Black-Box Secure Multi-Party Computation: New Possibilities and Limitations

by

Xiao Liang

A dissertation submitted in partial satisfaction of

the requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate School

of the

Stony Brook University

August 2021

Abstract of the Dissertation

Black-Box Secure Multi-Party Computation: New Possibilities and Limitations

by

Xiao Liang

Doctor of Philosophy

in

Computer Science

Stony Brook University

2021

Secure multi-party computation (MPC) allows two or more mutually distrustful parties to compute any functionality without compromising the privacy of their inputs. It has been an important theme in this area to obtain constructions that make only *black-box* use of their cryptographic building blocks. Such constructions are usually preferable since their efficiency is not affected by the implementation details of the underlying primitives.

New Black-Box Constructions of MPC. We present new composable MPC protocols that makes only black-box access to lower-level primitives. In particular:

– We construct a black-box MPC protocol which remains secure under (a-priori) bounded-concurrent composition. Prior to our work, constructions of such protocols required non-black-box usage of cryptographic primitives; alternatively, black-box constructions could only be achieved for super-polynomial simulation based notions of security, which offers incomparable security guarantees.

  Our protocol has a constant number of rounds and relies on the existence of semi-honest oblivious transfers and collision-resistant hash functions. Previously, such protocols were not known even under sub-exponential assumptions.

– We present a black-box MPC protocol with $\widetilde{O}(\log \lambda)$ rounds achieving angel-based universal composability, or more precisely, composability with super-polynomial helpers. In this notion, both the simulator and the adversary are given access to an oracle called an *angel* that can perform some predefined super-polynomial time task. Angel-based security maintains the attractive properties of the universal composition framework while providing meaningful security guarantees in complex environments without having to trust anyone.

  Our construction is obtained under the minimal assumption of semi-honest oblivious transfer. Prior to our work, there exist black-box constructions in $\widetilde{O}(\log^2 \lambda)$ rounds and non-black-box constructions in $\widetilde{O}(\log \lambda)$ rounds. Thus, we close the gap between these two types of protocols.

Zero-Knowledge with Stronger Black-Box Properties. Zero-Knowledges Proofs are of great importance both as a building block for MPC and a stand-alone primitive. Next, we explore the (im)possibility of zero-knowledge proofs with a strong black-box property.

General-purpose zero-knowledge proofs inherently require the code of the relation to be proven. As shown by Rosulek (Crypto'12), zero-knowledge proofs for even simple statements, such as membership in the range of a one-way function, require non-black-box access. In this work, we propose an alternative approach to bypass Rosulek's impossibility result. Instead of asking for a ZK proof directly for the given one-way function f, we seek to construct a *new* one-way function F given only black-box access to f, *and* an associated ZK protocol for proving non-trivial statements, such as range membership, over its output. We say that F, along with its proof system, is a *proof-based* one-way function. We similarly define proof-based versions of other primitives, specifically pseudo-random generators and collision-resistant hash functions.

We show how to construct proof-based versions of each of the primitives mentioned above from their ordinary counterparts under mild but necessary restrictions over the input. More specifically,

- We first show that if the prover entirely chooses the input, then proof-based pseudo-random generators cannot be constructed from ordinary ones in a black-box manner, thus establishing that some restrictions over the input are necessary.
- We next present black-box constructions handling inputs of the form (x, r) where r is chosen uniformly by the verifier. This is similar to the restrictions in the widely used Goldreich-Levin theorem. The associated ZK proofs support range membership over the output as well as arbitrary predicates over prefixes of the input.

Our results open up the possibility that general-purpose ZK proofs for relations that require black-box access to the primitives above may be possible in the future without violating their black-box nature by instantiating them using proof-based primitives instead of ordinary ones.

Dedicated to

Ms. Fangfang Song,

who has always been, and will always be, my muse and Aphrodite.

Table of Contents

Acknowledgments

To a picky person like me, aesthetically appealing food and beverages can be necessities for research. On that account, thanks to the BLT from Northside Cafe in Berkeley, the Irresist-A-Bowls from Applebee's$^{®}$ in Stony Brook, the iced tea (unsweetened) from Brewed Awakening in Berkeley, and Caffè Americano all over the world.

Same thanks also go to J. S. Bach, since *The Well-Tempered Clavier* is as essential as food!

My wonderful Ph.D. journey began with the unbelievable luck of having Omkant Pandey as my advisor. I am deeply grateful for the endless patience and unparalleled amount of time he devoted to me, which greatly helped convert me from a layman of (not only cryptography but also) computer science to an independent researcher. His astute advice ranged from mathematical formalism to career/life navigation, which always turned out to be wise, although most of the time, I only managed to see it in hindsight. Aside from his great expertise and quick wit, I am most impressed by his sharp mind—over the years, it has become certain to me that his train of thought must be hailed as the *Train à Grande Vitesse*. I also appreciate his continuous efforts to convince me of the beauty of problem simplification and rescue me from my pathological obsession with mathematical details. However, to date, the former could not be more successful, and the latter could not be successful. Moreover, he shows tremendous tolerance for my fickle research interest and has always been there for timely support, which really made the past four years the most enjoyable time of my school days. Indeed, Omkant is the best mentor one could ever imagine.

I am also deeply indebted to Sanjam Garg for his extraordinary guidance at UC Berkeley. It is good fortune for graduate students to get one great advisor. With Sanjam around, my fortune doubles. Interactions with him serve as textbook examples of thinking out of the box, attacking the core of problems, and revealing unusual associations. As a super-nice person, he was never stingy with his encouragement, even when my ideas turned out to be failures. That was very crucial to my self-confidence in my early days of research. Although his work demystifies so many aspects of cryptography, it still remains a mystery to me that how he manages to maintain such a high level of research productivity while living a life with enough leisure.

During and after my days at Berkeley, Mohammad Hajiabadi has always been a genuine friend and almost-advisor. He has guided me through several topics, including key-dependent message security, selective-opening attacks, and IND-CCA encryptions, always being patient in answering my beginner's questions. He has the incredible ability to decompose complex objects into easy pieces and illustrate the main intuition while keeping track of *all* the details. Without his help, I would not have gotten familiar with these topics so fast and smoothly. Also, his career advice helped a lot when I was confused about the post-graduate future. Most importantly, I want to thank him for getting me interested in lower bounds in cryptography, which end up being a major topic of the current thesis.

I thank Romain Gay for teaching me about functional encryptions, though I learned that you tend to "talk about functional encryptions to everyone you meet" [Lep19]. I "appreciate" your skepticism, always questioning the practicability of your research, even when you were working on indistinguishable obfuscation. Thanks also for acknowledging me in your thesis [Gay19]. Indeed, you should practice the pronunciation of "café au lait" such that you will not be embarrassed in Paris. Or maybe this is the very reason why you escape to Zürich now? Let us always remember our motto—*Qui aime bien châtie bien*. Summer boy forever!

I thank Sruthi Sekar for convincing me of the glamor of information-theoretical cryptography. I was also happy that we had shared the belief in theoretical research until you suddenly betrayed us and started to publish CCS papers. Too often I made jokes about your family fortune. I think I will never be convinced by your rebuttal, especially in light of the aforementioned betrayal. But if you are not trying to play humble, then let those jokes be my best wishes to you.

I want to give special thanks to Prashant Nalini Vasudevan and Mohammad Mahmoody. It is ecstatic to find friends in cryptography who share similar interests in novels and classical music. Prashant has been very patient in teaching me the basics of quantum computing; he also brought me into the world of Etgar Krete, Richard Brautigan, and Sandra Cisneros. Mohammad gave helpful advice when I was trying to study ZKPCP and complexity theory; I also miss our enjoyable talks about opera (though mainly about Bizet). I am always expecting more interactions with both of you.

Thanks to my lab-mates, Rohit Chatterjee and Allen Kim, for enduring my fastidiousness and workaholism on a daily basis. Also, I benefit a lot from our discussions about English grammar, though our time is mostly spent on particular quirks, such as if a comma is necessary before coordinating and subordinating conjunctions. Rohit's unusual sense of humor and characteristic usage of adjectives keep me re-discovering the vigorousness of English; Allen's obsession with brain teasers and puzzle games always gives me the feeling (at least illusively) that my intelligence quotient has been increased through adequate interactions with him. You guys make our laboratory an attractive place to stay, both for research and entertainment.

I would like to thank Michael A. Bender, Jie Gao, and Omkant Pandey for serving on my Research Proficiency Examination Committee; I also thank Michael A. Bender, Sanjam Garg, Omkant Pandey, and Michalis Polychronakis for serving on my Dissertation Examination Committee. Special thanks to Michael for chairing both of the committees.

I would like to thank my amazing co-authors during my Ph.D. career. They are Rohit Chatterjee, Max Curran, Samir R. Das, Sanjam Garg, Himanshu Gupta, Mohammad Hajiabadi, Dakshita Khurana, Allen Kim, Giulio Malavolta, Nilu Nalini, Omkant Pandey, Sina Shiehian, and Ivan Visconti (thank you also for guiding me through my first sub-review). I also want to express my gratitude to a group of extremely nice people for taking time and effort to interact with me. They are James Bartusek, Chen Chen, Nai-Hui Chia, Kai-Min Chung, Fuyuki Kitagawa, Fermi Ma, Daniel Masny, Peihan Miao, Ryo Nishimaki, Willy Quach, Radu Sion, Pratik Soni, Nicholas Spooner, Akshayaram Srinivasan, Xiao Wang, and Takashi Yamakawa. Those collaborations and inspiring discussions had an indispensable influence on my taste in and style of research.

Graduating during the COVID-19 pandemic means massive uncertainty for (at least) the near future. That being so, I am very much obliged to Giulio Malavolta for inviting me to the Max Planck Institute during summer 2021. Without your generous backup, my transition to the post-doctoral career could not have been so smooth. Thanks a ton also go to Nai-Hui Chia and Kai-Min Chung. I am more than appreciative of all your efforts to arrange the post-doctoral position for me, and for boosting my research on quantum cryptography. I cannot wait to start with you the exciting adventures in the quantum wonderland.

I also want to give special thanks to Takashi Yamakawa. Though we just got to know each other very recently, I have already bombarded you with a thousand questions about quantum computing and information. You are a super nice and patient mentor whom I always wish I had known in my earlier Ph.D. days. Thank you for helping systematize my knowledge of quantum cryptography, and I am looking forward to more collaborations.

After enough words on research, it is time to talk about life. Only a small group of people end up together with their high school sweetheart; an even smaller group have their better half as their best friend. I could not be more grateful to my wonderful fiancée, Fangfang Song, who positions me at the intersection of both. Growing together for so many years, we have developed similar tastes in music, literature, history, politics, and tens of thousands of other things. But I still find it surprising that I can discuss Abstract Algebra with you, a beautiful and charming lady working as a professional lawyer! To me, you are scientific proof of the existence of soul mates. I am now ready to put the "Q.E.D." here, because you complete the theorem of my life.

Xiao Liang
August 16th, 2021

Chapter 1

Introduction

Secure multi-party computation (MPC) enables two or more mutually distrustful parties to compute any functionality without compromising the privacy of their inputs. Introduced in the innovative works of [Yao86, GMW87], this model has since received significant attention. Due to its generality, the setting of MPC can capture almost every cryptographic task. During the past decades, the efforts invested in this topic have led to numerous new results and beautiful techniques that benefit most (if not all) branches of cryptography. Thus, understanding the nature of MPC (e.g. its power, limits, and computation/communication complexity) is always of fundamental importance to the theory of cryptography. Moreover, the new millennium has witnessed significant progress in constructing *efficient* MPC protocols (see the recent survey [HHNZ19]), transforming the theory to pragmatic technologies that are even being commercialized by multiple companies.

An important factor influencing the efficiency of MPC is how the protocols make use of their building blocks. A cryptographic construction (not limited to MPC protocols) is *black-box* if it does not refer to the code of any cryptographic primitive it uses, and only depends on their input/output behavior. Such constructions are typically much more efficient than comparable non-black-box constructions, as they avoid expensive non-black-box operations such as running *zero-knowledge* protocols or *circuit-garbling* schemes on other cryptographic primitives. Also, they remain valid even if the building block is based on a *physical* object such as a noisy-channel or tamper-proof hardware [Wyn75, CK88, GLM$^+$04]. Furthermore, black-box constructions inherently ask us to reduce a cryptographic problem to basic and *general* hardness assumptions[1], thus leading to solutions addressing the essence of the given task. Indeed, starting from the seminal paper of Impagliazzo and Rudich [IR89], a rich line of work has developed rigorous methodologies to characterize the (im)possibility *and* (in)efficiency of black-box constructions (e.g., [Sim98, GKM$^+$00, GT00, GMR01, GGK03, RTV04, HR04, BM07, BM09, BBF13, Haj18, GHMM18]). Many insightful results have been obtained in this direction, shaping and re-shaping our view of cryptography.

It has been an important theme to improve the performance of MPC protocols by developing black-box techniques (e.g., [IPS08, IKOS07, GOSV14, KOS18]). But there are still some settings where *only* non-black-box constructions are known. This work seeks to further our understanding of efficient MPC constructions by narrowing the gap between black-box

[1]In contrast to specific hardness assumptions such as discrete log, factoring, and RSA.

and non-black-box constructions. In the remainder of this chapter, we will first provide background information and then summarize our contributions.

1.1 Background

The Simulation Paradigm. Formalizing the security requirement of MPC is not an easy task. Consider the setting where m parties $(P_1, \ldots, P_m)$ want to evaluate a function $f(\vec{x})$ on their corresponding private inputs $\vec{x} = (x_1, \ldots, x_m)$ such that party P_i learns (only) the i-th component of $f(\vec{x})$. Intuitively, a proper definition of MPC should (at least) guarantee:

- **Correctness:** the output of learned by each party is distributed according to the prescribed function $f(\cdot)$; **and**
- **Privacy:** the parties that do not follow the protocol honestly learn nothing other than their prescribed output.

This intuition was formalized using the so-called "simulation paradigm", which was originally developed to capture the notion of *zero-knowledge* [GMR85, GMR89]. Given a protocol Π that purports to implement f, this paradigm compares two worlds:

- In the *real world*, the m parties with their own input interact with each other following the protocol Π. Some of them might be corrupted by an adversary $\mathcal{A}$. If a party P_i is corrupted, we assume that all its input, randomness and internal states are exposed to $\mathcal{A}$; and $\mathcal{A}$ controls the behavior of P_i during the execution of Π.
- In the *ideal world*, each party simply sends its input to an idealized functionality $\mathcal{F}$. $\mathcal{F}$ is incorruptible. It will compute the function f properly and deliver the outputs to each party. Note that in this execution, there is no interactions among the parties, and each party learns nothing more than its input and the prescribed output (and the information that can be inferred). So, this ideal-world execution is considered as the most secure scenario that one can hope for.

It is then reasonable to believe that Π is secure if any adversary $\mathcal{A}$ participating in the real execution of Π can do no more harm than in an ideal-world execution. Formally, we imagine a simulator in the ideal world, who "corrupts" the same parties that are controlled by $\mathcal{A}$ in the real world. If any information learned by $\mathcal{A}$ can be generated (or "simulated") by this simulator in the ideal-world execution, then we can safely think that $\mathcal{A}$ gains nothing useful by corrupting those parties. That is, the protocol is *secure*.

Stand-Alone Security. The most basic security notion for MPC is known as *stand-alone security*, which guarantees the privacy of honest parties for a *single* execution of the underlying protocol. Early constructions of general-purpose[2] MPC (in this stand-alone setting) were non-black-box in nature, particularly due to NP-reductions required by underlying *zero-knowledge proofs* [GMW87]. Specifically, zero-knowledge (ZK) proofs enable each party to prove that he/she follows the protocol without revealing its private input and internal

[2]An MPC protocol is general-purpose if it can be used to compute any efficiently-computable functionalities.

states. This enforces honest behaviors of the parties while maintaining their privacy. However, this technique incurs expensive computation: MPC constructions usually make use of other cryptography primitives such as commitments and oblivious transfers; running ZK proofs for statements like "I perform the commitment honestly" is inefficient because (1) these primitives themselves are usually complex; and (2) general-purpose ZK proofs are typically designed for some NP complete language; converting such statements to an instances of the suitable NP complete language incurs huge overhead.

Due to the above drawback of non-black-box techniques, a beautiful line of work pursuits efficient constructions making only black-box use of other cryptographic primitives. Ishai et al. [IKLP06] presented the first black-box construction of general-purpose MPC based on enhanced trapdoor permutations or homomorphic public-key encryption schemes. This, together with the subsequent work of Haitner [Hai08], provided a black-box construction of a general MPC protocol under minimal assumptions of semi-honest *oblivious transfer* (OT). The round complexity of black-box MPC was improved to $O(\log^* n)$ rounds[3][4] by Wee [Wee10], and to constant rounds by Goyal [Goy11]. Very recently, Applebaum et al. [ABG⁺20] showed that 2-round MPC is unachievable by making only black-box use of 2-round OT. For two-party computations (2PCs), a constant round construction was first obtained by Pass and Wee [PW09]; subsequently a 5-round construction was given by Ostrovsky, Richelson, and Scafuro [ORS15], which is optimal w.r.t. *black-box proof techniques* [KO04].

Composable Security (in the Plain Model). While stand-alone security suffices for many applications, stronger notions of security are required for complex environments such as the Internet, where several MPC protocols may run *concurrently*. This setting is often referred to as the *concurrent* setting. Unfortunately, as shown by Feige and Shamir [FS90], stand-alone security does not necessarily imply security in the concurrent setting. To address this issue, Canetti [Can01] proposed the notion of *universally-composable* (UC) security which has two important properties: *concurrent security* and *modular analysis*. The former means that UC secure protocols maintain their security in the presence of other *concurrent* protocols, and the latter means that the security of a larger protocol in the UC framework can be derived from the UC security of its component protocols. This latter property is stated as a composition theorem which, roughly speaking, states that UC is closed under concurrent composition. These properties make UC the dream notion for composable security. It is also worth noting that UC is actually the "minimal" requirement for any meaningful notion of composable security [Lin03b]. Thus, it is fair to say that UC is "the correct" security definition for general protocol composition.

Unfortunately, UC security turns out to be impossible *in the plain model*[5] for most tasks [Can01, CF01, CKL03]. Relaxations of UC that consider composing the same protocol were also ruled out by Lindell [Lin03a, Lin04]. Moreover, [Lin03b] showed that any definition (with polynomial-time adversaries) seeking general composability in an ideal-real framework

[3]For $n \geq 1$, $\log^* n$ denotes the iterated logarithm of n, which can be evaluated by solving the recurrence relation $\log^* n = 1 + \log(\log^* n)$, with the base case being $\log^* 1 = 0$.

[4]We remark that the n denotes the number of parties, not the security parameter.

[5]Indeed, in the common reference string model, any polynomial-time functionality can be UC-computed, for any number of corrupted parties [CLOS02].

would suffer from the same impossibility results as the ordinary UC notion. We remark that these impossibility results hold even for non-black-box simulations (and constructions).

These strong negative results motivated the search for relaxed (but still appealing) notions of composable security by[6]:

1. resorting to weaker notions such as bounded concurrency [Bar02, Pas04b], input indistinguishability [MPR06], or a combination thereof [GGJ13]; **or**
2. endowing more power to the simulator, e.g., super-polynomial running time [Pas03, PS04, BDH$^+$17], or the ability to receive multiple outputs [GJO10, GJ13].

Same as in the stand-alone setting, black-box techniques are critical to the efficiency of composable MPC. As we will show in Section 1.2 and Section 1.3, both of the above directions leave interesting open questions regarding black-box constructions to be answered.

1.2 Bounded-Concurrent MPC

In the bounded-concurrent model, a bound m is fixed a-priori, and the protocol design may depend on m. The adversary is allowed to participate in at most m simultaneous executions of the *same* protocol. We consider security against *dishonest majority* with *interchangeable* roles, i.e., the adversary can choose an arbitrary subset of (all but one) parties to corrupt in each session[7]. As in the original (unbounded) setting, the ideal-world simulator is required to run in (expected) polynomial time. That is, we put an upper bound on the number of sessions the protocol can securely compute; other than that, we do not give up anything on the security requirements. Due to the a-priori bound, it is feasible to bypass the aforementioned negative results. Lindell [Lin03a] presented a m-bounded concurrent two-party protocol in $O(m)$ rounds using black-box simulation. Subsequently, Pass and Rosen [PR03] presented a constant-round two-party protocol, and Pass [Pas04b] a constant round MPC protocol (under improved assumptions), using non-black-box simulation. All general-purpose secure-computation protocols in this setting make non-black-box use of the underlying cryptographic primitives. Therefore, we wonder if it is possible do the same but in a black-box manner:

> **Question 1:** *Can we construct constant-round bounded-concurrent MPC making only black-box use of other primitives?*

The state-of-the-art non-black-box construction in [Pas04b] relies on collision-resistant hash functions (CRHFs) and (enhanced) trapdoor permutations (TDPs). Ideally, we want a black-box version of it making the same (or even weaker) assumptions.

[6]Another important approach is to introduce setup assumptions such as common reference strings. The current work only focuses on solutions in the plain model. Also, we refer the reader to [BS05, Section 1.1] for a thorough summary of relevant models.

[7]A formal treatment of this model is provided in Section 3.2.3.

1.3 Angel-Based Universally-Composable MPC

Roughly speaking, angel-based UC security is the same as UC security except that it allows the simulator as well as the adversary access to a super-polynomial resource called an "angel" which can perform a pre-defined task such as inverting a one-way function[8]. Due to the super-polynomial power of the simulator (from the angel), one can hope to circumvent the lower bounds for UC-secure constructions without setup.

We remark that although weaker than the standard UC security, this angel-based UC notion provides meaningful security guarantees in many settings. For example, because the angels can be implemented in super-polynomial time, angel-based security implies super-polynomial-time-simulation (SPS) security [Pas03, BS05], a model where the simulator is allowed to run in super-polynomial time. That means angel-based UC security guarantees that whatever an adversary can do in the real world can also be done in the ideal world in super-polynomial time. Because the ideal-world functionalities usually provide information-theoretic security, the simulator (aka the ideal-world adversary) can do no harm even with super-polynomial power[9]. Furthermore, akin to the standard UC security, different protocols achieving angel-based UC security *w.r.t. the same angel* can be composed safely. In contrast, SPS security does not guarantee composability of (even) the same protocols.

Early constructions of angel-based security were based on non-standard/super-polynomial hardness assumptions [PS04, BS05, MMY06]. The beautiful work of Canetti, Lin, and Pass [CLP10] presented the first construction under polynomial hardness assumptions, and the subsequent work of Goyal et al. [GLP+15] improved the round complexity to $\widetilde{O}(\log \lambda)$ under general assumptions, where λ denotes the security parameter. Lin and Pass [LP12] gave the first black-box construction under the (minimal) assumption of semi-honest OT. The main drawback of [LP12] is that it requires polynomially many rounds even if the underlying OT protocol has constant rounds. To address this issue, Kiyoshima [Kiy14] presented a $\widetilde{O}(\log^2 \lambda)$-round construction assuming constant-round semi-honest OT. Therefore, there is still a $\log(\lambda)$ gap between the black-box and non-black-box constructions of angel-based UC MPC under standard assumptions. It is then interesting to ask if we can close this gap:

> **Question 2:** *Assuming (only) the existence of constant-round semi-honest OT, can we have a black-box construction for $\widetilde{O}(\log \lambda)$-round MPC satisfying the angel-based UC security?*

We remark that Broadnax et al. [BDH+17] present a constant-round black-box construction for (the weaker but still composable) shielded-oracle security, utilizing prior work by Hazay and Venkitasubramaniam [HV15] who provide a constant-round protocol in the CRS-hybrid model; however, they require stronger assumptions, specifically, homomorphic commitments and public-key encryption with oblivious public-key generation.

[8]A formal treatment of the Angel-Based UC model is provided in Section 4.2.2.

[9]The introduction of [PS04] contains a thorough discussion about the meaningfulness of SPS security.

1.4 Functionally-Black-Box Zero-Knowledge Proofs

Zero-knowledge proofs (ZKPs) are one of the most important building blocks for secure MPC. Optimizing the performance of ZKPs is of great importance to build better MPC protocols. Moreover, ZKPs have found numerous applications beyond MPC. ZKP itself is an interesting topic that forms a rich area of research. Therefore, investigating the (im)possibility of zero-knowledge proofs with better efficiency and security (e.g., non-malleablity, resettability, preciseness, etc.) is interesting on its own.

A significant achievement in cryptography has been the construction of ZKPs for NP-complete problems [GMW86]. Since every NP relation can be efficiently reduced to any NP-complete relation [Coo71, Kar72, Lev73], this yields a ZKP for all languages in NP. Due to this reason, ZKPs for NP-complete problems are often called *general-purpose* proofs. As evidenced by numerous follow-up works, general-purpose proofs have been incredibly useful to the theory of cryptography.

Early constructions of general-purpose ZKPs required only *black-box* access to any one-way function (OWF), i.e., they used the given OWF as an *oracle*. A *black-box construction* of this kind thus depends only on the input/output behavior of the given cryptographic primitive. In particular, it is independent of the specific implementation (or *code*) of the primitive.

A black-box construction is often preferred over a non-black one due to its attractive properties. For example, it remains valid even if the primitive/oracle is based on a physical object such as a noisy-channel or tamper-proof hardware [Wyn75, CK88, GLM$^+$04]. Also, its efficiency does not depend on the implementation details of the primitive; thus, efficiency can be theoretically independent of the primitive's code.

Unfortunately, general-purpose proofs are unsuitable when seeking a black-box construction for some desired cryptographic task since they inherently require the full code of the underlying relation to perform the NP reduction. In other words, if the relation requires black-box access to a OWF, the code of the OWF must be known *even though* neither the ZKP nor the relation needs it. In fact, this has been the main reason for the non-black-box nature of many cryptographic constructions that are otherwise optimal. Analogous black-box constructions often require significant effort and technical innovation, as evidenced by the secure computation literature, e.g., [Kil88, DI05, IKLP06, IKOS07, Hai08, IPS08, PW09, Wee10, Goy11, GLOV12, LP12, Kiy14, GGMP16, HV16b, GKP18, CLP20a, GLPV20].

In light of the above situation, it is tempting to imagine a "dream version" of general-purpose proofs where, if the underlying relation $\mathcal{R}$ requires black-box access to a cryptographic function f, say from a specified class such as the class of OWFs, then so should the general-purpose ZKP for proving membership in $\mathcal{R}$. We informally refer to such relations as *black-box relations*. Such a result, if possible, would greatly simplify the task of future black-box constructions and potentially unify the diverse set of techniques that exist in this area.

As one might suspect, this dream version is too good to be true. In his beautiful work, Rosulek [Ros12] rules out ZKPs for proving *membership in the range of a OWF f* given as an oracle. More specifically, assuming injective OWFs, Rosulek rules out (even honest-verifier) *witness-hiding* protocols [FS90] for the relation $\mathcal{R}^f = \{(y, x) \mid y = f(x)\}$ where f is chosen from the class of all OWFs and provided as an oracle to the protocol.

In contrast to the negative result for OWFs, a large body of literature constructs so-called *black-box commit-and-prove* protocols [IKOS07, GLOV12, GOSV14, HV18, KOS18, Kiy20]. Informally speaking, a commit-and-prove protocol between a committer and a receiver ensures that at the end of the protocol, the committer is committed to some hidden value satisfying a predefined property. This primitive can be constructed with only black-box access to an ordinary commitment scheme which may originally not support any proofs whatsoever. In many situations, commit-and-prove protocols serve as a good substitute for ordinary commitments; moreover, their ability to support proofs over committed values makes them a great tool for constructing larger black-box protocols.

In hindsight, we can view black-box commit-and-prove protocols as an alternative to bypass the aforementioned negative result of [Ros12]. That is, instead of constructing ZKP directly for every OWF, we ask the following indirect question:

> **Question 3:** *Given only black-box access to an OWF f, can we construct a new OWF F and a ZKP system Π_F for proving membership in the range of F?*

Of course, we can ask for general properties instead of merely range membership.

The idea is that F can be used as a substitute for f in any computation $C^{(\cdot)}$ that requires only black-box access to OWFs. More importantly, it gives hope that general-purpose black-box ZKPs for proving the correctness of computation $C^{(\cdot)}$ may be possible since the correctness of responses from F can be ensured using Π_F, all while requiring only black-box access to f. We remark that we do not obtain such a result for general computations in this work and merely point out that the existence of (F, Π_F) may open a path towards it.

We call the pair (F, Π_F) a *proof-based one-way function* (PB-OWF). Analogously, we consider proof-based versions of other primitives, specifically pseudo-random generators (PRGs) and collision-resistant hash functions (CRHFs). Motivated by the aforementioned possibility of a general-purpose proof system for black-box cryptographic computations $C^{(\cdot)}$, this work initiates a study of black-box constructions of proof-based cryptographic primitives.

1.5 Our Contributions

The current thesis contains a set of results that answer the aforementioned three questions.

1.5.1 For Question 1

To answer Question 1, we seek to construct general-purpose MPC protocols that make only black-box use of cryptographic primitives and remain secure under bounded-concurrent self composition. Furthermore, we seek constructions whose security can be proven under standard polynomial hardness assumptions (although, to the best of our knowledge, such protocols are not known even under, say, sub-exponential assumptions since the simulator must still run in polynomial time).

Towards this goal, we first aim to construct a black-box bounded-concurrent oblivious transfer (OT) protocol. At a high level, this construction relies on non-black-box simulation to handle simulation in the bounded-concurrent setting (along the lines of [Bar01, Pas04b]);

to ensure that this does not result in non-black-box use of cryptographic primitives, we implement this idea using the "black-box non-black-box" protocol of Goyal et al. [GOSV14]. Once we have control over bounded-concurrent simulation, we rely on the OT protocol of Garg et al. [GKP18] to achieve the full oblivious transfer functionality. Unfortunately, implementing this idea is somewhat complex, perhaps in part because abstractions such as "straight-line simulation/extraction" are not straightforward to formalize despite their intuitive appeal. We mitigate this situation by defining a new abstraction which we call (bounded) *robust zero-knowledge*; this notion asks for simulators to work even in the presence of (bounded) external communication which cannot be "rewound" (and therefore, looks very close to UC zero-knowledge [Can01]). Similar notion has been defined by [LP09] in the context of non-malleable commitment w.r.t. an external party. Zero-knowledge (ZK) with this robust property allows us to combine the non-black-box simulation techniques with the SPS based proof techniques of [GKP18] to achieve black-box bounded-concurrent OT. An additional feature of our protocol is that it has *constant* rounds.

Along the way, we also present the first *straight-line*[10] extractable commitment scheme that only makes black-box use of semi-honest OTs. This primitive may be useful for other applications, especially for black-box constructions of MPC protocols from minimal assumptions.

Having obtained bounded-concurrent security for OT, we proceed to construct bounded-concurrent MPC protocols for all functionalities. This step is executed almost identically to a similar step in [GKP18] and does not require any additional assumptions. It also maintains the black-box and constant round properties of the original OT protocol. Consequently, we obtain the first general-purpose bounded-concurrent secure MPC protocol that makes only black-box use of cryptographic primitives; furthermore, the protocol has constant rounds and relies only on standard polynomial hardness assumptions.

Theorem 1.5.1. *Assume the existence of constant-round semi-honest oblivious transfer protocols and collision-resistant hash functions. Then, there exists a constant-round black-box construction of general-purpose MPC that achieves bounded-concurrent security.*

This result is essentially a black-box version of Pass's result [Pas04b] (with slightly improved assumptions). We will elaborate on this in Chapter 3.

Other Related Work. In addition to the work mentioned in the introduction, there are several works that study security in the concurrent setting. For SPS-security, Pass, Lin, and Venkitasubramaniam [PLV12] present a constant-round non-black-box construction of MPC from constant-round semi-honest OT. Dachman-Soled et al. and Venkitasubramaniam [DMRV13, Ven14] present a non-black-box construction that satisfies adaptive security. And very recently, Badrinarayanan et al. [BGJ$^+$17] present a non-black-box 3-round construction assuming sub-exponential hardness assumptions. For angel-based security, Kiyoshima, Manabe, and Tatsuaki Okamoto [KMO14] present a constant-round black-box construction albeit under a sub-exponential hardness assumption, and Hazay and Venkitasubramaniam [HV16a] present a black-box construction that achieves adaptive security.

We have not discussed works that focus on other security notions, e.g., input-indistinguishable

[10]It means the extraction strategy does not involve rewinding techniques.

computation and multiple ideal-query model [Pas04b, MPR06, GJ13].

Black-box constructions have been extensively explored for several other primitives such as non-malleable or CCA-secure encryption, non-malleable commitments, zero-knowledge proofs and so on (e.g., [CHH$^+$07, PW08, CDMW08, GLOV12, GOSV14, OSV15]). For concurrent OT, Garay and MacKenzie [GM00] presented a protocol for independent inputs under the DDH assumption, and Garg et al. [GKOV12] proved the impossibility of this task for general input distributions.

1.5.2 For Question 2

We answer Question 2 affirmatively by the following theorem, which closes the gap of round complexity between black-box and non-black-box constructions of angel-based MPC under minimal assumptions:

Theorem 1.5.2. *Assume the existence of* R_{OT}*-round semi-honest oblivious transfer protocols. Then, there exists a* $\max(\widetilde{O}(\log\lambda), O(\mathrm{R}_{\mathsf{OT}}))$*-round black-box construction of a general MPC protocol that satisfies angel-based UC security in the plain model.*

Note that this yields a $\widetilde{O}(\log\lambda)$-round construction under the general assumption of enhanced trapdoor permutations since they imply constant-round semi-honest OT.

We follow the framework of [CLP10] and its extension in [LP12, Kiy14]. The main building block in [CLP10] is a special commitment scheme called a CCA-Secure Commitment. Roughly speaking, a CCA-secure commitment is a tag-based commitment scheme that maintains hiding even in the presence of a decommitment oracle $\mathcal{O}$. More specifically, the adversary receives one commitment from an honest committer and may simultaneously make concurrently many commitments to $\mathcal{O}$ (similar to non-malleable commitments [DDN91]). The oracle immediately extracts and sends back any value adversary commits successfully provided that it used a tag that is different from the one used by the honest committer. Lin and Pass [LP12] show that $O(\max(\mathrm{R}_{\mathsf{CCA}}, \mathrm{R}_{\mathsf{OT}}))$-round black-box angel-based MPC can be obtained from a $\mathrm{R}_{\mathsf{CCA}}$-round CCA commitment and a R_{OT}-round semi-honest OT protocol. Kiyoshima [Kiy14] demonstrated that $\widetilde{O}(k\cdot\log\lambda)$-round CCA-secure commitments can be obtained in a black-box manner from a k-round commitment scheme with slightly weaker security called "one-one CCA" where the adversary can open only one session each with the committer as well as the oracle; they further construct a $O(\log\lambda)$-round one-one CCA scheme from one-way functions in a black-box manner. We instead present a *constant round* construction of one-one CCA, which implies $\widetilde{O}(\log\lambda)$-round (fully) CCA-secure commitments using [Kiy14] (and Theorem 1.5.2 using [LP12]):

Theorem 1.5.3 (CCA Secure Commitments)**.** *Assume the existence of one-way functions. Then, there exists a* $\widetilde{O}(\log\lambda)$*-round black-box construction of a CCA-secure commitment scheme.*

We will elaborate on the above results in Chapter 4.

Other Related Works. The focus of our work is constructions in the plain model. Hazay and Venkitasubramaniam [HV16a] gave a black-box construction of an MPC protocol without any setup assumptions that achieves composable security against an *adaptive* adversary.

UC security can be achieved by moving to other trusted setup models such as the common reference string model [CLOS02, CF01, GO07], assuming an honest majority of parties [CKL03], trusted hardware [MR04, GLM+04, Kat07, CCOV19], timing assumptions on the network [KLP05], registered public-key model [BCNP04], setups that may be expressed as a hybrid of two or more of these setups [GGJS11], and so on. Lin, Pass, and Venkitasubramaniam [LPV09, PLV12] show that a large number of these setup models could be treated in a unified manner, and black-box analogues of these results were obtained by Kiyoshima, Lin, and Venkitasubramaniam [KLV17].

1.5.3 For Question 3

Toward answering Question 3, We obtain a mix of both negative and positive results as outlined below.

Negative Results via Black-Box Separation. In common applications of non-interactive primitives such as OWFs and PRGs, the entire input is usually controlled by the evaluator of these functions. We show that *proof-based PRGs* where the input (i.e., the seed) is *entirely* chosen by the evaluator cannot be constructed in a black-box manner from an (ordinary) OWF chosen from the class of all OWFs. Since PRGs can be constructed in a fully-black-box manner from OWFs [GL89, ILL89, HILL99a], this separates proof-based PRGs from ordinary PRGs.

More specifically, black-box construction of a proof-based PRG from (ordinary) OWFs consists of a *deterministic* and efficient oracle algorithm $G^{(\cdot)}$, along with an efficient protocol, $\Pi_G^{(\cdot)} = \langle P^{(\cdot)}, V^{(\cdot)} \rangle$, of two interactive oracle machines.[11] For every OWF f, algorithm G^f should be a PRG, and protocol $\Pi^f = \langle P^f, V^f \rangle$ should be a ZKP system for the relation $\mathcal{R}_G^f = \{(y, x) \mid \text{s.t. } y = G^f(x)\}$. Then, we show that a *fully-black-box* reduction [IR89, RTV04] from proof-based PRGs to ordinary OWFs does not exist if the prover chooses the entire seed.

The range-membership relation $\mathcal{R}^f = \{(y, x) \mid y = f(x)\}$ ruled out in [Ros12] is a special case of the aforementioned relation $\mathcal{R}_G^f = \{(y, x) \mid \text{s.t. } y = G^f(x)\}$. In our terminology, Rosulek rules out a special type of proof-based OWF $(F^{(\cdot)}, \Pi_F^{(\cdot)})$ where F is just a "delegate" for the oracle OWF; i.e., it returns the oracle's response when queried on the given input. This is captured in [Ros12] by formally defining the notion of *functionally-black-box* (FBB) protocols. In contrast, the relation we consider can make polynomially many queries to the oracle on arbitrary inputs and compute over the responses to produce the output. We extend the notion of FBB protocols to formally capture these extensions.

Partly due to these differences and our overall goals, our negative result is incomparable to Rosulek's. While Rosulek rules out black-box proofs for the range membership for OWFs assuming injective OWFs, ours is only a black-box separation, albeit without any additional assumptions. A black-box separation is the best one can hope for in our setting since *non-black-box* constructions of proof-based OWFs that use the code of the oracle trivially exist.

Positive Results. We next investigate whether mild restrictions on the inputs can help by-

[11]Note that the protocol is allowed to depend on $G^{(\cdot)}$ but not on the oracle which may be arbitrarily chosen later. The same holds for the relation $\mathcal{R}_G^f$ introduced next.

pass the black-box separation result. One option is to consider modifications along the lines of the Goldreich-Levin (GL) hardcore predicate [GL89], where one considers a OWF F constructed from any given OWF f on inputs of the form (x, r). This makes it possible to show that predicate $\mathsf{hc}(x, r) := \oplus_i(x_i \cdot r_i)$ is hardcore for the modified function $F(x, r) := r \| f(x)$ even though a hardcore predicate for arbitrary OWFs is still unknown. These changes to the function and the input do not seem to affect the applicability of their result significantly.

We adopt a similar approach to construct proof-based primitives. Continuing with OWFs as an example, we seek to construct a proof-based OWF $F^{(\cdot)}$ which can be instantiated with only black-box access to any OWF f and takes inputs of the form (x, r). As in the GL setting, x will act as the "main input" chosen by the evaluator/prover, and r will be publicly accessible from the output of $F^f(x, r)$. However, in a crucial difference, r will be *chosen by the verifier* during the execution of ZKP Π_F^f. There are no other restrictions on any of the objects. Some remarks are in order.

1. In light of our black-box separation result, it is essential to let the verifier choose r since no other restrictions are present. This means that the computation of $y = F^f(x, r)$ must be performed during the proof. We formalize this by modeling Π_F^f as *a secure two-party computation* protocol for evaluating the functionality that on inputs x and r from relevant parties, returns y. The ZK property is captured by requiring simulation-based security against malicious receivers; for soundness, we only require that the honest verifier, with high probability, does not output a y^* that is not in the range. This is effectively a black-box ZKP for the relation $\mathcal{R}_F^f(r) = \{(y, x) \mid \text{s.t. } y = F^f(x, r)\}$.[12]

2. The verifier must choose r from an unpredictable distribution such as the uniform distribution over sufficiently long strings, since otherwise, the soundness would be impossible as a cheating prover can simply guess r, bringing us back to the setting of the separation result.

3. Since the verifier may maliciously choose r to violate the one-way property of F^f, we require that for *every string* r, the function defined by $F^f(\cdot, r)$ is one-way as long as f is one-way.

We follow the same approach for formally defining proof-based versions of PRGs and CRHFs. Having settled on a satisfactory definition, we present black-box constructions of the proof-based versions of OWFs (for range membership), as well as PRGs and CRHFs, directly from their ordinary counterparts.

Theorem 1.5.4 (Informal)**.** *There is a fully-black-box construction of* proof-based $\mathsf{primitive}$ *as described above for the range-membership relation with two-party inputs of the form (x, r), assuming that* $\mathsf{primitive}$ *exists, where* $\mathsf{primitive} \in \{\text{OWF}, \text{PRG}, \text{CRHF}\}$.

At first glance, one may wonder whether black-box commit-and-prove protocols already yield proof-based OWFs. That is, the commit stage of such protocols can be viewed as a one-way function over the input (x, r) where x is the value to be committed and r is the randomness, the output y is the transcript of the commit-phase, and the proof-phase plays the

[12]For now, we only focus on range-membership proofs. The definitional approach is consistent with the commit-and-prove literature, although there are important differences since we are dealing with deterministic primitives.

role of associated ZKP. This approach does not work since the commit-and-prove protocols merely *bind* the prover to a well-defined value x. They do not guarantee that w.h.p. every accepting transcript has a valid "preimage" (x, r) that maps to it. In contrast, the soundness of range-membership proofs of proof-based OWFs requires that w.h.p. a preimage must exist for the output accepted by the honest verifier. At a technical level, the black-box commit-and-prove protocols are based on cut-and-choose techniques that can only guarantee that the accepted value is *close* to an honestly generated value, which is insufficient to guarantee a preimage.

Supporting Predicates. We show that it is possible to construct a slightly more general proof-system than merely range membership for each of our proof-based primitives. Continuing with the OWF example, we can construct a black-box proof-based OWF F^f such that for any predicate ϕ, the verifier learns a value y with the guarantee that there exists an input (x, r) such that: (1) $y = F^f(x, r)$ where r is chosen uniformly by the honest receiver, (2) $x = \alpha \| x'$, and (3) $\phi(\alpha) = 1$. That is, we can support any predicate (in fact, computation of any function) over a *prefix* of the preimage of the output. The ZKP system here depends on the code of ϕ but not that of f as before.

This extension is motivated by similar results for commit-and-prove, which are quite useful in constructing larger black-box protocols [GLOV12, KOS18]. We achieve this by presenting a new construction that combines our ideas for range-membership with the "MPC-in-the-head" technique [IKOS07].

We will elaborate on the above results in Chapter 5.

1.5.4 Citations to Published Work

The results presented herein are extended from the following conference papers:

[1] **Black-Box Constructions of Bounded-Concurrent Secure Computation**
Sanjam Garg, Xiao Liang, Omkant Pandey, and Ivan Visconti
The 12th International Conference on Security and Cryptography for Networks (SCN 2020)

[2] **Improved Black-Box Constructions of Composable Secure Computation**
Rohit Chatterjee, Xiao Liang, and Omkant Pandey
The 47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)

[3] **Towards a Unified Approach to Black-Box Constructions of Zero-Knowledge Proofs**
Xiao Liang and Omkant Pandey
The 41st International Cryptology Conference (CRYPTO 2021)

Chapter 2

Definitions and Preliminaries

2.1 Basic Notation

We assume the familiarity with standard concepts of complexity theory such as Turing machines, arithmetic/Boolean circuits, standard complexity classes (e.g. NP) and so on. These materials can be found in standard complexity or cryptography textbooks (e.g. [Gol01, Gol08, AB09]).

Following the convention in cryptography, we call an algorithm *efficient* if it can be implemented by a probabilistic Turing machine which runs in polynomial time. We refer to such algorithms as PPT (standing for "probabilistic polynomial time") algorithms/machines.

Let $\mathbb{N}$ be the set of natural numbers, $\mathbb{Z}$ the set of integers, and $\mathbb{R}$ the set of real numbers. The security parameter is denoted by $\lambda \in \mathbb{N}$. For any $k \in \mathbb{N}$, $[k]$ denotes the set $\{1, 2, \ldots, k\}$. For a distribution D we use $x \leftarrow D$ to mean x is sampled according to D. Unless emphasized otherwise, we assume uniform distribution by default, in which case we usually use the notation $\xleftarrow{\$}$. We use $y \in D$ to mean y is in the support of D. For a set S we overload the notation to use $x \xleftarrow{\$} S$ to indicate that x is chosen uniformly at random from S.

Let p be a predicate and $D_1, D_2, \cdots$ probability distributions, then the notation

$$\Pr\left[x_1 \leftarrow D_1; x_2 \leftarrow D_2; \cdots : p(x_1, x_2, \cdots)\right]$$

denotes the probability that $p(x_1, x_2, \cdots)$ holds after the ordered execution of the probabilistic assignments $x_1 \leftarrow D_1; x_2 \leftarrow D_2; \cdots$. The notation

$$\{x_1 \leftarrow D_1; x_2 \leftarrow D_2; \cdots : p(x_1, x_2, \cdots)\}$$

denotes the new probability distribution over $\{(x_1, x_2, \cdots)\}$.

Definition 2.1.1 (Probability Ensembles)**.** *Let I be a countable index set. An* ensemble *indexed by I is a sequence of random variables indexed by I. Namely, any $X = \{X_i\}_{i \in I}$, where each X_i is a random variable, is an ensemble index by I.*

Definition 2.1.2 (Negligibility)**.** *A function* $\mathsf{negl} : \mathbb{N} \to \mathbb{R}$ *is* negligible *if for every constant $c > 0$ there exist an N such that for all $n > N$,* $\mathsf{negl}(n) < \frac{1}{n^c}$.

Definition 2.1.3 (Computational Indistinguishability)**.** *Two ensembles* $X = \{X_i\}_{i \in I}$ *and* $Y = \{Y_i\}_{i \in I}$ *are said to be* computationally indistinguishable *if for every probabilistic polynomial time* (PPT) *algorithm* $\mathcal{D}$*, there exists a negligible function* $\mathsf{negl}(\cdot)$ *so that for every* $i \in I$*,*

$$|\Pr[\mathcal{D}(X_i, 1^i) = 1] - \Pr[\mathcal{D}(Y_i, 1^i) = 1]| < \mathsf{negl}(i)$$

Computational indistinguishability says that two ensembles looks the same in the eye of an efficient (PPT) distinguisher. But sometimes (see our discussion for commitment schemes in Definition 2.2.1), we require even stronger notion of indistinguishability such that even a unbounded-power distinguisher can not tell the difference (except for negligible probability). That means the distribution of two random variables from are close enough. This information-theoretical flavored closeness is called statistical indistinguishability.

Definition 2.1.4 (Statistical indistinguishability)**.** *Two ensembles* $X = \{X_i\}_{i \in I}$ *and* $Y = \{Y_i\}_{i \in I}$ *are* statistically indistinguishable (statistically close) *if their* statistical difference *is negligible, where the statistical difference (also known as variation distance) between* X *and* Y *is defined as the function*

$$\Delta(n) = \frac{1}{2} \cdot \sum_{\alpha} \big| \Pr[X_n = \alpha] - \Pr[Y_n = \alpha] \big|.$$

Statistical indistinguishability says that the $X_n \in \{X_i\}_{i \in I}$ and $Y_n \in \{Y_i\}_{i \in I}$ have negligible statistical distance when n is large enough. In another words, even a computationally-unbounded distinguisher cannot tell the difference *except with negligible probability*. An even stronger notion, called *perfect indistinguishable*, requires that X_n and Y_n are identically distributed. That means a computationally-unbounded distinguisher cannot tell the difference *at all*. since they are identical now.

We will use the following notation for these 3 kinds of indistinguishability: if two ensembles $\{X_i\}_{i \in I}$ and $\{Y_i\}_{i \in I}$ are:

– computationally-indistinguishable, we write: $\{X_i\}_{i \in I} \overset{\text{c}}{\approx} \{Y_i\}_{i \in I}$

– statistically-close, we write: $\{X_i\}_{i \in I} \overset{\text{s}}{\approx} \{Y_i\}_{i \in I}$

– identically-distributed, we write: $\{X_i\}_{i \in I} \overset{\text{i.d.}}{=\!=} \{Y_i\}_{i \in I}$

Remark 2.1.1 (A Note on Indistinguishability)**.** *The definition for indistinguishability given in Definition 2.1.3 is the standard one. This definition extends analogously to non-uniform adversaries characterized as circuits. These definitions works for poly-time hardness assumption very well.*

In [Pas04a], Pass defined the notion of "strong-indistinguishability" to handle super-polynomial hardness assumptions. Intuitively, two ensembles are strong $T(n)$*-indistinguishable if every* $T(n)$*-time probabilistic TM's advantage is upper-bounded by* $\frac{1}{T(n)}$*. We want to remark that even though this definition is called "strong", it is only stronger than the standard one in the super-polynomial setting (i.e.* $T(n)$ *is super-polynomial). When* $T(n)$ *is polynomial, this definition is actually weaker than the standard definition of computational indistinguishability (Definition 2.1.3). To see this, consider an adversary whose advantage is smaller than* $\frac{1}{T(n)}$ *but lager than* $\frac{1}{T^2(n)}$*. It satisfies the strong* $T(n)$*-indistinguishable definition, but does* *not satisfy the Definition 2.1.3.*

In a version of [Bar01][1], Barak used a similar definition for indistinguishability.

2.2 Commitment Schemes

Next we give the definition of commitment schemes with respect to a single bit. This definition can be easily extended to general commitment schemes, enabling the committer to commit to a string rather than just one bit. In many cases, given a construction of a bit-commitment scheme, it is straightforward to generalize it to a multi-bit commitment scheme.

Definition 2.2.1 (Bit Commitment Scheme)**.** *A bit commitment scheme* $\mathsf{Com} = (S, R)$ *is an efficient two-party protocol consisting of the following two stages. Throughout, both parties receive the security parameter* 1^λ *as input.*

- **Commit:** *The sender (committer)* S *has a private input* $b \in \{0, 1\}$*, and a sequence of coin tosses* r*. At the end of this stage, both parties receive as common output a commitment* $c = \mathsf{Com}(b; r)$*.*
- **Decommit:** *Both parties receive as input a commitment* $c = \mathsf{Com}(b; r)$*.* S *also receives the private input* b *and coin tosses* r *for* c*. This stage is non-interactive:* S *sends a single message to* R*, and* R *either outputs a bit and accepts or rejects.*

The following properties should be satisfied:

- **Completeness:** *If both parties are honest, then for any input bit* $b \in \{0, 1\}$ *that* S *gets,* R *outputs* b *and accepts at the end of the decommit stage.*
- **(Statistically) Hiding:** *For every unbounded deterministic strategy* R^**, the distributions of the view of* R^* *in the commit stage while interacting with an honest* S *are* **statistically close** *for* $b = 0$ *and* $b = 1$*.*
- **(Computationally) Binding:** *For every nonuniform* PPT S^**,* S^* *succeeds in the following game (breaks the commitment) with negligible probability:*

  1. S^* *interacts with an honest* R *and outputs a commitment* c'*.*
  2. S^* *outputs two messages* τ_0*,* τ_1 *such that for both* $b = 0$ *and* $b = 1$*,* R *on input* (c', τ_b) *accepts and outputs* b*.*

We have different types of hiding/binding properties as follows:

- **For hiding property:** *If the distributions of the view of* R^* *are identically distributed for* $b = 0$ *and* $b = 1$*, it is called* **perfectly-hiding***; If the hiding property is only against a* PPT *receiver (except for negligible probability), it is called* **computationally hiding***.*
- **For binding property:** *If a computationally-unbounded* S^* *can succeeds with at most negligible probability, it is called* **statistically binding***; If a computationally-unbounded* S^* *cannot succeeds at all, it is called* **perfectly binding***.*

[1]This version can be found at https://www.boazbarak.org/Papers/nonbb.pdf.

We remark that a commitment scheme cannot be statistically-hiding and statistically binding at the same time. To show the logic behind it, we now argue that it is impossible for a commitment scheme Com to be both perfectly hiding and binding. A similar yet more involved argument can be established in the statistical version. On the one hand, Com is perfectly hiding, which means any value $c = \mathsf{Com}(x)$, can be decommit to both 1 or 0. That is because the receiver is computationally unbounded in the "perfectly-hiding" setting. It can always reverse Com (or brute force the domain of $\mathsf{Com}(\cdot)$) to see the "pre-image" x for c. If there exist only a unique x which commits to c, the unbounded receiver breaks the hiding property since it learns that the unique x must be the message committed by the committer. On the other hand, Com is perfectly-binding, which means there is a distinct decommitment to any value $c = \mathsf{Com}(x)$. This contradicts the previous discussion about perfectly-hiding property.

Feasibility Results. First, note that any commitment scheme in the two-player model implies the existence of one-ways. Statistically-binding commitment schemes are known from one-way functions [Nao90]. Statistically-hiding commitment schemes are also known from one-way functions [HR07]. Perfectly-hiding commitment schemes are known from one-way permutations [NOVY98].

Remark 2.2.1. *Naor and Yung [NY89] proposed a statistically-hiding commitment from* CRHF*s, which was later extended to a multi-bit commitments in [DPP98]. Since* CRHF *implies one-way functions, their result is theoretically weaker than the construction in [HR07]. But the construction is more efficient and practical.*

2.3 Extractable Commitments

A commitment scheme is extractable if there exist an efficient extractor such that, as long as the committer behaves honestly, the committed value can be extracted. Constructions for such commitment already existed implicitly in the implementation of concurrent zero-knowledge protocols in [PRS02, Ros04]. This concept and construction was made explicit in [MOSV06], which also inherited the concurrent extractability from [PRS02]. The standalone version was later formalized and used in other works [PW09, GGJS12, GKP17]. It suffices our purpose once the extractability property holds in a standalone setting. We now present the definition (in Definition 2.3.1) and construction (in Protocol 2.3.1) used in [PW09].

Definition 2.3.1 (Extractable Commitment)**.** *A commitment scheme* $\mathsf{ExtCom} = (S, R)$ *is extractable if there exists an expected polynomial-time probabilistic oracle machine (the extractor)* $\mathcal{E}$ *that given oracle access to any PPT cheating sender* S^* *outputs a pair* (τ, σ^*) *such that:*

- **Simulation:** *τ is identically distributed to the view of S^* at the end of interacting with an honest receiver R in commitment phase.*
- **Extraction:** *the probability that τ is accepting and $\sigma^* = \bot$ is negligible.*
- **Binding:** *if $\sigma^* \neq \bot$, then it is statistically impossible to open τ to any value other than σ^*.*

Protocol 2.3.1: Extractable Commitment Scheme

The extractable commitment scheme, based on any commitment scheme Com, works in the following way.

Input:

– Both S and R get security parameter 1^λ as the common input.

– S gets a string σ as his private input.

Commitmment Phase:

– The sender (committer) S commits using Com to λ pairs of strings $\{(v_i^0, v_i^1)\}_{i=1}^{\lambda}$ where $(v_i^0, v_i^1) = (\eta_i, \sigma \oplus \eta_i)$ and η_i are random strings in $\{0,1\}^{\ell(\lambda)}$ for $1 \leq i \leq \lambda$.[a]

– Upon receiving a challenge $\vec{c} = (c_1, \ldots, c_\lambda)$ from the receiver R, S opens the commitments to $(v_1^{c_1}, \ldots, v_\lambda^{c_\lambda})$.

