-interactive proof protocol by a Prover P of knowledge of a pre-image x,*

$$y = \phi(x)$$

of a value y:

- *Prover P selects random $r \in W$ and computes $a = \phi(r)$ to Verifier V.*
- *Prover P computes $c = H(y, a)$.*
- *Prover P computes $d = r + cx$ (in field $\mathbb{F}$).*
- *Prover P outputs the transcript (y, a, c, d) (the proof).*
- *Repeat above steps with independently chosen a (until desired confidence level).*

Theorem 5.4. *For $\phi : W \to V$ a $\mathbb{F}$-linear map and $H : V \times V \to \mathbb{F}$ a cryptographic hash function, Σ_ϕ^H is a zero-knowledge proof protocol.*

Proof. The proof follows directly from the previous theorem. Note H is essentially mimicking an honest verifier in Σ_ϕ. Since Σ_ϕ is an "honest-verifier" zero-knowledge proof protocol, it follows then that Σ_ϕ^H is a zero-knowledge proof protocol. □

Note the key distinction between the interactive and non-interactive protocols. In the interactive protocol, the choice of c is determined by the verifier, whereas in the non-interactive case c is determined by an external function H which is taken as indistinguishable from a random function. Therefore, in the instance of an honest verifier, the interactive and non-interactive protocol are essentially identical as the choice of c is random. Note any Verifier V can verify the proof by checking

$$c = H(y, a) \quad \text{and} \quad \phi(d) = a + cy.$$

Therefore, the Prover P is providing a full transcript of an interactive proof that will be sufficient to convince any verifier. Hence, this non-interactive proof allows us to avoid the impracticality of participating in excessively many interactive protocols.

5.1.3 Implementation of Secure E-Voting

As the image below make clear, electronic voting is becoming increasingly common.

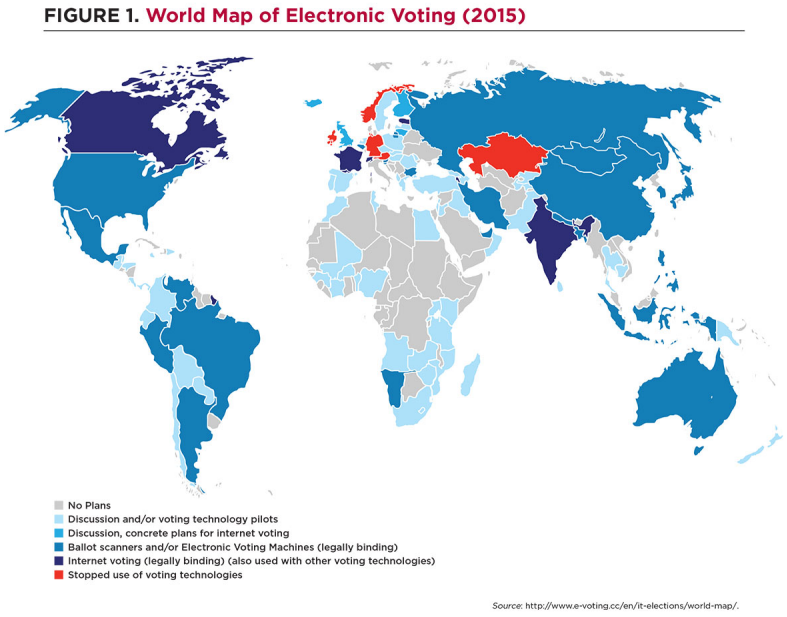


Figure 5.1: E-Voting is becoming increasingly prevalent across the globe, [35]

However, few of the currently implemented e-voting systems utilize the types of security protocols presented above. Yet, in recent years, there has been the implementation of a few secure zero-knowledge based electronic voting systems. Though not necessarily widely used at the moment, their implementation is a good proof-of-concept. We now present an implementation.

Helios

Helios is the e-voting system most widely discussed in the literature. Developed in 2008 by Ben Adida (see [2]), Helios is considered one of the earliest and best maintained open-source secure e-voting system. Helios provides individual verifiability, which means that a user can verify that his/her vote was actually taken into account in the final tally. It also provides universal verifiability, implying that any observer (or user) can check that all votes were counted and tallied correctly. However, Helios does little in terms of coercion resistance, and hence Helios themselves do not recommend their own platform for federal or state elections (see [2]). Unsurprisingly then, Helios has yet to be implemented in elections of consequence, though the platform has been used widely for low-coercion elections such as student body elections.

5.2 Nuclear Disarmament

Equally surprising is the application of zero-knowledge proofs for certain physical properties, such as nuclear warhead verification.

5.2.1 The Fundamental Problem

The majority of the developed world, including the United States and Russia, agree that nuclear weapons are dangerous and hence are willing participants in various arms-control agreements such as the "Non-Proliferation Treaty" (see below).

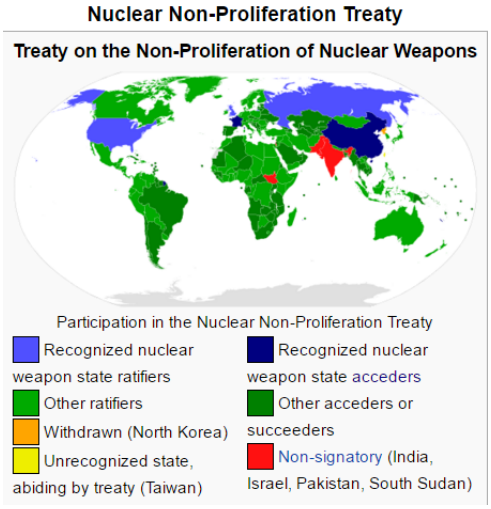


Figure 5.2: Depiction of participation in the treaty from [49]

Namely, nations around the world have agreed to take steps to dismantle their nuclear arsenal. But to reduce weapon stockpiles, nations need to verifiably prove that they are taking steps to dismantle warheads. In particular, member nations must authenticate that their warheads in storage and their warheads in their respective dismantlement queue are

genuine. However this yields a problem: no nation wants to reveal any information on the design of their nuclear warhead (for security and military purposes), and hence are wary of a direct inspection of their warheads. A common offered solution is the use of *trusted third parties* for authentication. This is problematic as well due to the lack of a truly genuine "third-party" in these situations.

The problem can then be restated as follows. For a particular purported warhead W in Country A, is it possible for Country B to verify that W is a genuine warhead without revealing any additional information (i.e. the construction, etc).

5.2.2 A Protocol for Warhead Verification

The problem described above is a natural candidate for a zero-knowledge protocol. However, while the property to be verified in prior protocols was fundamentally mathematical or computational, the property here is a physical one, and hence it is less obvious how one should design a protocol. In 2013, Alexander Glaser, Boaz Barak, and Robert Goldston developed (see [3]) a zero-knowledge proof protocol to verify that nuclear warheads designated for disarmament are actually what they purport to be. We will present their protocol here.

A Simpler Reduction

We consider a reduction of the protocol to illustrate the general idea/approach (from [3]). Consider a Prover P with two jars each containing $x \in [0, N]$ marbles. Prover P wants to convince a Verifier V that the jars contain the same number of marbles. We have the following basic zero-knowledge protocol:

- Prover P prepares k pairs of buckets ($k << N$) with both buckets in the i^{th} pair each containing a random number of marbles $R_i \in [0, N]$ marbles.
- Verifier V choose one of the k pairs at random. He inspects the other $k - 1$ pairs to check the bucket pairs each contain an identical number of marbles.
- Prover P pours the marbles in the first jar into the first bucket (of the selected pair) and the marbles in the second jar into the second bucket (of the selected pair).
- Verifier V checks that the buckets (of the selected pair) have the same number of marbles and accepts/rejects accordingly.
- Repeat above steps (until desired confidence level).

The protocol above is clearly zero-knowledge. Since the Verifier checks that the remaining bucket pairs have an identical number of marbles, the soundness error in a single iteration is at most $\frac{1}{k}$.

Some Physics

The primary issue here is how to turn a fundamentally physically property into something more tractable. The authors had the following key idea/observation (see [3]).

Say we have a template warhead. We wish to verify that a certain test warhead is similar. Both are inside containers (so as to maintain certain privacy constraints). Consider the following procedure:

- Place "detectors" on an arc around each container (see image below).
- Subject each container to a stream of neutrons.
- Measure the number of neutrons arriving at each detector.

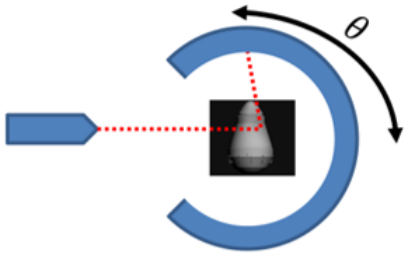


Figure 5.3: Depiction of detector setup from [3]

Then, the warheads must be similar if and only if the measurements (of neutrons) on the detectors at position θ are close for a random θ. While this is admittedly a gross over-simplification of the physics and measurement scheme in this procedure (see [3] for more details), the general idea is that the similarity between the warheads may be ascertained by comparing the induced pattern on the detectors.

A Warhead Verification Protocol

With the observation above, the authors designed the following protocol (a variant on the simplified protocol presented earlier):

Definition 5.5 (Protocol for Nuclear Warhead Verification, [3])**.** *A Prover P wishes to convince a Verifier V that a test warhead W is similar to a template warhead W^*. We have the following protocol.*

- *Prover P prepares k pairs of detectors ($k << N$) with both detectors in the i^{th} pair each initialized to a random offset $R_i \in [0, N]$ marbles.*
- *Verifier V choose one of the k pairs at random. He inspects the other $k-1$ pairs to check the detector pairs each contain an identical offset. Verifier V selects random angle θ.*
- *Prover P places the first detector (of the selected pair) at position θ around warhead W and the second detector (of the selected pair) at position θ around warhead W^* and runs the neutron streams for each.*

- *Verifier V checks that the neutron measurements are close and accepts/rejects accordingly.*
- *Repeat above steps (until desired confidence level).*

Note the use of the offset R_i is to inject some randomness to the detectors such that the final output is distributed randomly and hence, nothing can be determined about the test warhead W. Note this protocol is isomorphic to the simplification stated earlier, and hence, this is a zero-knowledge protocol. Hence, using this protocol, one can verify that a purported warhead W is genuine.

5.2.3 Implementation of Secure Nuclear Warhead Verification

The authors in [3] also ran simulations to confirm the efficacy of their protocol (over different neutron sources, detectors, etc). For the most part, their results were quite good (see below image for example of such a simulation), suggesting the great potential for this protocol. Since the publication of the paper and introduction of this protocol, much similar research has been developed, most notably from the *NRF Zero Knowledge Project* (see [29]).

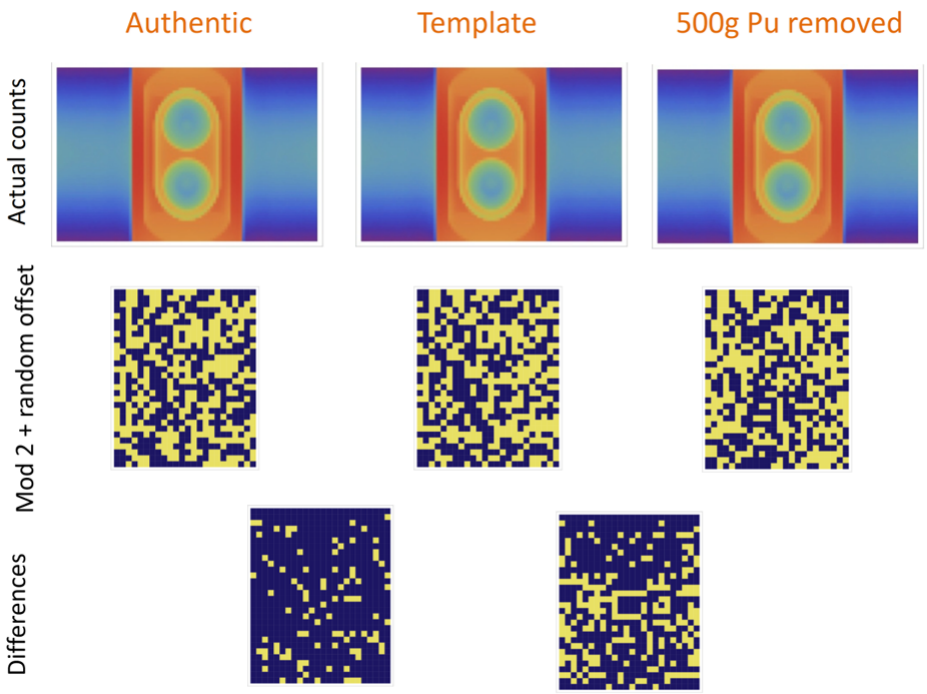


Figure 5.4: Depiction of a simulation run of the protocol (from [3]). Note the measurement is in the form of a 2D bit array that flips a bit every time a neutron hits the corresponding region on the detector. Here, the simulation shows an authentic warhead exhibits few differences from a template, whereas an inauthentic warhead (with 500g less Plutonium) exhibits many differences from a template

As of now, however, it is currently unknown if this secure nuclear warhead verification protocol is being seriously considered by nuclear committees across the globe.

5.3 Other Applications

The above applications serve to present some of the more exotic types of zero-knowledge protocols beyond standard security and cryptography based applications. There are a number of other similar applications, for example a zero-knowledge protocol for auctions without revealing bid values to auctioneers ([39]), etc. The existence of applications of zero-knowledge to this diverse array of fields in part motivates our study of zero-knowledge applications to financial regulation.

Part II

Applications of Zero Knowledge to Financial Regulation

Chapter 6

Introduction to Financial Regulation

We now turn to financial regulation. Standard discussions of cryptography and security with respect to finance typically deal with maintaining the security of transactions and privacy of financial entities. While these are undoubtedly important considerations in finance, they are usually fundamentally cryptographic identity protocols, and hence these protocols are not unique to finance but secure data transfer in general. In contrast, here we will consider a more unique application of crypto, and in particular zero-knowledge proofs, to balance the trade-off between confidentiality and transparency with respect to financial regulation. In this chapter, we will introduce this trade-off and motivate the protocols we will discuss.

