Retraction: Theorem 5 in this thesis is not known to be true. The subtle difficulties in the proof, which were not understood by me at the time, were pointed out by [SV12], where a new four-round RZK protocol that is concurrently (and, therefore, also sequentially) sound is proposed. Because the mistake was discovered ten years after the thesis was published, I chose not to modify the thesis in order to present a historically accurate record and assist the reader in understanding [SV12].

References

[SV12] Alessandra Scafuro and Ivan Visconti. On round-optimal zero knowledge in the bare public-key model. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT*, volume 7237 of *Lecture Notes in Computer Science*, pages 153–171. Springer, 2012.

Zero-Knowledge with Public Keys

by

Leonid Reyzin

A.B., Harvard University (1996)
M.S., Massachusetts Institute of Technology (1999)

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

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2001

Author . . . .
Department of Electrical Engineering and Computer Science
May 21, 2001

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

Accepted by . . . .
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Zero-Knowledge with Public Keys
by
Leonid Reyzin

Submitted to the Department of Electrical Engineering and Computer Science
on May 21, 2001, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

In STOC 2000, Canetti, Goldreich, Goldwasser, and Micali put forward the strongest notion of zero-knowledge to date, *resettable zero-knowledge* (RZK) and implemented it in constant rounds in a new model, where the verifier simply has a public key registered before any interaction with the prover. This work explores their new *public-key model* for zero-knowledge protocols.

First, it shows that the soundness notion in this model has not been sufficiently understood and is, in fact, more subtle and complex than in the classical model. It identifies four *meaningful* notions of soundness, and proves that they are *distinct*. Thus, protocol designers should understand the needs of the application in order to avoid designing protocols whose soundness is too weak (thus resulting in insecure protocols) or too strong (thus resulting in protocols that are less efficient than necessary).

Second, having precisely defined the model, this work proceeds to demonstrate that stronger notions of soundness require more rounds to implement. Specifically, it provides upper and lower bounds on the numbers of rounds needed to implement the various soundness notions.

Finally, to achieve both ultimate round efficiency and strong soundness, this work puts forward a slightly stronger model. Informally, as long as the honest verifier does not use a given public key more than a fixed-polynomial number of times, there exist 3-round (provably optimal) RZK protocols for all of NP that possess strong soundness. This is particularly surprising, because such 3-round protocols provably do not exist in the public-key model without such an upper bound.

Thesis Supervisor: Silvio Micali
Title: Professor of Computer Science and Engineering

Acknowledgments

Receiving a doctorate is a wonderful occasion to thank the many people who have taught, supported, and inspired me. It is, of course, impossible to thank all of them, and I am sure I am inadvertently leaving someone out, for which I apologize in advance.

Silvio Micali has been an inspiring mentor and a caring advisor. I will consider my grad school career worthwhile if I managed to learn even half of what he tried to teach me. He spent many hours teaching me how to work on interesting problems, write eloquent papers, and deliver clear presentations. Perhaps even more of his time was spent advising me on life, career and the pursuit of happiness. The research in this thesis is a joint work with him. I am also grateful that he taught me how to make espresso.

I enjoyed discussing research ideas with Ron Rivest, who has kindly agreed to be a reader for this thesis. He has, on a number of occasions, listened to my incoherent ramblings on a research topic and then explained to me what I meant. I also appreciated being able to count on him for frank and unbiased advice. I am grateful for his suggestion, made five years ago, that I try working in cryptography.

I would like to thank Madhu Sudan for agreeing to be another reader for this thesis, and for providing valuable career advice. I very much enjoyed taking the advanced courses he taught at MIT and feasting on the delicious pizzas he served at his home.

I benefited greatly from the opportunity to learn from Shafi Goldwasser. I thank her for taking an active interest in my research, for teaching a fascinating class on advanced cryptography, and for possessing an acute sense of humor.

I've also had the good fortune of learning from many others before attending graduate school. I would like to thank my colleagues at RSA Labs—Burt Kaliski, Matt Robshaw, Ray Sidney and Yiqun Lisa Yin—for teaching me much about cryptography and about surviving graduate school at MIT. I am grateful to my undergraduate thesis advisor, Michael Rabin, for all the time he took to mentor me. I appreciate his confidence in me and all the kind advice he offered. I am also indebted to my high school teachers, Grisha Kondakov and Aleksey Belov, for skillfully guiding me through my first research project.

I didn't know my way around when I first arrived in the U.S. I thank Mark Saul, David Weinstein, Barbara Strudler and Jacob Bradler for pointing the way and then helping me get there.

Graduate school would not have been nearly as rewarding without my peers. I thank them all for the great time I had at MIT. In particular, Anna Lysyanskaya could always be relied upon for her vast knowledge and keen observations. More importantly, she has been a true friend and a vigilant advisor. I thank Yevgeniy Dodis and Salil Vadhan for enlightening conversations about research and otherwise. I enjoyed collaborating with Zulfikar Ramzan, Moses Liskov and Adam Smith. Thanks to Moses Liskov and Jonathan Herzog for the champagne.

The theory group would not be the same without Be Blackburn, also known (to all the graduate students) as Mom. She knows how to throw splendid parties, where to get large supplies of chocolate, and just what to put in the subject of your email to get a professor's response. She can find rooms, projectors, lost reimbursement checks, and theses from many years ago. Her cheerfulness and kindness have helped me tremendously.

I am most thankful to my parents, Marina and Natan Reyzin, for everything they have done for me and all of their worldly wisdom. They encouraged me and helped me keep things in perspective. They were also my first teachers of mathematics and computer science, and tremendously influenced my tastes in research. I am grateful to them and to my sister,

Alice, for the opportunities to discuss puzzles, problems, and my own research.

I thank my entire extended family for their unconditional pride in my accomplishments.

I will be forever grateful to my wife, Monica Perez, for all her support. For all the times I was unsure of myself, stressed from the workload, or doubtful of my plans, she provided a listening ear, a loving smile, and just the right words to improve my mood. She helped me greatly in so many ways, from editing my papers to making sure I got enough sleep. Most of all, I thank Monica for her love.

I thank my wonderful son, Jonathan, for giving me so much joy now and so much to look forward to.

Contents

Chapter 1

Introduction

Sections 1.1 and 1.2 provide some background information. The remaining sections of this chapter explain the new results obtained in this work. The new results were obtained jointly by Silvio Micali and myself. Most of them appear in [MR01a] and [MR01b].

1.1 The Bare Public-Key Model for Interactive Proofs

A novel protocol model, which I call the *bare public-key (BPK) model,* was introduced by Canetti, Goldreich, Goldwasser and Micali in the context of resettable zero-knowledge [CGGM00]. Although introduced with a specific application in mind, the BPK model applies to interactive proofs in general, regardless of their knowledge complexity. The model simply assumes that the verifier has a public key, PK, that is registered before any interaction with the prover begins. No special protocol needs to be run to publish PK, and no authority needs to check any property of PK. It suffices for PK to be a string known to the prover, and chosen by the verifier prior to any interaction with him.

The BPK model is very simple. In fact, it is a *weaker* version of the frequently used public-key infrastructure (PKI) model, which underlies any public-key cryptosystem or digital signature scheme. In the PKI case, a secure association between a key and its owner is crucial, while in the BPK case no such association is required. The single security requirement of the BPK model is that a bounded number of keys (chosen beforehand) are "attributable" to a given user. Indeed, having a prover $\mathcal{P}$ work with an incorrect public key for a verifier $\mathcal{V}$ does not affect soundness nor resettable zero-knowledgeness; at most, it may affect completeness. (Working with an incorrect key may only occur when an active adversary is present—in which case, strictly speaking, completeness does not even apply: this fragile property only holds when all are honest.)

Despite its apparent simplicity, the BPK model is quite powerful. In particular, it was introduced in order to dramatically improve the round-efficiency of resettable zero-knowledge (RZK) protocols: while RZK protocols exist in both the standard and the BPK models [CGGM00], only in the latter case can they be constant-round (even the weaker notion of concurrent zero knowledge (CZK) [DNS98] is not implementable in a constant number of rounds [CKPR01] [1]).

[1] To be precise, the lower bound of [CKPR01] rules out only the slightly stronger *black-box* constant-round RZK and CZK protocols (see Section 2.1 for the definition of "black-box"). While non-black-box constructions are not ruled out, it is very unlikely one can build such protocols without resorting to strong assumptions that are not complexity-theoretic in nature, such as those made in [HT98].

1.2 Resettable Zero-Knowledge

A zero-knowledge (ZK) proof [GMR85, GMR89], is a proof that conveys nothing but the verity of a given statement. As put forward by Canetti, Goldreich, Goldwasser, and Micali [CGGM00], *resettable zero-knowledge* (RZK) is the strongest form of zero-knowledge known to date. In essence, an RZK proof is a proof that remains ZK even if a polynomial-time verifier can force the prover to execute the proof multiple times with the same coin tosses. More specifically,

- *The verifier can reset the prover.* In each execution, the verifier can choose whether the prover should execute the protocol with a new random tape or with one of the tapes previously used.

- *The verifier can arbitrarily interleave executions.* The verifier can always start (in particular, in a recursive way) new executions in the middle of old ones, and resume the old ones whenever it wants.

- *The prover is oblivious.* As far as it is concerned, the prover is always executing a single instance of the protocol.

Resettable ZK is a strengthening of Dwork, Naor and Sahai's [DNS98] notion of concurrent ZK (CZK). In essence, in a CZK protocol, a malicious verifier acts as in an RZK protocol, except that it lacks the power of resetting the prover's random tapes.

Perhaps surprisingly, it is possible to implement such a strong notion without public keys: RZK protocols for NP-complete languages are constructed in [CGGM00] under specific complexity assumptions. Their construction is obtained by properly modifying the CZK protocol of Richardson and Kilian [RK99]. Because this underlying CZK protocol is not constant-round, neither is the resulting RZK protocol. (The minimum known number of rounds for implementing the protocol of [RK99] is polylogarithmic in the security parameter, as shown by Kilian and Petrank [KP01].) In fact, as already pointed out, obtaining a constant-round RZK protocol in the standard model may not be possible due to the lower bound of Canetti, Kilian, Petrank and Rosen [CKPR01] (although, to the best of my knowledge, the [CKPR01] result was not yet known when [CGGM00] was written).

In the BPK model, however, a constant-round protocol is possible. Specifically, a 5-round RZK protocol for any NP language is presented in [CGGM00][2].

1.3 The Notions of Soundness in the Bare Public-Key Model

Despite its simple mechanics, the soundness property of the bare public-key model was not fully understood when the model was first proposed. Indeed, it is more complex than the soundness notion in the classical case.

In the standard model for interactive proofs, soundness can be defined quite easily: essentially, there should be no efficient malicious prover $\mathcal{P}^*$ that can convince $\mathcal{V}$ of the verity of a false statement with non-negligible probability. This simple definition suffices regardless of whether $\mathcal{P}^*$ interacts with the verifier only once, or several times in a sequential

[2] Actually, [CGGM00] presents two related constructions: (1) a 4-round protocol with an additional 3-round preprocessing stage with a trusted third party, and (2) an 8-round protocol without such preprocessing. Their constructions can be easily modified to yield the 5-round protocol attributed above.

manner, or several times in a concurrent manner. The reason for this sufficiency is that, in the standard model, $\mathcal{V}$ is polynomial-time and has no "secrets" (i.e., all of its inputs are known to $\mathcal{P}^*$). Thus, if there were a $\mathcal{P}^*$ successful "against a multiplicity of verifiers," then there would also be a malicious prover successful against a single verifier $\mathcal{V}$: it would simply let $\mathcal{P}^*$ interact with $\mathcal{V}$ while "simulating all other verifiers."

In the BPK model, however, $\mathcal{V}$ has a secret key SK, corresponding to its public key PK. Thus, $\mathcal{P}^*$ could potentially gain some knowledge about SK from an interaction with $\mathcal{V}$, and this gained knowledge might help $\mathcal{P}^*$ to convince $\mathcal{V}$ of a false theorem in a subsequent interaction. Therefore,

> *in the BPK model, the soundness property may be affected by the type of interaction a malicious prover is entitled to have with the verifier, as well as the sheer number of these interactions.*

In addition, other totally new issues arise in the BPK model. For example, should $\mathcal{P}^*$ be allowed to determine the exact false statement of which it tries to convince $\mathcal{V}$ before or after it sees PK? Should $\mathcal{P}^*$ be allowed to change that statement after a few interactions with $\mathcal{V}$?

In sum, a better understanding of the soundness properties of the BPK model is necessary to avoid designing protocols that are unsound (and thus insecure) or "too sound" (and thus, potentially, less efficient than otherwise possible).

Four Notions of Soundness in the Bare Public-Key Model

Having identified the above issues, I formalize four *meaningful* notions of soundness in the BPK model. (These notions correspond in spirit to the commonly used notions of zero-knowledge in the standard model. That is, the ways in which a malicious prover is allowed to interact with the honest verifier correspond to those in which a malicious verifier is allowed to interact with the honest prover in various notions of zero-knowledgeness.) Roughly speaking, here are the four notions, each of which implies the previous one:

1. **one-time soundness,** when $\mathcal{P}^*$ is allowed a single interaction with $\mathcal{V}$ per theorem statement;

2. **sequential soundness**, when $\mathcal{P}^*$ is allowed multiple but sequential interactions with $\mathcal{V}$;

3. **concurrent soundness,** when $\mathcal{P}^*$ is allowed multiple interleaved interactions with the same $\mathcal{V}$; and

4. **resettable soundness,** when $\mathcal{P}^*$ is allowed to reset $\mathcal{V}$ with the same random tape *and* interact with it concurrently.

All four notions are meaningful. Sequential soundness (implicitly used in [CGGM00]) is certainly a very natural notion, and concurrent and resettable soundness are natural extensions of it. As for one-time soundness, it is also quite meaningful when it is possible to enforce that a prover who fails to convince the verifier of the verity of a given statement S does not get a second chance at proving S. (E.g., the verifier may memorize the theorem statements for which the prover failed, or make suitable use of timestamps assuming some notion of time is available to both prover and verifier.)

These four notions of soundness apply both to interactive proofs (where a malicious prover may have unlimited power [GMR89]) and argument systems (where a malicious prover is restricted to polynomial time [BCC88]).

Separating the Four Notions

I prove that the above four notions are not only meaningful, but also *distinct.* Though conceptually important, these separations are technically simple. They entail exhibiting three protocols, each satisfying one notion but not the next one. Specifically, in Section 2.3 I prove the following theorems.

Theorem 1 *If one-way functions exist, there is a compiler-type algorithm that, for any language L, and any interactive argument system for L satisfying one-time soundness, produces another interactive argument system for the same language L that satisfies one-time soundness but not sequential soundness.*

Theorem 2 *If one-way functions exist, there is a compiler-type algorithm that, for any language L, and any argument system for L satisfying sequential soundness, produces another argument system for the same language L that satisfies sequential soundness but not concurrent soundness.*

Theorem 3 *There exists a compiler-type algorithm that, for any language L, and any interactive proof (or argument) system for L satisfying concurrent soundness, produces another interactive proof (respectively, argument) system for the same language L that satisfies concurrent soundness but not resettable soundness.*

Note that the above separation theorems hold with complexity assumptions that are indeed minimal: the third theorem holds *unconditionally;* while the first and second rely only on the existence of one-way functions. (This is why Theorems 1 and 2 only hold for bounded provers.)

Realizing that there exist separate notions of soundness in the BPK model is crucial to avoid errors. By relying on a single, undifferentiated, and intuitive notion of soundness, one might design a BPK protocol sound in settings where malicious provers are limited in their interactions, while someone else might erroneously use it in settings where malicious provers have greater powers.

1.4 Upper Bounds on Round Complexity

Given the four distinct notions of soundness, it reasonable to ask how many rounds are sufficient to achieve zero-knowledge (or, better yet, resettable zero-knowledge) protocols for each of the soundness notions in the BPK model. For one-time soundness, I prove the following theorem in Section 3.1.

Theorem 4 *Assuming the security of RSA with large prime exponents against adversaries that are subexponentially strong, for any $L \in$ NP, there exists a 3-round black-box RZK protocol in the BPK model that possesses one-time, but not sequential, soundness.*

For sequential soundness, [CGGM00] already provided a 5-round protocol for RZK in the BPK model (although the four notions of soundness were not distinguished at the time

[CGGM00] was written, what is informally proven in [CGGM00] is essentially sequential soundness; in Section 3.2 I provide evidence for why the CGGM protocol is not concurrently sound). I reduce the number of rounds by one with the following theorem, proven in Section 3.2.

Theorem 5 *Assuming there exist certified trapdoor permutation families*[3] *secure against subexponentially-strong adversaries, for any $L \in \mathrm{NP}$, there exists a 4-round black-box RZK protocol in the BPK model that possesses sequential soundness.*

I provide no upper bounds for concurrent or resettable soundness. In the latter case, this is for a good reason: resettable soundness and black-box ZK cannot be simultaneously achieved in the BPK model (see the next section). In the former case, concurrent soundness does not seem inherently unachievable, and finding a concurrently sound protocol in the BPK model is an open problem.

1.5 Lower Bounds on Round Complexity

Similarly to upper bounds, it is reasonable to ask for lower bounds for achieving resettable zero-knowledge (or, better yet, achieving even just zero-knowledge) for different notions of soundness in the BPK model.

For any soundness notion, I prove the following lower bound:

Theorem 6 *Any (resettable or not) auxiliary-input*[4] *ZK protocol (satisfying one-time or stronger soundness) in the BPK model for a language outside of BPP requires at least three rounds.*

Note that the above lower bound matches the known upper bound only for the one-time soundness case. It remains open whether 3-round sequentially-sound RZK protocols exist.