– R checks that the openings are valid.

Decommitment Phase:

– S sends σ and opens the commitments to all λ pairs of strings.

– R checks that all the openings are valid, and also that $\sigma = v_1^0 \oplus v_1^1 = \cdots = v_\lambda^0 \oplus v_\lambda^1$.

[a]Actually, the scheme will be secure as long as we use Com to commit $k = \omega(\log \lambda)$ pairs of strings.

2.4 Zero-Knowledge Proofs and Arguments (of Knowledge)

For an NP language $\mathcal{L}$, let $\mathcal{L}_\lambda = \mathcal{L} \cap \{0,1\}^\lambda$. For any $x \in \mathcal{L}_\lambda$, $R(x)$ denote the set of its witness (in terms of the NP relation for $\mathcal{L}$).

For a probabilistic Turing machine A, we use $A(x, y; r)$ to denote the output of A on input x and auxiliary tape y, when the random tape is fixed to r. We may drop r from the argument list when it is not necessary to be explicit about the random tape. When it is clear from the context, we (ab)use $A(x, y; r)$ to represent the machine A specified with the corresponding (auxiliary) input and random tape.

For a pair of (potentially probabilistic) interactive machines A and B, $\langle A(y), B(z)\rangle(1^\lambda, x)$ denotes the random variable representing the output of B when interacting with A on common input x and the security parameter 1^λ, when the random input to each machine is uniformly and independently chosen, and A (resp. B) has private input y (resp. z). The private input y (resp. z) will be dropped for A (resp. B) in the above notation when it is an empty string. When we want to be specific about the underlying protocol Π which A and B implements, we will write $\langle A(y), B(z)\rangle(1^\lambda, x)$. When it is clear from the context, we (ab)use $\langle A(y), B(z)\rangle(1^\lambda, x)$ to represent the execution of A and B specified with the corresponding common and private input.

Definition 2.4.1 (Interactive Proofs (IP) and Arguments)**.** *A pair of* PPT *interactive Tur-*

ing machines $\langle P, V\rangle$ is called an interactive argument for a language $\mathcal{L} \in \mathsf{NP}$ if the following conditions hold:

– **Completeness.** *For every $\lambda \in \mathbb{N}$, every $x \in \mathcal{L}_\lambda$ and every $w \in R(x)$,*

$$\Pr[\langle P(w), V\rangle(1^\lambda, x) = 1] = 1,$$

where the probability is taken over the random coins of P and V.

– **Computational Soundness.** *For every* PPT *machine P^* and every $y \in \{0,1\}^*$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for every $x \notin \mathcal{L}_\lambda$ with $|x| \geq n$,*

$$\Pr[\langle P^*(y), V\rangle(1^\lambda, x) = 1] \leq \mathsf{negl}(\lambda)$$

where the probability is taken over the random coins of P^* and V.

An interactive argument is an interactive proof *if the soundness property holds against all (potentially unbounded) P^*'s.*

Definition 2.4.2 (Zero-Knowledge Proofs and Arguments)**.** *An interactive proof (resp. argument) system $\langle P, V\rangle$ for $\mathcal{L} \in \mathsf{NP}$ is a* zero-knowledge argument (resp. argument) *for the same $\mathcal{L}$ if it additionally satisfies the following zero-knowledge property.*

– **Zero-Knowledge.** *There exists an expected* PPT *machine* Sim *such that for every* PPT *machine V^*, every $x \in \mathcal{L}_\lambda$ and every $z \in \{0,1\}^*$,*

$$\{\mathsf{Sim}(1^\lambda, x, z)\}_{\lambda\in\mathbb{N}} \overset{\text{c}}{\approx} \{\mathsf{View}_{V^*}(1^\lambda, x, z)\}_{\lambda\in\mathbb{N}},$$

where $\mathsf{View}_{V^*}(1^\lambda, x, z)$ denotes the view of V^* from the interaction $\langle P(w), V^*(z)\rangle(1^\lambda, x)$ for an arbitrary $w \in R(x)$.

Definition 2.4.3 (Interactive Arguments of Knowledge)**.** *An interactive argument system $\langle P, V\rangle$ for $\mathcal{L} \in \mathsf{NP}$ is an interactive argument* of knowledge *for the same $\mathcal{L}$ if it additionally satisfies the following argument of knowledge (AoK) property.*

– **AoK Property (with knowledge error $\kappa(\cdot)$).** *There exists a positive polynomial $q(\cdot)$ and a probabilistic oracle machine K such that for every* PPT *machine P^*, every $\lambda \in \mathbb{N}$ and every $x \in \mathcal{L}_\lambda$, machine K satisfies the following condition:*

   ∗ *Let $p(x, y, r) := \Pr[\langle P^*(y; r), V\rangle(1^\lambda, x) = 1]$. If $p(x, y, r) > \kappa(\lambda)$, then on input 1^λ and x, and with oracle access to $P^*(1^\lambda, x, y; r)$, machine K runs in expected polynomial time and output a $w \in R(x)$ with probability at least:*

$$\frac{p(x, y, r) - \kappa(\lambda)}{q(\lambda)}$$

   If $\kappa(\cdot)$ is negligible, then we say the argument has negligible knowledge error.

An interactive argument of knowledge is an interactive proof *of knowledge if the above AoK property holds against all (potentially unbounded) P^*'s.*

Definition 2.4.4 (Zero-Knowledge Proofs (Arguments) of Knowledge)**.** *An zero-knowledge proof (resp. argument) of knowledge for a language $\mathcal{L} \in \mathsf{NP}$ is an interactive proof (resp. argument) of knowledge for $\mathcal{L}$ that additionally achieves the zero-knowledge property in Definition 2.4.2.*

2.5 (Computationally) Secure Multi-Party Computation

The formal definition for secure multi-party computations can be found in standard cryptography textbooks (e.g., [Gol09, HL10, Lin16]). For completeness, we include a brief description of the model here. The description is taken from [AL11] with a few syntactic changes.

Consider malicious adversaries that can follow an arbitrary strategy in order to carry out their attack. Security is formalized by comparing a real protocol execution to an ideal model where the parties just send their inputs to the trusted party and receive back outputs.

Remark 2.5.1. *In this work, we only focus on the static corruption model. In this model, the adversary is given a fixed set of parties whom it controls. Honest parties remain honest throughout and corrupted parties remain corrupted.*

Execution in the Real Model. In the real model, there are n parties $(P_1, \ldots, P_n)$, modeled as PPT interactive Turing machines, running the protocol Π. We consider a synchronous network with private point-to-point channels, and an authenticated broadcast channel. This means that the computation proceeds in rounds, and in each round parties can send private messages to other parties and can broadcast a message to all other parties. We stress that the adversary cannot read or modify messages sent over the point-to-point channels, and that the broadcast channel is authenticated, meaning that all parties know who sent the message and the adversary cannot tamper with it in any way. Nevertheless, the adversary is assumed to be *rushing*, meaning that in every given round it can see the messages sent by the honest parties before it determines the messages sent by the corrupted parties.

Let $\mathcal{A}$ be an arbitrary machine[2] with auxiliary input z, and let $I \subseteq [n]$ be the set of corrupted parties controlled by $\mathcal{A}$. We denote by $\mathrm{REAL}_{\Pi,\mathcal{A}(z),I}(\vec{x})$ the random variable consisting of the view of the adversary $\mathcal{A}$ and the outputs of the honest parties, following a real execution of Π in the aforementioned real model, where for every $i \in [n]$, party P_i has input x_i, i.e. the i-th element of $\vec{x}$.

Execution in the Ideal Model. In the ideal model for a functionality f, the parties send their inputs to an incorruptible trusted party who computes the output for them. We denote the ideal adversary by Sim, and the set of corrupted parties by I. An execution in the ideal model works as follows:

– **Input stage:** The adversary Sim for the ideal model receives auxiliary input z and sees the inputs x_i of the corrupted parties P_i (for all $i \in I$). Sim can substitute any x_i with any $x_i'$of its choice under the condition that $|x_i'| = |x_i|$.

[2]$\mathcal{A}$ may not be efficient. The choice of $\mathcal{A}$'s running time depends on the security one wants to achieve.

– **Computation:** Each party sends its (possibly modified) input to the trusted party; denote the inputs sent by $x'_1, \ldots, x'_n$. The trusted party computes $(y_1, \ldots, y_n) = f(x_1, \ldots, x'_n)$ and sends y_j to P_j for every $j \in [n]$.

– **Outputs:** Each honest party P_j (for $j \notin I$) outputs y_j, the corrupted parties output $\perp$, and the adversary Sim outputs an arbitrary function of its view.

We denote by $\mathrm{IDEAL}_{f,\mathsf{Sim}(z),I}(\vec{x})$ the outputs of the ideal adversary Sim controlling the corrupted parties in I and of the honest parties after an ideal execution with a trusted party computing f, upon inputs $x_1, \ldots, x_n$ for the parties and auxiliary input z for Sim. We stress that the communication between the trusted party and $P_1, \ldots, P_n$ is over an ideal private channel.

Definition of security. Informally, we say that a protocol is secure if its real-world behavior can be emulated in the ideal model. That is, we require that for every real-model adversary $\mathcal{A}$ there exists an ideal-model adversary Sim such that the result of a real execution of the protocol with $\mathcal{A}$ has the same distribution as the result of an ideal execution with Sim. (We remark that the "distribution" here refers to the *joint* distribution of the outputs of all parties, not only the corrupted ones.) This means that the adversarial capabilities of $\mathcal{A}$ in a real protocol execution are just what Sim can do in the ideal model.

In the definition of security, we require that the ideal-model adversary Sim run in time that is polynomial in the running time of $\mathcal{A}$, whatever the latter may be. As argued in [Can00a, Gol09] this definitional choice is important since it guarantees that information-theoretic security implies computational security. In such a case, we say that Sim is of comparable complexity to $\mathcal{A}$. In the following, we define computationally secure MPC, where both $\mathcal{A}$ and Sim are PPT machines. Looking ahead, we will deal with the information-theoretical setting in Section 2.7, where $\mathcal{A}$ and Sim could be unbounded machines.

Definition 2.5.1 (Computationally-Secure MPC)**.** *Let $f : (\{0,1\}^*)^n \mapsto (\{0,1\}^*)^n$ be an n-ary functionality, and let Π be a protocol. We say that Π is t-secure for f if for every PPT adversary $\mathcal{A}$ in the real model, there exists a probabilistic adversary Sim of comparable complexity in the ideal model, such that for every $I \subseteq [n]$ of cardinality at most t, every $\vec{x} \in (\{0,1\})^n$ where $|x_1| = \ldots = |x_n|$, and every $z \in \{0,1\}^*$, it holds that:*

$$\{\mathrm{IDEAL}_{f,\mathsf{Sim}(z),I}(\vec{x})\} \overset{c}{\approx} \{\mathrm{REAL}_{\Pi,\mathcal{A}(z),I}(\vec{x})\}.$$

2.6 Verifiable Secret Sharing Schemes

A verifiable secret sharing (VSS) [CGMA85] scheme is a two-stage secret sharing protocol for implementing the following functionality. In the first stage, denoted by $\mathsf{VSS}_{\mathsf{Share}}$, a special player referred to as dealer, shares a secret s among n players, in the presence of at most t corrupted players. In the second stage, denoted by $\mathsf{VSS}_{\mathsf{Recon}}$, players exchange their views of the **Share** stage, and reconstruct the values. The functionality ensures that when the dealer is honest, before the second stage begins, the t corrupted players have no information about the secret. Moreover, when the dealer is dishonest, at the end of the **Share** stage the honest players would have realized it through an accusation mechanism that disqualifies the dealer.

The formal definition is presented in Definition 2.6.1. [BGW88, CDD+99] implemented $(n+1, \lfloor n/3 \rfloor)$-perfectly secure VSS schemes, and $(n+1, \lfloor n/4 \rfloor)$-perfectly secure VSS schemes can be found in [GIKR01]. These constructions suffices for all the applications in the current thesis.

Definition 2.6.1 (Verifiable Secret Sharing)**.** *An $(n+1, t)$-perfectly secure VSS scheme Π_{VSS} consists of a pair of protocols $(\mathsf{VSS}_{\mathsf{Share}}, \mathsf{VSS}_{\mathsf{Recon}})$ that implement respectively the sharing and reconstruction phases as follows.*

- **Sharing Phase $\mathsf{VSS}_{\mathsf{Share}}$:** *Player P_{n+1} (referred to as dealer) runs on input a secret s and randomness r_{n+1}, while any other player P_i ($i \in [n]$) runs on input a randomness r_i. During this phase players can send (both private and broadcast) messages in multiple rounds.*
- **Reconstruction Phase $\mathsf{VSS}_{\mathsf{Recon}}$:** *Each shareholder sends its view v_i ($i \in [n]$) of the sharing phase to each other player, and on input the views of all players (that can include bad or empty views) each player outputs a reconstruction of the secret s.*

All computations performed by honest players are efficient. The computationally unbounded adversary can corrupt up to t players that can deviate from the above procedures. The following security properties hold.

- **Commitment:** *if the dealer is dishonest, then one of the following two cases happen: 1) during the sharing phase honest players disqualify the dealer, therefore they output a special value $\perp$ and will refuse to play the reconstruction phase; 2) during the sharing phase honest players do not disqualify the dealer, therefore such a phase determines a unique value s^* that belongs to the set of possible legal values that does not include $\perp$, which will be reconstructed by the honest players during the reconstruction phase.*
- **Secrecy:** *if the dealer is honest, then the adversary obtains no information about the shared secret before running the protocol* Recon*.*
- **Correctness:** *if the dealer is honest throughout the protocols, then each honest player will output the shared secret s at the end of protocol* Recon*.*

2.7 Information-Theoretical MPC and the MPC-in-the-Head Paradigm

We first recall *information-theoretically secure* MPCs and relevant notion that will be employed in the MPC-in-the-head paradigm shown later.

Information-Theoretical MPC. We now define MPC in the information-theoretical setting. Both the ideal and real executions are identical to the ones described in Section 2.5, except that the machines $\mathcal{A}$ and Sim are computationally unbounded now.

Definition 2.7.1 (Perfectly/Statistically-Secure MPC)**.** *Let $f : (\{0,1\}^*)^n \longrightarrow (\{0,1\}^*)^n$ be an n-ary functionality, and let Π be a protocol. We say that Π (n,t)*-perfectly (resp., statistically) securely *computes f if for every static, malicious, and (possibly-inefficient) probabilistic adversary $\mathcal{A}$ in the real model, there exists a probabilistic adversary* Sim *of* *comparable complexity (i.e., with runtime polynomial in that of $\mathcal{A}$) in the ideal model, such that for every $I \subset [n]$ of cardinality at most t, every $\vec{x} = (x_1, \ldots, x_n) \in (\{0,1\}^*)^n$ (where $|x_1| = \cdots = |x_n|$), and every $z \in \{0,1\}^*$, it holds that:*

$$\{\mathrm{REAL}_{\Pi,\mathcal{A}(z),I}(\vec{x})\} \overset{\text{i.d.}}{=\!=} \{\mathrm{IDEAL}_{f,\mathsf{Sim}(z),I}(\vec{x})\} \quad \big(\textit{resp.}, \{\mathrm{REAL}_{\Pi,\mathcal{A}(z),I}(\vec{x})\} \overset{\text{s}}{\approx} \{\mathrm{IDEAL}_{f,\mathsf{Sim}(z),I}(\vec{x})\}\big).$$

Recall that the MPC protocol from [BGW88] achieves (n,t)-perfect security (against static and malicious adversaries) with t being a constant fraction of n.

Theorem 2.7.1 ([BGW88])**.** *Consider a synchronous network with pairwise private channels. Then, for every n-ary functionality f, there exists a protocol that (n,t)-perfectly securely computes f in the presence of a static malicious adversary for any $t < n/3$.*

Consistency, Privacy, and Robustness. We now define some notation related to MPC protocols. Their roles will become clear when we discuss the MPC-in-the-head technique later.

Definition 2.7.2 (View Consistency)**.** *A view View_i of an honest player P_i during an MPC computation Π contains input and randomness used in the computation, and all messages received from and sent to the communication tapes. We have that a pair of views $(\mathsf{View}_i, \mathsf{View}_j)$ are consistent with each other if*

1. *Both corresponding players P_i and P_j individually computed each outgoing message honestly by using the random tapes, inputs and incoming messages specified in View_i and View_j respectively, and:*
2. *All output messages of P_i to P_j appearing in View_i are consistent with incoming messages of P_j received from P_i appearing in View_j (and vice versa).*

We further define the notions of correctness, privacy and robustness for multiparty protocols.

Definition 2.7.3 (Semi-Honest Computational Privacy)**.** *Let $1 \le t < n$, let Π be an MPC protocol, and let $\mathcal{A}$ be any* static, PPT, and semi-honest *adversary. We say that Π realizes a function $f : (\{0,1\}^*)^n \longrightarrow (\{0,1\}^*)^n$ with* semi-honest (n,t)-computational privacy *if there is a PPT simulator Sim such that for any inputs $x, w_1, \ldots, w_n$, every subset $T \subset [n]$ $(|T| \le t)$ of players corrupted by $\mathcal{A}$, and every D with circuit size at most $\mathsf{poly}(\lambda)$, it holds that*

$$\big|\Pr[D(\mathsf{View}_T(x, w_1, \ldots, w_n)) = 1] - \Pr[D(\mathsf{Sim}(T, x, \{w_i\}_{i \in T}, f_T(x, w_1, \ldots, w_n))) = 1]\big| \le \mathsf{negl}(\lambda), \quad (2.1)$$

where $\mathsf{View}_T(x, w_1, \ldots, w_n)$ is the joint view of all players.

Definition 2.7.4 (Statistical/Perfect Correctness)**.** *Let Π be an MPC protocol. We say that Π realizes a deterministic n-party functionality $f(x, w_1, \ldots, w_n)$ with* perfect (resp., statistical) *correctness if for all inputs $x, w_1, \ldots, w_n$, the probability that the output of some party is different from the output of some party is different from the actual output of f is 0 (resp., negligible in k), where the probability is over the independent choices of the random inputs $r_1, \ldots, r_n$ of these parties.*

Definition 2.7.5 (Perfect/Statistical Robustness)**.** *Assume the same setting as the previous definition. We say that Π realizes f with* (n,t)-perfect (resp., statistical) robustness *if in addition to being perfectly (resp., statistical) correct in the presence of a semi-honest adversary as above, it enjoys the following* robustness *property against any computationally unbounded malicious adversary corrupting a set T of at most t parties, and for any inputs $(x, w_1, \ldots, w_n)$: if there is no $(w'_1, \ldots, w'_n)$ such that $f(x, w'_1, \ldots, w'_n) = 1$, then the probability that some uncorrupted player outputs 1 in an execution of Π in which the inputs of the honest parties are consistent with $(x, w_1, \ldots, w_n)$ is 0 (resp., negligible in λ).*

MPC-in-the-Head. MPC-in-the-head is a technique developed for constructing black-box ZK protocols from MPC protocols [IKOS07]. Very roughly, the MPC-in-the-head idea is the following. Let $\mathcal{F}_{\text{ZK}}$ be the zero-knowledge functionality for an NP language. $\mathcal{F}_{\text{ZK}}$ takes as public input x and one share from each party, and outputs 1 iff the secret reconstructed from the shares is a valid witness. To build a ZK protocol, the prover runs in his head an execution of MPC w.r.t. $\mathcal{F}_{\text{ZK}}$ among n imaginary parties, each one participating in the protocol with a share of the witness. Then, it commits to the view of each party separately. The verifier obtains t randomly chosen views, checks that such views are "consistent" (see Definition 2.7.2), and accepts if the output of every party is 1. The idea is that, by selecting the t views at random, V will catch inconsistent views if the prover cheats.

We emphasize that, in this paradigm, a malicious prover decides the randomness of each virtual party, including those not checked by the verifier (corresponding to honest parties in the MPC execution). Therefore, MPC protocols with standard computational security may not protect against such attacks. We need to ensure that the adversary cannot force a wrong output even if it additionally controls the honest parties' random tape. The $(n, \lfloor n/3 \rfloor)$-perfectly secure MPC protocol in Theorem 2.7.1 suffices for this purpose (see also Remark 2.7.1).

One can extend this technique further (as in [GLOV12]), to prove a general predicate ϕ about an arbitrary value α. Namely, one can consider the functionality $\mathcal{F}_\phi$ in which party i participates with input a VSS share $[\alpha]_i$. $\mathcal{F}_\phi$ collects all such shares, and outputs 1 iff $\phi(\mathsf{VSS}_{\mathsf{Recon}}([\alpha]_1, \ldots, [\alpha]_n)) = 1$.

Remark 2.7.1 (Exact Security Requirements on the Underlying MPC.)**.** *To be more accurate, any MPC protocol that achieves* semi-honest (n, t)-computational privacy (see Definition 2.7.3) and (n, t)-perfect robustness (see Definition 2.7.5) *will suffice for the MPC-in-the-head application.*[3] *These two requirements are satisfied by any (n, t)-perfectly secure MPC (and, in particular, the one from Theorem 2.7.1).*

[3]It is also worth noting that the (n, t)-perfect robustness could be replaced with *adaptive (n, t)-statistical robustness*. See [IKOS07, Section 4.2] for more details.

Chapter 3

Black-Box Bounded-Concurrent MPC in Constant Rounds

3.1 Overview of Our Techniques

Before describing our approach, we first make some observations. We start by noting that in the context of concurrent secure computation, it is not possible to use rewinding-based simulation techniques since the simulator will have to provide additional outputs during rewinding but the ideal functionality does not deliver more than one output. This is in sharp contrast to concurrent zero-knowledge where the output is simply "yes" since the statement is in the language. While this can be salvaged for certain functionalities as shown by Goyal [Goy12], it is essential to move to straight-line simulators for general functionalities. In particular, in the bounded-concurrent setting we must move to non-black-box simulation techniques [Bar02].

Let us also note that in some situations, particularly in the setting of resettable zero-knowledge, a long line of work shows that it is possible to perform non-black-box simulation under one-way functions [BP12, BP13, CPS13]. Furthermore, a black-box version of these simulation techniques under one-way functions was obtained by Ostrovsky, Scafuro, and Venkitasubramaniam [OSV15]. It therefore seems possible to construct bounded-concurrent MPC under the minimal assumption of semi-honest OT in a black-box manner.[1] Unfortunately, this approach is flawed since *all* known non-black-box simulation techniques are based on rewinding and therefore cannot be applied to the concurrent MPC setting. It is also not at all clear if "straight-line" simulatable zero-knowledge based only on one-way functions can be constructed from known approaches. Therefore, we stress that *even without the requirement of black-box usage of primitives, constructing bounded-concurrent MPC under semi-honest OT only remains as a fascinating open problem.*

We therefore attempt to obtain a construction that exploits collision-resistant hash functions, in addition to the minimal assumption of semi-honest OTs. Toward this goal, we build upon techniques developed in the following two works:

1. Garg, Kiyoshima, and Pandey [GKP18] construct a constant-round black-box MPC pro-

[1]In some works, when the construction is black-box but the proof of security uses non-black-box techniques (as in this paper), this is referred to as a semi-black-box protocol.

tocol with SPS-security under polynomial hardness assumptions. The simulator works by extracting crucial information from adversary's messages via brute-force. The simulator is straight-line and such extraction steps are the only non-polynomial work in its execution.

2. Goyal et al. [GOSV14] present a black-box implementation of the non-black-box simulation techniques that rely on adversary's code [Bar01]. Such techniques often (and certainly those of [Bar01, GOSV14]) extend to situations where the adversary may receive arbitrary but *a-priori bounded* amount of external communication.

At a high level, our main idea is to use the simulation technique of [GOSV14] to replace the brute-force extraction steps in [GKP18] with polynomial-time extraction using adversary's code. The corresponding commitment scheme will be interactive. Since this simulator is polynomial time, we can hope to get bounded-concurrent MPC (in contrast to SPS MPC). Implementing this idea turns out to be rather involved. The fact that the commitment protocol is interactive brings its own issues of non-malleability and also interferes with some key proof steps in [GKP18] which rely on rewinding. It is also not enough that the underlying commitment protocol be extractable in a "bounded-concurrent" setting; instead we need a more flexible notion (that, roughly speaking, mirrors straight-line simulation).

Although we have non-black-box simulation techniques at our disposal, we do not rely on the multiple slots approach of Pass [Pas04b] to build simulation soundness directly into our protocols. Instead, by relying on the techniques in the aforementioned two works, we obtain a more modular approach where non-malleability and simulation soundness are obtained with the help of an underlying non-malleable commitment. In this sense, the structure of our bounded-concurrent protocol is fundamentally different from that of [Pas04b] to achieve bounded-concurrent MPC. We now provide more details.

The high-level structure of our protocol is similar to that of [GKP18] where the MPC protocol is obtained in two steps. First, we obtain a (constant-round) black-box construction of a bounded-concurrent OT protocol. Next, we compose this OT protocol with an existing constant-round OT-hybrid UC-secure MPC protocol. We elaborate on each step below. We remark that we consider concurrent security in the interchangeable-roles setting. So, in the case of OT, the adversary can participate in a session as the sender while concurrently participating in another session as the receiver.

3.1.1 Black-Box (Constant-Round) Bounded-Concurrent OT

Our OT protocol is very similar to the OT protocol of [GKP18] (which in turn is based on the high-level cut-and-choose structure of [LP12] inspired from [HIK$^+$11, CDMW09, Wee10]) except that we will implement the basic commitment scheme using a "straight-line extractable" commitment (with some other properties that we will discussion soon). At a high level, the OT protocol of [GKP18] proceeds as follows:

1. The protocol is based on cut-and-choose techniques. Therefore, as the first step of the protocol, the sender S and the receiver R commit to their challenges for future stages in advance. This step uses a two-round statistically binding commitment scheme Com. This step avoids selective opening attacks. The ideal-world simulator can extract these challenges by brute-force to perform the simulation. This is the only non-polynomial time step of this simulator (and the one we wish to replace).

2. Next, S and R execute many instances of a semi-honest OT protocol in parallel, where in each instance S and R use the inputs and the randomness that are generated by a coin-tossing protocol.
3. Next, S and R use a non-malleable commitment scheme NMCom to set up a "trapdoor statement" which, roughly speaking, commits a witness to the fact that the trapdoor statement is false. This step, following [GGJS12], makes it possible to commit to a false witness in the security proof while ensuring (due to non-malleability of NMCom) that the adversary still continues to commit to a correct witness (so that his statement is still false). The step is performed by modifying different stages of **one session at a time**. This ensures that changes in one interactive part of the protocol are not affected by what happens in later stages of that same session.
4. Finally, S and R use OT combiner which allows them to execute an OT with their real inputs securely when most of the OT instances in the previous steps are correctly executed. To check that most of the OT instances in the previous steps were indeed correctly executed, S and R do use cut-and-choose where S (resp., R) chooses a constant fraction of the OT instances randomly and R (resp., S) reveals the input and randomness that it used in those instances so that S (resp., R) can verify whether R executed those instances correctly.

3.1.1.1 Replacing Com with Straight-Line Extractable Commitment

Our goal is to eliminate brute-force extraction using code of the adversary. In doing so, we have to ensure that (1) the interactive nature of the commitment protocol so obtained does not result into new malleability issues in the proof; and (2) the extraction step can be done in a modular fashion (especially in straight-line) so that we can keep the overall proof structure of [GKP18] where one session is modified at a time.

As a starting point, let us consider the Barak-Lindell extractable commitment scheme [BL02]. In their construction, the committer C first sends an enhanced trapdoor permutation f.[2] Then the two parties involve in the following 3-step coin tossing: (1) R sends a commitment $\mathsf{Com}(r_1)$ to a random string r_1; (2) C replies with a random string r_2; (3) R then sends r_1 with a ZK argument on the fact that this r_1 is indeed the random string he committed in step (1). Both parties learn the value $r = r_1 \oplus r_2$ as the output of the coin tossing. To commit to a (single-bit) message σ, C sends σ masked by the hard-core bit of $f^{-1}(r)$. An extractor can use the ZK simulator to bias the coin-tossing result to some value r', for which it knows the preimage of $f^{-1}(r')$. Thus, it can extract the committed value.

Straight-Line Extraction. To adapt the above scheme for our purpose, we need to ensure that the construction is black-box *and* that the committed value can be extracted in a straight-line fashion. Toward this goal, we replace R's commitment and ZK argument with the protocol of Goyal et al. [GOSV14]. More specifically, [GOSV14] provides a "commit-and-prove" primitive Π_{ZK} where:

[2] In their original construction, C sends a trapdoor permutation (TDP) f and then proves in zero-knowledge that f is indeed a valid TDP. To make this step black-box, C can send an enhanced TDP instead (without the need of ZK proof).

– they provide a (non-interactive statistically-binding) commitment scheme[3] called VSSCom using which one can send a commitment y to a string x;

– and later, prove to a verifier, that "y is a commitment to string x such that $\phi(x) = 1$" where ϕ is an arbitrary function.

In particular, ϕ is chosen to be the NP-relation for an NP-complete language in [GOSV14] to get a black-box version of Barak's result [Bar01].

In our case, we will choose ϕ to be the identity function $I_x(\cdot)$.[4] Therefore, the Barak-Lindell commitment protocol mentioned above can be implemented in a black-box manner by ensuring that: (1) R uses VSSCom to prepare the commitment to r_1, and (2) protocol Π_{ZK} is the aforementioned proof protocol with $\phi := I_{r_1}(\cdot)$.

At a high level, this approach meets our needs for a black-box construction that supports straight-line extraction. But more caution is needed to handle the actually simulation as we are in the (bounded) *concurrent* setting. We will address this concern in Section 3.1.1.2.

Removing TDPs. Since we aim to have a construction assuming only semi-honest OTs (and CRHFs), we also want to remove the reliance on the (enhanced) TDPs. As the first attempt, we ask C to secret-share the message σ to λ random shares using exclusive-or. Then let the receiver learn through a special OT (e.g. an $n/2$-out-of-n OT) half of these shares. Next, we invoke the above (black-box) version of coin-tossing in Barak-Lindell protocol to determine another $n/2$ shares that C will decommit to. Due to the pseudo-randomness of the coin-tossing result, R will learn the the shares that "complement" what he learned through OT with only negligible probability. Thus, we can hope to achieve (computational) hiding. Meanwhile, an extractor could always bias the coin-tossing result to the complement shares, thus allowing it to extract the value σ.

However, there are several issues with this approach. First, the sender's (committer's) input to the OT must be the decommitment information to the secret shares. Otherwise, a malicious sender can use arbitrary values in the OT execution, which will disable our extraction strategy.[5] Also, this construction suffers from *selective opening attacks* (SOAs) as the values in the commitments are correlated. It is not clear how we can use standard techniques (e.g. asking R to commit to his challenges in advance, or using another coin-tossing to determine his challenges) to get rid of SOAs. This is because we need to keep R's challenges in this stage hidden from C (to ensure extractability).

To solve this problem, we let C commit to 2λ secret shares of σ, denoted as $\{\mathsf{Com}(s_{i,b})\}_{i\in[\lambda],b\in\{0,1\}}$. Then λ 1-out-of-2 OT instances are executed in parallel, where R learns (the decommitment to) one share out of $(s_{i,0}, s_{i,1})$ in the i-th OT. Next, we can use the Barak-Lindell coin tossing to determine an λ-bit string $r = r_1\|\ldots\|r_\lambda$. Finally, C decommits to $\{\mathsf{Com}(s_{i,r_i})\}_{i\in[\lambda]}$. In this construction, R's input to (a single) OT can be guessed correctly with probability 1/2.

[3]In [GOSV14], this commitment was required to be statistically-hiding. But it can be replaced with a statistically-binding scheme if certain modifications are made to the proof phase. See Remark 3.3.1 for more details.

[4]Note $I_x(y) = 1$ if and only if $y = x$ is well defined and the "code" of I_x requires only the knowledge of x.

[5]Note that we cannot ask the committer to prove in zero-knowledge that he uses the committed shares as sender's input in the OT execution, because such proof will make non-black-box use of both the commitment and OT.

By a careful design of hybrids, we show this is sufficient to for us to get rid of SOAs, thus allowing us to prove hiding property (see Section 3.4). Moreover, the extractor can still learn all the shares by biasing r_i to the complement to its input in the i-th OT instance (for all $i \in [\lambda]$).

Merging with [GKP18]. Finally, to ensure that the interactive nature does not create non-malleability issues, we will ask each party to commit to a long-enough random string, using the above extractable commitment. This step is done as the foremost step in our OT protocol (called "Step 0"). Then each party will use the long random string as one-time pad to "mask" the values that they want to commit to during the execution of our OT protocol. Now, we can rely on the structure of the hybrid proof of [GKP18], which first deals with all stages of a given session and then moves on to the next session in a specific order (determined by the transcript). The key observation here is that since Step 0 is performed ahead of all other steps for a fixed session s, changes in later stages of s cannot affect what happens in Step 0 (for example, issues of malleability and simulation-soundness do not arise). Furthermore, since any rewinding-based proofs of [GKP18] are only relevant to later stages, they do not rewind Step 0 of sessions s.

Remark 3.1.1. *Ostrovsky et al. [OSV15] showed how to achieve the same as [GOSV14] while relaxing the assumption from CRHFs to one-way functions (OWFs). But we cannot use their approach (or any of the prior approaches that perform non-black-box simulation under OWFs) since simulators in these approaches are not straight-line. It uses both the adversary's code* and *rewinding to get a OWF-based construction.*

3.1.1.2 Robust-ZK for Dealing with Bounded Concurrency

The final issue that we need to address is how the non-black-box simulation will actually be performed corresponding to protocol Π_{ZK} (in Step 0) mentioned above. The main issue is that there are concurrently many sessions of Π_{ZK} executing simultaneously. In particular, if there are m sessions of OT protocol, then there will be $\ell = 2m$ sessions of Π_{ZK}. Simply replacing the prover with the non-black-box simulator may not result in polynomial-time simulation.

An immediate idea is that if Π_{ZK} is *bounded-concurrent* ZK for up to ℓ sessions, then we can use the *concurrent* non-black-box simulator to simulate Step 0 of all m sessions of the OT protocol at once. This allows us to bias coin-tossing for all m sessions and then we can design hybrids exactly as in [GKP18].

Unfortunately, bounded-concurrent ZK only guarantees *self* composition; i.e., it can only deal with messages of protocol Π_{ZK}. In our case, Π_{ZK} is part of a larger protocol execution and the adversary receives messages from different stages of all sessions. We thus need a more robust notion of non-black-box simulation which, roughly speaking, (a) is straight-line, and (b) enables bounded-concurrent composition of ZK protocols *even in the presence of external messages* as long as the total communication outside the ZK protocol is a-priori bounded.

We formulate this notion explicitly in Section 3.3 and call it *robust zero-knowledge*. The notion requires that the view of a (standalone) verifier V^* who interacts with an external party B can be simulated by a simulator S only on input the code of V^*. The simulator is not allowed to rewind V^* or B. However, both B and S are allowed to see each others messages (which is essential to make sure that many concurrent instances of the simulators compose seamlessly). This yields a notion that is similar in spirit to UC zero-knowledge [Can01] and implies bounded-concurrent ZK.

We remark that most ZK protocols based on non-black-box simulation, with suitable adjustment of parameters, can actually handle *arbitrary* external messages (and not just the messages of the same protocol) without any modification. This observation was first used in Barak's original work [Bar01], and finds applications in other places [BL02, PR03, Pas04b]. In particular, it also holds for the protocol of Goyal et al. [GOSV14] and is implicit in their security proof. Thus, these protocols already achieve the (bounded) robust-ZK notion. Robust-ZK is just a convenient tool to help in the hybrid proofs.

By setting the parameters of Π_{ZK} so that it is ℓ-robust-ZK allows us to replace the provers of Π_{ZK} with simulator instances in Step 0 of any given session s while maintaining the overall structure and sequence of hybrids in [GKP18] where stages of one session are handled at any given time. This gives us m-bounded concurrent OT.

3.1.2 Composition of OT with OT-hybrid MPC

The final step of our construction is the same as in [GKP18]. Namely, we compose our bounded-concurrent OT protocol with a OT-hybrid UC-secure MPC protocol (i.e., replace each invocation of the ideal OT functionality in the latter with an execution of the former), thereby obtaining a MPC protocol in the plain model. While selecting the parameters, we have to ensure we adjust the parameters of Π_{ZK} to allow long enough messages so that simulation can be performed for the MPC protocol instead of the OT protocol. Since we only proved bounded-concurrent self composition for OT (not full UC-security), we do not get a proof for the MPC protocol right away. Hence, we prove the security by analyzing the MPC protocol directly. In essence, what we do is to observe that the security proof for our OT protocol (which consists of a hybrid argument from the real world to the ideal world) still works even after the OT protocol is composed with a OT-hybrid MPC protocol.

3.2 Preliminaries

In the following, we present additional preliminaries that are necessary for this chapter.

3.2.1 Shamir's Secret Sharing

We first recall Shamir's secret sharing scheme. (In this chapter, we use only the $(6n+1)$-out-of-$10n$ version of it.) To compute a $(6n+1)$-out-of-$10n$ secret sharing $\boldsymbol{s} = (s_1, \ldots, s_{10n})$ of a value $v \in GF(2^n)$, we choose random $a_1, \ldots, a_{6n} \in GF(2^n)$, let $p(z) \overset{\text{def}}{=} v + a_1 z + \cdots + a_{6n} z^{6n}$, and set $s_i := p(i)$ for each $i \in [10n]$. Given $\boldsymbol{s}$, we can recover v by obtaining polynomial $p(\cdot)$ through interpolation and then computing $p(0)$. We use $\mathsf{Decode}(\cdot)$ to denote the function that recovers v from $\boldsymbol{s}$ as above.

For any positive real number $x \leq 1$ and any $\boldsymbol{s} = (s_1, \ldots, s_{10n})$ and $\boldsymbol{s'} = (s'_1, \ldots, s'_{10n})$, we say that $\boldsymbol{s}$ and $\boldsymbol{s'}$ are x-*close* if $|\{i \in [10n] \text{ s.t. } s_i = s'_i\}| \geq x \cdot 10n$. If $\boldsymbol{s}$ and $\boldsymbol{s'}$ are not x-close, we say that they are $(1-x)$-*far*. Since the shares generated by $(6n+1)$-out-of-$10n$ Shamir's secret sharing scheme are actually a codeword of the Reed-Solomon code with minimum relative distance 0.4, if a (possibly incorrectly generated) sharing $\boldsymbol{s}$ is 0.8-close to a valid codeword $\boldsymbol{w}$, we can recover $\boldsymbol{w}$ from $\boldsymbol{s}$ efficiently by using, for example, the Berlekamp-Welch algorithm.

3.2.2 Non-Malleable Commitment Schemes.

We recall the definition of non-malleable commitment schemes from [LP09]. Let $\langle C, R\rangle$ be a tag-based commitment scheme (i.e., a commitment scheme that takes a n-bit string (a *tag*) as an additional input). For any man-in-the-middle adversary $\mathcal{M}$, consider the following experiment. On input security parameter 1^n and auxiliary input $z \in \{0,1\}^*$, $\mathcal{M}$ participates in one left and one right interactions simultaneously. In the left interaction, $\mathcal{M}$ interacts with the committer of $\langle C, R\rangle$ and receives a commitment to value v using identity $\mathsf{id} \in \{0,1\}^n$ of its choice. In the right interaction, $\mathcal{M}$ interacts with the receiver of $\langle C, R\rangle$ and gives a commitment using identity $\widetilde{\mathsf{id}}$ of its choice. Let $\widetilde{v}$ be the value that $\mathcal{M}$ commits to on the right. If the right commitment is invalid or undefined, $\widetilde{v}$ is defined to be $\bot$. If $\mathsf{id} = \widetilde{\mathsf{id}}$, value $\widetilde{v}$ is also defined to be $\bot$. Let $\mathsf{mim}(\langle C, R\rangle, \mathcal{M}, v, z)$ be a random variable representing $\widetilde{v}$ and the view of $\mathcal{M}$ in the above experiment.

Definition 3.2.1. *A commitment scheme $\langle C, R\rangle$ is* non-malleable *if for any* PPT *adversary $\mathcal{M}$, the following are computationally indistinguishable.*

– $\{\mathsf{mim}(\langle C, R\rangle, \mathcal{M}, v, z)\}_{n\in\mathbb{N}, v\in\{0,1\}^n, v'\in\{0,1\}^n, z\in\{0,1\}^*}$

– $\{\mathsf{mim}(\langle C, R\rangle, \mathcal{M}, v', z)\}_{n\in\mathbb{N}, v\in\{0,1\}^n, v'\in\{0,1\}^n, z\in\{0,1\}^*}$

The above definition can be generalized naturally so that the adversary gives multiple commitments *in parallel* in the right interaction. The non-malleability in this generalized setting is called *parallel non-malleability.* (It is known that this "one-many" definition implies the "many-many" one, where the adversary receives multiple commitments in the left session [LPV08].)

Robust Non-Malleability. We next recall the definition of k-robust non-malleability (a.k.a. non-malleability w.r.t. k-round protocols) [LP09]. Consider a man-in-the-middle adversary $\mathcal{M}$ that participates in one left interaction—communicating with a machine B—and one right interaction—communicating with a receiver a commitment scheme $\langle C, R\rangle$. As in the standard definition of non-malleability, $\mathcal{M}$ can choose the identity in the right interaction. We denote by $\mathsf{mim}^{B,\mathcal{M}}_{\langle C,R\rangle}(y, z)$ the random variable consisting of the view of $\mathcal{M}(z)$ in a man-in-the-middle execution when communicating with $B(y)$ on the left and an honest receiver on the right, combined with the value $\mathcal{M}(z)$ commits to on the right. Intuitively, $\langle C, R\rangle$ is non-malleable w.r.t. B if $\mathsf{mim}^{B,\mathcal{M}}_{\langle C,R\rangle}(y_1, z)$ and $\mathsf{mim}^{B,\mathcal{M}}_{\langle C,R\rangle}(y_2, z)$ are indistinguishable whenever interactions with $B(y_1)$ and $B(y_2)$ are indistinguishable.

Definition 3.2.2. *Let $\langle C, R\rangle$ be a commitment scheme and B be a* PPT *ITM. We say that a commitment scheme $\langle C, R\rangle$ is* non-malleable w.r.t. B *if the following holds: For every two sequences $\{y^1_n\}_{n\in\mathbb{N}}$ and $\{y^2_n\}_{n\in\mathbb{N}}$ such that $y^1_n, y^2_n \in \{0,1\}^n$, if it holds that for any* PPT *ITM*

$\mathcal{A}$,

$$\left\{\langle B(y_n^1), \mathcal{A}(z)\rangle(1^n)\right\}_{n\in\mathbb{N}, z\in\{0,1\}^*} \overset{c}{\approx} \left\{\langle B(y_n^2), \mathcal{A}(z)\rangle(1^n)\right\}_{n\in\mathbb{N}, z\in\{0,1\}^*} \quad ,$$

it also holds that for any PPT *man-in-the-middle adversary $\mathcal{M}$,*

$$\left\{\mathsf{mim}_{\langle C,R\rangle}^{B,\mathcal{M}}(y_1, z)\right\}_{n\in\mathbb{N}, z\in\{0,1\}^*} \overset{c}{\approx} \left\{\mathsf{mim}_{\langle C,R\rangle}^{B,\mathcal{M}}(y_2, z)\right\}_{n\in\mathbb{N}, z\in\{0,1\}^*} \quad .$$

$\langle C, R\rangle$ is *k-robust* if $\langle C, R\rangle$ is non-malleable w.r.t. any machine that interacts with the adversary in k rounds. We define parallel k-robustness naturally.

Black-Box Instantiation. There exists a constant-round black-box construction of a parallel (actually, concurrent) non-malleable commitment scheme based on one-way functions [GLOV12]. Furthermore, Garg, Kiyoshima, and Pandey [GKP18] show that any parallel non-malleable commitment can be transformed into a parallel k-robust non-malleable one in the black-box way by using collision-resistant hash functions (more precisely, by using statistically hiding commitment schemes, which can be constructed from collision-resistant hash functions). If k is constant, the round complexity of their transformation increases only by a constant factor in this transformation. Thus, there exists a $O(1)$-round parallel $O(1)$-robust nonmalleable commitment scheme assuming the existence of CRHFs [GLOV12, GKP18].

3.2.3 Bounded-Concurrent MPC with Interchangeable Roles

We recall the definition of m-bounded concurrent secure computation. Parts of this section are taken verbatim from [Pas04b] with minor modification, following [GGS15], to allow for *interchangeable* roles; these in turn are a slight generalization of "security with abort and no fairness" of [GL02] and concurrent secure two-party computation with adaptive inputs of [Lin04]. The basic formulation and setup of secure computation follows [GL91, MR92, Bea92, Can00a].

We consider the case of self composition where m simultaneous executions of the same MPC protocol Π take place. We will consider security against *interchangeable* roles where a party controlled by the adversary can play different roles in different sessions (see description below). We will only consider the *malicious* and *static* setting where the set of corrupted parties is fixed at the beginning of the protocol and the corrupted parties execute the instructions provided by the adversary. The scheduling of message delivery is decided by the adversary. Since security against interchangeable roles is impossible without identities, we assume each party has a unique identity $\mathsf{id} \in \{0,1\}^n$. Since we do not consider fairness, the adversary will always receive its own output and can then decide when (if at all) the honest parties will receive their output.

Multi-Party Computation. A multi-party protocol problem for k parties $P_1, \ldots, P_k$ is cast by specifying a random process that maps vectors of inputs to vectors of outputs (one input and one output for each party). We refer to such a process as a k-ary functionality and denote it $f : (\{0,1\}^*)^k \to (\{0,1\}^*)^k$, where $f = (f_1, ..., f_k)$. That is, for every vector of inputs $\overline{x} = (x_1, ..., x_k)$, the output-vector is a random variable $\big(f_1(\overline{x}), ..., f_k(\overline{x})\big)$ ranging over vectors of strings. The output of the i'th party (with input x_i) is defined to be $f_i(\overline{x})$. In the context of concurrent composition, each party actually uses many inputs (one for each execution) and these may be chosen adaptively based on previous outputs. The fact that m-bounded concurrency is considered relates to the allowed scheduling of messages by the adversary in the protocol executions; see the description of the real model below.

Concurrent Execution in the Ideal Model. Next, we describe the concurrent execution of the protocol in the ideal world. Unlike the stand-alone setting, here the trusted party computes the functionality many times, each time upon different inputs.

Let $\Pi := (P_1, \dots, P_k)$ be an MPC protocol for computing a k-ary functionality f and n be the security parameter. For simplicity we assume that the length of the inputs of each party is n. In total, let there be N parties: $Q_1, \dots, Q_N$ and let P_i^j denote the party playing the role of P_i in session j (for $i \in [k], j \in [m]$). The adversary can corrupt an arbitrary subset of these parties.

Let $I \subset [N]$ denote the subset of corrupted parties. An ideal execution with an adversary who controls the parties I proceeds as follows:

1. **Inputs:** The inputs of the parties $Q_1, ..., Q_N$ in each session j are determined using PPT machines $M_1, ..., M_k$ which take as input the session number j, some inputs $x_1, ..., x_N$, and the outputs that were obtained from executions that have already concluded. Note that the number of previous outputs range from zero (when no previous outputs have been obtained) to some polynomial in n that depends on the number of sessions initiated by the adversary.

2. **Session initiation:** When the adversary initiates the session number $j \in [m]$ by sending a ($\mathsf{start\text{-}session}$, j) to the trusted party, the trusted party sends ($\mathsf{start\text{-}session}$, j) to parties P_i^j where $i \in [k]$.

3. **Honest parties send inputs to trusted party:** Upon receiving ($\mathsf{start\text{-}session}$, j) from the trusted party, each honest party P_i^j applies its input-selecting machine M_i to its initial input x_i, the session number j and its previous outputs, and obtains a new input $x_{i,j}$. In the first session $x_{i,1} = M_i(x, 1)$. In later sessions j, $x_{i,j} = M_i(x, j, \alpha_{i,1} ... \alpha_{i,\omega})$ where ω sessions have concluded and the outputs of P_i^j were $\alpha_{i,1}, ..., \alpha_{i,\omega}$. Each honest party P_i^j then sends $(j, x_{i,j})$ to the trusted party.

4. **Corrupted parties send inputs to trusted party:** Whenever the adversary wishes it may ask a corrupted party P_i^j to send a message $(j, x'_{i,j})$ to the trusted third party, for any $x'_{i,j} \in \{0,1\}^n$ of its choice. A corrupted party P_i^j can send the pairs $(j, x'_{i,j})$ in any order it wishes and can also send them *adaptively* (i.e., choosing inputs based on previous outputs). The only limitation is that for any j, at most one pair indexed by j can be sent to the trusted party.

5. **Trusted party answers corrupted parties:** When the trusted third party has received messages $(j, x'_{i,j})$ from all parties (both honest and corrupted) it sets $\overline{x}_j = (x'_{1,j}, ..., x'_{k,j})$. It then computes $f(\overline{x}_j)$ and sends $(j, f_i(\overline{x}'_j))$ to every corrupted P_i^j.

6. **Adversary instructs the trusted party to answer honest parties:** When the adversary sends a message of the type ($\mathsf{send\text{-}output}$, j, i) to the trusted party, the trusted party directly sends $(j,\ f_i(x'_j))$ to party P_i^j. If all inputs for session j have not yet been received by the trusted party the message is ignored. If the output has already been

delivered to the honest party, or i is the index so that P_i^j is a corrupted party, the message is ignored as well.

7. **Outputs:** Each honest party always outputs the vector of outputs that it received from the trusted party. The corrupted parties may output an arbitrary (probabilistic polynomial-time computable) function of its initial input and the messages obtained from the trusted party.

Let $f : (\{0,1\}^*)^k \rightarrow (\{0,1\}^*)^k$ be a k-ary functionality, where $f = (f_1, ..., f_k)$. Let S be a non-uniform PPT machine (representing the ideal-model adversary) and let $I \subset [N]$ (the set of corrupted parties) be such that for every $i \in I$, the adversary S controls Q_i. Then the *ideal execution* of f with security parameter n, input-selecting machines $M = M_1, ..., M_k$, initial inputs $\overline{x} = (x_1, ..., x_N)$ and auxiliary input z to S, denoted $\text{IDEAL}_{f,I,S,M}(n, \overline{x}, z)$, is defined as the output vector of the parties and S resulting from the ideal process described above.

We remark that the definition of the ideal model includes the bound m on the concurrency although it is possible to define it without it.

Execution in the Real Model. We next consider the execution of Π in the real world. We assume that the parties communicate through an *asynchronous* fully connected and authentic point-to-point channel but without guaranteed delivery of messages.

Let f, I be as above and let Π be a multi-party protocol for computing f. Furthermore, let $\mathcal{A}$ be a non-uniform PPT machine such that for every $i \in I$, the adversary $\mathcal{A}$ controls Q_i. Then, the real m-bounded concurrent execution of Π with security parameter n, input-selecting machines $M = M_1, ..., M_k$, initial inputs $\overline{x} = (x_1, ..., x_N)$ and auxiliary input z to $\mathcal{A}$, denoted $\text{REAL}^m_{\Pi,I,A,M}(n, \overline{x}, z)$, is defined as the output vector of the honest parties and the adversary $\mathcal{A}$ resulting from the following process. The parties run concurrent executions of the protocol, where every party initiates a new session whenever it receives a start-session from the adversary. The honest parties then apply their input-selecting machines to their initial input, the session number and their previously received outputs, and obtain the input for this new session. The scheduling of all messages throughout the executions is controlled by the adversary.

Security as Emulation of a Real Execution in the Ideal Model. The security of Π under bounded composition is defined by saying that for every real-model adversary there exists an ideal model adversary that can simulate an execution of the secure real-model protocol. Formally:

Definition 3.2.3 (m-Bounded Concurrent Security in the Malicious Model)**.** *Let $m = m(n)$ be a polynomial and let f, k, N and Π be as above. Protocol Π is said to securely compute f under m-*bounded concurrent composition *if for every real-model non-uniform* PPT *adversary $\mathcal{A}$, there exists an ideal-model non-uniform probabilistic expected polynomial-time adversary S, such that for all input-selecting machines $M = M_1, ..., M_k$, every $z \in \{0,1\}^*$, every $\overline{x} = (x_1, ..., x_N)$, where $x_1, ..., x_N \in \{0,1\}^n$ and every $I \subset [N]$,*

$$\left\{\text{IDEAL}_{f,I,S,M}(n, \overline{x}, z)\right\}_{n\in\mathbb{N}} \overset{\text{c}}{\approx} \left\{\text{REAL}^m_{\Pi,I,A,M}(n, \overline{x}, z)\right\}_{n\in\mathbb{N}}$$

That is, concurrent executions of Π with $\mathcal{A}$ cannot be distinguished from concurrent invoca-

tions of f with S in the ideal model.

3.3 Robust Zero-Knowledge Commit-and-Prove Protocols

Goyal et al. [GOSV14] present a new non-black-box zero-knowledge argument for NP. Their protocol (with slight modification for the "commit-and-prove" form) is presented in Protocol 3.3.1.

Protocol 3.3.1: ℓ-Robust Commit-and-Prove for ϕ [GOSV14]

Common Input: Security parameter 1^n, robustness parameter ℓ, property ϕ

Auxiliary Input to C': String $w \in \{0,1\}^n$ to be committed.

Commit Phase:

1. C' generate VSS representation of w: $\mathsf{VSS}^w = (w_1^{\mathsf{VSS}}, ..., w_n^{\mathsf{VSS}})$.
2. C' creates commitments to each share with independent randomness $\rho_i \in \{0,1\}^n$, to get $c_i = \mathsf{Com}(w_i^{\mathsf{VSS}}; \rho_i)$ for $i = 1, ..., n$.
3. C' sends $\mathsf{VSSCom}(w) := (c_1, ..., c_n)$.

   Comment: Note that ℓ, ϕ are not required in this phase. In [GOSV14], the commit-phase is actually a part of the "proof phase" since the goal is to describe a system for NP*. We choose this form to emphasize the commit-and-prove nature of their protocol.*

Proof Phase:

1. Trapdoor-generation:

   (a) C' runs $\mathsf{BBCom}(0^n)$ with R'. Let z be the commitment so obtained.

   (b) R' sends a random string r **of length** $n + \ell(n)$. The public theorem a is defined as: $a = (z, r, t)$. This message is referred to as the **long message**.

2. Actual proof for ϕ:

   (a) Commitment of PCPP: C' runs $\mathsf{BBCom}(0^n)$ and sends the commitments.

   (b) PCPP Queries: R' sends random tapes $r_1, ..., r_{\ell_d}$ from which C' and R' compute $(q_i^j, p_i^j) = Q_{\mathsf{pcpx}}(a, r_j, i)$ with $i \in [k]$, where k is the security parameter for the PCPP. Let $I_j^M = \{q_1^j, ..., q_k^j\}$ and $I_j^\pi = \{p_1^j, ..., p_k^j\}$.

   (c) Proof: C' runs $\mathsf{BBProve}(\psi, I^M, I^\pi)$, where the predicate ψ is true iff:

      – D_{pcpx} outputs 1 on selected positions of M and π; **or**
      – There exist $\{(w_i^{\mathsf{VSS}}, \rho_i)\}_{i=1}^n$ such that $c_i = \mathsf{Com}(w_i^{\mathsf{VSS}}; \rho_i)$ for all i and $\phi\left(\mathsf{Recon}(w_1^{\mathsf{VSS}}, ..., w_n^{\mathsf{VSS}})\right) = 1$.

      R' accepts the proof if and only if the verifier of $\mathsf{BBProve}$ accepts.