6.1 Transparency and Confidentiality Trade-off

Within the realm of financial regulation, there is a natural trade-off between transparency and confidentiality. On the one hand, increased transparency is good in that it allows people to check that various financial agents are behaving "properly". However, too much transparency may crowd out financial entities' incentives to participate and innovate, as their actions/financial strategies are too public. On the other hand, confidentiality is good in that it allows agents to invest and participate in the financial markets without fear of information leakage, hence promoting liquidity. Yet, too much confidentiality allows the rise of malicious entities that can actively break rules due to the lack of scrutinization.

Therefore, transparency and confidentiality are typically seen as diametrically opposite goals, as the pursuit of one is to the detriment of the other. However, with the use of modern cryptographic techniques, perhaps we can somewhat eliminate this tension and better navigate this trade-off between confidentiality and transparency.

6.1.1 Prior Work

There has been limited work on this trade-off with respect to cryptography. The most notable example is a paper from Mark Flood, Jonathan Katz, Stephen Ong, and Adam Smith (see [17]), in which they introduce cryptographic tools for balancing transparency and confidentiality. In particular, they consider the cryptographic primitives of Multi-Party

Computation and Statistical Data Privacy (i.e. Differential Privacy). They outline the potential of these tools by presenting 3 case studies:

- Publication of Aggregated Sensitive Data: Develop public indexes such that it is impossible (within specific tolerances) to reverse engineer the confidential inputs from the published index
- Retail Loan Information to Support Research: Randomize the data in ways that preserve aggregate patterns, while disguising individual details, and to use techniques of differential privacy to calculate the corresponding guarantees and tolerances
- International Data Sharing: Secure multiparty computation may allow for international signals and statistics to be calculated in a distributed fashion such that the underlying granular data never needs to be centralized or shared

See [17] for greater explication of the above case studies. This work is significant in that it indicates and serves as a model for the potential that cryptographic tools have in helping to navigate the transparency and confidentiality trade-off.

Most additional and related work has built off of this paper (for example, see [40]), primarily focusing on Multi-Party Computation and Differential Privacy techniques.

6.1.2 The Trade-off and Zero-Knowledge

Zero-knowledge proofs serve as a natural candidate for the trade-off between transparency and confidentiality. One can view the "proof" part as providing *transparency* while maintaining "zero-knowledge", the *confidentiality*. In fact, these sorts of considerations have motivated the zero-knowledge protocols we have considered so far in this paper. For example, with respect to voting, we discussed the existence of zero-knowledge proof protocols that allowed anyone to verify that a given vote was tallied correctly (transparency) without revealing the actual vote (confidentiality).

In this paper, we will restrict ourselves to primarily using zero-knowledge proofs. Namely, we will develop novel applications of zero-knowledge to financial regulation (specifically the implicit trade-off between transparency and confidentiality). While [17] focused on applications with respect to supervisory information, i.e. developing aggregate data to benefit federal regulators and researchers/academics, we will adopt a more wide mandate and consider a broad range of applications with respect to financial regulation.

Our goal here is not necessarily to show what we believe to be the most important applications of zero-knowledge to financial regulation. Rather, in illustrating zero-knowledge protocols, our intention is to demonstrate the power and potential of zero-knowledge proof protocols to financial regulation.

NB: The types of applications we present here can likely also be solved in principal via a Multi-Party Computation. However, the use of zero-knowledge protocols is to present something more efficient and tailor-made.

6.2 Vignettes

We wish to identify problems in financial regulation for which zero-knowledge proofs may offer a solution. The best mechanism to do this was to reach out to financial parties and entities themselves, and query them on standard financial regulatory concerns they experience. Therefore, we polled a number of professionals from various banks, funds, and agencies via in-person discussions, e-mails, and phone calls. Per request, all these comments have been anonymized.

In compiling these statements, three major categories of financial regulatory concerns emerged (note the categories defined below are not standardized technical jargon, but rather terms we will use in the paper for ease of explanation). We present a subset of related comments to motivate the applications we then studied:

Personal Compliance: an individual ensuring his financial holdings and activity do not generate any conflicts of interest.

- "...and I've always felt uncomfortable that my employer knows my personal portfolio. While I understand that for compliance purposes this is required to make sure there is nothing in my portfolio that would go against client interests, it would be nice if there were some other way to convince my employer..."
- "...Even though a company has a separate compliance department that is supposed to be the sole people who deal with compliance reporting, I think theoretically a CEO or other C-Level executive could really obtain that information if they tried. While this probably isn't that big a deal at [Company Name Retracted], a large employer such as IBM or Boeing will possess a great deal of information..."
- "...Any time I make a trade, I need to send it to my company for approval so that they can make sure it isn't on their blacklist. Surely there's a better way to do this - with the SEC or something?"

Investment Compliance: a fund demonstrating that it satisfies its role as a fiduciary by demonstrating its investments are in accordance with restrictions placed by its investors.

- "...so in the wake of the financial crisis, there was a ton of finger pointing. Pension funds got it the worst. They usually have restrictions on the amount of risk they can take in their investments. I honestly believe most of these pension guys aren't to blame. They gave money to various funds who promised to meet their risk requirements. But at the end of the day, there is really no paper trail. There are 10 levels of people giving money to each other..."
- "My wife's retirement account is managed by some guys who promise not to exceed certain volatility thresholds. We have no idea if they actually follow these set guidelines. If they are ever put under investigation, they can easily go back and bullshit volatility numbers to claim compliance..."

- "...Our fund manages accounts for large universities, state pension funds, as well as HNWI. As you can imagine, we are under a number of restrictions. We update our clients regularly, letting them know how the portfolio is performing and updating them on the greeks. Of course we can't reveal our actual holdings due to issues of confidentiality, but our investors have our personal guarantee that we accurately report these cumulative numbers."

Fund Compliance: verifying that a fund satisfies legal and federal constraints i.e. the fund is not doing anything illegal or unethical.

- "...pre-Madoff, capital was flowing easy for hedge funds and VC's that were popping up left and right. People were willing to risk their capital with investors with limited track records. Now, post-Madoff and other similar cases, you really don't see that anymore. People are increasingly likely to give their money only to people with strong established track records. This is a case of a few bad apples spoiling the bunch, and what makes it hard for people to get established nowadays. I would love to leave [Company Name Retracted] but that's not really an option in this environment..."
- "...Funds have found it increasingly difficult to prove that they are legit. There are a lot of great ideas and funds out there, but they just don't get the chance since there is an increased implied risk premium for investing in these funds. Optimally, there should be some way for a fund or its investors to prove its legitimacy and assuage any concerns about a potential ponzi scheme..."

6.3 Proposed Applications

Inspired by the vignettes above, we identify the following financial regulatory problems to be considered. In the following chapters, we will present zero-knowledge protocols to tackle these problems. We present a total of three protocols: two of these new applications developed in this paper and the other from [47].

6.3.1 Reporting Portfolio Holdings to Employers

In almost any large company, there will be a maintained blacklist of companies that its employees cannot invest in. The reason for this is to prevent any potential conflicts of interest. While this blacklist seems natural for financial services firms, in fact firms in all industries maintain such blacklists. For example, a tech company such as Google will restrict its employees from investing in companies for which Google has a significant relationship or contracts with. The SEC (Securities Exchange Commission) ensures that all companies regularly maintain a blacklist so as to prevent any potential insider trading. To ensure that its employees are compliant, many companies require their employees disclose their financial holdings, so they can ensure none of these holdings are on the blacklist. Understandably, this may make employees uncomfortable. Our goal hence is to have a protocol by which an employee can prove none of their holdings are on the blacklist without revealing the actual holdings.

More formally, let A denote the set of possible assets. Denote $B \subset A$ the blacklist and $H \subset A$ an employee's holdings. Essentially, the problem reduces to showing

$$B \cap H = \emptyset$$

in zero-knowledge. Therefore, the crux of this problem is equivalent to showing a zero-knowledge set intersection. We develop a protocol for this problem in Chapter 7.

6.3.2 Risk Assurance for Financial Holders

When an individual makes an investment at a fund, they will typically also specify a series of restrictions that limits the funds behavior with the individual's assets. These restriction may be things such as a maximum risk threshold (i.e. volatility), a maximum exposure to a particular sector, etc. The issue is that in practice it is hard to ensure compliance of the fund to these restrictions. Namely, the fund wants to maintain confidentiality over its own holdings and strategies, and hence is limited in its information disclosure. In particular, the fund often behaves like a black box: simply providing returns to capital and not providing any concrete evidence that they are even following the stated restrictions. Our goal hence is to have a protocol by which funds can prove their investments fit within the promised restrictions (i.e. risk measures).

More formally, Let I denote a set of possible investments. For $i \in I$, define x_i the volume of investment in i. Essentially, the problem reduces to verifying expressions of the form

$$v_0 < a_1 x_1 + \cdots + a_n x_n < v_1$$

where a_i is a publicly known feature of i (ex. volatility) and $[v_0, v_1]$ is the stated restriction on the feature (ex. volatility lies between v_0 and v_1). Therefore, the crux of this problem is equivalent to showing zero-knowledge interval proofs. We present a protocol for this problem, originally presented in [47], in Chapter 8.

6.3.3 Pooling Information by Anonymous Investors

An investor in a fund may have reason to be suspicious of the actions/behavior of a fund. Such cynicism is more prevalent in the post-Madoff world in which people are more aware of the kinds of scams that may occur, that has hence induced a lesser willingness to invest in newer funds in recent years. The primary issue is that investors in funds tend to be anonymous, as they wish their investments to be non-public. Yet, as a result, this prevents any communication between these anonymous investors to verify the aggregate measures the fund provides them. In particular, consider the following. One of the primary signs of a Ponzi scheme is an incredibly rapid inflow of money into a fund. Yet, investors really have no way of knowing how much money is in the fund (due to this anonymity). Our goal hence is to have a protocol by which anonymous investors can pool information for certain calculations (we will primarily consider "sum" here).

More formally, let I denote the set of investors. For each $i \in I$, denote x_i the volume of investment by agent i. Essentially the problem reduces to computing

$$\sum_{i \in I} x_i$$

in zero-knowledge. Therefore, the crux of the problem is equivalent to showing a zero-knowledge sum mechanism (that has no direct communication between any two investors). We develop a protocol for this problem in Chapter 9.

Chapter 7

Application 1: Reporting Portfolio Holdings to Employers

We now turn to the first of our applications of zero-knowledge to financial regulation. We will introduce the importance of employees reporting portfolio holdings to their employers and discuss potential issues inherent in this. With this understanding, in this chapter we will then develop a novel zero-knowledge protocol by which employees can convince employers that their holdings are valid and legal. The protocol we present herein will utilize zero-knowledge set intersection protocols.

7.1 Introduction

7.1.1 Exposure to Non-Public Information

It is widely accepted than *insider trading*, the trading of stock or other financial assets by individuals with access to *non-public* information, is intolerable and illegal. Namely, insider trading confers an unfair advantage to those with non-public information, and hence insider trading undermines the security of the financial market. Now, when people think about exposure to non-public information, they often imagine the kinds of nefarious stories that dominate the news: illegal wiretaps, tipping information to relatives, etc. However, in fact, many (well-meaning) individuals are exposed to non-public information on an almost daily basis, namely through their job.

Consider an individual working at the Food and Drug Administration (FDA), involved in the drug review process for various pharmaceutical products and drugs. Now, for pharmaceutical and biotechnology companies, their success is primarily driven by their ability to bring their products to market, which is governed/restricted by the FDA's drug review process. The consequence of this is that an individual who works at the FDA has non-public information on whether a drug is likely to pass or stall in the drug review process, and by proxy has non-public information on the future success/failure of the company.

Analogously, consider an employee at a large, multi-national construction company. The employee notices one day the company switches from using cement from Supplier A to Supplier B. Therefore, this may indicate that the construction company declined to renew

their contract with Supplier A and instead signed a contract with Supplier B. However, this new development (which has important implications for the long term viability of the Suppliers) may not have been publicized by either Supplier A or Supplier B, and hence the construction employee possesses non-public information.

The above examples serve to illustrate the point that an employee at a major firm will likely be exposed to non-public information regarding a certain subset of companies or companies from a particular industry.

7.1.2 Reporting Holdings to Employers

Now, how are these employees to be restricted from engaging in insider trading? This is where the employers step in. Typically, the onus of proof is placed on the employers: they are required to verify that their employees do not have any financial holdings that may constitute a conflict of interest. In practice, this usually works as follows:

- Employer compiles a *blacklist* of companies for which employees cannot hold (the companies') assets. Depending on the situation, the blacklist may (or may not) be distributed to the employees.
- Employee is forced to report all their financial holdings to the employer. Typically, this includes providing the employer access (i.e. viewing permission) to all disclosed personal brokerage or trading accounts of the employee.
- Employer periodically checks that none of the employee's holdings are on the blacklist.

Clearly, this checking process is undoubtedly important in ensuring that employees are compliant. However, it can understandably make some employees uncomfortable. Namely, every employee is revealing their *entire portfolio*, and therefore, an employer is gaining a great deal of extra information on the employee.

In practice, a large company will typically have a separate compliance department that deals with checking for conflicts of interest. However, in many major companies, any C-level executive can wield immense power over the company and the various departments, including compliance. Therefore, in practice, these executives may be able to obtain this information.

This is problematic on a number of levels. For many people, a large percentage of their net worth is in the financial markets (ex. 401(k) accounts). Therefore, in having access to all this information, one can essentially track an employee's wealth/value vis a vis the movement of the market. Another issue lies in looking at the aggregated data of employees' holdings. For a large company that hires many people, say IBM or Target, the aggregated holdings of all its employees may be a good sample of the entire market. Therefore, when employees submit permission to make certain trades, the employer is privy to buyer/seller momentum before any trades actually occur. In particular, to an extent, the employer then has an approximate snapshot of the future market, which is clearly non-public.