For concurrent soundness, I prove a stronger lower bound for the case of black-box zero-knowledgeness, using the techniques of Goldreich and Krawczyk [GK96]:

Theorem 7 *Any (resettable or not) black-box ZK protocol satisfying concurrent soundness in the BPK model for a language outside of BPP requires at least four rounds.*

To reiterate, I do not know of any concurrently sound RZK protocol, let alone a 4-round one. A non-resettable ZK protocol is known in the standard model [FS89] (such a protocol is, of course, automatically concurrently sound in the BPK model, with null public and secret keys).

Finally, for resettable soundness, I prove the ultimate lower bound (but, again, only for the case of *black-box* zero-knowledge):

Theorem 8 *There is no (resettable or not) black-box ZK protocol satisfying resettable soundness in the BPK model for a language outside of BPP.*

[3] *A trapdoor permutation family is* certified *if it is easy to verify that a given function belongs to the family.*

[4] *Auxiliary-input ZK is a stronger notion than ordinary ZK [GMR85], but is weaker than black-box ZK. All known ZK protocols satisfy the auxiliary-input ZK property. See Section 2.1 for the precise definition.*

1.6 Beating the Lower Bounds with a New Model

The above lower bounds preclude the existence of a 3-round concurrently sound black-box RZK protocol. In addition, they demonstrate the difficulty of designing such a protocol even if one settles for only sequential soundness: one has to design a protocol that is sequentially, but provably not concurrently, sound.

Because rounds of communication are an expensive resource in an interactive protocol, it is highly desirable to reduce them as much as possible. Moreover, concurrent soundness is highly desirable when zero-knowledge protocols are used for identification of users (provers) to a large system (verifier), when the verifier needs to be able to handle multiple provers at once. A minimal-round concurrently sound protocol is thus highly desirable.

Unfortunately, as pointed out above, no such protocol is currently known in the BPK model. However, I show that by strengthening the model only slightly, concurrent soundness is not only achievable, it is achievable in just three rounds—something provably impossible in the BPK model!

The Upperbounded Public-Key Model

How many public keys can a verifier establish before it interacts with the prover? Clearly, no more than a polynomial (in the security parameter) number of keys. Though innocent-looking, this bound is a source of great power for the BPK model: it allows for the existence of constant-round black-box RZK, which is impossible in the standard model.

How many times can a verifier use a given public key? Of course, at most a polynomial (in the security parameter) number of times. Perhaps surprisingly, I show that if such an innocent-looking polynomial upperbound U is made explicit *a priori*, then one can further increase the round efficiency of RZK protocols.

In the new *upperbounded public-key* (UPK) model, the honest verifier is allowed to fix a polynomial upperbound, U, on the number of times a public key will be used; keep track, via a counter, of how many times the key has been used; and refuse to participate once the counter has reached the upperbound.

Let me now make the following remarks about the UPK model:

- In the RZK setting, the "strong party" is the verifier (who controls quite dramatically the prover's executions). Such a strong party, therefore, should have no problems in keeping a counter in order to save precious rounds of interaction.

- The UPK model does not assume that the prover knows the current value of the verifier's counter. (Guaranteeing the accuracy of such knowledge would de facto require public keys that "change over time.")

- While the specific protocol in the UPK model presented in Chapter 5 satisfies interesting efficiency constraints with respect to U, these should be considered properties of the protocol rather than requirements of the UPK model.

  (For instance, in the protocol of Chapter 5, the public key length is independent of U, while the secret key length and each execution of the protocol depend on U only logarithmically. Only the verifier's key-generation phase depends linearly on U.)

- The UPK model is somewhat similar to the one originally envisaged in [GMR88] for secure digital signatures, where the signer posts an explicit upperbound on the number of signatures it will produce relative to a given public key, and keeps track of the number of signatures produced so far. (The purpose and use of the upperbound in the UPK model, however, are totally different.)

- While sufficient for constant-round implementation of the stronger RZK notion, the UPK model is perhaps simpler than those of [DS98] (using "timing assumptions") and [Dam00] (using trusted third parties to choose some system parameters), which were proposed for efficient implementations of CZK.

3-Round RZK in the UPK Model

Theorem 9 *In the UPK model there exist 3-round concurrently sound black-box RZK arguments for any language in NP, assuming collision-resistant hashing and the subexponential hardness of discrete logarithm and integer factorization.*[5]

Note that this result is round-optimal in the UPK model. Even one-time soundness with non-resettable zero-knowledge in the UPK model cannot be achieved in two rounds, by exactly the same argument as the one used for Theorem 6.

[5] *Alternative complexity assumptions are given in Chapter 5.*

Chapter 2

Definitions

2.1 Resettable ZK in the Bare Public-Key Model

To define an RZK protocol, one needs to define the usual three properties: completeness, soundness and resettable zero-knowledgeness.

The Honest Parties and Completeness

Let

— A *public file* F be a polynomial-size collection of records (id, PK_{id}), where id is a string identifying a verifier, and PK_{id} is its (alleged) public key.

— An *(honest) prover* $\mathcal{P}$ *(for a language* L*)* be an interactive deterministic polynomial-time TM that is given as inputs (1) a security parameter 1^n, (2) a n-bit string $x \in L$, (3) an auxiliary input y, (4) a public file F, (5) a verifier identity id, and (6) a random tape ω.

For the purposes of defining RZK, one can view $\mathcal{P}$ as a *non-interactive* TM that is given, as an additional input, the entire history of the messages already received in the interaction, and outputs the next message to be sent. Fixing all inputs, this view allows one to think of $\mathcal{P}(1^n, x, y, F, id, \omega)$ as a simple deterministic oracle that outputs the next message given the history of the interaction.

— An *(honest) verifier* $\mathcal{V}$ be an interactive deterministic polynomial-time TM that works in two stages. In stage one (the *key-generation* stage), on input a security parameter 1^n and random tape r, $\mathcal{V}$ outputs a public key PK and the corresponding secret key SK. In stage two (the *verification* stage), on input SK, an n-bit string x and a random string ρ, $\mathcal{V}$ performs an interactive protocol with a prover, and outputs "accept x" or "reject x."

Fixing SK and ρ, one can view the verification stage $\mathcal{V}(SK, \rho)$ as a deterministic oracle that is given x and the entire history of the messages already received in the interaction, and outputs the next message to be sent, or "accept x"/"reject x." This view is helpful in defining the notion of resettable soundness below (however, I will use the interactive view of $\mathcal{V}$ in defining one-time, sequential and concurrent soundness).

Completeness for a pair $(\mathcal{P}, \mathcal{V})$ is defined the usual way. Consider the following procedure for $(\mathcal{P}, \mathcal{V})$, a string $x \in L$ of length n and a string y.

Procedure Normal-Interaction

1. Run the key-generation stage of $\mathcal{V}$ on input 1^n and a random string r to obtain PK, SK.
2. Pick any id, and let F be a public file that contains the record (id, PK).
3. Pick strings ω and ρ at random and run $\mathcal{P}$ on inputs $1^n, x, y, id, \omega$, and the verification stage of $\mathcal{V}$ on inputs SK, x, ρ, letting them interact.

Definition 1 A pair $(\mathcal{P}, \mathcal{V})$ is *complete* for an NP-language L if for all n-bit strings $x \in L$ and their NP-witnesses y, the probability that in an execution of Normal-Interaction $\mathcal{V}$ outputs "accept" differs from 1 by a quantity negligible in n.

The Various Cheating Provers and Soundness

Now I formally define the four new soundness notions in the BPK model, each stating that a malicious prover should be unable to get the verifier to accept a false statement. Note that it is possible to formalize the four notions of soundness by insisting that the verifier give zero knowledge to the (one-time, sequential, concurrent or resetting) malicious prover. This would highlight the correspondence of our soundness notions to the notions of zero-knowledge, and would be simpler to define, because the definitions of zero-knowledge are already well established. However, such an approach is an overkill, and would result in unnecessarily restrictive notions of soundness in the BPK model: we do not care if the prover gains knowledge so long as the knowledge does not allow the prover to cheat.

The definitions below are for interactive arguments [BCC88] rather than interactive proofs [GMR85]. That is, a malicious prover is limited to polynomial time, and soundness is computational: the definitions do not preclude cheating by computationally unbounded provers. All the currently known examples of protocols in any public-key model (including the ones presented in this work) are arguments anyway, because they enable a malicious prover to cheat if it can recover the secret key SK from the public key PK. In principle, however, a public-key model does not seem to rule out interactive proofs: it seems possible to make clever use of a verifier public key that has no secrets associated with it.

The definitions below can be straightforwardly modified for proofs. Because proofs are strictly stronger than arguments, all the lower bounds proved in this work extend to proofs as well.

I now need to define three types of malicious provers. Let

— A *s-sequential malicious prover* $\mathcal{P}^*$ for a positive polynomial s be a probabilistic polynomial-time TM that, on first input 1^n, runs in at most $s(n)$ stages, so that

1. In stage 1, $\mathcal{P}^*$ receives a public key PK and outputs a string x_1 of length n.
2. In every even stage, $\mathcal{P}^*$ starts in the final configuration of the previous stage and performs a single interactive protocol: it outputs outgoing messages and receives incoming messages (the machine with which it performs the interactive protocol will be specified below, in the definition of sequential soundness). It can choose to abort an even stage at any point and move on to the next stage by outputting a special message.
3. In every odd stage $i > 1$, P^* starts in the final configuration of the previous stage and outputs a string x_i of length n.

— An *s-concurrent malicious prover* $\mathcal{P}^*$, for a positive polynomial s, be a probabilistic polynomial-time TM that, on inputs 1^n and PK, performs at most $s(n)$ interactive protocols as follows:

1. If $\mathcal{P}^*$ is already running $i-1$ interactive protocols $1 \leq i-1 < s(n)$, it can output a special message "Start x_i," where x_i is a string of length n.
2. At any point it can output a message for any of its (at most $s(n)$) interactive protocols (the protocol is unambiguously identified in the outgoing message). It then immediately receives the party's response and continues.

— An *s-resetting malicious prover* $\mathcal{P}^*$, for a positive polynomial s, be a probabilistic polynomial-time TM that, on inputs 1^n and PK, gets access to $s(n)$ oracles for the verifier (to be precisely specified below, in the definition of resettable soundness).

A pair $(\mathcal{P}, \mathcal{V})$ can satisfy one or more of the four different notions of soundness defined below. Note that each subsequent notion trivially implies the previous one.

For the purposes of defining one-time and sequential soundness, consider the following procedure for a given s-sequential malicious prover $\mathcal{P}^*$, a verifier $\mathcal{V}$ and a security parameter n.

Procedure Sequential-Attack

```
1. Run the key-generation stage of $\mathcal{V}$ on input $1^n$ and a random string $r$ to obtain
   $PK, SK$.
2. Run the first stage of $\mathcal{P}^*$ on inputs $1^n$ and $PK$ to obtain an $n$-bit string $x_1$.
3. For $i$ ranging from 1 to $s(n)/2$:
   3.1 Select a random string $\rho_i$.
   3.2 Run the $2i$-th stage of $\mathcal{P}^*$, letting it interact with the verification stage of $\mathcal{V}$ with
       input $SK, x_i, \rho_i$.
   3.3 Run the $(2i+1)$-th stage of $\mathcal{P}^*$ to obtain an $n$-bit string $x_i$.
```

Definition 2 $(\mathcal{P}, \mathcal{V})$ satisfies *one-time soundness for a language L* if for all positive polynomials s, for all s-sequential malicious provers $\mathcal{P}^*$, the probability that in an execution of Sequential-Attack, there exists i such that $1 \leq i \leq s(n)$, $x_i \notin L$, $x_j \neq x_i$ for all $j < i$ and $\mathcal{V}$ outputs "accept x_i" is negligible in n.

Sequential soundness differs from one-time soundness only in that the malicious prover is allowed to have $x_i = x_j$ for $i < j$.

Definition 3 $(\mathcal{P}, \mathcal{V})$ satisfies *sequential soundness for a language L* if for all positive polynomials s, for all s-sequential malicious provers $\mathcal{P}^*$, the probability that in an execution of Sequential-Attack, there exists i such that $1 \leq i \leq s(n)$, $x_i \notin L$, and $\mathcal{V}$ outputs "accept x_i" is negligible in n.

For the purposes of defining concurrent soundness, consider the following procedure for a given s-concurrent malicious prover $\mathcal{P}^*$, a verifier $\mathcal{V}$ and a security parameter n.

Procedure Concurrent-Attack

```
1. Run the key-generation stage of $\mathcal{V}$ on input $1^n$ and a random string $r$ to obtain
   $PK, SK$.
2. Run $\mathcal{P}^*$ on inputs $1^n$ and $PK$.
3. Whenever $\mathcal{P}^*$ outputs "Start $x_i$," select a fresh random string $\rho_i$ and let the $i$-th
   machine with which $\mathcal{P}^*$ interacts be the verification stage of $\mathcal{V}$ on inputs $SK, x_i, \rho_i$.
```

Definition 4 $(\mathcal{P}, \mathcal{V})$ satisfies *concurrent soundness for a language* L if for all positive polynomials s, for all s-concurrent malicious provers $\mathcal{P}^*$, the probability that in an execution of Concurrent-Attack, $\mathcal{V}$ ever outputs "accept x" for $x \notin L$ is negligible in n.

Finally, for the purposes of defining resettable soundness, consider the following procedure for a given s-resetting malicious prover $\mathcal{P}^*$, a verifier $\mathcal{V}$ and a security parameter n.

Procedure Resetting-Attack

1. Run the key-generation stage of $\mathcal{V}$ on input 1^n and a random string r to obtain PK, SK.
2. Run $\mathcal{P}^*$ on inputs 1^n and PK.
3. Generate $s(n)$ random strings ρ_i for $1 \le i \le s(n)$.
4. Let $\mathcal{P}^*$ interact with oracles for the second stage of the verifier, the i-th oracle having input SK, ρ_i.

Note that concurrent soundness and resettable soundness differ in one crucial aspect: for the former, every instance of $\mathcal{V}$ is an interactive TM that keeps state between rounds of communication, and thus cannot be rewound; whereas for the latter, every instance of $\mathcal{V}$ is just an oracle, and thus can effectively be rewound.

Definition 5 $(\mathcal{P}, \mathcal{V})$ satisfies *resettable soundness for a language* L if for all positive polynomials s, for all s-resetting malicious provers $\mathcal{P}^*$, the probability that in an execution of Resetting-Attack, $\mathcal{P}^*$ ever receives "accept x" for $x \notin L$ from any of the oracles is negligible in n.

The Malicious Verifier and Resettable Zero-Knowledgeness

The notion of RZK was first introduced in [CGGM00]. The reader is referred to that paper for intuition and discussion of the notion. Here, I just present the definitions.

I need to define the malicious verifier and the simulators. Let

— An (s,t)*-resetting malicious verifier* $\mathcal{V}^*$, for any two positive polynomials s and t, be a TM that runs in two stages so that, on first input 1^n,

1. In stage 1, $\mathcal{V}^*$ receives $s(n)$ values $x_1, \ldots, x_{s(n)} \in L$ of length n each, and outputs an arbitrary public file F and a list of $s(n)$ identities $id_1, \ldots, id_{s(n)}$.
2. In stage 2, $\mathcal{V}^*$ starts in the final configuration of stage 1, is given oracle access to $s(n)^3$ provers, and then outputs its "view" of the interactions: its random string and the messages received from the provers.
3. The total number of steps of $\mathcal{V}^*$ in both stages is at most $t(n)$.

Following [GO94], such a verifier is *auxiliary-input* if it has one additional input, $z \in \{0,1\}^*$, of arbitrary length (note that, because it is limited to $t(n)$ steps, it may be unable to read all of z during its execution).

— A *(non-black-box) simulator* $M_{\mathcal{V}^*}$ for an (s,t)-resetting malicious verifier $\mathcal{V}^*$ be a machine that is given $s(n)$ values $x_1, \ldots, x_{s(n)} \in L$ of length n each as input, and runs in time polynomial in n. If $\mathcal{V}^*$ is auxiliary-input, then $M_{\mathcal{V}^*}$ is also given and additional input, $z \in \{0,1\}^*$.

— A *black-box simulator* M be a polynomial-time machine that is given oracle access to an (s,t)-resetting $\mathcal{V}^*$. By this I mean that it can run $\mathcal{V}^*$ multiple times, each time picking $\mathcal{V}^*$'s inputs, random tape and (because $\mathcal{V}^*$ makes oracle queries itself) the answers to all of $\mathcal{V}^*$'s queries. M is also given $s(n)$ values $x_1, \ldots, x_{s(n)} \in L$ as input.

Now I formally define three variations on the resettable-zero-knowledgeness property.

Definition 6 $(\mathcal{P}, \mathcal{V})$ is *resettable zero-knowledge for an NP-language* L if for every pair of positive polynomials (s,t), for every (s,t)-resetting verifier $\mathcal{V}^*$, there exists a simulator $M_{\mathcal{V}^*}$ such that for every $x_1, \ldots, x_{s(n)} \in L$ and their corresponding NP-witnesses $y_1, \ldots, y_{s(n)}$, the following probability distributions are indistinguishable (in time polynomial in n):

1. The output of $\mathcal{V}^*$ obtained from the experiment of choosing $\omega_1, \ldots, \omega_{s(n)}$ uniformly at random, running the first stage of $\mathcal{V}^*$ to obtain F, and then letting $\mathcal{V}^*$ interact in its second stage with the following $s(n)^3$ instances of $\mathcal{P}$: $\mathcal{P}(x_i, y_i, F, id_k, \omega_j)$ for $1 \leq i, j, k \leq s(n)$.