Let us briefly recall how their protocol works. They first construct a black-box size-hiding commit-and-prove protocol ($\mathsf{BBCom}, \mathsf{BBProve}$). In Protocol 3.3.1, the committer commits to the secret shares of the witness via BBCom. The Proof Phase combines PCP of Proximity (PCPP) [BGH+04] and Barak's non-black-box ZK protocol [Bar01]. The committer C' (the prover) first sends z which is supposed to be a commitment to a Turing machine M. An honest prover will just commit to 0^n. Once R' replies with a string r, the trapdoor theorem is set to a of the pair language $\mathcal{L}_{\mathcal{P}} = \{(a := (z, r, t), Y) : \exists M \in \{0,1\}^* \text{ s.t. } Y \leftarrow \mathsf{ECC}(M),$ and $M(z) = r$ within t steps.$\}$ (where $\mathsf{ECC}(\cdot)$ is a binary error correcting code tolerating a constant fraction $\delta > 0$ errors). Then C' uses $\mathsf{BBProve}$ to prove either the trapdoor theorem is true or $\phi(w) = 1$.

Note that the proof for the trapdoor theorem is conducted via PCPP. Specifically, commitment to PCPP proof π is sent to R' (honest prover commits to 0^n, as shown in 2-(a) of Proof Phase). R' generates PCPP queries on Y (the private theorem) and π by running algorithm Q_{pcpx}. C' then proves that the PCPP decision algorithm D_{pcpx} verifies to 1. Details of component protocols such as $\mathsf{BBCom}, \mathsf{BBProve}$, etc. are not necessary and omitted; see [GOSV14] for their details.

This protocol makes only black-box use of CRHF; it is also public-coin, constant-rounds, and has negligible soundness error. In fact, it enjoys the following properties:

(i) The protocol is actually a "commit-and-prove" protocol for arbitrary polynomial-size circuits ϕ. That is, it consists of two phases: in the "commit" phase, the committer commits an arbitrary string $w \in \{0,1\}^n$ using a special commitment scheme called VSSCom, and later, in the "proof" phase, it can prove in zero-knowledge that the committed string satisfies ϕ; i.e., $\phi(w) = 1$ where w is uniquely determined from the transcript of the commit phase. For concreteness, the "commit-and-prove" form of [GOSV14] ZK is depicted in Protocol 3.3.1.[6]

(ii) To prove zero-knowledge, the simulator relies on Barak's technique of committing the verifier's code [Bar01]. Consequently, the protocol inherits several properties of Barak's original protocol (e.g., public-coin and constant rounds). In particular, the protocol has a "preamble" phase where the verifier sends a random string r; the simulator is "straight-line" *even in the presence of arbitrary (external) communication of a-priori bounded length* $\ell(n)$ provided that $|r|$ is sufficiently bigger than $\ell(n)$.

Remark 3.3.1 (On the Hiding Property of VSSCom). *In the* **Commit Phase** *of Protocol 3.3.1, we define* VSSCom, *which consists of* statistically-binding *commitments* Com *on each* VSS *shares of the value to be committed to (the witness w in our protocol). However, in the original construction of the [GOSV14] ZK, the underlying* Com *actually needs to be* statistically-hiding. *This is because that their construction relies on the MPC-in-the-head technique, where a subset (verifier's challenge set) of the commitments are revealed to the verifier for the view-consistency checking. Moreover, the security of their construction relies on the hiding of the remaining unopened commitments. Since the challenge set is picked by the verifier, a statically-hiding commitments must be used to resolve the selective-opening*

[6]The protocol for proving $x \in L$ for $L \in \mathsf{NP}$ is obtained by setting w to be a witness for x (under an appropriate relation R for L) and committing to it as the first step of the proof using "commit" phase, followed by the "proof" phase for $\phi(\cdot) := R(x, \cdot)$.

problem [Hof11].

We remark that there are two alternative ways to avoid the selective-opening problem, while relying only on statistically-binding commitment:

(1) Ask V to commit to the challenge sets before the P's first message, and to decommit to the challenge sets once it receives P's first message. This approach is taken by, e.g., [PW09, GLOV12, Kiy14].

(2) After P's first message, determine the challenge set by a coin-tossing protocol between P and V, instead of letting V pick the challenge set. This approach appears in [Lin13, CLP20a] (see [CLP20b, Section 4.1] for a detailed demonstration in the MPC-in-the-head setting).

Both of these approaches can be taken if one wants to replace the statistically-hiding commitment in VSSCom *and* BBCom*, while maintaining the security of the [GOSV14] construction. But they were not exploited as [GOSV14] pursued a public-coin construction. As another concern, these approaches only lead to* computational *ZK property, while the original construction in [GOSV14] is* statistical *ZK.*

In contrast, we are able to make use of these approaches in the current work. We take approach (2) as it not only gives a cleaner construction, but also maintains the (weak) Proof-of-Knowledge property of the original [GOSV14] ZK. Concretely, we modify the BBProve *such that V's challenge set will be determined by the following coin-tossing:*

- *V first sends an extractable commitment to a random string r_1,*
- *P responds with a random string r_1*
- *V then sends r_1 with the decommitment information.*

Other parts of BBProve *remain unchanged, except that the challenge set is now defined by the (pseudo)random string $r_1 \oplus r_2$.*

Such a modified BBProve *allows us to replace the statistically-hiding commitment with a statistically-binding one in* both BBCom *and* VSSCom*. Thus, we can safely use the statically-binding* VSSCom *(as currently presented in Protocol 3.3.1).*

We also remark that BBCom *will not be statistically-binding, even though its underlying commitment is replaced by a statistically-binding one. This is because that* BBCom *applies the (underlying) commitment to (the paths of) a Merkle hashing tree on the target value, resulting in information loss.*

3.3.1 Robust Zero-Knowledge

To capture the above property (ii) (i.e., "straight-line simulation in the presence of bounded external communication"), we define the notion of *robust zero-knowledge*. It roughly captures the fact that the simulator does not rewind the external party to perform the simulation. This property is implicit in the relations defined for bounded-concurrent simulation in [Bar01, PR03]; a related but very different notion of robustness appears explicitly in the context of non-malleability in [LPV09, GLP$^+$15]. This notion is useful in constructing security proofs even though it follows from [Bar01] (and similar protocols).

Let $L \in \mathsf{NP}$ with witness relation R_L, and let $R_L(x) := \{w : R_L(x,w) = 1\}$. Let $\Pi := \langle P, V \rangle$ be an (efficient) interactive argument system for L and B be an arbitrary PPT ITM.

For $n \in \mathbb{N}, L \in \mathsf{NP}, x \in L, w \in R_L(x), z \in \{0,1\}^*$ and $y \in \{0,1\}^*$, we define the following two experiments:

Real Experiment: The experiment starts the execution of V^* on input $(1^n, x, z)$ where z denotes the auxiliary input of V^*. During its execution, V^* can simultaneously participate in two interactions (1) an execution of Π with the honest prover machine $P(1^n, x, w)$ and (2) arbitrary (unspecified) interaction with the machine $B(1^n, y)$.

The interaction occurs over a network where each message is processed as follows:

- If V^* sends a message of Π (resp., for B), it is delivered to P (resp., to B).
- If P receives a message from V^*, it prepares the next message of Π, denoted a; a is then sent to **both V^* as well as B**.
- If B receives a message from V^*, it prepares the next message (according to the unspecified interaction protocol between B and V^*), say b; message b is then sent to V^*.

The output of this experiment is the *(joint) view* of V^*, and denoted as:

$$\mathsf{Rview}^{B(y)}_{\Pi,n,x}\langle P(w), V^*(z)\rangle.$$

Simulated Experiment: This experiment is identical to the real experiment except that: (1) the honest prover algorithm $P(1^n, x, w)$ is *replaced* with a "simulator" algorithm S which receives the code of V^* as input, and (2) any message V^* receives from B is also provided to S.

Formally, the experiment starts an execution of $V^*(1^n, x, z)$; V^* can simultaneously participate in two interactions (1) an execution of Π with the *simulator machine* $S(1^n, x, \mathsf{code}[V^*], z)$ and (2) arbitrary (unspecified) interaction with the machine $B(1^n, y)$.

The interaction occurs over a network where each message is processed as follows:

- If V^* sends a message of Π (resp., for B), it is delivered to S (resp., to B).
- If S receives a message from V^*, it prepares the next message, denoted a; a is sent to **both V^* as well as B**.
- If B receives a message from V^*, it prepares the next message (according to the unspecified interaction protocol between B and V^*), say b; message b is then sent to **both V^* and S**.

The output of this experiment is the *(joint) view* of V^*, and denoted by:

$$\mathsf{Sview}^{B(y)}_{\Pi,n,x}\langle S(\mathsf{code}[V^*], z), V^*(z)\rangle.$$

Remark 3.3.2. *Two important remarks are in order. First, the simulated experiment* does not allow rewinding *by definition. Instead, it requires S to "act like the prover" of protocol*

Π*; the only help* S *has is the code of* V^* *as well as immediate access to all messages that* V^* *receives. In particular, rewinding* V^* *may involve rewinding* B *and this is not allowed by the experiment.*

Second, both B *and* S *have access to all messages* V^* receives *from the network.* S *must have access to all such messages to simulate in "straight line" (since it does not have the code of* B*).* B *is given access to these messages to facilitate (bounded concurrent)* composition. *In particular,* B *has access to all message* S *(or* P*) sends to* V^* *and* S *has access to all messages* B *sends to* V^**.*

Protocol Π is robust zero-knowledge if V^* cannot tell whether it is in the real experiment or the simulated one. If it is robust w.r.t. only machines B that send at most ℓ bits, it is called ℓ-robust zero-knowledge. Formally:

Definition 3.3.1 (Robust Zero-Knowledge)**.** *An interactive argument system* Π *for a language* $L \in \mathsf{NP}$ *is* robust ZK w.r.t. a PPT ITM B *if for all* PPT ITM V^* *there exists a* PPT ITM S *(called the* robust simulator*), such that:*

$$\left\{\mathsf{Rview}^{B(y)}_{\Pi,n,x}\langle P(w), V^*(z)\rangle\right\}_{n,x,w,z,y} \overset{\mathrm{c}}{\approx} \left\{\mathsf{Sview}^{B(y)}_{\Pi,n,x}\langle S(\mathsf{code}[V^*], z), V^*(z)\rangle\right\}_{n,x,z,y}.$$

where $n \in \mathbb{N}, x \in L, w \in R_L(x), z \in \{0,1\}^*, y \in \{0,1\}^*$*.*

For a polynomial $\ell : \mathbb{N} \to \mathbb{N}$, Π *is* ℓ-robust zero-knowledge *if it is robust w.r.t.* every PPT ITM B *that sends at most* $\ell(n)$ *bits.* Π *is* robust zero-knowledge *if it is* ℓ*-robust zero-knowledge for* every *polynomial* ℓ*.*

Remark 3.3.3. *We remark that robust (i.e., unbounded)* ZK *is actually impossible (for non-trivial languages) in the plain model since, if unbounded external communication was allowed with* B, V^* *can just be a "dummy" adversary so that access to its code provides no advantage to the simulator to complete the proof. This is akin to the use of dummy adversary in UC setting and impossibility of UC-*ZK *for languages outside of* BPP *[Can01, GK90].*

3.3.1.1 (Bounded) Robust ZK Implies Bounded $c\mathcal{ZK}$

We now demonstrate the flexibility of using robust ZK in concurrent settings. More specifically, we show that any ℓ-robust ZK protocol Π remains ZK under bounded composition of ℓ' instances for sufficiently large ℓ.

Recall that in the ℓ'-bounded $c\mathcal{ZK}$ composition of protocol Π, an adversarial verifier V^* participates in ℓ' simultaneous executions of Π while controlling the scheduling of messages of various sessions. For simplicity (only) we assume that all provers prove the same statement x using same witness w and let $\mathsf{view}_{\Pi,n,x,w,z}$ denote the view of $V^*(n, x, z)$ in this concurrent execution. We say that Π is ℓ'-bounded-$c\mathcal{ZK}$ for language L if for every such V^* there exists a simulator S_{V^*} such that for all $x \in L, w \in R_L(x), z \in \{0,1\}^*$:

$$\left\{\mathsf{view}_{\Pi,n,x,w,z}\right\}_{n,x,w,z} \overset{\mathrm{c}}{\approx} \left\{S_{V^*}(n, x, z)\right\}_{n,x,z}.$$

Claim 3.3.1. *If a protocol* Π *is* ℓ*-robust zero-knowledge, then it is* ℓ'*-bounded* $c\mathcal{ZK}$*, for any* ℓ' *such that* $\ell' \cdot m \leq \ell$ *where* m *is the length of all messages sent by the prover of protocol* Π*.*

Proof. We show that a simple composition of individual robust-ZK simulators for each session yields a simulator for bounded-concurrent composition of Π.

Let V^* be a concurrent verifier participating in ℓ' concurrent sessions of Π. Let S be the robust-ZK simulator for Π. The bounded-concurrent simulator S_{V^*}, on input the code of V^*, x, and z, proceeds as follows:

– For each session i, S_{V^*} prepares the "fake" prover algorithm S_i which behaves identically to the algorithm $S(n, x, \mathsf{code}[V^*], z)$ with fresh randomness and interacts with V^* in session i.

– S_{V^*} initiates an execution of V^* with fresh randomness, relaying messages between V^* and fake provers $(S_1, \dots, S_{\ell'})$ as in the bounded-concurrent execution.

– When V^* halts, S_{V^*} outputs its view.

It is straightforward to see that S_{V^*} runs in polynomial time since each S_i and V^* are polynomial time. To prove indistinguishability, consider hybrids $H_0, \dots, H_{\ell'}$:

– **Hybrid H_0.** The real experiment where V^* concurrently interacts with $(P_1, .., P_{\ell'})$, where P_i $(i \in [\ell'])$ denotes the i-th prover instance of Π on input $(1^n, x, w)$.

– **Hybrid H_k (for $(k \in [\ell'])$.** This hybrid is same as H_{k-1} except that prover instance P_k is replaced by the simulator instance of S_k (defined above). Therefore, V^* interacts with algorithms $(S_1, ..., S_k, P_{k+1}, ...P_{\ell'})$, as the "provers."

Note that $H_{\ell'}$ is the simulator S_{V^*}. It is easy to see that each H_k is polynomial time. We prove that $H_{k-1} \approx_c H_k$ using the robust-ZK property of Π, where $k \in [\ell']$.

Let B_k be the following machine: B_k incorporates $(S_1, ..., S_{k-1}, P_{k+1}, ..., P_{\ell'})$, and interacts with V^* in the robust-ZK experiment as follows: B_k proceeds identically to H_{k-1} so that messages of all sessions $i \neq k$, are received from or sent to B_k (which internally simulates H_{k-1}). All prover messages of the k-th session are expected to come from an external machine, say M. If M is the prover instance P_k, the view of V^* is distributed identically to H_{k-1}. Note that a copy of each message of P_k in this case is also sent to B_k at the same time as V^*; consequently, the internal execution of B_k (which includes $S_1, \dots, S_{k-1}$) continues without any problems. Likewise if M is the simulator instance S_k, V^*'s view is distributed identically to H_k; note that a copy of each message of S_k (resp., B_k) in this case is also sent to B_k (resp., S_k) at the same time as V^*. Consequently executions of both S_k and B_k continues without any problems.

Finally, if m is the total communication from a single prover instance, the external communication to V^* from B_k is bounded by $m\ell' \leq \ell$; furthermore, this condition also holds from the point of view of each S_i instance internal to B_k (as desired). It follows that if Π is ℓ-robust-ZK, it is also ℓ'-bounded concurrent. □

3.3.2 Constructions of ℓ-Robust ZK

As noted earlier, Barak's bounded $c\mathcal{ZK}$ protocol is also ℓ-robust ZK, although it requires non-black-box use of hash functions [Bar01]. The variant of Barak's technique by Goyal et al. [GOSV14] makes only black-box use of such functions and achieves the same result.[7] To summarize, we have the following theorem from [GOSV14] (restated in our language).

Theorem 3.3.1 (Black-Box ℓ-Robust Zero-Knowledge for NP)**.** *If there exists a family $\mathcal{H}$ of collision-resistant hash functions, then for every polynomial ℓ, there exists a constant round public coin ℓ-robust zero-knowledge interactive argument for NP which requires only oracle access to functions in $\mathcal{H}$.*

As noted earlier, the preceding theorem is actually a corollary of the more general theorem that proof-phase of the commit-and-prove protocol depicted in Protocol 3.3.1 is ℓ-robust. We refer the reader to [GOSV14] for a formal definition of "commit-and-prove" protocols. We only recall the following properties for Protocol 3.3.1:

– The proof-phase is performed only for the statement defined by the transcript of the commit-phase.

– For each transcript, the receiver gets *only one* (interactive) proof from the committer during the proof-phase. The zero-knowledge property (as well as the implicit ℓ-robust zero-knowledge) is then required only for this single execution of the proof-phase. This suffices for Theorem 3.3.1 (by simply repeating the commit-phase before every proof-phase). See also Footnote 6.

– To get the ℓ-robust ZK property, the length of the challenge from the verifier is modified to be sufficiently larger than ℓ (as in bounded $c\mathcal{ZK}$ in Barak [Bar01]). Note that this requires modifying the pair language for the universal argument (and PCPP) to allow strings of length at most ℓ. In particular, this language is the following one:

  * $\mathcal{L}_{\mathcal{P}} = \big\{(a = (z, r, t), (Y)) : \exists M \in \{0,1\}^*$ and $\exists y \in \{0,1\}^*$ such that $Y \leftarrow \mathsf{ECC}(M)$, $M(z, y) = r$ within t steps, and $|y| \leq |r| - n.\big\}$,

  where $\mathsf{ECC}(\cdot)$ is a binary error correcting code tolerating a constant fraction $\delta > 0$ of errors, M is the description of a Turing machine and n is the security parameter. We use $R_{\mathcal{L}_{\mathcal{P}}}$ to denote the relation defined on $\mathcal{L}_{\mathcal{P}}$.

To summarize, we have the following theorem (from [GOSV14]):

Theorem 3.3.2 (Black-Box ℓ-Robust Commit-and-Prove)**.** *If there exists a family $\mathcal{H}$ of collision-resistant hash functions, then for every polynomial ℓ and every polynomial-size circuit ϕ, there exists a commit-and-prove protocol such that the commit-phase is statistically binding (with at most two rounds), the proof-phase is a constant-round public-coin ℓ-robust zero-knowledge interactive argument for ϕ, and both phases require only oracle access to functions in $\mathcal{H}$.*

Remark 3.3.4. *We note that [GOSV14] also provides a size-hiding commitment scheme (which cannot be statistically-binding) along with a ℓ-robust* ZK *proof-phase for every ϕ. However, we will not need this version of their protocol.*

[7] Although only standalone case is discussed in [GOSV14], their security proof (just like Barak's) also works for bounded-concurrent case by increasing the length of verifier's challenge and slightly modifying the relation for the UARG appropriately.

3.4 Straight-Line Extractable Commitments

In this section, we construct an extractable commitment scheme, assuming black-box access to any semi-honest oblivious transfer. The construction (shown in Protocol 3.4.1) makes black-box use of a statistically-binding commitment Com and a maliciously-secure oblivious transfer OT. For the OT, we require (computational) indistinguishability-based security against malicious senders, and simulation-based security (ideal/real paradigm) against malicious receivers. Such OTs can be constructed in a black-box manner from any semi-honest OT [Hai08]. To ease the presentation, we show in Protocol 3.4.1 a single-bit commitment, and talk about how to extend it to commit to strings toward the end of this section (Remark 3.4.1).

Theorem 3.4.1. *Protocol 3.4.1 is a straight-line extractable statistically-binding commitment scheme, which only accesses the underlying primitives in a black-box manner.*

Protocol 3.4.1: ℓ-Robust Extractable Statistically-Binding Commitment

Common Input: Security parameter 1^λ, robustness parameter ℓ.

Auxiliary Input to C: A bit $\sigma \in \{0,1\}$ to be committed.

Commit Phase:

1. C samples 2λ random bits $\{s_{i,b}\}_{i\in[\lambda],b\in\{0,1\}}$, whose exclusive-or equals σ.
2. C and R involves in $2n$ independent executions of Com in parallel, where C commits to each values in $\{s_{i,b}\}_{i\in[\lambda],b\in\{0,1\}}$ separately. Let $c_{i,b}$ denote the commitment to $s_{i,b}$. Let $d_{i,b}$ denote the decommitment information w.r.t. $c_{i,b}$.
3. R samples independently $\lambda-1$ random bits $\tau_1,\ldots,\tau_{\lambda-1} \xleftarrow{\$} \{0,1\}^{\lambda-1}$. C and R involves in n independent executions of OT in parallel. For the i-th OT execution ($i \in [\lambda-1]$), C acts as the sender with the two private input set to $\mathsf{Inp}^{(i)}_0 = d_{i,0}$ and $\mathsf{Inp}^{(i)}_1 = d_{i,1}$. R acts as the receiver with input τ_i. Note that at the end of this stage R learns $\{d_{i,\tau_i}\}_{i\in[\lambda-1]}$. R rejects if any of these decommitments are invalid.
4. R samples uniformly at random a bit $\tau_\lambda \xleftarrow{\$} \{0,1\}$. C and R involves in an execution of OT where C acts as the sender with the two private input set to $\mathsf{Inp}^{(\lambda)}_0 = d_{n,0}$ and $\mathsf{Inp}^{(\lambda)}_1 = d_{\lambda,1}$. R acts as the receiver with input τ_λ. Note that at the end of this stage R learns d_{λ,τ_λ}. R rejects if d_{λ,τ_λ} is not a valid decommitment w.r.t. c_{λ,τ_λ}.
5. C and R run a coin-tossing protocol:
   (a) R samples a random string $r_1 \xleftarrow{\$} \{0,1\}^\lambda$ and runs the VSS **Commit Phase** of Protocol 3.3.1 to generate $c_{r_1} = \mathsf{VSSCom}(r_1)$. R sends c_{r_1}.
   (b) C chooses a random string $r_2 \xleftarrow{\$} \{0,1\}^\lambda$ and sends r_2.
   (c) R sends r_1 (without decommitment information)
   (d) R and C run the **Proof Phase** of Protocol 3.3.1 with robustness parameter $\ell(\lambda)$ to prove that the string r_1 sent by R in Step 5-(c) is indeed the value it committed to

in Step 5-(a).

The output of the coin-tossing phase is $\mathsf{ch} = r_1 \oplus r_2$. For $i \in [\lambda]$, let ch_i denote the i-th bit of ch.

6. C sends to R the values $\{d_{i,\mathsf{ch}_i}\}_{i\in[\lambda]}$. Note that these are the decommitments to $\{c_{i,\mathsf{ch}_i}\}_{i\in[\lambda]}$ in Step 2. R rejects if any of these decommitments are invalid.

Reveal Phase:

1. C sends to R the values $\{d_{i,b}\}_{i\in[\lambda],b\in\{0,1\}}$ (aka all the decommitments).
2. R rejects if any of the decommitments is invalid; otherwise, R computes the decommitted value as $\sigma = \oplus_{i,b} s_{i,b}$. (Note that $s_{i,b}$ is contained in $d_{i,b}$.)

3.4.1 Proof of Theorem 3.4.1

The construction is black-box as we use the black-box commit-and-prove protocol from [GOSV14] (presented in Protocol 3.3.1 in Section 3.3) in the coin-tossing step. Statistically-binding property follows directly from that of the Step-2 commitment scheme Com. Next, we focus on computationally-hiding property and extractability.

3.4.1.1 Computationally-Hiding

Let σ be an arbitrary bit in $\{0,1\}$. For any PPT receiver R^*, we denote by $\mathcal{V}^{R^*}(\lambda,\sigma)$ the distribution over R^*'s view from an execution $\langle C(\sigma), R^*\rangle$ of Protocol 3.4.1, where the honest C commits to the value σ to R^*. To prove the hiding property, we need to show that for any PPT machine $\mathcal{D}$,

$$\mathsf{Adv}^{\mathcal{D}}_{\lambda} := \left|\Pr[\mathcal{D}\big(\mathcal{V}^{R^*}(\lambda,1)\big) = 1] - \Pr[\mathcal{D}\big(\mathcal{V}^{R^*}(\lambda,0)\big) = 1]\right| \le \mathsf{negl}(\lambda). \tag{3.1}$$

In the following, we prove Inequality (3.1) by a sequence of hybrids.

Hybrid $H_0(\lambda,\sigma)$: in this hybrid, we change the way the values $\{s_{i,b}\}$ are chosen. Specifically, the hybrid does the following:

(a) It samples independently at random a bit $\eta \xleftarrow{\$} \{0,1\}$ and a bit $\mathsf{g} \xleftarrow{\$} \{0,1\}$.

(b) For $i \in [\lambda-1]$ and $b \in \{0,1\}$, it samples independently $s_{i,b} \xleftarrow{\$} \{0,1\}$.

(c) It defines $s_{\lambda,1-\mathsf{g}} := \eta \oplus \sigma$ and

$$s_{\lambda,\mathsf{g}} := (s_{1,0} \oplus s_{1,1}) \oplus \ldots \oplus (s_{\lambda-1,0} \oplus s_{\lambda-1,1}) \oplus \eta$$

(d) It then uses the honest commiter's strategy and $\{s_{i,b}\}$ defined above to finish Step 2 to 6 in Protocol 3.4.1.

(e) Once R^* terminates, H_0 gets the view $\mathcal{V}_0^{R^*}(\lambda,\sigma)$ of R^* in this execution. It invokes $\mathcal{D}$ on input $\mathcal{V}_0^{R^*}(\lambda,\sigma)$, and outputs whatever $\mathcal{D}$ outputs. Let $H_0(\lambda,\sigma)$ also denote the output of this hybrid.

It is straightforward to see the values $\{s_{i,b}\}$ defined above are identically distributed as in the real execution $\langle C(\sigma), R^*\rangle$, i.e. they constitute random secret shares whose exclusive-or equals σ. Also, we note that the value of g does not affect this hybrid at all, as it only introduces syntax changes. Therefore, we have $\mathcal{V}_0^{R^*}(\lambda,\sigma) \overset{\text{i.d.}}{=\!=} \mathcal{V}^{R^*}(\lambda,\sigma)$, which implies that $\forall \sigma \in \{0,1\}$:

$$\Pr[\mathcal{D}\big(\mathcal{V}^{R^*}(\lambda,\sigma)\big) = 1] = \Pr[\mathcal{D}\big(\mathcal{V}_0^{R^*}(\lambda,\sigma)\big) = 1] = \Pr[H_0(\lambda,\sigma) = 1] \tag{3.2}$$

Hybrid $H_1(\lambda,\sigma)$**:** this hybrid is identical to $H_0(\lambda,\sigma)$ except that the λ-th OT (in Step 4) are replaced with the ideal OT functionality $\mathcal{F}_{\text{OT}}$. Concretely, in Step 4, the hybrid emulates $\mathcal{F}_{\text{OT}}$ internally in the following way:

- On the committer side, it sets $\mathsf{Inp}_0^{(\lambda)} = d_{n,0}$ and $\mathsf{Inp}_1^{(\lambda)} = d_{n,1}$ (same as the honest committer).
- On the receiver side, it invokes the PPT ideal-world simulator $S^{\hat{R}^*}$ with oracle access to $\hat{R}^*$, which is the residual strategy of R^* with the view fixed up to the beginning of Step 4. Note that the existence of S is guaranteed by the security of OT against corrupted receiver.
- During the execution, $S^{\hat{R}^*}$ may send a bit b which is meant to the ideal-world receiver's input to $\mathcal{F}_{\text{OT}}$ (the actually input of R^* "extracted" by S). In this case, the hybrid responds with $\mathsf{Inp}_b^{(\lambda)} = d_{n,b}$.
- Once $S^{\hat{R}^*}$ stops and outputs the simulated view for $\hat{R}^*$, the hybrid continues to finish the execution of Step 5-6 in the same way as in $H_0(\lambda,\sigma)$. (Note that simulated view for $\hat{R}^*$ contains necessary information to recover the status of R^* up to the end of Step 4. The hybrid can then use it to finish the remaining steps.)

Similar as in H_0, we use $\mathcal{V}_1^{R^*}(\lambda,\sigma)$ to denote the view of R^* in this execution, and use $H_1(\lambda,\sigma)$ to denote the output of this hybrid. By the security of OT against malicious R^*, the $\mathcal{V}_1^{R^*}(\lambda,\sigma)$ should be computationally indistinguishable from $\mathcal{V}_0^{R^*}(\lambda,\sigma)$. This implies that $\forall \sigma \in \{0,1\}$:

$$\Pr[H_1(\lambda,\sigma) = 1] = \Pr[H_0(\lambda,\sigma) = 1] \pm \mathsf{negl}(\lambda) \tag{3.3}$$

Hybrid $H_2(\lambda,\sigma)$**:** $H_2(\lambda,\sigma)$ is identical to $H_1(\lambda,\sigma)$ except that it aborts and outputs a special symbol $\perp$ if $b = 1 - \mathsf{g}$. Recall that b is the query of $S^{\hat{R}^*}$ in Step 4 of H_1 (and also H_2). Recall that the bit g is picked uniformly at random, independent of the view of R^*. Therefore, H_2 aborts with probability exactly $1/2$. This implies $\forall \sigma \in \{0,1\}$,

$$\Pr[H_2(\lambda,\sigma)\big) = 1] = \frac{1}{2} \cdot \Pr[H_1(\lambda,\sigma)\big) = 1] \tag{3.4}$$

Hybrid $H_3(\lambda,\sigma)$**:** $H_3(\lambda,\sigma)$ is identical to $H_2(\lambda,\sigma)$ except that it aborts and outputs a special symbol $\perp$ if $\mathsf{ch}_\lambda = 1 - \mathsf{g}$ (note that the change in H_2 already ensures that $\mathsf{g} = b$, given that this hybrid does not abort). Recall that ch_λ is the last bit of the result of the

Step-5 coin-tossing in H_2 (and also H_3). Since the output of Step-5 coin-tossing should be pseudo-random, the probability $\Pr[\mathsf{ch}_\lambda = 1 - \mathsf{g}]$ is negligibly close to 1/2. Thus, we have $\forall \sigma \in \{0, 1\}$,

$$\Pr[H_3(\lambda, \sigma)\big) = 1] = \frac{1}{2} \cdot \Pr[H_2(\lambda, \sigma)\big) = 1] \pm \mathsf{negl}(\lambda) \tag{3.5}$$

where the $\mathsf{negl}(\lambda)$ term is due to the negligible possibility that R^* breaks the security of Step-5 coin-tossing.

We now finish the proof for computationally-hiding property by showing the following claim.

Claim 3.4.1.

$$\big|\Pr[H_3(\lambda, 1)\big) = 1] - \Pr[H_3(\lambda, 0) = 1]\big| \leq \mathsf{negl}(\lambda) \tag{3.6}$$

Before presenting the proof for Claim 3.4.1, let us show why it closes the proof for computationally-hiding property of Protocol 3.4.1. First, note that Equations (3.2) to (3.5) imply that:

$$\begin{aligned}
& \big|\Pr[\mathcal{D}\big(\mathcal{V}^{R^*}(\lambda, 1)\big) = 1] - \Pr[\mathcal{D}\big(\mathcal{V}^{R^*}(\lambda, 0)\big) = 1]\big| \\
= \ & \big|\Pr[H_0(\lambda, 1) = 1] - \Pr[H_0(\lambda, 0) = 1]\big| \\
= \ & \big|\Pr[H_1(\lambda, 1) = 1] - \Pr[H_1(\lambda, 0) = 1] \pm \mathsf{negl}(\lambda)\big| \\
= \ & 2 \cdot \big|\Pr[H_2(\lambda, 1) = 1] - \Pr[H_2(\lambda, 0) = 1] \pm \mathsf{negl}(\lambda)\big| \\
= \ & 4 \cdot \big|\Pr[H_3(\lambda, 1) = 1] - \Pr[H_3(\lambda, 0) = 1] \pm \mathsf{negl}(\lambda)\big| \\
\leq \ & 4 \cdot \big|\Pr[H_3(\lambda, 1) = 1] - \Pr[H_3(\lambda, 0) = 1]\big| \pm \mathsf{negl}(\lambda)
\end{aligned} \tag{3.7}$$

Combining Inequalities (3.6) and (3.7) proves Inequality (3.1), which finishes the proof for hiding property of Protocol 3.4.1.

In the following, we finish this part by presenting the proof for Claim 3.4.1.

Proof for Claim 3.4.1. First note that $\forall \sigma \in \{0, 1\}$, $H_3(\lambda, \sigma)$ does not need to know $d_{\lambda, 1-\mathsf{g}}$, the decommitment information for $c_{\lambda, 1-\mathsf{g}} = \mathsf{Com}(\eta \oplus \sigma)$. That is because once the query b of $S^{\hat{R}^*}$ or the value ch_λ equals $1 - \mathsf{g}$, the hybrid H_3 will simply abort. Therefore, if Inequality (3.6) does not hold, we can build a PPT machine $\mathcal{D}_{\mathsf{com}}$ that breaks the computationally-hiding property of Com. The distinguisher $\mathcal{D}_{\mathsf{com}}$ runs H_3 but define $c_{\lambda, 1-\mathsf{g}}$ in the following way:

- It forwards two values $m_0 \coloneqq \eta \oplus 0$ and $m_1 \coloneqq \eta \oplus 1$ to the outsider challenger for the hiding game of Com.
- Once it receives the commitment c^* from the challenger, it sets $c_{\lambda, 1-\mathsf{g}} = \mathsf{c}^*$.

Upon the halt of H_3, $\mathcal{D}_{\mathsf{com}}$ outputs whatever H_3 outputs. From the above description, it is easy to see that if the outside challenger commits to m_0, the view of R^* is identical to that in $H_3(\lambda, 0)$; if the outside challenger commits to m_0, the view of R^* is identical to that in $H_3(\lambda, 1)$. Therefore, if Inequality (3.6) does not hold, $\mathcal{D}_{\mathsf{com}}$ breaks the computationally-hiding property of Com. This finishes the proof for Claim 3.4.1. □

3.4.1.2 Straight-Line Extractability

At a high level, the extractor $\mathcal{E}$ works by biasing the outcome $\mathsf{ch} = r_1 \oplus r_2$ of the Step-5 coin-tossing, such that $\mathsf{ch}_i \oplus \tau_i = 1$ for all $i \in [n]$. In this case, $\mathcal{E}$ learns the decommitments to all the values $\{s_{i,b}\}_{i\in[\lambda],b\in\{0,1\}}$ at the end of **Commit Phase**, thus being able to extract σ.

Extractor $\mathcal{E}$ works as follows:

1. $\mathcal{E}$ invokes C^* and interacts with it using the honest receiver strategy up to the beginning of Step 5.
2. In Step 5, $\mathcal{E}$ acts as follows:
   (a) $\mathcal{E}$ runs the VSS **Commit Phase** of Protocol 3.3.1 to generate $c_{r_1} = \mathsf{VSSCom}(0^\lambda)$. R sends c_{r_1}.
   (b) $\mathcal{E}$ receives from C^* the value r_2.
   (c) $\mathcal{E}$ sends to C^* the value $r_1 \coloneqq r_2 \oplus (\overline{\tau}_1 \| \ldots \| \overline{\tau}_\lambda)$, where $\overline{\tau}_i = 1 \oplus \tau_i$ for $i \in [\lambda]$.
   (d) $\mathcal{E}$ invokes the (straight-line) simulator of Protocol 3.3.1, with the residual C^* as the verifier, for the (false) statement that c_{r_1} is a VSSCom commitment to the value r_1.

   Note that the output of this (biased) coin-tossing is $\mathsf{ch} = r_1 \oplus r_2$, which equals $\overline{\tau}_1 \| \ldots \| \overline{\tau}_\lambda$.
3. $\mathcal{E}$ receives from C^* the values $\{d_{i,\mathsf{ch}_i}\}$. It aborts if any of these decommitments are invalid; otherwise, it outputs $\sigma = \oplus_{i,b} s_{i,b}$. Note that if it does not abort, $\mathcal{E}$ learns all the $\{s_{i,b}\}$ values. Because it learns $\{s_{i,\tau_i}\}_{i\in[\lambda]}$ from the OT executions in Step 3 and 4; it also learns $\{s_{i,\overline{\tau}_i}\}_{i\in[\lambda]}$ from the Step-6 decommitments.
4. **Output:** $\mathcal{E}$ outputs C^*'s view of the above execution along with σ.

From the above description, it is clear that $\mathcal{E}$ runs in expected polynomial-time, because the simulator of Protocol 3.3.1 runs in expected polynomial time and all the remaining steps run in polynomial time. Also, if C^* does not abort, $\mathcal{E}$ will be able to extract the value σ. In the following, we show that C^*'s behavior (actually its view) will not change (up to negligible probability) between the real execution and its interaction with $\mathcal{E}$.

We now show that the view output by $\mathcal{E}$ is computationally indistinguishable from C^*'s view in a real execution through the following sequence of hybrids:

- **Hybrid H_0:** This hybrid runs the real execution $\langle C^*, R\rangle$ between the (malicious) committer C^* and the honest receiver R. At the end of the execution, H_1 outputs the view of C^*.
- **Hybrid H_1:** this hybrid is identical to the H_0 except that the zero-knowledge argument verified by C^* in Step 5-(d) is replaced by a simulated one using the simulator of Protocol 3.3.1.
- **Hybrid H_2:** this hybrid is identical to H_1 except that the commitment received by C^* in Step 5-(a) is to 0^λ rather than to a random string r_1;
- **Hybrid H_3:** this hybrid is identical to H_2 except that the value r_1 in Step 5-(c) is set to $r_1 \coloneqq r_2 \oplus (\overline{\tau}_1 \| \ldots \| \overline{\tau}_\lambda)$, where $\overline{\tau}_i = 1 \oplus \tau_i$. (Note that the output of this hybrid is identical to the view of C^* output by $\mathcal{E}$).

The computational indistinguishability between (the output of) H_0 and H_1 can be established by the ZK property of the proof stage of Protocol 3.3.1. The computational indistinguishability between H_1 and H_2 can be established by the hiding property of the committing stage of Protocol 3.3.1 (simply by forwarding the commitment on which these two hybrids differ to an outside challenger for the hiding property of VSSCom).

The computational indistinguishability between H_2 and H_3 can be established by standard hybrid arguments. More specifically, we consider the following intermediate hybrids:

– **Hybrids H^i_3 $(i \in [\lambda])$:** this hybrid is identical to H_3 except that the value r_1 in Step 5-(c) is set to $r_1 \coloneqq r_2 \oplus (\overline{\tau}_1 \| \ldots \| \overline{\tau}_i \| u_{i+1} \| \ldots \| u_\lambda)$, where $\overline{\tau}_j = 1 \oplus \tau_j$ $(j \in [i])$ and u_k $(k \in \{i+1, \ldots, \lambda\})$ is a random bit sampled independently.

Note that H^λ_3 is identical to H_3. We additional define $H^0_3 \coloneqq H_2$. Then the computational indistinguishability between H^{i-1}_3 and H^i_3 $(\forall i \in [n])$ follows from the security of OT against malicious senders. Concretely, the i-th OT execution is forwarded between C^* and an external OT challenger. If the view of C^* between H^{i-1}_3 and H^i_3 changes in a non-negligible way, the hybrid constitutes a PPT machine that tells the secret input of the challenger non-negligibly better than random guess. Thus, we have $H_0 \overset{\text{c}}{\approx} H_1 \overset{\text{c}}{\approx} H_2 \overset{\text{c}}{\approx} H_3$, which finishes the proof of extractability.

This finishes the proof of Theorem 3.4.1.

Remark 3.4.1 (Committing to Strings)**.** *One obvious approach to extend Protocol 3.4.1 to support multi-bit strings is to commit to each bit independently in parallel. A more efficient way is to replace the single-bit commitments in Step 2 and OTs in Step 3 and 4 with their multi-bit version. It is straightforward to see that same proof for correctness and security holds for this the multi-bit version.*

3.5 Our Bounded-Concurrent OT Protocol

In this section, we prove the following theorem.

Theorem 3.5.1. *Assume the existence of constant-round semi-honest oblivious transfer protocols and collision-resistant hash functions. Let $\mathcal{F}_{\mathrm{OT}}$ be the ideal oblivious transfer functionality (Figure 3.5.1). Then, for every polynomial m, there exists a constant-round protocol that securely computes $\mathcal{F}_{\mathrm{OT}}$ under m-bounded concurrent composition, and it uses the underlying primitives in the black-box way.*

Figure 3.5.1: The Oblivious Transfer Functionality $\mathcal{F}_{\mathrm{OT}}$.

The ideal OT functionality $\mathcal{F}_{\mathrm{OT}}$ interacts with a sender S and a receiver R.

– Upon receiving a message $(\mathsf{sid}, \mathsf{sender}, v_0, v_1)$ from S, where each $v_i \in \{0,1\}^n$, store (v_0, v_1).

– Upon receiving a message $(\mathsf{sid}, \mathsf{receiver}, u)$ from R, where $u \in \{0,1\}$, check if a $(\mathsf{sid}, \mathsf{sender}, \ldots)$ message was previously sent. If yes, send (sid, v_u) to R and (sid) to the adversary Sim and halt. If not, send nothing to R.

At a high-level, we obtain the OT claimed in Theorem 3.5.1 by replacing the statistically-binding commitment Com in the OT of [GKP18] (denoted as GKP-OT) with a new commitment based on Protocol 3.4.1. In the following, we will first describe the intuition behind our construction in Section 3.5.1, and then present our protocol in Section 3.5.2.

3.5.1 The High-Level Idea

As mentioned in the technical overview (Section 3.1.1), we want to employ the same simulation technique for GKP-OT, but with an (efficient) alternative way in which the simulator can "extract" the value committed in Com. Let us first recall the (only) two places where Com is used in GKP-OT:

1. In the very beginning (Stage 1), S (resp. R) uses Com to commit to a random set Γ_S (resp. Γ_R), which is used later for cut-and-chose.
2. Next, S (resp. R) uses Com to commit to a random string $\boldsymbol{a}^S$ (resp. $\boldsymbol{a}^R$) in the (Stage-2) coin tossing, which will later be used as inputs to the parallel execution of several random OT instances (which are in turned used for an OT-combiner stage later).

Since we now have the straight-line extractable commitment (Protocol 3.4.1) at our disposal, we may replace Com with Protocol 3.4.1. We notice that the GKP simulator $\mathsf{Sim}_{\mathrm{OT}}$ can be extended to our setting by substituting the brute-forcing with the extractor for Protocol 3.4.1 extractor. However, this method requires us to insert many intermediate hybrids in carefully-chosen places as we need to ensure that the extractions happen in time, while not disturbing the adjacent hybrids. We thus take the following alternative approach.

Our Approach. We add a new step (called **Stage 0**) in the beginning of GKP-OT, where S (resp. R) commits using Protocol 3.4.1 to two random strings ϕ^S and ψ^S (resp. ϕ^R and ψ^R) of proper length. We then continue identically as in GKP-OT with the following modifications:

- In Stage 1, when S (resp. R) needs to commit to Γ_S (resp. Γ_R), he simply sends $\phi^S \oplus \Gamma_S$ (resp. $\phi^R \oplus \Gamma_R$);
- In Stage 2, when S (resp. R) needs to commit to $\boldsymbol{a}^S$ (resp. $\boldsymbol{a}^R$), he simply sends $\psi^S \oplus \boldsymbol{a}^S$ (resp. $\psi^R \oplus \boldsymbol{a}^R$).

Intuitively, we ask both parties commit to random strings which will later be used as one-time pads to "mask" the values they committed to by Com in the original GKP-OT. The hiding of Γ_S and $\boldsymbol{a}^S$ follows straightforwardly. To allow the simulator to extract them efficiently, it is sufficient to let $\mathsf{Sim}_{\mathrm{OT}}$ use the extractor of Protocol 3.4.1 to extract ϕ^S and ψ^S. This can be done based on two important properties of the extractor for Protocol 3.4.1:

1. **Straight-line Extraction:** this guarantees that $\mathsf{Sim}_{\mathrm{OT}}$ can finish the extraction efficiently, free of the exponential-time problem due to recursive rewinding (similar as that for concurrent zero-knowledges [DNS98]).
2. **Robustness:** since Protocol 3.4.1 is based on the ℓ-robust ZK (Section 3.3), its extractor inherits the ℓ-robustness. By setting the parameter ℓ carefully, we make sure that the

simulator can switch from honest receiver's strategy to the extractor's strategy session by session, even in the presence of (bounded-ly) many other sessions.

Since we put the commitments to those masks in the very beginning, all the extractions can be done before further hybrids are defined. Similar arguments also apply when the receiver is corrupted. Therefore, we can make use of the GKP technique in a modular way to finish the proof of Theorem 3.5.1.

3.5.2 Protocol Description

Our protocol makes use of the following building blocks:

– The robust-extractable commitment scheme defined in Protocol 3.4.1, to which we refer as RobCom. Note that the commitment protocol is based only on CRHFs and semi-honest OTs in a black-box manner.

– A four-round statistically-binding extractable commitment ExtCom, which can be constructed from one-way functions in the black-box way [Nao91, HILL99b, PW09].

– A $O(1)$-round OT protocol $\mathsf{mS\text{-}OT}$ that is secure against malicious senders and semi-honest receivers.[8] As shown in [HIK+11], such a OT protocol can be obtained from any semi-honest one in the black-box way.

– A $O(1)$-round parallel non-malleable commitment NMCom that is parallel k-robust for sufficiently large constant k. (Concretely, we require that k is larger than the round complexity of the above three building blocks.) Such a non-malleable commitment scheme can be constructed from CRHFs in the black-box way [GKP18].

Our OT protocol Π_{OT} is described below. As explained in Section 3.1.1, (1) our protocol is based on the OT protocol of [GKP18], which roughly consists of coin-tossing, semi-honest OT, OT combiner, and cut-and-choose, and relies on non-malleable commitments to make sure adversary cannot setup the "trapdoor statement" to be true even in the bounded-concurrent setting; and (2) our protocol additionally uses a black-box "commit-and-prove" protocol that is ℓ-robust-ZK for a suitably large ℓ to commit a string and later prove in zero-knowledge that the opened value is indeed what was committed. Below, we give intuitive explanations in italic.

Parameters: The security parameter is n, and the bounded-concurrent composition parameter is $m := m(n)$.

Inputs: The input to the sender S is $v_0, v_1 \in \{0,1\}^n$.The input to the receiver R is $u \in \{0,1\}$. The identities of S and R are $\mathsf{id}_S, \mathsf{id}_R$ respectively.

Stage 0: (Extractable Commitments to Randomness)

1. **Commitments to S's randomness.**

[8]We only requires $\mathsf{mS\text{-}OT}$ to be secure under a game-based definition (which is preserved under parallel composition). For details, see the the proofs of Lemma 3.7.6 and Claim 3.7.6.

(a) S samples independently two random strings ϕ^S and $\psi^S = \psi^S_1 \| \ldots \| \psi^S_{11n}$ of proper length (see the comment at the end of this stage).

(b) S and R involve in $11n + 1$ executions of RobCom in parallel, where S commits to ϕ^S and $\psi^S_1, \ldots, \psi^S_{11n}$ respectively.

Note that for the ZK argument in Step 5d of Protocol 3.4.1, we set the robustness parameter to be $\ell(n) = m \cdot \nu_{\text{OT}}(n)$ where ν_{OT} is defined towards the end. This **Proof Phase** includes the **long message** of Protocol 3.3.1. We call this message **receiver's long message**. (Note that although the sender commits in this step, the long message actually flows from the receiver to the sender. Thus, it is called the receiver's long message.)

2. **Commitments to R's randomness.**

(a) R samples independently two random strings ϕ^R and $\psi^R = \psi^R_1 \| \ldots \| \psi^R_{11n}$ of proper length (see the comment at the end of this stage).

(b) S and R involve in $11n + 1$ executions of RobCom in parallel, where R commits to ϕ^R and $\psi^R_1, \ldots, \|\psi^R_{11n}$ respectively.

Note that for the ZK argument in Step 5d of Protocol 3.4.1, we set the robustness parameter to be $\ell(n) = m \cdot \nu_{\text{OT}}(n)$ where ν_{OT} is defined towards the end. This **Proof Phase** includes the **long message** of Protocol 3.3.1. We call this message **sender's long message**.

COMMENT: *In step 1 of this Stage, ϕ^S will be used in Stage 1-1 as a One-Time Pad to "mask" the sender's secrete Γ^S (which in turn is used as the sender's challenge for cut-and-choose). Similarly, ψ^S will be used in Stage 2-1 to mask the sender's secrete $\boldsymbol{a}^S$.*
Step 2 is just the symmetric execution of the same protocol where S and R exchange their role.

Stage 1: (Preprocess for cut-and-choose)

1. S samples a random subset $\Gamma_S := \{\gamma^S_1, ..., \gamma^S_n\} \subset [11n]$ of size n.[9] It then sends to R the value $\Gamma_S \oplus \phi^S$, i.e. the bit representation of Γ_S masked by the string ϕ^S using exclusive-or.
2. R samples a random subset $\Gamma_R := \{\gamma^R_1, ..., \gamma^R_n\} \subset [11n]$ of size n. It then sends to S the value $\Gamma_R \oplus \phi^R$.

COMMENT: *As in the OT protocols of [LP12, GKP18], the subsets to for the cut-and-choose stages are committed in advance to prevent selective opening attacks.*

Stage 2: (Coin-tossing for sub-protocols)

1. **(Coin tossing for S)** S samples random strings $\boldsymbol{a}^S = (a^S_1, \ldots, a^S_{11n})$. It then sends to R the values $z^S_i := a^S_i \oplus \psi^S_i$ for each $i \in [11n]$. Let d^S_i be the decommitments w.r.t. the Stage-0-1 RobCom of ϕ^S_i. R then sends random strings $\boldsymbol{b}^S = (b^S_1, \ldots, b^S_{11n})$

[9]Note that Γ_S can be represented using a bit-string of length $11n$.

to S. S then defines $\boldsymbol{r}^S = (r_1^S, \ldots, r_{11n}^S)$ by $r_i^S \overset{\text{def}}{=} a_i^S \oplus b_i^S$ for each $i \in [11n]$ and parses r_i^S as $s_{i,0} \| s_{i,1} \| \tau_i^S$ for each $i \in [11n]$.

2. **(Coin tossing for R)** R samples random strings $\boldsymbol{a}^R = (a_1^R, \ldots, a_{11n}^R)$. It then sends to S the values $z_i^R \coloneqq a_i^R \oplus \psi_i^R$ for each $i \in [11n]$. Let d_i^R be the decommitments w.r.t. the Stage-0-2 RobCom of ϕ_i^R. S then sends random strings $\boldsymbol{b}^R = (b_1^R, \ldots, b_{11n}^R)$ to R. R then defines $\boldsymbol{r}^R = (r_1^R, \ldots, r_{11n}^R)$ by $r_i^R \overset{\text{def}}{=} a_i^R \oplus b_i^R$ for each $i \in [11n]$ and parses r_i^R as $c_i \| \tau_i^R$ for each $i \in [11n]$.

Stage 3: (mS-OTs with random inputs)

S and R execute $11n$ instances of mS-OT in parallel. In the i-th instance, S uses $(s_{i,0}, s_{i,1})$ as the input and τ_i^S as the randomness, and R uses c_i as the input and τ_i^R as the randomness, where $\{s_{i,0}, s_{i,1}, \tau_i^S\}_i$ and $\{c_i, \tau_i^R\}_i$ are the random coins that were obtained in Stage 2. The output to R is denoted by $\widetilde{s}_1, \ldots, \widetilde{s}_{11n}$, which are supposed to be equal to $s_{1,c_1}, \ldots, s_{11n,c_{11n}}$.

Stage 4: (NMCom and ExtCom for checking honesty of R)

1. R commits to $(a_1^R, d_1^R), \ldots (a_{11n}^R, d_{11n}^R)$ using NMCom and identity id_R. Let $e_1^R, \ldots, e_{11n}^R$ be the decommitments.
2. R commits to $(a_1^R, d_1^R, e_1^R), \ldots (a_{11n}^R, d_{11n}^R, e_{11n}^R)$ using ExtCom.

COMMENT: *Roughly, the commitments in this stage, along with the cut-and-choose in the next stage, will be used in the security proof to argue that even cheating R must behave honestly in most instances of* mS-OT *in Stage 3. A key point is that given the values that are committed to in* NMCom *or* ExtCom *in this stage, one can obtain the random coins that R obtained in Stage 2 and thus can check whether R behaved honestly in Stage 3.*

Stage 5: (Cut-and-choose against R)

1. S reveals Γ_S by sending ϕ^S and the decommitment information w.r.t. Stage-0-1 RobCom of ϕ^S.
2. For every $i \in \Gamma_S$, R reveals (a_i^R, d_i^R, e_i^R) by decommitting the i-th ExtCom commitment in Stage 4.
3. For every $i \in \Gamma_S$, S checks the following.
   (a) $((a_i^R, d_i^R), e_i^R)$ is a valid decommitment of the i-th NMCom commitment in Stage 4.
   (b) d_i^R is a valid decommitment of ψ_i^R w.r.t. Stage-0-1 RobCom, and $a_i^R \oplus \psi_i^R$ equals the value z_i^R it received in Stage-2-1.
   (c) R executed the i-th mS-OT in Stage 3 honestly using $c_i \| \tau_i^R$, which is obtained from $r_i^R = a_i^R \oplus b_i^R$ as specified by the protocol.

COMMENT: *In other words, for each index that it randomly selected in Stage 1, S checks whether R behaved honestly in Stages 3 and 4 on that index.*

Stage 6: (OT combiner) Let $\Delta := [11n] \setminus \Gamma_S$.

1. R sends $\alpha_i := u \oplus c_i$ to S for every $i \in \Delta$.
2. S computes a $(6n+1)$-out-of-$10n$ secret sharing of v_0, denoted by $\boldsymbol{\rho}_0 = (\rho_{0,i})_{i\in\Delta}$, and computes a $(6n+1)$-out-of-$10n$ secret sharing of v_1, denoted by $\boldsymbol{\rho}_1 = (\rho_{1,i})_{i\in\Delta}$. Then, S sends $\beta_{b,i} := \rho_{b,i} \oplus s_{i,b\oplus\alpha_i}$ to R for every $i \in \Delta$, $b \in \{0,1\}$.
3. R computes $\widetilde{\rho}_i := \beta_{u,i} \oplus \widetilde{s}_i$ for every $i \in \Delta$. Let $\widetilde{\boldsymbol{\rho}} := (\widetilde{\rho}_i)_{i\in\Delta}$.

COMMENT: *In this stage, S and R execute OT with their true inputs by using the outputs of* mS-OT *in Stage 3. Roughly speaking, this stage is secure as long as most instances of* mS-OT *in Stage 3 are correctly executed.*

Stage 7: (NMCom and ExtCom for checking honesty of S)

1. S commits to $(a_1^S, d_1^S), \dots (a_{11n}^S, d_{11n}^S)$ using NMCom and identity id_S. Let $e_1^S, \dots, e_{11n}^S$ be the decommitments.
2. S commits to $(a_1^S, d_1^S, e_1^S), \dots (a_{11n}^S, d_{11n}^S, e_{11n}^S)$ using ExtCom.

Stage 8: (Cut-and-choose against S)

1. R reveals Γ_R by sending ϕ^R and the decommitment information w.r.t. Stage-0-2 RobCom of ϕ^R.
2. For every $i \in \Gamma_R$, S reveals (a_i^S, d_i^S, e_i^S) by decommitting the i-th ExtCom commitment in Stage 7.
3. For every $i \in \Gamma_R$, R checks the following.
   (a) $((a_i^S, d_i^S), e_i^S)$ is a valid decommitment of the i-th NMCom commitment in Stage 7.
   (b) d_i^S is a valid decommitment of ψ_i^S w.r.t. Stage-0-2 RobCom, and $a_i^S \oplus \psi_i^S$ equals the value z_i^S it received in Stage-2-2.
   (c) S executed the i-th mS-OT in Stage 3 honestly using $s_{i,0} \parallel s_{i,1} \parallel \tau_i^S$, which is obtained from $r_i^S = a_i^S \oplus b_i^S$ as specified by the protocol.