Therefore, while reporting financial holdings to employers is critical to minimize insider trading by employees, the standard verification procedure (above) can potentially be taken advantage of by malicious employers to learn information on its employees. Hence, ideally, we would like a procedure that better balances transparency (showing that holdings are legal and fair) and confidentiality (not revealing all holdings to employers).

7.2 The Fundamental Problem

Essentially, we wish to verify that for every asset in an employee's financial holdings, the asset is not contained in the blacklist maintained by the employer. Therefore, an equivalent problem is to show that the set of employee holdings and the set of blacklist holdings have an empty intersection. We now proceed via formal notation.

Namely, define A to be the set of all assets. The company maintains a blacklist $A_B \subset A$. An employee e possesses holdings $A_e \subset A$. The problem then is to verify

$$A_B \cap A_e = \emptyset.$$

For example, consider the following toy scenario:

$$A_e = \{\text{Google}, \text{Microsoft}, \text{Apple}\},$$

$$A_B = \{\text{IBM, Oracle}\}.$$

In the current scheme, the employer learns/is provided A_e for each employer e. Therefore the employer will verify that $A_B \cap A_e = \emptyset$, but in the process will learn all of the employee's other stock holdings (Google, Microsoft, Apple). Ideally, however, we would not like this to be the case; we want the employer to learn no addition information about A_e (just verify A_e does not contain IBM or Oracle as elements). With this observation, the fundamental problem is as follows (we assume a blacklist that is not disclosed to employees, as this is the strictest and strongest security constraint):

Problem (Reporting Portfolio Holdings to Employers Problem)**.** *Given a blacklist A_B and a set of employee holdings A_e, we wish to develop a zero-knowledge proof protocol that verifies*

$$A_B \cap A_e = \emptyset.$$

In particular, if

$$|A_B \cap A_e| \neq 0$$

we have the further condition (desire of the protocol) that the employer does not learn anything about A_e and the employee does not learn anything about A_B (beyond what can be deduced from $A_B \cap A_e$). Namely, the parties gain "zero-knowledge" except for $A_B \cap A_e$.

Now that we have formally defined the problem, we now develop and present a zero-knowledge protocol to the tackle the stated problem.

7.3 Design of a Protocol

We now design a protocol to tackle the problem above. We compartmentalize the various aspects of the desired protocol below and using these, present a solution.

7.3.1 Zero-Knowledge Set Intersection

The core of the problem is computing a zero-knowledge set intersection. For an upper bound U, define

$$\mathbb{N}_{\leq U} = \{n \in \mathbb{N} \mid n \leq U\}.$$

Let $A, B \subset N_{\leq U}$. We wish to compute $A \cap B$. We now make a critical observation. Since $N_{\leq U}$ is a finite set, we can write

$$A = \{v_1, v_2, \cdots, v_n\},$$

$$B = \{w_1, w_2, \cdots, w_m\}.$$

Now, consider a different representation of this set. Define the following polynomial representations of the sets as follows:

$$p_A(x) = (x - v_1)(x - v_2) \cdots (x - v_n),$$

$$p_B(x) = (x - w_1)(x - w_2) \cdots (x - w_m).$$

Let f, g be two random polynomials. We then have the following claim:

Claim 7.1. *Let A, B be two sets and p_A, p_B be their polynomial representations (as above). For random polynomials f, g, define*

$$u(x) = p_A(x) \cdot f + p_B(x) \cdot g.$$

Then

$$s \in A \cap B \implies u(s) = 0.$$

Proof. If $s \in A \cap B$, then $s \in A$ and $s \in B$. Therefore, $p_A(s) = p_B(s) = 0$. But then,

$$u(s) = p_A(s) \cdot f(s) + p_B(s) \cdot g(s)$$

$$= 0 \cdot f(s) + 0 \cdot g(s) = 0,$$

as desired. □

Therefore this polynomial representation of a set allows us to transform from the elements of a set to the roots of a polynomial, which is much easier to massage and use algebraically. Using this observation, we may present a zero-knowledge set-intersection protocol (courtesy of [32]).

Definition 7.2 (Zero-Knowledge Set Intersection Protocol, variant of [32])**.** *Agent 1 and Agent 2 possess respective sets $A, B \subset \mathbb{N}_{\leq U}$, where $|A| = n, |B| = m$. Let p be a large prime such that $U << p$. We then have the following set-intersection protocol:*

- *Agents compute the polynomial representation of their respective sets, p_A and p_B. They send p_A, p_B respectively to a Trusted Third Party T*

- *Third Party T randomly picks polynomials f, g such that:*

$$f = \sum_{i=0}^{max(n,m)} f[i]x^i, \quad g = \sum_{i=0}^{max(n,m)} g[i]x^i$$

where for all $0 \leq i \leq max(n, m)$,

$$f[i] \leftarrow \mathbb{Z}_p, \quad g[i] \leftarrow \mathbb{Z}_p$$

(coefficients of f and g are chosen independently and uniformly at random from $\mathbb{Z}_p$).

T then computes a new polynomial

$$u = p_A \cdot f + p_B \cdot g \pmod{p}$$

and sends u to the agents.

- *Agents 1 and 2 (respectively) determine $A \cap B$ as follows:*

$$(A \cap B)_1 = \{a \in A \mid u(a) = 0\}$$
$$(A \cap B)_2 = \{b \in B \mid u(b) = 0\}$$

We are guaranteed that the true intersection is contained within the respective calculations i.e

$$A \cap B \subset (A \cap B)_1$$
$$A \cap B \subset (A \cap B)_2.$$

In particular, note the computations $(A \cap B)_1$ and $(A \cap B)_2$ may contain additional elements beyond $A \cap B$ (if $\gcd(p_A, g) \neq 1$ or $\gcd(p_B, f) \neq 1$). However, since the coefficients of f and g are drawn randomly from $\mathbb{Z}_p$ and $U << p$, f or g will share a root with p_B or p_A respectively with extremely small probability. Another way to think of this is that this protocol has extremely small *completeness error*. (To make this completeness error arbitrarily small, One could run multiple iterations and take the intersection of the calculated sets over all the iterations). Note this is our first example of a protocol that does not exhibit perfect completeness!

In addition, the procedure above will terminate quickly, since polynomial multiplication requires at worst $\Theta(n^2)$ run-time, and hence the protocol is efficient.

It remains to show this procedure satisfies the zero-knowledge property. [32] showed the following lemma:

Claim 7.3 ([32])**.** *Let f, g be selected as in the procedure above. Let*

$$r = max(2n, 2m) - deg[gcd(p_A, p_B)].$$

Write

$$u = p_A \cdot f + p_b \cdot g \pmod{p}$$
$$= gcd(p_A, p_B) \cdot \sum_{i=0}^{r} u[i]x^i \pmod{p}.$$

Then, for all $0 \leq i \leq r$, the u_i are (over the choice of f, g) distributed independently and uniformly at random over $\mathbb{Z}_p$.

The proof of the claim is quite long and involved, hence we omit it here. However, this result essentially directly gives us zero-knowledge, as we show in the following claim.

Claim 7.4 (Adapted from [32])**.** *The protocol above is zero-knowledge; one does not learn any more information about A or B beyond $A \cap B$.*

Proof. Here, we are essentially trying to show zero-knowledge conditioned on knowing $A \cap B$. As noted in the previous claim, the protocol outputs the polynomial

$$u = \gcd(p_A, p_B) \cdot \sum_{i=0}^{r} u[i]x^i$$

where the u_i are distributed independently and uniformly at random over $\mathbb{Z}_p$ in the output distribution. Therefore, for zero-knowledge, we simply need to show the existence of a simulator that matches the output distribution. Namely, if the output of the protocol matches the output of a simulator that just knows $A \cap B$, this implies zero-knowledge beyond $A \cap B$. We present the following simple simulator:

- Compute the polynomial representation $p_{A \cap B}$ of $A \cap B$
- Select a random polynomial $u' \leftarrow {Z_p}^r[x]$ (i.e. polynomial of degree r with coefficients in Z_p)
- Output $\overline{u} = p_{A \cap B} \cdot u'$.

Observe that the polynomial representation $p_{A \cap B}$ is equivalent to $\gcd(p_A, p_B)$. In addition, from the previous lemma and the construction, coefficients $u[i]$ and $u'[i]$ are both selected at random. Therefore, clearly the distributions

$$u \approx \overline{u}$$

are equivalent. Therefore, the protocol is zero-knowledge as desired, as (conditioned on knowing $A \cap B$) either agent can independently run the above simulation. □

Therefore, we have demonstrated a (honest-verifier, honest-prover; see note below) zero-knowledge set-intersection protocol.

NB: Our notion of a zero-knowledge protocol here may seem a bit confusing/different. Previously, our protocols had a separate Prover and Verifier. However, in this case, both agents are behaving as both Provers and Verifiers. Hence, since each agent is engaging in both roles, it doesn't really make sense to talk about a "malicious verifier" and "malicious prover". Therefore, for the sake of this protocol, it is implicitly a *honest-verifier and honest-prover* zero-knowledge protocol (i.e zero-knowledge only if both parties are honest).

7.3.2 Mapping Assets to Values

Now, note our zero-knowledge set intersection protocol in the previous section utilized sets of natural numbers. However, the assets are not inherently values; therefore we need to have some way of mapping assets to natural numbers. In the protocol above, the arithmetic was in the ring $\mathbb{Z}_p$. Therefore, a natural candidate is some mapping from the set of assets A into $\mathbb{Z}_p$. Namely,

Definition 7.5 (Asset to Value Map)**.** *We denote a mapping*

$$\mathcal{M} : A \to \mathbb{Z}_p$$

to be a valid asset to value map if $\mathcal{M}$ is injective.

In our instance, we are guaranteed the existence of such a map, since $|A| = U << p$, and hence there are

$$\binom{p}{U}$$

choices of such a map. The use of such a mapping will allow us to utilize the protocol defined in the prior section.

7.3.3 Homomorphic Encryption with Polynomials

We may consider a point of concern regarding the set-intersection protocol in Section 7.3.1. Namely, it heavily relied on the use of a Trusted Third Party T; if T were to obtain the asset to value mapping used, it can essentially determine the holdings of the employer and the blacklist companies of the employer. Yet, the role of T in doing calculations is undoubtedly important. How can we get around this? Ideally we would like (1) the agents to send obfuscated versions of their polynomials and (2) the third party T to still be able to do the required calculations with these obfuscated polynomials.

This is the idea behind homomorphic encryption (see [22, 4]), a form of encryption that allows computation on encrypted data. It allows computation on ciphertexts, generating an encrypted result which, when decrypted, matches the result of the operations as if they had been performed on the plaintext.

For example if we have an encryption algorithm $E : R_1 \to R_2$, such that for $r, r' \in R_1$,

$$E(r + r') = E(r) + E(r')$$

where '+' is the addition operator in the rings R_1, R_2, we say E is an additive homomorphism. If

$$E(r \cdot r') = E(r) \cdot E(r')$$

where '$\cdot$' is the multiplicative operator in the rings R_1, R_2, we say E is a multiplicative homomorphism.

An encryption that supports both additive and multiplicative homomorphisms is a fully homomorphic encryption. Namely, a *fully homomorphic encryption* scheme is one that allows arbitrary arithmetic functions f to be computed on encrypted data (see below image for a depiction).

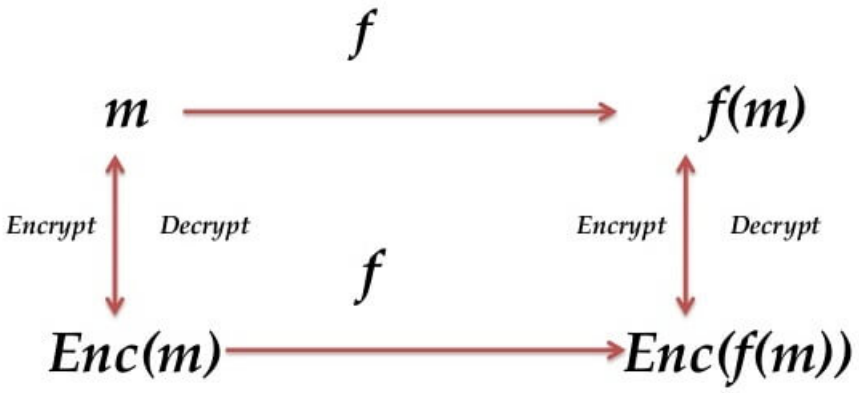


Figure 7.1: Depiction of a homomorphic encryption scheme from [36]

Fully homomorphic encryption is a relatively recent construction (see [19] for the original presentation of a fully homomorphic encryption scheme) and hot area of study (see [4] for an overview on the history). For the sake of this section, we assume the existence of an efficient fully homomorphic encryption scheme. As we will discuss later in the chapter, however, this may not be a totally reasonable assumption.

Let us now return to considering polynomials and the protocol from earlier. Say we have an encryption-decryption pair (E, D) where E is fully homomorphic with domain $\mathbb{Z}_p$. Consider plain-text polynomials

$$f = a_0 + a_1 x + \cdots + a_n x^n$$

$$g = b_0 + b_1 x + \cdots + b_m x^m.$$

Since we don't want to obfuscate the polynomial, we send third party T encrypted polynomials

$$E[f] = E(a_0) + E(a_1)x + \cdots + E(a_n)x^n$$
$$E[g] = E(b_0) + E(b_1)x + \cdots + E(b_m)x^m.$$

Since E is fully-homomorphic, T can compute the standard product of polynomials

$$E[f]E[g] = E(a_0)E(b_0) + [E(a_0)E(b_1) + E(a_1)(b_0)]x + \cdots + E(a_n)E(b_m)x^{n+m}$$
$$= E(a_0 b_0) + [E(a_0 b_1) + E(a_1 b_0)]x + \cdots + E(a_n b_m)x^{n+m}$$
$$= E(a_0 b_0) + [E(a_0 b_1 + a_1 b_0)]x + \cdots + E(a_n b_m)x^{n+m}$$
$$= E[fg]$$

Clearly, this polynomial can then be decrypted via D to reveal fg.