2. The output of M_{V^*} with input $x_1, \ldots, x_{s(n)}$.

$(\mathcal{P}, \mathcal{V})$ is, additionally, *auxiliary-input resettable zero-knowledge for an NP-language* L, if the above holds for every auxiliary-input $\mathcal{V}^*$ and for every string z that is given as input to both $\mathcal{V}^*$ and M_{V^*}.

Finally, $(\mathcal{P}, \mathcal{V})$ is *black-box resettable zero-knowledge for an NP-language* L, if the above holds for a single black-box simulator M that is simply allowed to interact with $\mathcal{V}^*$.

Note that Goldreich and Oren prove in [GO94] that any black-box ZK protocol is also auxiliary-input ZK. The same holds for the definition of RZK, above. Thus, the three notions are in order of non-decreasing strength.

Also note that all known RZK protocols (including the ones in this paper) are black-box and, therefore, auxiliary input. Indeed, it is difficult to imagine how one could prove zero-knowledgeness without using a black-box simulator (the one exception to this is a non-resettable ZK protocol of Hada and Tanaka [HT98], which explicitly makes a non-black-box assumption in order to construct a non-black-box simulator).

The lower bounds proven in this work will hold for auxiliary-input RZK or black-box RZK (in fact, they will hold for non-resettable ZK as well). All the upper bounds in this paper are black-box RZK.

2.2 Resettable ZK in the Upperbounded Public-Key Model

To define the UPK model, I need to make a few changes to the above definitions. First of all, I change the verifier. Let

— A *U-bounded (honest) verifier* $\mathcal{V}$, for a positive polynomial U, be an interactive polynomial-time TM that runs in two stages. In stage one (the *key-generation* stage), on input a security parameter 1^n and random tape r, $\mathcal{V}$ outputs a public key PK and the corresponding secret key SK. In stage two (the *verification* stage), on input SK, and n-bit string x, a counter value c and a random string ρ, $\mathcal{V}$ performs an interactive protocol with a prover, and outputs "accept x," "reject x," or "counter too large."

The definition of completeness changes as follows: it has to hold only if $\mathcal{V}$ is given a value c as input and $c < U(n)$.

The definitions of one-time and sequential soundness require the following changes to the sequential malicious prover and the procedure Sequential-Attack: every odd stage of the malicious prover outputs not only the common input x_i, but also a value c_i to be input to $\mathcal{V}$. The c_i values have to be in strictly increasing order. The definition of concurrent soundness changes similarly: whenever a concurrent malicious prover outputs "start x_i," it also has to output c_i, which is input to $\mathcal{V}$ in the procedure Concurrent-Attack. The c_i values have to be in strictly increasing order.

I do not define resettable soundness in the UPK model, because a resetting prover should be able to reset the counter value as well, and therefore the counter becomes meaningless.

The definitions of resettable zero-knowledgeness do not change.

2.3 Separating the Four Notions of Soundness in the BPK Model

In this section, I prove Theorems 1, 2 and 3: namely, that the four soundness notions are distinct (the first two separations are conditioned on the existence of one-way functions).

The Common Idea

Given a protocol $(\mathcal{P}, \mathcal{V})$ that satisfies the i-th soundness notion (for $i = 1, 2$, or 3), I deliberately weaken the verifier to come up with a protocol $(\mathcal{P}', \mathcal{V}')$ that does not satisfy the $(i+1)$-th soundness notion, but still satisfies the i-th. In each case, I add rounds at the beginning of $(\mathcal{P}, \mathcal{V})$ (and sometimes information to the keys) that have nothing to do with the language or the theorem being proven. At the end of these rounds, either $\mathcal{V}'$ accepts, or $(\mathcal{P}', \mathcal{V}')$ proceed with the protocol $(\mathcal{P}, \mathcal{V})$. In each case, it will be easy for a malicious prover for the $(i + 1)$-th notion of soundness to get $\mathcal{V}'$ to accept at the end of these additional rounds.

To prove that the resulting protocol $(\mathcal{P}', \mathcal{V}')$ still satisfies the i-th notion of soundness, it will suffice to show that if a malicious prover $\mathcal{P}'^*$ for $(\mathcal{P}', \mathcal{V}')$ exists, then it can be used to construct a malicious prover $\mathcal{P}^*$ for $(\mathcal{P}, \mathcal{V})$. In each case, this is easily done: $\mathcal{P}^*$ simply simulates the additional rounds to $\mathcal{P}'^*$ (one also has to argue that $\mathcal{V}'$ interacting with $\mathcal{P}'^*$ is unlikely to accept during these additional rounds).

Finally, to ensure that zero-knowledgeness of $(\mathcal{P}, \mathcal{V})$ is not affected, during the additional rounds the honest $\mathcal{P}'$ will simply send some fixed values to $\mathcal{V}'$ and disregard the values sent by $\mathcal{V}'$.

Each of the subsections below described the specific additional information in the keys and the additional rounds. I do not provide the details of proofs, as they can be easily derived from the discussion above.

Proof of Theorem 1

Let F be a pseudorandom function [GGM86]; I denote by $F_s(x)$ the output of F with seed s on input x. Note that such functions exist assuming one-way functions exist [HILL99]. Let x denote the theorem that the prover is trying to prove to the verifier.

Additional Key Gen. Step: Generate random n-bit seed s; add s to the secret key SK.

Additional Prover Step: Set $\beta = 0$; send β to the verifier.

Additional Verifier Step: If $\beta = F_s(x)$, accept and stop. Else send $F_s(x)$ to the prover.

Note that a sequential malicious prover can easily get $\mathcal{V}'$ to accept: it finds out the value of $F_s(x)$ in the first interaction, and sets $\beta = F_s(x)$ for the second. If, on the other hand, the malicious prover is not allowed to use the same x twice, then it cannot predict $F_s(x)$ before sending β, and thus cannot get $\mathcal{V}'$ to accept.

Proof of Theorem 2

Let (SigKeyGen, Sign, Ver) be a signature scheme secure against adaptive chosen message attacks [GMR88]. Note that such a scheme exists assuming one-way functions exist [Rom90].

Additional Key Generation Step: Generate a key pair $(\mathit{SigPK}, \mathit{SigSK})$ for the signature scheme; add SigPK to the public key PK and SigSK to the secret key SK.

First Additional Prover Step: Set $M = 0$, and send M to the verifier.

First Additional Verifier Step:
1. Send a signature s of M to the prover.
2. Let M' be random n-bit string; send M' to prover.

Second Additional Prover Step: Set $s' = 0$. Send s' to the verifier.

Second Additional Verifier Step: If s' is a valid signature of M', then accept and stop.

Note that a concurrent malicious prover can easily get $\mathcal{V}'$ to accept. It starts a protocol with $\mathcal{V}'$, sends $M = 0$, receives M' from $\mathcal{V}$, and then pauses the protocol. During the pause, it starts a second protocol, and sends $M = M'$ to $\mathcal{V}'$ to obtain a signature s of M' in first message from $\mathcal{V}'$. It then resumes the first protocol, and sends $s' = s$ to $\mathcal{V}'$ as its second message, which $\mathcal{V}'$ accepts.

Also note that a sequential malicious prover will most likely not be able to come up with a valid signature of M', because of the signature scheme's security against adaptive chosen message attacks.

Proof of Theorem 3

Additional Prover Step: Set β be the string of n zeroes; send β to the verifier.

Additional Verifier Step: Set α be a random string.
If $\beta = \alpha$, accept and stop. Else send α to the prover.

Note that a resetting malicious prover can easily get $\mathcal{V}'$ to accept: it finds out the value of α in the first interaction, then resets $\mathcal{V}'$ with the same random tape and sets $\beta = \alpha$ for the second interaction. A concurrent malicious prover, on the other hand, knows nothing about α when it determines β, and thus cannot get $\mathcal{V}'$ to accept.

Note that this separation holds in the standard model as well—I never used the BPK model in this proof.

Chapter 3

Upper Bounds on Round-Complexity of RZK in the BPK Model

This chapter is concerned with constructing protocols that satisfy one-time or sequential soundness in the BPK model, in as few rounds as possible.

3.1 One-Time Sound RZK

Recall Theorem 4:

Theorem 4 *Assuming the security of RSA with large prime exponents against adversaries that are subexponentially strong, for any* $L \in \mathrm{NP}$*, there exists a 3-round black-box RZK protocol in the BPK model that possesses one-time, but not sequential, soundness.*

Proof The proof of the theorem is constructive: I demonstrate such a protocol $(\mathcal{P}, \mathcal{V})$.

Basic Tools. The protocol $(\mathcal{P}, \mathcal{V})$ relies on three techniques: a pseudorandom function PRF [GGM86], a verifiable random functions VRF [MRV99], and a non-interactive zero-knowledge (NIZK) proof system $(\mathrm{NIP}, \mathrm{NIV})$ [BFM88, BDMP91] (these notions are recalled and defined in Appendix A). Note that both PRFs and NIZKs can be constructed using general assumptions [HILL99, FLS99], and it is only for VRFs that I need the specific RSA assumption (which is formally stated in Section A.4).

Let me briefly introduce the notation used for VRFs and NIZKs:

- The keys *VRFPK*, *VRFSK* for VRF are produced by VRFGen. The evaluation is performed by VRFEval, and the proof is computed by VRFProve. The verification is performed by VRFVer.
- The NIZK proof with security parameter n requires a shared random string σ of length $\mathrm{NI}\sigma\mathrm{Len}(n)$. The proof is computed by NIP and verified by NIV. The shared string and the proof together can by simulated by NIS.

The construction works for any language L for which an NIZK proof system exists, and, therefore, for all of NP.

This construction also uses "complexity leveraging" [CGGM00], although in a somewhat unusual way. Namely, let α be the pseudorandomness constant for VRF (that is, the output of the VRFEval is indistinguishable from random for circuits of size 2^{k^α}, where k is VRF the security parameter). Let γ_1 be the following constant: for all sufficiently large n, the length of the NIZK proof Π for a theorem $x \in L$, where x is of length n, is upper bounded by n^{γ_1}. Let γ_2 be the following constant: for all sufficiently large n, the length of the NP-witness y for an n-bit $x \in L$ is upper bounded by n^{γ_2}. I then set $\gamma = \max(\gamma_1, \gamma_2)$, and $\epsilon > \gamma/\alpha$. The protocol will use NIZK with security parameter n and VRF with a (larger) security parameter $k = n^\epsilon$. This ensures that one can enumerate all potential NIZK proofs Π, or all potential NP-witnesses y, in time 2^{n^γ}, which is less than the time it would take to break the residual pseudorandomness of VRF (because $2^{n^\gamma} < 2^{k^\alpha}$).

The Protocol. For a security parameter n, $\mathcal{V}$ generates a key pair for the VRF with output length NIσLen(n) and security parameter $k = n^\epsilon$. *VRFSK* is $\mathcal{V}$'s secret key, and *VRFPK* is $\mathcal{V}$'s public key.

Protocol $(\mathcal{P}, \mathcal{V})$

Public File: A collection F of records $(id, \mathit{VRFPK}_{id})$, where VRFPK_{id} is allegedly the output of VRFGen(1^k)

Common Input: An element $x \in L$

$\mathcal{P}$ *Private Input:* The NP-witness y for $x \in L$; $\mathcal{V}$'s id and the file F; a random string ω

$\mathcal{V}$ *Private Input:* A secret key *SK*

$\mathcal{P}$ *Step One:*

1. Using the random string ω as a seed for PRF, generate a string σ_P of length NIσLen(n) from the inputs x, y and id.
2. Send $\sigma_{\mathcal{P}}$ to $\mathcal{V}$.

$\mathcal{V}$ *Step One:*

1. Compute a string $\sigma_{\mathcal{V}}$ of length NIσLen(n) as $\sigma_{\mathcal{V}} = \text{VRFEval}(\mathit{VRFSK}, x)$, and the VRF proof $pf = \text{VRFProve}(\mathit{VRFSK}, x)$.
2. Send $\sigma_{\mathcal{P}}$ and pf to $\mathcal{P}$.

$\mathcal{P}$ *Step Two:*

1. Verify that $\sigma_{\mathcal{V}}$ is correct by invoking VRFVer$(\mathit{VRFPK}, x, \tau, pf)$. If not, abort.
2. Let $\sigma = \sigma_{\mathcal{V}} \oplus \sigma_{\mathcal{P}}$. Using NIP$(\sigma, x, y)$, compute and send to $\mathcal{V}$ the proof Π of the statement "$x \in L$."

$\mathcal{V}$ *Step Two:*

1. Let $\sigma = \sigma_{\mathcal{V}} \oplus \sigma_{\mathcal{P}}$. Using NIV$(\sigma, x, \Pi)$, verify if Π is valid. If so, accept. Else reject.

As far as I know, the above protocol is the first application of VRFs. The very strong properties of this new tool yield surprisingly simple proofs of one-time soundness and black-box resettable zero-knowledgeness, presented below (completeness, as if often the case, is trivially verified).

Soundness. First of all, note that soundness of the above protocol is provably not sequential, because $\sigma_{\mathcal{V}}$ depends only on the input x, and hence will repeat if $\mathcal{V}$ is run with the same x twice. Thus, once a sequential malicious prover $\mathcal{P}^*$ knows $\sigma_{\mathcal{V}}$, it can run the NIZK simulator $\mathrm{NIS}(x)$ to obtain (σ', Π'), restart with the same x, and use $\sigma'_{\mathcal{P}} = \sigma' \oplus \sigma_{\mathcal{V}}$ as its first message and Π' as its second message.

To show one-time soundness, first assume (for simplicity) that $\mathcal{P}^*$ interacts with $\mathcal{V}$ only once (I will deal with the general case later). Then I will construct a machine $T = (T_J, T_E)$ to break the residual pseudorandomness of the VRF (see the definition of VRF in Section A.4). Namely, given the public key *VRFPK* of a VRF with security parameter k, T_J runs the first stage of $\mathcal{P}^*$ on input *VRFPK* to receive a string x. It then checks if $x \in L$ by simply enumerating all potential NP witnesses y in time $2^{n^{\gamma_2}}$. If it is, then T_J outputs (x, state), where $\mathit{state} = 0$. Otherwise, it runs the second stage of $\mathcal{P}^*$ to receive σ_P, and outputs (x, state), where $\mathit{state} = (x, \sigma_{\mathcal{P}})$.

Now, T_E receives v, and T_E's job is to find out whether v is a random string or $\mathrm{VRFEval}(\mathit{VRFSK}, x)$. If $\mathit{state} = 0$, then T_E simply guesses at random. Otherwise, $\mathit{state} = (x, \sigma_{\mathcal{P}})$. Let $\sigma = \sigma_{\mathcal{P}} \oplus v$. If v is a random string, then σ is also random, so most likely there is no NIZK proof Π of the statement "$x \in L$" with respect to σ (by soundness of the NIZK proof system). Otherwise, $v = \sigma_{\mathcal{V}}$, so, if $\mathcal{P}^*$ has a better than negligible probability of success, then there is a better than negligible probability that Π exists with respect to σ. Thus, T_E simply searches whether a proof Π exists (in time $2^{n^{\gamma_1}}$) to determine whether v is random or the output of VRFEval.

Note that complexity leveraging is crucial here: I am using the fact that the VRF is "stronger" than the non-interactive proof system. Otherwise, the output of VRFProve (which the prover gets, but T does not) could help a malicious prover find Π. By using a stronger VRF, I am ensuring that such Π will most likely not even exist.

Now I address the general case, when $\mathcal{P}^*$ is allowed multiple interactions with $\mathcal{V}$. Suppose $\mathcal{P}^*$ is an s-sequential malicious prover. Then $\mathcal{P}^*$ initiates at most $s(n)$ *sequential* conversations and wins if $\mathcal{V}$ accepts at least one of them (say, the i-th one) for $x_i \notin L$. Moreover, because I am only proving one-time soundness, $\mathcal{P}^*$ is not allowed to attempt proving $x_i \in L$ in any previous protocol. Then T_J simply guesses, at random, the conversation number i for which $\mathcal{P}^*$ will succeed, and simulates conversations before the i-th one by querying VRFEval and VRFProve on x_j for $j < i$ (it is allowed to do so, as long as $x_j \neq x_i$). Because the guessed i may be incorrect, this reduces T's advantage by a polynomial factor of $s(n)$.

Resettable Zero-Knowledgeness. The RZK property can be shown in a way similar to (and simpler than) the RZK property is shown for the public-key protocol of [CGGM00]. One simply builds an RZK simulator who finds out $\mathrm{VRFEval}(\mathit{VRFSK}, x)$ for every pair (VRFPK, x) that $\mathcal{V}^*$ is likely to input to $\mathcal{P}$, and then rewinds and uses the NIZK simulator $\mathrm{NIS}(x)$ just like the sequential malicious prover described above. ■

3.2 Sequentially-Sound RZK

The CGGM Upper Bound

Although [CGGM00] did not provide formal definitions of soundness in the BPK model, their soundness proof essentially shows that their BPK model protocol is sequentially sound. Also, a careful interleaving of rounds allows for their protocol to be 5-round. Thus, they prove that, assuming subexponential hardness of discrete logarithms, 5-round sequentially sounds RZK protocols in the BPK model exist for all of NP.