Parameter ν_{OT}: All messages of this OT protocol except the sender's and receiver's long messages are called **short messages**. Then, $\nu_{\text{OT}}(n)$ denotes the total length of all short messages of this protocol.

Output: R outputs $\mathsf{Value}(\widetilde{\boldsymbol{\rho}}, \Gamma_R \cap \Delta)$, where $\mathsf{Value}(\cdot,\cdot)$ is the function that is defined in Fig. 3.5.2.

COMMENT: *As in the OT protocols of [LP12, GKP18], a carefully designed reconstruction procedure* $\mathsf{Value}(\cdot,\cdot)$ *is used here so that the simulator can extract correct implicit inputs from cheating S by obtaining sharing that is sufficiently "close" to $\widetilde{\boldsymbol{\rho}}$.*

Figure 3.5.2: The function $\mathsf{Value}(\cdot,\cdot)$.

Reconstruction Procedure $\mathsf{Value}(\cdot,\cdot)$: For a sharing $\boldsymbol{s} = (s_i)_{i\in\Delta}$ and a set $\Theta \subset \Delta$, the output of $\mathsf{Value}(\boldsymbol{s},\Theta)$ is computed as follows. If $\boldsymbol{s}$ is 0.9-close to a valid codeword $\boldsymbol{w} = (w_i)_{i\in\Delta}$ that satisfies $s_i = w_i$ for every $i \in \Theta$, then $\mathsf{Value}(\boldsymbol{s},\Theta)$ is the value decoded from $\boldsymbol{w}$; otherwise, $\mathsf{Value}(\boldsymbol{s},\Theta) = \perp$.

3.5.3 Security Proof

The security proof for our OT protocol is similar to that of [GKP18], except that we substitute the "brute-force" extraction of the simulator with polynomial-time straight-line extractions to learn the adversary's secrets. As mentioned in the technical overview part, our modification does not introduce new malleability issues, and the session-by-session substitution in the hybrids of [GKP18] will still apply (with careful modification). We give the full security proof in Section 3.7.

3.6 Our Bounded-Concurrent MPC Protocol

In this section, we prove the following theorem.

Theorem 3.6.1. *Assume the existence of constant-round semi-honest oblivious transfer protocols and collision-resistant hash functions. Let $\mathcal{F}$ be any well-formed functionality. Then, for every polynomial m, there exists a constant-round protocol that securely computes $\mathcal{F}$ under m-bounded concurrent composition; furthermore, it uses the underlying primitives in the black-box way.*

The protocol and the proofs are identical to those in [GKP18] except that we use the bounded-concurrent secure OT protocol described in previous section. We now provide more details. We focus on the two-party case below (the MPC case is analogous).

Protocol Description. Roughly speaking, we obtain our bounded-concurrent 2PC protocol by composing our bounded-concurrent OT protocol in Section 3.5 with a UC-secure OT-hybrid 2PC protocol. Concretely, let Π_{OT} be our ℓ-bounded-concurrent OT protocol in Section 3.5, and $\Pi_{\mathrm{2PC}}^{\mathcal{F}_{\mathrm{OT}}}$ be a UC-secure OT-hybrid 2PC protocol with the following property: The two parties use the OT functionality $\mathcal{F}_{OT}$ only at the beginning of the protocol, and they send only randomly chosen inputs to $\mathcal{F}_{OT}$. Then, we obtain our bounded-concurrent 2PC protocol Π_{2PC} by replacing each invocation of $\mathcal{F}_{OT}$ in $\Pi_{\mathrm{2PC}}^{\mathcal{F}_{\mathrm{OT}}}$ with an execution of Π_{OT} (i.e. , the two parties execute Π_{OT} instead of calling to $\mathcal{F}_{OT}$), where all the executions of Π_{OT} are carried out in a synchronous manner, i.e. , in a manner that the first message of all the executions are sent before the second message of any execution is sent etc.; furthermore, the bounded-concurrency parameter for Π_{OT} is set to be m' defined as follows: let ν_{2PC} denote the length of all messages of the hybrid 2PC protocol $\Pi_{\mathrm{2PC}}^{\mathcal{F}_{\mathrm{OT}}}$ protocol (which does not include the length of messages corresponding to OT calls since we are in the hybrid model). Then, we set m' so that the length ℓ of long messages of Π_{OT} would be n bits longer than $\nu_{\mathrm{OT}} + \nu_{\mathrm{2PC}}$. This can be ensured by setting $m' = a \cdot m$ where a is the smallest integer that is bigger than $\max(\nu_{\mathrm{OT}}/\nu_{\mathrm{2PC}}, \nu_{\mathrm{2PC}}/\nu_{\mathrm{OT}})$.

As the UC-secure OT-hybrid 2PC protocol, we use the constant-round 2PC (actually, MPC) protocol of Ishai et al. [IPS08], which makes only black-box use of pseudorandom generators (which in turn can be obtained in the black-box way from any semi-honest OT protocol). (The protocol of Ishai et al. [IPS08] itself does not satisfy the above property, but as shown in [GKP18], it can be easily modified to satisfy it.) Since the OT-hybrid protocol of Ishai et al. [IPS08] (as well as its modification in [GKP18]) is a black-box construction and has only constant number of rounds, our protocol Π_{2PC} is also a black-box construction and has only constant number of rounds.

The security of this protocol can be proved in a similar way as our OT protocol. The formal proof is given in Section 3.8.

3.7 Security Proof For Our OT Protocol

3.7.1 Simulator $\mathsf{Sim}_{\mathrm{OT}}$

To prove the security of Π_{OT}, we consider the following simulator $\mathsf{Sim}_{\mathrm{OT}}$. Recall that our goal is to prove that Π_{OT} securely realizes $\mathcal{F}_{OT}$ (see Figure 3.5.1) under m-bounded concurrent (self) composition. We therefore consider a simulator that works against adversaries that participate in at most (and w.l.o.g., exactly) m sessions of Π_{OT} both as senders and as receivers.

Let $\mathcal{A}$ be any adversary that participates in m sessions of Π_{OT}. Our simulator $\mathsf{Sim}_{\mathrm{OT}}$ internally invokes $\mathcal{A}$ and simulates each of the sessions for $\mathcal{A}$ as follows.

When R is corrupted: In a session where the receiver R is corrupted, $\mathsf{Sim}_{\mathrm{OT}}$ simulates the sender S for $\mathcal{A}$ by extracting the implicit input $u^* \in \{0,1\}$ from $\mathcal{A}$. During the simulation, $\mathsf{Sim}_{\mathrm{OT}}$ extracts the values ϕ^R and ψ^R (in straight-line, using the code of $\mathcal{A}$) such that it can later extract the value Γ^R and $\boldsymbol{a}^R$ in Stages 1 and 2 ; the former extraction is needed to execute most instances mS-OT in Stage 3 with true randomness (which is crucial to use their security in the analysis), and the latter extraction is needed to infer what information $\mathcal{A}$ obtained in the mS-OT instances in Stage 3 (which is crucial to extract the implicit input $u^* \in \{0,1\}$ from $\mathcal{A}$).

Concretely, $\mathsf{Sim}_{\mathrm{OT}}$ simulates all steps of Stage 0 in the same way as an honest S, except that in Stage 0-2, $\mathsf{Sim}_{\mathrm{OT}}$ uses the strategy of the (straight-line) extractor for Protocol 3.4.1 in its interaction with $\mathcal{A}$. At the end of this Stage 0-2, it learns the value ϕ^R and $\psi^R = \psi_1^R \| \dots \| \psi_{11n}^R$ committed by $\mathcal{A}$.

Remark 3.7.1. *Note that $\mathsf{Sim}_{\mathrm{OT}}$ can extract Γ_R using ϕ^R in Stage 1-2. Likewise, in Stage 2-2 it can extract $\boldsymbol{a}^R$ using relevant parts of ψ^R.*

Next, from Stage 1 to Stage 5, $\mathsf{Sim}_{\mathrm{OT}}$ interacts with $\mathcal{A}$ in the same way as an honest S except for the following.

– For the commitments from $\mathcal{A}$ in Stages 1-2 and 2-2, the committed subset Γ_R and the committed strings $\boldsymbol{a}^R = (a_1^R, \dots, a_{11n}^R)$ are extracted by $\mathsf{Sim}_{\mathrm{OT}}$ as described in Remark 3.7.1.

  $\mathsf{Sim}_{\mathrm{OT}}$ then defines $\boldsymbol{r}^R = (r_1^R, \dots, r_{11n}^R)$ by $r_i^R \stackrel{\text{def}}{=} a_i^R \oplus b_i^R$ for each $i \in [11n]$ and parses r_i^R as $c_i \,\|\, \tau_i^R$ for each $i \in [11n]$. (Notice that $\boldsymbol{r}^R$ is the outcome of the coin-tossing that $\mathcal{A}$

must have obtained.)

- In Stage 3, the i-th mS-OT is executed with a random input and true randomness rather than with $(s_{i,0}, s_{i,1})$ and τ_i^S for every $i \notin \Gamma_R$.

In Stage 6, $\mathsf{Sim}_{\mathrm{OT}}$ interacts with $\mathcal{A}$ as follows.

1. Receive $\{\alpha_i\}_{i\in\Delta}$ from $\mathcal{A}$ in Stage 6-1.
2. Determine the implicit input u^* of $\mathcal{A}$ as follows. Let I_0, I_1 be the sets such that for $b \in \{0,1\}$ and $i \in \Delta$, we have $i \in I_b$ if and only if:

   - $i \in \Gamma_R$, or
   - $\mathcal{A}$ did not execute the i-th mS-OT in Stage 3 honestly using $c_i \parallel \tau_i^R$ as the input and randomness, or
   - $c_i = b \oplus \alpha_i$, and $\mathcal{A}$ executed the i-th mS-OT in Stage 3 honestly using $c_i \parallel \tau_i^R$ as the input and randomness.

   Abort the simulation if both of $|I_0| \geq 6n+1$ and $|I_1| \geq 6n+1$ hold. Otherwise, define u^* by $u^* \stackrel{\text{def}}{=} 0$ if $|I_0| \geq 6n+1$ and $u^* \stackrel{\text{def}}{=} 1$ otherwise. (Roughly, $|I_b|$ is the number of strings that $\mathcal{A}$ can obtain out of $\{s_{i,b\oplus\alpha_i}\}_{i\in\Delta}$ by requiring S to reveal them in Stage 8, by cheating in mS-OT, or by executing mS-OT honestly with input $b \oplus \alpha_i$. We remind the readers that $\{s_{i,b\oplus\alpha_i}\}_{i\in\Delta}$ are the strings that are used to mask $\boldsymbol{\rho}_b = (\rho_{b,i})_{i\in\Delta}$ in Stage 6.)
3. Send u^* to the ideal functionality and obtains v^*.
4. Subsequently, interact with $\mathcal{A}$ in the same way as an honest S assuming that the inputs to S are $v_{u^*} = v^*$ and random v_{1-u^*}.

From Stage 7 to Stage 8, $\mathsf{Sim}_{\mathrm{OT}}$ interacts with $\mathcal{A}$ in the same way as an honest S except that in Stage 7, an all-zero string is committed in the i-th NMCom rather than (a_i^S, d_i^S) for every $i \notin \Gamma_R$, and an all-zero string is committed in the i-th ExtCom rather than (a_i^S, d_i^S, e_i^S) for every $i \notin \Gamma_R$.

When S is corrupted: In a session where the sender S is corrupted, $\mathsf{Sim}_{\mathrm{OT}}$ simulates the receiver R for $\mathcal{A}$ by extracting the implicit input v_0^*, v_1^* from $\mathcal{A}$. During the simulation, $\mathsf{Sim}_{\mathrm{OT}}$ extracts the values ϕ^S and ψ^S (in straight-line, using the code of $\mathcal{A}$) such that it can later extract the value Γ^S and $\boldsymbol{a}^S$ in Stages 1 and 2; the former extraction is needed to execute most instances $\mathsf{mS\text{-}OT}$ in Stage 3 with true randomness (which is crucial to use their security in the analysis), and the latter extraction is needed to learn what input $\mathcal{A}$ used in the $\mathsf{mS\text{-}OT}$ instances in Stage 3 (which is crucial to extract the implicit input v_0^*, v_1^* from $\mathcal{A}$).

Concretely, $\mathsf{Sim}_{\mathrm{OT}}$ simulates all steps of Stage 0 in the same way as an honest R, except that in Stage 0-1, $\mathsf{Sim}_{\mathrm{OT}}$ uses the strategy of the (straight-line) extractor for Protocol 3.4.1 in its interaction with $\mathcal{A}$. At the end of this Stage 0-1, it learns the value ϕ^S and $\psi^S = \psi_1^S \| \ldots \| \psi_{11n}^S$ committed by $\mathcal{A}$.

Remark 3.7.2. *As in Remark 3.7.1, $\mathsf{Sim}_{\mathrm{OT}}$ can extract Γ_S and $\boldsymbol{a}^S$ in Stage 1-1 and Stage 2-1 respectively.*

Next, $\mathsf{Sim}_{\mathrm{OT}}$ interacts with $\mathcal{A}$ in the same way as an honest R in all the stages except for the following.

– For the commitments from $\mathcal{A}$ in Stages 1-1, the committed subset Γ_S is extracted by Sim_{OT} as described in Remark 3.7.2.

– In Stage 3, the i-th mS-OT is executed with a random input and true randomness rather than with c_i and τ_i^R for every $i \notin \Gamma_S$.

– In Stage 4, an all-zero string is committed in the i-th NMCom rather than (a_i^S, d_i^S) for every $i \notin \Gamma_S$, and an all-zero string is committed in the i-th ExtCom rather than (a_i^S, d_i^S, e_i^S) for every $i \notin \Gamma_S$.

– In Stage 6, α_i is a random bit rather than $\alpha_i = u \oplus c_i$ for every $i \in \Delta$, and $\widetilde{\rho}_i$ is not computed for any $i \in \Delta$.

Then, Sim_{OT} determines the implicit inputs v_0^*, v_1^* of $\mathcal{A}$ as follows.

1. For the commitments from $\mathcal{A}$ in Stage 2-1, the committed strings $\boldsymbol{a}^S = (a_1^S, \ldots, a_{11n}^S)$ are extracted by Sim_{OT} as described in Remark 3.7.2.
2. Define $\boldsymbol{r}^S = (r_1^S, \ldots, r_{11n}^S)$ by $r_i^S \stackrel{\text{def}}{=} a_i^S \oplus b_i^S$ for each $i \in [11n]$ and parse r_i^S as $s_{i,0} \,\|\, s_{i,1} \,\|\, \tau_i^S$ for each $i \in [11n]$. (Notice that $\boldsymbol{r}^S$ is the outcome of the coin-tossing that $\mathcal{A}$ must have obtained.)
3. Define $\boldsymbol{\rho}_b^{\mathsf{ext}} = (\rho_{b,i}^{\mathsf{ext}})_{i\in\Delta}$ for each $b \in \{0,1\}$ as follows: $\rho_{b,i}^{\mathsf{ext}} \stackrel{\text{def}}{=} \beta_{b,i} \oplus s_{i,b\oplus\alpha_i}$ if $\mathcal{A}$ executed the i-th mS-OT in stage 3 honestly using $s_{i,0} \,\|\, s_{i,1} \,\|\, \tau_i^S$, and $\rho_{b,i}^{\mathsf{ext}} \stackrel{\text{def}}{=} \bot$ otherwise.
4. For each $b \in \{0,1\}$, define $v_b^* \stackrel{\text{def}}{=} \mathsf{Value}(\boldsymbol{\rho}_b^{\mathsf{ext}}, \Gamma_R \cap \Delta)$.

Then, Sim_{OT} sends v_0^*, v_1^* to the ideal functionality if both of the following hold for each $b \in \{0,1\}$:

1. $|\{i \in \Delta \text{ s.t. } \rho_{b,i}^{\mathsf{ext}} \neq \bot\}| < 0.1n$.
2. $\boldsymbol{\rho}_b^{\mathsf{ext}}$ is either 0.9-close to a valid codeword $\boldsymbol{w} = (w_i)_{i\in\Delta}$ that satisfies $w_i = \rho_{b,i}^{\mathsf{ext}}$ for every $i \in \Gamma_R$ or 0.14-far from any such valid codeword.

Otherwise (i.e. if there exists $b \in \{0,1\}$) such that one of the above does not holds), Sim_{OT} aborts the simulation.

3.7.2 Proof of Indistinguishability

We show the indistinguishability by using a hybrid argument. Before defining hybrid experiments, we define *special messages*, which we use in the definitions of the hybrid experiments. (Essentially, they are the messages on extracted by the simulator in straight-line using the code of $\mathcal{A}$.)

– first special message is the message sent by S in Stage 1-1 (which is supposed to be $\Gamma^S \oplus \phi^S$).

– second special message is the message sent by R in Stage 1-2 (which is supposed to be $\Gamma^R \oplus \phi^R$).

– third special message is the message sent by S in Stage 2-1 (which is supposed to be $\{z_i^S\}_{i\in[11n]}$).

– $\mathsf{fourth\ special\ message}$ is the message sent by R in Stage 2-2 (which is supposed to be $\{z_i^R\}_{i\in[11n]}$).

3.7.2.1 Hybrid experiments

Now, we define hybrid experiments. Let m be the bound on the number of the sessions that $\mathcal{A}$ starts. Note that the number of special messages among m sessions can be bounded by $4m$. We order those $4m$ special messages by the order of their appearances; we use SM_k to denote the k-th special message, and $s(k)$ to denote the session that SM_k belongs to.

We start by defining hybrids H_0, H_0^* and $H_{k:1}, \ldots, H_{k:7}$ for $k \in [4m]$.(For convenience, in what follows we occasionally denote H_0^* as $H_{0:7}$.)

Remark 3.7.3 (Rough idea of the hybrids)**.** *In the sequence of the hybrid experiments, we gradually modify the real-world experiment to the ideal-world one. We make sure that $H_{k:i}$ ($i \in [7]$) deviates from the previous hybrid only after SM_k. These properties help us prove the indistinguishability of each neighboring hybrids by using the extracted commitment as non-uniform advice and rely on the non-uniform security of the underlying primitives to prove indistinguishability.*[10]

Hybrid H_0. H_0 is the same as the real experiment.

Hybrid H_0^*. Recall that Stage 0-1 (resp. Stage 0-2) contains $11n+1$ (independent) parallel executions of RobCom (Protocol 3.4.1), where S (resp. R) commits to $\phi^S, \psi_1^S, \ldots, \psi_{11n}^S$ (resp. $\phi^R, \psi_1^R, \ldots, \psi_{11n}^R$). Hybrid H_0^* is identical to H_0 except that for each session $i \in [m]$

– if S is corrupted, $\mathsf{Sim}_{\mathrm{OT}}$ uses the (straight-line) extractor's strategy of Protocol 3.4.1 in all the $11n+1$ RobCom executions in Stage 0-1b;

– if R is corrupted, $\mathsf{Sim}_{\mathrm{OT}}$ uses the (straight-line) extractor's strategy of Protocol 3.4.1 in all the $11n+1$ RobCom executions in Stage 0-2b.

Note that in hybrid H_0^*, $\mathsf{Sim}_{\mathrm{OT}}$ extracts all the values $\phi^S, \psi_1^S, \| \ldots, \psi_{11n}^S$ for each session $i \in [m]$ where S is corrupted, and all the values $\phi^R, \psi_1^R, \| \ldots, \psi_{11n}^R$ for each session $i \in [m]$ where R is corrupted. With these values, the hybrid is able to

– *(if S is corrupted)* extract the values of Γ_S and $\boldsymbol{a}^S$ that S commits to in Stages 1-1 and 2-1 respectively, as described in Remark 3.7.2;

– *(if R is corrupted)* extracts the values of Γ_R and $\boldsymbol{a}^R$ that R commits to in Stages 1-2 and 2-2 respectively, as described in Remark 3.7.1;

For future use, these extracted values are stored in a global table $\mathcal{T}$ with the corresponding session number.

Hybrid $H_{k:1}$. $H_{k:1}$ is the same as $H_{k-1:7}$ except that in session $s(k)$, if S is corrupted and SM_k is $\mathsf{first\ special\ message}$,

– Query table $\mathcal{T}$ to get the extracted value Γ_S corresponding to session $s(k)$,

[10]We remark that, unlike [GKP18], in our case it is possible to get rid of the non-uniform argument by using (a slightly more involved) averaging argument since in our case, the extraction procedure is polynomial time. The proof using non-uniform advice is simpler.

– the value committed to in the i-th NMCom commitment in Stage 4 is switched to an all-zero string for every $i \notin \Gamma_S$,

– the value committed to in the i-th ExtCom commitment in Stage 4 is switched to an all-zero string for every $i \notin \Gamma_S$.

Hybrid $H_{k:2}$. $H_{k:2}$ is the same as $H_{k:1}$ except that in session $s(k)$, if S is corrupted and SM_k is $\mathsf{first\ special\ message}$, the i-th mS-OT in Stage 3 is executed with a random input and true randomness for every $i \notin \Gamma_S$.

Hybrid $H_{k:3}$. $H_{k:3}$ is the same as $H_{k:2}$ except that in session $s(k)$, if S is corrupted and SM_k is $\mathsf{third\ special\ message}$, the following modifications are made.

1. Query table $\mathcal{T}$ to get the extracted value $\boldsymbol{a}^S$ corresponding to session $s(k)$. Define $\boldsymbol{r}^S = (r_1^S, \ldots, r_{11n}^S)$ by $r_i^S \overset{\text{def}}{=} a_i^S \oplus b_i^S$ for each $i \in [11n]$, and parse r_i^S as $s_{i,0} \,\|\, s_{i,1} \,\|\, \tau_i^S$ for each $i \in [11n]$. Define $\boldsymbol{\rho}_b^{\mathsf{ext}} = (\rho_{b,i}^{\mathsf{ext}})_{i \in \Delta}$ for each $b \in \{0,1\}$ as follows: $\rho_{b,i}^{\mathsf{ext}} \overset{\text{def}}{=} \beta_{b,i} \oplus s_{i,b\oplus\alpha_i}$ if $\mathcal{A}$ executed the i-th mS-OT in stage 3 honestly using $s_{i,0} \,\|\, s_{i,1} \,\|\, \tau_i^S$, and $\rho_{b,i}^{\mathsf{ext}} = \perp$ otherwise.
2. R outputs $\mathsf{Value}(\boldsymbol{\rho}_u^{\mathsf{ext}}, \Gamma_R \cap \Delta)$ rather than $\mathsf{Value}(\widetilde{\rho}, \Gamma_R \cap \Delta)$. (Recall that u is the real input to R.) if both of the following hold for each $b \in \{0,1\}$:

   (a) $|\{i \in \Delta \text{ s.t. } \rho_{b,i}^{\mathsf{ext}} \neq \perp\}| < 0.1n$.

   (b) $\boldsymbol{\rho}_b^{\mathsf{ext}}$ is either 0.9-close to a valid codeword $\boldsymbol{w} = (w_i)_{i\in\Delta}$ that satisfies $w_i = \rho_{b,i}^{\mathsf{ext}}$ for every $i \in \Gamma_R$ or 0.15-far from any such valid codeword.

   Otherwise (i.e. if there exists $b \in \{0,1\}$) such that one of the above does not holds), the execution of the hybrid is aborted.

Hybrid $H_{k:4}$. $H_{k:4}$ is the same as $H_{k:3}$ except that in session $s(k)$, if S is corrupted and SM_k is $\mathsf{third\ special\ message}$, α_i is a random bit rather than $\alpha_i = u \oplus c_i$ for every $i \in \Delta$ in Stage 6-1 and $\widetilde{\rho}_i$ is no longer computed for any $i \in \Delta$ in Stage 6-3.

Hybrid $H_{k:5}$. $H_{k:5}$ is the same as $H_{k:4}$ except that in session $s(k)$, if R is corrupted and SM_k is $\mathsf{second\ special\ message}$,

– Query table $\mathcal{T}$ to get the extracted value Γ_R corresponding to session $s(k)$,

– the value committed in the i-th NMCom commitment in Stage 7 is switched to an all-zero string for every $i \notin \Gamma_R$,

– the value committed in the i-th ExtCom commitment in Stage 7 is switched to an all-zero string for every $i \notin \Gamma_R$.

Hybrid $H_{k:6}$. $H_{k:6}$ is the same as $H_{k:5}$ except that in session $s(k)$, if R is corrupted and SM_k is $\mathsf{second\ special\ message}$, the i-th mS-OT in Stage 3 is executed with a random input and true randomness for every $i \notin \Gamma_R$.

Hybrid $H_{k:7}$. $H_{k:7}$ is the same as $H_{k:6}$ except that in session $s(k)$, if R is corrupted and SM_k is $\mathsf{fourth\ special\ message}$, the following modifications are made.

1. Query table $\mathcal{T}$ to get the extracted value $\boldsymbol{a}^R$ corresponding to session $s(k)$. Define $\boldsymbol{r}^R = (r_1^R, \ldots, r_{11n}^R)$ by $r_i^R \overset{\text{def}}{=} a_i^R \oplus b_i^R$ for each $i \in [11n]$, and parse r_i^R as $c_i \,\|\, \tau_i^R$ for each $i \in [11n]$.

Define u^* as follows. Let I_0 and I_1 be the set such that for $b \in \{0,1\}$ and $i \in \Delta$, we have $i \in I_b$ if and only if:

– $i \in \Gamma_R$, or
– $\mathcal{A}$ did not execute the i-th mS-OT in Stage 3 honestly using $c_i \,\|\, \tau_i^R$ as the input and randomness, or
– $c_i = b \oplus \alpha_i$, and $\mathcal{A}$ executed the i-th mS-OT in Stage 3 honestly using $c_i \,\|\, \tau_i^R$ as the input and randomness.

Abort the execution if both of $|I_0| \geq 6n+1$ and $|I_1| \geq 6n+1$ hold. Otherwise, define u^* by $u^* \stackrel{\text{def}}{=} 0$ if $|I_0| \geq 6n+1$ and $u^* \stackrel{\text{def}}{=} 1$.

2. In Stage 6, $\boldsymbol{\rho}_{1-u^*}$ is a secret sharing of a random bit rather than that of v_{1-u^*}.

We remark that in $H_{4m:7}$, all the messages from the honest parties and their output are computed as in Sim_{OT}.

3.7.2.2 Indistinguishability of each neighboring hybrids

Below, we show that each neighboring hybrids are indistinguishable, and additionally show, for technical reasons, that an invariant condition holds in each session of every hybrid.

First, we define the invariant condition.

Definition 3.7.1 (Invariant Condition (when R is corrupted))**.** *For any session in which R is corrupted, we say that the invariant condition holds in that session if the following holds when the cut-and-choose in Stage 5 is accepted.*

1. Let $(\hat{a}_1^R, \hat{d}_1^R), \ldots (\hat{a}_{11n}^R, \hat{d}_{11n}^R)$ be the values that are committed in NMCom *in Stage 4. Let $I_{\text{bad}} \subset [11n]$ be the set such that $i \in I_{\text{bad}}$ if and only if*

(a) $(\hat{a}_i^R, \hat{d}_i^R)$ is not valid in terms of the check in Stage 5-3b, i.e. $\hat{d}_i^R$ is not a valid decommitment of ψ_i^R w.r.t. Stage-0-2 RobCom*, or $\hat{a}_i^R \oplus \psi_i^R$ does not equal the value z_i^R it received in Stage-2-2;* **or**

(b) R does not execute the i-th mS-OT in Stage 3 honestly using $\hat{c}_i \,\|\, \hat{\tau}_i^R$ as the input and randomness, where $\hat{c}_i \,\|\, \hat{\tau}_i^R$ is obtained from $\hat{r}_i^R = \hat{a}_i^R \oplus b_i^R$.

Then, it holds that $|I_{\text{bad}}| < n$.

Remark 3.7.4. *Roughly speaking, this condition guarantees that most of the mS-OTs in Stage 3 are honestly executed using the outcome of the coin tossing, which in turn guarantees that the cheating receiver's input can be extracted by extracting the outcome of the coin tossing.*

Remark 3.7.5. *When Stage 5 is accepted, we also have $I_{\text{bad}} \cap \Gamma_S = \emptyset$ from the definition of I_{bad}.*

Definition 3.7.2 (Invariant Condition (when S is corrupted))**.** *For any session in which S is corrupted, we say that the invariant condition holds in that session if the following hold when the cut-and-choose in Stage 8 is accepted.*

1. *Let $(\hat{a}^S_1, \hat{d}^S_1), \dots (\hat{a}^S_{11n}, \hat{d}^S_{11n})$ be the values that are committed in* NMCom *in Stage 7. Let $I_{\mathrm{bad}} \subset [11n]$ be the set such that $i \in I_{\mathrm{bad}}$ if and only if*
   (a) *$(\hat{a}^S_i, \hat{d}^S_i)$ is not valid in terms of the check in Stage 8-3b, i.e. $\hat{d}^S_i$ is not a valid decommitment of ψ^S_i w.r.t. Stage-0-1* RobCom*, or $\hat{a}^S_i \oplus \psi^S_i$ does not equal the value z^S_i it received in Stage-2-1;* **or**
   (b) *S does not execute the i-th mS-OT in Stage 3 honestly using $\hat{s}_{i,0} \,\|\, \hat{s}_{i,1} \,\|\, \hat{\tau}^S_i$ as the input and randomness, where $\hat{s}_{i,0} \,\|\, \hat{s}_{i,1} \,\|\, \hat{\tau}^S_i$ is obtained from $\hat{r}^S_i = \hat{a}^S_i \oplus b^S_i$.*

   Then, it holds that $|I_{\mathrm{bad}}| < 0.1n$.
2. *For each $b \in \{0,1\}$, define $\boldsymbol{\rho}^{\mathsf{nm}}_b = (\rho^{\mathsf{nm}}_{b,i})_{i\in\Delta}$ as follows: $\rho^{\mathsf{nm}}_{b,i} \overset{\mathrm{def}}{=} \beta_{b,i} \oplus \hat{s}_{i,b\oplus\alpha_i}$ if $i \notin I_{\mathrm{bad}}$ and $\rho^{\mathsf{nm}}_{b,i} \overset{\mathrm{def}}{=} \perp$ otherwise. Then, for each $b \in \{0,1\}$, ρ^{nm}_b is either 0.9-close to a valid codeword $\boldsymbol{w} = (w_i)_{i\in\Delta}$ that satisfies $w_i = \rho^{\mathsf{nm}}_{b,i}$ for every $i \in \Gamma_R$ or 0.15-far from any such valid codeword.*

Remark 3.7.6. *Roughly speaking, this condition guarantees that the cheating sender's input can be extracted from the outcome of the coin tossing. In particular, it guarantees that the sharing that is computed from the outcome of mS-OTs (i.e., the sharing that is computed by the honest receiver) and the sharing that is computed from the outcome of the coin tossing (i.e., the sharing that is computed by the simulator) are very "close" (see Claim 3.7.4 below).*

Remark 3.7.7. *When Stage 8 is accepted, we also have $I_{\mathrm{bad}} \cap \Gamma_R = \emptyset$ from the definition of I_{bad}.*

Next, we show that the invariant condition holds in every session in H_0 (i.e., the real experiment).

Definition 3.7.3. *We say that $\mathcal{A}$* cheats *in a session if the invariant condition does not hold in that session.*

Next, we establish the computational indistinguishability among hybrids by a sequence of lemmata. We start with the following lemma:

Lemma 3.7.1. *In H_0, $\mathcal{A}$ does not cheat in every session except with negligible probability.*

Proof. The proof of this lemma is identical to the proof in [GKP18]. We include it here for completeness.

Assume for contradiction that in H_0, $\mathcal{A}$ cheats in a session with non-negligible probability. Since the number of the sessions is bounded by a polynomial, there exists a function $i^*(\cdot)$ and a polynomial $p(\cdot)$ such that for infinitely many n, $\mathcal{A}$ cheats in the $i^*(n)$-th session with probability at least $1/p(n)$; furthermore, since $\mathcal{A}$ cheats only when either R or S is corrupted, in the $i^*(n)$-th session either R is corrupted for infinitely many such n or S is corrupted for infinitely many such n. In both cases, we derive contradiction by using $\mathcal{A}$ to break the hiding property of RobCom.

Case 1. R is corrupted in the $i^*(n)$-th session. We show that when $\mathcal{A}$ cheats, we can break the hiding property of the $\mathsf{RobCom}(\phi^S)$ commitment in Stage 0-1 (i.e. , the commitment by which ϕ^S is committed to). From the definition of the invariant condition (Definition 3.7.1), when $\mathcal{A}$ cheats, we have $|I_{\mathrm{bad}}| \geq n$ even though the cut-and-choose in Stage 5 is accepting (and hence $I_{\text{bad}} \cap \Gamma_S = \emptyset$ as remarked in Remark 3.7.5), where $I_{\text{bad}} \subseteq [11n]$ is the set defined from the committed values of the NMCom commitments in Stage 4. If we can compute I_{bad} efficiently, we can use it to distinguish Γ_S from a random subset of size n (this is because a random subset Γ of size n satisfies $I_{\text{bad}} \cap \Gamma = \emptyset$ only with negligible probability when $|I_{\text{bad}}| \geq n$), so we can use it to break the hiding property of the RobCom commitment to ϕ^S, which is used to mask Γ_S. However, I_{bad} is not efficiently computable since the committed values of the NMCom commitments are not efficiently computable. We thus first show that we can "approximate" I_{bad} by extracting the committed values of the ExtCom commitments in Stage 4. Details are given below.

First, we observe that if we extract the committed values of the ExtCom commitments in Stage 4 of the $i^*(n)$-th session, the extracted values $(\hat{a}_1^R, \hat{d}_1^R, \hat{e}_1^R), \ldots, (\hat{a}_{11n}^R, \hat{d}_{11n}^R, \hat{e}_{11n}^R)$, satisfy the following condition.

– Let $\hat{I}_{\text{bad}} \subset [11n]$ be a set such that $i \in \hat{I}_{\text{bad}}$ if and only if

1. $((\hat{a}_i^R, \hat{d}_i^R), \hat{e}_i^R)$ is not a valid decommitment of the i-th NMCom commitment in Stage 4; **or**
2. $(\hat{a}_i^R, \hat{d}_i^R)$ is not valid in terms of the check in Stage 5-3b, i.e. $\hat{d}_i^R$ is not a valid decommitment of ψ_i^R w.r.t. Stage-0-2 RobCom, or $\hat{a}_i^R \oplus \psi_i^R$ does not equal the value z_i^R it received in Stage-2-2; **or**
3. R does not execute the i-th mS-OT in Stage 3 honestly using $\hat{c}_i \,\|\, \hat{\tau}_i^R$ as the input and randomness, where $\hat{c}_i \,\|\, \hat{\tau}_i^R$ is obtained from $\hat{r}_i^R = \hat{a}_i^R \oplus b_i^R$.

Then, $|\hat{I}_{\text{bad}}| \geq n$ and $\hat{I}_{\text{bad}} \cap \Gamma_S = \emptyset$ with probability at least $1/2p(n)$.

The extracted values satisfy this condition because when $\mathcal{A}$ cheats, we have $|\hat{I}_{\text{bad}}| \geq n$ and $\hat{I}_{\text{bad}} \cap \Gamma_S = \emptyset$ except with negligible probability. (We have $|\hat{I}_{\text{bad}}| \geq n$ since we have $I_{\text{bad}} \subset \hat{I}_{\text{bad}}$ from the definitions of $I_{\text{bad}}, \hat{I}_{\text{bad}}$ and the binding property of NMCom. We have $\hat{I}_{\text{bad}} \cap \Gamma_S = \emptyset$ since when the cut-and-choose in Stage 5 is accepting, for every $i \in \Gamma_S$ the i-th ExtCom commitment is a valid decommitment of the i-th NMCom commitment, and $I_{\text{bad}} \cap \Gamma_S = \emptyset$.)

Based on this observation, we derive contradiction by considering the following adversary $\mathcal{A}_{\mathsf{RobCom}}$ against the hiding property of RobCom.

> $\mathcal{A}_{\mathsf{RobCom}}$ receives a RobCom commitment c^* in which either ϕ_S^0 or ϕ_S^1 is committed. Then, $\mathcal{A}_{\mathsf{RobCom}}$ internally executes the experiment H_0 honestly except that in the $i^*(n)$-th session, $\mathcal{A}_{\mathsf{RobCom}}$ uses c^* as the commitment in Stage 0-1 (i.e. , as the RobCom commitment in which S commits to string which will be used to mask Γ_S in Stage 1-1). In Stage 1-1, $\mathcal{A}_{\mathsf{RobCom}}$ always use ϕ_S^1 to mask Γ_S. When the experiment H_0 reaches Stage 4 of the $i^*(n)$-th session, $\mathcal{A}_{\mathsf{RobCom}}$ extracts the committed values of the ExtCom commitments in this stage by using its extractability.[11] Let $\hat{I}_{\text{bad}} \subset [11n]$ be the set that is defined as above from the extracted values. Then, $\mathcal{A}_{\mathsf{RobCom}}$ outputs 1 if and only if $|\hat{I}_{\text{bad}}| \geq n$ and $\hat{I}_{\text{bad}} \cap \Gamma_S = \emptyset$.

If $\mathcal{A}_{\mathsf{RobCom}}$ receives a commitment to ϕ_S^1, $\mathcal{A}_{\mathsf{RobCom}}$ outputs 1 with probability at least $1/2p(n)$ (this follows from the above observation). In contrast, if $\mathcal{A}_{\mathsf{RobCom}}$ receives a commitment to

[11]This extraction involves rewinding the execution of the whole experiment, i.e. , the adversary as well as all the other parties.

ϕ_S^0, $\mathcal{A}_{\mathsf{RobCom}}$ outputs 1 with exponentially small probability (this is because $\phi_1^S \oplus \Gamma_S$ is a pure random string now, so the probability that $|\hat{I}_{\mathrm{bad}}| \geq n$ but $\hat{I}_{\mathrm{bad}} \cap \Gamma_S^1 = \emptyset$ is exponentially small). Hence, $\mathcal{A}_{\mathsf{RobCom}}$ breaks the hiding property of RobCom.

Case 2. S is corrupted in the $i^*(n)$-th session. The proof for this case is similar to (but a little more complex than) the one for Case 1. Specifically, we show that if the invariant condition does not hold, we can break the hiding property of $\mathsf{RobCom}(\phi^R)$ in Stage 0-2 by approximating I_{bad} using the extractability of ExtCom. We give a formal proof for this case in Section 3.9.1. (A somewhat similar proof is given as the proof of Lemma 3.7.5 later.) □

Next, we show the indistinguishability between each neighboring hybrids.

Lemma 3.7.2. *Hybrids H_0 and H_0^* are indistinguishable, and in H_0^*, $\mathcal{A}$ does not cheat in every sessions except with negligible probability.*

Proof. We first prove the indistinguishability by a sequence of intermediate hybrids where the RobCom is replaced one-by-one. This relies on the robust extractability of RobCom. Then, using the established indistinguishability and the robust non-malleability of NMCom, we show that $\mathcal{A}$ does not cheat in H_0^*.

We first set $\hat{H}_0^0 = H_0$. Then, we define the following sequence of m intermediate hybrids:

Hybrid $\hat{H}_0^i$ ($i \in [m]$). This hybrid is identical to $\hat{H}_0^{i-1}$ except that in session i,

- if S is corrupted, $\mathsf{Sim}_{\mathrm{OT}}$ uses the (straight-line) extractor's strategy of Protocol 3.4.1 in all the $11n+1$ RobCom executions in Stage 0-1b (in session i);
- if R is corrupted, $\mathsf{Sim}_{\mathrm{OT}}$ uses the (straight-line) extractor's strategy of Protocol 3.4.1 in all the $11n+1$ RobCom executions in Stage 0-2b (in session i);

It is easy to see that $\hat{H}_0^m = H_0^*$. Thus, to prove Lemma 3.7.2, we only need to show that each adjacent $\hat{H}_0^i$ and $\hat{H}_0^{i+1}$ is indistinguishable and $\mathcal{A}$ does not cheat in $\hat{H}_0^i$ for all $i \in [m]$. For this purpose, we provide a proof for $\hat{H}_0^0$ (i.e. H_0) and $\hat{H}_0^1$ in the following Claim 3.7.1. The same argument extends straightforwardly to other adjacent $\hat{H}_0^i$ and $\hat{H}_0^{i+1}$.

Claim 3.7.1. *Hybrids $\hat{H}_0^0$ and $\hat{H}_0^1$ are indistinguishable, and in $\hat{H}_0^1$, $\mathcal{A}$ does not cheat in every sessions except with negligible probability.*

Proof. Note that our OT protocol contains $11n+1$ RobCom executions in Stage 0-1b (and in Stage 0-2b). To conduct the proof, we actually need a finer-grained sequence of hybrids, which is listed in the following. To provide an intuitive explanation, the following hybrids are obtained by inserting $11n+1$ hybrids between $\hat{H}_0^0$ and $\hat{H}_0^1$, where the $11n+1$ RobCom instances are substituted one-by-one.

Hybrid $\hat{H}_0^{0:i}$ ($i \in [11n+1]$). this hybrid is the same as $\hat{H}_0^0$ except that

- if S is corrupted in session 1, $\mathsf{Sim}_{\mathrm{OT}}$ uses the (straight-line) extractor's strategy of Protocol 3.4.1 in *the first i* of the $11n+1$ RobCom executions in Stage 0-1b;
- if R is corrupted in session 1, $\mathsf{Sim}_{\mathrm{OT}}$ uses the (straight-line) extractor's strategy of Protocol 3.4.1 in *the first i* of the $11n+1$ RobCom executions in Stage 0-2b;

It is easy to see that $\hat{H}_0^{0:11n+1} = \hat{H}_0^1$. To prove Claim 3.7.1, we need to show that each adjacent $\hat{H}_0^{0:i}$ and $\hat{H}_0^{0:i+1}$ is indistinguishable and $\mathcal{A}$ does not cheat in $\hat{H}_0^{0:i}$ for all $i \in [11n+1]$. In the following, we focus on the switch between $\hat{H}_0^{0:0}$ and $\hat{H}_0^{0:1}$ (the same argument extends to the switch from $\hat{H}_0^{0:i-1}$ to $\hat{H}_0^{0:i}$ for all $i \in [11n+1]$). Concretely, in the following we prove that:

– $\hat{H}_0^{0:0}$ and $\hat{H}_0^{0:1}$ are indistinguishable, and $\mathcal{A}$ does not cheat in $\hat{H}_0^{0:1}$.

Let us stress that the only difference between $\hat{H}_0^{0:0}$ and $\hat{H}_0^{0:1}$ lies in the *first* RobCom in Stage 0-1b of session 1 (if $\mathcal{A}$ corrupts S in session 1), **or** in Stage 0-2b of session 1 (if $\mathcal{A}$ corrupts R in session 1).

Indistinguishability. First note that, in session 1, if no party is corrupted, $\hat{H}_0^{0:0}$ and $\hat{H}_0^{0:1}$ are identical. So the statement holds trivially.

When one party is corrupted (S or R), we prove the indistinguishability based on the robust extractability of RobCom.

The argument is actually identical to the proof of the extractability for Protocol 3.4.1 (Section 3.4.1.2), with the only difference that there are other sessions running besides the RobCom under our consideration. To deal with this, we note that Step 5d in Protocol 3.4.1 is instantiated with the commit-and-proof scheme shown in Protocol 3.3.1, whose **Proof Phase** gives us ℓ-robustness (as we proved in Theorem 3.3.2). In the design of our OT, we set the robustness to be $\ell(n) = m \cdot \nu_{\text{OT}}(n)$, which is large enough to encompass all the messages from the remaining part of the execution. With this modification, the same sequence of hybrids for the extractability of Protocol 3.4.1 also works for proving indistinguishability among $\hat{H}_0^{0:0}$ to $\hat{H}_0^{0:1}$. More specifically, we only need to modify the switch from H_0 to H_1 (where we switch from the real prover to the robust-ZK simulator in Step 5d) in Section 3.4.1.2, by relying additional on the ℓ-robustness. For completeness, we provide the full proof for this switch in the following.

Assume for contradiction that the indistinguishability does not hold due to the switch from the real prover to the robust-ZK simulator (from H_0 to H_1 as mentioned above). We can then construct a robust-ZK verifier V^* and a machine B, who interact with either a prover or the simulator $S_{\mathsf{ZK}}(\mathsf{code}[\mathcal{A}])$ and violate the robust-ZK property, as follows.

Machine B: incorporates all honest parties, including the honest party for session 1. B performs all steps honestly for each party except the corresponding Step 5d in session 1; the messages of this phase are expected to come from an external machine (either the prover or the simulator).

The messages of B are sent to the network which delivers them appropriately to the cheating verifier (specified below). We note that, by definition of robust-ZK B receives a copy of all messages that V^* receives.

Verifier V^*: this algorithm is just the adversary $\mathcal{A}$, with the understanding that all messages that do not belong to the corresponding Step 5d in session 1 are viewed as external messages sent to (or received from) machine B.

Observe that B is polynomial time, and since it receives a copy of all messages sent to V^* (by the prover or the simulator), it can indeed function correctly even though it runs the simulator algorithm for the ZK-proof stage in all sessions different from session 1. (This is not necessary in the switch from $\hat{H}_0^0$ to $\hat{H}_0^1$, as the only session 1 contains simulated RobCom. But it will be important in later hybrids as more sessions contain simulated (Step 5d of) RobCom.)

Notice that if V^* interacts with honest prover of the robust-ZK, then this experiment is identical to the aforementioned H_0. On the other hand, if it interacts with S_{ZK} then the experiment is identical to the aforementioned H_1;[12] furthermore, since S_{ZK} receives a copy of all messages that V^* receives from B, it can indeed run in polynomial time.

It follows that if the output of hybrids are not indistinguishable, we violate the robust-ZK property.

Remark 3.7.8 (On the Bound $\ell(n)$)**.** *The above argument relies on the assumption that the size of messages coming from B to V^* is bounded by $\ell(n)$ (the definition of $\ell(n)$-robust ZK). As a vigilant reader may have realized already, this is not quite true for our definition of V^*. More specifically, consider all the messages coming from B to V^*. There must be one* **long message** *for each session where V^* plays the role of* RobCom *sender (note that the other* **long message** *in the same session comes from V^* to B, so we do not need to worry)*[13]*. Recall that we set $\ell(n) = m \cdot \nu_{\mathrm{OT}}(n)$. This is enough to capture all the short messages, but not these long messages (from B to V^*). This can be resolved in the following way. Note that the long messages coming from B to V^* are nothing more than random strings. Thus, we can extend V^* to incorporate the subroutine in B that samples these random strings. Let us denote this extended adversary as $\widetilde{V}^*$. We modify the above argument by passing the code of $\widetilde{V}^*$ to the ZK simulator. Now everything works as the size of messages flowing from B to $\widetilde{V}^*$ is bounded by $\ell(n)$.*

Invariant Condition. We next show that in $\hat{H}_0^{0:1}$, $\mathcal{A}$ does not cheat. Assume for contradiction that $\mathcal{A}$ cheats in some session $i^*(n) \in [m]$ with non-negligible probability. Note that the only difference between $\hat{H}_0^{0:0}$ and $\hat{H}_0^{0:1}$ is that the adversary in session 1 sees a real proof in RobCom of $\hat{H}_0^{0:0}$, but a simulated one (from the straight-line extractor) in RobCom of $\hat{H}_0^{0:1}$. Then, by expecting this stage *in session 1* from an external prover or simulator (RobCom extractor), we can break the robust non-malleability of NMCom *in session* $i^*(n)$. We elaborate on this argument in the following.

> The man-in-the-middle adversary $\mathcal{A}_{\mathsf{NMCom}}$ internally executes $\hat{H}_0^{0:0}$. If S (resp. R) is corrupted in session 1, then in Stage 0-1b (resp. Stage 0-2b) of session 1, $\mathcal{A}_{\mathsf{NMCom}}$ forward the message between the external party, which is either the honest RobCom receiver or the straihgt-line simulator. Also, in session $i^*(n)$, $\mathcal{A}_{\mathsf{NMCom}}$ forwards the NMCom commitments from $\mathcal{A}$ to the external receiver (specifically, the NMCom commitments in Stage 4 if R is corrupted in session $i^*(n)$, and in Stage 7 if S is corrupted in session $i^*(n)$). After the execution of $\hat{H}_0^{0:0}$ finishes, $\mathcal{A}_{\mathsf{NMCom}}$ outputs its view.
>
> The distinguisher $\mathcal{D}_{\mathsf{NMCom}}$ takes as input the view of $\mathcal{A}_{\mathsf{NMCom}}$ and the values

[12] We stress that the H_0 and H_1 here are the ones as defined in Section 3.4.1.2, but in our current context of the concurrent OT setting. They should not be confused with other hybrids defined in this section

[13] More accurately, using our definition for B and V^*, in any session, if S is corrupted, then V^* will incorporate S and the **receiver's long massages** flows from B to V^*; if R is corrupted, the **sender's long massages** flows from B to V^*.

committed by $\mathcal{A}_{\mathsf{NMCom}}$ (which are equal to the values committed to by $\mathcal{A}$ in session $i^*(n)$ in the internally executed experiment). $\mathcal{D}_{\mathsf{NMCom}}$ then outputs 1 if and only if $\mathcal{A}$ cheated in session $i^*(n)$. (Notice that given the committed values of the NMCom commitments, $\mathcal{D}_{\mathsf{NMCom}}$ can check whether $\mathcal{A}$ cheated or not in polynomial time.)

When the external party mentioned above is the honest RobCom receiver, the view of $\mathcal{A}$ is identical to that in $\hat{H}_0^{0:0}$; whereas when the external party mentioned above is the extractor for RobCom, the view of $\mathcal{A}$ is identical to that in $\hat{H}_0^{0:1}$. Hence, from the assumption that $\mathcal{A}$ cheats in session $i^*(n)$ with negligible probability in $\hat{H}_0^{0:0}$ but with non-negligible probability in $\hat{H}_0^{0:1}$, $\mathcal{A}_{\mathsf{NMCom}}$ breaks the robust non-malleability of NMCom.

This finishes the proof for Claim 3.7.1. □

This finishes the proof for Lemma 3.7.2. □

From here onwards, the proof of indistinguishability of hybrids is very similar to the proofs in [GKP18] (with minor notational changes) except that we do not require brute-force extraction of inputs; instead they are accessed directly from table $\mathcal{T}$. We provide the proofs here for completeness.

Lemma 3.7.3. *Assume that in $H_{k-1:7}$ ($k \in [4m]$), $\mathcal{A}$ does not cheat in sessions $s(k), \ldots, s(4m)$ except with negligible probability. Then,*

- *$H_{k-1:7}$ and $H_{k:1}$ are indistinguishable, and*
- *in $H_{k:1}$, $\mathcal{A}$ does not cheat in sessions $s(k), \ldots, s(4m)$ except with negligible probability.*

Proof. This proof for this lemma is identical to that in [GKP18], except that the hybrid uses the table $\mathcal{T}$ to get the extracted values it needs, instead of extracting by brute force. We present the full proof here for completeness.

We prove the lemma by using a hybrid argument. Specifically, we consider the following intermediate hybrid $H'_{k-1:7}$.

Hybrid $H'_{k-1:7}$. $H'_{k-1:7}$ is the same as $H_{k-1:7}$ except that in session $s(k)$, if S is corrupted and SM_k is $\mathsf{first\ special\ message}$,

- the committed subset Γ_S is extracted by querying $\mathcal{T}$ in Stage 1-1, and
- the value committed to in the i-th ExtCom commitment in Stage 4 is switched to an all-zero string for every $i \notin \Gamma_S$.

Claim 3.7.2. *Assume that in $H_{k-1:7}$, $\mathcal{A}$ does not cheat in sessions $s(k), \ldots, s(4m)$ except with negligible probability. Then,*

- *$H_{k-1:7}$ and $H'_{k-1:7}$ are indistinguishable, and*
- *in $H'_{k-1:7}$, $\mathcal{A}$ does not cheat in sessions $s(k), \ldots, s(4m)$ except with negligible probability.*

Proof. We first show the indistinguishability between $H_{k-1:7}$ and $H'_{k-1:7}$. Assume for contradiction that $H_{k-1:7}$ and $H'_{k-1:7}$ are distinguishable. From an average argument, we can fix the execution of the experiment up until SM_k (inclusive) in such a way that even after being fixed, $H_{k-1:7}$ and $H'_{k-1:7}$ are still distinguishable. By considering the transcript (including the inputs and randomness of all the parties) up until SM_k and the table $\mathcal{T}$ as non-uniform advice, we can break the hiding property of ExtCom as follows.

> The adversary $\mathcal{A}_{\mathsf{ExtCom}}$ internally executes $H_{k-1:7}$ from SM_k using the non-uniform advice. In Stage 4 of session $s(k)$, $\mathcal{A}_{\mathsf{ExtCom}}$ sends $(a_i^R, d_i^R, e_i^R)_{i \notin \Gamma_S}$ and $(0,0,0)_{i \notin \Gamma_S}$ to the external committer, receives back ExtCom commitments (in which either $(a_i^R, d_i^R, e_i^R)_{i \notin \Gamma_S}$ or $(0,0,0)_{i \notin \Gamma_S}$ are committed to), and feeds them into $H_{k-1:7}$. After the execution of $H_{k-1:7}$ finishes, $\mathcal{A}_{\mathsf{ExtCom}}$ outputs whatever $\mathcal{Z}$ outputs in the experiment.

When $\mathcal{A}_{\mathsf{ExtCom}}$ receives commitments to $(a_i^R, d_i^R, e_i^R)_{i \notin \Gamma_S}$, the internally executed experiment is identical with $H_{k-1:7}$, whereas when $\mathcal{A}_{\mathsf{ExtCom}}$ receives a commitments to $(0,0,0)_{i \notin \Gamma_S}$, the internally executed experiment is identical with $H'_{k-1:7}$. Hence, from the assumption that $H_{k-1:7}$ and $H'_{k-1:7}$ are distinguishable (even after being fixed up until SM_k), $\mathcal{A}_{\mathsf{ExtCom}}$ distinguishes ExtCom commitments.

We next show that in $H'_{k-1:7}$, $\mathcal{A}$ does not cheat in sessions $s(k), \ldots, s(4m)$. Assume for contradiction that in $H'_{k-1:7}$, $\mathcal{A}$ cheats in one of those sessions, say, session $s(j)$, with non-negligible probability. Then, from an average argument, we can fix the execution of the experiment up until SM_k (inclusive) in such a way that even after being fixed, $\mathcal{A}$ cheats in session $s(j)$ only with negligible probability in $H_{k-1:7}$ but with non-negligible probability in $H'_{k-1:7}$.

Then, by considering the transcript up until SM_k and the table $\mathcal{T}$ as non-uniform advice, we can break the robust non-malleability of NMCom as follows.

> The man-in-the-middle adversary $\mathcal{A}_{\mathsf{NMCom}}$ internally executes $H_{k-1:7}$ from SM_k using the non-uniform advice. In Stage 4 of session $s(k)$, $\mathcal{A}_{\mathsf{NMCom}}$ sends $(a_i^R, d_i^R, e_i^R)_{i \notin \Gamma_S}$ and $(0,0,0)_{i \notin \Gamma_S}$ to the external committer, receives back ExtCom commitments (in which either $(a_i^R, d_i^R, e_i^R)_{i \notin \Gamma_S}$ or $(0,0,0)_{i \notin \Gamma_S}$ are committed to), and feeds them into $H_{k-1:7}$. Also, in session $s(j)$, $\mathcal{A}_{\mathsf{NMCom}}$ forwards the NMCom commitments from $\mathcal{A}$ to the external receiver (specifically, the NMCom commitments in Stage 4 if R is corrupted and in Stage 7 if S is corrupted). After the execution of $H_{k-1:7}$ finishes, $\mathcal{A}_{\mathsf{NMCom}}$ outputs its view.
>
> The distinguisher $\mathcal{D}_{\mathsf{NMCom}}$ takes as input the view of $\mathcal{A}_{\mathsf{NMCom}}$ and the values committed by $\mathcal{A}_{\mathsf{NMCom}}$ (which are equal to the values committed to by $\mathcal{A}$ in session $s(j)$ in the internally executed experiment). $\mathcal{D}_{\mathsf{NMCom}}$ then outputs 1 if and only if $\mathcal{A}$ cheated in session $s(j)$. (Notice that given the committed values of the NMCom commitments, $\mathcal{D}_{\mathsf{NMCom}}$ can check whether $\mathcal{A}$ cheated or not in polynomial time.)