Therefore, the existence of homomorphic encryption allows us to obfuscate polynomials while still being able to do computations on them.

7.3.4 A Protocol for Reporting Portfolio Holdings

Combining the zero-knowledge set-intersection, the asset to value map, and the fully homomorphic encryption with polynomials, we are ready to construct a unified protocol.

In this language $\mathcal{L}$, statements are of the form (using earlier notation)

$$A_e \cap A_B = \emptyset.$$

We have the following protocol

Definition 7.6 (Zero-Knowledge Protocol for Reporting Portfolio Holdings to Employers)**.** *Let A be the (finite) set of all assets, where $|A| = U$. Denote $A_e \subset A$ as the set of employee holdings and $A_B \subset A$ as the blacklist of holdings maintained by the employer, such that $|A_e| = n, |A_B| = m$. Choose p a large prime such that $U << p$. Let (E, D) be an encryption-decryption scheme where E is fully homomorphic with domain $\mathbb{Z}_p$.*

An Honest Verifier V (Employer) verifies (confirms compliance of) an Honest Prover P (Employee) in zero-knowledge, utilizing Trusted Third Parties T_1 and T_2, as follows [agents P, V, T_1 know A and (E, D); all agents know p]:

- *T_1 generates a random "Asset to Value Map" $\mathcal{M} : A \to \mathbb{Z}_p$, and sends $\mathcal{M}$ to Prover P and Verifier V*
- *Prover P computes set $V_e = \mathcal{M}(A_e)$ (image of the values in A_e), and computes the polynomial representation p_e of the set. Prover P sends $E(p_e)$ to T_2*
- *Verifier V computes set $V_B = \mathcal{M}(A_B)$ (image of the values in A_B), and computes the polynomial representation p_B of the set. Verifier V sends $E(p_B)$ to T_2*
- *T_1 randomly picks polynomials f, g such that:*

  $$f = \sum_{i=0}^{U} f[i]x^i, \quad g = \sum_{i=0}^{U} g[i]x^i$$

  where for all $0 \leq i \leq U$,

  $$f[i] \leftarrow \mathbb{Z}_p, \quad g[i] \leftarrow \mathbb{Z}_p$$

  (coefficients of f and g are chosen independently and uniformly at random from $\mathbb{Z}_p$).

  T_1 sends $E(f)$ and $E(g)$ to T_2
- *T_2 computes a new polynomial*

  $$u_{encrypted} = E(p_e) \cdot E(f) + E(p_B) \cdot E(g) = E(p_e \cdot f + p_B \cdot g)$$

  T_2 sends $u_{encrypted}$ to Verifier V
- *Verifier V decrypts with D: $D(u_{encrypted}) = u_{decrypted}$. Verifier V checks*

  $$[A_e \cap A_B]_{approx} = \{a \in A_B \mid u_{decrypted}(\mathcal{M}(a)) = 0\} = \emptyset$$

  and accepts/rejects accordingly

The above protocol serves as a zero-knowledge proof protocol for the language $\mathcal{L}$ for an honest verifier and honest prover. We verify the properties below.

Completeness

Take an $\ell \in \mathcal{L}$. This indicates $A_e \cap A_B = \emptyset$. With overwhelmingly high probability (since $U << p$), $\gcd(p_B, f) = 1$, and therefore $[A_e \cap A_B]_{\text{approx}} = \emptyset$ as desired. There is only a small probability of rejecting in this case. As we discussed in section 7.3.1, this completeness error can me made arbitrarily small, by running multiple iterations and taking the intersection of the $[A_e \cap A_B]_{\text{approx}}$ generated in each trial. Therefore, completeness is satisfied.

Soundness

Here we have an $\ell \notin \mathcal{L}$ (note, however, we still assume honest prover here). Therefore, there exists a $c \in A_e \cap A_B$. But (since verifier and prover are truthful) this then implies

$$u_{\text{decrypted}}(\mathcal{M}(c)) = p_e(c) \cdot f(c) + p_B(c) \cdot g(c) = 0 \cdot f(c) + 0 \cdot g(c) = 0,$$

and therefore the protocol will reject 100% of the time in this case. Hence, we have perfect soundness.

Zero-Knowledge

The protocol is zero-knowledge, as follows directly from Claim 7.4 earlier. In fact, we actually have a "stronger" kind of zero-knowledge; namely, Claim 7.4 implies that even if $|A_e \cap A_B| > 0$, (assuming honest prover and honest verifier) the Verifier V gains no additional information about A_e (beyond what can be deduced from $A_e \cap A_B$).

Therefore, we have indeed designed an (honest verifier, honest prover) zero-knowledge protocol that serves as a solution to the "Reporting Portfolio Holdings to Employers Problem". This protocol can be used to better balance financial transparency and confidentiality for personal employee compliance.

7.4 Potential Limitations

7.4.1 Honest Prover and Honest Verifier

A clear limitation that may be levied against our protocol is its reliance on an honest prover and an honest verifier. This is undoubtedly a limitation, as the protocol is not "clasically" zero-knowledge, rather only zero-knowledge in this case of honest verifiers and honest provers. Regardless, this protocol is useful in that it is a non-strict security improvement over the alternative, current protocol (i.e. employees reporting all holdings to employers).

Consider the perspective of the employee, the prover. In the standard protocol, the employer will learn his entire reported set A_e. However, in the protocol described above, this is only a worst case scenario; namely, the employer is guaranteed to fully learn A_e only if he submits $A_B = A$ (the entire set of holdings). Note in either the standard protocol or the protocol defined above, the employee has equal ability to lie about A_e. Therefore, it follows our protocol developed here is a non-strict improvement for the prover P.

Now, consider the perspective of the employer, the verifier. In either the standard protocol or the protocol defined above, the employee will learn nothing additional (i.e. only provided the assets that clash) about A_B. In addition, in either the standard protocol or the protocol defined above, the employee has equal ability to lie about A_B. Therefore, it follows our protocol developed here is a non-strict improvement for the verifier V.

Therefore, this can serve as a useful protocol. Namely, as long as some agents behave truthfully, this developed protocol marks a positive improvement to society with respect to the transparency and confidentiality trade-off.

7.4.2 Trusted Third Party

Another complaint that may be stated is the heavy use of third parties. The use of the third parties is the operative mechanism by which we inject randomness into the protocol. Namely, we utilize T_1 to generate the random polynomials f, g and hence guarantee the zero-knowledge property. In addition, since we want to minimize the information a single third party can gather, we utilize T_2 to make the calculation. The reason for this is that if any of $P, V,$ or T_1 make the aggregated polynomial calculation, since they possess knowledge $\mathcal{M}$ and (E, D), they can essentially deduce A_e or A_B. Therefore, the utilization of two separate Third Parties T_1, T_2 guarantees that neither learns A_e or A_B unless agents T_1 and T_2 collude.

Who might be a natural candidate for the trusted third parties then to prevent collusion? A natural candidate for T_1 may be the SEC or other federal regulator. They have the least incentive to act maliciously. In addition, as regulators they are likely to have good knowledge of the space of possible financial assets and therefore will be will equipped to construct a mapping $\mathcal{M}$. As for T_2, note all T_2 is doing is multiplying and adding polynomials. Therefore, T_2 can be essentially anyone (independent from P, V, T_1), since T_2 won't learn anything since he/she has no knowledge of (E, D) or $\mathcal{M}$.

Therefore, while the use of third parties is admittedly not ideal, the protocol ensures security given that T_1 and T_2 are independent, non-colluding agents (and neither are colluding with P, V).

7.4.3 Fully Homomorphic Encryption

Our protocol assumed E to be an efficient, fully homomorphic encryption scheme. While fully homomorphic encryption schemes (that are homomorphic with respect to both addition and multiplication) have been constructed (see [19] for the earliest instance), early implementations of fully homomorphic encryption were prohibitively slow. However, in recent years, there has been much work into developing more efficient fully homomorphic encryption schemes (see [51]). As research continues in this hot area, it is conceivable that increasingly efficient homomorphic encryption schemes and implementation will continue to be developed. In addition, depending on the sizes of A_e and A_B, the inefficiency of E may be negligible in terms of absolute run-time.

One may note that even if we drop the usage of (E, D), the party T_2 may not learn about A_e or A_B due to T_2 not knowing the mapping $\mathcal{M}$. While this is true, dropping the use of (E, D) may still compromise the security of the protocol. For example, consider an eavesdropper who is listening in on communication channels involving the Prover P. Then,

the eavesdropper may learn $\mathcal{M}$ when T_1 transmits it to Prover P, and learn p_e when P transmits it to T_2. But, with $\mathcal{M}$ and p_e, the eavesdropper learns A_e! However, if we have an extra layer of security, namely (E, D), an eavesdropper will not learn A_e since (E, D) is never transmitted in the protocol. Therefore, this encryption scheme offers an additional layer of security.

Chapter 8

Application 2: Risk Assurance for Financial Holders

In this chapter, we will consider the second of our applications of zero-knowledge to financial regulation. We will introduce the importance of a fund revealing that its investments satisfy investor constraints and discuss potential issues inherent in this. With this understanding, we then present a zero-knowledge protocol, courtesy of [47], by which a fund can prove its investments fit within particular risk constraints. The protocol we present herein will utilize zero-knowledge interval proof protocols.

8.1 Introduction

8.1.1 Setting Limitations on Investment Funds

For the grand majority of individuals, a portion of their money is being managed by another agent, be it a hedge fund or other investment/fiduciary agent. For example, many people place their retirement savings (such as in the form of 401(k) plans) to be managed/invested by external agents. Similarly, pension funds, funds from which an agent's pension is to be paid after retirement, often provide capital to or invest in various hedge funds and private equity firms. Therefore, an individual's wealth (and stability) is often a function of the performance of these other financial agents.

Now, imagine an individual giving a large chunk of capital to an investment fund. Ideally, the individual hopes the fund makes money i.e. generating positive return on his/her capital investment. But is just maximizing expected returns sufficient for the individual? For example, consider if the fund uses the capital to invest solely in Bitcoin, as the fund believes (per its own "analysis") that Bitcoin offers the highest expected return. Understandably, the individual may be extremely opposed to this strategy due to the high risk that Bitcoin carries. Therefore, the individual may request the fund to only invest in lower risk assets.

The above example serves to illustrate that an individual will usually wish to impose certain risk limitations on their invested capital.

8.1.2 Providing Aggregate Risk Measures to Holders

Now, a fund will therefore be placed under certain restrictions by its investors. How then does a fund convince its investors that its investments follow the stated restrictions? Typically, the fund will provide aggregated risk measures to illustrate compliance. In practice, this usually works as follows:

- Fund compiles the list of investments made using the investor's capital.
- Fund determines the risk measure (ex. volatility) associated with each financial holding.
- Fund computes a weighted average of the individual risk measures, and reports this aggregated risk quantity to the investors.

The idea is that typically investors will place restriction on aggregate risk measures; therefore, by reporting this aggregate measure, the fund can confirm compliance to the investor. In addition, funds will typically wish to maintain their financial holdings a secret (otherwise their strategy can be replicated/copied directly, compromising their business). Therefore, in reporting the aggregate risk measure as opposed to the actual portfolio, secrecy is maintained. Hence, this procedure appears a nice balance between transparency and confidentiality that we desire. What exactly then is the problem here?

For one, in learning the the specific aggregated risk number, the investor may be able to impute additional information about the fund's holdings (such as the portion of the portfolio invested in certain asset classes, i.e. stocks or bonds), since the individual risk numbers are assumed to be public.

In addition, on a more sinister level, the fund could simply make up a compliant, aggregated number. Since the fund is just reporting the summary number, in the situation of a later audit by the SEC and federal regulators, a fund could potentially go back and cook its books to feign compliance and accuracy of the reported summary numbers. In particular, the current protocol is limited in that it generates no paper trail.

Having a paper trail is extremely important given the complexity of the financial system. There are usually chains of investment, i.e. (1) Investor A invests in Fund B, (2) Fund B invests in Fund C, (3) Fund C invests in Fund D, and so on. Each of these investors will have different priorities (see below image for an example) and will be subject to separate restrictions.

Yet, any wrong-doing or malicious behavior by any fund in the chain can serve to harm the investor. A paper trail would provide the ability to zero-in on where in the chain this malicious behavior (i.e. not following risk constraints) occurred. This reflects what occurred during the financial crisis of 2008. The lack of a paper trail made it difficult to pinpoint precisely where in the chain funds were behaving improperly; in consequence, this led to a great deal of speculative finger pointing, such as a (likely unjustified) demonization of pension funds.

Therefore, while it is critical for investors to ensure funds follow investors' risk constraints, the standard verification procedure (above) is flimsy as funds can easily lie due to the lack of a paper trail. Hence, ideally, we would like a risk assurance procedure that better balances

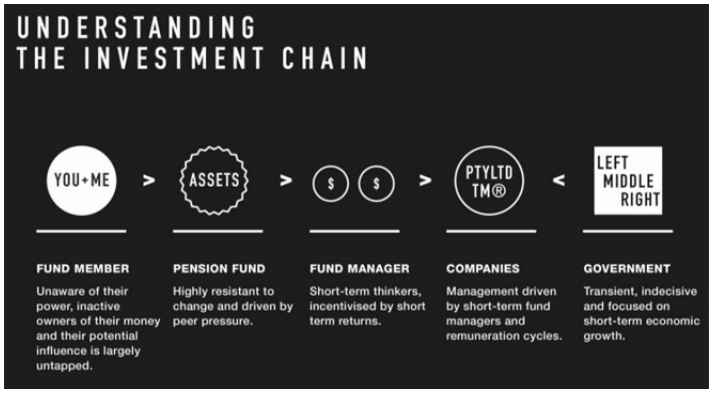


Figure 8.1: Depiction of the investment chain from [50]

transparency (fund required to prove that they satisfy risk constraints and generate a paper trail) and confidentiality (investor learns nothing about the fund's specific holdings).