Let me now explain why it will probably not be possible to prove their protocol concurrently sound. The CGGM protocol begins with $\mathcal{V}$ proving to $\mathcal{P}$ knowledge of the secret key by means of parallel repetitions of a three-round proof of knowledge subprotocol. The subprotocol is as follows: in the first round, $\mathcal{V}$ sends to $\mathcal{P}$ a *commitment*; in the second round, $\mathcal{P}$ sends to $\mathcal{V}$ a one-bit *challenge*; in the third round, $\mathcal{V}$ sends to $\mathcal{P}$ a *response*. This is repeated k times in parallel in order to reduce the probability of $\mathcal{V}$ cheating to roughly 2^{-k}.

In order to prove soundness against a malicious prover $\mathcal{P}^*$, these parallel repetitions of the subprotocol need to be simulated to $\mathcal{P}^*$ (by a simulator that does not know the secret key). The best known simulation techniques for this general type of proof of knowledge run in time roughly 2^k. This exponential in k simulation time is not a concern, because of their use of "complexity leveraging" in the proof of soundness. Essentially, the soundness of their protocol relies on an underlying much harder problem: for instance, one that is assumed to take more than 2^{3k} time to solve. Thus, the soundness of the CGGM protocol is proved by contradiction: by constructing a machine from $\mathcal{P}^*$ that runs in time $2^k < 2^{3k}$ and yet solves the underlying harder problem.

A concurrent malicious prover $\mathcal{P}^*$, however, may choose to run L parallel copies of $\mathcal{V}$. Thus, to prove soundness against such a $\mathcal{P}^*$, the proof-of-knowledge subprotocol would have to be simulated Lk times in parallel, and this simulation would take roughly 2^{Lk} time. If $L > 3$, then one will not be able to solve the underlying hard problem in time less than 2^{3k}, and thus will not be able to derive any contradiction.

Thus, barring the emergence of a polynomial-time simulation for parallel repetitions of 3-round proofs of knowledge (or a dramatically new proof technique for soundness), the CGGM protocol is not provably concurrently sound.

A New Upper Bound

Recall Theorem 5:

Theorem 5 *Assuming there exist certified trapdoor permutation families*[1] *secure against subexponentially-strong adversaries, for any* $L \in \mathrm{NP}$*, there exists a 4-round black-box RZK argument in the BPK model that possesses sequential soundness.*

Proof The proof is, again, constructive. The construction is a modification of the CGGM protocol (which has 8 rounds, and can easily be modified to have 5 by combining the first three rounds with later rounds). The high-level description is as follows.

Main Ideas. The CGGM protocol starts with a three-round proof of knowledge subprotocol in which $\mathcal{V}$ proves to $\mathcal{P}$ knowledge of the secret key. After that, $\mathcal{P}$ proves to $\mathcal{V}$ that a graph is three-colorable using a five-round protocol.

The main idea is to replace the five-round protocol with a single round using non-interactive zero-knowledge (see Appendix A for the definition). The first three rounds are then used both for the proof of knowledge and for agreeing on a shared random auxiliary string σ needed for the NIZK proof. To agree on σ, $\mathcal{V}$ sends to $\mathcal{P}$ an encryption of a random string $\sigma_{\mathcal{V}}$, $\mathcal{P}$ sends to $\mathcal{V}$ its own random string $\sigma_{\mathcal{P}}$, and then $\mathcal{V}$ reveals $\sigma_{\mathcal{V}}$ (and the coins used to encrypt it). The string σ is computed as $\sigma_{\mathcal{P}} \oplus \sigma_{\mathcal{V}}$. Thus, $\mathcal{V}$'s key pair is simply a key pair for an encryption scheme. The protocol is RZK essentially for the same reasons that

[1] *A trapdoor permutation family is* certified *if it is easy to verify that a given function belongs to the family.*

the CGGM protocol is RZK: because the simulator can extract the decryption key from the proof of knowledge and thus find out $\sigma_{\mathcal{V}}$ before needing to submit $\sigma_{\mathcal{P}}$. This will allow it to select σ as it wishes and thus use the NIZK simulator.

The protocol is sequentially sound because if the theorem is false, then with respect to only a negligible portion of the possible strings σ does a NIZK proof of the theorem exist. Thus, if a malicious prover $\mathcal{P}^*$, after seeing only an encryption of $\sigma_{\mathcal{V}}$, is able to come up with $\sigma_{\mathcal{P}}$ such that the NIZK proof exists with respect to the resulting string $\sigma = \sigma_P \oplus \sigma_{\mathcal{V}}$, then one can use $\mathcal{P}^*$ to break the security of the encryption scheme.

The computational assumption for this protocol follows from the fact that trapdoor permutations are sufficient for encryption [GM84, Yao82, GL89], certified trapdoor permutations are sufficient for NIZKs [FLS99], one-way permutations are sufficient for the proof of knowledge [Blu86] (which is the same as in the CGGM protocol) and one-way functions are sufficient for PRFs [HILL99].

Details of The Construction. This construction, like the previous one, works for any languages L for which an NIZK proof system exists. Hence it works for all $L \in NP$.

The protocol relies on parallel executions of three-round proofs of knowledge, which are performed in exactly the same way as in [CGGM00]. It also uses "complexity leveraging," in a way similar to the above three-round one-time-sound construction. Namely, let α be the indistinguishability constant for the encryption scheme (that is, the encryptions of two different strings are indistinguishable from each other for circuits of size 2^{k^α}, where k is the security parameter). Let γ_1 be the following constant: for all sufficiently large n, the length of the NIZK proof Π for x of length n is upper bounded by n^{γ_1}. Let γ_2 be following constant: n parallel repetitions of the proof-of-knowledge subprotocol can be simulated in time less that $2^{n^{\gamma_2}}$. Then set $\gamma = \max(\gamma_1, \gamma_2)$, and $\epsilon > \gamma/\alpha$. The protocol uses NIZK with security parameter n and performs n parallel repetitions of the proof-of-knowledge subprotocol, while the encryption scheme has a (larger) security parameter $k = n^\epsilon$. This ensures that one can enumerate all potential NIZK proofs Π and simulate the proof of knowledge subprotocol in time 2^{n^γ}, which is less than the time it would take to break the indistinguishability of the encryption scheme (because $2^{n^\gamma} < 2^{k^\alpha}$).

To generate its key pair for security parameter n, $\mathcal{V}$ generates a pair $(\mathit{EncPK}, \mathit{EncSK})$ of keys for the encryption scheme with security parameter $k = n^\epsilon$. EncSK is $\mathcal{V}$'s secret key, and EncPK is $\mathcal{V}$'s public key.

Protocol $(\mathcal{P}, \mathcal{V})$

Public File: A collection F of records $(id, \mathit{EncPK}_{id})$, where EncPK_{id} is allegedly the output of $\mathcal{V}$'s key generation

Common Inputs: An element $x \in L$

$\mathcal{P}$ *Private Input:* The NP-witness y for $x \in L$; $\mathcal{V}$'s id and the file F; a random string ω

$\mathcal{V}$ *Private Input:* A secret key EncSK; a random string ρ

$\mathcal{V}$ *Step One:*

1. Generate a random string $\sigma_{\mathcal{V}}$ of length NIσLen(n).
2. Encrypt $\sigma_{\mathcal{V}}$, using a portion ρ_E of the input random string ρ, to get a ciphertext c. Send c to $\mathcal{P}$.

3. Generate and send to $\mathcal{P}$ the first message of the n parallel repetitions of the proof of knowledge of EncSK.

$\mathcal{P}$ Step One:

1. Using the input random string ω as a seed for PRF, generate a sufficiently long "random" string from the input to be used in the remaining computation by $\mathcal{P}$.
2. Generate and send to $\mathcal{V}$ a random string $\sigma_{\mathcal{P}}$ of length $\mathrm{NI}\sigma\mathrm{Len}(n)$.
3. Generate and send to $\mathcal{V}$ the second message of the n parallel repetitions of the proof of knowledge of EncSK.

$\mathcal{V}$ Step Two:

1. Send $\sigma_{\mathcal{V}}$ and the coins ρ_E used to encrypt it to $\mathcal{P}$.
2. Generate and send the third message of the n parallel repetitions of the proof of knowledge of EncSK.

$\mathcal{P}$ Step Two:

1. Verify that $\sigma_{\mathcal{V}}$ encrypted with coins ρ_E produces ciphertext c.
2. Verify the n parallel repetitions proof of knowledge of EncSK.
3. If both verifications hold, let $\sigma = \sigma_{\mathcal{V}} \oplus \sigma_{\mathcal{P}}$. Using the NIZK prover $\mathrm{NIP}(\sigma, x, y)$, compute and send to $\mathcal{V}$ the proof Π of the statement "$x \in L$."

$\mathcal{V}$ Step Three:

1. Let $\sigma = \sigma_{\mathcal{V}} \oplus \sigma_{\mathcal{P}}$. Using the NIZK verifier $\mathrm{NIV}(\sigma, x, \Pi)$, verify if Π is valid. If so, accept. Else reject.

The proofs of soundness and black-box resettable zero-knowledgeness are presented below (completeness, again, is easy to verify).

Soundness. Sequential soundness can be shown as follows. Suppose $\mathcal{P}^*$ is a malicious prover that can make $\mathcal{V}$ accept a false theorem with probability $p(n)$ (where the probability is taken over the coin tosses of the $\mathcal{V}$ and $\mathcal{P}^*$). First, assume (for simplicity) that $\mathcal{P}^*$ interacts with $\mathcal{V}$ only once (I will deal with the general case of a sequential malicious prover later).

I will use $\mathcal{P}^*$ to construct an algorithm A that breaks the encryption scheme. A is given, as input, the public key EncPK for the encryption scheme. Its job is to pick two strings τ_0 and τ_1, receive an encryption of τ_b for a random bit b and tell whether $b = 0$ or $b = 1$. It picks τ_0 and τ_1 simply as random strings of length $\mathrm{NI}\sigma\mathrm{Len}(n)$. Let c be the encryption of τ_b. Then A publishes EncPK as its public key, runs the first stage of $\mathcal{P}^*$ to receive x, and initiates a protocol with the second stage of $\mathcal{P}^*$.

In the first message, A sends c for the encryption of $\sigma_{\mathcal{V}}$ (for the proof-of-knowledge subprotocol, A uses the simulator, which runs in time $2^{n^{\gamma_2}}$). It then receives $\sigma_{\mathcal{P}}$ from $\mathcal{P}^*$, computes $\sigma_i = \sigma_{\mathcal{P}} \oplus \tau_i$ and determines (by exhaustive search, which takes time $2^{n^{\gamma_1}}$) if there exists an NIZK proof Π_i for the statement $x \in L$ with respect to σ_i (for $i = 0, 1$). If Π_i exists and Π_{1-i} does not, then A outputs $b = i$. If neither Π_0 nor Π_1 exists, or if both exist, then A outputs a random guess for b.

I now need to compute the probability that A correctly guessed b. Of course, by construction,
$\Pr[A \text{ outputs } b] = \Pr[\exists\Pi_b \text{ and } \nexists\Pi_{1-b}] + \Pr[\exists\Pi_b \text{ and } \exists\Pi_{1-b}]/2 + \Pr[\nexists\Pi_b \text{ and } \nexists\Pi_{1-b}]/2\,.$
Note that
$\Pr[\exists\Pi_b \text{ and } \exists\Pi_{1-b}] + \Pr[\nexists\Pi_b \text{ and } \nexists\Pi_{1-b}] = 1 - (\Pr[\exists\Pi_b \text{ and } \nexists\Pi_{1-b}] + \Pr[\nexists\Pi_b \text{ and } \exists\Pi_{1-b}]).$
Therefore, $\Pr[A \text{ outputs } b] = 1/2 - \Pr[\nexists\Pi_b \text{ and } \exists\Pi_{1-b}]/2 + \Pr[\exists\Pi_b \text{ and } \nexists\Pi_{1-b}]/2\,.$

Note that the either of the events $\nexists\Pi_b$ and $\nexists\Pi_{1-b}$ can occur only if $x \notin L$, by completeness of the NIZK system. Therefore,

$$\begin{aligned}
\Pr[A \text{ outputs } b] &= 1/2 - \Pr[\nexists\Pi_b \text{ and } \exists\Pi_{1-b} \text{ and } x \notin L]/2 \\
&\quad + \Pr[\exists\Pi_b \text{ and } \nexists\Pi_{1-b} \text{ and } x \notin L]/2 \\
&= 1/2 - \Pr[\nexists\Pi_b \text{ and } \exists\Pi_{1-b} \text{ and } x \notin L]/2 \\
&\quad + \Pr[\exists\Pi_b \text{ and } x \notin L]/2 - \Pr[\exists\Pi_b \text{ and } \exists\Pi_{1-b} \text{ and } x \notin L]/2 \\
&\geq 1/2 + p(n)/2 - \Pr[\exists\Pi_{1-b} \text{ and } x \notin L]\,.
\end{aligned}$$

However, τ_{1-b} is picked uniformly at random and $\mathcal{P}^*$ receives no information about it, so the string $\sigma_{1-b} = \sigma_{\mathcal{P}} \oplus \tau_{1-b}$ is distributed uniformly at random, so, by soundness of NIZK, $\Pr[\exists\Pi_{1-b}$ and $x \notin L]$ is negligible in n. Thus, A's advantage is only negligibly less than $p(n)/2$.

Now I address the case of sequential malicious provers. Suppose $\mathcal{P}^*$ is an s-sequential malicious prover. Then $\mathcal{P}^*$ initiates at most $s(n)$ *sequential* conversations and wins if $\mathcal{V}$ accepts at least one of them for $x \notin L$. Then A simply guesses, at random, the conversation for which $\mathcal{P}^*$ will succeed, and simulates the other conversations by using the simulator for the proof of knowledge and honestly encrypting random strings. Only for that one conversation does it use the procedure described above. This reduces A's advantage by a polynomial factor of at most $s(n)$.

Resettable Zero-Knowledgeness. The proof of resettable zero-knowledgeness is very similar to that of [CGGM00]: once the simulator recovers SK from the proof of knowledge, it can find out $\sigma_{\mathcal{V}}$ before having to send $\sigma_{\mathcal{P}}$, and thus can run the NIZK simulator to get (σ, Π) and set $\sigma_{\mathcal{P}} = \sigma \oplus \sigma_{\mathcal{V}}$. ■

Chapter 4

Lower Bounds on Round-Complexity of ZK in the BPK Model

This chapter is concerned with proving lowerbounds for the number of rounds necessary for a ZK (and, therefore, RZK) protocol in the BPK model. Note that the lowerbounds work both for proofs (which have unconditional soundness) and arguments (which have only computational soundness).

4.1 Auxiliary-Input ZK with Any Soundness

Recall Theorem 6:

Theorem 6 *Any (resettable or not) auxiliary-input ZK protocol (satisfying one-time or stronger soundness) in the BPK model for a language outside of BPP requires at least three rounds.*

Proof The theorem follows immediately from the following lower bound by Goldreich and Oren [GO94]: any auxiliary-input ZK protocol (in the standard model) for a language outside of BPP requires at least 3 rounds.

More precisely, if a 2-round protocol $(\mathcal{P}, \mathcal{V})$ in the BPK model existed, one could construct from it a 2-round protocol $(\mathcal{P}', \mathcal{V}')$ in the standard model by simply sending the verifier's public key to the prover in the first round (note that this does not add an extra round, because $\mathcal{V}$ goes first in a 2-round protocol). The resulting protocol would still be (trivially) complete. It would be sound, as well, because if it weren't, one could construct a cheating prover $\mathcal{P}^*$ who breaks one-time soundness of $(\mathcal{P}, \mathcal{V})$ from a cheating prover $\mathcal{P}'^*$ for $(\mathcal{P}', \mathcal{V}')$ (because $\mathcal{P}'^*$, working in the standard model, is allowed only one interaction with $\mathcal{V}'$, the resulting $\mathcal{P}^*$ would only need one-time access to $\mathcal{V}$). Finally, it would be auxiliary-input ZK: one can simply view a malicious standard-model verifier $\mathcal{V}'^*$ as a malicious BPK-model verifier $\mathcal{V}^*$, who "publishes" the public key, instead of sending it in the first message. Because the $(\mathcal{P}, \mathcal{V})$ is ZK, for $\mathcal{V}^*$ there is a simulator $M_{\mathcal{V}^*}$. This same simulator would work for $\mathcal{V}'^*$, as well, because the views of $\mathcal{V}^*$ and $\mathcal{V}'^*$ are the same.

The existence of $(\mathcal{P}', \mathcal{V}')$ would contradict the [GO94] lower bound. ■

4.2 Black-Box ZK with Concurrent Soundness

Recall Theorem 7:

Theorem 7 *Any (resettable or not) black-box ZK protocol satisfying concurrent soundness in the BPK model for a language outside of BPP requires at least four rounds.*

Proof This theorem follows from a careful examination of the proof of the following theorem by Goldreich and Krawczyk [GK96]: in the standard model, any black-box ZK protocol for a language outside of BPP requires at least four rounds. Familiarity with that (very well-written) proof is helpful for understanding this one, because I do not repeat the parts of the proof that are identical, and only highlight the crucial differences.

Note that, in the standard model, one-time, sequential and concurrent soundness coincide. Thus, the above theorem essentially says that, although the [GK96] proof can be easily extended to verifiers that have public and secret keys, this extension fails to apply to some types of soundness (as already clear from the three-round one-time sound black-box RZK protocol of Section 3.1). The reasons for this are explained below.