When $\mathcal{A}_{\mathsf{NMCom}}$ receives commitments to $(a_i^R, d_i^R, e_i^R)_{i \notin \Gamma_S}$, the internally executed experiment is identical with $H_{k-1:7}$, whereas when $\mathcal{A}_{\mathsf{NMCom}}$ receives a commitments to $(0,0,0)_{i \notin \Gamma_S}$, the internally executed experiment is identical with $H'_{k-1:7}$. Hence, from the assumption that $\mathcal{A}$ cheats in session $s(j)$ with negligible probability in $H_{k-1:7}$ but with non-negligible probability in $H'_{k-1:7}$, $\mathcal{A}_{\mathsf{NMCom}}$ breaks the robust non-malleability of NMCom.

This completes the proof of Claim 3.7.2. $\square$

Claim 3.7.3. *Assume that in $H'_{k-1:7}$, $\mathcal{A}$ does not cheat in sessions $s(k), \ldots, s(4m)$ except with negligible probability. Then,*

– *$H'_{k-1:7}$ and $H_{k:1}$ are indistinguishable, and*

– *in $H_{k:1}$, $\mathcal{A}$ does not cheat in sessions $s(k), \ldots, s(4m)$ except with negligible probability.*

This claim can be proven very similarly to Claim 3.7.2 (the only difference is that we use the hiding property of NMCom rather than that of ExtCom in the first part, and use the non-malleability of NMCom rather than its robust non-malleability in the second part). We thus omit the proof.

This completes the proof of Lemma 3.7.3. □

Lemma 3.7.4. *Assume that in $H_{k:1}$ ($k \in [4m]$), $\mathcal{A}$ does not cheat in sessions $s(k), \ldots, s(4m)$ except with negligible probability. Then,*

– *$H_{k:1}$ and $H_{k:2}$ are indistinguishable, and*

– *in $H_{k:2}$, $\mathcal{A}$ does not cheat in sessions $s(k), \ldots, s(4m)$ except with negligible probability.*

Recall that hybrids $H_{k:1}, H_{k:2}$ differ only in the input and the randomness that are used in some of the mS-OTs in Stage 3, where those that are derived from the outcomes of the coin tossing is used in $H_{k:1}$ and random inputs and true randomness are used in $H_{k:2}$. Intuitively, we prove this lemma by using the security of the Stage-2-2 coin tossing (which is guaranteed by the hiding property of $\mathsf{RobCom}(\psi_i^R)$'s) because it guarantees that the outcome of the coin tossing is pseudorandom. The proof is quite similar to the proof of Claim 3.7.2 (we use the hiding of $\mathsf{RobCom}(\psi_i^R)$'s rather than that of ExtCom), and given in Section 3.9.2.

Lemma 3.7.5. *Assume that in $H_{k:2}$ ($k \in [4m]$), $\mathcal{A}$ does not cheat in sessions $s(k), \ldots, s(4m)$ except with negligible probability. Then,*

– *$H_{k:2}$ and $H_{k:3}$ are indistinguishable, and*

– *in $H_{k:3}$, $\mathcal{A}$ does not cheat in sessions $s(k), \ldots, s(4m)$ except with negligible probability.*

Proof. Recall that $H_{k:2}$ and $H_{k:3}$ differ only in that in session $s(k)$ of $H_{k:3}$, if S is corrupted and SM_k is $\mathsf{third\ special\ message}$, either R outputs $\mathsf{Value}(\boldsymbol{\rho}_u^{\mathsf{ext}}, \Gamma_R \cap \Delta)$ rather than $\mathsf{Value}(\widetilde{\boldsymbol{\rho}}, \Gamma_R \cap \Delta)$ or the hybrid is aborted.

For proving the lemma, it suffices to show that in session $s(k)$ of $H_{k:3}$,

1. the hybrid is not aborted except with negligible probability, and
2. if the hybrid is not aborted we have $\mathsf{Value}(\boldsymbol{\rho}_u^{\mathsf{ext}}, \Gamma_R \cap \Delta) = \mathsf{Value}(\widetilde{\boldsymbol{\rho}}, \Gamma_R \cap \Delta)$

To see that showing these two is indeed sufficient for proving the lemma, observe the following. First, these two imply that in session $s(k)$ of $H_{k:3}$, the probability that the hybrid is aborted or we have $\mathsf{Value}(\boldsymbol{\rho}_u^{\mathsf{ext}}, \Gamma_R \cap \Delta) = \mathsf{Value}(\widetilde{\boldsymbol{\rho}}, \Gamma_R \cap \Delta)$ is negligible, so $H_{k:2}$ and $H_{k:3}$ are statistically close. Second, since $H_{k:2}$ and $H_{k:3}$ proceed identically until the end of session $s(k)$, and

1. if the experiment is not aborted in session $s(k)$, $H_{k:2}$ and $H_{k:3}$ continue to proceed identically after the end of session $s(k)$, and

2. if the hybrid is aborted in session $s(k)$, $\mathcal{A}$ clearly does not cheat in any session after the end of session $s(k)$

the probability that $\mathcal{A}$ cheats in sessions $s(k), ..., s(4m)$ is not increased in $H_{k:3}$.

Now, we first show that in session $s(k)$ of $H_{k:3}$, the hybrid is not aborted except with negligible probability. Since $H_{k:2}$ and $H_{k:3}$ proceed identically until the end of session $s(k)$, we have that in $H_{k:3}$, $\mathcal{A}$ does not cheat in session $s(k)$ except with negligible probability. So, it suffices to show that when session $s(k)$ is accepting and $\mathcal{A}$ does not cheat in session $s(k)$, the hybrid is not aborted in session $s(k)$. Recall that if $\mathcal{A}$ does not cheat in an accepting session (in which S is corrupted), we have the following.

1. Let $(\hat{a}_1^S, \hat{d}_1^S), \dots (\hat{a}_{11n}^S, \hat{d}_{11n}^S)$ be the values that are committed in NMCom in Stage 7. Let $I_{\text{bad}} \subset [11n]$ be the set such that $i \in I_{\text{bad}}$ if and only if
   (a) $(\hat{a}_i^S, \hat{d}_i^S)$ is not valid in terms of the check in Stage 8-3b, i.e. $\hat{d}_i^S$ is not a valid decommitment of ψ_i^S w.r.t. Stage-0-1 RobCom, or $\hat{a}_i^S \oplus \psi_i^S$ does not equal the value z_i^S it received in Stage-2-1; **or**
   (b) S does not execute the i-th mS-OT in Stage 3 honestly using $\hat{s}_{i,0} \,\|\, \hat{s}_{i,1} \,\|\, \hat{\tau}_i^S$ as the input and randomness, where $\hat{s}_{i,0} \,\|\, \hat{s}_{i,1} \,\|\, \hat{\tau}_i^S$ is obtained from $\hat{r}_i^S = \hat{a}_i^S \oplus b_i^S$.

   Then, it holds that $|I_{\text{bad}}| < 0.1n$.
2. For each $b \in \{0,1\}$, define $\boldsymbol{\rho}_b^{\mathsf{nm}} = (\rho_{b,i}^{\mathsf{nm}})_{i \in \Delta}$ as follows: $\rho_{b,i}^{\mathsf{nm}} \overset{\text{def}}{=} \beta_{b,i} \oplus \hat{s}_{i,b \oplus \alpha_i}$ if $i \notin I_{\text{bad}}$ and $\rho_{b,i}^{\mathsf{nm}} \overset{\text{def}}{=} \perp$ otherwise. Then, for each $b \in \{0,1\}$, ρ_b^{nm} is either 0.9-close to a valid codeword $\boldsymbol{w} = (w_i)_{i \in \Delta}$ that satisfies $w_i = \rho_{b,i}^{\mathsf{nm}}$ for every $i \in \Gamma_R$ or 0.15-far from any such valid codeword.

We show that the above two imply that the hybrid is not aborted at the end of the session, i.e. that both of the following hold for each $b \in \{0,1\}$.

1. $|\{i \in \Delta \text{ s.t. } \rho_{b,i}^{\mathsf{ext}} \neq \perp\}| < 0.1n$.
2. $\boldsymbol{\rho}_b^{\mathsf{ext}}$ is either 0.9-close to a valid codeword $\boldsymbol{w} = (w_i)_{i \in \Delta}$ that satisfies $w_i = \rho_{b,i}^{\mathsf{ext}}$ for every $i \in \Gamma_R$ or 0.14-far from any such valid codeword.

Fix any $b \in \{0,1\}$. First, we notice that we can obtain $|\{i \in \Delta \text{ s.t. } \rho_{b,i}^{\mathsf{ext}} \neq \perp\}| < 0.1n$ from $|I_{\text{bad}}| < 0.1n$ since we have $\{i \in \Delta \text{ s.t. } \rho_{b,i}^{\mathsf{ext}} \neq \perp\} \subseteq I_{\text{bad}}$ from the definition of $\boldsymbol{\rho}_b^{\mathsf{ext}}$ and I_{bad}. Next, we observe that $\boldsymbol{\rho}_b^{\mathsf{ext}}$ is either 0.9-close to a valid codeword $\boldsymbol{w} = (w_i)_{i \in \Delta}$ that satisfies $w_i = \rho_{b,i}^{\mathsf{ext}}$ for every $i \in \Gamma_R$ or 0.14-far from any such valid codeword. From the assumption that $\mathcal{A}$ does not cheat, it suffices to consider the following two cases.

Case 1. $\boldsymbol{\rho}_b^{\mathsf{nm}}$ is 0.9-close to a valid codeword $\boldsymbol{w} = (w_i)_{i \in \Delta}$ that satisfies $w_i = \rho_{b,i}^{\mathsf{nm}}$ for every $i \in \Gamma_R \cap \Delta$: In this case, $\boldsymbol{\rho}_b^{\mathsf{ext}}$ is 0.9-close to $\boldsymbol{w}$, and $w_i = \rho_{b,i}^{\mathsf{ext}}$ holds for every $i \in \Gamma_R$. This is because for every i such that $\rho_{b,i}^{\mathsf{nm}} = w_i$, we have $\rho_{b,i}^{\mathsf{nm}} \neq \perp$ and thus we have $\rho_{b,i}^{\mathsf{nm}} = \rho_{b,i}^{\mathsf{ext}}$ from the definition of $\boldsymbol{\rho}_b^{\mathsf{nm}}$.

Case 2. $\boldsymbol{\rho}_b^{\mathsf{nm}}$ is 0.15-far from any valid codeword $\boldsymbol{w} = (w_i)_{i \in \Delta}$ that satisfies $w_i = \rho_{b,i}^{\mathsf{nm}}$ for every $i \in \Gamma_R \cap \Delta$: In this case, $\boldsymbol{\rho}_b^{\mathsf{ext}}$ is 0.14-far from any valid codeword $\boldsymbol{w'}$ that satisfies $w'_i = \rho_{b,i}^{\mathsf{ext}}$ for every $i \in \Gamma_R \cap \Delta$. This can be seen by observing the following:

(1) for every $i \in \Gamma_R \cap \Delta$, we have $i \notin I_{\text{bad}}$ (this is because the session is accepting) and hence $\rho^{\mathsf{ext}}_{b,i} = \rho^{\mathsf{nm}}_{b,i}$; (2) therefore, for any valid codeword $\boldsymbol{w'}$ that satisfies $w'_i = \rho^{\mathsf{ext}}_{b,i}$ for every $i \in \Gamma_R \cap \Delta$, we have that $\boldsymbol{w'}$ also satisfies $w'_i = \rho^{\mathsf{nm}}_{b,i}$ for every $i \in \Gamma_R \cap \Delta$; (3) then, from the assumption of this case, $\boldsymbol{\rho}^{\mathsf{nm}}_b$ is 0.15-far from $\boldsymbol{w'}$; (4) now, since $\boldsymbol{\rho}^{\mathsf{nm}}_b$ and $\boldsymbol{\rho}^{\mathsf{ext}}_b$ are 0.99-close (this follows from $|I_{\text{bad}}| < 0.1n$), $\boldsymbol{\rho}^{\mathsf{ext}}_b$ is 0.14-far from $\boldsymbol{w'}$.

We therefore conclude that when session $s(k)$ is accepting and $\mathcal{A}$ does not cheat in session $s(k)$, the hybrid is not aborted in session $s(k)$.

Next, we show that in session $s(k)$ of $H_{k:3}$ if the hybrid is not aborted, we have $\mathsf{Value}(\boldsymbol{\rho}^{\mathsf{ext}}_u, \Gamma_R \cap \Delta) = \mathsf{Value}(\widetilde{\boldsymbol{\rho}}, \Gamma_R \cap \Delta)$. To show this, it suffices to show the following two claims.

Claim 3.7.4. *For any $\boldsymbol{x} = (x_i)_{i\in\Delta}, \boldsymbol{y} = (y_i)_{i\in\Delta}$ and a set Θ, we have $\mathsf{Value}(\boldsymbol{x}, \Theta) = \mathsf{Value}(\boldsymbol{y}, \Theta)$ if the following conditions hold.*

1. *$\boldsymbol{x}$ and $\boldsymbol{y}$ are 0.99-close, and $x_i = y_i$ holds for every $i \in \Theta$.*
2. *If $x_i \neq \perp$, then $x_i = y_i$.*
3. *$\boldsymbol{x}$ is either 0.9-close to a valid codeword $\boldsymbol{w} = (w_i)_{i\in\Delta}$ that satisfies $w_i = x_i$ for every $i \in \Theta$ or 0.14-far from any such valid codeword.*

Claim 3.7.5. *In session $s(k)$ of $H_{k:3}$, if the sender S is corrupted, the session is accepting, and the session is not aborted the following hold.*

1. *$\boldsymbol{\rho}^{\mathsf{ext}}_u$ and $\widetilde{\boldsymbol{\rho}}$ are 0.99-close, and $\rho^{\mathsf{ext}}_{u,i} = \tilde{\rho}_i$ holds for every $i \in \Gamma_R \cap \Delta$.*
2. *If $\rho^{\mathsf{ext}}_{u,i} \neq \perp$, then $\rho^{\mathsf{ext}}_{u,i} = \tilde{\rho}_i$.*
3. *$\boldsymbol{\rho}^{\mathsf{ext}}_u$ is either 0.9-close to a valid codeword $\boldsymbol{w} = (w_i)_{i\in\Delta}$ that satisfies $w_i = \rho^{\mathsf{ext}}_{u,i}$ for every $i \in \Gamma_R \cap \Delta$ or 0.14-far from any such valid codeword.*

We prove each of the claims below.

Proof of Claim 3.7.4. We consider the following two cases.

Case 1. $\boldsymbol{x}$ is 0.9-close to a valid codeword $\boldsymbol{w} = (w_i)_{i\in\Delta}$ that satisfies $w_i = x_i$ for every $i \in \Theta$: First, we observe that $\boldsymbol{y}$ is 0.9-close to $\boldsymbol{w}$. Since $\boldsymbol{w}$ is a valid codeword, we have $w_i \neq \perp$ for every $i \in \Delta$; thus, for every i such that $x_i = w_i$, we have $x_i \neq \perp$. Recall that from the assumed conditions, for every i such that $x_i \neq \perp$, we have $x_i = y_i$. Therefore, for every i such that $x_i = w_i$, we have $y_i = w_i$, which implies that $\boldsymbol{y}$ is 0.9-close to $\boldsymbol{w}$.

Next, we observe that $\boldsymbol{w}$ satisfies $w_i = y_i$ for every $i \in \Theta$. From the assumed conditions, we have $x_i = y_i$ for every $i \in \Theta$. Also, from the condition of this case, $\boldsymbol{w}$ satisfies $w_i = x_i$ for every $i \in \Theta$. From these two, we have that $\boldsymbol{w}$ satisfies $w_i = y_i$ for every $i \in \Theta$.

Now, from the definition of $\mathsf{Value}(\cdot,\cdot)$, we have $\mathsf{Value}(\boldsymbol{x}, \Theta) = \mathsf{Value}(\boldsymbol{y}, \Theta) = \mathsf{Decode}(\boldsymbol{w})$.

Case 2. $\boldsymbol{x}$ is 0.14-far from any valid codeword $\boldsymbol{w} = (w_i)_{i\in\Delta}$ that satisfies $w_i = x_i$ for every $i \in \Theta$: For any valid codeword $\boldsymbol{w'} = (w'_i)_{i\in\Delta}$ that satisfies $w'_i = y_i$ for every $i \in \Theta$, we observe that $\boldsymbol{y}$ is 0.1-far from $\boldsymbol{w'}$. Since we assume that $x_i = y_i$ holds for every $i \in \Theta$, we have $w'_i = x_i$ for every $i \in \Theta$. Therefore, from the assumption of this case, $\boldsymbol{x}$ is 0.14-far from $\boldsymbol{w'}$. Now, since we assume that $\boldsymbol{x}$ and $\boldsymbol{y}$ are 0.99-close, $\boldsymbol{y}$ is 0.1-far from $\boldsymbol{w'}$.

Now, from the definition of $\mathsf{Value}(\cdot,\cdot)$, we conclude that:

$$\mathsf{Value}(\boldsymbol{x},\Theta) = \mathsf{Value}(\boldsymbol{y},\Theta) = \perp.$$

Notice that from the assumed conditions, either Case 1 or Case 2 is true. This concludes the proof of Claim 3.7.4. □

Proof of Claim 3.7.5. Recall that if the hybrid is not aborted in an accepting session in which S is corrupted, we have the following for each $b \in \{0,1\}$ in that session.

1. $|\{i \in \Delta \text{ s.t. } \rho^{\mathsf{ext}}_{b,i} \neq \perp\}| < 0.1n$.
2. $\boldsymbol{\rho}^{\mathsf{ext}}_b$ is either 0.9-close to a valid codeword $\boldsymbol{w} = (w_i)_{i\in\Delta}$ that satisfies $w_i = \rho^{\mathsf{ext}}_{b,i}$ for every $i \in \Gamma_R$ or 0.14-far from any such valid codeword.

Thus, it suffices to show that the above two imply the first condition in the claim statement.

First, we show that $\boldsymbol{\rho}^{\mathsf{ext}}_u$ and $\widetilde{\boldsymbol{\rho}}$ are 0.99-close and that $\rho^{\mathsf{ext}}_{u,i} = \tilde{\rho}_i$ holds for every $i \in \Gamma_R \cap \Delta$. From the definition of $\boldsymbol{\rho}^{\mathsf{ext}}_u$, we have $\rho^{\mathsf{ext}}_{u,i} = \tilde{\rho}_i$ for every i such that $\rho^{\mathsf{ext}}_{b,i} \neq \perp$ (this is because for every such i, $\mathcal{A}$ executed the i-th mS-OT in Stage 3 honestly using the coin obtained in Stage 2-1, which implies that the value $\widetilde{s}_i$ that was obtained from the i-th mS-OT is equal to the value s_{i,c_i} that was obtained by extracting the coin in Stage 2-1 by brute-force). Then, since $|\{i \in \Delta \text{ s.t. } \rho^{\mathsf{ext}}_{b,i} \neq \perp\}| < 0.1n$ and $\{i \in \Delta \text{ s.t. } \rho^{\mathsf{ext}}_{b,i} \neq \perp\} \cap \Gamma_R = \emptyset$ (the latter holds since the session would be rejected otherwise), we have that $\boldsymbol{\rho}^{\mathsf{ext}}_u$ and $\widetilde{\boldsymbol{\rho}}$ are 0.99-close and that $\rho^{\mathsf{ext}}_{u,i} = \tilde{\rho}_i$ holds for every $i \in \Gamma_R \cap \Delta$.

Next, we show that if $\rho^{\mathsf{ext}}_{u,i} \neq \perp$ then $\rho^{\mathsf{ext}}_{u,i} = \tilde{\rho}_i$. From the definition of $\boldsymbol{\rho}^{\mathsf{ext}}_u$, if $\rho^{\mathsf{ext}}_{u,i} \neq \perp$, $\mathcal{A}$ executed the i-th mS-OT in Stage 3 honestly using the coin obtained in Stage 2-1, so we have $\rho^{\mathsf{ext}}_{u,i} = \tilde{\rho}_i$ from the argument same as above.

This concludes the proof of Claim 3.7.5. □

This concludes the proof of Lemma 3.7.5. □

Lemma 3.7.6. *Assume that in $H_{k:3}$ ($k \in [4m]$), $\mathcal{A}$ does not cheat in sessions $s(k),\ldots,s(4m)$ except with negligible probability. Then,*

– *$H_{k:3}$ and $H_{k:4}$ are indistinguishable, and*

– *in $H_{k:4}$, $\mathcal{A}$ does not cheat in sessions $s(k),\ldots,s(4m)$ except with negligible probability.*

Recall that $H_{k:3}$ and $H_{k:4}$ differ only in that in session $s(k)$ of $H_{k:4}$, if S is corrupted and SM_k is $\mathsf{third\ special\ message}$, α_i is a random bit rather than $\alpha_i = u \oplus c_i$ for every $i \in \Delta$ in Stage 6-1. Intuitively, we can prove this lemma by using the security of $\mathsf{mS\text{-}OT}$: For every $i \notin \Gamma_S$, the choice bit c_i of the i-th mS-OT in Stage 3 is hidden from $\mathcal{A}$ and hence $\alpha_i = u \oplus c_i$ in $H_{k:3}$ is indistinguishable from a random bit. Formally, we prove this Lemma in the same way as we do for Claim 3.7.2 (we use the security of $\mathsf{mS\text{-}OT}$ rather than the hiding of ExtCom); the proof is given in Section 3.9.3.

The next lemma is the counterpart of Lemma 3.7.3 when R is corrupted.

Lemma 3.7.7. *Assume that in $H_{k:4}$ ($k \in [4m]$), $\mathcal{A}$ does not cheat in sessions $s(k), \ldots, s(4m)$ except with negligible probability. Then,*

- *$H_{k:4}$ and $H_{k:5}$ are indistinguishable, and*
- *in $H_{k:5}$, $\mathcal{A}$ does not cheat in sessions $s(k), \ldots, s(4m)$ except with negligible probability.*

Note that hybrids $H_{k:4}$ and $H_{k:5}$ differ only in the values committed to in NMCom and ExtCom for the indices outside of Γ_R, in session $s(k)$, when R is corrupted. This lemma can be proven identically with Lemma 3.7.3. For completeness, we give a formal proof in Section 3.9.4.

Lemma 3.7.8. *Assume that in $H_{k:5}$ ($k \in [4m]$), $\mathcal{A}$ does not cheat in sessions $s(k), \ldots, s(4m)$ except with negligible probability. Then,*

- *$H_{k:5}$ and $H_{k:6}$ are indistinguishable, and*
- *in $H_{k:6}$, $\mathcal{A}$ does not cheat in sessions $s(k), \ldots, s(4m)$ except with negligible probability.*

Since hybrids $H_{k:5}, H_{k:6}$ differ only in the inputs and the randomness that are used in some of the mS-OTs in Stage 3, this lemma can be proven identically with Lemma 3.7.4 (which in turn can be proven quite similarly to Lemma 3.7.3). For completeness, we give a formal proof in Section 3.9.5.

Lemma 3.7.9. *Assume that in $H_{k:6}$ ($k \in [4m]$), $\mathcal{A}$ does not cheat in sessions $s(k), \ldots, s(4m)$ except with negligible probability. Then,*

- *$H_{k:6}$ and $H_{k:7}$ are indistinguishable, and*
- *in $H_{k:7}$, $\mathcal{A}$ does not cheat in sessions $s(k), \ldots, s(4m)$ except with negligible probability.*

Proof. We prove the lemma by considering the following intermediate hybrids $H'_{k:6}$, $H''_{k:6}$, and $H'''_{k:6}$.

Hybrid $H'_{k:6}$. $H'_{k:6}$ is the same as $H_{k:6}$ except that in session $s(k)$, if R is corrupted and SM_k is $\mathsf{fourth\ special\ message}$, the following modifications are made.

1. As in $H_{k:7}$, the committed strings $\boldsymbol{a}^R = (a_1^R, \ldots, a_{11n}^R)$ are extracted by querying table $\mathcal{T}$, $\boldsymbol{r}^R = (r_1^R, \ldots, r_{11n}^R)$ is defined by $r_i^R \overset{\text{def}}{=} a_i^R \oplus b_i^R$ for each $i \in [11n]$, and r_i^R is parsed as $c_i \| \tau_i^R$ for each $i \in [11n]$. Also, I_0, I_1, and u^* are defined as in $H_{k:7}$.
2. In Stage 6, $\beta_{b,i}$ is a random bit rather than $\beta_{b,i} = \rho_{b,i} \oplus s_{i,b\oplus\alpha_i}$ for every $b \in \{0,1\}$ and $i \in \Delta \setminus I_b$. (Recall that, roughly, $I_b \subset \Delta$ is the set of indices on which $\mathcal{A}$ could have obtained $s_{i,b\oplus\alpha_i}$.)

Hybrid $H''_{k:6}$. $H''_{k:6}$ is the same as $H'_{k:6}$ except that in session $s(k)$, if R is corrupted and SM_k is $\mathsf{fourth\ special\ message}$, the following modification is made.

1. The execution of the hybrid is aborted if both of $|I_0| \geq 6n+1$ and $|I_1| \geq 6n+1$ holds.
2. In Stage 6, $\boldsymbol{\rho}_{1-u^*} = \{\rho_{1-u^*,i}\}_{i\in\Delta}$ is a secret sharing of a random bit rather than that of v_{1-u^*}.

Hybrid $H'''_{k:6}$. $H'''_{k:6}$ is the same as $H''_{k:6}$ except that in session $s(k)$, if R is corrupted and SM_k is $\mathsf{fourth\ special\ message}$, the following modification is made.

1. In Stage 6, $\beta_{b,i}$ is $\beta_{b,i} = \rho_{b,i} \oplus s_{i,b\oplus\alpha_i}$ rather than a random bit for every $b \in \{0,1\}$ and $i \in \Delta \setminus I_b$.

Notice that $H'''_{k:6}$ is identical with $H_{k:7}$.

Claim 3.7.6. *Assume that in $H_{k:6}$, $\mathcal{A}$ does not cheat in sessions $s(k), \ldots, s(4m)$ except with negligible probability. Then,*

- *$H_{k:6}$ and $H'_{k:6}$ are indistinguishable, and*
- *in $H'_{k:6}$, $\mathcal{A}$ does not cheat in sessions $s(k), \ldots, s(4m)$ except with negligible probability.*

Recall that $H_{k:6}$ and $H'_{k:6}$ differ only in that in session $s(k)$ of $H'_{k:6}$, if R is corrupted and SM_k is fourth special message, $\beta_{b,i}$ is a random bit rather than $\beta_{b,i} = \rho_{b,i} \oplus s_{i,b\oplus\alpha_i}$ for every $b \in \{0,1\}$ and $i \in \Delta \setminus I_b$. Intuitively, we can prove this claim by using the security of mS-OT: For every $i \in \Delta \setminus I_b$, $\mathcal{A}$ executed the i-th mS-OT honestly with choice bit $(1-b) \oplus \alpha_i$, and the sender's input and randomness of this mS-OT are not revealed in Stage 8; therefore, the value of $s_{i,b\oplus\alpha_i}$ is hidden from $\mathcal{A}$ and thus $\beta_{b,i} = \rho_{b,i} \oplus s_{i,b\oplus\alpha_i}$ is indistinguishable from a random bit. Formally, we prove this claim in the same way as we do for Claim 3.7.2 (we use the security of mS-OT rather than the hiding of ExtCom); a formal proof is given in Section 3.9.6.

Claim 3.7.7. *Assume that in $H'_{k:6}$, $\mathcal{A}$ does not cheat in sessions $s(k), \ldots, s(4m)$ except with negligible probability. Then,*

- *$H'_{k:6}$ and $H''_{k:6}$ are indistinguishable, and*
- *in $H''_{k:6}$, $\mathcal{A}$ does not cheat in sessions $s(k), \ldots, s(4m)$ except with negligible probability.*

Proof. Recall that hybrid $H''_{k:6}$ differs from $H'_{k:6}$ in that in Stage 6 of session $s(k)$, either the hybrid is aborted or $\boldsymbol{\rho}_{1-u^*} = \{\rho_{1-u^*,i}\}_{i\in\Delta}$ is a secret sharing of a random bit rather than that of v_{1-u^*}.

For proving the lemma, it suffices to show that in session $s(k)$ of $H''_{k:6}$, the hybrid is not aborted (i.e. we have $|I_0| \le 6n$ or $|I_1| \le 6n$) except with negligible probability. To see that showing this is indeed sufficient for proving the lemma, observe the following: First, if the hybrid is not aborted, we have $|I_{1-u^*}| \le 6n$, so $\beta_{1-u^*,i}$ is a random bit on at least $4n$ indices and thus $\rho_{1-u^*,i}$ is hidden on at least $4n$ indices, which implies that $H'_{k:6}$ and $H''_{k:6}$ are statistically indistinguishable. Second, since $H'_{k:6}$ and $H''_{k:6}$ proceed indentically until the beginning of Stage 6-2 of session $s(k)$, and

1. if the experiment is not aborted in session $s(k)$, $H'_{k:6}$ and $H''_{k:6}$ continue to proceed identically after Stage 6-2 of session $s(k)$, and
2. if the hybrid is aborted in session $s(k)$, $\mathcal{A}$ clearly does not cheat in any session after Stage 6-2 of session $s(k)$,

the probability that $\mathcal{A}$ cheat in sessions $s(k), ..., s(4m)$ is not increased in $H''_{k:6}$.

Hence, we show that in sessoion $s(k)$ of $H''_{k:7}$, the hybrid is not aborted in except with negligible probability, or equivalently, that we have $|I_0| \le 6n$ or $|I_1| \le 6n$ except with negligible probability. Since $H''_{k:7}$ proceeds identically with $H'_{k:7}$ until Stage 6-2 of session $s(k)$, we have that $\mathcal{A}$ does not cheat in session $s(k)$ of $H''_{k:7}$ except with negligible probability, so it suffices to show that in session $s(k)$ of $H''_{k:7}$, we have either $|I_0| \le 6n$ or $|I_1| \le 6n$

whenever $\mathcal{A}$ does not cheat. Assume that $\mathcal{A}$ does not cheat in session $s(k)$ of $H''_{k:7}$. Then, since $|\Gamma_R| = n$ and that the number of indices on which $\mathcal{A}$ does not execute mS-OT using the outcome of coin-tossing is at most n, we have $|I_0 \cap I_1| \leq 2n$. Now, since $I_0, I_1 \subset \Delta$ and thus $|I_0 \cup I_1| \leq |\Delta| = 10n$, we have $|I_0| + |I_1| \leq 12n$, and hence, we have either $|I_0| \leq 6n$ or $|I_1| \leq 6n$. □

Claim 3.7.8. *Assume that in $H''_{k:6}$, $\mathcal{A}$ does not cheat in sessions $s(k), \ldots, s(4m)$ except with negligible probability. Then,*

– *$H''_{k:6}$ and $H'''_{k:6}$ are indistinguishable, and*

– *in $H'''_{k:6}$, $\mathcal{A}$ does not cheat in sessions $s(k), \ldots, s(4m)$ except with negligible probability.*

Proof. This claim can be proven identically with Claim 3.7.6. □

This completes the proof of Lemma 3.7.9. □

From Lemma 3.7.3 to Lemma 3.7.9, we conclude that the output of H_0 and that of $H_{4m:7}$ are indistinguishable, i.e. , the output of the real world and that of the ideal world are indistinguishable. This concludes the proof of Theorem 3.5.1.

3.8 Security Proof for Our MPC Protocol

Simulator Sim. As in Section 3.7.1, we consider a simulator that works against any adversary, say $\mathcal{A}$, that participates in m sessions of Π_{2PC}. Our simulator Sim internally invokes the adversary $\mathcal{A}$, and simulates each of the sessions by using the simulator of Π_{OT} (Section 3.7.1) and that of $\Pi_{\mathrm{2PC}}^{\mathcal{F}_{\mathrm{OT}}}$ as follows.

1. In each execution of Π_{OT} at the beginning of Π_{2PC}, Sim simulates the honest party's messages for $\mathcal{A}$ in the same way as $\mathsf{Sim}_{\mathrm{OT}}$.

   Recall that $\mathsf{Sim}_{\mathrm{OT}}$ makes a query to $\mathcal{F}_{OT}$ during the simulation. When $\mathsf{Sim}_{\mathrm{OT}}$ makes a query to $\mathcal{F}_{OT}$, Sim sends those queries to the simulator of $\Pi_{\mathrm{2PC}}^{\mathcal{F}_{\mathrm{OT}}}$ in order to simulate the answer from $\mathcal{F}_{OT}$. (Recall that the simulator of $\Pi_{\mathrm{2PC}}^{\mathcal{F}_{\mathrm{OT}}}$ simulates $\mathcal{F}_{OT}$ for the adversary.)

2. In the execution of $\Pi_{\mathrm{2PC}}^{\mathcal{F}_{\mathrm{OT}}}$ during Π_{2PC}, Sim simulates the honest party's messages for $\mathcal{A}$ by using the simulator of $\Pi_{\mathrm{2PC}}^{\mathcal{F}_{\mathrm{OT}}}$, who obtained the queries to $\mathcal{F}_{OT}$ as above.

We remark that here we use the simulator of $\Pi_{\mathrm{2PC}}^{\mathcal{F}_{\mathrm{OT}}}$ in the setting where multiple sessions of $\Pi_{\mathrm{2PC}}^{\mathcal{F}_{\mathrm{OT}}}$ are concurrently executed. However, the use of it in this setting does not cause any problem because it runs in the black-box straight-line manner.

3.8.1 Proof of Indistinguishability.

We show that the view of the adversary in the real world and the view output by the simulator in the ideal world are indistinguishable. The proof proceeds very similarly to the proof for our bounded concurrent OT protocol (Section 3.5). To simplify the exposition, below we assume that $\Pi_{\mathrm{2PC}}^{\mathcal{F}_{\mathrm{OT}}}$ makes only a single call to $\mathcal{F}_{OT}$. (The proof can be modified straightforwardly when $\Pi_{\mathrm{2PC}}^{\mathcal{F}_{\mathrm{OT}}}$ makes multiple calls to $\mathcal{F}_{OT}$.)

Recall that Π_{2PC} is obtained by composing our OT protocol Π_{OT} with an OT-hybrid 2PC protocol $\Pi_{\mathrm{2PC}}^{\mathcal{F}_{\mathrm{OT}}}$. Roughly, we consider a sequence of hybrid experiments in which:

– Each execution of Π_{OT} is gradually changed to simulation as in the sequence of hybrid experiments that we considered in the proof of Π_{OT} (Section 3.7.2.1).

– Once the execution of Π_{OT} in a session of Π_{2PC} is changed to simulation completely, the execution of $\Pi_{\mathrm{2PC}}^{\mathcal{F}_{\mathrm{OT}}}$ in that session is changed to simulation.

More concretely, we consider hybrids H_0, H_0^* and $H_{k:1}, \dots, H_{k:9}$ for $k \in [4m]$, where hybrids $H_{k:8}$ and $H_{k:9}$ are defined in the following, and the others are defined as in Section 3.7.2.1.

Hybrid $H_{k:8}$. $H_{k:8}$ is the same as $H_{k:7}$ except that in session $s(k)$, if S is corrupted and SM_k is third special message, all the messages of $\Pi_{\mathrm{2PC}}^{\mathcal{F}_{\mathrm{OT}}}$ from R are generated by the simulator of $\Pi_{\mathrm{2PC}}^{\mathcal{F}_{\mathrm{OT}}}$. More concretely, the messages of $\Pi_{\mathrm{2PC}}^{\mathcal{F}_{\mathrm{OT}}}$ from R are generated as follows. Recall that from the definition of Hybrid $H_{k:3}$, the implicit input $v_b^* \stackrel{\mathrm{def}}{=} \mathsf{Value}(\boldsymbol{\rho}_b^{\mathsf{ext}}, \Gamma_R \cap \Delta)$ ($b \in \{0,1\}$) to Π_{OT} is extracted from the adversary in session $s(k)$ (as $\boldsymbol{\rho}_b^{\mathsf{ext}}$ are computed for both $b \in \{0,1\}$). Now, the messages of $\Pi_{\mathrm{2PC}}^{\mathcal{F}_{\mathrm{OT}}}$ from R are simulated by feeding those extracted implicit input and the subsequent messages to the simulator of $\Pi_{\mathrm{2PC}}^{\mathcal{F}_{\mathrm{OT}}}$.

Hybrid $H_{k:9}$. $H_{k:9}$ is the same as $H_{k:8}$ except that in session $s(k)$, if R is corrupted and SM_k is fourth special message, all the messages of $\Pi_{\mathrm{2PC}}^{\mathcal{F}_{\mathrm{OT}}}$ from S are generated by the simulator of $\Pi_{\mathrm{2PC}}^{\mathcal{F}_{\mathrm{OT}}}$.

Lemma 3.8.1. *Assume that in $H_{k:7}$ ($k \in [4m]$), $\mathcal{A}$ does not cheat in sessions $s(k), \dots, s(4m)$ except with negligible probability. Then,*

– *$H_{k:7}$ and $H_{k:8}$ are indistinguishable, and*

– *in $H_{k:8}$, $\mathcal{A}$ does not cheat in sessions $s(k), \dots, s(4m)$ except with negligible probability.*

Lemma 3.8.2. *Assume that in $H_{k:8}$ ($k \in [4m]$), $\mathcal{A}$ does not cheat in sessions $s(k), \dots, s(4m)$ except with negligible probability. Then,*

– *$H_{k:8}$ and $H_{k:9}$ are indistinguishable, and*

– *in $H_{k:9}$, $\mathcal{A}$ does not cheat in sessions $s(k), \dots, s(4m)$ except with negligible probability.*

Lemma 3.8.2 can be proven identically with Lemma 3.8.1, and Lemma 3.8.1 can be proven quite similarly to Claim 3.7.2 (Section 3.7.2); the only difference is that we use the security of $\Pi_{\mathrm{2PC}}^{\mathcal{F}_{\mathrm{OT}}}$ rather than the hiding of ExtCom. We give a proof of Lemma 3.8.1 in Section 3.8.2.

By combining Lemmas 3.8.1 and 3.8.2 with Lemma 3.7.3 to 3.7.9 in Section 3.7.2, we conclude that the output of H_0 and that of $H_{4m:9}$ are indistinguishable, i.e. , the output of the real world and that of the ideal world are indistinguishable. This concludes the proof of Theorem 3.6.1.

3.8.2 Proof of Lemma 3.8.1

Proof of Lemma 3.8.1. We first show the indistinguishability between $H_{k:7}$ and $H_{k:8}$. Assume for contradiction that $H_{k:7}$ and $H_{k:8}$ are distinguishable. From an average argument, we can fix the execution of the experiment up until SM_k (inclusive) in such a way that even after being fixed, $H_{k:7}$ and $H_{k:8}$ are still distinguishable. Then, by considering the transcript up until SM_k and the table $\mathcal{T}$ as non-uniform advice, we can break the UC security of $\Pi_{\mathrm{2PC}}^{\mathcal{F}_{\mathrm{OT}}}$ as follows.

> The environment $\mathcal{Z}$ internally executes $H_{k:7}$ from SM_k using the non-uniform advice while externally participating in a single session of $\Pi_{\mathrm{2PC}}^{\mathcal{F}_{\mathrm{OT}}}$ via the dummy adversary that corrupts S. In session $s(k)$, $\mathcal{Z}$ forwards all the messages of $\Pi_{\mathrm{2PC}}^{\mathcal{F}_{\mathrm{OT}}}$ from the internal $\mathcal{A}$ to the external dummy adversary (including the query to $\mathcal{F}_{OT}$),[14] and those from the external dummy adversary to the internal $\mathcal{A}$. After the execution of $H_{k:7}$ finishes, $\mathcal{Z}$ outputs the output of the internally emulated experiment.

When $\mathcal{Z}$ interacts with the dummy adversary, the internally executed experiment is identical with $H_{k:7}$, whereas when $\mathcal{Z}$ interacts with the simulator of $\Pi_{\mathrm{2PC}}^{\mathcal{F}_{\mathrm{OT}}}$, the internally executed experiment is identical with $H_{k:8}$. Hence, from the assumption that $H_{k:7}$ and $H_{k:8}$ are distinguishable, $\mathcal{Z}$ breaks the security of $\Pi_{\mathrm{2PC}}^{\mathcal{F}_{\mathrm{OT}}}$

We next show that in $H_{k:8}$, $\mathcal{A}$ does not cheat in sessions $s(k), \ldots, s(4m)$. Assume for contradiction that in $H_{k:8}$, $\mathcal{A}$ cheats in one of those sessions, say, session $s(j)$, with non-negligible probability. Then, from an average argument, we can fix the execution of the experiment up until SM_k (inclusive) in such a way that even after being fixed, $\mathcal{A}$ cheats in session $s(j)$ only with negligible probability in $H_{k:7}$ but with non-negligible probability in $H_{k:8}$. Then, by considering the transcript up until SM_k and the table $\mathcal{T}$ as non-uniform advice, we can break the robust non-malleability of NMCom as follows.

> The adversary $\mathcal{A}_{\mathsf{NMCom}}$, who participates in an execution of $\Pi_{\mathrm{2PC}}^{\mathcal{F}_{\mathrm{OT}}}$ as the environment (where the dummy adversary corrupts S) while interacting with a receiver of NMCom, internally executes $H_{k:7}$ from SM_k using the non-uniform advice. In session $s(k)$, $\mathcal{A}_{\mathsf{NMCom}}$ forwards all the messages of $\Pi_{\mathrm{2PC}}^{\mathcal{F}_{\mathrm{OT}}}$ from the internal $\mathcal{A}$ to the external dummy adversary (including the query to $\mathcal{F}_{OT}$), and those from the external dummy adversary to the internal $\mathcal{A}$. Also, in session $s(j)$, $\mathcal{A}_{\mathsf{NMCom}}$ forwards the NMCom commitments from $\mathcal{A}$ to the external receiver. After the execution of $H_{k:7}$ finishes, $\mathcal{A}_{\mathsf{NMCom}}$ outputs the output of the internally emulated experiment.
>
> The distinguisher $\mathcal{D}_{\mathsf{NMCom}}$ takes as input the view of $\mathcal{A}_{\mathsf{NMCom}}$ and the values committed by $\mathcal{A}_{\mathsf{NMCom}}$ (which are equal to the values committed to by $\mathcal{A}$ in session $s(j)$ in the internally executed experiment). $\mathcal{D}_{\mathsf{NMCom}}$ then outputs 1 if and only if $\mathcal{A}$ cheated in session $s(j)$.

When $\mathcal{A}_{\mathsf{NMCom}}$ interacts with the dummy adversary in the execution of $\Pi_{\mathrm{2PC}}^{\mathcal{F}_{\mathrm{OT}}}$, the internally executed experiment is identical with $H_{k:7}$, whereas when $\mathcal{A}_{\mathsf{NMCom}}$ interacts with the simulator there, the internally executed experiment is identical with $H_{k:8}$. Hence, from the assumption that $\mathcal{A}$ cheats in session $s(j)$ with negligible probability in $H_{k:7}$ but with non-negligible probability in $H_{k:8}$, $\mathcal{A}_{\mathsf{NMCom}}$ breaks the robust non-malleability of NMCom.

This completes the proof of Lemma 3.8.1. $\square$

[14] Note that these messages appear after SM_k

3.9 Postponed Proofs

3.9.1 The Second Half of the Proof for Lemma 3.7.1

Case 2. S is corrupted in the $i^*(n)$-th session. We show that when $\mathcal{A}$ cheats, we can break the hiding property of the $\mathsf{RobCom}(\phi^S)$ commitment in Stage 0-2 (i.e., the commitment by which ϕ^R is committed to). From the definition of the invariant condition (Definition 3.7.2), when $\mathcal{A}$ cheats, we have $I_{\text{bad}} \cap \Gamma_R = \emptyset$ and either $|I_{\text{bad}}| \geq 0.1n$ or $\exists b \in \{0,1\}$ s.t. $\boldsymbol{\rho}_b^{\mathsf{nm}}$ is 0.85-close to but 0.1-far from a valid codeword $\boldsymbol{w} = (w_i)_{i\in\Delta}$ that satisfies $w_i = \rho_{b,i}^{\mathsf{nm}}$ for every $i \in \Gamma_R$, where I_{bad} and $\boldsymbol{\rho}_b^{\mathsf{nm}}$ are defined from the committed values of the NMCom commitments in Stage 7. Similar to Case 1, we first show that we can "approximate" I_{bad} and $\boldsymbol{\rho}_b^{\mathsf{nm}}$ by extracting the committed values of the ExtCom commitments in Stage 7 using its extractability.

First, we observe that if we extract the committed values of the ExtCom commitments in Stage 7 of the $i^*(n)$-th session, the extracted values, $(\hat{a}_1^S, \hat{d}_1^S, \hat{e}_1^S), \ldots, (\hat{a}_{11n}^S, \hat{d}_{11n}^S, \hat{e}_{11n}^S)$, satisfy the following.

- Let $\hat{I}_{\text{bad}} \subset [11n]$ be a set such that $i \in \hat{I}_{\text{bad}}$ if and only if

  1. $((\hat{a}_1^S, \hat{d}_1^S), \hat{e}_1^S)$ is not a valid decommitment of the i-th NMCom commitment in Stage 7, **or**
  2. $(\hat{a}_i^S, \hat{d}_i^S)$ is not valid in terms of the check in Stage 8-3b, i.e. $\hat{d}_i^S$ is not a valid decommitment of ψ_i^S w.r.t. Stage-0-1 RobCom, or $\hat{a}_i^S \oplus \psi_i^S$ does not equal the value z_i^S it received in Stage-2-1; **or**
  3. S does not execute the i-th mS-OT in Stage 3 honestly using $\hat{s}_{i,0} \,\|\, \hat{s}_{i,1} \,\|\, \hat{\tau}_i^S$ as the input and randomness, where $\hat{s}_{i,0} \,\|\, \hat{s}_{i,1} \,\|\, \hat{\tau}_i^S$ is obtained from $\hat{r}_i^S = \hat{a}_i^S \oplus b_i^S$.

  Also, for each $b \in \{0,1\}$, let $\hat{\boldsymbol{\rho}}_b = (\hat{\rho}_{b,i})_{i\in\Delta}$ be defined as follows: $\hat{\rho}_{b,i} \overset{\text{def}}{=} \beta_{b,i} \oplus \hat{s}_{i,b\oplus\alpha_i}$ if $i \notin \hat{I}_{\text{bad}}$ and $\hat{\rho}_{b,i} \overset{\text{def}}{=} \bot$ otherwise. Then, we have

  * $\hat{I}_{\text{bad}} \cap \Gamma_R = \emptyset$, and
  * either $|\hat{I}_{\text{bad}}| \geq 0.1n$ or there exists $b \in \{0,1\}$ such that $\hat{\rho}_b$ is 0.8-close to but 0.1-far from a valid codeword $\boldsymbol{w} = (w_i)_{i\in\Delta}$ that satisfies $w_i = \hat{\rho}_{b,i}$ for every $i \in \Gamma_R$

  with probability at least $1/2p(n)$.

More precisely, we observe that when $\mathcal{A}$ cheats in the $i^*(n)$-th session, the extracted values satisfied the above condition except with negligible probability. Recall that when $\mathcal{A}$ cheats, the cut-and-choose in Stage 8 is accepting but we have

- $|I_{\text{bad}}| \geq 0.1n$, or
- $\exists b \in \{0,1\}$ s.t. ρ_b^{nm} is 0.85-close to but 0.1-far from a valid codeword $\boldsymbol{w} = (w_i)_{i\in\Delta}$ that satisfies $w_i = \rho_{b,i}^{\mathsf{nm}}$ for every $i \in \Gamma_R$.

Also, notice that we have $\hat{I}_{\text{bad}} \cap \Gamma_R = \emptyset$ when the cut-and-choose in Stage 8 is accepting, and have $|\hat{I}_{\text{bad}}| \geq 0.1n$ when $|I_{\text{bad}}| \geq 0.1n$ (this is because we have $I_{\text{bad}} \subseteq \hat{I}_{\text{bad}}$ from the definitions of $I_{\text{bad}}, \hat{I}_{\text{bad}}$). Hence, to show that the extracted values satisfy the above condition when $\mathcal{A}$ cheats, it suffices to show that when $\exists b^* \in \{0,1\}$ s.t. $\boldsymbol{\rho}_{b^*}^{\mathsf{nm}}$ is 0.85-close to but 0.1-far from a valid codeword $\boldsymbol{w} = (w_i)_{i \in \Delta}$ that satisfies $w_i = \rho_{b^*,i}^{\mathsf{nm}}$ for every $i \in \Gamma_R$, we have either $|\hat{I}_{\text{bad}}| \geq 0.1n$ or $\hat{\boldsymbol{\rho}}_{b^*}$ is 0.8-close to but 0.1-far from $\boldsymbol{w}$ and satisfies $w_i = \hat{\rho}_{b^*,i}$ for every $i \in \Gamma_R$. This can be shown as follows.

– If $|\hat{I}_{\text{bad}}| \geq 0.1n$, we are done.

– If $|\hat{I}_{\text{bad}}| < 0.1n$, we have that $\hat{\rho}_{b^*}$ is 0.8-close to but 0.1-far from $\boldsymbol{w}$ and satisfies $w_i = \hat{\rho}_{b^*,i}$ for every $i \in \Gamma_R$. This is because if $|\hat{I}_{\text{bad}}| < 0.1n$,

1. $\hat{\boldsymbol{\rho}}_{b^*}$ is 0.8-close to $\boldsymbol{w}$ since it is 0.99-close to $\boldsymbol{\rho}_{b^*}^{\mathsf{nm}}$ when $|\hat{I}_{\text{bad}}| < 0.1n$, and $\boldsymbol{\rho}_{b^*}^{\mathsf{nm}}$ is 0.85-close to $\boldsymbol{w}$,
2. $\hat{\boldsymbol{\rho}}_{b^*}$ is 0.1-far from $\boldsymbol{w}$ since for every i such that $\rho_{b^*,i}^{\mathsf{nm}} \neq w_i$, we have $\hat{\rho}_{b^*,i} \neq w_i$ from the definition of $\hat{\boldsymbol{\rho}}$, and
3. $\hat{\boldsymbol{\rho}}_{b^*}$ satisfies $w_i = \hat{\rho}_{b^*,i}$ for every $i \in \Gamma_R$ since we have $\hat{\rho}_{b^*,i} = \rho_{b^*,i}^{\mathsf{nm}}$ for every $i \in \Gamma_R$ when the cut-and-choose in Stage 8 is accepting, and $\boldsymbol{\rho}_{b^*}^{\mathsf{nm}}$ satisfies $w_i = \rho_{b^*,i}^{\mathsf{nm}}$ for every $i \in \Gamma_R$.

Based on this observation, we derive contradiction by considering the following adversary $\mathcal{A}_{\mathsf{RobCom}}$ against the hiding property of RobCom.

$\mathcal{A}_{\mathsf{RobCom}}$ receives a RobCom commitment c^* in which either ϕ_R^0 or ϕ_R^1 is committed. Then, $\mathcal{A}_{\mathsf{RobCom}}$ internally executes the experiment H_0 honestly except that in the $i^*(n)$-th session, $\mathcal{A}_{\mathsf{RobCom}}$ uses c^* as the commitment in Stage 0-2 (i.e. , as the RobCom commitment in which R commits to string which will be used to mask Γ_R in Stage 1-2). In Stage 1-2, $\mathcal{A}_{\mathsf{RobCom}}$ always use ϕ_R^1 to mask Γ_R. When the experiment H_0 reaches Stage 7 of the $i^*(n)$-th session, $\mathcal{A}_{\mathsf{RobCom}}$ extracts the committed values of the ExtCom commitments in this stage by using its extractability. Let $\hat{I}_{\text{bad}}$ and $\hat{\boldsymbol{\rho}}_b$ ($b \in \{0,1\}$) be defined as above from the extracted values. Then, $\mathcal{A}_{\mathsf{RobCom}}$ outputs 1 if and only if

– $\hat{I}_{\text{bad}} \cap \Gamma_R = \emptyset$, and
– either $|\hat{I}_{\text{bad}}| \geq 0.1n$ or there exists $b \in \{0,1\}$ such that $\hat{\boldsymbol{\rho}}_b$ is 0.8-close to but 0.1-far from a valid codeword $\boldsymbol{w} = (w_i)_{i \in \Delta}$ that satisfies $w_i = \hat{\rho}_{b,i}$ for every $i \in \Gamma_R$.

When $\mathcal{A}_{\mathsf{RobCom}}$ receives a commitment to ϕ_R^1, $\mathcal{A}_{\mathsf{RobCom}}$ outputs 1 with probability $1/2p(n)$ (this follows from the above observation). It thus suffices to see that when $\mathcal{A}_{\mathsf{RobCom}}$ receives a commitment to ϕ_R^0, $\mathcal{A}_{\mathsf{RobCom}}$ outputs 1 with exponentially small probability. This can be seen by noting that $\phi_1^R \oplus \Gamma_R$ is a pure random string now, and thus the following probabilities are exponentially small.

1. the probability that $|\hat{I}_{\text{bad}}| \geq 0.1n$ but $\hat{I}_{\text{bad}} \cap \Gamma_R = \emptyset$
2. the probability that there exists $b \in \{0,1\}$ such that $\hat{\boldsymbol{\rho}}_b$ is 0.8-close to but 0.1-far from a valid codeword $\boldsymbol{w} = (w_i)_{i \in \Delta}$ that satisfies $w_i = \hat{\rho}_{b,i}$ for every $i \in \Gamma_R$

Hence, $\mathcal{A}_{\mathsf{RobCom}}$ breaks the hiding property of RobCom.

3.9.2 Proof of Lemma 3.7.4

Proof. Recall that hybrids $H_{k:1}, H_{k:2}$ differ only in the input and the randomness that are used in some of the mS-OTs in Stage 3, where those that are derived from the outcomes of the coin tossing is used in $H_{k:1}$ and random inputs and true randomness are used in $H_{k:2}$. We first show the indistinguishability between $H_{k:1}$ and $H_{k:2}$, relying on the hiding property of RobCom.

Assume for contradiction that $H_{k:1}$ and $H_{k:2}$ are distinguishable. We build an efficient adversary $\mathcal{A}_{\mathsf{RobCom}}$ that breaks the hiding property of RobCom.

> The adversary $\mathcal{A}_{\mathsf{RobCom}}$ internally executes $H_{k:1}$ with the following modification: in Stage 0-2 of session $s(k)$, it picks two random strings $\psi^R = \psi^R_1 \| \ldots, \psi^R_{11n}$ and $\tilde{\psi}^R = \tilde{\psi}^R_1 \| \ldots, \tilde{\psi}^R_{11n}$ and sends $\{\psi^R_i\}_{i \notin \Gamma_S}$ and $\{\tilde{\psi}^R_i\}_{i \notin \Gamma_S}$ to the external committer and receives back RobCom^{f_R} commitments (in which either $\{\psi^R_i\}_{i \notin \Gamma_S}$ or $\{\tilde{\psi}^R_i\}_{i \notin \Gamma_S}$ are committed in parallel). Then in Stage 2-2 of session $s(k)$, $\mathcal{A}_{\mathsf{RobCom}}$ always use ψ^R_i's to mask a^R_i (i.e. $z^R_i := a^R_i \oplus \psi^R_i$ for all $i \in [11n]$). in the subsequent stages, $\mathcal{A}$ proceeds the experiment as in $H_{k:1}$. After the execution of $H_{k:1}$ finishes, $\mathcal{A}_{\mathsf{RobCom}}$ outputs whatever $\mathcal{Z}$ outputs in the experiment.