8.2 The Fundamental Problem

Essentially, we wish to verify that every aggregated risk measure satisfies the constraints set by the investor. Therefore, an equivalent problem is to show that the aggregated risk measure falls within a specified interval.

Namely, define the set of financial assets

$$A = \{A_1, \cdots, A_U\}$$

as a finite set such that $|A| = U \in \mathbb{Z}$. Define a weighted portfolio of a fund as

$$W = \{w_1, \cdots, w_U\}$$

where w_i denotes the portion of the fund's portfolio (AUM) invested in asset A_i. Note

$$\sum_{i=1}^{U} w_i = 1.$$

Say the fund is limited by an aggregated risk constraint "r" to lie in $[Q_0, Q_1]$. In other words, if the individual risk constraint of an asset i can be represented as r_i, the problem then is to verify

$$Q_0 \leq r = \sum_{i=1}^{U} w_i r_i \leq Q_1$$

where each r_i is public.

Note when we use the ambiguous language of "risk constraint", we simply are referring to some factor on which a limitation is imposed by the investor. A classic example may be volatility (standard deviation of trading price, measured as a percentage). For now, we will continue using this general language; as we will later show, this construction allows us to verify a great variety of properties.

Problem (Risk Assurance for Financial Holders Problem)**.** *Given a weighted portfolio W of a fund and a factor f subject to the risk constraint $f \in [Q_0, Q_1]$, we wish to develop a zero-knowledge proof protocol that verifies*

$$Q_0 \leq f = \sum_{i=1}^{U} w_i f_i \leq Q_1$$

where the values w_i are public.

In particular, we have the further condition (desire of the protocol) that the fund "locks-in" the values w_i i.e. generates a paper trail.

8.3 Design of a Protocol

We now present a protocol from [47] to tackle the problem above. We compartmentalize the various aspects of the desired protocol below and using these, present a solution

8.3.1 Pedersen Commitments

In Section 4.2.2 we introduced the general notion of *commitment schemes*. Commitment schemes essentially serve to lock a prover into a particular value. Therefore, this property is a natural candidate for us, as we wish some way to generate a paper trail. Here we will present a particular type of commitment scheme, known as *pedersen commitments* (see [42]).

Definition 8.1 (Pedersen Commitment, [42])**.** *Let p, q be two large primes such that $q \mid p-1$. Consider elements $g, h \in \mathbb{Z}_p$ that each have order q, and such that the discrete logarithm $\log_g(h)$ is unknown. Then for public (known) parameters (p, q, g, h), a Pedersen commitment to $x \in \mathbb{Z}_q$ with randomness $r \in \mathbb{Z}_q$ is the value*

$$C_r(x) = g^x h^r \pmod{p}.$$

Namely, an agent who is provided (p, q, g, h) "commits" to the value x (via imparting randomness r).

Pedersen commitments are useful in that they have the following interesting linear (homomorphic) properties:

$$C_r(x)^t = (g^x h^r)^t = g^{tx} h^{tr} = C_{tr}(tx)$$
$$C_r(x) C_{r'}(x') = (g^x h^r)(g^{x'} h^{r'}) = g^{x+x'} h^{r+r'} = C_{r+r'}(x + x').$$

Therefore, given the commitments to any two values x, x' and for $m, m' \in \mathbb{Z}$, one can directly compute a commitment to the linear combination to the value $mx + m'x'$:

$$C_r(x)^m C_{r'}(x')^{m'} = C_{mr+m'r'}(mx + m'x').$$

Namely, Pedersen commitments are useful in that they allow us to compute linear combinations.

8.3.2 Bit Proof Protocol

Now, consider if we commit to a value x via a Pedersen commitment i.e. $C_r(x) = g^x h^r$. Our primary intention is to show that $x \in [Q_0, Q_1]$ (lies in an interval) in zero-knowledge. Here, let us consider a simplification: a protocol to show $x \in \{0, 1\}$ in zero-knowledge. As we will show in the following section, we can build off of this protocol to develop a zero-knowledge interval proof protocol.

Consider the following protocol (described in [20]),

Definition 8.2 (Bit Proof Protocol, [20])**.** *Let $x \in \{0, 1\}$ and $C_r(x) = g^x h^r$ be its Pedersen commitment with public parameters (p, q, g, h). Then we have the following zero-knowledge protocol between a Prover P and Verifier V that $x \in \{0, 1\}$.*

- *Prover P selects random $r \in \mathbb{Z}_q$ and computes and sends the commitment $C = C_r(x) = g^x h^r$ to Verifier V.*
- *If $x = 0$, Prover P selects random $w, y_1, e_1 \in \mathbb{Z}_q$, computes*

$$v_0 = h^w \pmod{p}, \quad v_1 = h^{y_1} \left(\frac{C}{g}\right)^{-e_1} \pmod{p},$$

  and sends (v_0, v_1) to Verifier V

  If $x = 1$, Prover P selects random $w, y_0, e_0 \in \mathbb{Z}_q$, computes

$$v_0 = h^{y_0} C^{-e_0} \pmod{p}, \quad v_1 = h^w \pmod{p},$$

  and sends (v_0, v_1) to Verifier V

- *Verifier V selects random $e \in \mathbb{Z}_q$ and sends to Prover P*
- *If $x = 0$, Prover P computes*

$$e_0 = e - e_1 \pmod{q}, \quad y_0 = w + re_0 \pmod{q}$$

  and sends (y_0, y_1, e_0, e_1) to Verifier V

  If $x = 1$, Prover P computes

$$e_1 = e - e_0 \pmod{q}, \quad y_1 = w + re_1 \pmod{q}$$

  and sends (y_0, y_1, e_0, e_1) to Verifier V

- *Verifier V verifies*

$$e = e_0 + e_1 \pmod{q}$$

$$v_0 = h^{y_0} C^{-e_0} \pmod{p}, \quad v_1 = h^{y_1} \left(\frac{C}{g}\right)^{-e_1} \pmod{p}$$

  and accepts/rejects accordingly.

We verify the non-obvious equalities to ensure the protocol checks out:

- (If $x = 0$),
$$v_0 = h^{y_0} C^{-e_0} = h^{w+re_0}(h^r)^{-e_0} = h^w$$
- (If $x = 1$),
$$v_1 = h^{y_1}\left(\frac{C}{g}\right)^{e_1} = h^{w+re_1}(h^r)^{-e_1} = h^w$$

The protocol essentially asks a prover to prove that he/she knows a solution to both $\log_h(C)$ and $\log_h(\frac{C}{g})$. However, the prover only knows a solution to one of these (i.e. depending on whether $x = 0$ or 1). Therefore, the selection of y_{1-x}, e_{1-x} by the prover serves to "cheat" the solution he/she does not know. The solution he does know is confirmed via a process analogous to the discrete log protocol from Chapter 1. We omit the checking of zero-knowledge properties here (see [20, 47] for a brief outline). Note that since the prover is making random selections w, y_{1-x}, e_{1-x}, the distribution of output to the verifier in the protocol is indistinguishable for $x \in \{0, 1\}$, and therefore zero-knowledge follows.

For the sake of notation, throughout the rest of this chapter, we will use the notation

$$\text{ZKBIT}[C_r(x)]$$

to denote the zero-knowledge bit proof protocol described here that verifies the value $x \in \{0, 1\}$.

8.3.3 Zero-Knowledge Interval Proof

In the previous section we showed a protocol that verifies a committed value $x \in [0, 1]$. We will now extend this protocol to show a protocol that verifies a commited value $x \in [Q_0, Q_0 + 2^k - 1]$ for $k \in \mathbb{Z}$. The core leap is to consider the base 2 representation of $x - Q_0$:

$$x - Q_0 = \sum_{i=0}^{k-1} x_i 2^i.$$

If we can demonstrate that every $x_i \in [0, 1]$ (i.e. via ZKBIT), we may then conclude that $x - Q_0 \in [0, 2^k - 1] \implies x \in [Q_0, Q_0 + 2^k - 1]$. Hence, we will again make usage of Pedersen commitments.

Consider the following protocol (described in [8]),

Definition 8.3 (Interval Proof Protocol, [8])**.** *Let $x \in [Q_0, Q_0 + 2^k - 1]$ (for $k \in \mathbb{Z}$) and $C_r(x) = g^x h^r$ be its Pedersen commitment with public parameters (p, q, g, h). Then we have the following zero-knowledge protocol between a Prover P and Verifier V that $x \in [Q_0, Q_0 + 2^k - 1]$.*

- *Prover P selects random $r \in \mathbb{Z}_q$ and computes the commitment of x,*
$$C = C_r(x) = g^x h^r \pmod p,$$
and sends C to Verifier V

- *Prover P computes the binary representation of $x - Q_0$,*

$$x - Q_0 = \sum_{i=0}^{k-1} x_i 2^i.$$

For each $1 \leq i \leq k-1$, Prover P selects random $r_i \in \mathbb{Z}_q$, and sets

$$r_0 = r - \sum_{i=1}^{k-1} r_i 2^i \pmod{p}.$$

Prover P then computes the commitment of x_i (for $0 \leq i < k$),

$$C_i = C_{r_i}(x_i) = g^{x_i} h^{r_i} \pmod{p},$$

and sends the pairs (i, C_i) to Verifier V

- *Prover P and Verifier V engage in the bit proof protocols*

$$\text{ZKBIT}[C_i]$$

to certify that for all i, $x_i \in \{0, 1\}$ (if not, verifier rejects)

- *Verifier V verifies*

$$\frac{C}{g^{Q_0}} = \prod_{i=0}^{k-1} (C_i)^{2^i} \pmod{p}$$

and accepts/rejects accordingly.

Essentially this protocol simply utilizes a series of bit proof protocols to certify that the base 2 representation of the number x is valid, and does a final calculation check to certify that the product of the commitments of the base 2 decomposition yield the original commitment of x. Note this final verification step checks out as, using the homomorphic properties of Pedersen commitments,

$$\prod_{i=0}^{k-1} (C_i)^{2^i} = \prod_{i=0}^{k-1} (g^{x_i} h^{r_i})^{2^i} = \left(g^{\sum_i x_i 2^i}\right) \left(h^{\sum_i r_i 2^i}\right) =$$

$$g^{x - Q_0} h^r = \frac{g^x h^r}{g^{Q_0}} = \frac{C}{g^{Q_0}} \pmod{p}$$

as desired. These proofs are known to be reasonably efficient as long as k is not too large.

The protocol is zero-knowledge, as this follows directly from the fact that ZKBIT is zero-knowledge and since the r_i are randomly drawn from $\mathbb{Z}_p$. See [8] for a more explicit proof that the protocol is zero-knowledge.

Note that our protocol only shows that $x \in [Q_0, Q_0 + 2^k - 1]$ for $k \in \mathbb{Z}$. However, ideally we would like bounds on either side, i.e. to be able to show $x \in [Q_0, Q_1]$. In fact, we can do this using two iterations of the interval proof protocol. In particular, consider the quantity

$$K = \lceil \log_2(Q_1 - Q_0 + 1) \rceil$$

where $\lceil v \rceil$ denotes the ceiling function of v (i.e. smallest $v \in \mathbb{Z}$ such that $s \geq x$). We may then run two separate interval proof protocols, one for $x \in [Q_0, Q_0 + 2^K - 1]$ and one for $x \in [Q_1 - (2^K - 1), Q_1]$. Since

$$Q_1 - Q_0 < 2^K - 1,$$

it then follows that

$$Q_1 \in [Q_0, Q_0 + 2^K - 1], Q_0 \in [Q_1 - (2^K - 1), Q_1],$$

and therefore we have shown a protocol for

$$x \in [Q_0, Q_0 + 2^K - 1] \cap [Q_1 - (2^K - 1), Q_1] = [Q_0, Q_1].$$

For the sake of notation, throughout the rest of this chapter, we will use the notation

$$\text{ZKINTERVAL}_{[Q_0,Q_1]}[C_r(x)]$$

to denote the zero-knowledge bit proof protocol described here that verifies the value $x \in [Q_0, Q_1]$.

8.3.4 A Protocol for Risk Assurance

Now that we have built up the machinery to show zero-knowledge interval proofs for any arbitrary interval $[Q_0, Q_1]$, we may present a unified protocol from [47].

In this language $\mathcal{L}$, statements are of the form (using earlier notation)

$$Q_0 \leq f = \sum_{i=1}^{U} w_i f_i \leq Q_1.$$

We have the following protocol (from [47])

Definition 8.4 (Zero-Knowledge Protocol for Risk Assurance for Financial Holders, [47])**.** *Let A be the set of financial assets such that $|A| = U \in \mathbb{Z}$. Denote W be the weighted portfolio of a fund such that w_i denotes the portion of the portfolio invested in asset A_i. Consider a risk factor f, where f_i denotes the (publicly known) individual risk factor of asset A_i, subject to the constraint $f \in [Q_0, Q_1]$ by the investor. Let us also have a Pedersen commitment scheme with public parameters (p, q, g, h).*

A Verifier V (Investor) verifies (confirms compliance of) an Honest Prover P (Fund) in zero-knowledge, utilizing Trusted Third Party T_1 as follows:

- *Prover P selects random r_i (for each $1 \leq i \leq U$) and commits to w_i,*

$$C_i = C_{r_i}(w_i) = g^{w_i} h^{r_i} \pmod{p}.$$

Prover P calculates

$$r = \sum_{i=1}^{U} f_i r_i \pmod{p}$$

and commits to f,

$$C = C_r(f) = g^f h^r \pmod{p}.$$

Prover P sends pairs (A_i, C_i) *and C to Verifier V*

Prover P sends tuples (A_i, C_i, w_i, r_i) *to the Trusted Third Party T*

- *Prover P and Verifier V engage in the interval proof protocol*

$$\text{ZKINTERVAL}_{[Q_0,Q_1]}[C]$$

to certify that $f \in [Q_0, Q_1]$ *(if not, verifier rejects)*

- *Verifier V verifies*

$$C = \prod_{i=1}^{U} C_i^{f_i} \pmod{p}$$

and accepts/rejects accordingly.