The [GK96] proof proceeds by contradiction. Assuming the existence of a black-box zero-knowledge simulator M, it constructs a BPP machine $\bar{M}$ for L. Recall that M interacts with a verifier in order to output the verifier's view. On input x, $\bar{M}$ works essentially as follows: it simply runs M on input x, simulating a verifier to it. For this simulation, $\bar{M}$ uses the algorithm of the honest verifier $\mathcal{V}$ and the messages supplied by M, but ignores the random strings supplied by M and uses its own random strings (if the same message is given twice by M, then $\bar{M}$ uses the same random string—thus making the verifier appear deterministic to M). If the view that M outputs at the end is accepting, then $\bar{M}$ concludes that $x \in L$. Otherwise, it concludes that $x \notin L$.

To show that $\bar{M}$ is a BPP machine for L, Goldreich and Krawczyk demonstrate two statements: that if $x \in L$, M is likely to output an accepting conversation, and that if $x \notin L$, M is unlikely to output an accepting conversation. The first statement follows because, by zero-knowledgeness, M's output is indistinguishable from the view generated by the true prover and the true verifier on input x, and, by completeness, this view is accepting. The second statement follows from soundness: if M can output an accepting conversation for $x \notin L$, then one can construct a malicious prover $\mathcal{P}^*$ that can convince $\mathcal{V}$ of the false statement "$x \in L$." Such a $\mathcal{P}^*$ needs in essence to "execute M" and simply let it interact with $\mathcal{V}$.

Having $\mathcal{P}^*$ execute M requires some care. At first glance, because simulator M is capable of resetting the verifier, it would seem that, in order to execute M, also $\mathcal{P}^*$ should have this capability. However, for 3-round protocols only, [GK96] show that

($*$) $\mathcal{P}^*$ can execute M without resetting $\mathcal{V}$, so long as it has one-time access to $\mathcal{V}$.

Notice that by the term "one-time access" I make retroactive use of the current terminology: [GK96] make no mention of one-time provers, because they work in the standard model. However, this terminology allows me to separate their proof of ($*$) into two distinct steps:

($*'$) $\mathcal{P}^*$ can execute M so long as it has concurrent access to $\mathcal{V}$; and

($*''$) losing only a polynomial amount of efficiency, concurrent access to $\mathcal{V}$ is equivalent to one-time access.

Tedious but straightforward analysis shows that $(*')$ and the rest of their proof—except for $(*'')$—carries through in the BPK model (where the 3-round protocol is modified to include verifier key generation, and public and secret verifier keys are then involved). Step $(*'')$, however, only holds in the standard model (where, as pointed out, one-time, sequential and concurrent soundness coincide).

In sum, therefore, once verifier keys are introduced, one is left with a concurrent prover. ■

4.3 Black-Box ZK with Resettable Soundness

Recall Theorem 8:

Theorem 8 *There is no (resettable or not) black-box ZK protocol satisfying resettable soundness in the BPK model for a language outside of BPP.*

Proof This proof, like the proof of Theorem 7, is similar to the Goldreich's and Krawczyk's [GK96] proof that no 3-round black-box ZK protocols exist for languages outside of BPP. I will provide more details here than in the proof of Theorem 7. However, I will only sketch some of the parts that are very similar to those of the [GK96] proof.

Suppose a protocol $(\mathcal{P}, \mathcal{V})$ for a language L is resettably sound and black-box ZK. Let M be the black-box ZK simulator. I will construct a BPP machine $\bar{M}$ for L, using M as a subroutine. Note that M expects to interact with an oracle for the (malicious) verifier, so $\bar{M}$ will have to answers M's oracle queries.

Denote the prover messages in the protocol by $\alpha_1, \alpha_2, \ldots, \alpha_R$, and the verifier messages by $\beta_1, \beta_2, \beta_R$ (note that R is not necessarily a constant). Assume, without loss of generality, that the first message β_1 is sent by the verifier, and the last message α_R is sent by the prover. For simplicity of notation, let β_{R+1} be the output of the $\mathcal{V}$ after the interaction ("accept x" or "reject x").

Denote the honest verifier's random tape for key generation by r, and for the protocol itself by ρ. Denote the first (key-generation) stage of the honest verifier by $\mathcal{V}^1$, and the second (interaction) stage by $\mathcal{V}^2$. Then $(PK, SK) = \mathcal{V}^1(1^n, r)$, and $\beta_i = \mathcal{V}^2(SK, x, \rho, \alpha_1, \ldots, \alpha_{i-1})$.

Let $\mathcal{V}^*$ be a non-resetting malicious verifier. Denote its random tape by s. For $x \in L$ and its NP witness y, $\mathcal{V}^*$ on input (s, x) outputs a single-record public file $F = \{(id, PK_{id})\}$, and then expects to interact with $\mathcal{P}$ who is given inputs x, y, id, F. Including (id, PK_{id}) into the first verifier message β_1, one can view $\mathcal{V}^*$ simply as a deterministic oracle: $\beta_i = \mathcal{V}^*(s, x, \alpha_1, \ldots, \alpha_{i-1})$.

The black-box simulator M receives x as an input and $\mathcal{V}^*$ as an oracle, has to simulate the view of $\mathcal{V}^*$ when conversing with $\mathcal{P}$ on common input x (the view of $\mathcal{V}^*$ consists of its input random string and the messages it receives from $\mathcal{P}$).

The BPP machine $\bar{M}$, on the other hand, receives x as input, and has to decide whether $x \in L$. $\bar{M}$ simply runs M on input x and answers M's oracle queries just like the honest verifier $\mathcal{V}$, except that $\bar{M}$ substitutes its own random strings for those given by M. Note that no $\mathcal{V}^*$ is present in this interaction: rather, $\bar{M}$ simulates $\mathcal{V}^*$ for M using $\mathcal{V}$. More specifically,

- $\bar{M}$ maintains a table of triples of strings: $(s_1, r_1, \rho_1), (s_2, r_2, \rho_2), \ldots$ This table records the substitutions of random strings that $\bar{M}$ will make in the inputs to the verifier. The table is initially empty.

- If M queries $\mathcal{V}^*$ on $(s, x', \alpha_1, \ldots, \alpha_{i-1})$ (where x' may or may not equal x), then then $\bar{M}$ looks up a triple (s, r, ρ) in its table, if one exists. If one doesn't exist, then $\bar{M}$ generates a random r and ρ and adds (s, r, ρ) to its table. $\bar{M}$ then runs the key generation stage of the honest verifier with random string r: $(PK, SK) = \mathcal{V}^1(1^n, r)$; and the interaction stage of the honest verifier with random string ρ: $\beta_i = V^2(SK, x', \rho, \alpha_1, \ldots, \alpha_{i-1})$. Then $\bar{M}$ returns β_i as the response to M's query (if $i = 1$, then $\bar{M}$ also selects an id and adds (id, PK) to the β_1).

At the end, M outputs some "view": $(s, \alpha_1, \ldots, \alpha_R)$. Assume, without loss of generality, that M already queried $\mathcal{V}^*$ on the string s, so that the corresponding r and ρ are in the table (if M did not make such a query, $\bar{M}$ can make the query on its own after seeing M's output). Then $\bar{M}$ gets SK from running $\mathcal{V}^1(1^{|x|}, r)$ and β_{R+1} from running $\mathcal{V}^2(SK, x, \rho, \alpha_1, \ldots, \alpha_R)$. If $\beta_{R+1} =$"accept x," then $\bar{M}$ outputs "accept". Else, $\bar{M}$ outputs "reject."

I now have to prove two lemmas.

Lemma 1 *If $x \notin L$, then $\bar{M}$ outputs "accept" with negligible probability.*

Proof I will construct a resetting malicious prover $\mathcal{P}^*$ out of M. Recall that $\mathcal{P}^*$ receives as input a public key PK, and gets to interact with oracles for $\mathcal{V}$ (without knowing their random strings r and ρ and the secret key SK). In fact, in this construction $\mathcal{P}^*$ will need to interact with only one such oracle. The job of $\mathcal{P}^*$ is get the oracle to output "accept x."

Suppose that M will use no more that q distinct random strings $s_1, \ldots, s_q$ in its queries (q is polynomial in n, because the running time of M is polynomial). P^* will first guess a random j between 1 and q. Then, $\mathcal{P}^*$ will run M just like $\bar{M}$, simulating $\mathcal{V}^*$'s answers, with one exception: the first time that $\mathcal{P}^*$ sees the j-th distinct random string s_j in a query from M, and every time thereafter that a query from M uses s_j, P^* will pass M's query to the honest verifier $\mathcal{V}$ whom $\mathcal{P}^*$ is trying to cheat. That is, if $\mathcal{P}^*$ receives a query $(s_j, x', \alpha_1, \ldots, \alpha_{i-1})$, it gives $(x', \alpha_1, \ldots, \alpha_{i-1})$ as input to its oracle for $\mathcal{V}$, and returns $\mathcal{V}$'s output β_i to M (if $i = 1$, then $\mathcal{P}^*$ also selects an id and adds (id, PK) to β_1).

At the end, M will output $(s, \alpha_1, \ldots, \alpha_R)$. Because j was chosen at random, with probability $1/q$, $s = s_j$. If this is the case, then $\mathcal{P}^*$ gives $(x, \alpha_1, \ldots, \alpha_R)$ as input to its oracle for V. $\mathcal{P}^*$ succeeds if β_{R+1} received from $\mathcal{V}$ is "accept x."

Note that $\mathcal{P}^*$'s behavior is the same as $\bar{M}$'s, except that $\mathcal{P}^*$ does not know r_j and ρ_j, and therefore has to go to the oracle for $\mathcal{V}$ whenever M uses s_j. Therefore, the probability that the output of M is an "accepting" conversation does not change. Thus, the probability that $\mathcal{P}^*$ succeeds is simply q times less than the probability that $\bar{M}$ outputs "accept." Because the protocol is assumed to be resettably sound, this quantity is negligible; hence the probability that $\bar{M}$ outputs "accept" is also negligible. ■

Lemma 2 *If $x \in L$, then $\bar{M}$ outputs "accept" with high probability.*

Proof Assume, again, that M will use no more that q distinct random strings $s_1, \ldots, s_q$ in its queries (where q is polynomial in n).

Let H be a family of q-wise independent hash functions [Jof74, WC81, CG89] Consider a family of (malicious) verifiers indexed by H. For $h \in H$, $\mathcal{V}_h^*$ is the verifier that, on input random string s, generates $(r, \rho) = h(s)$ and then behaves like the honest verifier $\mathcal{V}$ on random strings r and ρ. By completeness of $(\mathcal{P}, \mathcal{V})$ and because $\mathcal{V}_h^*$ looks to the honest prover $\mathcal{P}$ just like the honest verifier $\mathcal{V}$, when $\mathcal{P}$ interacts with $\mathcal{V}_h^*$, the resulting view $(s, \alpha_1, \ldots, \alpha_R)$ of $\mathcal{V}_h^*$ will be "accepting" (that is, $\mathcal{V}^2(SK, x, \rho, \alpha_1, \ldots, \alpha_R) =$"accept x," where $SK = \mathcal{V}^1(1^{|x|}, r)$ and $(r, \rho) = h(s)$).

Thus, by zero-knowledgeness, when M interacts with $\mathcal{V}_h^*$, it should also producing an "accepting" view with high probability. On the other hand, an interaction with $\bar{M}$ looks to M just like an interaction with $\mathcal{V}_h^*$ for a random h (in fact, $\bar{M}$ simply constructs a portion of h on the fly). Therefore, M should output an "accepting" conversation with high probability. ■

Note that in the proofs of the above two lemmas, I assumed that M's running time is a priori bounded by a polynomial. However, the lemmas extend to simulators that run in expected polynomial time, as well. This is done by the same techniques as used in [GK96] (see remark 6.1 therein): one simply truncates the execution of M after a certain number of steps, chosen in such a way that M still has a good chance of faithfully simulating the conversation. ■

Chapter 5

Three-Round Concurrently Sound RZK Protocol for NP in the UPK Model

Theorem 7 precludes the existence of three-round concurrently sound black-box RZK protocols in the BPK model. Surprisingly, only a slight strengthening of the model (namely, endowing $\mathcal{V}$ with an upper bound and a counter) allows for such protocols to exist for all of NP. Recall Theorem 9:

Theorem 9 *In the UPK model there exist 3-round concurrently sound black-box RZK arguments for any language in NP, assuming collision-resistant hashing and the subexponential hardness of discrete logarithm and integer factorization.*[1]

Proof This proof is constructive. The constructed protocol is rather lengthy and technical. It is therefore helpful to first present a high-level overview of the construction.

Why the Obvious Solution Does Not Work. Before I begin, let me demonstrate that my goal cannot be more easily achieved by the following simpler construction.

Let $cmax = U(n)$ be the upperbound on the number of uses of the verifier's public key (i.e., the max value for the verifier's counter). Take a four-round ZK protocol (e.g., the one of [FS89]), and have the verifier simply post *cmax* independently generated first-round messages in its public key. Then execution number c simply uses first-round message number c appearing in the public key, and then performs the remaining three rounds of the protocol as before.

The above construction does not work, because the prover does not know the real value c of the verifier's counter. This enables a malicious verifier to choose the value of c after it sees the prover's first message. Thus, if such a verifier resets the prover while varying c, it will typically gain knowledge. (Typically, in a 4-round ZK protocol, the verifier commits to a question without revealing it, the prover sends a first message, the verifier asks the question, and the prover answers it. However, if the prover were to answer two different questions relative to the same first message, then zero-knowledgeness disappears. Now, in the above construction, varying c enables the verifier to ask different questions.)

[1] *One can replace the integer factorization assumption with the more general assumption that subexponentially secure dense public-key cryptosystems [DDP00] and subexponentially secure certified trapdoor permutations [FLS99] exist. Or one can replace both the DL and the factorization assumptions with the assumption that decision Diffie-Hellman is subexponentially hard.*

High-Level Description. I therefore have to present an entirely new construction. There are, however, many similarities to the protocol of [CGGM00]. Like the CGGM protocol, this protocol uses the NP-complete language of graph 3-colorability and the parallel repetition of the protocol of [GMW91] as the starting point. Thus, in the first round, $\mathcal{P}$ commits to a number of random recolorings of a graph G, in the second round $\mathcal{V}$ requests to reveal the colors of one edge for each committed recoloring, and in the third round $\mathcal{P}$ opens the relevant commitments.

To allow the RZK simulator to work, the protocol also uses trapdoor commitment schemes (see Appendix A for a definition), as in many prior ZK protocols (e.g., the RZK one of [CGGM00], the CZK one of [DNS98], and the ZK one of [FS89]). That is, $\mathcal{V}$'s public key contains a key for a trapdoor commitment scheme, and $\mathcal{P}$'s first-round commitments are with respect to that public key. If the simulator knows the trapdoor, then it can open the commitments any way it needs in the third round.

To ensure that the simulator knows the trapdoor, the CGGM protocol uses a three-round proof-of-knowledge subprotocol, with $\mathcal{V}$ proving to $\mathcal{P}$ knowledge of the trapdoor. This requires $\mathcal{V}$ to send two messages to $\mathcal{P}$. Because I want to have a *total* of only three rounds, I cannot use such a subprotocol—in three rounds $\mathcal{V}$ only sends one message to $\mathcal{P}$. I therefore use *non-interactive* ZK proofs of knowledge (NIZKPKs; see Appendix A for a definition). This, of course, requires $\mathcal{P}$ and $\mathcal{V}$ to agree on a shared random string σ.

It is because of the string σ that one cannot use the BPK model directly, and has to strengthen it with a counter. Let $cmax = U(n)$ be the bound on the number of times public key is used. During key generation, $\mathcal{V}$ generates $cmax$ random strings $\sigma_1, \ldots, \sigma_{cmax}$, and commits to each one of them using non-interactive hash-based commitments, also recalled in Appendix A (to make the public key length independent of $cmax$, the resulting commitments are then put into a Merkle tree). In its first message, $\mathcal{P}$ sends a fresh random string $\sigma_{\mathcal{P}}$, and in its message $\mathcal{V}$ decommits σ_c (where c is the current counter value) and provides the NIZKPK proof with respect to $\sigma = \sigma_{\mathcal{P}} \oplus \sigma_c$.

The RZK simulator, after seeing the value of σ_c, can rewind the verifier and choose $\sigma_{\mathcal{P}}$ so that $\sigma = \sigma_{\mathcal{P}} \oplus \sigma_c$ allows it to extract the trapdoor from the NIZKPK. Of course, there is nothing to prevent a malicious verifier $\mathcal{V}^*$ from choosing a value of c after seeing $\sigma_{\mathcal{P}}$; but because the number of choices for $\mathcal{V}^*$ is only polynomial, the simulator has an inverse polynomial probability of guessing c correctly.

One question still remains unresolved: how to ensure that a malicious verifier $\mathcal{V}^*$ does not ask $\mathcal{P}$ multiple different queries for the same recoloring of the graph? If $\mathcal{V}^*$ resets $\mathcal{P}$, then it will get the same committed recolorings in the first round; if it can then ask a different set of queries, then it will gain a lot of information about the coloring of the graph (eventually even recovering the entire coloring). To prevent this, the CGGM protocol makes the verifier commit to its queries before it receives any information from $\mathcal{P}$. The current protocol, however, cannot afford to do that, because it only has three rounds. Instead, during key generation the verifier commits (using hash-based commitments) to a seed *PRFKey* for a pseudorandom function PRF, and adds the commitment to the public key. The verifier's queries are then computed using PRF$(\mathit{PRFKey}, \cdot)$ applied to the relevant information received from $\mathcal{P}$ in the first round and the counter value c. To prove to $\mathcal{P}$ that they are indeed computed correctly, the verifier has to include, in its NIZKPK, proofs of knowledge of *PRFKey* that leads to such queries and knowledge of the decommitment to *PRFKey*.