When $\mathcal{A}_{\mathsf{RobCom}}$ receives commitments to $\{\psi^R_i\}_{i \notin \Gamma_S}$, the internally executed experiment is identical with $H_{k:1}$, whereas when $\mathcal{A}_{\mathsf{RobCom}}$ receives commitments to $\{\tilde{\psi}^R_i\}_{i \notin \Gamma_S}$, the internally executed experiment is identical with $H_{k:2}$ (this is because when $\mathcal{A}_{\mathsf{RobCom}}$ receives commitments to $(\tilde{\psi}^R_i)_{i \notin \Gamma_S}$, the values $z^R_i = \psi^R_i \oplus a^R_i$ (thus the values $r^R_i = a^R_i \oplus b^R_i$) for each $i \notin \Gamma_S$ are uniformly random for $\mathcal{A}$. Hence the mS-OT for each $i \notin \Gamma_S$ is executed with a random input and true randomness). Hence, from the assumption that $H_{k:1}$ and $H_{k:2}$ are distinguishable, $\mathcal{A}_{\mathsf{RobCom}}$ distinguishes RobCom commitments.

We next show that in $H_{k:2}$, $\mathcal{A}$ does not cheat in sessions $s(k), \ldots, s(4m)$. Assume for contradiction that in $H_{k:2}$, $\mathcal{A}$ cheats in one of those sessions, say, session $s(j)$, with non-negligible probability. Then, from an average argument, we can fix the execution of the experiment up until SM_k (inclusive) in such a way that even after being fixed, $\mathcal{A}$ cheats in session $s(j)$ only with negligible probability in $H_{k:1}$ but with non-negligible probability in $H_{k:2}$. Then, we can break the robust non-malleability of NMCom as follows.

> The adversary $\mathcal{A}_{\mathsf{NMCom}}$, who interacts with a committer of RobCom and a receiver of NMCom, internally executes $H_{k:1}$ from SM_k using the non-uniform advice. In Stage 0-2 of session $s(k)$, $\mathcal{A}_{\mathsf{NMCom}}$ chooses random strings $\tilde{\psi}^R = \tilde{\psi}^R_1 \| \ldots \| \tilde{\psi}^R_{11n}$ in addition to $\psi^R = \psi^R_1 \| \ldots \| \psi^R_{11n}$, sends $\{\psi^R_i\}_{i \notin \Gamma_S}$ and $\{\tilde{\psi}^R_i\}_{i \notin \Gamma_S}$ to the external committer and receives back parallel RobCom commitments (in which either $\{\psi^R_i\}_{i \notin \Gamma_S}$ or $\{\tilde{\psi}^R_i\}_{i \notin \Gamma_S}$ are committed to), and feeds them into $H_{k:1}$. Then in Stage 2-2 of session $s(k)$, $\mathcal{A}_{\mathsf{NMCom}}$ always use ψ^R_i's to mask a^R_i (i.e. $z^R_i := a^R_i \oplus \psi^R_i$ for all $i \in [11n]$). Also, in session $s(j)$, $\mathcal{A}_{\mathsf{NMCom}}$ forwards the NMCom commitments from $\mathcal{A}$ to the external receiver. After the execution of $H_{k:1}$ finishes, $\mathcal{A}_{\mathsf{NMCom}}$ outputs its view.
>
> The distinguisher $\mathcal{D}_{\mathsf{NMCom}}$ takes as input the view of $\mathcal{A}_{\mathsf{NMCom}}$ and the values committed by $\mathcal{A}_{\mathsf{NMCom}}$ (which are equal to the values committed to by $\mathcal{A}$ in session $s(j)$ in the internally executed experiment). $\mathcal{D}_{\mathsf{NMCom}}$ then outputs 1 if and only if $\mathcal{A}$ cheated in session $s(j)$.

When $\mathcal{A}_{\mathsf{NMCom}}$ receives commitments to $\{\psi_i^R\}_{i \notin \Gamma_S}$, the internally executed experiment is identical with $H_{k:1}$, whereas when $\mathcal{A}_{\mathsf{Com}}$ receives commitments to $\{\tilde{\psi}_i^R\}_{i \notin \Gamma_S}$, the internally executed experiment is identical with $H_{k:2}$. Hence, from the assumption that $\mathcal{A}$ cheats in session $s(j)$ with negligible probability in $H_{k:1}$ but with non-negligible probability in $H_{k:2}$, $\mathcal{A}_{\mathsf{NMCom}}$ breaks the robust non-malleability of NMCom.

This completes the proof of Lemma 3.7.4. $\square$

3.9.3 Proof of Lemma 3.7.6

Proof. Recall that $H_{k:3}$ and $H_{k:4}$ differ only in that in session $s(k)$ of $H_{k:4}$, if S is corrupted and SM_k is third special message, α_i is a random bit rather than $\alpha_i = u \oplus c_i$ for every $i \in \Delta$ in Stage 6-1.

We first show the indistinguishability between $H_{k:3}$ and $H_{k:4}$. Intuitively, the indistinguishability follows from the security of mS-OT: For every $i \notin \Gamma_S$, the choice bit c_i of the i-th mS-OT in Stage 3 is hidden from $\mathcal{A}$ and hence $\alpha_i = u \oplus c_i$ in $H_{k:3}$ is indistinguishable from a random bit. Formally, we consider the following security game against cheating sender S^* of mS-OT.

> The cheating sender S^* first participates in $10n$ instances of mS-OTs in parallel with an honest receiver R, who uses a random input $c_i \in \{0,1\}$ in the i-th instance. After the execution with R, S^* receives either the choice bits $\{c_i\}$ or random bits and then guesses which is the case. If S^* guesses correctly, we say that S^* wins the game.

From the security of mS-OT against malicious senders, any cheating S^* wins the game with probability at most $1/2 + \mathsf{negl}(n)$. Now, we assume for contradiction that $H_{k:3}$ and $H_{k:4}$ are distinguishable, and we derive a contradiction by constructing an adversary who wins the above game with probability non-negligibly higher than $1/2$. From an average argument, we can fix the execution of the experiment up until SM_k (inclusive) in such a way that even after being fixed, $H_{k:3}$ and $H_{k:4}$ are still distinguishable. Then, by considering the transcript up until SM_k and the table $\mathcal{T}$ as non-uniform advice, we can obtain an adversary who wins the above game with probability non-negligibly higher than $1/2$ as follows.

> The adversary $\mathcal{A}_{\mathsf{OT}}$ internally executes $H_{k:3}$ from SM_k using the non-uniform advice. In Stage 3 of session $s(k)$, $\mathcal{A}_{\mathsf{OT}}$ executes the i-th mS-OT by itself for every $i \in \Gamma_S$ but obtains the other $10n$ instances of mS-OT from the external receiver. (Recall that in $H_{k:3}$, the subset Γ_S is extracted in Stage 1-1.) Then, in Stage 6 of session $s(k)$, $\mathcal{A}_{\mathsf{OT}}$ receives bits $\{c_i^*\}_{i \in \Delta}$ from the external receiver and uses them to compute $\{\alpha\}_{i \in \Delta}$, i.e. , $\alpha_i \stackrel{\text{def}}{=} u \oplus c_i^*$. After the execution of $H_{k:3}$ finishes, $\mathcal{A}_{\mathsf{OT}}$ outputs whatever $\mathcal{Z}$ outputs in the experiment.

When $\mathcal{A}_{\mathsf{OT}}$ receives the choice bits of the mS-OTs as $\{c_i^*\}_{i \in \Delta}$, the internally executed experiment is identical with $H_{k:3}$, whereas when $\mathcal{A}_{\mathsf{OT}}$ receives random bits as $\{c_i^*\}_{i \in \Delta}$, the internally executed experiment is identical with $H_{k:4}$. Hence, from the assumption that $H_{k:3}$ and $H_{k:4}$ are distinguishable, $\mathcal{A}_{\mathsf{OT}}$ wins the game with probability non-negligibly higher than $1/2$.

We next show that in $H_{k:4}$, $\mathcal{A}$ does not cheat in sessions $s(k), \ldots, s(4m)$. (The argument below is similar to the one in the proof of Lemma 3.7.3.) Assume for contradiction that in $H_{k:4}$, $\mathcal{A}$ cheats in one of those sessions, say, session $s(j)$, with non-negligible probability. Then, from an average argument, we can fix the execution of the experiment up until SM_k (inclusive) in such a way that even after being fixed, $\mathcal{A}$ cheats in session $s(j)$ only with negligible probability in $H_{k:3}$ but with non-negligible probability in $H_{k:4}$. Then, by considering the transcript up until SM_k and the table $\mathcal{T}$ as non-uniform advice, we can break the robust non-malleability of NMCom as follows.

> The adversary $\mathcal{A}_{\mathsf{NMCom}}$, who participates in the above game of mS-OT while interacting with a receiver of NMCom, internally executes $H_{k:3}$ from SM_k using the non-uniform advice. In Stage 3 of session $s(k)$, $\mathcal{A}_{\mathsf{OT}}$ executes the i-th mS-OT by itself for every $i \in \Gamma_S$ but obtains the other $10n$ instances of mS-OT from the external receiver. Then, in Stage 6 of session $s(k)$, $\mathcal{A}_{\mathsf{OT}}$ receives bits $\{c_i^*\}_{i\in\Delta}$ from the external receiver and uses them to compute $\{\alpha\}_{i\in\Delta}$, i.e. , $\alpha_i \overset{\text{def}}{=} u \oplus c_i^*$. Also, in session $s(j)$, $\mathcal{A}_{\mathsf{NMCom}}$ forwards the NMCom commitments from $\mathcal{A}$ to the external receiver. After the execution of $H_{k:3}$ finishes, $\mathcal{A}_{\mathsf{NMCom}}$ outputs its view.
>
> The distinguisher $\mathcal{D}_{\mathsf{NMCom}}$ takes as input the view of $\mathcal{A}_{\mathsf{NMCom}}$ and the values committed by $\mathcal{A}_{\mathsf{NMCom}}$ (which are equal to the values committed to by $\mathcal{A}$ in session $s(j)$ in the internally executed experiment). $\mathcal{D}_{\mathsf{NMCom}}$ then outputs 1 if and only if $\mathcal{A}$ cheated in session $s(j)$.

When $\mathcal{A}_{\mathsf{OT}}$ receives the choice bits of the mS-OTs as $\{c_i^*\}_{i\in\Delta}$, the internally executed experiment is identical with $H_{k:3}$, whereas when $\mathcal{A}_{\mathsf{OT}}$ receives random bits as $\{c_i^*\}_{i\in\Delta}$, the internally executed experiment is identical with $H_{k:4}$. Hence, from the assumption that $\mathcal{A}$ cheats in session $s(j)$ with negligible probability in $H_{k:3}$ but with non-negligible probability in $H_{k:4}$, $\mathcal{A}_{\mathsf{NMCom}}$ breaks the robust non-malleability of NMCom.

This completes the proof of Lemma 3.7.6. □

3.9.4 Proof of Lemma 3.7.7

Proof. Recall that hybrids $H_{k:4}, H_{k:5}$ differ only in the values committed to in NMCom and ExtCom for the indices outside of Γ_R. Since the binding property of RobCom guarantees that the subset opened in Stage 7 is equal to Γ_R, those commitments are never opened, and the check in Stage 8 does not fail in both hybrids.

We prove the lemma by using a hybrid argument. Specifically, we consider the following intermediate hybrid $H'_{k:5}$.

– $H'_{k:5}$ is the same as $H_{k:4}$ except that in session $s(k)$, if R is corrupted and SM_k is second special message,

  * the committed subset Γ_R is extracted by querying the table $\mathcal{T}$, and
  * the value committed in the i-th ExtCom commitment in Stage 7 is switched to an all-zero string for every $i \notin \Gamma_R$.

Claim 3.9.1. *Assume that in $H_{k:4}$, $\mathcal{A}$ does not cheat in sessions $s(k), \ldots, s(4m)$ except with negligible probability. Then,*

– *$H_{k:4}$ and $H'_{k:5}$ are indistinguishable, and*

– *in $H'_{k:5}$, $\mathcal{A}$ does not cheat in sessions $s(k), \ldots, s(4m)$ except with negligible probability.*

Proof. We first show the indistinguishability between $H_{k:4}$ and $H'_{k:5}$. Assume for contradiction that $H_{k:4}$ and $H'_{k:5}$ are distinguishable. From an average argument, we can fix the execution of the experiment up until SM_k (inclusive) in such a way that even after being fixed, $H_{k:4}$ and $H'_{k:5}$ are still distinguishable. Then, by considering the transcript up until SM_k and the table $\mathcal{T}$ as non-uniform advice, we can break the hiding property of ExtCom as follows.

> The adversary $\mathcal{A}_{\mathsf{ExtCom}}$ internally executes $H_{k:4}$ from SM_k using the non-uniform advice. In Stage 7 of session $s(k)$, $\mathcal{A}_{\mathsf{ExtCom}}$ sends $(a_i^S, d_i^S, e_i^S)_{i \notin \Gamma_R}$ and $(0,0,0)_{i \notin \Gamma_R}$ to the external committer, receives back ExtCom commitments (in which either $(a_i^S, d_i^S, e_i^S)_{i \notin \Gamma_R}$ or $(0,0,0)_{i \notin \Gamma_R}$ are committed to), and feeds them into $H_{k:4}$. After the execution of $H_{k:4}$ finishes, $\mathcal{A}_{\mathsf{ExtCom}}$ outputs whatever $\mathcal{Z}$ outputs in the experiment.

When $\mathcal{A}_{\mathsf{ExtCom}}$ receives commitments to $(a_i^S, d_i^S, e_i^S)_{i \notin \Gamma_R}$, the internally executed experiment is identical with $H_{k:4}$, whereas when $\mathcal{A}_{\mathsf{ExtCom}}$ receives a commitments to $(0,0,0)_{i \notin \Gamma_R}$, the internally executed experiment is identical with $H'_{k:5}$. Hence, from the assumption that $H_{k:4}$ and $H'_{k:5}$ are distinguishable (even after being fixed up until SM_k), $\mathcal{A}_{\mathsf{ExtCom}}$ distinguishes ExtCom commitments.

We next show that in $H'_{k:5}$, $\mathcal{A}$ does not cheat in sessions $s(k), \ldots, s(4m)$. Assume for contradiction that in $H'_{k:5}$, $\mathcal{A}$ cheats in one of those sessions, say, session $s(j)$, with non-negligible probability. Then, from an average argument, we can fix the execution of the experiment up until SM_k (inclusive) in such a way that even after being fixed, $\mathcal{A}$ cheats in session $s(j)$ only with negligible probability in $H_{k:4}$ but with non-negligible probability in $H'_{k:5}$. Then, by considering the transcript up until SM_k and the table $\mathcal{T}$ as non-uniform advice, we can break the robust non-malleability of NMCom as follows.

> The man-in-the-meddle adversary $\mathcal{A}_{\mathsf{NMCom}}$ internally executes $H_{k:4}$ from SM_k using the non-uniform advice. In Stage 7 of session $s(k)$, $\mathcal{A}_{\mathsf{NMCom}}$ sends $(a_i^S, d_i^S, e_i^S)_{i \notin \Gamma_R}$ and $(0,0,0)_{i \notin \Gamma_R}$ to the external committer, receives back ExtCom commitments (in which either $(a_i^S, d_i^S, e_i^S)_{i \notin \Gamma_R}$ or $(0,0,0)_{i \notin \Gamma_R}$ are committed to), and feeds them into $H_{k:4}$. Also, in session $s(j)$, $\mathcal{A}_{\mathsf{NMCom}}$ forwards the NMCom commitments from $\mathcal{A}$ to the external receiver. After the execution of $H_{k:4}$ finishes, $\mathcal{A}_{\mathsf{NMCom}}$ outputs its view.
>
> The distinguisher $\mathcal{D}_{\mathsf{NMCom}}$ takes as input the view of $\mathcal{A}_{\mathsf{NMCom}}$ and the values committed by $\mathcal{A}_{\mathsf{NMCom}}$ (which are equal to the values committed to by $\mathcal{A}$ in session $s(j)$ in the internally executed experiment). $\mathcal{D}_{\mathsf{NMCom}}$ then outputs 1 if and only if $\mathcal{A}$ cheated in session $s(j)$.

When $\mathcal{A}_{\mathsf{NMCom}}$ receives commitments to $(a_i^S, d_i^S, e_i^S)_{i \notin \Gamma_R}$, the internally executed experiment is identical with $H_{k:4}$, whereas when $\mathcal{A}_{\mathsf{NMCom}}$ receives a commitments to $(0,0,0)_{i \notin \Gamma_R}$, the internally executed experiment is identical with $H'_{k:5}$. Hence, from the assumption that $\mathcal{A}$ cheats in session $s(j)$ with negligible probability in $H_{k:4}$ and $H'_{k:5}$, $\mathcal{A}_{\mathsf{NMCom}}$ breaks the non-malleability of NMCom. □

Claim 3.9.2. *Assume that in $H'_{k:5}$, $\mathcal{A}$ does not cheat in sessions $s(k), \ldots, s(4m)$ except with negligible probability. Then,*

- *$H'_{k:5}$ and $H_{k:5}$ are indistinguishable, and*
- *in $H_{k:5}$, $\mathcal{A}$ does not cheat in sessions $s(k), \ldots, s(4m)$ except with negligible probability.*

Proof. We first notice that the indistinguishability between $H'_{k:5}$ and $H_{k:5}$ can be shown as in the proof of Claim 3.9.1. (The only difference is that we use the hiding property of NMCom rather than that of ExtCom.)

We next show that in $H_{k:5}$, $\mathcal{A}$ does not cheat in sessions $s(k), \ldots, s(4m)$. Assume for contradiction that in $H_{k:5}$, $\mathcal{A}$ cheats in one of those sessions, say, session $s(j)$, with non-negligible probability. Then, from an average argument, we can fix the execution of the experiment up until SM_k (inclusive) in such a way that even after being fixed, $\mathcal{A}$ cheats in session $s(j)$ only with negligible probability in $H'_{k:5}$ but with non-negligible probability in $H_{k:5}$. Then, by considering the transcript up until SM_k and the table $\mathcal{T}$ as non-uniform advice, we can break the non-malleability of NMCom as follows.

> The man-in-the-meddle adversary $\mathcal{A}_{\mathsf{NMCom}}$ internally executes $H'_{k:5}$ from SM_k using the non-uniform advice. In Stage 7 of session $s(k)$, $\mathcal{A}_{\mathsf{NMCom}}$ sends $(a_i^S, d_i^S)_{i \notin \Gamma_R}$ and $(0,0)_{i \notin \Gamma_R}$ to the external committer, receives back NMCom commitments (in which either $(a_i^S, d_i^S)_{i \notin \Gamma_R}$ or $(0,0)_{i \notin \Gamma_R}$ are committed to), and feeds them into $H'_{k:5}$. Also, in session $s(j)$, $\mathcal{A}_{\mathsf{NMCom}}$ forwards the NMCom commitments from $\mathcal{A}$ to the external receiver. After the execution of $H'_{k:5}$ finishes, $\mathcal{A}_{\mathsf{NMCom}}$ outputs its view.
>
> The distinguisher $\mathcal{D}_{\mathsf{NMCom}}$ takes as input the view of $\mathcal{A}_{\mathsf{NMCom}}$ and the values committed by $\mathcal{A}_{\mathsf{NMCom}}$ (which are equal to the values committed to by $\mathcal{A}$ in session $s(j)$ in the internally executed experiment). $\mathcal{D}_{\mathsf{NMCom}}$ then outputs 1 if and only if $\mathcal{A}$ cheated in session $s(j)$.

When $\mathcal{A}_{\mathsf{NMCom}}$ receives commitments to $(a_i^S, d_i^S)_{i \notin \Gamma_R}$, the internally executed experiment is identical with $H'_{k:5}$, whereas when $\mathcal{A}_{\mathsf{NMCom}}$ receives a commitments to $(0,0)_{i \notin \Gamma_R}$, the internally executed experiment is identical with $H_{k:5}$. Hence, from the assumption that $\mathcal{A}$ cheats in session $s(j)$ with negligible probability in $H'_{k:5}$ but with non-negligible probability in $H_{k:5}$, $\mathcal{A}_{\mathsf{NMCom}}$ breaks the non-malleability of NMCom. □

This completes the proof of Lemma 3.7.7. □

3.9.5 Proof of Lemma 3.7.8

Proof. Recall that hybrids $H_{k:5}, H_{k:6}$ differ only in the inputs and the randomness that are used in some of the mS-OTs in Stage 3, where those that are derived from the outcomes of the coin tossing is used in $H_{k:5}$ and random inputs and true randomness are used in $H_{k:6}$.

First, we show the indistinguishability. Assume for contradiction that $H_{k:5}$ and $H_{k:6}$ are computationally distinguishable. We build an efficient adversary $\mathcal{A}_{\mathsf{RobCom}}$ that breaks the hiding property of RobCom.

> The adversary $\mathcal{A}_{\mathsf{RobCom}}$ internally executes $H_{k:5}$ with the following modification: in Stage 0-1 of session $s(k)$, it picks two random strings $\psi^S = \psi_1^S \| \ldots, \psi_{11n}^S$ and $\tilde{\psi}^S = \tilde{\psi}_1^S \| \ldots, \tilde{\psi}_{11n}^S$ and sends $\{\psi_i^S\}_{i \notin \Gamma_R}$ and $\{\tilde{\psi}_i^S\}_{i \notin \Gamma_R}$ to the external committer and receives back RobCom commitments (in which either $\{\psi_i^S\}_{i \notin \Gamma_R}$ or $\{\tilde{\psi}_i^S\}_{i \notin \Gamma_R}$ are committed in parallel). Then in Stage 2-1 of session $s(k)$, $\mathcal{A}_{\mathsf{RobCom}}$ always use ψ_i^S's to mask a_i^S (i.e. $z_i^S := a_i^S \oplus \psi_i^S$ for all $i \in [11n]$). in the subsequent stages, $\mathcal{A}$ proceeds the experiment as in $H_{k:1}$. After the execution of $H_{k:1}$ finishes, $\mathcal{A}_{\mathsf{RobCom}}$ outputs whatever $\mathcal{Z}$ outputs in the experiment.

When $\mathcal{A}_{\mathsf{RobCom}}$ receives commitments to $\{\psi_i^S\}_{i \notin \Gamma_R}$, the internally executed experiment is identical with $H_{k:5}$, whereas when $\mathcal{A}_{\mathsf{RobCom}}$ receives commitments to $\{\tilde{\psi}_i^S\}_{i \notin \Gamma_R}$, the internally executed experiment is identical with $H_{k:6}$ (this is because when $\mathcal{A}_{\mathsf{RobCom}}$ receives commitments to $(\tilde{\psi}_i^S)_{i \notin \Gamma_R}$, the values $z_i^S = \psi_i^S \oplus a_i^S$ (thus the values $r_i^S = a_i^S \oplus b_i^S$) for each $i \notin \Gamma_R$ are uniformly random for $\mathcal{A}$. Hence the mS-OT for each $i \notin \Gamma_R$ is executed with a random input and true randomness). Hence, from the assumption that $H_{k:5}$ and $H_{k:6}$ are distinguishable, $\mathcal{A}_{\mathsf{RobCom}}$ distinguishes RobCom commitments.

We next show that in $H_{k:6}$, $\mathcal{A}$ does not cheat in sessions $s(k), \ldots, s(4m)$. Assume for contradiction that in $H_{k:6}$, $\mathcal{A}$ cheats in one of those sessions, say, session $s(j)$, with non-negligible probability. Then, from an average argument, we can fix the execution of the experiment up until SM_k (inclusive) in such a way that even after being fixed, $\mathcal{A}$ cheats in session $s(j)$ only with negligible probability in $H_{k:5}$ but with non-negligible probability in $H_{k:6}$. Then, we can break the robust non-malleability of NMCom as follows.

> The adversary $\mathcal{A}_{\mathsf{NMCom}}$, who interacts with a committer of RobCom and a receiver of NMCom, internally executes $H_{k:5}$ from SM_k using the non-uniform advice. In Stage 0-1 of session $s(k)$, $\mathcal{A}_{\mathsf{NMCom}}$ chooses random strings $\tilde{\psi}^S = \tilde{\psi}_1^S \| \ldots \| \tilde{\psi}_{11n}^S$ in addition to $\psi^S = \psi_1^S \| \ldots \| \psi_{11n}^S$, sends $\{\psi_i^S\}_{i \notin \Gamma_R}$ and $\{\tilde{\psi}_i^S\}_{i \notin \Gamma_R}$ to the external committer and receives back parallel RobCom commitments (in which either $\{\psi_i^S\}_{i \notin \Gamma_R}$ or $\{\tilde{\psi}_i^S\}_{i \notin \Gamma_R}$ are committed to), and feeds them into $H_{k:5}$. Then in Stage 2-1 of session $s(k)$, $\mathcal{A}_{\mathsf{NMCom}}$ always use ψ_i^S's to mask a_i^S (i.e. $z_i^S := a_i^S \oplus \psi_i^S$ for all $i \in [11n]$). Also, in session $s(j)$, $\mathcal{A}_{\mathsf{NMCom}}$ forwards the NMCom commitments from $\mathcal{A}$ to the external receiver. After the execution of $H_{k:5}$ finishes, $\mathcal{A}_{\mathsf{NMCom}}$ outputs its view.
>
> The distinguisher $\mathcal{D}_{\mathsf{NMCom}}$ takes as input the view of $\mathcal{A}_{\mathsf{NMCom}}$ and the values committed by $\mathcal{A}_{\mathsf{NMCom}}$ (which are equal to the values committed to by $\mathcal{A}$ in session $s(j)$ in the internally executed experiment). $\mathcal{D}_{\mathsf{NMCom}}$ then outputs 1 if and only if $\mathcal{A}$ cheated in session $s(j)$.

When $\mathcal{A}_{\mathsf{NMCom}}$ receives commitments to $\{\psi_i^S\}_{i \notin \Gamma_R}$, the internally executed experiment is identical with $H_{k:1}$, whereas when $\mathcal{A}_{\mathsf{Com}}$ receives commitments to $\{\tilde{\psi}_i^S\}_{i \notin \Gamma_R}$, the internally executed experiment is identical with $H_{k:6}$. Hence, from the assumption that $\mathcal{A}$ cheats in session $s(j)$ with negligible probability in $H_{k:5}$ but with non-negligible probability in $H_{k:6}$, $\mathcal{A}_{\mathsf{NMCom}}$ breaks the robust non-malleability of NMCom.

This completes the proof of Lemma 3.7.8. □

3.9.6 Proof of Claim 3.7.6

Proof. Recall that $H_{k:6}$ and $H'_{k:6}$ differ only in that in session $s(k)$ of $H'_{k:6}$, if R is corrupted and SM_k is fourth special message, $\beta_{b,i}$ is a random bit rather than $\beta_{b,i} = \rho_{b,i} \oplus s_{i,b\oplus\alpha_i}$ for every $b \in \{0,1\}$ and $i \in \Delta \setminus I_b$.

First, we show the indistinguishability between $H_{k:6}$ and $H'_{k:6}$. Roughly, we prove the indistinguishability using the security of mS-OT: For every $i \in \Delta \setminus I_b$, $\mathcal{A}$ executed the i-th mS-OT honestly with choice bit $(1-b) \oplus \alpha_i$, and the sender's input and randomness of this mS-OT are not revealed in Stage 8; therefore, the value of $s_{i,b\oplus\alpha_i}$ is hidden from $\mathcal{A}$ and thus $\beta_{b,i} = \rho_{b,i} \oplus s_{i,b\oplus\alpha_i}$ is indistinguishable from a random bit. Formally, we consider the following security game against cheating receiver R^* of mS-OT.

> The cheating receiver R^* gets random input-randomness pairs $(c_i, \tau_i^R)_i$ of mS-OT instances as input. R^* then participates in $9n$ instances of mS-OTs in parallel with an honest sender S, who uses a random input $(s_{i,0}, s_{i,1})$ in the i-th instance. After the execution with S, R^* receives bits $(s^*_{i,0}, s^*_{i,1})_i$ that are defined as follows: Let $b^* \in \{0,1\}$ be a randomly chosen bit; if $b^* = 0$, then for every i, $s^*_{i,0} \overset{\text{def}}{=} s_{i,0}$ and $s^*_{i,1} \overset{\text{def}}{=} s_{i,1}$; if $b^* = 1$, then for every i such that R^* behaved honestly in the i-th mS-OT using (c_i, τ_i^R) as input and randomness, $s^*_{i,c_i} \overset{\text{def}}{=} s_{i,c_i}$ but $s^*_{i,1-c_i}$ is a random bit, and for every other i, $s^*_{i,0} \overset{\text{def}}{=} s_{i,0}$ and $s^*_{i,1} \overset{\text{def}}{=} s_{i,1}$. Then, R^* guesses the value of b^*, and if the guess is correct, we say that R^* wins the game.

From the security of mS-OT against semi-honest receivers, any cheating R^* wins the game with probability at most $1/2 + \mathsf{negl}(n)$. Now, we assume for contradiction that $H_{k:6}$ and $H'_{k:6}$ are distinguishable, and we derive a contradiction by constructing an adversary who wins the above game with probability non-negligibly higher than $1/2$. From an average argument, we can fix the execution of the experiment up until SM_k (inclusive) in such a way that even after being fixed, $H_{k:6}$ and $H'_{k:6}$ are still distinguishable. Then, by considering the transcript up until SM_k and the table $\mathcal{T}$ as non-uniform advice, we can obtain an adversary who wins the above game with probability non-negligibly higher than $1/2$ as follows.

> The adversary R^* gets random input-randomness pairs $(c_i, \tau_i^R)_{i\in\Delta\setminus\Gamma_R}$ of mS-OT instances as its input, and internally executes $H'_{k:6}$ from SM_k using the non-uniform advice. In Stage 2-2, R^* chooses $\boldsymbol{b}^R = (b_1^R, \dots, b_{11n}^R)$ in such a way that $\boldsymbol{r}^R = (r_1^R, \dots, r_{11n}^R)$ satisfies $r_i^R = c_i \,\|\, \tau_i^R$ for every $i \in \Delta \setminus \Gamma_R$, namely, chooses $\boldsymbol{b}^R$ such that $b_i^R = a_i^R \oplus (c_i \,\|\, \tau_i^R)$ for every $i \in \Delta \setminus \Gamma_R$. (Recall that in $H'_{k:6}$, the subset Γ_R and the strings $\boldsymbol{a}^R = (a_1^R, \dots, a_{11n}^R)$ are extracted by brute force and they are included in the non-uniform advice.) In Stage 3 of session $s(k)$, $\mathcal{A}_{\mathsf{NMCom}}$ obtains the i-th mS-OT from the external sender for every $i \in \Delta \setminus \Gamma_R$ and executes other instances of mS-OT by itself. Then, in Stage 6 of session $s(k)$, R^* receives bits $(s^*_{i,0}, s^*_{i,1})_{i\in\Delta\setminus\Gamma_R}$ from the external sender and uses them to compute $\beta_{b,i}$ for every

$i \in \Delta \setminus \Gamma_R$, i.e. , $\beta_{b,i} := \rho_{b,i} \oplus s^*_{i,b\oplus\alpha_i}$. After the execution of $H'_{k:6}$ finishes, R^* outputs whatever $\mathcal{Z}$ outputs in the experiment.

When $b^* = 0$ in the security game (and hence $s^*_{i,b\oplus\alpha_i} = s_{i,b\oplus\alpha_i}$ for every i and b), the internally executed experiment is identical with $H_{k:6}$, whereas when $b^* = 1$ (and hence $s^*_{i,b\oplus\alpha_i}$ is a random bit if $i \in \Delta \setminus I_b$ and $s^*_{i,b\oplus\alpha_i} = s_{i,b\oplus\alpha_i}$ otherwise), the internally executed experiment is identical with $H'_{k:6}$. Hence, from the assumption that $H_{k:6}$ and $H'_{k:6}$ are distinguishable, R^* wins the game with probability non-negligibly higher than $1/2$.

Next, we show that in $H'_{k:6}$, $\mathcal{A}$ does not cheat in sessions $s(k), \ldots, s(4m)$. (The argument below is similar to the one in the proof of Lemma 3.7.3.) Assume for contradiction that in $H'_{k:6}$, $\mathcal{A}$ cheats in one of those sessions, say, session $s(j)$, with non-negligible probability. Then, from an average argument, we can fix the execution of the experiment up until SM_k (inclusive) in such a way that after being fixed, $\mathcal{A}$ cheats in session $s(j)$ only with negligible probability in $H_{k:6}$ but with non-negligible probability in $H'_{k:6}$. Then, by considering the transcript up until SM_k and the table $\mathcal{T}$ as non-uniform advice, we can break the robust non-malleability of NMCom as follows.

The adversary $\mathcal{A}_{\mathsf{NMCom}}$, who participates in the above game while interacting with a receiver of NMCom, gets random input-randomness pairs $(c_i, \tau^R_i)_{i\in\Delta\setminus\Gamma_R}$ of mS-OT instances as its input, and internally executes $H'_{k:6}$ from SM_k using the non-uniform advice. In Stage 2-2, $\mathcal{A}_{\mathsf{NMCom}}$ chooses $\boldsymbol{b}^R = (b^R_1, \ldots, b^R_{11n})$ in such a way that $\boldsymbol{r}^R = (r^R_1, \ldots, r^R_{11n})$ satisfies $r^R_i = c_i \parallel \tau^R_i$ for every $i \in \Delta \setminus \Gamma_R$, namely, chooses $\boldsymbol{b}^R$ such that $b^R_i = a^R_i \oplus (c_i \parallel \tau^R_i)$ for every $i \in \Delta \setminus \Gamma_R$. In Stage 3 of session $s(k)$, $\mathcal{A}_{\mathsf{NMCom}}$ obtains the i-th mS-OT from the external sender for every $i \in \Delta \setminus \Gamma_R$ and executes other instances of mS-OT by itself. Then, in Stage 6 of session $s(k)$, $\mathcal{A}_{\mathsf{NMCom}}$ receives bits $(s^*_{i,0}, s^*_{i,1})_{i\in\Delta\setminus\Gamma_R}$ from the external sender and uses them to compute $\beta_{b,i}$ for every $i \in \Delta \setminus \Gamma_R$, i.e. , $\beta_{b,i} := \rho_{b,i} \oplus s^*_{i,b\oplus\alpha_i}$. Also, in session $s(j)$, $\mathcal{A}_{\mathsf{NMCom}}$ forwards the NMCom commitments from $\mathcal{A}$ to the external receiver. After the execution of $H'_{k:6}$ finishes, $\mathcal{A}_{\mathsf{NMCom}}$ outputs its view.

The distinguisher $\mathcal{D}_{\mathsf{NMCom}}$ takes as input the view of $\mathcal{A}_{\mathsf{NMCom}}$ and the values committed by $\mathcal{A}_{\mathsf{NMCom}}$ (which are equal to the values committed to by $\mathcal{A}$ in session $s(j)$ in the internally executed experiment). $\mathcal{D}_{\mathsf{NMCom}}$ then outputs 1 if and only if $\mathcal{A}$ cheated in session $s(j)$.

When $b^* = 0$ in the security game (and hence $s^*_{i,b\oplus\alpha_i} = s_{i,b\oplus\alpha_i}$ for every i and b), the internally executed experiment is identical with $H_{k:6}$, whereas when $b^* = 1$ (and hence $s^*_{i,b\oplus\alpha_i}$ is a random bit if $i \in \Delta \setminus I_b$ and $s^*_{i,b\oplus\alpha_i} = s_{i,b\oplus\alpha_i}$ otherwise), the internally executed experiment is identical with $H'_{k:6}$. Hence, from the assumption that $\mathcal{A}$ cheats in session $s(j)$ with negligible probability in $H_{k:6}$ but with non-negligible probability in $H'_{k:6}$, $\mathcal{A}_{\mathsf{NMCom}}$ breaks the robust non-malleability of NMCom.

This completes the proof. $\square$

Chapter 4

Black-Box Angle-Based UC MPC in $\widetilde{O}(\log \lambda)$ Rounds

4.1 Overview of Our Techniques

4.1.1 Existing Approaches

Let us briefly review the current approaches for constructing CCA secure commitments. The main difficulty in constructing CCA secure commitments under polynomial hardness is to move from the real world—which contains the exponential time decommitment oracle $\mathcal{O}$—to a hybrid where $\mathcal{O}$'s responses can be efficiently simulated. A standard way to do this is to use a proof-of-knowledge (PoK): the protocol should require the (man-in-the-middle) adversary, say $\mathcal{A}$, to give a PoK of the value it commits. The main difficulty in employing this is that $\mathcal{A}$ may open concurrently many sessions with $\mathcal{O}$ (referred here to as "right" side sessions), interleaved in an arbitrary manner; furthermore, these values have to be extracted *immediately* within each session irrespective of what happens in other sessions. This is precisely the issue in constructing (black-box simulatable) concurrent zero-knowledge (CZK) protocols [DNS98] as well, and ideas from there are applied in this setting too. A second difficulty is that these extractions must happen without rewinding the commitment $\mathcal{A}$ *receives* (referred to as "left" side session).

It is worthwhile to quickly recall the (tag-based) non-malleable commitment construction in the original work of [DDN91]. In this construction, $\mathcal{A}$ has only one right session; to prove that the value on the right is (computationally) independent from that on the left, the value on the right is extracted without rewinding the sensitive parts of the left side commitments. This is done by creating two types of PoK—one each for two possible values of a bit. These PoK create rewinding "slots" for extraction such that if $\mathcal{A}$ uses a different bit in the tag, it risks the possibility of having to perform a PoK on its own—i.e., without any "dangerous" rewinding on the left—in one of the slots (called a "free" slot). These special PoK are performed for each bit of the tag *sequentially* so that at least one free slot is guaranteed since the left and right tags are different by definition. While this requires λ rounds for λ-bit tags, it is possible to split the tag into λ smaller tags of $\log \lambda$ bits and run the protocol for each of them in parallel [DDN91, LPV08]. Referred to as "LOG trick," this yields a $O(\log \lambda)$-round protocol.

The key idea for CCA commitments in [CLP10], at a high level, is to ensure that in the *concurrent* setting, many free slots exist for each session so that extraction succeeds before the end of that session. This is achieved by creating a polynomial-round protocol consisting of sequential repetition of special PoK as above and then relying on an analysis that is, at a high level, similar to early rewinding techniques from CZK literature [RK99, CGGM00]. Once the issue of concurrent extraction is handled, the additional ideas in [LP12] are (again, at a high level) to enforce this approach using cut-and-choose protocols to obtain a black-box construction. The work of Goyal et al. [GLP$^+$15] shows how to separate the tasks of "concurrent extraction" and "non-malleability" in this approach by proving a "robust extraction lemma". This allows them to follow a structure similar to that of concurrent non-malleable zero-knowledge (CNMZK) from [BPS06] which matches the round complexity of CZK, i.e., $\widetilde{O}(\log\lambda)$. However, their approach requires non-black usage of one-way functions. Kiyoshima [Kiy14] shows that the robust-extraction lemma can actually be applied to the previous black-box protocol of [LP12] to get $\widetilde{O}(k\cdot\log\lambda)$ rounds if one has a slightly stronger primitive than non-malleable commitments: namely k-round 1-1 CCA commitments. To build such commitments, Kiyoshima builds non-malleability "from scratch" by combining the DDN "LOG trick" with cut-and-choose components of [LP12] so that the extraction on right in the standalone setting, can be done without any dangerous rewinding on left. This however results in $O(\log\lambda)$ rounds for 1-1 CCA and $\widetilde{O}(\log^2\lambda)$ for full CCA.

4.1.2 Our Approach

We significantly deviate from current approaches for constructing 1-1 CCA commitments. Instead of attempting to build non-malleability *from scratch*, our goal is to have a generic construction built around existing non-malleable commitments. The resulting protocol will not only have a simpler and more modular proof of security, but will also benefit from the efficiency and assumptions of the underlying non-malleable commitment (NMCom). Towards this goal, we return to investigate the structure of CNMZK protocols even for the simpler case of 1-1 CCA.

Setting aside the issue of round-complexity for the moment, a key idea in the construction of CNMZK protocols [BPS06, LPTV10, OPV10, LP11] is to have the prover give a non-malleable commitment (NMCom) which can later be switched to a "trapdoor value" set by the verifier; the non-malleability of NMCom ensures that $\mathcal{A}$ cannot switch his value to a trapdoor on the right (unless he did so in the real world, which can be shown impossible through other means). The prover later proves that either the statement is true or it committed the trapdoor. The main problem with this approach is that it requires us to prove a predicate over the value committed in NMCom which requires non-black-box use of cryptographic primitives.

Non-Malleable Commit-and-Prove. One potential idea to avoid non-black-box techniques is to turn to black-box commit-and-prove protocols in the literature and try to re-develop them in the context of non-malleability. Commit-and-prove protocols allow a committer to commit to a value v so that later, it can prove a predicate ϕ over the committed value in zero-knowledge. These protocols can be constructed in constant rounds using the powerful "MPC-in-the-head" approach introduced by Ishai et al. [IKOS07]. The approach allows committing multiple values $v_1, \dots, v_n$ and then proving a joint predicate ϕ over them. One such construction is implicit in the work of Goyal et al. [GLOV12]. Such commitments were also used extensively by Goyal et al. to build size-hiding commit-and-prove [GOSV14] and an optimal four round construction was obtained by Khurana, Ostrovsky, and Srinivasan [KOS18]. As noted above, if we can develop an appropriate non-malleable version of such protocols, it is conceivable that they can yield constant-round 1-1 CCA commitment. Note however that non-malleable commitments are not usually equipped to handle proofs. Therefore, such an approach will necessarily have to "open up" the construction of non-malleable commitments. In particular, like previous constructions, this approach cannot be based on non-malleable commitments in a black-box manner.

Changing the Direction of NMCom. In order to rely on non-malleable commitments directly, it is essential that we do not prove anything about the values committed inside the NMCom. Instead, we should restrict all proofs to be performed only over standard commitments since for them we can use standard black-box commit-and-prove protocols. Towards building this property, what if we change the direction of NMCom and ask the receiver of 1-1 CCA to send non-malleable commitments, which, for example, can be opened later? More specifically, in our 1-1 CCA protocol, the receiver will send a NMCom to a random value σ which it will open subsequently. The committer will send a "trapdoor commitment" t before it sees σ opened. Later, the committer will commit to the desired value v and give a PoK that either it knows v or t is a commitment to σ (the "trapdoor"). Observe that this structure completely avoids any proof directly over non-malleable commitments; all proofs only need to be performed over ordinary commitments. Therefore, if we use the commit phase of black-box commit-and-prove protocols to commit to σ and v we can easily complete the PoK in a black-box manner: the predicate ϕ in the proof phase will simply test for the presence of trapdoor σ. Some standard soundness issues arise in this approach but they can be handled by ensuring that the commit phase is extractable.

Although this approach yields a black-box construction directly from NMCom, it is hard to prove the 1-1 CCA property. At a high level, this is because of the following: if in the 1-1 CCA game, $\mathcal{A}$ schedules the completion of the left NMCom *before* the right one[1], the simulator in the security proof must extract σ from this NMCom while the right NMCom is still in play (so that it can generate t to be a commitment to σ). This involves rewinding the left NMCom (assuming it is extractable) which in turn rewinds the right session.[2] A similar issue arises in the work of Jain and Pandey [JP14] on black-box non-malleable zero-knowledge where it is resolved by using a NMCom that is already 1-1 CCA secure. We do not have this flexibility in our setting.

A possible fix for this issue is to rely on some kind of "delayed input" property: i.e., the commitment to t will be an extractable commitment that does not require the message m to be committed until the last round. This property can be obtained by committing to a key k in an extractable manner and then in the last round committing to m by simply encrypting with k. This however will no longer be compatible with the black-box commit-and-prove

[1]Note that NMCom's direction is opposite to that of 1-1 CCA: the receiver of 1-1 CCA is the sender of right NMCom.

[2]This is not an issue in the synchronous schedule since in that case, the value $\mathcal{A}$ commits to in NMCom is provided to the distinguisher along with the joint view.

strategy since we will now have to take encryption into account.

We overcome this issue by making extensive use of extractable commitments. More specifically, we first *prepend* the NMCom with a standard "slot-based" extractable commitment which commits to the same value σ as the NMCom. If the NMCom also has a slot like extractable structure (e.g., the *three round* scheme of [GPR16]), we can argue that non-synchronous adversaries must always leave a free slot either on top or at the bottom of NMCom. For example, in the troublesome scheduling discussed above, $\mathcal{A}$ can be easily rewound in the last two messages of NMCom (if we use [GPR16]) *without* rewinding the right NMCom. In other non-synchronous schedules it will have a free slot in the top extractable commitment on the left. On the other hand, synchronous adversaries will fail in the NMCom step (and synchronous non-malleability suffices for our purposes). In summary, this will suffice for us to show that even if our simulator sets up the trapdoor statement on the left (by committing σ in t), $\mathcal{A}$ cannot do the same on the right. Other NMCom, particularly public-coin extractable NMCom also seem sufficient.

A second issue here is the intertwining of the left PoK[3] with "extractable" components on the right, e.g., the right PoK (or extractable commitment steps). In order to prove that $\mathcal{A}$ cannot setup the trapdoor, extraction from right PoK will be necessary in the proof and this will be troublesome when changing the witness in the left PoK during hybrids. This issue can be handled using the sequential repetition technique from [LP09]: we use $k+1$ PoK where k is the (constant) rounds in a single PoK. It is worthwhile to note that other common methods for handling this issue do not work: e.g., we cannot rely on *statistical* WI since it requires stronger assumptions for constant rounds; we also cannot use proofs that are secure against a fixed number of rewinds since they usually allow only a noticeable probability of extraction which is insufficient for a 1-1 CCA commitment, where extraction must succeed with overwhelming probability.

4.2 Preliminaries

In the following, we present additional preliminaries that are necessary for this chapter.

4.2.1 CCA Commitments

We define the notion of CCA-secure commitments (and 1-1 CCA security in particular). These definitions rely on the notion of a *decommitment oracle*, which provide decommitments given valid transcripts to a particular (tag based) commitment protocol. Specifically, a decommitment oracle $\mathcal{O}$ for a given commitment protocol acts as follows:

- $\mathcal{O}$ acts as an honest reciever against some committer C, participating faithfully according to the specified commitment scheme. C is allowed to pick a tag for this interaction adaptively.
- At the end of this interaction, if the honest reciever were to accept the transcript as containing a valid commitment with respect to the given tag, $\mathcal{O}$ returns the value v

[3] Observe that the PoK will just be the proof part of appropriate black-box commit-and-prove with right parameters to ensure black-box property; they will also satisfy witness-indistinguishability [FS90].

committed by C to it. Otherwise, it returns $\perp$.

We denote an adversary with access to the decommitent oracle as $\mathcal{A}^{\mathcal{O}}$. CCA security then essentially constitutes preservation of the hiding property even against adversaries enjoying such oracle access. More formally, we define the following game $\mathsf{IND}_b(\langle C, R\rangle, \mathcal{A}, \mathcal{O}, \lambda, z)$ ($b \in \{0,1\}$) as follows: given the public parameter 1^λ and auxiliary input z, the adversary $\mathcal{A}^{\mathcal{O}}$ adaptively generates two challenge values v_0, v_1 of length λ, and a tag $\mathsf{tag} \in \{0,1\}^\lambda$. Then, $\mathcal{A}^{\mathcal{O}}$ receives a commitment to v_b with tag tag from the challenger. Let y be the output of $\mathcal{A}$ in this game. The output of the game is $\perp$ if during the game, $\mathcal{A}$ sends $\mathcal{O}$ any commitment using tag tag. Otherwise, the output of the game is y. We abuse notation to denote the output of the game $\mathsf{IND}_b(\langle C, R\rangle, \mathcal{A}, \mathcal{O}, \lambda, z)$ by the same symbol $\mathsf{IND}_b(\langle C, R\rangle, \mathcal{A}, \mathcal{O}, \lambda, z)$.

Definition 4.2.1 (CCA Commitment)**.** *Let $\langle C, R\rangle$ be a tag-based commitment scheme, and $\mathcal{O}$ be an associated decommitment oracle. Then $\langle C, R\rangle$ is said to be* ***CCA secure w.r.t.*** *$\mathcal{O}$, if for every nonuniform* PPT *machine $\mathcal{A}$, the following ensembles are computationally indistinguishable:*

– $\{\mathsf{IND}_0(\langle C, R\rangle, \mathcal{A}, \mathcal{O}, \lambda, z)\}_{\lambda\in\mathbb{N}, z\in\{0,1\}^*}$

– $\{\mathsf{IND}_1(\langle C, R\rangle, \mathcal{A}, \mathcal{O}, \lambda, z)\}_{\lambda\in\mathbb{N}, z\in\{0,1\}^*}$

It is customary to call any commitment scheme that is CCA secure with respect to some decommitment oracle as just CCA secure (but in general the oracle is usually also described, and is of course necessary to prove such security). It is also customary to call the interaction between the challenger and adversary as the *left* interaction, and that between adversary and oracle as the *right* interaction, in the fashion of non-malleable commitments, where the security property chiefly considers man in the middle attacks.

1-1 CCA. A scheme is *1-1 CCA secure* (denoted as $\mathrm{CCA}^{1:1}$) if the corresponding adversary is only allowed one interaction with the oracle.

4.2.2 Angel-Based Universally Composable (or UC-SPS) MPC

4.2.2.1 UC Security

We first briefly review UC security. For full details see [Can00b]. A large part of this section has been taken verbatim from [CLP10, GGJS12, Lin12]. We first review the model of computation, ideal protocols, and the general definition of securely realizing an ideal functionality. Next we present hybrid protocols and the composition theorem.

The Basic Model of Execution. At a high level, UC security is defined following the similar ideal/real paradigm as in Section 2.5 with the following major differences:

– instead of a single execution, the n parties are executing several instances of the same protocol simultaneously.

– There is an *environment* that prepares inputs to all the parties, corrupts a subset of parties, and interacts with the corrupted parties *during the execution.*

Following [GMR89, Gol01], a protocol is represented as an interactive Turing machine (ITM), which represents the program to be run within each participant. Specifically, an ITM has three tapes that can be written to by other ITMs: the input and subroutine tapes model the inputs from and the outputs to other programs running within the same "entity" (say, the same physical computer), and the incoming communication tapes and outgoing communication tapes model messages received from and to be sent to the network. It also has an identity tape that cannot be written to by the ITM itself. The identity tape contains the program of the ITM (in some standard encoding) plus additional identifying information specified below.Adversarial entities are also modeled as ITMs.

We distinguish between ITMs (which represent static objects, or programs) and *instances of ITMs*, or ITIs, that represent interacting processes in a running system. Specifically, an ITI is an ITM along with an identifier that distinguishes it from other ITIs in the same system. The identifier consists of two parts: a session-identifier (SID) which identifies which protocol instance the ITM belongs to, and a party identifier (PID) that distinguishes among the parties in a protocol instance. Typically the PID is also used to associate ITIs with "parties", or clusters, that represent some administrative domains or physical computers.

The model of computation consists of a number of ITIs that can write on each other's tapes in certain ways (specified in the model). The pair (SID,PID) is a unique identifier of the ITI in the system.

With one exception (discussed within) we assume that all ITMs are PPT. An ITM is PPT if there exists a constant $c > 0$ such that, at any point during its run, the overall number of steps taken by M is at most n^c, where n is the overall number of bits written on the input tape of M in this run. (In fact, in order to guarantee that the overall protocol execution process is bounded by a polynomial, we define n as the total number of bits written to the input tape of M, minus the overall number of bits written by M to input tapes of other ITMs (see [Can01]).

The Model for Protocol Execution. The model of computation consists of the parties running an instance of a protocol Π, an adversary $\mathcal{A}$ that controls the communication among the parties, and an *environment* $\mathcal{Z}$ that controls the inputs to the parties and sees their outputs. We assume that all parties have a security parameter $\lambda \in \mathbb{N}$. (We remark that this is done merely for convenience and is not essential for the model to make sense). The execution consists of a sequence of *activations*, where in each activation a single participant (either $\mathcal{Z}$, $\mathcal{A}$, or some other ITM) is activated, and may write on a tape of at most one other participant, subject to the rules below. Once the activation of a participant is complete (i.e., once it enters a special waiting state), the participant whose tape was written on is activated next. (If no such party exists then the environment is activated next.)

The environment is given an external input z and is the first to be activated. In its first activation, the environment invokes the adversary $\mathcal{A}$, providing it with some arbitrary input. In the context of UC security, the environment can from now on invoke (namely, provide input to) only ITMs that consist of a single instance of protocol Π. That is, all the ITMs invoked by the environment must have the same SID and the code of Π.

Once the adversary is activated, it may read its own tapes and the outgoing communication tapes of all parties. It may either deliver a message to some party by writing this message on the party's incoming communication tape or report information to $\mathcal{Z}$ by writing this information on the subroutine output tape of $\mathcal{Z}$. For simplicity of exposition, in the rest of this paper we assume authenticated communication; that is, the adversary may deliver only messages that were actually sent. (This is however not essential as shown in [Can04, BCL+05].)

Once a protocol party (i.e., an ITI running Π) is activated, either due to an input given by the environment or due to a message delivered by the adversary, it follows its code and possibly writes a local output on the subroutine output tape of the environment, or an outgoing message on the adversary's incoming communication tape.

The protocol execution ends when the environment halts. The output of the protocol execution is the output of the environment. Without loss of generality we assume that this output consists of only a single bit.

Let $\mathrm{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}(\lambda, z, r)$ denote the output of the environment $\mathcal{Z}$ when interacting with parties running protocol Π on security parameter λ, input z, and random input

$$r = r_{\mathcal{Z}}, r_{\mathcal{A}}, r_1, r_2, \ldots, r_n$$

as described above (z and $r_{\mathcal{Z}}$ for $\mathcal{Z}$, $r_{\mathcal{A}}$ for $\mathcal{A}$, and r_i for party P_i). Let $\mathrm{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}(\lambda, z)$ random variable describing $\mathrm{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}(\lambda, z, r)$ where r is uniformly chosen. Let $\mathrm{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$ denote the ensemble $\{\mathrm{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}(\lambda, z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^*}$.

Ideal functionalities and ideal protocols. Security of protocols is defined via comparing the protocol execution to an ideal protocol for carrying out the task at hand. A key ingredient in the ideal protocol is the ideal functionality that captures the desired functionality, or the specification, of that task. The ideal functionality is modeled as another ITM (representing a "trusted party") that interacts with the parties and the adversary. More specifically, in the ideal protocol for functionality $\mathcal{F}$ all parties simply hand their inputs to an ITI running $\mathcal{F}$. (We will simply call this ITI $\mathcal{F}$. The SID ofFis the same as the SID of the ITIs running the ideal protocol. The PID of $\mathcal{F}$ is null.) In addition, $\mathcal{F}$ can interact with the adversary according to its code. Whenever $\mathcal{F}$ outputs a value to a party, the party immediately copies this value to its own output tape. We call the parties in the ideal protocol dummy parties. Let $\Pi_{\mathcal{F}}$ denote the ideal protocol for functionality $\mathcal{F}$.

Securely realizing an ideal functionality. We say that a protocol Π emulates protocol ϕ if for any adversary $\mathcal{A}$ there exists an adversary Sim such that no environment $\mathcal{Z}$, on any input, can tell with non-negligible probability whether it is interacting withAand parties running Π, or it is interacting with Sim and parties running ϕ. This means that, from the point of view of the environment, running protocol Π is *just as good* as interacting with ϕ. We say that Π securely realizes an ideal functionality $\mathcal{F}$ if it emulates the ideal protocol $\Pi_{\mathcal{F}}$. More precise definitions follow. A distribution ensemble is called *binary* if it consists of distributions over $\{0, 1\}$.