- *(In case of dispute, Trusted Third Party is queried to check the legitimacy of the commitments and the values* w_i*)*

The above protocol serves as a zero-knowledge proof protocol for the language $\mathcal{L}$ for a verifier and honest prover. We verify the properties below.

Completeness

Take an $\ell \in \mathcal{L}$. This indicates $Q_0 \leq f = \sum_{i=1}^{U} w_i f_i \leq Q_1$. Following the protocol (completeness utilizes honest prover and honest verifier), we will have that ZKINTERVAL$[C]$ checks correctly (from completeness of the interval proof protocol) and verify the calculation

$$\prod_{i=1}^{U} C_i^{f_i} = \prod_{i=1}^{U} (g^{w_i} h^{r_i})^{f_i} = \prod_{i=1}^{U} g^{w_i f_i} h^{f_i r_i} =$$

$$g^{\sum_i w_i f_i} h^{\sum_i f_i r_i} = g^f h^r = C \pmod{p},$$

and therefore we will have perfect completeness.

Soundness

Here we have an $\ell \notin \mathcal{L}$ (note, however, we still assume an honest prover here). Therefore, we must have $f \notin [Q_0, Q_1]$. But then, from the soundness of ZKINTERVAL, with high probability, the verifier will reject ZKINTERVAL$[C]$, and hence we have soundness (multiple iterations may be done to make the error arbitrarily small).

Zero-Knowledge

That the protocol is zero-knowledge follows trivially from the fact that the r_i are picked independently and at random, and that ZKINTERVAL is a zero-knowledge proof protocol. Therefore, one can clearly create an efficient simulator to mimic the output distribution of the commitments.

Therefore, we have presented the (honest prover) zero-knowledge protocol from [47] that serves as a solution to the "Risk Assurance for Financial Holders Problem". This protocol can be used to better balance financial transparency and confidentiality for investment risk compliance.

8.4 Potential Limitations

8.4.1 Honest Prover

A clear limitation of our protocol that may be levied is that the protocol requires an honest prover, i.e, for the prover (the fund) to accurately follow the protocol. The reason for this is that at the end of the day, the holdings of the fund need to be self-reported, as even federal agencies such as the SEC will not necessarily possess knowledge of all the fund's holdings (ex. private equity or private debt holdings do not necessarily need to be reported).

However, the role of the Trusted Third Party serves to mitigate this to an extent. Namely, in sending commitments to the Third Party, the fund is essentially forced to generate a paper trail. Therefore, in the case of a conflict or even far into the future, the Third Party can open the commitments and confirm the correctness of these contents, for example comparing to records on file. While this safeguard may not be perfect, as hedge funds are often free from reporting many of their assets, the Third Party can likely still discern whether certain committed holdings are feasible and/or match the return profile of the fund in the time period. Regardless, the existence of this paper trail restricts a fund's ability to be dishonest, as its commitments may be opened and partially verified.

Regardless, this protocol constitutes an improvement over the current protocol. Namely, in this protocol an investor is simply provided a proof that the fund satisfies risk constraints, without providing the actual risk value, and hence better manages fund confidentiality. On the other hand, while the fund can still fully lie in this protocol, the paper trail sent to the Third Party yields a non-trivial risk the fund gets caught, and hence exemplifies better (non-strict) fund transparency.

8.4.2 Trusted Third Party

Another point of concern is the heavy use of the Third Party in the protocol. Namely, the Trusted Third Party is directly provided full information on the holdings of the fund. While this may at first seem a gross breach of confidentiality, it is important to consider the type of agent that may act as the Third Party. Namely, per federal restrictions, hedge funds are required to report certain classes of holdings to the SEC and other federal regulators. In particular, there has been recent discussion of tighter regulations, and hence many believe

that soon funds will be required to report all classes of holdings to the SEC and other federal agencies. Therefore, the SEC is a natural candidate for this Trusted Third Party T.

In particular, given that the SEC is directly involved in fraud prevention, the SEC acting as the Third Party is a natural fit. Investors can query the SEC to verify the legitimacy of their commitments. In addition, as the SEC will then possess data on all funds, it will make it much easier to determine where in a chain of investments (as discussed in Section 8.1), an agent is behaving maliciously. Therefore, this protocol can further enforce the stability and oversight of the financial system.

8.4.3 Varieties of Risk Constraints

Lastly, another important consideration is the types of risk constraints admitted in the current scheme. Our current formulation allows any linear (aggregate) risk constraint

$$f = f_1 w_1 + f_2 w_2 + \cdots + f_U w_U.$$

This allows for constraints such as a weighted average constraint, a proportion constraint (setting all f_i to binary values 0 or 1), etc.

However, this is limiting in that this constraint does not admit any non-linear risk constraint. For example, some constraints have second order terms (for example, a cross-correlation constraint), and hence in this case, the aggregate risk constraint may be of the form

$$f = \sum_{i \neq j} f_{i,j} \cdot w_i w_j.$$

Our Pedersen commitments unfortunately are not homomorphic with respect to products, hence it may appear we are out of luck as we would be unable to compute $C(w_i \cdot w_j)$ from $C(w_i)$, $C(w_j)$. Yet, there is a way around this. Namely, we can instead commit to the second-order terms directly, i.e. provide commitments to $C(w_i \cdot w_j)$ directly and proceed as in the protocol. Since the risk constraints are defined a priori, based on the form of the aggregate risk constraint, the prover (fund) and verifier (investor) will usually be able to determine appropriate commitments for the verification procedure.

Another concern is that we are working over modulo p and hence working with integers. However, clearly f_i, w_i may not be integers. A solution to this is to apply appropriate scaling factors to the f_i, w_i and then round to the nearest integer. Choosing a large enough integer (though nowhere near as large as p or q) for the scaling factor would be sufficient and generate limited error. In particular, a natural candidate would be some scaling factor of

$$10^k \cdot [\text{Fund's AUM}]$$

where $k \in \mathbb{Z}$ (for small k). Essentially we are picking a small k to ensure the requisite number of significant digits. Therefore, this should serve as a close approximation to the desired quantity, and hence the protocol can be easily adjusted/scaled in this way.

In particular, as noted in [47], this protocol can be utilized and modified for a wide variety of risk constraints, namely:

- Individual Asset Bounds: a specific $f_i \in [Q_{0i}, Q_{1i}]$

- Asset Class and Sector Allocation: restricting proportion of assets that may be invested in a certain class of assets
- Factor Exposures: aggregate sensitivities (for example, weighted averages)
- Scenario Analysis
- Trading Volume

Chapter 9

Application 3: Pooling Information by Anonymous Investors

Here we will consider our third application of zero-knowledge to financial regulation. We observe the importance of investor confidentiality, but note the limitations this can lead to, namely Ponzi schemes or other types of financial fraud. With this understanding, in this chapter we will develop a novel zero-knowledge protocol by which anonymous investors can pool information to verify certain aggregate information. The protocol we present herein will utilize a zero-knowledge sum protocol.

9.1 Introduction

9.1.1 Financial Fraud from Investor Anonymity

In December 2008, news of Bernie Madoff's Ponzi scheme shocked the world. How someone was able to dupe thousands of investors seemed baffling. Overnight, the vocabulary of financial fraud entered the standard lexicon, as people displayed an increased weariness towards money managers.

Before we understand the underlying factors governing how Madoff and other similar fraudsters were able to cheat investors, it is helpful to first understand what a *Ponzi scheme* is. Typically when an investor provides capital to a hedge fund or some other money manager, they expect the capital will be invested in various financial markets. They also expect to receive some return to their capital at regular intervals. Hence, in this relationship, the fund or money manager often essentially behaves like a black box: one inputs some capital and receives a regular stream of output returns. Therefore, as long as the investor keeps receiving solid returns, they are usually satisfied and do not question what's going on in the black box. A Ponzi scheme essentially takes advantage of this sentiment. Namely, instead of providing returns derived from financial markets, a Ponzi scheme generates returns for older investors through revenue paid by new investors.

Consider an example. Say there are 10 investors who have each invested \$10 million in a fund. Say the fund promises a 10% annual return. A fradulent fund could convince a new investor to invest \$10 million, and use this money to pay the returns to the old

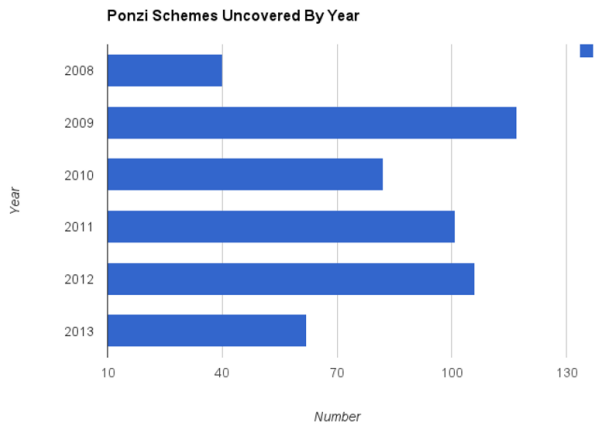


Figure 9.1: Annual statistics on Ponzi schemes from [33]

investors. Therefore, a Ponzi scheme sustains itself through consistently obtaining new sources of capital. In particular, as most investors simply reinvest their returns and rarely pull out any capital, Ponzi schemes can survive simply by persuading most existing participants to reinvest their money, with a relatively small number of new participants.

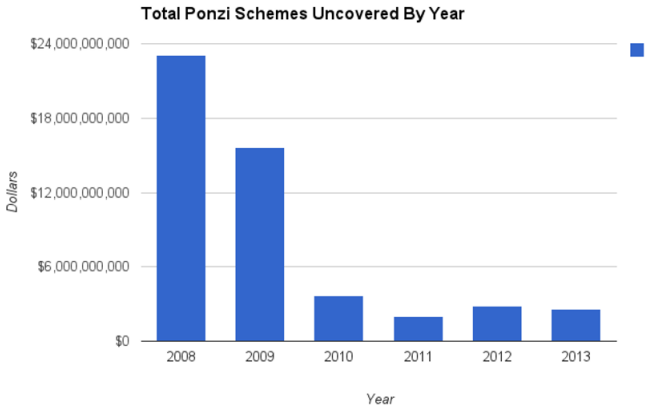


Figure 9.2: Annual statistics on Ponzi schemes from [33]

As seen in the images above, these schemes result in billions of dollars in losses, and significantly harm confidence in the financial system.

Now, how are these schemes able to fly under the radar? The heart of the issue lies in investor anonymity. Namely, an investor in a fund typically does not know who the other investors are. This is a crucial and desired property of the financial system, in that an agents holdings are private. A Ponzi scheme takes advantage of this, essentially providing

an individual plausible transcript for each investor, as any discrepancies that may occur from combining these transcripts will go undetected from the anonymity. This also explains why Ponzi schemes tend to go undetected for years, as they typically collapse in adverse market situations where anxious investors are pulling out their capital from everywhere.

9.1.2 Pooling Information to Prevent Fraud

Hence, these types of fraud are essentially insulated from an individual action. They are really only discovered through (intentional or unintentional) collective action. In particular, one of the clearest signal of a Ponzi scheme is a rapid and consistent growth of total funds under management. While a fund usually reports an amount under management (AUM) number to an investor, the investor has no guarantee this number is accurate. Namely, since the investors are unable to communicate, they cannot verify a reported number.

Therefore, while investor anonymity is crucial, the current lack of a verification process is dangerous as it allows Ponzi schemes to go undetected for periods of time. Hence, ideally we would like a procedure that better balances transparency (verifying certain calculations through pooled information) and confidentiality (maintaining investor anonymity).

9.2 The Fundamental Problem

Here we will restrict ourselves to considering the problem of verifying the aggregate total funds under management number. Essentially, we wish to verify that the reported total funds under management is accurate, i.e. the reported aggregate value equals the sum of the individual investments.

Namely, let I denote the set of investors. Let $x_i \in \mathbb{Z}$ be the amount invested by agent i. Say the fund reports total funds V. The problem then is to verify

$$V = \sum_{i \in I} x_i.$$

Note the amounts here are integers. If we are worried about precision, we can scale the monetary amount by a factor 10^k to ensure all the values we are working with are integers.

Problem (Pooling Information by Anonymous Investors Problem)**.** *Given a set of investors I of a fund, such that x_i denotes the capital committed by investor i, and a reported (by the fund) total AUM of V, we wish to develop a zero-knowledge proof protocol that verifies*

$$V = \sum_{i \in I} x_i.$$

In particular, we have the further condition (desire of the protocol) that no two agents $i, j \in I$ communicate directly i.e. maintain their pairwise anonymity.

9.3 Design of a Protocol

We now design a protocol to tackle the problem above. We compartmentalize the various aspects of the desired protocol below and using these, present a solution.

9.3.1 Additive Homomorphic Encryption

The core of the problem is computing a sum without any direct communication between any two agents. Therefore, a natural solution would be the use of a third party that would receive each value and output the sum to each of the investors. Of course, however, the investors would likely be wary of this third party knowing the volume of their investment.

Hence, a possible adjustment is to utilize an additive homomorphic encryption scheme. In Chapter 7, we introduced the notion of homomorphic encryption, where operations on ciphertexts translate into operations on plaintexts. Here, we are only interested in homomorphism with respect to addition, i.e given encryptions, $E(r), E(r')$ the third party can easily compute $E(r + r')$ without gaining any knowledge of r, r'. Namely, we desire the property

$$E(r) \cdot E(r') = E(r + r').$$

Given an additively homomorphic encryption scheme, the strategy is clear. Simply (1) pass the encrypted amounts to the third party, (2) have the third party compute the encrypted sum of plaintexts, and (3) decrypt the returned ciphertext to determine the sum.

9.3.2 Paillier Cryptosystem

We now introduce a suitable candidate for this additive homomorphic encryption: the Paillier cryptosystem. The use of this cryptosystem is motivated by applications and usage of Paillier in electronic voting (see [9, 12]), in which the additive homomorphic properties are utilized for tallying votes. We now present the cryptosystem.