A Few More Technical Details. All the techniques used in this protocol are recalled in Appendix A. Here, let me briefly recall the notation used:

- The NIZKPK with security parameter n requires a shared random string σ of length $\mathrm{NI}\sigma\mathrm{Len}(n)$. The proof is computed by NIP and verified by NIV. The shared string and the proof together can by simulated by NIS.
- The non-interactive hash-based commitment scheme HC uses algorithms HCCom for commitment and HCVer for verification. The commitment is only computationally binding, and can be opened arbitrarily by an exponential-time algorithm HCFake.
- The trapdoor commitment scheme TC consists of the key generation algorithm TCGen (which generates keys *TCSK* and *TCPK*), the key verification algorithm TCKeyVer (which, looking at *TCPK*, can check whether there really does exist a trapdoor), the commitment algorithm TCCom and the verification algorithm TCVer. The commitments are only computationally binding; moreover, using the trapdoor, any commitment can be opened to any value, using a polynomial-time algorithm TCFake.
- The pseudorandom function PRF uses the seed *PRFKey*.

In the current protocol, just like in the CGGM protocol, all probabilistic choices of the prover are generated as a pseudorandom function of the input. (This is indeed the first step towards resettability, as it reduces the advantages of resetting the prover with the same random tape.) Because the prover makes no probabilistic choices in its second step, verifier's message need not be included in the input to the pseudorandom function.

To ensure soundness and avoid problems with malleability of $\mathcal{V}$'s commitments, I use complexity leveraging in a way similar to the CGGM protocol. That is, there will be two polynomially-related security parameters: n for all the components except the hash-based commitment scheme HC, and $k = n^\epsilon$ for HC.

This will ensure that any algorithm that is endowed with a subroutine for breaking HC commitments, but is polynomial-time otherwise, is still unable (simply by virtue of its running time) of breaking any other of the components. This property will be used in the proof of soundness.

The constant ϵ is chosen in the following way. If the protocol uses a trapdoor commitment scheme TC with soundness constant α_1, an NIZKPK system $(\mathrm{NIP}, \mathrm{NIV})$ with zero-knowledgeness constant α_2, and a pseudorandom function PRF with pseudorandomness constant α_3, the set $\epsilon < \min(\alpha_1, \alpha_2, \alpha_3)$.

The Full Description. The complete details of $\mathcal{P}$ and $\mathcal{V}$ are given below.

Key Generation Algorithm for U-bounded Verifier $\mathcal{V}$

System Parameter:
A polynomial U

Security Parameter:
1^n

Procedure:

```
1. Let $cmax = U(n)$.
2. Generate random strings $\sigma_1, \ldots, \sigma_{cmax}$ of length $\mathrm{NI}\sigma\mathrm{Len}(n)$ each.
   (Note: to save secret key length, the strings $\sigma_c$ can be generated using a
   pseudorandom function of $c$, whose short seed can be made part of the secret key).
```

3. Let $k = n^{\epsilon}$.
4. Commit to each σ_c using $(\sigma Com_c, \sigma Decom_c) \overset{R}{\leftarrow} \text{HCCom}(1^k, \sigma_c)$.
5. Combine the values σCom_c into a single Merkle tree with root R.
   (Note: If the values σ_c's are generated via a PRF to save on secret key length, then also the values σCom_c, the resulting Merkle tree, etc. can be computed efficiently in space logarithmic in $cmax$.)
6. Generate a random string $PRFKey$ of length n.
7. Commit to the $PRFKey$ using
   $(PRFKeyCom, PRFKeyDecom) \overset{R}{\leftarrow} \text{HCCom}(1^n, PRFKey)$.
8. Generate keys for trapdoor commitment scheme: $(TCPK, TCSK) \overset{R}{\leftarrow} \text{TCGen}(1^n)$.

Output:

$PK = (R, PRFKeyCom, TCPK)$
$SK = (\{(\sigma_c, \sigma Decom_c)\}_{c=1}^{cmax}, (PRFKey, PRFKeyDecom), TCSK)$.

Protocol $(\mathcal{P}, \mathcal{V})$

Public File:
A collection F of records (id, PK_{id}), where PK_{id} is allegedly the output of the Key Generation Algorithm above

Common Inputs:
A graph $G = (V, E)$, and a security parameter 1^n

$\mathcal{P}$ **Private Input:**
A valid coloring of G, $col : V \to \{0, 1, 2\}$; $\mathcal{V}$'s id and the file F; a random string ω

$\mathcal{V}$ **Private Input:**
A secret key SK, a counter value c, and a bound $cmax$.

$\mathcal{P}$ **Step One :**

1. Using the random string ω as a seed for PRF, generate a sufficiently long "random" string from the input to be used in the remaining computation.
2. Find PK_{id} in F; let $PK_{id} = (R, PRFKeyCom, TCPK)$
   (if more than one PK_{id} exist in F, use the alphabetically first one).
3. Verify $TCPK$ by invoking $\text{TCKeyVer}(1^n, TCPK)$.
4. Let $\sigma_{\mathcal{P}}$ be a random string of length $\text{NI}\sigma\text{Len}(n)$.
5. Commit to random recolorings of the graph G as follows.
   Let $\pi_1, \ldots, \pi_n$ be random permutations on $\{0, 1, 2\}$.
   For all i $(1 \le i \le n)$ and $v \in V$, commit to $\pi_i(col(v))$ by computing
   $(cCom_{i,v}, cDecom_{i,v}) \overset{R}{\leftarrow} \text{TCCom}(TCPK, \pi_i(col(v)))$.
6. If all the verifications hold, send $\sigma_{\mathcal{P}}$ and $\{cCom_{i,v}\}_{1 \le i \le n, v \in V}$ to $\mathcal{V}$.

$\mathcal{V}$ **Step One:**

1. Increment c and check that it is no greater than $cmax$.
2. For each j $(1 \le j \le n)$, compute a challenge edge $e_j \in E$ by applying PRF to the counter value c, j and the commitments received from $\mathcal{P}$:
   $e_j = \text{PRF}(PRFKey, c \circ j \circ \{cCom_{i,v}\}_{1 \le i \le n, v \in V})$
3. Let $\sigma = \sigma_{\mathcal{P}} \oplus \sigma_c$. Compute a NIZKPK proof Π using NIP on σ and the following statement:
   "$\exists$ key K for PRF that generated the challenge edges $\{e_j\}_{1 \le j \le n}$;
   $\exists$ decommitment D s. t. $\text{HCVer}(1^n, PRFKeyCom, K, D) = \text{YES}$;

$\exists$ secret key S corresponding to the public key $TCPK$."
(Note: this can computed efficiently because $\mathcal{V}$ knows witnesses
$PRFKey$ for K, $PRFKeyDecom$ for D, and $TCSK$ for S).

4. Send c, σ_c, σCom_c together with its authenticating path in the Merkle tree, $\sigma Decom_c$, Π and $\{e_j\}_{1\leq j\leq n}$ to $\mathcal{P}$.

$\mathcal{P}$ **Step Two:**

1. Verify the authenticating path of σCom_c in the Merkle tree
2. Verify that $\mathrm{HCVer}(1^k, \sigma_c, \sigma Com_c, \sigma Decom_c) = \mathrm{YES}$.
3. Let $\sigma = \sigma_P \oplus \sigma_c$. Verify Π using NIV.
4. If all the verifications hold, for each $e_j = (v_j^0, v_j^1)$ and $b \in \{0,1\}$, send $c_j^b = \pi_j(col(v_j^b))$ and $cDecom_{j,v_j^b}$ to $\mathcal{V}$.

$\mathcal{V}$ **Step Two:**

1. Verify that, for all j $(1 \leq j \leq n)$, and for all $b \in \{0,1\}$ $\mathrm{TCVer}(TCPK, cCom_{j,v_j^b}, c_j^b, cDecom_{j,v_j^b}) = \mathrm{YES}$.
2. Verify that for all j $(1 \leq j \leq n)$, $c_j^0 \neq c_j^1$.
3. If all the verifications hold, accept. Else reject.

As usual, completeness is easily verified. Concurrent soundness and black-box resettable zero-knowledgeness are shown below.

Soundness. First, I consider the case of a malicious prover $\mathcal{P}^*$ who interacts with $\mathcal{V}$ only once. The case of a concurrent malicious prover is considered afterwards.

Suppose G is a graph that is not 3-colorable, and $\mathcal{P}^*$ is a circuit of size $t < 2^k$ that can make $\mathcal{V}$ accept G with probability $p > 1/2^k$. Then, I shall construct a small circuit A that receives $TCPK$ as input, and, using $\mathcal{P}^*$, will output two trapdoor decommitments for the same TC commitment. The size of A will be $\mathrm{poly}(n) \cdot t \cdot 2^k/\mathrm{poly}(p)$. Thus, A will violate the soundness of TC, because its size is less (for a sufficiently large n) than $2^{n^{\alpha_1}}$ allowed by the soundness property of TC (recall in fact that $k = n^\epsilon$ and $\epsilon < \alpha_1$).

A is constructed as follows. It receives as input a public key $TCPK$ for TC generated by $\mathrm{TCGen}(1^n)$. A then generates PK as if it were the public key of the specified honest verifier $\mathcal{V}$, using the $\mathcal{V}$'s key generation procedure with the exception of step 8, for which it simply uses $TCPK$. Note that A knows all the components of corresponding secret key of $\mathcal{V}$, with the exception of $TCSK$. A selects an identity id and creates a file F to contain the single record (id, PK) (or embeds it into a larger such file containing other identities and public keys, but honestly generated).

A will now run $\mathcal{P}^*$ multiple times with inputs F and id (G is already known to $\mathcal{P}^*$), each time with the same random tape. Thus, each time, $\mathcal{P}^*$ will send the same set of strings $\sigma_\mathcal{P}$ and $\{cCom_{i,v}\}_{1\leq i\leq n, v\in V}$. The goal, each time, is to allow A to respond with a different random set of challenges $\{e_j'\}_{1\leq j\leq n}$. Then, after an expected number of tries that is inversely polynomial in p, there will exist a recoloring i and a node v such that $cCom_{i,v}$ has been opened by $\mathcal{P}^*$ in two different ways. That is, there will be a "break" of the commitment scheme TC.

Therefore, all there remains to be shown is *how* A can ask a different random set of challenges, despite the fact that it has committed to $\mathcal{V}$'s $PRFKey$ in PK. Recall that honest $\mathcal{V}$ executes the protocol at most $cmax$ time, and that the current value of $\mathcal{V}$'s counter will be known to P^*. If P^* has such an overall success probability p of proving G 3-colorable, then there exists a value of $\mathcal{V}$'s counter for which the success probability of $\mathcal{P}^*$ is at least p. Let c be such a value. Because of A's non-uniformity, I can assume A "knows" c.

To issue a set of (different) random challenges in response to the same first message of $\mathcal{P}^*$, A uses the NIZKPK simulator NIS as follows. First, A selects a set of random challenges $\{e'_j\}_{1\leq j\leq n}$. Second, it invokes NIS to obtain a "good looking proof" σ' and Π' for the following statement Σ:

$\Sigma =$"$\exists$ key K for PRF that generated the challenge edges $\{e'_j\}_{1\leq j\leq n}$;
$\exists$ decommitment D s. t. $\mathrm{HCVer}(1^n, \mathit{PRFKeyCom}, K, D) = \mathrm{YES}$;
$\exists$ secret key S corresponding to the public key TCPK."

(Note that Σ is potentially false, because it may be the case that no such K exists at all; I address this below.) Third, A sets $\tau = \sigma' \oplus \sigma_{\mathcal{P}}$. Fourth, A comes up with a decommitment $\tau\mathit{Decom}$ that decommits $\sigma\mathit{Com}_c$ (the commitment to the c-th shared string computed during key generation) to τ rather than the originally committed σ_c. This can be done by implementing HCFake by means of a (sub)circuit of size $\mathrm{poly}(k)2^k$. Fifth, A sends $\tau, \sigma\mathit{Com}_c$ together with its authenticating path in the Merkle tree (A knows that path from key generation), $\tau\mathit{Decom}$, Π' and $\{e'_j\}_{1\leq j\leq n}$ to $\mathcal{P}^*$.

Thus, all that's left to show is that $\mathcal{P}^*$ will behave the same way as it would for the true verifier $\mathcal{V}$, even though it received random, rather than pseudorandom, challenges, together with a faked decommitment and a simulated proof of a potentially false statement Σ. This is done by a properly constructed hybrid argument that relies on the zero-knowledgeness of $(\mathrm{NIP}, \mathrm{NIV})$, the pseudorandomness of PRF and the statistical secrecy and breakability of HC.

First, note that random $\{e'_j\}_{1\leq j\leq n}$ cannot be distinguished from pseudorandomly generated $\{e'_j\}_{1\leq j\leq n}$ (without knowledge of PRFKey): otherwise, the pseudorandomness of PRF would be violated. Moreover, this holds even in the presence of $\mathit{PRFKeyCom}$, because $\mathit{PRFKeyCom}$ is statistically secret, and thus reveals a negligible amount of information about PRFKey. Thus, the tuple $(\mathit{PRFKeyCom}, \{e'_j\}_{1\leq j\leq n}, \sigma', \Pi')$ is indistinguishable from the tuple $(\mathit{PRFKeyCom}, \{e_j\}_{1\leq j\leq n}, \sigma'', \Pi'')$, where the challenge edges $\{e_j\}_{1\leq j\leq n}$ are produced by the true PRF with the true committed-to PRFKey, and σ'', Π'' are produced by NIS. This, in turn, by zero-knowledgeness is indistinguishable from $(\mathit{PRFKeyCom}, \{e_j\}_{1\leq j\leq n}, \sigma, \Pi)$, with the pseudorandomly generated $\{e_j\}_{1\leq j\leq n}$, a truly random σ and Π honestly generated by NIP. By a hybrid argument, therefore, the tuple $(\mathit{PRFKeyCom}, \{e_j\}_{1\leq j\leq n}, \sigma, \Pi)$ is indistinguishable from the tuple $(\mathit{PRFKeyCom}, \{e'_j\}_{1\leq j\leq n}, \tau, \Pi')$. Of course, if one replaces σ by the pair $(\sigma_{\mathcal{P}}, \tau = \sigma \oplus \sigma_{\mathcal{P}})$ and σ' by the pair $(\sigma_{\mathcal{P}}, \sigma_c = \sigma \oplus \sigma_{\mathcal{P}})$, the statement still holds. Moreover, it holds in the presence of $\sigma\mathit{Com}_c$, because the commitment to σ_c is statistically secret (and thus is almost equally as likely to be a commitment to τ). The authenticating path of $\sigma\mathit{Com}_c$ in the Merkle tree is just a (randomized) function of $\sigma\mathit{Com}_c$ and root R of the tree, and thus does not affect indistinguishability. Finally, note that this indistinguishability holds with respect to any distinguishing circuit of size $2^k\mathrm{poly}(n)$, because the zero-knowledgeness and pseudorandomness constants α_2 and α_3 are greater than ϵ. Therefore, indistinguishability holds even in the presence of the decommitment $\tau\mathit{Decom}$ or $\sigma\mathit{Decom}_c$, because this decommitment can be computed by such a circuit from $\sigma\mathit{Com}_c$ using HCFake.

Now let me consider the case of an s-concurrent malicious prover $\mathcal{P}^*$. A now has to simulate to $\mathcal{P}^*$ multiple interactions with $\mathcal{V}$. This is no harder than simulating one interaction, because A never needs to rewind $\mathcal{P}^*$ in order to perform a simulation that is indistinguishable from interacting with true $\mathcal{V}$. (Note the crucial difference between this protocol and the protocols of [CGGM00] and Section 3.2: in the latter case, the simulator needs to rewind $\mathcal{P}^*$ in order to simulate the proof of knowledge of the secret key, and hence only sequential simulation is possible).

A does need to rewind $\mathcal{P}$, however, in order to be able to ask it multiple challenges on the same commitment. To that end, A guesses a random j between 1 and $s(n)$, hoping that $\mathcal{P}^*$ will cheat in the j-th protocol with non-negligible probability. It then runs $\mathcal{P}^*$ multiple times with the same random tape, always answering all of its questions the same way until it's time to send the challenges for the j-th protocol, which it varies by the technique described above (of course, because the challenges in the j-th protocol vary, $\mathcal{P}^*$'s messages after this point can also vary; however, before this point $\mathcal{P}^*$'s messages are guaranteed to be the same). If j and other random guesses of A were "lucky," then $\mathcal{P}^*$ will cheat in j-th protocol multiple times, and A will be able to break the binding property of TC. The probability of A's being "lucky" is non-negligible, because j is chosen from among polynomially many (namely, $s(n)$) possibilities.

Resettable Zero-Knowledgeness. Let $\mathcal{V}^*$ be an (s,t)-resetting verifier. I will show how to construct the black-box simulator M as required by Definition 6.

As already proven in [CGGM00], resettability is such a strong capability of a malicious verifier, that it has nothing else to gain by interleaving its executions with the honest prover. Thus, assume that $\mathcal{V}^*$ executes with $\mathcal{P}$ only sequentially.

Recall that $\mathcal{V}^*$ runs in two stages. Then M operates as follows. First, M runs the first stage of $\mathcal{V}^*$ to obtain a public file F. Then, for every record (id, PK_{id}) in F, M remembers some information (whose meaning will be explained later on):

1. M remembers whether PK_{id} is "broken" or not

2. If PK_{id} is broken, M also remembers the value of $TCSK$

3. If PK_{id} is not broken, M also remembers a list of tuples $(c, \sigma_c, \sigma Com_c)$

Initially, every PK_{id} in F is marked as not broken, and the list of pairs for each record is empty.