Definition 4.2.2. *Let* Π *and* ϕ *be protocols. We say that* UC-emulates ϕ *if for any adversary* $\mathcal{A}$*, there exists an adversary* Sim *such that for any environment* $\mathcal{Z}$ *that obeys the rules of interaction for UC security we have* $\mathrm{EXEC}_{\phi,\mathsf{Sim},\mathcal{Z}} \overset{c}{\approx} \mathrm{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$.

Definition 4.2.3. *Let* $\mathcal{F}$ *be an ideal functionality, and let* Π *be a protocol. We say that* Π UC-realizes $\mathcal{F}$ *if* Π *UC-emulates the ideal process* $\Pi_{\mathcal{F}}$.

Hybrid protocols. Hybrid protocols are protocols where, in addition to communicating as usual as in the standard model of execution, the parties also have access to (multiple copies of) an ideal functionality. Hybrid protocols represent protocols that use idealizations of underlying primitives,or alternatively make trust assumptions on the underlying network. They are also instrumental instating the universal composition theorem. Specifically, in an $\mathcal{F}$-hybrid protocol (i.e., in a hybrid protocol with access to an ideal functionality $\mathcal{F}$), the parties may give inputs to and receive outputs from an unbounded number of copies of $\mathcal{F}$.

The communication between the parties and each one of the copies of $\mathcal{F}$ mimics the ideal process. That is, giving input to a copy of $\mathcal{F}$ is done by writing the input value on the input tape of that copy. Similarly, each copy of $\mathcal{F}$ writes the output values to the subroutine output tape of the corresponding party. It is stressed that the adversary does not see the interaction between the copies of $\mathcal{F}$ and the honest parties.

The copies of $\mathcal{F}$ are differentiated using their SIDs. All inputs to each copy and all outputs from each copy carry the corresponding SID. The model does not specify how the SIDs are generated, nor does it specify how parties "agree" on the SID of a certain protocol copy that is to be run by them. These tasks are left to the protocol. This convention seems to simplify formulating ideal functionalities, and designing protocols that securely realize them, by freeing the functionality from the need to choose the SIDs and guarantee their uniqueness. In addition, it seems to reflect common practice of protocol design in existing networks.

The definition of a protocol securely realizing an ideal functionality is extended to hybrid protocols in the natural way.

The Universal Composition Operation. We define the universal composition operation and state the universal composition theorem. Let ρ be an $\mathcal{F}$-hybrid protocol, and let Π be a protocol that securely realizes $\mathcal{F}$. The composed protocol ρ^{Π} is constructed by modifying the code of each ITM in ρ so that the first message sent to each copy of $\mathcal{F}$ is replaced with an invocation of a new copy of Π with fresh random input, with the same SID, and with the contents of that message as input. Each subsequent message to that copy of $\mathcal{F}$ is replaced with an activation of the corresponding copy of Π, with the contents of that message given to Π as new input. Each output value generated by a copy of Π is treated as a message received from the corresponding copy of $\mathcal{F}$. The copy of Π will start sending and receiving messages as specified in its code. Notice that if Π is a $\mathcal{G}$-hybrid protocol (i.e., ρ uses ideal evaluation calls to some functionality $\mathcal{G}$) then so is ρ^{Π}.

The Universal Composition Theorem. Let $\mathcal{F}$ be an ideal functionality. In its general form, the composition theorem basically says that if Π is a protocol that UC-realizes $\mathcal{F}$ then, for any $\mathcal{F}$-hybrid protocol ρ, we have that an execution of the composed protocol ρ^{Π} "emulates" an execution of protocol ρ. That is, for any adversary $\mathcal{A}$ there exists a simulator Sim such that no environment machine $\mathcal{Z}$ can tell with non-negligible probability whether it is interacting with $\mathcal{A}$ and protocol ρ^{Π} or with Sim and protocol ρ, in a UC interaction. As a corollary, we get that if protocol ρ UC-realizes $\mathcal{F}$, then so does protocol ρ^{Π}.[4]

[4]The universal composition theorem in [Can01] applies only to "subroutine respecting protocols", namely protocols that do not share subroutines with any other protocol in the system.

Theorem 4.2.1 (Universal Composition [Can01])**.** *Let $\mathcal{F}$ be an ideal functionality. Let ρ be a $\mathcal{F}$-hybrid protocol, and let Π be a protocol that UC-realizes $\mathcal{F}$. Then protocol ρ^{Π} UC-emulates ρ.*

An immediate corollary of this theorem is that if the protocol ρ UC-realizes some functionality $\mathcal{G}$, then so does ρ^{Π}.

4.2.2.2 UC Security with Super-Polynomial Helpers (aka Angles)

Angle-based UC security, or UC with super-polynomial-helpers (SPS-UC), is obtained by modifying the definitions of UC security, by giving the corrupted parties access to an external "helper" entity, in a conceptually similar way to [PS04]. This entity, denoted $\mathcal{H}$, is computationally unbounded, and can be thought of as providing the corrupted parties with some judicious help. (As we will see, this help will be used to assist the simulator to "reverse engineering" the adversary in order to extract relevant information hidden in its communication.)

The definition uses the formalism of extended-UC (EUC) security [CDPW07]. Specifically, the helper entity is modeled as an ITM that is invoked directly by the environment, and that interacts with the environment and the corrupted parties. More formally, let $\mathcal{H}$ be an ITM. An environment $\mathcal{Z}$ is called aided by $\mathcal{H}$ if: (a) $\mathcal{Z}$ invokes a single instance $\mathcal{H}$ immediately after invoking the adversary; (b) As soon as a party (i.e., an ITI) P is corrupted (i.e., P receives a corrupted message), $\mathcal{Z}$ lets $\mathcal{H}$ know of this fact; (c) $\mathcal{H}$ interacts only with the corrupted parties. Then:

Definition 4.2.4 (SPS-UC)**.** *Let π and ϕ be protocols, and let $\mathcal{H}$ be a helper functionality (i.e., an ITM). We say that π $\mathcal{H}$-EUC-emulates ϕ if for any adversary $\mathcal{A}$ there exists an adversary* Sim *such that for any environment $\mathcal{Z}$ aided by $\mathcal{H}$, we have* $\mathrm{EXEC}_{\phi,\mathsf{Sim},\mathcal{Z}} \overset{c}{\approx} \mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$.

The meaningfulness of the above relativizing UC security of course depends on the particular helper ITM in use. Still, it is easy to see that if protocol π $\mathcal{H}$-EUC-emulates protocol ϕ where $\mathcal{H}$ obeys the above rules and runs in time $T(\lambda)$, then π UC-emulates ϕ according to a relaxed notion where the adversary Sim can run in time $\mathsf{poly}(T(\lambda))$. As noted in the past, for many protocols and ideal functionalities, this relaxed notion of security suffices even when $T(\lambda) = e^{\lambda}$ [Pas03, PS04, BS05, MMY06].

Universal Composition with Super-Polynomial Helpers. The universal composition theorem generalizes naturally to the case of EUC, even with super-polynomial helper functionalities:

Theorem 4.2.2 (SPS Universal Composition)**.** *Let $\mathcal{F}$ be an ideal functionality, let $\mathcal{H}$ be a helper functionality, let π be an $\mathcal{F}$-hybrid protocol, and let ρ be a protocol that $\mathcal{H}$-EUC-realizes $\mathcal{F}$. Then protocol π^{ρ} $\mathcal{H}$-EUC-emulates π.*

4.3 A New $\mathrm{CCA}^{1:1}$ Commitment Scheme

We will require the following ingredients for our $\mathrm{CCA}^{1:1}$ protocol:

- A statistically-binding commitment Com. In particular, we use Naor's construction [Nao90].
- A 3-round slot-based extractable commitment scheme ExtCom; for concreteness we will use the standard 3-round scheme (shown in Protocol 2.3.1) based on Naor's commitment (the first message ρ of Naor's commitment is not counted in rounds and assumed to be available from other parts of the protocol).
- An (extractable) commitment scheme ENMC that is *non-malleable* against *synchronizing* adversaries. We will need this protocol to be "compatible with slots" of the ExtCom defined above. For concreteness, we assume that ENMC is the 3-round commitment scheme of [GPR16] which satisfies all our requirements.
- A k round witness indistinguishable argument of knowledge WIAoK.

We stress that all of these ingredients have constant rounds, and can be constructed from standard OWFs in a black-box manner.

Our Protocol. We now describe our first protocol for $\mathrm{CCA}^{1:1}$ commitments. This protocol does not specifically try to achieve the black-box usage of cryptographic primitives. This allows us to focus on proving CCA security. However, it achieves two important properties: it is based on minimal assumptions, and it has a constant number of rounds. Moreover, the structure of this protocol is chosen in such a way that later, it will be possible to convert into a fully black-box construction. We remark that we also directly use identities of length λ directly (this is in keeping with the [GPR16] construction which does the same).

The formal description of the protocol appears in Protocol 4.3.1. At a high level, the protocol proceeds as follows. First, it requires the receiver to commit to a trapdoor string α using two extractable primitives: ExtCom as well as ENMC. Next, the committer will commit to an all zero-string β using ExtCom. Jumping ahead, in the security proof a "simulator machine" on left will set $\beta = \alpha$ and use it as a "fake witness" in a WIAoK; later we shall instantiate ExtCom with, roughly speaking, a "black-box commit-and-prove" to obtain a black-box construction. The receiver simply opens α in the next step, and the committer commits to the desired value, say v, followed by a proof of knowledge of v or that $\beta = \alpha$. A crucial observation here is that *proofs are not required to deal with values inside* ENMC—by ensuring that ENMC values opened in the protocol execution.

Protocol 4.3.1: $\mathrm{CCA}^{1:1}$ **Commitment Scheme** $\langle C, R\rangle_{\mathsf{CCA}}$

We let $\lambda \in \mathbb{N}$ denote the security parameter. All primitives used in the protocol by default have 1^λ as part of their input. We omit this detail in the following. Further, we assume that the execution involves a tag or identity $\mathsf{id} \in \{0,1\}^\lambda$.

Input: The committer C and reciever R have common input as the security parameter 1^λ. Additionally, C has as private input a value v which it wishes to commit to.

Commitment Phase. This proceeds as follows:

- **Stage 0:** C commits to the value v using Com, and sends this along with the identity id to R.
- **Stage 1:** This consists of the following steps:

(a) R picks a value $\alpha \xleftarrow{\$} \{0,1\}^\lambda$.
(b) R commits to $\alpha_1 = \alpha$ using ExtCom.

– **Stage 2:** R commits to $\alpha_2 = \alpha$ using ENMC, using identity id.

  For future reference, we denote by $\mathsf{CombinedCom}$ the joint execution of Stage 1 and 2 up to this point. Observe that $\mathsf{CombinedCom}$ is a statistically binding commitment scheme.

– **Stage 3:** C now commits to $\beta = 0^\lambda$ using ExtCom.

– **Stage 4:** This goes as follows:

  1. R decommits to both its commitments so far, revealing α_1 and α_2.
  2. C checks these decommitments, aborting if $\alpha_1 \neq \alpha_2$.

– **Stage 5:** C and R engage in $k+1$ WIAoK protocols *sequentially*. We denote these WIAoK executions as WIAoK_i for $i = 1, \ldots, k+1$. In all these WIAoKs, C proves the *same* (compound) statement which is true if and only if:

  (a) there exists randomness η s.t. $c = \mathsf{Com}(v; \eta)$; **or**
  (b) $\beta = \alpha_1 = \alpha_2$, where β is the unique string committed in the transcript of Stage-3.

  Note that an honest prover will always use the witness for part-(a) of the above compound statement, which we refer as the "original witness". We will refer the witness for part-(b) of the compound statement. Looking ahead, some hybrids will use the trapdoor witness to go through the WIAoKs.

Decommitment Phase. The committer C decommits to v and β. R checks if these decommitments are valid, and accepts if so.

Theorem 4.3.1. *The protocol $\langle C, R\rangle_{\mathsf{CCA}}$ (described in Protocol 4.3.1) is a 1-1 CCA commitment scheme.*

Proof. The statistical-binding property of protocol $\langle C, R\rangle_{\mathsf{CCA}}$ is straightforward. The computational hiding property is implied by the 1-1 CCA security as per Definition 4.2.1. In what follows, we focus on the proof of 1-1 CCA security. We prove this property in two steps: we first exhibit a proof of security against synchronizing adversaries in Section 4.3.1, and then consider non-synchronizing adversaries in Section 4.3.2. $\square$

4.3.1 Proof for Synchronous Adversaries

1-1 CCA Security. Recall that in the CCA challenge for commitments, the adversary is a man-in-the-middle adversary that interacts with an honest committer on the left and a decommitment oracle on the right that acts as an honest reciever till the end of the interaction and then reveals the committed value to the adversary if the commitment was valid. The idea is that such an adversary cannot tell apart two different values being committed on the left even given access to the decommitment returned by the oracle on the right.

We will show that the adversary's ultimate output in such a game is indistinguishable for any two distinct values being committed on the left (this is because the values to be committed can be chosen adaptively by the adversary).

Thus consider that there is a man-in-the-middle adversary $\mathcal{A}$ that participates in the CCA challenge outlined above. As before, we will use the convention that unmarked symbols indicate values used in the left interaction and symbols marked with a tilde indicate values used in the right interaction. Fix two arbitrary values v_0 and v_1 in the message space. We will now show

$$\{\mathsf{IND}_0(\langle C, R\rangle_{\mathsf{CCA}}, \mathcal{A}, \mathcal{O}, \lambda, z)\}_{\lambda\in\mathbb{N}, z\in\{0,1\}^*} \overset{c}{\approx} \{\mathsf{IND}_1(\langle C, R\rangle_{\mathsf{CCA}}, \mathcal{A}, \mathcal{O}, \lambda, z)\}_{\lambda\in\mathbb{N}, z\in\{0,1\}^*}$$

To this end, we will use a hybrid argument.

We now describe the hybrids, and prove indistinguishability between contiguous ones. In the process, we will also mark out particular concerns that may render these arguments invalid in the non-synchronous case, and resolve these concerns later.

An Invariant Condition. In each hybrid we will need to refer to the value committed by the man-in-the-middle $\mathcal{A}$ in **Stage 3** of the protocol, denoted by $\widetilde{\beta}$. Since ExtCom is a statistically binding commitment, the value $\widetilde{\beta}$ is *always* uniquely defined given the transcripts of ExtCom[5]. We can formally refer to this value w.r.t. any given machine M: if t is the output of M, we parse t to uniquely obtain the transcript corresponding to ExtCom in the right execution. We then define $\widetilde{\beta}$ to be the value in that transcript and $\widetilde{\beta} = \perp$ if this transcript in t is not uniquely defined. Furthermore, we define $\widetilde{\alpha}$ to be the value that corresponds to the opening in **Stage 3** on right in the output t, setting $\widetilde{\alpha} = \perp$ if this value is not uniquely defined for the given t or the decommitments are invalid. We refer to these values by $\widetilde{\beta}(t)$ and $\widetilde{\alpha}(t)$ if we wish to be explicit about t, and unless specified otherwise, M is assumed to receive a parameter λ as its first input. Note that the role of M will be taken by hybrid machines in the proof. We can now define:

Definition 4.3.1 (Invariant Condition). *For a Turing machine M, the invariant condition is said to hold for M if there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that:*

$$\Pr_{t\leftarrow M(1^\lambda)}\left[\widetilde{\beta}(t) = \widetilde{\alpha}(t)\right] \leq \mathsf{negl}(\lambda).$$

Hybrid H_0^0: This hybrid is identical to the experiment $\mathsf{IND}_0(\langle C, R\rangle_{\mathsf{CCA}}, \mathcal{A}, \mathcal{O}, \lambda, z)$ where the man-in-the-middle $\mathcal{A}$ receives a commitment to v_0 on left. We view H_0^0 (and all other subsequent hybrids) as a machine.

Lemma 4.3.1. *The invariant condition holds for H_0^0.*

Proof. Observe that **Stage-1** and **Stage-2** together are referred to as $\mathsf{CombinedCom}$; this defines a secure statistically binding commitment scheme since it consist of a sequential execution of two commitments (ExtCom and ENMC) which commit to the same value $\widetilde{\alpha}$. We

[5]If the transcript can be decommitted to more than one value or no value at all, we define $\widetilde{\beta} = \perp$.

show that if the invariant condition does not hold for H_0^0 then, we can construct a PPT adversary $\mathcal{A}_{\mathsf{hid}}$ to break hiding of $\mathsf{CombinedCom}$. More specifically, $\mathcal{A}_{\mathsf{hid}}$ incorporates H_0^0; it sends two random values $(\widetilde{\alpha}_0, \widetilde{\alpha}_1)$ to an outside committer of $\mathsf{CombinedCom}$; it then starts to run machine H_0^0 with the following exception:

– It does not run the exponential time oracle or the **Stage-1** and **Stage-2** executions internally; instead it forwards the message from the outside committer to complete these two stages.

– It halts once the left **Stage-2** execution is done, outputting the its view $\mathsf{View}_{\mathcal{A}_{\mathsf{hid}}}$ (which is the same as the view of H_0^0 but "truncated" at the end of the right **Stage-2**).

Next, we construct a distinguisher D who incorporates $\mathcal{A}$ (and hence H_0^0); it gets as input the view $\mathsf{View}_{\mathcal{A}_{\mathsf{hid}}}$ and proceeds as follows: D continues the execution of H_0^0 from the state where $\mathcal{A}$ halts, denoted st. Observe that D has all the information it needs to continue this execution. D halts at the end of **Stage-3** on right. If $\mathcal{A}$ completes right **Stage 3** successfully, D runs the extractor of ExtCom to extract the committed value. By definition, if the invariant condition does not hold, it follows that $\mathcal{A}$ commits to valid value such that $\tilde{\beta} = \tilde{\alpha}$ with noticeable probability ϵ (for infinitely many λ), and therefor (by using standard averaging argument to account for good values of st) D learns $\tilde{\alpha}$ in expected PPT time. This violates hiding of $\mathsf{CombinedCom}$. □

Hybrid H_1^0: This hybrid is identical to H_0^0, except that it does not run the exponential time oracle $\mathcal{O}$; instead, if the right executions are accepting, it learns the committed value $\widetilde{v}$ by extracting it from WIAoK_{k+1} (on right). If extraction fails, the extracted value is assumed to be $\perp$. Note that H_1^0 is expected PPT.

Observe that H_1^0 and H_0^0 have identical executions up until the $\mathcal{A}$ finishes its execution on right. Therefore, invariant condition holds in H_1^0. Consequently, by properties of the extractor for WIAoK, this hybrid *always* (i.e., with probability 1) extracts a valid witness (which includes the committed value) in expected PPT time. Thus, H_1^0 and H_0^0 are identically distributed.

Hybrid H_2^0: This hybrid is identical to H_1^0, except that whenever the left ENMC is accepting, H_1^0 extracts the committed value α from the left ENMC. If extraction fails, H_1^0 outputs $\perp$ and halts; otherwise it continues as H_1^0.

The outputs of H_1^0 and H_2^0 differ only when extraction fails, which happens with negligible probability. Therefore the two hybrids have statistically close outputs, and consequently, the invariant condition also holds in H_2^0.

Remark 4.3.1. *The above proofs for both indistinguishability and invariant condition are independent of $\mathcal{A}$'s scheduling, and work for the non-synchronizing case.*

Hybrid H_3^0: This hybrid is identical to H_2^0, except that H_3^0 sets $\beta = \alpha$ (the value extracted from the left ENMC in H_2^0) in **Stage-3** ExtCom on left.

First, note that if the invariant condition holds in H_3^0, the indistinguishability of H_2^0 and H_3^0 follows directly from the hiding property of left ENMC. The proof of the invariant condition for this hybrid is rather involved. We prove it in Lemma 4.3.2 towards the end. In the following, let us continue to assume that the invariant condition holds in H^0_3.

We will now define a number of hybrids in sequence:

- **Hybrid H^0_{3+i} ($i \in [k]$):** This hybrid switches from proving statement (1) to statement (2) in WIAoK_i (and also therefore switching from using the "original" witness to the "fake" one).

  Indistinguishability. We note that if the invariant condition holds in H^0_3, it should also hold in H^0_{3+1} through H^0_{3+k}. This is because for each $i \in [k]$, H^0_{3+i} and H^0_3 are identical up to the end of **Stage-4**, and any changes after this stage do not affect the invariant condition in the synchronizing case. Now, if the invariant condition holds in H^0_{3+i} ($i \in [k]$), indistinguishability between H^0_{3+i-1} and H^0_{3+i} for every i follows directly from the WI property (since the extraction only happens from WIAoK_{k+1}).

- **Hybrid H^0_{3+k+1}:** This hybrid is identical to H^0_{3+k}, except that instead of extracting the "witness" (i.e., the committed value $\widetilde{v}$) from WIAoK_{k+1}, it extracts from WIAoK_1 (which are both on the right).

  Indistinguishability. Hybrids H^0_{3+k} and H^0_{3+k+1} proceed identically until the extraction is performed on right. Therefore, the invariant condition holds in H^0_{3+k+1}. Consequently, by the knowledge soundness of WIAoK, H^0_{3+k} and H^0_{3+k+1} are statistically close. This implies both that the invariant holds in H_{3+k+1}, and also that the outputs in hybrids H_{3+k} and H_{3+k+1} are indistinguishable.

- **Hybrid H^0_{3+k+2}:** This hybrid is identical to H^0_{3+k+1} except that it switches from the original witness to the trapdoor witness (i.e., values and randomness corresponding to $\beta = \alpha$) in the left WIAoK_{k+1}.

  Indistinguishability. The proofs of both indistinguishability as well as the invariant condition are exactly as for H^0_{3+k} (or any of the other similar hybrids).

Note that in hybrid H^0_{3+k+2}, we can safely substitute v_0 with v_1(thanks to the hiding of Com). Then we can build a sequence of hybrids similar to the above one, but in the reverse order, to finally reach the real execution $\mathsf{IND}_1(\langle C, R\rangle_{\mathsf{CCA}}, \mathcal{A}, \mathcal{O}, n, z)$. More formally, for $j = 0, \ldots, 3 + k + 2$, define hybrid H^1_j analogously to hybrid H^0_j (by replacing v_0 with v_1 on left).

First, note that the indistinguishability of H^0_{3+k+2} and H^1_{3+k+2} follows from that of Com since these hybrids do not use the "real witness" (i.e., the committed values) in their executions and they are both expected PPT. Then, using the same arguments as above, we conclude that H^1_{3+k+2} and H^1_0 are computationally indistinguishable and invariant condition holds in each of them. This eventually finishes the proof for 1-1 CCA security against synchronous adversaries.

We now prove the following lemma, used earlier in the proof.

Lemma 4.3.2. *The invariant condition holds for (hybrid) machine H^0_3.*

Proof. We reduce the veracity of the invariant in this experiment to the (synchronous) non-malleability of ENMC. Recall that $\mathcal{A}$ is the adversary for our CCA$^{1:1}$ scheme in H^0_3. We construct two machines to violate non-malleability of ENMC: a man-in-the-middle adversary A_{NMC} who attempts to commit a related value, and a corresponding distinguisher D_{NMC} who distinguishes the (joint) distribution of values committed by A_{NMC} on right.

At a high level, we cannot directly reduce to non-malleability of ENMC due to the presence of ExtCom in **Stage-1** which commits to the same value as ENMC. Since ExtCom is not non-malleable, adversary $\mathcal{A}$ may be able to rely on this commitment to create related values on right in our protocol. Specifically, in **Stage-3**, when the hybrid sets $\beta = \alpha$ on left, $\mathcal{A}$ may succeed in violating the invariant condition since **Stage-3** uses the (possibly malleable) ExtCom. It is also not sufficient to replace the **Stage-1** ExtCom with a non-malleable commitment since committed value is well defined only when both stages (1 and 2) commit to the same value. This is a relation over two values but ENMC is not concurrently non-malleable. We therefore proceed in a different manner where adversary A_{NMC} will simulate stage 1 on right (by commiting one of the two random values of its choice) while receiving an ENMC commitment from outside for **Stage-2**. This will simulate the conditions of hybrid H_3^0 with noticeable probability; and thus, if the invariant condition does not hold, the distinguisher can extract the committed value $\widetilde{\beta}$ to violate hiding of ENMC on right.

Adversary A_{NMC}. This adversary participates in the non-malleability experiment w.r.t. commitment scheme ENMC. It does so by proceeding exactly as hybrid H_3^0 internally while interacting with an outside committer as follows:

- A_{NMC} picks two random values a_0 and a_1 and sends them to the outside committer of ENMC. (Note that the outside committer will commit to one of a_0 or a_1, but A_{NMC} does not know which one).
- A_{NMC} also starts the execution of adversary $\mathcal{A}$ internally, proceeding exactly as H_3^0 except that in **Stage-1** ExtCom on right, it commits to a randomly chosen value from $\{a_0, a_1\}$. We denote this value by a_b where b is a random bit.
- Next, in **Stage-2**, A_{NMC} does not run the ENMC internally. Instead, it sends all messages of the external (ENMC) committer as **Stage-2** messages of the *right* session for the internal adversary $\mathcal{A}$. Likewise, the messages of the left side stage 2 are sent to an outside receiver of ENMC.
- A_{NMC} halts at the end of **Stage-2**. For future reference, let state be state of machine A_{NMC} at this point.

Note that if outside committer commits to a_b, the state state of A_{NMC} is distributed identically to that of hybrid H_3^0 at the end of stage 2. Let us now describe the distinguisher D_{NMC}.

Distinguisher D_{NMC}. The input to the distinguisher is a pair (m, View) distributed either as $\mathsf{MIM}_{\mathsf{ENMC}}^{A_{\mathsf{NMC}}}(a_0, \lambda, z)$ or $\mathsf{MIM}_{\mathsf{ENMC}}^{A_{\mathsf{NMC}}}(a_1, \lambda, z)$ where z is an arbitrary advice string for algorithm A_{NMC}. Note that m is the value committed by A_{NMC} and View is the joint view of both executions it participates in. The distinguisher incorporates A_{NMC} and proceeds as follows:

- By definition, View includes the joint view of A_{NMC}, which in turn contains the view of $\mathcal{A}$ and hence state state as well as (a_0, a_1). Recall that hybrid H_3^0 extracts the value committed in the left ENMC, denoted α_2. However, this extraction cannot be performed by A_{NMC} as the ENMC execution now happens between A_{NMC} and the external challenger for the non-malleable game, who cannot be rewound. But notice that α_2 is exactly equal to m; looking ahead, the D_{NMC} will finish emulating H_3^0 using m in place of the extracted α_2.

– D_{NMC} defines the following machine C^*: C^* incorporates machine A_{NMC} and has values (m, View) hardwired. It starts the machine A_{NMC} from state state and continues to proceed exactly as H_3^0 in the next stage. In particular, it does not extract anything from ENMC and simply uses m in its place. That is, it sets $\beta = m$ in the **Stage-3** execution of ExtCom on left. Furthermore, C^* forwards all messages corresponding to *right* stage 3 to an external receiver of ExtCom. Note that C^* is simply a valid committer of ExtCom.

– D_{NMC} runs C^* interacting with it as an honest receiver. If the commitment is accepting, it extracts the value $\tilde{\beta}$ committed by C^* (using the extractor of ExtCom).

– If $\tilde{\beta} = a_0$, it outputs 0; if $\tilde{\beta} = a_1$, it outputs 1. Otherwise, it outputs a random bit.

Let $\nu := \nu(\lambda)$ denote the probability that the claim is false, i.e., the invariant condition does not hold in this hybrid: $\nu = \Pr[\widetilde{\beta} = \widetilde{\alpha}]$ where the probability is taken over transcripts (suppressed in the notation) sampled by H_3^0. Let us calculate the advantage $|\Delta|$ of D_{NMC} where

$$\Delta := \Pr\left[D\left(\mathsf{MIM}_{\mathsf{ENMC}}^{A_{\mathsf{NMC}}}(a_0, \lambda, z)\right) = 1\right] - \Pr\left[D\left(\mathsf{MIM}_{\mathsf{ENMC}}^{A_{\mathsf{NMC}}}(a_1, \lambda, z)\right) = 1\right]$$

For succinctness, let $X_{\lambda,z}(a) := \mathsf{MIM}_{\mathsf{ENMC}}^{A_{\mathsf{NMC}}}(a, \lambda, z)$. We have,

$$\Pr\left[D_{\mathsf{NMC}}\left(\mathsf{MIM}_{\mathsf{ENMC}}^{A_{\mathsf{NMC}}}(a_0, \lambda, z)\right) = 1\right] = \Pr\left[D_{\mathsf{NMC}}\left(X_{\lambda,z}(a_0)\right) = 1\right]$$
$$= \frac{1}{2} \cdot \Big(\underbrace{\Pr\left[D_{\mathsf{NMC}}\left(X_{\lambda,z}(a_0)\right) = 1 | b = 0\right]}_{:=z_0} + \underbrace{\Pr\left[D_{\mathsf{NMC}}\left(X_{\lambda,z}(a_0)\right) = 1 | b = 1\right]}_{:=\delta_0} \Big)$$

Observe that $1 - z_0 = \Pr\left[D_{\mathsf{NMC}}\left(X_{\lambda,z}(a_0)\right) = 0 | b = 0\right]$. Note that in this equation, since $b = 0$, the input to D_{NMC} has distribution identical to that of stage 1 and 2 on right in the execution H_3^0. We split the probability based on the invariant condition (i.e., whether $\widetilde{\beta} = \widetilde{\alpha}_0$). That is,

$$1 - z_0 = \Pr\left[D_{\mathsf{NMC}}\left(X_{\lambda,z}(a_0)\right) = 0 \wedge (\widetilde{\beta} = \widetilde{\alpha}_0) | b = 0\right] + \Pr\left[D_{\mathsf{NMC}}\left(X_{\lambda,z}(a_0)\right) = 0 \wedge (\widetilde{\beta} \neq \widetilde{\alpha}_0) | b = 0\right]$$
$$= \nu + (1 - \nu) \cdot \left(\frac{1}{2} - 2^{-\lambda}\right)$$
$$\Rightarrow z_0 = 1/2 - \nu/2 + \mathsf{negl}(\lambda).$$

where the first term (of the second equality above) comes from our assumption about the invariant condition, and in that case, the extractor always extracts $\widetilde{\alpha}_0$ and hence outputs 0; otherwise (i.e., with $1 - \nu$ probability), it outputs a random guess; note that in this case it is possible that $\widetilde{\beta} = \widetilde{\alpha}_1$ (the other string) in which case D_{NMC} will output the "wrong" guess 1 but since $\widetilde{\alpha}_1$ is outside the view of (internal) $\mathcal{A}$, this happens only with probability $2^{-\lambda}$.

We use an analogous calculation for the case when outside committer commits to a_1.

$$\begin{aligned}
&\Pr\left[D_{\mathsf{NMC}}\left(\mathsf{MIM}^{A_{\mathsf{NMC}}}_{\mathsf{ENMC}}(a_1,\lambda,z)\right)=1\right]\\
=\ &\frac{1}{2}\cdot\Big(\underbrace{\Pr\left[D_{\mathsf{NMC}}\left(X_{\lambda,z}(a_1)\right)=1|b=1\right]}_{:=z_1}+\underbrace{\Pr\left[D_{\mathsf{NMC}}\left(X_{\lambda,z}(a_1)\right)=1|b=0\right]}_{:=\delta_1}\Big)
\end{aligned}$$

where

$$\begin{aligned}
z_1 &= \Pr\left[D_{\mathsf{NMC}}\left(X_{\lambda,z}(a_1)\right)=1\wedge(\widetilde{\beta}=\widetilde{\alpha}_1)|b=1\right]+\Pr\left[D_{\mathsf{NMC}}\left(X_{\lambda,z}(a_1)\right)=1\wedge(\widetilde{\beta}\neq\widetilde{\alpha}_1)|b=1\right]\\
&=\nu+(1-\nu)\cdot\left(\frac{1}{2}-2^{-\lambda}\right)\\
&\Rightarrow z_1=1/2+\nu/2-\mathsf{negl}(\lambda).
\end{aligned}$$

Finally, we observe that $\delta_0=\delta_1$ since both of these cases correspond to committing two random and independently chosen strings in stage 1 and 2 on right respectively; in other words, the input to D_{NMC} in these cases are identically distributed. Putting everything together, we obtain $|\Delta|=\nu/2+\mathsf{negl}(\lambda)$. This violates the non-malleability of ENMC if ν is not negligible.

□

4.3.2 Proof for Non-synchronous Adversaries

We first define some terms and notations.

Alignments and Free Slots. Recall that ExtCom has exactly 3-rounds. Let (m_1,m_2,m_3) and $(\widetilde{m}_1,\widetilde{m}_2,\widetilde{m}_3)$ be messages of stage-1 ExtCom on left and right respectively. We say that stage-1 ExtCom on left and right are *aligned* in a schedule, if m_1 follows immediately after $\widetilde{m}_1$, $\widetilde{m}_2$ follows immediately after m_2, and finally m_3 follows immediately after $\widetilde{m}_3$. We define the aligning of stage-2 ENMC on left and right, as well as stage-3 ExtCom on left and right, analogously. We refer to the last two messages of ExtCom and ENMC as slots. Next, recall that CombinedCom refers to the sequential execution of stage 1 and stage 2 (see Protocol 4.3.1); since the last message of stage 1 and first message of stage 2 can be sent together as a single message, and both stages commit to the same value, protocol CombinedCom is a 5-round commitment scheme which has 2 slots (one for ExtCom and one for ENMC). We say that left and right executions of CombinedCom are aligned if its component stages 1 and 2 are aligned with their left and right counterparts respectively.

Consider an arbitrary schedule of left and right sessions. A *free slot* of left CombinedCom is a slot that does not contain any message of the CombinedCom on right; it may however contain other protocol messages. It is not hard to see that by definition of alignment (and our modeling that the honest parties immediately respond with their next message) it follows that if left and right CombinedCom sessions are not aligned in a schedule, there must exist a free slot on left. which does not contain *any* message of the right execution of CombinedCom. The existence of free slot is not required until later in the proof; we will do a case-by-case analysis to demonstrate that such a free slot must exist.

Hybrids. We now define the hybrids for the non-synchronous case following roughly the same structure as the synchronous case. The new hybrids will be called NewH_i^0 for $i = 1, \ldots, 3 + 2(k+1)$.

Hybrid NewH_0^0: Identical to H_0^0.

Hybrid NewH_1^0: Identical to H_1^0.

Hybrid NewH_2^0: Identical to NewH_1^0 except that it extracts a value α^* on left as follows: if the left and right executions of $\mathsf{CombinedCom}$ are not aligned, it extracts from the free slot. Such a free slot always exists by definition. Otherwise, it proceeds exactly as H_2^0 and extracts from ENMC.

We remark that if $\mathcal{A}$ chooses to commit different values in stage-1 and 2 on left, depending upon which slot is free, extractor may get different values for α^*.

Hybrid NewH_3^0: Identical to NewH_2^0 except that it sets $\beta = \alpha^*$.

Now, for $i \in [k+1]$, we define:

Hybrid $\mathsf{NewH}_{3+(2i-1)}^0$: Identical to the previous hybrid, except that instead of extracting the "witness" (i.e., the committed value $\widetilde{v}$) as in the previous hybrid, this hybrid extracts from WIAoK_j on the right where $j \in [k]$ is an index such that WIAoK_j does not contain any message of *left* WIAoK_i. Note that such an index j must exist: the left WIAoK_i execution has but k messages, and each message can occur within at most *one* WIAoK execution on the right (recall that the right WIAoK executions are all sequential, so they cannot overlap by definition), and we have $k+1$ WIAoK executions on the right.

Hybrid $\mathsf{NewH}_{3+(2i)}^0$: Identical to the previous hybrid except that it switches from the original witness to the trapdoor witness (i.e., values and randomness corresponding to $\beta = \alpha^*$) in the left WIAoK_i.

Recall that from Remark 4.3.1, the proofs for indistinguishability and the invariant condition remain unchanged up to hybrid NewH_2^0. We now prove similar claims for the remaining hybrids.

Indistinguishability of NewH_2^0 and NewH_3^0: The main concern here is if $\mathcal{A}$ overlaps the left stage-3 ExtCom with the right WIAoK_{k+1}, then since these hybrids extract from WIAoK_{k+1} via rewinding, we cannot rely directly on the hiding of left stage-3 ExtCom. This is easy to fix by considering some intermediate hybrids where we first switch to extraction from a 'free' WIAoK on the right and later switch back. We describe the intermediate hybrids below.

– **Hybrid $\mathsf{NewH}_{2,1}^0$:** This hybrid is identical to NewH_2^0 except that if last two messages of stage-3 ExtCom on *left* appear after the first message of WIAoK_{k+1} on *right*, then this hybrid performs extraction from a 'free' WIAoK session WIAoK_* (instead of WIAoK_{k+1} used by the previous hybrid).

  Hybrids NewH_2^0 and $\mathsf{NewH}_{2,1}^0$ are statistically close since they only differ when the extractor fails, which happens with negligible probability. Thus the invariant condition holds for $H_{2,1}^0$ since it holds for H_2^0.

- **Hybrid $\mathsf{NewH}_{2,2}^0$:** This hybrid is identical to $\mathsf{NewH}_{2,1}^0$ except that it sets $\beta = \alpha^*$.

  In hybrid NewH_3^0, we will show that (a) $\mathsf{NewH}_{2,2}^0$ is statistically close to NewH_3^0, and (b) invariant condition holds in NewH_3^0. It follows that invariant condition must also hold in this hybrid.

  We now prove the indistinguishability of $\mathsf{NewH}_{2,2}^0$ and $\mathsf{NewH}_{2,1}^0$. This follows directly from the hiding of stage-3 ExtCom since if it does not then we can define a machine B to break hiding of ExtCom as follows: B receives ExtCom to either 0^λ or α^* and uses it as the stage-3 commitment. Observe that this machine does not rewind the outside ExtCom when performing extraction from right side WIAoK: this is because stage-3 ExtCom on left has only 3 rounds, and thus, the last two messages of this stage can only be contained in one of the WIAoK_i executions on the right—so all the others are always "free". The hybrid only rewinds and extract from a free WIAoK.

- **Hybrid $\mathsf{NewH}_{2,3}^0$:** This hybrid is identical to the previous hybrid except that it always extracts from WIAoK_{k+1} on right.

  It is straightforward to see that $\mathsf{NewH}_{2,2}^0$ is statistically close to the previous hybrid and thus invariant condition also holds.

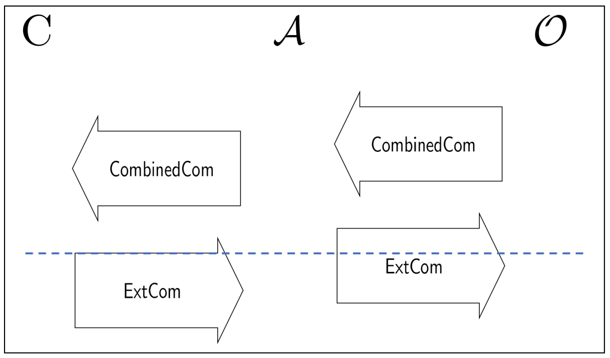
Observe that $\mathsf{NewH}_{2,2}^0$ is in fact the original hybrid NewH_3^0. Therefore, NewH_3^0 and $\mathsf{NewH}_{2,2}^0$ are also statistically close as claimed above.

Invariant Condition in NewH_3^0. Recall that this hybrid involves setting β to be the extracted value. It seems reasonable to expect that the invariant condition will depend on the relationship between the left and right executions of the stage 3 ExtCom (this makes intuitive sense because we expect a cheating adversary to gain in success by possibly 'mauling' this changed β and trying to violate the invariant). Accordingly, consider the following three cases involving the relative positions of the left and right stage 3 ExtCom executions:

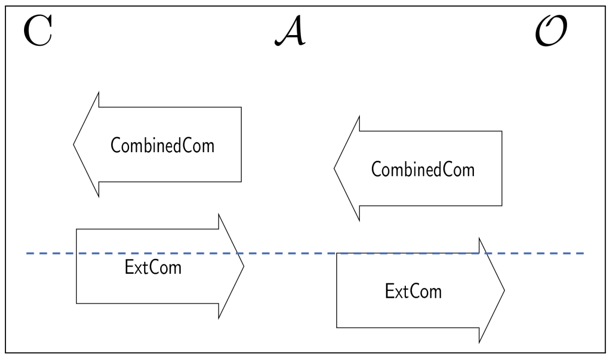
- **Right stage 3 ExtCom occurs *before* left stage 3 ExtCom:** If this is the case (see Figure 4.1a), then it must be that the first message of the right stage 3 ExtCom is sent *before any message of the left* ExtCom *is sent.* Note that that the first message of ExtCom binds the commitment to the underlying value, this implies that the right stage 3 ExtCom cannot possibly commit to a value that depends on the left ExtCom. Thus the invariant holds in NewH_3^0 for such schedulings by the same argument as used for $\mathsf{NewH}_{2,2}^0$ (and therefore NewH_2^0).

- **Right stage 3 ExtCom aligns with left stage 3 ExtCom:** Note that this hybrid (just as NewH_2^0) uses a conditional extraction strategy on the left. Accordingly, we consider the following two subcases:

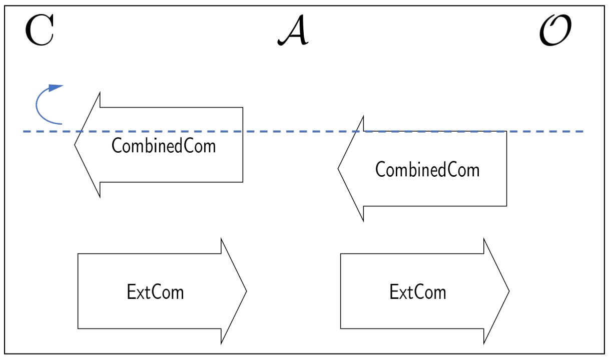
  * **Left ENMC aligns with right ENMC:** In this case, our reduction to non-malleability of ENMC (arguing the validity of the invariant in H_3^0 in the synchronous case) again applies (note that in this case, *all* of stage 0 through stage 3 is aligned on the left and the right, so that $\mathcal{A}$ is synchronous up till stage 3, and our argument in that case makes no assumptions about what happens after that stage).



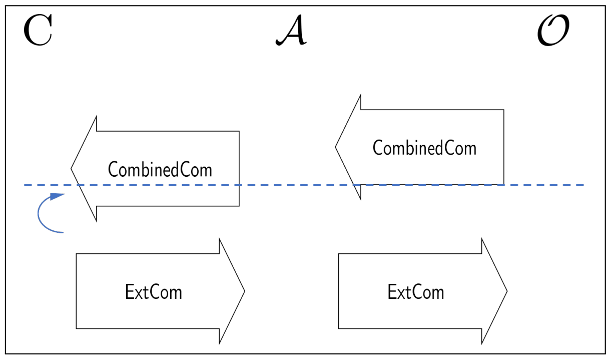
(a) Right ExtCom occurs before left ExtCom



(b) Right ExtCom occurs after left ExtCom



(c) Left CombinedCom occurs before right CombinedCom



(d) Left CombinedCom occurs after right CombinedCom

Figure 4.1: Various representative schedulings considered for invariant in NewH_3^0

∗ **Left ENMC does not align with right ENMC:** Recall that we had defined CombinedCom in Protocol 4.3.1. We claim in this case that if the left and right ENMC executions are not aligned, then there must be a 'free slot' in the left CombinedCom (i.e., one of stage 1 ExtCom or ENMC on the left). There are 4 possible representative schedulings, and we deal with each separately.

1. **Left CombinedCom 'occurs before' right CombinedCom:** By this we mean that the left CombinedCom both starts and ends *before* the right CombinedCom (see Figure 4.1c). In this case, the first slot in the left stage-1 ExtCom is *free* since no slot in CombinedCom on the right occurs within it.
2. **Left CombinedCom 'occurs after' right CombinedCom:** By this we mean that the left CombinedCom both starts and ends *after* the right CombinedCom (see Figure 4.1d). In this case, the final slot in the left stage-2 ENMC is *free* since no slot in CombinedCom on the right occurs within it.
3. **Left CombinedCom 'occurs inside' right CombinedCom:** By this we mean that the left CombinedCom both starts after and ends before the right CombinedCom. This implies that the left stage-1 ExtCom ends *after* the right one, and the left ENMC starts after the right one. Then both the final slot in the left stage-1 ExtCom and the first slot in the left stage-2 ENMC are free.

4. **Left** $\mathsf{CombinedCom}$ **'envelopes' right** $\mathsf{CombinedCom}$**:** By this we mean that the left $\mathsf{CombinedCom}$ both starts before and ends after the right $\mathsf{CombinedCom}$. Then both the first slot in the left stage-1 ExtCom and the final slot in the left stage-2 ENMC are free.

   We claim that if the invariant condition is violated in this case, then we can break hiding of $\mathsf{CombinedCom}$. In fact, we can apply the same proof as for Lemma 4.3.1, noting that the reduction is unchanged because we *do not* rewind any slot in the right $\mathsf{CombinedCom}$ (since we extract from a free slot on the left).

– **Right stage 3** ExtCom **occurs *after* left stage 3** ExtCom**:** The argument presented for the previous case also applies to this one (see Figure 4.1b).

– **Right stage 3** ExtCom **'occurs inside' left stage 3** ExtCom**:** In this case again the first message for the right stage-3 ExtCom is sent *before* the corresponding first message for the left stage-3 ExtCom. So our argument for the first case applies here too.

– **Right stage 3** ExtCom **'envelopes' left stage 3** ExtCom**:** Once again, we can use our argument for the case where the left and right stage-3 ExtCom sessions are aligned, without change.

Indistinguishability of $\mathsf{NewH}^0_{3+(2i-1)}$ **and** $\mathsf{NewH}^0_{3+(2i)}$**:** These hybrids are statistically close since they only differ when the extractor fails, which happens with negligible probability. This implies that the invariant condition must also hold in the latter hybrids. Indistinguishability of outputs follows immediately.

Note that this argument also serves to prove indistinguishability between NewH^0_3 and NewH^0_4, in particular.

Invariant Condition in $\mathsf{NewH}^0_{3+(2i)}$**:** We consider two cases: if the execution of the stage 3 ExtCom on the right is aligned with or occurs after that on the left, then we can resort to the same argument as for NewH^0_3 (for the corresponding schedulings). If not, then we can use the corresponding argument showing the invariant for H^0_{3+i} in the synchronous case, since that relies only on the right ExtCom occurring before or during the stage 3 ExtCom on the left.

Indistinguishability of $\mathsf{NewH}^0_{3+(2i)}$ **and** $\mathsf{NewH}^0_{3+(2i+1)}$**:** This follows directly from the witness indistinguishability of WIAoK_i since if it does not then we can define a machine B that breaks witness indistinguishability of WIAoK as follows: B receives prover messages proving either statement (i) (using real witness) or (ii) (using trapdoor witness) and uses it as the WIAoK_i messages on the left (it forwards the replies of $\mathcal{A}$ outside to this prover). Observe that this machine does not rewind the outside WIAoK execution when performing extraction from right side WIAoK: this is because we have ensured that the left WIAoK_i and the right WIAoK_j we extract from do not overlap. We conclude that these hybrids are indeed indistinguishable.

Thus we show that the outputs of hybrids NewH^0_0 and $\mathsf{NewH}^0_{3+(2k+3)}$ are indistinguishable. As before, we can define an analogous set of hybrids $\mathsf{NewH}^1_0, \dots, \mathsf{NewH}^1_{3+(2k+3)}$ where hybrids commit to v_1 on the left. Further, once again we observe that the indistinguishability of

$\mathsf{NewH}^0_{3+(2i+1)}$ and $\mathsf{NewH}^1_{3+(2k+3)}$ follows from the hiding of Com since these hybrids do not use the "real witness" (i.e., the committed values) in their executions and they are both expected PPT (this is the same argument as in the synchronous case, because it does not depend on the adversary's scheduling: this commitment is sent at the start of the execution on the left and takes only one round, hence cannot be rewound by the adversary). This finishes the proof of the non-synchronous case, and hence that of Theorem 4.3.1.

4.4 Our Black-Box CCA Commitment

In this section, we describe a fully black-box instantiation of our commitment scheme. We first describe black-box versions of all components of our protocol which involve a proof. The final construction follows by simply plugging in the black-box components into our protocol. To this end, we require an instantiation of a WIAoK scheme that can handle proofs over committed values in a black-box manner and is consistent with commitments performed in multiple stages. To achieve this, we also need a new instantiation for our extractable commitment scheme.

4.4.1 Black-Box Commit-and-Prove ZKAoK

There are several formulations of "black-box commit-and-prove" protocols in the literature, usually tailored to their intended applications. For our purposes, we need a black-box commit-and-prove that, has the argument-of-knowledge property (i.e., an appropriate witness can be extracted from the prover), as well as the ZK property. We will also need the ability to give proofs over multiple commitments, each of which may have been performed independently at different times. But the proof, given the witnesses for each of these executions, should be able to prove any predicate ϕ in zero-knowledge; furthermore sequential composition of a constant number of such proofs (for potentially different predicates) should be zero-knowledge. To capture these properties, we start by defining the primitive we need below.

Definition 4.4.1. *A black-box s-commit-and-prove* ZKAoK *scheme consists of a pair of protocols (*BBCom, BBProve*) executed between a pair of* PPT *machines P and V.* BBCom *is a statistically binding commitment scheme, and* BBProve *is an interactive argument system. These protocols are executed in the following stages:*

- **Commit Stage:** *P and V invoke* $\mathsf{BBCom}(x)$ *such that at the end of this protocol, P is statistically committed to the value x.*

  If desired, P can commit to up to s values by invoking s independent BBCom *instances. For $i \in [s]$, we use τ_i to denote the transcript from* $\mathsf{BBCom}(x_i)$ *execution. P stores private state* state*.*

- **Prove Stage:** *P and V take the transcripts $\{\tau_1, \ldots, \tau_s\}$ and a predicate ϕ as common input. P takes* state *as its private input. P proves to V using* BBProve *that there exists some values $(x_1, \ldots, x_s)$ such that $\{\tau_1, \ldots, \tau_s\}$ are valid commitments to them, and also $\phi(x_1, \ldots, x_s) = 1$.*

We require that the following properties are satisfied:

- **Black-Box.** *Both stages only require black-box access to cryptographic primitives.*
- **Completeness.** *If P and V are honest, then V accepts the proof with probability 1.*
- **Zero-Knowledge.** *For every* PPT *verifier V^*, there exist an (expected)* PPT *simulator* Sim *such that for all $(x_1, \ldots, x_s)$, for every polynomial time predicate ϕ, for every auxiliary input $z \in \{0,1\}^*$, it holds that*

$$\mathsf{Sim}^{V^*}(z, \phi) \overset{c}{\approx} \{\langle P(x_1, \ldots, x_s), V^*(z)\rangle\}_{\phi}$$

  where $\langle P(x_1, \ldots, x_s), V^*(z)\rangle_{\phi}$ denotes the view of V^* at the end of both the Commit Stage and the Prove Stage.
- **Argument of Knowledge.** *There exists an (expected)* PPT *oracle algorithm E such that for every* PPT *machine P^* aand every polynomial time predicate ϕ, and every auxiliary input $z \in \{0,1\}^*$, if $\langle P^*, V\rangle_{\phi}$ constitutes an accepting view of V, with corresponding commit stage transcripts $(\tau_1, \ldots, \tau_s,\ E^{P^*})$ will output $(\widetilde{x}_1, \ldots, \widetilde{x}_s)$ such that $\phi(\widetilde{x}_1, \ldots, \widetilde{x}_s) = 1$ and it is statistically impossible to decommit $(\tau_1, \ldots, \tau_s)$ to any tuple other than $(\widetilde{x}_1, \ldots, \widetilde{x}_s)$.*

Some remarks are in order. First, the prover can sequentially prove *multiple* predicates $\phi_1, \ldots, \phi_k$ over the *same* commit stage transcripts $(\tau_1, \ldots, \tau_s)$ (or any subset of these). The zero knowledge property of this sequential composition is implied by the *auxiliary input* nature of the zero knowledge definition above. Furthermore, even though the simulator simulates the commit stage as well, the presence of such a simulator trivially guarantees *witness indistinguishability* of the *proof stage* as well; that is, if there are multiple witnesses for $(\tau_1, \ldots, \tau_s)$, polynomial time verifiers cannot tell which witness was used in the proof stage.

We remark that known black-box commit-and-prove protocols (such as in [KOS18]) do not directly satisfy Definition 4.4.1. In fact, it is unclear if the construction in [KOS18] can be easily modified for our purposes. At a high level, this is since:

(a) there is no stand-alone "commitment stage", by the end of which the committer is statistically bound to some (committed) value; this is because [KOS18] considers "commit-and-prove" as a single object so that at the end of the execution, it is guaranteed that the committer is committed to a value m such that $\phi(m) = 1$; we need the guarantee that m is already defined after a "commitment stage", which will be used in a "proof stage" that happens latter;

(b) it is also not clear how to extend [KOS18] to support multiple commitments and multiple proofs; in fact it seems that it can only support one proof since two valid responses from the prover in their protocol may lead to extraction of the committed value.

4.4.1.1 Constructions against Honest Verifiers

We note that the "MPC-in-the-head" construction in [IKOS07] already achieves the honest-verifier version of Definition 4.4.1. In the following, we recall their protocol Π_{IKOS}. This protocol makes use of the following primitives:

– A statistically-binding commitment scheme Com (e.g. Naor's commitment [Nao90]);

– $(n+1,t)$-perfectly verifiable secret sharing scheme $\Pi_{\mathsf{VSS}} = (\mathsf{VSS}_{\mathsf{Share}}, \mathsf{VSS}_{\mathsf{Recon}})$

– A t-secure MPC protocol in the malicious model.[6]

We will pick our parameters such that t is a constant multiple of the security parameter λ, and n is a constant multiple of t. The prover of Π_{IKOIS} first commits to a value x and then prove a predicate ϕ on x. It consists of the following two stages:

Commit Stage ($\mathsf{IKOS\text{-}Com}(x)$): To commit to a value x, the prover P runs an MPC protocol "in his head" where one dealer and n parties runs a $(n+1,t)$-VSS protocol such that each party holds one VSS share of x in the end. P commits to the views of each party (separately) using Com.

Prove Stage ($\mathsf{IKOS\text{-}Prove}(\phi)$): This consists of the following steps:

(1) To prove that a predicate ϕ is satisfiable, the prover asks the n parties "in his head" to execute an MPC protocol where each party learns the value $\phi(x)$ as the output (note that this can be done as each party holds a VSS share of x at the end of $\mathsf{BBCom}(x)$). When the computation is finished, the prover commits to the views of each party.

(2) The verifier sends a random subset $\mathsf{ch} \subset [n]$ of size t as his challenge.

(3) The prover then decommits to the views (in both $\mathsf{BBCom}(x)$ and the computation of $\phi(x)$) of the parties specified by ch. The verifier accepts only if all the decommitments are valid and all the views are consistent as per **??**.

Remark 4.4.1 (Committing to Multiple Strings)**.** *In the above, the commit-and-prove is performed on a single value x. It can be extended to multiple values $\{x_1,\ldots,x_s\}$ by invoking* BBCom *on each of them independently (where s is polynomially related to the security parameter λ). In the proving stage, simply run the MPC protocol w.r.t. the functionality $\phi(x_1,\ldots,x_s)$. This construction is due to Goyal et al. (in the works [GLOV12, GOSV14]).*

4.4.1.2 Security against Dishonest Verifiers

The above Π_{IKOS} protocol only achieves honest-verifier ZKAoK property. To make it secure against malicious verifiers (and also to prevent selective-opening attacks), [GLOV12] ask the verifier to commit its challenge ch before $\mathsf{BBCom}(x)$ starts. However, this approach only gives us a zero-knowledge protocol, which does not have the AoK property.