The Algorithm

Definition 9.1 (Paillier Cryptosystem, Key Generation, variant on [38] from [28])**.**

- *Select two large prime p, q such that p, q are of the same "bit length", i.e*

  $$\lfloor \log_2 p \rfloor = \lfloor \log_2 q \rfloor.$$

  (This length restriction is to ensure $gcd[pq, (p-1)(q-1)] = 1$).
- *Compute $n = pq$ and $\lambda = \phi(n) = (p-1)(q-1)$.*
- *Compute $g = n + 1$ and $\mu = \lambda^{-1} \pmod{n}$.*
- *Outputs keys:*
  - *Encryption key (n, g)*
  - *Decryption key (λ, μ)*

Definition 9.2 (Paillier Cryptosystem, Encryption Step, from [38])**.**

- *Agent wishes to encrypt value $m \in \mathbb{Z}_n$.*
- *Select random $r \in \mathbb{Z}_n$*

- *Compute and output ciphertext*

$$E(m) = g^m r^n \pmod{n^2}.$$

Definition 9.3 (Paillier Cryptosystem, Decryption Step, from [38])**.**

- *Agent wishes to decrypt ciphertext* $c \in \mathbb{Z}_{n^2}$*.*
- *Compute and output plaintext message*

$$D(c) = \left\lfloor \frac{[c^\lambda \pmod{n^2}] - 1}{n} \right\rfloor \cdot \mu \pmod{n}.$$

Note the encryption, decryption pair checks out as

$$D(E(m)) = D(g^m r^n) = \left\lfloor \frac{[(g^m r^n)^\lambda \pmod{n^2}] - 1}{n} \right\rfloor \cdot \mu \pmod{n}$$

$$= \left\lfloor \frac{[(g^{m\lambda})(r^{\phi(n^2)}) \pmod{n^2}] - 1}{n} \right\rfloor \cdot \mu \pmod{n}$$

By Euler's Theorem

$$= \left\lfloor \frac{[(g^{m\lambda})(1) \pmod{n^2}] - 1}{n} \right\rfloor \cdot \mu \pmod{n}$$

$$= \left\lfloor \frac{[(1+n)^{m\lambda} \pmod{n^2}] - 1}{n} \right\rfloor \cdot \mu \pmod{n}$$

By the Binomial Theorem

$$= \left\lfloor \frac{[1 + (m\lambda)n] - 1}{n} \right\rfloor \cdot \mu \pmod{n} = (m\lambda) \cdot \mu \pmod{n}$$

$$= m \pmod{n}$$

as desired.

NB: The Pailier cryptosystem assumes that even if an agent is given n, they cannot deduce the factorization $n = pq$. Namely, it assumes the difficulty of integer factorization (n is large), which is an assumption made by many cryptographic protocols, such as RSA.

Intuition

One of the most intriguing aspects of the protocol is that one does not need to pass the randomization element r for decryption. (This randomization r ensures the encryption output is probabilistic and distributed uniformly at random over possible outputs). Essentially, there is a lot of deep and rich number theory going on below the surface that makes this protocol work. See [38] for an exposition and formal proof of the correctness of the protocol.

Here, we will try to provide intuition for the most crucial aspect. Essentially, the Paillier cryptosystem primarily takes advantage of the fact that discrete logarithms can be computed easily. For example, consider solving for x in

$$y = (1+n)^x \pmod{n^2}.$$

Note that applying the Binomial theorem yields

$$y = (1+n)^x = 1 + nx + n^2 \left(\sum_{k=2}^{x} \binom{x}{k} n^{k-2} \right)$$

$$= 1 + nx \pmod{n^2}.$$

But then,

$$x = \frac{y-1}{n}$$

(division in natural numbers) is clearly a solution (note if a solution exists we must have $y = 1 \pmod{n}$). Note this observation coincides precisely what we are doing in the decryption step above (see verification above) since $g = n + 1$; essentially we are taking advantage of the fact that it is easy to compute the discrete log of $(1 + n)$ in modulo n^2.

Homomorphic Properties

As discussed earlier, the Paillier cryptosystem is homomorphic with respect to addition. Namely, we can verify

$$E(m_1) \cdot E(m_2) = E(m_1, r_1) \cdot E(m_2, r_2) = (g^{m_1} r_1^n)(g^{m_2} r_2^n)$$

$$= g^{m_1+m_2}(r_1 r_2)^n = E(m_1 + m_2, r_1 r_2) = E(m_1 + m_2).$$

Note these encryptions will decrypt correctly from the verification shown earlier. Therefore, we have

$$D(E(m_1) \cdot E(m_2)) = D(E(m_1 + m_2)) = m_1 + m_2,$$

and therefore this scheme is homomorphic with respect to addition as desired.

An Example

As this cryptosystem may appear a bit complicated, we proceed with an example. Say we conduct a key generation such that $p = 101, q = 103$, and therefore the Encryption and Decryption keys are

$$(n, g) = (10403, 10404), (\lambda, \mu) = (10200, 6867).$$

(Note p, q selected here are not large but are merely for illustrative purposes). Say 10 agents are trying to compute a sum (where each value contained in $[0, 99]$). They encrypt their value (see table below) and send to a Third Party (who is only given n; no knowledge of p or q).

m	r	$E(m,r)$
17	1613	30794166
15	1211	12370782
88	9331	75502784
68	1418	8823525
9	7291	10958390
55	7044	47240829
1	8642	84419799
3	8209	103142851
3	5286	8654806
3	5848	39953691

Using the homomorphic properties of the Paillier cryptosystem, the product of the ciphertexts is computed to be 279172 (mod n^2). This product is sent to the agents, who can then decrypt this value

$$D(c) = \left\lfloor \frac{[(279172)^{10200} \pmod{(10403)^2}] - 1}{10403} \right\rfloor \cdot 6867 \pmod{10403}$$

$$= 262 \pmod{10403},$$

which is indeed the desired sum.

One of the most salient aspects of this protocol is that every agent makes their own choice of r. Namely, as seen in the table above, even though agents had the same value m, the different r significantly impacts the encryption (a Third Party cannot back out whether two of the agents had the same value).

9.3.3 Secure Sum via Third Party

As seen in the above example, we can utilize the Pailier cryptosystem for a secure sum protocol (using a Third Party). Hence, we have the following protocol

Definition 9.4 (Secure Sum Protocol, adapted from [9])**.** *A set of finite agents $A = \{A_1, \cdots, A_k\}$ wish to compute a sum of respective values v_i, where $v_i \in \mathbb{N}_{\leq L}$ (for upper-bound $L \in \mathbb{Z}$) is the value held by agent A_i. Let T be a Trusted Third Party. We then have the following sum protocol:*

- *The agents in A select two large primes p, q such that $L << p, q$, and p, q are the same bit length. They then proceed with key generation as in the Paillier cryptosystem using p, q, producing the encryption key (n, g) and decryption key (λ, μ)*
- *Agent A_i selects random $r_i \in \mathbb{Z}_n$ and encrypts his/her value v_i:*

  $$E(v_i) = g^{v_i} r_i^n \pmod{n^2}.$$

  Agent A_i sends the pair $(E(v_i), n^2)$ to Trusted Third Party T

- *Third Party T computes the product*

$$E(\text{SUM}) = \prod_{i=1}^{k} E(v_i) \pmod{n^2}$$

and sends $E(\text{SUM})$ to each agent A_i

- *Each Agent A_i can decrypt*

$$D(E(\text{SUM})) = \left\lfloor \frac{[(E(\text{SUM}))^{\lambda} \pmod{n^2}] - 1}{n} \right\rfloor \cdot \mu \pmod{n}$$

to yield the desired sum.

The correctness of this protocol follows directly from the correctness of the valid encryption-decryption of and the additively homomorphic properties of the Pailier cryptosystem (both verified in the prior section). The sum is secure in that no agent learns additional information about the specific value held by another agent, and the Third Party T learns no information on the values (from the hardness of factoring n).

9.3.4 A Protocol for Pooling Information by Investors

Now that we have built to a secure sum protocol from the Pailier cryptosystem, we are ready to construct a unified protocol.

In this language $\mathcal{L}$, statements are of the form (using earlier notation)

$$V = \sum_i x_i.$$

We have the following protocol

Definition 9.5 (Zero-Knowledge Protocol for Pooling Information by Anonymous Investors)**.** *Let I be the (finite) set of investors in a fund, where $|I| = n$. Denote $x_i \in \mathbb{N}_{\leq 10^{12}}$ to be the amount invested in the fund by agent $i \in I$. Say $V \in \mathbb{N}_{\leq 10^{12}}$ is the sum (AUM) reported by the fund.*

The Honest Verifiers $V = \{V_1, \cdots, V_n\}$ (Investors) verify (confirm compliance of) a Prover P (Fund) in zero-knowledge, utilizing Trusted Third Party T as follows:

- *Prover P generates a Paillier cryptosystem by selecting primes p, q such that $10^{12} << p, q$, generating encryption key $EK = (n, g)$ and decryption key $DK = (\lambda, \mu)$.*

  He also selects a password w. Prover P sends the tuple (EK, DK, V, w) to each Verifier V_i and sends w to Third Party T

- *Each Verifier V_i encrypts v_i as $E(v_i)$ (see secure sum protocol above) using EK. Verifier V_i sends the tuple $(w_i = w, E(v_i), n^2)$ to Third Party T. (Note if a Verifier V_i received nothing from P, V_i rejects automatically)*

- *Third Party T computes the product of encrypted values for all correctly submitted passwords w_i i.e.*

$$E(\text{SUM}) = \prod_{i=1}^{k} E(v_i) \cdot I[w_i = w] \pmod{n^2}$$

where $I[w_i = w]$ is 0 or 1 depending on if $w_i = w$. Third Party T sends $E(\text{SUM})$ to each Verifier V_i who correctly submitted password w. Note if all agents who submit the correct w_i do not submit the same n^2, T sends nothing to anyone

- *Each Verifier V_i decrypts using DK (see secure sum protocol above) and confirms*

$$D(E(\text{SUM})) = V$$

and accepts/rejects accordingly (also reject automatically if V_i doesn't receive anything from T). Note the entire protocol rejects if at least one V_i rejects (i.e. reported to SEC for investigation or similar consequence)

The above protocol serves as a zero-knowledge proof protocol for the language $\mathcal{L}$ for honest verifiers and a prover. We verify the properties below.

Completeness

Take an $\ell \in \mathcal{L}$. Therefore, this indicates $V = \sum_i x_i$. From the correctness of the secure sum protocol (and Pailier cryptosystem) shown in the previous sections, an honest prover and honest verifiers will verify the sum with 100% probability. Hence, we have perfect completeness.

Soundness

Take an $\ell \in \mathcal{L}$. Therefore, this indicates $V \neq \sum_i x_i$. Hence, if a prover follows the protocol accurately, then clearly this will yield a rejection with high probability. It remains to consider the case of a cheating prover. Note we assume that the Prover P can only set up a single verification process with the Third Party T (otherwise such behavior would tip off T there is something sketchy going on) i.e. Prover P can't maliciously separate the investors into different buckets. Hence, there is only one valid password. Therefore, P must submit the correct password w to each verifier V_i. Analogously, P must submit the same n to each V_i (otherwise T receives different n^2). One can easily verify that this implies that each V_i must receive the same keys DK, EK. Lastly, Third Party T sends the same value to each Verifier V_i. But this then implies that each V_i must then receive the same V for all to accept (conditional on same EK, DK). Therefore, this implies the only way for all agents to accept if all V_i receive the same tuple (EK, DK, V, w). Hence, we have soundness as desired.

Zero-Knowledge

Zero-Knowledge here is straightforward. Clearly, if all the verifiers are honest, we can easily create a simulator to mimic the output distribution (i.e. since then all agents follows the protocol).

Therefore, we have indeed designed an (honest verifier) zero-knowledge protocol that serves as a solution to the Pooling Information by Anonymous Investors Problem. This protocol can be used to better balance financial transparency and confidentiality for fund compliance.

9.4 Potential Limitations

9.4.1 Honest Verifier

One limitation that can be levied against the protocol is the requirement that all the verifiers be honest. In general, we are unable to make broad statement on zero-knowledge in the midst of malicious verifiers. Namely, there may be malicious agents who instead of caring about verifying the sum, wish to attempt to determine the amount of holdings of other investors. For example, consider a situation in which all but one agent is malicious, such that each malicious agent reports 0. Then every malicious verifier will learn the amount of the truthful verifier (albeit not necessarily know who the value belongs to). In general, this is not too much of a concern, as we assume agents are generally unaware of how many truthful or malicious agents there are (or how other malicious agents behave), and hence the number they receive is likely meaningless, and hence dishonest verifiers will likely gain little, if any knowledge.

In addition, note that being a non-truthful verifier will almost surely guarantee some agent to reject and therefore raise an investigation to the SEC. Hence, in an investigation, a malicious verifier may be discovered to be tampering with the verification process and hence have severe consequences. Therefore, given these potentially severe repercussions, the issue of non-honest verifiers is of minimal concern.

9.4.2 Trusted Third Party

Another concern may be regarding our protocol's use of a Third Party. This is unavoidable as we require no direct communication between any two of the Verifiers V_i, V_j. The Third Party is essentially doing nothing more than confirming an identity token and computing a product of values; therefore any independent Third Party that has no affiliation to either the Prover or any of the Verifiers should be sufficient. In particular, a natural candidate would be the SEC or a similar regulatory agency, as they would be the ones to be flagged in the case of any funny business or suspicious behavior.

9.4.3 Collusion between Prover and Certain Verifiers

A significant concern may be the potential for the Prover and a particular "Verifier" to act in collusion. Namely, note that the Prover (the fund) is the one providing the password w to each Verifier V_i. As discussed above, if a Verifier were to not receive the correct password, this would be a red-flag. Hence, while every Verifier V_i must be included, nothing stops the Prover from creating "shadow" Verifiers. Namely, say the actual holdings form the set

$$\{x_1, \cdots, x_n\}.$$

The Prover could simulate an additional (clearly dishonest) verifier V^* that reports

$$x^* = V - \sum_i x_i,$$

and hence the protocol on the values

$$\{x_1, \cdots, x_n, x^*\}$$

will verify the sum V. Therefore, the Prover can effectively cheat the protocol in this way.