Whenever $\mathcal{V}^*$ starts a new session for an id that is not broken and whose list of pairs is empty, M computes the "first prover message" as follows: it commits to arbitrary color values for graph G, and then selects $\sigma_{\mathcal{P}}$ at random. (Of course, if $\mathcal{V}^*$ dictates that M's random tape and inputs be equal to those in a prior interaction, M has no choice but to use the same first message as in that interaction.) When $\mathcal{V}^*$ responds with the verifier message, M takes $(c, \sigma_c, \sigma Com_c)$ from this message and adds it to the list of tuples maintained for PK_{id}. M then rewinds $\mathcal{V}^*$ to the beginning of $\mathcal{V}^*$'s second stage.

Whenever $\mathcal{V}^*$ starts a new session for an id that is not broken but whose list of pairs is non-empty, M randomly chooses a tuple $(c', \sigma_{c'}, \sigma Com_{c'})$ from the list of tuples for PK_{id}. M then uses the extractor of the non-interactive ZK proof of knowledge, NIExt_1, to obtain a shared string σ, and sets $\sigma_{\mathcal{P}} = \sigma \oplus \sigma_{c'}$. M then commits to arbitrary color values for graph G and sends the commitment and $\sigma_{\mathcal{P}}$ as the "first prover message" to $\mathcal{V}^*$. When $\mathcal{V}^*$ responds with the verifier message, M compares the counter value c included in this response to the value c' from the pair chosen above.

1. If $c = c'$, then it must be the case that $\sigma_c = \sigma_{c'}$. (Otherwise, if the commitment $\sigma Com_{c'}$ previously stored by M is equal to the commitment σCom_c included in $\mathcal{V}^*$'s response, σ_c and $\sigma_{c'}$ have been easily found, so as to violate the soundness of HC; and if the $\sigma Com_c \neq \sigma Com'_c$, then a collision has been easily found in the Merkle

tree). Thus, the string Π, also included in the response of V^*, is an NIZK proof of knowledge with respect to the string σ output by NIExt_1. Therefore, M can use NIExt_2 to extract a witness $TCSK$ for the secret key of the commitment scheme. In this case, PK_{id} is marked as broken and M remembers $TCSK$.

2. If $c \neq c'$, then M has learned a potentially new tuple $(c, \sigma_c, \sigma Com_c)$, which it remembers in its list of pairs for PK_{id}.

M then rewinds $\mathcal{V}^*$ to the beginning of $\mathcal{V}^*$'s second stage.

Whenever $\mathcal{V}^*$ starts a new session for an *id* that is broken, M can always simulate $\mathcal{P}$'s behavior because M knows the trapdoor to the commitment scheme. Thus, it can commit to arbitrary color values in its first message, and then decommit in its second message so that they look like a valid response to $\mathcal{V}^*$'s challenge edges.

The expected running time of M is polynomial, because the expected number of rewinds before M breaks a given PK_{id} is polynomial in *cmax* and inverse polynomial in the frequency with which $\mathcal{V}^*$ uses *id*.

It remains to show that $\mathcal{V}^*$ cannot ask for two different sets of challenge edges for the same first message of M (if it could, then, unless M knows the correct 3-coloring of the graph, it may be unable to faithfully simulate the decommitments). However, if $\mathcal{V}^*$ has a non-negligible probability of doing so, then one can build a machine ADV to violate the soundness of HC in polynomial time with non-negligible probability, as follows.

ADV guesses, at random, for what instance of $\mathcal{P}$ the machine $\mathcal{V}^*$ will first give two different sets of challenges on the same first message. A also guesses, at random, the counter values c_1 and c_2 that $\mathcal{V}^*$ will use in these two cases. A then attempts to find out σ_{c_1} and σ_{c_2} by using the same technique as M. A then runs the second stage of $\mathcal{V}^*$ two more times: once to extract a witness K for *PRFKey* and its decommitment D in the first case, and the other to extract a witness K' for *PRFKey* and its decommitment D' in the second case (this witness extraction is done the same way as M). $K \neq K'$ and D and D' are valid decommitments, which violates soundness of HC. ■

Appendix A

Tools

In this chapter, I recall the notation, the definitions and the constructions that are used throughout this work.

A.1 Probabilistic Notation

(The following is taken verbatim from [BDMP91] and [GMR88].) If $A(\cdot)$ is an algorithm, then for any input x, the notation "$A(x)$" refers to the probability space that assigns to the string σ the probability that A, on input x, outputs σ. The set of strings having a positive probability in $A(x)$ will be denoted by "$\{A(x)\}$". If S is a probability space, then "$x \overset{R}{\leftarrow} S$" denotes the algorithm which assigns to x an element randomly selected according to S. If F is a finite set, then the notation "$x \overset{R}{\leftarrow} F$" denotes the algorithm that chooses x uniformly from F.

If p is a predicate, the notation $\text{PROB}[x \overset{R}{\leftarrow} S; y \overset{R}{\leftarrow} T; \cdots : p(x, y, \cdots)]$ denotes the probability that $p(x, y, \cdots)$ will be true after the ordered execution of the algorithms $x \overset{R}{\leftarrow} S$; $y \overset{R}{\leftarrow} T; \cdots$. The notation $[x \overset{R}{\leftarrow} S; y \overset{R}{\leftarrow} T; \cdots : (x, y, \cdots)]$ denotes the probability space over $\{(x, y, \cdots)\}$ generated by the ordered execution of the algorithms $x \overset{R}{\leftarrow} S$, $y \overset{R}{\leftarrow} T, \cdots$.

A.2 Pseudorandom Functions

A pseudorandom function family, introduced by Goldreich, Goldwasser and Micali [GGM86] is a keyed family of efficiently computable functions, such that a function picked at random from the family is indistinguishable (via oracle access) from a truly random function with the same domain and range. More formally, let $\text{PRF}(\cdot, \cdot) : \{0,1\}^n \times \{0,1\}^* \to \{0,1\}^n$ be an efficiently computable function. The definition below is quite standard, except that it requires security against subexponentially strong adversaries.

Definition 7 A function PRF is a *pseudorandom function* if $\exists\ \alpha > 0$ such that for all sufficiently large n and all 2^{n^α}-gate adversaries ADV, the following difference is negligible in n:

$$\text{PROB}[\mathit{PRFKey} \overset{R}{\leftarrow} \{0,1\}^n : \text{ADV}^{\text{PRF}(\mathit{PRFKey},\cdot)} = 1] -$$
$$\text{PROB}[F \overset{R}{\leftarrow} (\{0,1\}^n)^{\{0,1\}^n \times \{0,1\}^*} : \text{ADV}^{F(\cdot)} = 1]$$

The value α is the *pseudorandomness constant.*

Pseudorandom functions can be constructed based on a variety of assumption. The reader is referred to [GGM86, NR97] (and references therein) for details.

A.3 Non-Interactive Zero-Knowledge Proofs

NIZK Proofs of Membership

Non-interactive zero-knowledge (NIZK) proofs for any language $L \in \mathrm{NP}$ were put forward and exemplified in [BFM88, BDMP91]. Ordinary ZK proofs rely on interaction. NIZK proofs replace interaction with a random *shared string*, σ, that enters the view of the verifier that a simulator must reproduce. Whenever the security parameter is 1^n, σ's length is $\mathrm{NI}\sigma\mathrm{Len}(n)$, where $\mathrm{NI}\sigma\mathrm{Len}$ is a fixed, positive polynomial.

Let me quickly recall their definition, adapted for polynomial-time provers.

Definition 8 Let NIP (for non-interactive prover) and NIV (for non-interactive verifier) be two probabilistic polynomial-time algorithms, and let $\mathrm{NI}\sigma\mathrm{Len}$ be a positive polynomial. A pair $(\mathrm{NIP}, \mathrm{NIV})$ is a NIZK argument system for an NP-language L if

1. *Completeness.* $\forall\ x \in L$ of length n, σ of length $\mathrm{NI}\sigma\mathrm{Len}(n)$, and NP-witness y for x,

$$\mathrm{PROB}[\Pi \overset{R}{\leftarrow} \mathrm{NIP}(\sigma, x, y) : \mathrm{NIV}(\sigma, x, \Pi) = \mathrm{YES}] = 1.$$

2. *Soundness.* $\forall\ x \in L$ of length n,

$$\mathrm{PROB}[\sigma \overset{R}{\leftarrow} \{0,1\}^{\mathrm{NI}\sigma\mathrm{Len}(n)} : \exists\ \Pi \text{ s. t. } \mathrm{NIV}(\sigma, x, \Pi) = \mathrm{YES}]$$

is negligible in n.

3. *Zero-Knowledgeness.* There exists a probabilistic polynomial-time simulator NIS such that, $\forall$ sufficiently large n, $\forall\ x$ of length n and NP-witness y for x, the following two distributions are indistinguishable by any polynomial-time adversary:

$$[(\sigma', \Pi') \overset{R}{\leftarrow} \mathrm{NIS}(x) : (\sigma', \Pi')] \text{ and } [\sigma \overset{R}{\leftarrow} \{0,1\}^{\mathrm{NI}\sigma\mathrm{Len}(n)} ; \Pi \overset{R}{\leftarrow} \mathrm{NIP}(\sigma, x, y) : (\sigma, \Pi)]$$

The authors of [BDMP91] show that non-interactive zero-knowledge proofs exist for all NP languages under the quadratic residuosity assumption. The authors of [FLS99] show the same under a general assumptions: namely, that certified trapdoor permutations exist (a family of trapdoor permutations is *certified* if it is easy to tell that a given function belongs to the family). The reader is referred to these papers for details.

NIZK Proofs of Knowledge

In [DP92], De Santis and Persiano propose to add a *proof of knowledge* property to NIZK. Let $R \subseteq \{0,1\}^* \times \{0,1\}^*$ be a polynomial-time relation (i.e., given a pair of strings (x, y), it is possible to check in time polynomial in $|x|$ whether $(x, y) \in R$). L be the NP language corresponding to R ($L = \{x : \exists\ y \text{ s.t. } (x, y) \in R\}$). Let $(\mathrm{NIP}, \mathrm{NIV})$ be a NIZK proof system for L. An *extractor* is a probabilistic polynomial-time TM that runs in two stages: in stage one, on input 1^n, it outputs a string σ of length $\mathrm{NI}\sigma\mathrm{Len}(n)$ (and saves any information it wants to use in stage two); in stage two, on input x of length n and a proof Π for x relative to shared string σ, it tries to find a witness y for x.

Definition 9 An NIZK argument (NIP, NIV) is a NIZKPK if there exists an extractor $\text{NIExt} = (\text{NIExt}_1, \text{NIExt}_2)$ such that, for all probabilistic polynomial-time malicious provers NIP^*, for all constants $a > 0$, for all sufficiently large n and for all x,

$$\text{PROB}[(\sigma, \mathit{state}) = \text{NIExt}_1(1^n) \; ; \; \Pi = \text{NIP}^*(\sigma, x) \; ; \; y = \text{NIExt}_2(\mathit{state}, x, \Pi) : \\ (x, y) \in R] \geq p_{n,x}(1 - n^{-a}),$$

where $p_{n,x} = \text{PROB}[\sigma \overset{R}{\leftarrow} \{0,1\}^n ; \Pi = \text{NIP}^*(\sigma, x) : \text{NIV}(\sigma, x, \Pi) = 1]$.

For the application in this paper, I need NIZKPKs that are zero-knowledge even against subexponential-time distinguishers. That is, I strengthen the zero-knowledgeness property as follows: the distributions generated by the true prover and the simulator are indistinguishable by any 2^{n^α}-gate adversary, where α is the *zero-knowledgeness constant.*

The authors of [DP92] show that NIZKPKs exist for all polynomial-time relations under the RSA assumption. Furthermore, the results of [DDP00] (combined with those of [FLS99]) show the same under more general assumptions: that dense public-key cryptosystems and certified trapdoor permutations exist. They also present constructions secure under the specific assumptions of factoring Blum integers or decision Diffie-Hellman. Because I need NIZKPKs to be secure against subexponentially strong adversaries, I need subexponentially strong versions of these assumptions. The reader is referred to these papers for details.

A.4 Verifiable Random Functions

A family of verifiable random functions (VRFs), as proposed in [MRV99], is essentially a pseudorandom function family with the additional property that the correct value of a function on an input can not only be computed by the owner of the seed, but also *proven* to be the unique correct value. The proof can be verified by anyone who knows the public key corresponding to the seed.

More precisely, a VRF is a quadruple of efficiently computable functions. The function VRFGen generates a key pair $(\mathit{VRFSK}, \mathit{VRFPK})$. The function $\text{VRFEval}(\mathit{VRFSK}, x)$ computes the pseudorandom output v; the function $\text{VRFProve}(\mathit{VRFSK}, x)$ computes pf_x, the proof that v is correct. This proof can be verified by anyone who knows the VRFPK by using $\text{VRFVer}(\mathit{VRFPK}, x, v, \mathit{pf}_x)$; moreover, no matter how maliciously VRFPK is constructed, for each x, there exists at most one v for which a valid proof pf_x exists. The pseudorandomness requirement states that, for all the points for which no proof has been provided, the function $\text{VRFEval}(\mathit{VRFSK}, \cdot)$ remains indistinguishable from random. The following formal definition is almost verbatim from [MRV99], adapted for subexponentially-strong distinguishers.

Definition 10 Let VRFGen, VRFEval, VRFProve, and VRFVer be polynomial-time algorithms (the first and last are probabilistic, and the middle two are deterministic). Let $a: \mathbb{N} \to \mathbb{N} \cup \{*\}$ and $b: \mathbb{N} \to \mathbb{N}$ be any two functions that are computable in time $\text{poly}(n)$ and bounded by a polynomial in n (except when a takes on the value $*$).

A quadruple (VRFGen, VRFEval, VRFProve, VRFVer) is a *verifiable pseudorandom function (VRF)* with *input length* $a(n)$,[1] and *output length* $b(n)$ if the following properties hold:

[1] When $a(n)$ takes the value $*$, it means that the VRF is defined for inputs of all lengths. Specifically, if $a(n) = *$, then $\{0,1\}^{a(n)}$ is to be interpreted as the set of all binary strings, as usual.

1. The following conditions hold with probability $1 - 2^{-\Omega(n)}$ over $(\mathit{VRFPK}, \mathit{VRFSK}) \overset{R}{\leftarrow} \mathrm{VRFGen}(1^n)$:

   (a) (Domain-Range Correctness): for all $x \in \{0,1\}^{a(n)}$, $\mathrm{VRFEval}(\mathit{VRFSK}, x) \in \{0,1\}^{b(n)}$.

   (b) (Complete Provability): for all $x \in \{0,1\}^{a(k)}$, if $v = \mathrm{VRFEval}(\mathit{VRFSK}, x)$ and $\mathit{pf} = \mathrm{VRFProve}(\mathit{VRFSK}, x)$, then

   $$\mathrm{PROB}[(\mathrm{VRFVer}(\mathit{VRFPK}, x, v, \mathit{pf}) = \mathrm{YES}] > 1 - 2^{-\Omega(k)}$$

   (this probability is over the coin tosses of VRFVer).

2. (Unique Provability): For every VRFPK, x, v_1, v_2, pf_1, and pf_2 such that $v_1 \neq v_2$, the following holds for either $i = 1$ or $i = 2$:

   $$\mathrm{PROB}[\mathrm{VRFVer}(\mathit{VRFPK}, x, v_i, \mathit{pf}_i) = \mathrm{YES}] < 2^{-\Omega(k)}$$

   (this probability is also over the coin tosses of VRFVer).

3. (Residual Pseudorandomness): Let $\alpha > 0$ be a constant. Let $T = (T_E, T_J)$ be any pair of algorithms such that $T_E(\cdot,\cdot)$ and $T_J(\cdot,\cdot,\cdot)$ run for a total of at most 2^{n^α} steps when their first input is 1^n. Then the probability that T succeeds in the following experiment is at most $1/2 + 1/2^{n^\alpha}$:

   (a) Run $\mathrm{VRFGen}(1^n)$ to obtain $(\mathit{VRFPK}, \mathit{VRFSK})$.

   (b) Run $T_E^{\mathrm{VRFEval}(\mathit{VRFSK},\cdot),\mathrm{VRFProve}(\mathit{VRFSK},\cdot)}(1^n, \mathit{VRFPK})$ to obtain (x, state).

   (c) Choose $r \overset{R}{\leftarrow} \{0,1\}$.

      i. if $r = 0$, let $v = \mathrm{VRFEval}(\mathit{VRFSK}, x)$.

      ii. if $r = 1$, choose $v \overset{R}{\leftarrow} \{0,1\}^{b(n)}$.

   (d) Run $T_J^{\mathrm{VRFEval}(\mathit{VRFSK},\cdot),\mathrm{VRFProve}(\mathit{VRFSK},\cdot)}(1^n, v, \mathit{state})$ to obtain guess.

   (e) $T = (T_E, T_J)$ *succeeds* if $x \in \{0,1\}^{a(n)}$, $\mathit{guess} = r$, and x was not asked by either T_E or T_J as a query to $\mathrm{VRFEval}(\mathit{VRFSK}, \cdot)$ or $\mathrm{VRFProve}(\mathit{VRFSK}, \cdot)$.

   The value α is the *pseudorandomness constant.*

The authors of [MRV99] show how to construct VRFs based on the following variant of the RSA assumption. (The reader is referred to that paper for details of the construction.) Let PRIMES_n be the set of the n-bit primes, and RSA_n be the set of composite integers that are the product of two primes of length $\lfloor (n-1)/2 \rfloor$.