Regardless, this protocol constitutes an improvement (over having no such protocol) in that it provides a potential time-stamp of wrongdoing. In particular, in situations when a Ponzi scheme is discovered, it is often very difficult to pinpoint precisely how long the scheme has been going on and hence, difficult to administer appropriate sentences. Oftentimes, Ponzi schemes are in fact originally legitimate funds that in the short-term begin committing transgressions in an attempt to recoup or cover up losses. This makes it very difficult to ascertain where malicious intent began. However, with this protocol, the SEC and regulators have a time series of information (i.e. values sent to the Third Party), over which they can go back and analyze when the Ponzi scheme began (i.e. see if/when the fund created "shadow" verifiers). Therefore, despite the potential for cheating by the fund (Prover) and potential collusion, this protocol is useful from a regulatory and legal perspective.

9.4.4 Incentivizing Prover to Participate in Protocol

Lastly, another important consideration is whether this protocol is useful at all; i.e. what incentivizes a (truthful) Prover (fund) to participate in the protocol? Note the protocol requires the full participation of the fund in that the Prover identifies all the parties that participate in the protocol. In general, there are many ways this incentivization could occur. For one, if an investor provides pressure on the fund to do this verification protocol, the fund would be greatly pressed to do so, as refusing to do so might make the fund appear to be hiding something and therefore, potentially lose business from this investor.

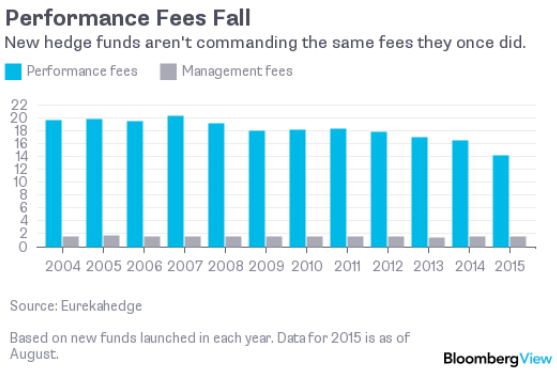


Figure 9.3: Depiction of average hedge fees over time, from [48]

In addition, on a pecuniary level, it may be in a truthful fund's interest to participate. Traditionally, average hedge fund fees have been the "2 and 20" fee model, charging a 2% management fee (on the total funds invested) and 20% performance fee (on the returns/profits generated). However, in recent years, these fees have fallen significantly, hovering around a 1.5% management fee and 15% performance fee (see image above).

While there are a number of factors explaining this trend, a significant determinant is the fallout from events such as the Madoff Ponzi scheme and related news. These events have induced investors to be more cautious of their capital, and hence funds have been forced to slash their margins. Now, if an honest fund participates in this protocol, this serve to signal lower risk (i.e. of being a Ponzi scheme or similar financial fraud) and hence remove the induced risk penalty (with respect to fees). Hence, these funds will likely therefore also be able to charge higher fees and hence get closer to returning to the historical "2 and 20" model. Therefore, this protocol is useful in that it can serve as a signaling mechanism for legitimate, high quality funds.

Chapter 10

Conclusion

The tradeoff between transparency and confidentiality is an important consideration in the context of financial regulation. In particular, as the preceding chapters make clear, zero-knowledge proof protocols can serve as a natural solution to navigating this tradeoff. Specifically we demonstrated the following applications of zero-knowledge to financial regulation:

- Reporting Portfolio Holdings to Employers: An employer can verify an employee has no financial holdings on a blacklist without revealing the other (allowed) holdings of the employee
- Risk Assurance for Financial Holders ([47]): A fund can convince its investors that its holdings subscribe to particular risk constraints (an interval), without disclosing the actual holdings
- Pooling Information by Anonymous Investors: A collection of investors of a fund can verify aggregate information (a sum) provided by the fund, while preserving pairwise anonymity.

The above applications serve to suggest the potential of zero-knowledge within the space of financial regulation. Namely, many current regulatory protocols are skewed, either enforcing too little (confidentiality) or too much regulation (transparency). Therefore, using modern cryptographic techniques, specifically in the form of zero-knowledge proofs as the above examples exemplify, we can revise and construct new regulatory protocols to better balance this tradeoff between confidentiality and transparency.

10.1 Next Steps and Future Directions

The protocols we developed in the preceding chapters were designed, for the most part, as efficient, tailor-made solutions. Therefore, they should be reasonably straightforward to implement. Regardless, the logical next step would be to implement the protocols as an experiment, to demonstrate and compute efficiency and run-time statistics as a proof of concept.

Contingent on solid results from the proof of concept, we could then attempt to bring variants on these cryptographic protocols to the implementation level and to market. Namely, we may seek to engage in partnerships with the SEC and other federal agencies or brokerage/information reporting firms. Such partnerships may significantly transform financial reporting and regulation in utilizing cryptographic methods and protocols, including zero-knowledge proofs.

Bibliography

[1] A. C. Yao, Protocols for secure computations (Extended Abstract), 23rd annual symposium on foundations of computer science (Chicago, Ill., 1982), 160164, IEEE, New York, 1982.

[2] Adida, Ben. Helios: Web-based open-audit voting. In Proc. USENIX Security, Aug. 2008

[3] Alexander Glaser, Boaz Barak, and Robert J. Goldston, A Zero-Knowledge Protocol for Nuclear Warhead Verification, Nature 510: (2014) 497502.

[4] Barak, Boaz. Zero Knowledge Proofs. Harvard CS 127. Cambridge, MA.

[5] Ben-Or, Michael, et al. "Everything provable is provable in zero-knowledge". In Goldwasser, S. Advances in Cryptology-CRYPTO '88. Lecture Notes in Computer Science. 403. Springer-Verlag. pp. 3756.

[6] Bernhard, David, and Bogdan Warinschi. Cryptographic Voting A Gentle Introduction. Foundations of Security Analysis and Design VII Lecture Notes in Computer Science, 2014, pp. 167211.

[7] Blum, Manuel, et al. Non-interactive zero-knowledge and its applications. In Proceedings of the twentieth annual ACM symposium on Theory of computing (STOC '88). ACM, New York, NY, USA, 103-112. 1988.

[8] Boudot, F. Efficient proofs that a committed number lies in an interval. In EUROCRYPT 2000, vol. 1807 of LNCS, pp. 431444. Springer Verlag, 2000.

[9] Choinyambuu, Sansar. Homomorphic Tallying with Paillier Cryptosystem. E-Voting Seminar. Zurich, Switzerland.

[10] Coloring graphs in Matgraph, www.mathworks.com/examples/matlab/community/20891-coloring-graphs-in-matgraph.

[11] Cryptographic hash function. Wikipedia, Wikimedia Foundation, 25 Feb. 2018, en.wikipedia.org/wiki/Cryptographic_hash_function.

[12] Damgrd, Ivan, et al. A generalization of Pailliers public-Key system with applications to electronic voting. International Journal of Information Security, vol. 9, no. 6, 2010, pp. 371385.

[13] Dodis, Yevgeniy. Zero Knowledge Proofs. Introduction to Cryptography. New York, NY.

[14] Dwork, Cynthia, et al. Concurrent zero-knowledge. J. ACM 51, 6 (November 2004), 851-898. 2004.

[15] Fiat, A. and Shamir, A. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In CRYPTO 1986, Springer-Verlag (LNCS 263) pages 186194, 1986

[16] Fiege, U., et al. Zero knowledge proofs of identity. Proceedings of the nineteenth annual ACM conference on Theory of computing - STOC 87, 1987, doi:10.1145/28395.28419.

[17] Flood, Mark D., et al. Cryptography and the Economics of Supervisory Information: Balancing Transparency and Confidentiality. SSRN Electronic Journal, 2013.

[18] Freedman, Michael J., et al. Efficient Private Matching and Set Intersection. Advances in Cryptology - EUROCRYPT 2004 Lecture Notes in Computer Science, 2004, pp. 119.

[19] Gentry, C. Fully homomorphic encryption using ideal lattices. In: STOC 2009, pp. 169178. ACM, New York, 2009.

[20] Ghadafi, E, et al. Practical zero-knowledge proofs for circuit evaluation. In Coding and Cryptography: IMACC 2009, Springer LNCS 5921, 469494, 2009.

[21] Goldreich, O., et al. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. J. ACM, 38(3):691729, 1991.

[22] Goldreich, Oded. Foundations of cryptography Volume I: basic tools. Cambridge University Press, 2003.

[23] Goldreich, Oded. Foundations of cryptography Volume II: basic applications. Cambridge University Press, 2004.

[24] Goldwasser, S., et al. The knowledge complexity of interactive proof-Systems. Proceedings of the seventeenth annual ACM symposium on Theory of computing - STOC 85, 1985, doi:10.1145/22145.22178.

[25] Graph isomorphism. Wikipedia, Wikimedia Foundation, 25 Feb. 2018, en.wikipedia.org/wiki/Graph_isomorphism.

[26] Groth, Jens. Non-Interactive Zero-Knowledge Arguments for Voting. Applied Cryptography and Network Security Lecture Notes in Computer Science, 2005, pp. 467482.

[27] Hurd, Joe. Blum Integers. Cambridge, 1997.

[28] Katz, Jonathan, and Yehuda Lindell. Introduction to modern cryptography. CRC Press, Taylor & Francis Group, 2015.

[29] Kemp, Scott, et al. Physical cryptographic verification of nuclear warheads. Proceedings of the National Academy of Sciences. Under review

[30] Kryptographie FAQ: Frage 107: What are Interactive Proofs and Zero-Knowledge Proofs?, altlasten.lutz.donnerhacke.de/mitarb/lutz/security/cryptfaq/q107.html.

[31] Labs, Cossack. Explain Like I'm 5: Zero Knowledge Proof (Halloween Edition). Hacker Noon, Hacker Noon, 26 Oct. 2017.

[32] L. Kissner and D. Song. Privacy-Preserving Set Operations. In CRYPTO 2005, Springer-Verlag (LNCS 3621), pages 241257, 2007.

[33] Maglich, Jordan. A Ponzi Pandemic: 500 Ponzi Schemes Totaling $50 Billion in 'Madoff Era'. Forbes, Forbes Magazine, 12 Feb. 2014, www.forbes.com/sites/jordanmaglich/2014/02/12/a-ponzi-pandemic-500-ponzi-schemes-totaling-50-billion-in-madoff-era.

[34] McCaughrean, Geraldine, and Rosamund Fowler. One Thousand and One Arabian Nights. Oxford University Press, 1999.

[35] McCormack, Conny B. Democracy Rebooted: The Future of Technology in Elections. Mar. 2016.

[36] Multiplicative vs Additive Homomorphic ElGamal. NVotes - Online Voting, 9 May 2017, nvotes.com/multiplicative-vs-additive-homomorphic-elgamal.

[37] Or, Ben, et al. Multi-prover interactive proofs: How to remove intractability assumptions. In Proceedings of the 20th Annual ACM Symposium on the Theory of Computing (Chicago, Ill., 1988). ACM, New York, pp. 113131.

[38] Paillier, Pascal. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. Advances in Cryptology EUROCRYPT 99 Lecture Notes in Computer Science, pp. 223238.

[39] Palmer, Ben, et al. A Protocol for Verification of an Auction Without Revealing Bid Values, Procedia Computer Science (1)1, pp. 26492658. 2010.

[40] Papadimitriou, Antonis, et al. DStress: Efficient Differentially Private Computations on Distributed Data. In EuroSys, 2017.

[41] Pass, Rafael. Computational Indistinguishability and Pseudorandomness. Cornell CS 6830. Ithaca, NY.

[42] Pedersen, Torben Pryds. A Threshold Cryptosystem without a Trusted Party. Advances in Cryptology EUROCRYPT 91 Lecture Notes in Computer Science, pp. 522526.

[43] Peikert, Chris. Proofs of Knowledge. Georgia Tech, Theoretical Foundations of Computer Science. Atlanta, GA.

[44] Priyadarsini, P. L. K. (2015). A Survey on some Applications of Graph Theory in Cryptography. Journal of Discrete Mathematical Sciences and Cryptography, 18(3), 209-217.

[45] Quisquater, Jean-Jacques, et al. How to Explain Zero-Knowledge Protocols to Your Children. Advances in Cryptology CRYPTO 89 Proceedings Lecture Notes in Computer Science, pp. 628631.

[46] Sipser, Michael. Introduction to the theory of computation. Cengage Learning, 2013.

[47] Szydlo, Michael. Risk Assurance for Hedge Funds Using Zero Knowledge Proofs. Financial Cryptography and Data Security Lecture Notes in Computer Science, 2005, pp. 156171.

[48] The Misguided Focus on Hedge Fund Fees. Sophisticated Investor, 12 Nov. 2015, sophisticatedinvestor.com/the-misguided-focus-on-hedge-fund-fees.

[49] Treaty on the Non-Proliferation of Nuclear Weapons. Wikipedia, Wikimedia Foundation, 25 Feb. 2018, en.wikipedia.org/wiki/Treaty_on_the_Non-Proliferation_of_Nuclear_Weapons.

[50] Understanding the Investment Chain. Institutional Investor, 12 Dec. 2012, www.institutionalinvestor.com/article/b14zpn27b87l6z/understanding-the-investment-chain.

[51] Varia, M, et al. HEtest - A Homomorphic Encryption Testing Framework, 3rd Workshop on Encrypted Computing and Applied Homomorphic Cryptography, Jan 2015.

[52] Wayner, Peter. Digital cash. AP Professional, 1997.

[53] Wikipedia contributors. Zero-knowledge proof. Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 18 Dec. 2017. Web. 24 Dec. 2017.

[54] ZSL: Zk-SNARKs for the Enterprise. Zcash - ZSL: Zk-SNARKs for the Enterprise. N.p., n.d. Web. 26 Dec. 2017.