The RSA' Subexponential Hardness Assumption: There exists a constant α such that, if A is any probabilistic algorithm which runs in time 2^{n^α} when its first input is 1^n, then,

$$\mathrm{PROB}[m \overset{R}{\leftarrow} \mathrm{RSA}_n \,;\, x \overset{R}{\leftarrow} \mathbb{Z}_m^* \,;\, p \overset{R}{\leftarrow} \mathrm{PRIMES}_{n+1} \,;\, y \overset{R}{\leftarrow} A(1^n, m, x, p) : \\ y^p = x \pmod{m}] < 1/2^{n^\alpha}.$$

A.5 Trapdoor Commitment Schemes

In this section I present trapdoor commitment schemes that are secure against subexponentially strong adversaries (satisfying an additional key-verification property).[2]

Informally, a trapdoor commitment scheme consists of a quintuple of algorithms. Algorithm TCGen generates a pair of matching public and secret keys. Algorithm TCCom takes two inputs, a value v to be committed to and a public key, and outputs a pair, (c, d), of commitment and decommitment values. Algorithm TCVer takes the public key and c, v, d and checks whether c was indeed a commitment to v.

What makes the commitment *computationally binding* is that without knowledge of the secret key, it is computationally hard to come up with a single commitment c and two different decommitments d_1 and d_2 for two different values v_1 and v_2 such that TCVer would accept both c, v_1, d_1 and c, v_2, d_2. What makes it *perfectly secret* is that the value c yields no information about the value v. Moreover, this has to hold even if the public key is chosen adversarially. Thus, there has to be an algorithm TCKeyVer that takes a public key as input and verifies whether the resulting commitment scheme is indeed perfectly secret. (More generally, TCKeyVer can be an interactive protocol between the committer and the key generator, rather than an algorithm; however, for this paper, the more restricted view suffices.)

Perfect secrecy ensures that, information-theoretically, any commitment c can be decommitted arbitrarily: for any given commitment c to a value v_1, and any value v_2, there exists d_2 such that TCVer accepts c, v_2, d_2 and the public key (indeed, if for some v_2 such d_2 did not exist, then c would leak information about the actual committed value v_1). The trapdoor property makes this assurance computational: knowing the secret key enables one to decommit arbitrarily through the use of the TCFake algorithm.

Definition 11 A *Trapdoor Commitment Scheme* (TC) is a quintuple of probabilistic polynomial-time algorithms TCGen, TCCom, TCVer, TCKeyVer and TCFake, such that

1. *Completeness.* $\forall n$, $\forall v$,

$$\begin{aligned} &\text{PROB}[(\mathit{TCPK}, \mathit{TCSK}) \overset{R}{\leftarrow} \text{TCGen}(1^n)\ ;\ (c, d) \overset{R}{\leftarrow} \text{TCCom}(\mathit{TCPK}, v) : \\ &\qquad \text{TCKeyVer}(\mathit{TCPK}, 1^n) = \text{TCVer}(\mathit{TCPK}, c, v, d) = \text{YES}] = 1 \end{aligned}$$

2. *Computational Soundness.* $\exists\ \alpha > 0$ such that for all sufficiently large n and for all 2^{n^α}-gate adversaries ADV

$$\begin{aligned} \text{PROB}[\quad & (\mathit{TCPK}, \mathit{TCSK}) \overset{R}{\leftarrow} \text{TCGen}(1^n)\ ; \\ & (c, v_1, v_2, d_1, d_2) \overset{R}{\leftarrow} \text{ADV}(1^n, \mathit{TCPK}) : \\ & \text{TCVer}(\mathit{TCPK}, c, v_1, d_1) = \text{YES and} \\ & \text{TCVer}(\mathit{TCPK}, c, v_2, d_2) = \text{YES and } v_1 \neq v_2] < 2^{-n^\alpha} \end{aligned}$$

   The value α is the *soundness constant.*

3. *Perfect Secrecy.* $\forall$ *TCPK* such that TCKeyVer$(\mathit{TCPK}, 1^n)$ = YES and $\forall v_1, v_2$ of equal length, the following two probability distributions are identical:

$$[(c_1, d_1) \overset{R}{\leftarrow} \text{TCCom}(\mathit{TCPK}, v_1) : c_1] \quad \text{and}$$

[2] I follow a similar discussion in [CGGM00] almost verbatim.

$$[(c_2, d_2) \overset{R}{\leftarrow} \text{TCCom}(\mathit{TCPK}, v_2) : c_2]$$

4. *Trapdoorness.* $\forall$ $(\mathit{TCPK}, \mathit{TCSK}) \in \{\text{TCGen}(1^n)\}$, $\forall v_1, v_2$ of equal length the following two probability distributions are identical:

$$\begin{aligned} [\quad & (c, d_1) \overset{R}{\leftarrow} \text{TCCom}(\mathit{TCPK}, v_1) \,; \\ & d_2' \overset{R}{\leftarrow} \text{TCFake}(\mathit{TCPK}, \mathit{TCSK}, c, v_1, d_1, v_2) \,:\, (c, d_2') \,] \quad \text{and} \\ [\quad & (c, d_2) \overset{R}{\leftarrow} \text{TCCom}(\mathit{TCPK}, v_2) \,:\, (c, d_2) \,] \end{aligned}$$

(In particular, the above states that faked commitments are correct: indeed, $d_2' \overset{R}{\leftarrow}$ TCFake$(\mathit{TCPK}, \mathit{TCSK}, c, v_1, d_1, v_2)$ implies that TCVer$(\mathit{TCPK}, c, v_2, d_2') =$ YES)

In this paper, I will also require that the relation $(\mathit{TCPK}, \mathit{TCSK})$ be polynomial-time; this is easy to satisfy by simply including the random string used in key generation into the secret key.

Such commitment schemes can be constructed, in particular, based on a subexponentially strong variant of the Discrete Logarithm assumption. The reader is referred to [BCC88] (where, in Section 6.1.2, it is called a DL-based "chameleon blob") for the construction.

A.6 Hash-Based Commitment Schemes

In addition to trapdoor commitment schemes, I also use non-trapdoor, non-interactive, computationally-binding commitment schemes (which, unlike trapdoor commitments, need not be secure against subexponentially strong adversaries). Because of the absence of the trapdoor requirement, these simpler commitment schemes can be implemented more efficiently if one replaces perfect secrecy by the essentially equally powerful property of *statistical secrecy* (i.e., even with infinite time one can get only a statistically negligible advantage in distinguishing the commitments of any two different values). In particular [DPP97, HM96] show how to commit to any value by just one evaluation of a collision-free hash function $H : \{0,1\}^* \to \{0,1\}^k$. To differentiate trapdoor commitments from these simpler ones, I shall call them *hash-based commitments.*

Though the trapdoor property does not hold, I still insist that, given any commitment and any value, it is possible in time 2^k to decommit to that value.

Definition 12 A *Hash-Based Commitment Scheme* (HC) is a pair of probabilistic polynomial-time algorithms HCCom, HCVer, along with the algorithm HCFake that runs in time 2^kpoly when its first input is 1^k and poly is some polynomial in the size of its input, such that

1. *Completeness.* $\forall k$, $\forall v$,

$$\text{PROB}[(c, d) \overset{R}{\leftarrow} \text{HCCom}(1^k, v) \,:\, \text{HCVer}(1^k, c, v, d) = \text{YES}] = 1$$

2. *Computational Soundness.* For all probabilistic polynomial-time machines ADV, and all sufficiently large k,

$$\begin{aligned} &\text{PROB}[(c, v_1, v_2, d_1, d_2) \overset{R}{\leftarrow} \text{ADV}(1^k) : \\ &\quad v_1 \neq v_2 \text{ and HCVer}(1^k, c, v_1, d_1) = \text{YES} = \text{HCVer}(1^k, c, v_2, d_2)] \end{aligned}$$

is negligible in k.

3. *Statistical Secrecy.* $\forall v_1, v_2$ of equal length, the statistical difference between the following two probability distribution is negligible in k:

$$[(c_1, d_1) \overset{R}{\leftarrow} \text{HCCom}(1^k, v_1) : c_1] \text{ and } [(c_2, d_2) \overset{R}{\leftarrow} \text{HCCom}(1^k, v_2) : c_2]$$

4. *Breakability.* $\forall v_1, v_2$ of equal length, the statistical difference between the following two probability distribution is negligible in k:
$[(c, d_1) \overset{R}{\leftarrow} \text{HCCom}(1^k, v_1)\ ;\ d_2' \overset{R}{\leftarrow} \text{HCFake}(1^k, c, v_1, d_1, v_2)\ :\ (c, d_2')]$ and
$[(c, d_2) \overset{R}{\leftarrow} \text{HCCom}(1^k, v_2)\ :\ (c, d_2)]$

The reader is referred to [DPP97, HM96] for the constructions of such schemes, which are based on the assumption that collisions-resistant hash functions exist.

A.7 Merkle Trees

The description below is almost verbatim from [Mic00].

Recall that a binary tree is a tree in which every node has at most two children, hereafter called the *0-child* and the *1-child.* A *Merkle tree* [Mer89] with security parameter n is a binary tree whose nodes store values, some of which are computed by means of a collision-free hash function $H : \{0,1\}^* \rightarrow \{0,1\}^n$ in a special manner. A leaf node can store any value, but each internal node should store a value that is the one-way hash of the concatenation of the values in its children. That is, if an internal node has a 0-child storing the value u and a 1-child storing a value v, then it stores the value $H(u \circ v)$. Thus, because H produces n-bit outputs, each internal node of a Merkle tree, including the root, stores an n-bit value. Except for the root value, each value stored in a node of a Merkle tree is said to be a 0-value, if it is stored in a node that is the 0-child of its parent, a 1-value otherwise.

The crucial property of a Merkle tree is that, unless one succeeds in finding a collision for H, *it is computationally hard to change any value in the tree (and, in particular, a value stored in a leaf node) without also changing the root value.* This property allows a party A to commit to L values, $v_1, \ldots, v_L$ (for simplicity assume that L is a power of 2 and let $d = \log L$), by means of a single n-bit value. That is, A stores value v_i in the i-th leaf of a full binary tree of depth d, and uses a collision-free hash function H to build a Merkle tree, thereby obtaining an n-bit value, R, stored in the root. This root value R "implicitly defines" what the L original values were. Assume in fact that, as some point in time, A gives R, but not the original values, to another party B. Then, whenever, at a later point in time, A wants to "prove" to B what the value of, say, v_i was, A may just reveal all L original values to B, so that B can recompute the Merkle tree and the verify that the newly computed root-value indeed equals R. More interestingly, A may "prove" what v_i was by revealing just $d+1$ (that is, just $1 + \log L$) values: v_i together with its *authenticating path*, that is, the values stored in the siblings of the nodes along the path from leaf i (included) to the root (excluded), $w_1, \ldots, w_d$. Party B verifies the received alleged leaf-value v_i and the received alleged authenticating path $w_1, \ldots, w_d$ as follows. She sets $u_1 = v_i$ and, letting $i_1, \ldots, i_d$ be the binary expansion of i, computes the values $u_2, \ldots, u_d$ as follows: if $i_j = 0$, she sets $u_{j+1} = H(w_j \circ u_j)$; else, she sets $u_{j+1} = H(u_j \circ w_j)$. Finally, B checks whether the computed n-bit value u_d equals R.

Bibliography

[ACM01] *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing*, Crete, Greece, 6–8 July 2001.

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37(2):156–189, October 1988.

[BDMP91] Manuel Blum, Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Non-interactive zero-knowledge. *SIAM Journal on Computing*, 20(6):1084–1118, December 1991.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pages 103–112, Chicago, Illinois, 2–4 May 1988.

[Blu86] Manuel Blum. How to prove a theorem so no one else can claim it. In *Proc. of the International Congress of Mathematicians, Berkeley, CA*, pages 1444–1451, 1986.

[Bra89] G. Brassard, editor. *Advances in Cryptology—CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*. Springer-Verlag, 1990, 20–24 August 1989.

[CG89] B. Chor and O. Goldreich. On the power of two-point based sampling. *Journal of Complexity*, 5:96–106, 1989.

[CGGM00] Ran Canetti, Oded Goldreich, Shafi Goldwasser, and Silvio Micali. Resettable zero-knowledge. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing*, Portland, Oregon, 21–23 May 2000. Updated version available at the Cryptology ePrint Archive, record 1999/022, `http://eprint.iacr.org/`.

[CKPR01] Ran Canetti, Joe Kilian, Erez Petrank, and Alon Rosen. Black-box concurrent zero-knowledge requires $\tilde{\Omega}(\log n)$ rounds. In ACM [ACM01].

[Dam00] Ivan Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In Bart Preneel, editor, *Advances in Cryptology—EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 418–430. Springer-Verlag, 14–18 May 2000.

[DDP00] Alfredo De Santis, Giovanni Di Crescenzo, and Giuseppe Persiano. Necessary and sufficient assumptions for non-interactive zero-knowledge proofs of knowledge for all np relations. In U. Montanari, J. D. P. Rolim, and E. Welzl, editors, *Automata Languages and Programming: 27th International Colloquim (ICALP 2000)*, volume 1853 of *Lecture Notes in Computer Science*, pages 451–462. Springer-Verlag, July 9–15 2000.

[DNS98] Cynthia Dwork, Moni Naor, and Amit Sahai. Concurrent zero knowledge. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, pages 409–418, Dallas, Texas, 23–26 May 1998.

[DP92] Alfredo De Santis and Giuseppe Persiano. Zero-knowledge proofs of knowledge without interaction. In *33rd Annual Symposium on Foundations of Computer Science*, pages 427–436, Pittsburgh, Pennsylvania, 24–27 October 1992. IEEE.

[DPP97] Ivan B. Damgård, Torben P. Pedersen, and Birgit Pfitzmann. On the existence of statistically hiding bit commitment schemes and fail-stop signatures. *Journal of Cryptology*, 10(3):163–194, Summer 1997.

[DS98] Cynthia Dwork and Amit Sahai. Concurrent zero-knowledge: Reducing the need for timing constraints. In Krawczyk [Kra98], pages 442–457.

[FLS99] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs under general assumptions. *SIAM Journal on Computing*, 29(1):1–28, 1999.

[FS89] Uriel Feige and Adi Shamir. Zero knowledge proofs of knowledge in two rounds. In Brassard [Bra89], pages 526–545.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, October 1986.

[GK96] Oded Goldreich and Hugo Krawczyk. On the composition of zero-knowledge proof systems. *SIAM Journal on Computing*, 25(1):169–192, February 1996.

[GL89] O. Goldreich and L. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing*, pages 25–32, Seattle, Washington, 15–17 May 1989.

[GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, April 1984.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. Knowledge complexity of interactive proofs. In *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, pages 291–304, Providence, Rhode Island, 6–8 May 1985.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18:186–208, 1989.

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(1):691–729, 1991.

[GO94] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7(1):1–32, 1994.

[HILL99] J. Håstad, R. Impagliazzo, L.A. Levin, and M. Luby. Construction of pseudo-random generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[HM96] Shai Halevi and Silvio Micali. Practical and provably-secure commitment schemes from collision-free hashing. In Neal Koblitz, editor, *Advances in Cryptology—CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 201–215. Springer-Verlag, 18–22 August 1996.

[HT98] Satoshi Hada and Toshiaki Tanaka. On the existence of 3-round zero-knowledge protocols. In Krawczyk [Kra98], pages 408–423.

[Jof74] A. Joffe. On a set of almost deterministic k-independent random variables. *Annals of Probability*, 2:161–162, 1974.

[KP01] Joe Kilian and Erez Petrank. Concurrent and resettable zero-knowledge in poly-logarithmic rounds. In ACM [ACM01].

[Kra98] Hugo Krawczyk, editor. *Advances in Cryptology—CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*. Springer-Verlag, 23–27 August 1998.

[Mer89] Ralph C. Merkle. A certified digital signature. In Brassard [Bra89], pages 218–238.

[Mic00] Silvio Micali. CS proofs. *SIAM Journal on Computing*, 30(4):1253–1298, 2000.

[MR01a] Silvio Micali and Leonid Reyzin. Min-round resettable zero knowledge in the public-key model. In Birgit Pfitzmann, editor, *Advances in Cryptology—EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 373–393. Springer-Verlag, 6–10 May 2001.

[MR01b] Silvio Micali and Leonid Reyzin. Soundness in the public-key model. In Joe Kilian, editor, *Advances in Cryptology—CRYPTO 2001*, Lecture Notes in Computer Science. Springer-Verlag, 19–23 August 2001.

[MRV99] Silvio Micali, Michael Rabin, and Salil Vadhan. Verifiable random functions. In *40th Annual Symposium on Foundations of Computer Science*, pages 120–130, New York, October 1999. IEEE.

[NR97] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *38th Annual Symposium on Foundations of Computer Science*, pages 458–467, Miami Beach, Florida, 20–22 October 1997. IEEE.

[RK99] Ransom Richardson and Joe Kilian. On the concurrent composition of zero-knowledge proofs. In Jacques Stern, editor, *Advances in Cryptology—EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 415–431. Springer-Verlag, 2–6 May 1999.

[Rom90] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the Twenty Second Annual ACM Symposium on Theory of Computing*, pages 387–394, Baltimore, Maryland, 14–16 May 1990.

[WC81] M.N. Wegman and J.L. Carter. New hash functions and their use in authentication and set equality. *Journal of Computer and System Sciences*, 22:265–279, 1981.

[Yao82] A. C. Yao. Theory and application of trapdoor functions. In *23rd Annual Symposium on Foundations of Computer Science*, pages 80–91, Chicago, Illinois, 3–5 November 1982. IEEE.