Remark 4.4.2. *We notice that [IKOS07] also presented a constructions that achieves* ZKAoK *against dishonest verifiers. However, they did that by invoking polynomially-many instances of Blum's coin-tossing protocol [Blu82]* sequentially*. It does not satisfy our needs as we aim to have a constant-round construction.*

To satisfy our purpose, we show how to make above construction to be a ZKAoK based

[6]In fact, we only need the MPC protocol to be t-private in the semi-honest model and (perfectly or statistically) t-robust in the malicious model (as defined in Definition 2.7.5). See [IKOS07] for more details.

Protocol 4.4.1: Black-Box Commit-and-Prove ZKAoK $\Pi^{\mathsf{BB}}_{\mathbf{ZKAoK}}$

This protocol, denoted by $\Pi^{\mathsf{BB}}_{\text{ZKAoK}}$, makes use of the same Com, ExtCom, Π_{VSS} and t-secure MPC protocol (with the same parameter settings) as in Π_{IKOS}. It consists of the following two stages:

Commit Stage (BBCom): this is the same as in protocol Π_{IKOS}. We remark that to commit to s values $\{x_1, \ldots, x_s\}$, P and V invokes $\mathsf{IKOS\text{-}Com}(x_i)$ for each $i \in [s]$ separately.

Prove Stage ($\mathsf{BBProve}(\phi)$): This stage consists of the following steps:

(1) Same as step (1) of protocol $\mathsf{IKOS\text{-}Prove}$ (i.e., **Prove Stage** of Π_{IKOS}). We remark that the MPC is used to compute the functionality on s committed values $\phi(x_1, \ldots, x_s)$.

(2) In this stage, P and V execute a coin-tossing to decide a value ch as V's challenge:

  (a) V samples a random string $\mathsf{ch}_1 \overset{\$}{\leftarrow} \{0,1\}^{\mathsf{poly}(\lambda)}$, and commits to ch_1 using the standard ExtCom (Protocol 2.3.1).

  (b) P sends a random string $\mathsf{ch}_2 \overset{\$}{\leftarrow} \{0,1\}^{\mathsf{poly}(\lambda)}$.

  (c) V sends ch_2 along with the corresponding decommitment.

  We remark that both parties agree on a size-t subset $\mathsf{ch} \subset [n]$, determined by the randomness $\mathsf{ch}_1 \oplus \mathsf{ch}_2$.

(3) Same as Step (3) of $\mathsf{IKOS\text{-}Prove}$. We remark that both parties use the ch defined in last step to do the corresponding execution.

on a modified approach of [Lin13].[7] At a high level, we substitute the verifier's challenge ch in the above honest-verifier construction by a coin-tossing between P and V, whose result will be "interpreted" as the verifier's challenge to finish the remaining execution. This coin-tossing step is based on a ExtCom protocol, such that a ZK simulator can extract the verifier's share, thus bias the coin-tossing result to its advantage to finish the execution. In addition, since the coin-tossing happens after the prover's first message, a knowledge extractor can be constructed by rewinding the prover with different queries (which is done by sending different shares in the coin-tossing) to extract the witness, thus obtains AoK property. We show our construction in Protocol 4.4.1.

The security of Protocol 4.4.1 is established by the following lemma.

Lemma 4.4.1. *The protocol $\Pi^{\mathsf{BB}}_{\mathsf{ZKAoK}}$ is a black-box commit-and-prove zero-knowledge argument of knowledge (Definition 4.4.1).*

Proof. First, note that BBCom is a statistically-binding commitment due to the statistical binding property of the underlying Com and the reconstruction guarantees of VSS scheme.

[7]The objective in [Lin13] is to construct a *proof* (of knowledge) system in (optimal) five rounds which requires the stronger assumption of two-round statistically hiding commitments. We can avoid this assumption since we only seek a constant-round argument system

Also, completeness and black-box property are immediate. Soundness follows from the AoK property,[8] which will be proved in the following. The ZK and AoK properties are achieved in the standard manner, using rewinding, to bias the coin-toss and issuing new challenges respectively. A more detailed sketch is provided below:

Zero-Knowledge. Our ZK simulator Sim can be built in the following way. For the commitment stage, Sim commits to 0-strings (i.e. it sets $x_1 = \ldots = x_s = 0^\lambda$). For the proof stage, Sim proceeds as follows: it samples a random challenge $\mathsf{ch}' \subset [n]$, and invokes the Π_{IKOS} simulator HVSim on ch' to get the simulated ZK'_1 and ZK'_3 messages. It then executes the protocol using the honest prover's strategy up to the end of step (2)-(a) of the **Prove Stage**. The simulator first extracts the value ch_1 committed by V^* in ExtCom, and then sends a string ch_2 as the step (2)-(b) message, such that $\mathsf{ch}_1 \oplus \mathsf{ch}_2$ is the randomness deciding the challenge ch'. Finally, it finishes the protocol with ZK'_3 as the step-(3) message. It is obvious that Sim runs in (expected) polynomial time. The indistinguishability of the simulated view and the view from real execution can be proved using standard techniques.

AoK Property. Our knowledge extractor E works during the **Prove Stage**, and can be built as follows. E executes the protocol using the honest verifier's strategy until the end. If P^* gives a convincing proof (otherwise, E aborts), E rewinds P^* to the beginning of step-(2) and finish the protocol from there with fresh randomness. E repeats this procedure until it gets to accepting transcript with different step-(2) coin-tossing results for sufficiently many challenges (e.g., $t + 1$ if that is the threshold of our VSS). By the property of VSS, E can recover the witness x from these (sufficiently many) VSS shares. One caveat here is that P^* may try to bias the coin-tossing results to a single or a small set of challenges so that it manages to successfully complete the proof even if it does not "know" a witness. However, observe that doing so reveals information about the committed value in ExtCom which compromises hiding. $\square$

Remark 4.4.3 (Multi-Proof Extension)**.** *In the* **Prove Stage** *of Protocol 4.4.1, P proves a* single *predicate ϕ. Actually, this scheme allows P to give proofs to (constantly) many predicates w.r.t the same commitments in the* **Commit Stage**. *Notice that the security of this above construction is guaranteed once we set t to be a constant fraction of n.*[9] *To support k (k is some constant) proofs, we simply use a $(kn + 1, kt)$-VSS schemes to run* BBCom. *Later in the* **Prove Stage***, we run the proof for each ϕ sequentially, where we still open t views (i.e. the coin-tossing result* ch *is still a size-t random subset of $[kn]$) in* BBCom *for the proof of each ϕ.*

4.4.1.3 A "Proof-Compatible" Extractable Commitment

As mentioned at the beginning of this section, we want to make our protocol $\langle C, R\rangle_{\mathsf{CCA}}$ (Protocol 4.3.1) black-box using the black-box commit-and-prove scheme to conduct the Stage-0, Stage-3 commitment and Stage-5 proofs. Note that the Stage-3 commitment should

[8]We remark that the standard definition of ZKAoK ([Gol01, Definition 4.7.2]) treats soundness and AoK property separately. Although there are definitions that build soundness directly into the AoK property, treating these properties separately is arguably the better choice (see [BG93]). In the current proof, AoK does imply soundness as our extractor does *not* assume that $x \in \mathcal{L}$.

[9]This is inherited from the original protocol Π_{IKOS}. See the soundness proof in [IKOS07] for more details.

be extractable. But in our commit-and-prove protocol, the commitment BBCom does not support extraction. Therefore, we have to modify the committing stage of $\Pi^{\mathsf{BB}}_{\mathrm{ZKAoK}}$ such that it becomes a extractable commitment, while still being compatible with the (black-box) proving stage. Fortunately, this can be done in the following way.

Observe that in Protocol 4.4.1, if we commit to a single value x in the committing stage, and use the identity predicate which outputs 1 on any *valid* input[10] in the proving stage, this protocol is already an extractable commitment to x. The (statistical) binding property follows from that of $\mathsf{BBCom}(x)$. The computational hiding property follows from ZK. The extractability follows from AoK property. Moreover, as discussed in Remark 4.4.3, we can set the parameters properly such that after the proof of $\phi(x) \equiv 1$ there are still enough unopened views (in $\mathsf{BBCom}(x)$) which can be used to support more proof stages later.

We refer to this commitment scheme as Π_{VSSCom}. For completeness, we provide the full description of Π_{VSSCom} in Protocol 4.4.2. Jumping ahead, we will use Π_{VSSCom} (instead of the standard ExtCom) to instantiate Stage-3 of our protocol.

Our Extractable Commitment Π_{VSSCom}. We show the extractable commitment in Protocol 4.4.2. It makes use of the following primitives:

- A statistically-binding commitment scheme Com;
- An extractable commitment scheme ExtCom;
- $(n+1,t)$-perfectly verifiable secret sharing scheme $\Pi_{\mathsf{VSS}} = (\mathsf{VSS}_{\mathsf{Share}}, \mathsf{VSS}_{\mathsf{Recon}})$
- A t-secure MPC protocol in the malicious model. In fact, we only need th MPC protocol to be t-private in the semi-honest model and (perfectly or statistically) t-robust in the malicious model, as defined in Definition 2.7.5.

Parameter Settings. We will pick our parameters such that t is a constant multiple of the security parameter λ, and n is a constant multiple of t.

Protocol 4.4.2: Black-Box Extractable Commitment Π_{VSSCom} Compatible with Proofs

Input: The committer C and receiver R have the security parameter λ as common input. Additionally, C has as private input a value v which it wishes to commit to.

Commitment Phase: We describe the commitment protocol.

- **Share-and-Commit:** This consists of the following steps: C starts emulating $n+1$ players "in his head". C sets the input of P_{n+1} (i.e., the Dealer) to the value v, while each other player has no input. Then C runs $\mathsf{VSS}_{\mathsf{Share}}$ and each player P_i obtains shares w_i, for any $i \in [n]$. Let $\{\mathsf{View}_1, \ldots, \mathsf{View}_{n+1}\}$ be the views of the $n+1$ players describing the execution of this $\mathsf{VSS}_{\mathsf{Share}}$. C sends commitment $\{\mathsf{cm}_i = \mathsf{Com}(\mathsf{View}_i)\}_{i\in[n]}$.
- **Coin-Tossing:** This consists of the following steps:

  1. R commits to a random string (of proper length) $\mathsf{ch}_1 \xleftarrow{\$} \{0,1\}^{\mathsf{poly}(\lambda)}$ using ExtCom.

[10] I.e., $\phi(x) \equiv 1$ if and only if x is a valid bit string of appropriate size.

2. C sends a random string $\mathsf{ch}_2 \xleftarrow{\$} \{0,1\}^{\mathsf{poly}(\lambda)}$.
3. R sends ch_1 along with the corresponding decommitment.

We remark that both parties agree on a (pseudo) random size-t subset $\mathsf{ch} \subset [n]$, determined by the randomness $\mathsf{ch}_1 \oplus \mathsf{ch}_2$.

– **Response:** C sends $\{\mathsf{View}_i\}_{i \in \mathsf{ch}}$ with the corresponding decommitments information.
– **Receiver's Decision:** the receiver accepts if and only if the following three conditions hold:

  * the committer's decommitments in **Response** stage is consistent with the set ch defined towards the end of **Coin-Tossing** stage;
  * all the decommitments R received are valid, and the dealer P_{n+1} has not been disqualified by any player;
  * all pairs of views in $\{\mathsf{View}\}_{i \in \mathsf{ch}}$ that R received in **Response** stage are consistent (according to Π_{VSS}).

Decommmitment phase: This proceeds as follows:

1. C decommits to $\{\mathsf{cm}_i\}_{i \in [n]}$ as $\{\mathsf{View}_i\}_{i \in [n]}$.
2. R checks that all commitments to the views are opened correctly in the previous step. If not, R outputs $\perp$ and halts.
3. R runs $\mathsf{VSS}_{\mathsf{Recon}}$ using $\{\mathsf{View}_i\}_{i \in [n]}$ as inputs to reconstruct a value v as its output.

4.4.2 Black-Box Instantiation of Our $\mathrm{CCA}^{1:1}$ Commitment

In this subsection, we show how we can instantiate our $\mathrm{CCA}^{1:1}$ protocol $\langle C, R \rangle_{\mathsf{CCA}}$ (Protocol 4.3.1) with the commit-and-prove ZKAoK protocol $\Pi^{\mathsf{BB}}_{\mathsf{ZKAoK}}$ we built in previous subsection, to get a constant-round black-box $\mathrm{CCA}^{1:1}$ commitment $\Pi^{\mathsf{BB}}_{\mathsf{CCA}}$.

This protocol makes use of a $\big((k+2)n+1, (k+2)t\big)$-perfect VSS scheme and a $(k+2)t$-secure MPC protocol against malicious adversaries. The **Stage-1**, **Stage-2** and **Stage-4** of $\Pi^{\mathsf{BB}}_{\mathsf{CCA}}$ are the same as those of $\langle C, R \rangle_{\mathsf{CCA}}$. In the following, we show the modifications to the remaining stages using $\Pi^{\mathsf{BB}}_{\mathsf{ZKAoK}}$:

– **Stage-0:** C commits to the value v using BBCom of $\Pi^{\mathsf{BB}}_{\mathsf{ZKAoK}}$. In the "MPC-in-the-head" execution, C uses $(k+2)n+1$ (imaginary) parties. In addition, C sends the tag id to R in the plain.
– **Stage-3:** C now commits to $\beta = 0^\lambda$ using Π_{VSSCom} (Protocol 4.4.2). We remark that the first stage of Π_{VSSCom} is exactly a BBCom commitment to β. Recall that in **Coin-Tossing** stage of Π_{VSSCom}, a size-t random subset ch is determined such that C will open t views specified in ch. Note that ch is still a fraction-t subset of $[(k+2)n]$ even though we now have $(k+2)n$ committed views.
– **Stage-5:** C and R engage in $k+1$ sequential executions of $\mathsf{BBProve}(\phi)$ of protocol $\Pi^{\mathsf{BB}}_{\mathsf{ZKAoK}}$ over the commit-stage transcripts of the two BBCom corresponding to **Stage-0** and

Stage-3. Predicate ϕ is true if and only if:

1. $c = \mathsf{Com}(v)$; **or**
2. $\beta = \alpha_1$

As remarked in **Stage-3**, here too ch is still a size-t random subset of $[(k+2)n]$ in each of the $k+1$ independent $\mathsf{BBProve}(\phi)$ instances. Note $\mathsf{BBProve}(\phi)$ proves statements over multiple commitments and thus is the multi-commitment extension of $\Pi^{\mathsf{BB}}_{\mathsf{ZKAoK}}$. I.e., C will decommit to both **Stage-0** and **Stage-3** commitments in addition to the views committed in the first-round of $\mathsf{BBProve}(\phi)$.

Remarks on the Security Proof. The security of $\Pi^{\mathsf{BB}}_{\mathsf{CCA}}$ follows from that of $\langle C, R\rangle_{\mathsf{CCA}}$ since each of the primitives we have used achieve the properties necessary in the security proof of $\langle C, R\rangle_{\mathsf{CCA}}$. This is due to the ZK and AoK properties of commit-and-prove protocol $\Pi^{\mathsf{BB}}_{\mathsf{ZKAoK}}$. Indeed, $\Pi^{\mathsf{BB}}_{\mathsf{ZKAoK}}$ is used to instantiate the WIAoK and it is WI due to being ZK.

The modifications to the commitments are also "proper" in the sense that they maintain hiding and binding properties. This is since our choice of parameters guarantees that the number of opened views is no more than $(k+2)t$. In particular, we open $(k+2)t$ views during **Stage-0**, and $(k+1)t$ views during **Stage-3**, each of which is at most $(k+2)t$. Therefore, the ZK property of $\Pi^{\mathsf{BB}}_{\mathsf{CCA}}$ is preserved. Also, in each instance of $\mathsf{BBProve}(\phi)$, we continue to open at least a constant fraction of n committed views, and therefore the AoK property is maintained as well (see Remark 4.4.3).

This completes the proof of the following theorem:

Theorem 4.4.1 (Black-Box 1-1 CCA Commitments)**.** *Assume the existence of one-way functions. Then, there exists a* constant round *black-box construction of a 1-1 CCA secure commitment scheme.*

4.5 Angel-Based MPC in $\widetilde{O}(\log \lambda)$ Rounds

As mentioned in the introduction, the security model that we consider is *angel-based* security, or UC security *with superpolynomial helpers*. Very briefly, this is essentially the same as the UC model used in [Can01], except that the adversary (in the real world) and the environment (in the ideal world) both have access to a superpolynomial time functionality that acts as an oracle or a *helper*. Formal definitions for this security model can be found in [CLP10] and [LP12]. In terms of notation, if there is a protocol Π that emulates a functionality $\mathcal{H}$ with helper $\mathcal{H}$ in this setting, we say that Π $\mathcal{H}$-EUC-realizes $\mathcal{F}$.

Kiyoshima [Kiy14] presents a black-box construction of a CCA-secure commitment scheme with the following ingredients:

– a two-round statistically-binding commitment scheme, and a constant round "strongly extractable" commitment, both of which are known from (black-box) one-way functions;

– a concurrently-extractable commitment (due to Micciancio et al. [MOSV06]) with a "robustness parameter" $\ell = O(R \cdot \log \lambda \cdot \log\log \lambda)$, where R is specified in the next item;

– an R-round 1-1 CCA-secure commitment.

The round-complexity of the resulting protocol is $O(\ell)$.

We first remark that if R is a constant, Kiyoshima's technique will yield a $\widetilde{O}(\log\lambda)$-round CCA-secure commitment. More precisely, Kiyoshima states his results with a specific value of ℓ, namely, $O(\log^2\lambda\cdot\log\log\lambda)$, since $R = O(\log\lambda)$ in his case. However, his construction and proof work for any value of R if ℓ is as described above, i.e., as long as the 1-1 CCA commitment has R rounds, the resulting (fully) CCA commitment will have $O(R\cdot\log\lambda\cdot\log\log\lambda)$ rounds. Next, note that we did construct a black-box 1-1 CCA commitment scheme with round number $R = O(1)$ (Theorem 4.4.1). This yields the Theorem 1.5.3.

Now, as in [Kiy14], we combine Theorem 1.5.3 with the following two results due to Canetti et al. [CLP10, CLP16] and Lin and Pass [LP12] respectively to obtain Theorem 1.5.2, the main theorem of this paper.

Theorem 4.5.1 ([LP12])**.** *Assume the existence of an R_{CCA}-round robust CCA-secure commitment scheme $\langle C, R\rangle$ and the existence of an R_{OT}-round semi-honest oblivious transfer protocol $\langle S, R\rangle$. Then, there is an $O(\max(R_{\mathsf{CCA}}, R_{\mathsf{OT}}))$-round protocol that $\mathcal{H}$-EUC-realizes $\mathcal{F}_{\mathsf{OT}}$. Furthermore, this protocol uses $\langle C, R\rangle$ and $\langle S, R\rangle$ only in a black-box way.*

Theorem 4.5.2 ([CLP10, CLP16])**.** *For every well-formed functionality $\mathcal{F}$, there exists a constant-round $\mathcal{F}_{\mathsf{OT}}$-hybrid protocol that $\mathcal{H}$-EUC-realizes $\mathcal{F}_{OT}$.*

Chapter 5

Toward a Unified Approach to Black-Box Zero-Knowledge Proofs

5.1 Technical Overview

5.1.1 Black-Box Separation

We first present a very brief overview of our black-box separation. A detailed overview is given in Section 5.3.2 after setting up necessary notation and definitions.

Let us first recall how Rosulek [Ros12] rules out FBB constructions of honest-verifier witness-hiding (HVWH) protocols for the range-membership of OWFs, assuming injective OWFs exist.

The proof starts by assuming that such protocols exist. In particular, when instantiated with an injective OWF f, the protocol (P^f, V^f) is HVWH for $\mathcal{R}^f = \{(y, x) \mid \text{s.t. } y = f(x)\}$. Since f is injective, for a pair $(x^*, y^* = f(x^*))$ remapping $f(x^*)$ to a value different from y^* will give us a new OWF f' whose range does not contain y^* anymore. Moreover, the verifier accepts in $\langle P^f(x^*, y^*), V^{f'}(y^*)\rangle$ with roughly the same probability as in $\langle P^f(x^*, y^*), V^f(y^*)\rangle$. This is because the only opportunity to distinguish these two executions is when the verifier queries its oracle on x^*; but this happens with negligible probability because of the HVWH property of the protocol. However, this contradicts the soundness: V's oracle now becomes f', and $\nexists x$ s.t. $(y^*, x) \in \mathcal{R}^{f'}$.

It is unclear how to reuse the above technique to rule out PB-OWFs. As mentioned earlier, there are no restrictions on how the $F^{(\cdot)}$ part behaves. In particular, F^f is not guaranteed to be injective even if f is injective. Thus, "carving out" a value from the range of f may not affect the range of F^f.

To derive the desired contradiction, we take a fundamentally different approach to construct f'. We first define a set Q^{Easy}, which consists of only the queries made by the verifier with "high" probability during the (honest) execution $\langle P^f(x^*, y^*), V^f(y^*)\rangle$. We then define f' by maintaining the same behavior as f on Q^{Easy}, *and* re-sampling all the remaining points uniformly at random. Note that the receiver will still accept with "high" probability even if we change its oracle to f', because f' and f only differ at the points that are queried with "low" probability (i.e., the points outside Q^{Easy}). Now, the only thing left is to show that y^* is not in the range of $F^{f'}$. Unfortunately, due to the generality of $F^{(\cdot)}$, we do not know how to do that.

However, if we switch our focus to a PB-PRG G^f (instead of PB-OWF F^f), we can prove the following claim which helps separate PB-PRGs from OWFs:

Claim 5.1.1 (Informal Statement of Claim 5.3.3)**.** *Let f, and f' be as defined above. If we start with a y^* in the range of G^f, y^* is still in the range of $G^{f'}$ with probability* $\leq 0.5+\mathsf{negl}(\lambda)$, *where $G^{(\cdot)}$ is a PRG when instantiated with any OWF f as its oracle.*

Let us show the intuition behind Claim 5.1.1. Assume the lemma is false. In the pseudo-randomness game for G^f, we show how to identify the case $y^* \in \mathsf{Range}(G^f)$ *correctly*, with probability noticeably greater than 0.5, thus contradicting pseudo-randomness. To do that, the adversary simply estimates the probability that $y^* \in \mathsf{Range}(G^{f'})$. We will show that, if this probability is noticeably greater than 0.5, then $\Pr[y^* \in \mathsf{Range}(G^f)]$ is also noticeably greater than 0.5.

To perform the above estimation successfully, the adversary must know Q^{Easy}, but it does not. However, the adversary can run the HVZK simulator many times to get an estimate $\widetilde{Q}^{\mathsf{Easy}}$ for the real Q^{Easy}. We will show that $\widetilde{Q}^{\mathsf{Easy}}$ suffices for our proof. There is a caveat that the adversary needs to perform exponential work to estimate the probability that $y^* \in \mathsf{Range}(G^{f'})$, even if it knows the set $\widetilde{Q}^{\mathsf{Easy}}$. Fortunately, it only makes *polynomially many* oracle queries (when executing the HVZK simulator), which suffices for establishing a *fully*-black-box separation.

5.1.2 Proof-Based One-Way Functions (and PRGs)

Let us start by considering the following basic construction for PB-OWF (F^f, Π_F^f) over inputs of the form (x, r). The construction is based on the "cut-and-choose" technique, where the sender queries the oracle f on "blocks" of x, and the receiver checks a size-t random subset (defined by r) of the responses. This method is not sound, since it can only guarantee that the sender's response is correct on most but not all blocks. We will handle this issue by introducing a new idea.

Basic Construction. PB-OWF (F^f, Π_F^f) handles inputs of the form (x, r). F^f computes as follows:

1. Parse x as $(x_1, \ldots, x_n)$.
2. Interpret r as a size-t $(t < n)$ subset of $[n]$, denoted by $\{b_1, \ldots, b_t\}$.
3. Output $\mathsf{y} = (y_1, \ldots, y_n)\|(x_{b_1}, \ldots, x_{b_t})\|\mathsf{r}$, where $y_i = f(x_i)$ for all $i \in [n]$.

On input x to S^f (the sender) and r to R^f (the receiver)[1], the execution $\langle S^f(\mathsf{x}), R^f(\mathsf{r})\rangle$ is as follows:

1. S^f parses x as $(x_1, \ldots, x_n)$, and computes $(y_1, \ldots, y_n)$ via its oracle access to f (i.e., $y_i = f(x_i)$). It sends $(y_1, \ldots, y_n)$ to the receiver.

[1]Henceforth, we will call the two parties "sender" and "receiver" instead of "prover" and "verifier". This is because our ZKP is captured by considering a secure computation style definition for two parties.

2. R^f sends its input r to S^f. Same as in F^f, the r specifies a size-t subset $\{b_1, \ldots, b_t\}$ of $[n]$. Recall that the honest receiver's input r is random. In this case, $\{b_1, \ldots, b_t\}$ is a *random* subset of $[n]$.
3. S^f sends $(x_{b_1}, \ldots, x_{b_t})$, i.e., the x_i's whose indices are specified by r.
4. R^f checks (via its oracle access to f) if $y_{b_i} = f(x_{b_i})$ for all $i \in [t]$. If all the checks pass, R^f output $\mathsf{y} = (y_1, \ldots, y_n)\|(x_{b_1}, \ldots, x_{b_t})\|\mathsf{r}$.

Completeness is straightforward; furthermore, $F^f(\cdot, \mathsf{r})$ is trivially one-way for every r since $t < n$ and f is a OWF. Let us first consider the *honest-verifier zero-knowledge* (HVZK) property of protocol Π^f_F.

Recall that the ZK property is defined via the ideal/real paradigm for secure computation, and it requires simulation-security against malicious receivers. Thus, to prove the HVZK property, we need to show an ideal-world simulator Sim for the honest receiver. This is easy since the honest receiver will always use the given input r, which is uniformly distributed. More specifically, Sim works by sampling a uniform r by itself, sending the r to the ideal functionality, and receiving in turn the output $\mathsf{y} = (y_1, \ldots, y_n)\|(x_{b_1}, \ldots, x_{b_t})\|\mathsf{r}$. With this y, Sim can easily generate a simulated transcript that is identically distributed to the real one.

ZK Against Malicious Receivers. The above simulation strategy does not work for malicious receivers, because they may not use the given input r. Therefore, the simulator needs to somehow extract the candidate input r^* from the malicious receiver. However, the receiver will not give out its r^* until the sender/simulator sends the $\{y_i\}_{i \in [n]}$ values.

We point out that this issue cannot be fixed using standard methods such as requiring the receiver to commit to r and to open it later. This is because later, we will introduce a pre-image editing condition and require that the sender's computation of F^f be consistent with this editing.

We therefore use a different idea. We modify the protocol to use a black-box commit-and-prove scheme $\Pi_{\mathsf{ZKCnP}} = (\mathsf{BBCom}, \mathsf{BBProve})$ with ZK property (see Definition 5.2.2). This scheme has a pair of simulators $(\mathsf{Sim}_1, \mathsf{Sim}_2)$ that can be used to simulate the receiver's view in the commit stage and the prove stage respectively. Our new Π^f_F is the same as before except for the following changes:

- In Step 1, instead of sending y_i's as before, the sender commits to them using BBCom. Formally, the sender sets $\nu = (y_1, \ldots, y_n)$ and executes $\mathsf{BBCom}(\nu)$ with the receiver.
- In Step 3, the sender sends both $\{x_{b_i}\}_{i \in [n]}$ *and* the value ν. It then proves using $\mathsf{BBProve}$ that this ν is indeed the value committed in BBCom.

As before, the receiver needs $(y_1, \ldots, y_n)$ to execute Step 4. Now, these values are contained in ν, and $\mathsf{BBProve}$ guarantees that the sender cannot change ν.

With these modifications, we can prove the ZK property for malicious receivers as follows. The simulator starts by running Sim_1 (the commit-phase simulator) with the malicious receiver R^{*f}. In this way, the simulator can go through Step 1 smoothly, without knowing the actual $\{y_i\}_{i \in [n]}$ values. Then, it will receive the r^* from R^{*f}. The simulator sends r^* to the ideal functionality and receives back $\mathsf{y} = (y_1, \ldots, y_n)\|(x_{b_1}, \ldots, x_{b_t})\|\mathsf{r}^*$. It sends $\nu = (y_1, \ldots, y_n)$ and $(x_{b_1}, \ldots, x_{b_t})$ to the receiver. Then, instead of executing $\mathsf{BBProve}$, the simulator invokes Sim_2 to help itself go through the $\mathsf{BBProve}$ stage. It is easy to see that the ν and $\{x_{b_i}\}_{i\in[t]}$ sent by the simulator meet the consistency requirement in Step 4. Relying on the ZK property of Π_{ZKCnP}, one can formally prove that the simulation is done properly.

Soundness and Preimage Editing. As mentioned earlier, the "cut-and-choose" structure is not sufficient to guarantee the existence of a preimage. To see that, consider a malicious sender who picks an $i^* \in [n]$ at random, sets y_{i^*} to some value not in the range of f, or behaves honestly otherwise. This malicious sender can still make the honest receiver accept with non-negligible probability, even if t is as large as $n-1$ (the upper bound for t to achieve any non-trivial ZK property). This is addressed by modifying the construction of F^f.

We start by noting that the "cut-and-choose" trick ensures that most of the y_i's are "good" (i.e., having preimages under f). For example, if t is a constant fraction of n, then the protocol ensures (except with negligible probability) that at most k of the y_i's are "bad", where k is another constant fraction of n. Therefore, our idea is to extend the range of F^f to include all the images y that have $\le k$ bad y_i's. More specifically, our new F^f works as follows. On input (x,r), it still interprets r as $\{b_1,\ldots,b_t\}$. But it will parse x as

$$\mathsf{x} = (x_1,\ldots,x_n)\|\underbrace{(p_1,y'_{p_1}),\ldots,(p_k,y'_{p_k})}_{\beta},$$

where the $\{p_1,\ldots,p_k\}$ form a size-k subset of $[n]$. The evaluation of $F^f(\mathsf{x},\mathsf{r})$ consists of two cases:

- **Non-Editing Case:** if $\{b_1,\ldots,b_t\}\cap\{p_1,\ldots,p_k\}\neq\emptyset$, then it computes y as before, ignoring the β part. That is, it outputs $\mathsf{y}=(s_1,\ldots,s_n)\|(x_{b_1},\ldots,x_{b_t})\|\mathsf{r}$, where $s_i=y_i$ for all $i\in[n]$.
- **Editing Case:** if $\{b_1,\ldots,b_t\}\cap\{p_1,\ldots,p_k\}=\emptyset$, then at positions specified by p_i's, it replace y_{p_i} with y'_{p_i}. Namely, it outputs $\mathsf{y}=(s_1,\ldots,s_n)\|(x_{b_1},\ldots,x_{b_t})\|\mathsf{r}$, where $s_i :=$ $\begin{cases} y'_i & i\in\{p_1,\ldots,p_k\} \\ y_i & i\in[n]\setminus\{p_1,\ldots,p_k\}\end{cases}$.

Let us explain how this editing technique resolves the soundness issue. Consider a y^* learned by the honest receiver with input r. As mentioned before, there are at most k y_i values (among those contained in y^*) that do not have preimages under f. These values can be expressed as $\{y^*_{p_1},\ldots,y^*_{p_k}\}$, i.e., their indices are $\{p_1,\ldots,p_k\}$. Moreover, this set of bad indices does not overlap with the $\{b_1,\ldots,b_t\}$ specified by r; otherwise, the receiver would abort when performing the checks in Step 4. Therefore, by setting the β part to $(p_1,y^*_{p_1}),\ldots,(p_k,y^*_{p_k})$, we will obtain a valid preimage for y^* under our new $F^f(\cdot,\mathsf{r})$.

One may wonder whether a malicious sender can cheat by taking advantage of the **editing** case. However, since the honest receiver will use a random r, the set $\{b_1,\ldots,b_t\}$ will always overlap with $\{p_1,\ldots,p_k\}$ (except with negligible probability). That is, although we prove soundness by relying on the **editing** case, it almost never happens in a real execution. So, this will not give malicious senders any extra power.

We remark that the above preimage-editing idea is compatible with our technique for achieving (full) ZK. Now, the sender will append $(y_1,\ldots,y_n)$ and β to the committed value

ν. Upon receiving R^f's challenge r, the sender computes $\mathsf{s} = (s_1, \ldots, s_n)$ according to the above definition of F^f. It sends both s and $\{x_{b_1}, \ldots, x_{b_t}\}$ to the receivers. Then, it runs $\mathsf{BBProve}$ to prove that it does the editing (or non-editing) honestly. Note that this statement can be expressed as a predicate on the values s, r, β, and $\{y_i\}_{i\in[n]}$, where the last two are committed in $\mathsf{BBCom}(\nu)$. Since it does not involve the code of f, the protocol remains black-box in f. We provide more details in Section 5.4.2.

Proof-Based PRGs. Following the above paradigm, we also obtain a proof-based PRG by simply replacing the oracle OWF f with a PRG in the above PB-OWF construction. We provide a formal treatment in Section 5.5.

5.1.3 Proof-Based Collision-Resistant Hash Functions

Recall that a PB-CRHF consists of a function H^h and a protocol Π^h_H such that for any CRHF h:

- For all r, $H^h(\cdot, \mathsf{r})$ is a CRHF; **and**
- $\Pi^h_H = (S^h, R^h)$ is protocol satisfying similar completeness, soundness, and ZK properties as for our PB-OWFs, but w.r.t. H^h.

Let us first try to reuse the idea from our PB-OWFs. On input (x, r), the H^h first parses x as $(x_1, \ldots, x_n)\|\beta$, where the β has the same structure as before, for the purpose of preimage editing. It then generates $\{y_i\}_{i\in[n]}$ where $y_i = h(x_i)$, and outputs $\mathsf{y} = \mathsf{s}\|(x_{b_1}, \ldots, x_{b_t})\|\mathsf{r}$, where the value $\mathsf{s} = (s_1, \ldots, s_n)$ is computed by editing $\{y_i\}$ (in the same way as for our PB-OWFs).

Since h is also a OWF, the H^h is surely one-way. However, it is not collision-resistant. To see that, recall that in the **non-editing** case, the β part is not used when computing $H^h(\mathsf{x}, \mathsf{r})$. This implies the following collision-finding attack. For a fix r, the adversary first computes $\mathsf{y}^* = H^h(\mathsf{x}^*, \mathsf{r})$ with an x^* whose β part does *not* trigger the **editing** condition. Then, it can easily find many preimages for y^* by using different β's as long as they do not trigger the **editing** condition. Therefore, we need to come up with a new editing method that does not compromise collision resistance.

To do that, we modify H^h as follows. We sample a public string z and hard-wire it in H^h. In this way, H^h_z can be viewed as a member of the public-coin collision-resistant hash *family* indexed by z instead of a single CRHF. Then, we can think of x as containing additionally two strings τ and μ. When evaluating $H^h_z(\mathsf{x}, \mathsf{r})$, we will perform the editing if $\{b_1, \ldots, b_t\} \cap \{p_1, \ldots, p_k\} = \emptyset$ *and* $\alpha \neq z$ *and* $h(\tau) = h(z)$. Moreover, we include the value $\mathsf{t} = h(\beta\|\tau\|\mu)$ in the output y. Intuitively, this hash of β in y prevents the adversary from constructing collisions using a different β.

We now explain how to perform editing in this setting. First, we will include in x an additional value τ such that $\tau \neq z$ and $h(\tau) = h(z)$. This allows us to trigger the **editing** condition. With z sampled randomly, it is not hard to see that such a τ exists with overwhelming probability[2]. We can then set β as before to ensure that the $(x_1, \ldots, x_n)$ part

[2]This holds if the size of range of h is exponentially larger than its image space. It is also worth noting that τ does not need to be efficiently computable, because our soundness proof (or the editing technique) is

is "edited" properly. However, note that the y^* here contains additionally a t^* value. To handle this, we modify the construction of H_z^f slightly—We require that, when the **editing** condition is triggered, H_z^f sets $\mathsf{t} = \mu$ in its output y. With this change, when performing editing, we can simply let μ equal the t^*. It is not hard to verify that this editing technique will lead to a valid preimage for y^*.

Finally, we remark that our real construction uses a Merkle tree for the prefix $(x_1, \ldots, x_n)$ of x. We only put the Merkle root in y instead of the element-wise hash values described above. The soundness can be proved following essentially the same idea as above, except that we now "edit" the Merkle tree, which is done by extending the editing ideas to the tree setting. This allows us to compress a prefix of any length to a *fixed-length* string, such as 256 bits if using SHA256 for h. We refer the reader to Section 5.6 for a formal treatment of PB-CRHF.

5.1.4 Supporting Predicates

We discuss how to extend our constructions using "MPC-in-the-head" to additionally guarantee not only that the output learned by the receiver is in the range of the deterministic primitives, but also that the set of preimages contains one whose prefix satisfies some predicate ϕ.

Let us take a fresh look at the PB-OWF construction. It first parses the input as $\mathsf{x} = \alpha \| \beta$. The β is for preimage editing; and the $\alpha = (x_1, \ldots, x_n)$ can be regarded as a form of **Encoding** the prefix of x, i.e. $\mathsf{Enc}(\alpha) = (x_1, \ldots, x_n)$. Then, it computes $y_i = f(x_i)$ for all $i \in [n]$. Since this is mainly to introduce hardness (or one-wayness) to the final output, we can refer to this step as **Hardness Inducing**.

To support the proof of a predicate ϕ, we update the construction with new **Encoding** and **Hardness Inducing** methods. We first secret-share α to $([\alpha]_1, \ldots, [\alpha]_n)$ using a verifiable secret sharing (VSS) scheme. This can be viewed as a new encoding method: $\mathsf{Enc}(\alpha) = \mathsf{VSS}(\alpha) = ([\alpha]_1, \ldots, [\alpha]_n)$.

Next, we commit to these shares using Naor's commitment [Nao90], which can be built from the oracle OWF f in a black-box manner. This can be thought of as a new **Hardness Inducing** method. Now, the output of F^f is of the following form:

$$F^f(\mathsf{x}, \mathsf{r}) = (\mathsf{Com}([\alpha]_1), \ldots, \mathsf{Com}([\alpha]_n)) \| ([\alpha]_{b_1}, \ldots, [\alpha]_{b_t}) \| \mathsf{r}.$$

In the protocol Π_F^f, we additionally ask the sender to compute the value $\phi(\alpha)$ using the MPC-in-the-head technique. That is, the sender imagines n virtual parties $\{P_i\}_{i \in [n]}$, where P_i has $[\alpha]_i$ as its input. These n parties then execute a MPC protocol w.r.t. to the ideal functionality, which recovers α from the VSS shares, and outputs $\phi(\alpha)$ to each party. Let v_i denote the view of party i from the execution. The sender first commits to these views, and then opens some of them (picked by the receiver) for the receiver to check that the MPC for $\phi(\alpha)$ was performed honestly. In this way, the receiver not only learns $\phi(\alpha)$, but also believes that the sender did not cheat.

Finally, we make a few remarks:

only an existential argument.

– To achieve soundness, we also need to apply the preimage editing idea to the above construction.

– Both the VSS and Com require randomness, which can come from x. That is, we require that the x is long enough such that it also contains an η part (in addition to α and β). This η will provide the randomness for VSS and Com.

– The above approach applies directly to the PB-PRG and PB-CRHF constructions to make them support predicates on the α part of the preimage.

5.2 Preliminaries

In the following, we present additional preliminaries that are necessary for this chapter.

5.2.1 Naor's Commitment Schemes

We will use Naor's two-round commitment scheme [Nao90], which is computationally-hiding and statistically-binding. It can be constructed from any OWFs in a black-box manner. The original scheme is for committing a single bit. But it can be extended to multi-bit messages by applying it bit-wise. We recall how this works. To commit a single bit b, R sends a string $\rho \in \{0,1\}^{3\lambda}$ to S. Then S picks a random seed $s \xleftarrow{\$} \{0,1\}^{\lambda}$ and applies a pseudo-random generator $\mathsf{PRG}(\cdot)$ (which can be constructed from any OWFs in a black-box way). If $b = 0$, S sends $\mathsf{cm} = \mathsf{PRG}(s)$; if $b = 1$, S sends $\mathsf{cm} = \mathsf{PRG}(s) \oplus \rho$.

5.2.2 Collision-Resistant Hash Families

We use the definition from [HR04].

Definition 5.2.1 (Collision-Resistant Hash Families). *A private-coin collision-resistant hash family (CRHF) is a collection of functions $\mathcal{H} = \{h_i\}_{i \in I}$ for some index set I, where $h_i : \{0,1\}^{\ell(|i|)} \to \{0,1\}^{\ell'(|i|)}$ and $\ell(|i|) > \ell'(|i|)$. It satisfies the following requirements:*

– **Key Generation.** *There exists a PPT key generating algorithm KGen, so that $\mathsf{KGen}(1^\lambda) \in \{0,1\}^{m(\lambda)} \cap I$, where $m(\lambda)$ is a polynomial on λ representing the length of the key.*

– **Efficient Evaluation.** *There exists a (deterministic) polynomial time algorithm Eval such that $\forall i \in I$ and $\forall x \in \{0,1\}^{\ell(|i|)}$, $\mathsf{Eval}(i, x) = h_i(x)$.*

– **Non-Uniform Collision Resistance.** *For any non-uniform PPT machine $\mathcal{A}$, the following holds:*

$$\Pr[i \xleftarrow{\$} \mathsf{KGen}(1^\lambda), (x, x') \leftarrow \mathcal{A}(i) : x \neq x' \wedge h_i(x) = h_i(x')] \leq \mathsf{negl}(\lambda).$$

Remark 5.2.1 (Public-Coin CRHFs). *The CRHFs defined above is private-coin. In the above definition, if the index set I is $\{0,1\}^*$ and $\mathsf{KGen}(1^\lambda)$ outputs a uniformly distributed string from $\{0,1\}^{m(\lambda)}$, then we say that it is a* public-coin *CRHF, i.e., the family remains collision-resistant even if the randomness used to generate the key is known to the adversary.*

Also, we emphasize that the collision-resistance property holds against non-uniform *PPT adversaries.*

5.2.3 Black-Box Zero-Knowledge Commit-and-Prove

We need a zero-knowledge commit-and-prove protocol Π_{ZKCnP} with the following additional properties:

– it consists of two *separate* phases: a **Commit** stage BBCom and a **Prove** stage $\mathsf{BBProve}$;

– the **Commit** stage itself constitutes a statistically-binding commitment scheme;

– for a public predicate $\phi(\cdot)$, the **Prove** stage constitutes a zero-knowledge argument for the value $\phi(x)$, where x is the value committed in BBCom;

– Π_{ZKCnP} can be constructed assuming only black-box access to OWFs.

We present the formal definition in Definition 5.2.2. We remark that this definition is very similar to Definition 4.4.1, except that we do not need the argument of knowledge property any more. For self-containment, we present the version used in this chapter in Definition 5.2.2. There exist constructions satisfying this definition while making only black-box use of OWFs (e.g., [IKOS07, GLOV12, CLP20a]).

Definition 5.2.2 (Zero-Knowledge Commit-and-Prove)**.** *A black-box zero-knowledge commit-and-prove scheme consists of a pair of protocols* $\Pi_{\mathsf{ZKCnP}} = (\mathsf{BBCom}, \mathsf{BBProve})$ *executed between a pair of* PPT *machines* P *and* V*. These protocols are executed in the following phases:*

– **Commit Stage:** P *and* V *invoke* $\mathsf{BBCom}(x)$ *such that at the end of this protocol,* P *is statistically committed to the value* x*. We use* τ *to denote the transcript from* $\mathsf{BBCom}(1^\lambda, x)$ *execution.* P *stores private state* ST*.*

– **Prove Stage:** P *and* V *take the transcript* τ *and a predicate* ϕ *as common input.* P *takes* ST *as its private input.* P *proves to* V *using* $\mathsf{BBProve}(1^\lambda, \phi)$ *that there exists some value* x *such that* τ *is a valid commitment to* x*, and also* $\phi(x) = 1$*.*

We require that the following properties are satisfied:

– **Black-Box.** *Both phases only require black-box access to cryptographic primitives.*

– **Committing.** *The* **Commit** *stage* $\mathsf{BBCom}(x)$ *constitutes a statistically-binding commitment to the value* x*.*

– **Completeness.** *If* P *and* V *are honest and* $\phi(x) = 1$*, then* V *accepts the proof with probability 1.*

– **Soundness.** *For any PPT prover* P^**, the following holds except with negligible probability:* V *accepts only if* $\phi(x) = 1$*, where* x *is the value to which the* **Commit** *stage is statistically bound.*

– **Zero-Knowledge.** *There exists a pair of* PPT *machines* $(\mathsf{Sim}_1, \mathsf{Sim}_2)$ *satisfying the following requirement. Given oracle access to a machine* V^**,* $\mathsf{Sim}_1^{V^*}(1^\lambda)$ *generates a transcript*

$\widetilde{\tau}$ and stores a private state ST; then for any predicate ϕ, $\mathsf{Sim}_2^{V^*}(1^\lambda, \phi, \mathsf{ST})$ generates a transcripts View. For every PPT verifier V^*, every $z \in \{0,1\}^*$, every PPT predicate ϕ and every x s.t. $\phi(x) = 1$, it holds that

$$\left\{\left(\mathsf{Sim}_1^{V^*}(1^\lambda),\ \mathsf{Sim}_2^{V^*}(1^\lambda, \phi, \mathsf{ST}, z)\right)\right\}_{\lambda\in\mathbb{N}} \overset{\text{c}}{\approx} \{\mathsf{View}_{V^*}^{P(x)}(1^\lambda, \phi, z)\}_{\lambda\in\mathbb{N}},$$

where the ST is the private state of $\mathsf{Sim}_1^{V^*}(1^\lambda)$ at termination, and $\mathsf{View}_{V^*}^{P(x)}(1^\lambda, \phi, z)$ denotes the view of $V^*(1^\lambda, \phi, z)$ in both **Commit** *stage and* **Prove** *stage, resulted from its interaction with $P(1^\lambda, x, \phi)$, where both parties learn ϕ at the beginning of* **Prove** *stage. We will refer to Sim_1 as the* **Commit***-stage simulator, and Sim_2 the* **Prove***-stage simulator.*

5.2.4 The One-Oracle Separation Paradigm

We first recall in Definition 5.2.3 the notion of *fully-black-box* reductions. We say that P cannot be obtained from Q in a fully-black-box way if there is no fully-black-box reduction from Q to P.

Definition 5.2.3 (Fully-Black-Box Reductions [RTV04])**.** *There exists a fully-black-box reduction from a primitive Q to a primitive P, if there exist PPT oracle machines G and S such that:*

- **Correctness:** *For every (possibly-inefficient) f that implements P, G^f implements Q;*
- **Security:** *For every (possibly-inefficient) f that implements P and every (possibly-inefficient) machine $\mathcal{A}$, if $\mathcal{A}$ breaks G^f (w.r.t. Q-security), then $S^{\mathcal{A},f}$ breaks f (w.r.t. P-security).*

A useful paradigm to rule out fully-black-box constructions is to design an oracle $\mathcal{O}$, and show that, relative to $\mathcal{O}$, primitive P exists but Q does not. A critical step in this proof is to construct an oracle machine $\mathcal{A}^{\mathcal{O}}$ that breaks the security of Q. We emphasize that $\mathcal{A}$ is allowed to be *computationally unbounded* as long as it only makes polynomially-many queries to $\mathcal{O}$ (see e.g., [IR89, BM17]). Our fully-black-box separation results in Section 5.3 will follow this paradigm.

5.3 The Impossibility Results

5.3.1 Meta-Functionally Black-Box Constructions

Functionally Black-Box Protocols. To capture MPC protocols that "do not know" the code of the target function g, Rosulek [Ros12] proposed the following notion of functionally-black-box protocols.

Definition 5.3.1 (Functionally-Black-Box Protocols [Ros12])**.** *Let $\mathcal{C}$ be a class of functions, and let $\mathcal{F}^{(\cdot)}$ be an ideal functionality that is an (uninstantiated) oracle machine. Let $A^{(\cdot)}$ and $B^{(\cdot)}$ be PPT interactive oracle machines. Then, we say that $(A^{(\cdot)}, B^{(\cdot)})$ is a functionally-black-box (FBB) protocol for $\mathcal{F}^{\mathcal{C}}$ in a certain security model if, for all $g \in \mathcal{C}$, the protocol (A^g, B^g) is a secure protocol (in the model in question) for the ideal functionality $\mathcal{F}^g$.*

By instantiating $\mathcal{C}$ and $\mathcal{F}^{(\cdot)}$ properly, Definition 5.3.1 could capture black-box constructions of many useful cryptographic protocols. For example, let $\mathcal{C}_{\text{OWF}}$ be the collection of OWFs. For any $g \in \mathcal{C}_{\text{OWF}}$, let $\mathcal{F}^g_{\text{ZK}}$ be the functionality that takes input x from party A, queries its oracle g to obtain $y = g(x)$, and outputs y to party B. Such an $\mathcal{F}^g_{\text{ZK}}$ is essentially a zero-knowledge argument (of knowledge) functionality for statements of the form "$\exists x$ s.t. $g(x) = y$". However, Rosulek showed that if injective OWFs exist, then it is impossible to have FBB protocols that implement $\mathcal{F}^{\mathcal{C}_{\text{OWF}}}_{\text{ZK}}$ with semi-honest security (in the standard MPC setting), even in the presence of an arbitrary trusted setup. Given the broad application of ZK proofs, this result is quite discouraging.

Meta-FBB Functionalities. Observe that the above $\mathcal{F}^g_{\text{ZK}}$ functionality simply collects input x from A, queries its oracle g, and sends $g(x)$ to B. It only plays the role of a delegate for A and B to interact with the OWF g. Therefore, it is temping to investigate whether we can circumvent Rosulek's lower bound by allowing the "delegate" $\mathcal{F}_{\text{ZK}}$ to perform extra computations, such as preprocessing x, post-processing $g(x)$, or making multiple queries to the oracle g, etc.

More formally, we want a non-cryptographic and deterministic computation F (used to capture the aforementioned extra computations), such that $\mathcal{C}'_{\text{OWF}} = \{F^g \mid g \in \mathcal{C}_{\text{OWF}}\}$ is a collection of OWFs. And we hope that there exists a FBB protocol (A^g, B^g) implementing $\mathcal{F}^{F^g}_{\text{ZK}}$ for all $F^g \in \mathcal{C}'_{\text{OWF}}$ (we can also denote it as $\mathcal{F}^{\mathcal{C}'_{\text{OWF}}}_{\text{ZK}}$). Note that we require $(A^{(\cdot)}, B^{(\cdot)})$ to access g in a black-box way only; they can make use the code of F. Since $\mathcal{C}'_{\text{OWF}}$ is also a collection of one-way families, $\mathcal{F}^{\mathcal{C}'_{\text{OWF}}}_{\text{ZK}}$ can be used as a substitute for $\mathcal{F}^{\mathcal{C}_{\text{OWF}}}_{\text{ZK}}$, with the only overhead coming from the computations represented by $F^{(\cdot)}$. Because $F^{(\cdot)}$ is supposed to contain only simple non-cryptographic operations, the implementation of $\mathcal{F}^{\mathcal{C}'_{\text{OWF}}}_{\text{ZK}}$ should be as efficient as that of $\mathcal{F}^{\mathcal{C}_{\text{OWF}}}_{\text{ZK}}$. Therefore, if this approach is possible, it will alleviate the negative implications of Rosulek's lower bound.

We can also interpret $\mathcal{F}^{\mathcal{C}'_{\text{OWF}}}_{\text{ZK}}$ as a new FBB functionality $\mathcal{F}^{\mathcal{C}_{\text{OWF}}}_{\text{ZK}}[F]$, i.e., a new oracle machine $\mathcal{F}^{(\cdot)}_{\text{ZK}}[F]$ to be instantiated with oracle OWFs from the original collection $\mathcal{C}_{\text{OWF}}$. For any $g \in \mathcal{C}_{\text{OWF}}$, $\mathcal{F}^g_{\text{ZK}}[F]$ collects the input X from Party A, evaluates $F^g(X)$, and sends $y = F^g(X)$ to Party B.

With this interpretation, $\mathcal{F}^{\mathcal{C}'_{\text{OWF}}}_{\text{ZK}}$ is just an instantiation of Definition 5.3.1 with $\mathcal{F}^{(\cdot)} = \mathcal{F}^{(\cdot)}_{\text{ZK}}[F]$ and $\mathcal{C} = \mathcal{C}_{\text{OWF}}$. To distinguish with Rosulek's $\mathcal{F}^{(\cdot)}_{\text{ZK}}$ functionality. We call $\mathcal{F}^{(\cdot)}_{\text{ZK}}[F]$ the *Meta-FBB* ZK Functionality. Similarly, one can also extend other FBB functionalities in [Ros12] (e.g., 2-party secure function evaluation $\mathcal{F}^{(\cdot)}_{\text{SFE}}$, pseudo-random generator $\mathcal{F}^{(\cdot)}_{\text{PRG}}$, where sender A holds the seed and receiver B holds the key) to the corresponding Meta-FBB version.

5.3.2 The Main Theorem

In this part, we show that although we relax Rosulek's FBB notion to the Meta-FBB one, there still exists strong impossibility result. More specifically, we prove that, given only black-box access to OWFs, it is impossible to build a PRG that admits Meta-FBB honest-verifier zero-knowledge protocols.

Definition 5.3.2 (Fully-Black-Box PRGs from OWFs)**.** *Let $\mathcal{C}$ be the collection of OWFs.*

A (deterministic) polynomial-time oracle machines $G^{(\cdot)}$ is a fully-black-box *construction of PRG from OWF if there exists a PPT oracle machines $\mathcal{A}^{(\cdot,\cdot)}$ such that:*

- **Correctness:** *$\forall f \in \mathcal{C}$, G^f is a PRG;*
- **Security:** *$\forall f \in \mathcal{C}$ and every (possibly inefficient) machine M, if M breaks the pseudorandomness of G^f, then $\mathcal{A}^{M,f}$ breaks the one-wayness of f.*

Theorem 5.3.1 (Main Theorem)**.** *Let $\mathcal{C} = \{f \mid f \text{ is a OWF}\}$. There does not exist a (deterministic) oracle machine $G^{(\cdot)}$ such that*

1. *$G^{(\cdot)}$ is a fully-black-box construction of PRG from OWF;* **and**
2. *for all $f \in \mathcal{C}$, there exists a stand-alone, Meta-FBB, honest-verifier zero-knowledge argument system $\Pi^f = \langle P^f, V^f \rangle$ for the functionality $\mathcal{F}^f_{\text{ZK}}[G]$.*

Before showing the full proof in Section 5.3.3, let us provide the high-level idea.

Proof Overview. We start by assuming (for contradiction) that the $G^{(\cdot)}$ and $\Pi^{(\cdot)}$ specified in the theorem exist. We will construct a special oracle denoted as $\mathsf{O} \diamond Q^{\mathsf{Easy}}$ (explained later) such that:

1. The oracle $\mathsf{O} \diamond Q^{\mathsf{Easy}}$ is one-way. Thus, $G^{\mathsf{O}\diamond Q^{\mathsf{Easy}}}$ will be a PRG and $\Pi^{\mathsf{O}\diamond Q^{\mathsf{Easy}}}$ will be the HVZK system for the language $\mathcal{L} = \{Y : \exists X \ s.t. \ Y = G^{\mathsf{O}\diamond Q^{\mathsf{Easy}}}(X)\}$.
2. There exist a $\ddot{Y} \notin \mathcal{L}$ (the false statement) and a $\mathbb{P}^{\mathsf{O}\diamond Q^{\mathsf{Easy}}}$ (the cheating prover $\mathbb{P}$ with the oracle $\mathsf{O} \diamond Q^{\mathsf{Easy}}$) that is able to make $V^{\mathsf{O}\diamond Q^{\mathsf{Easy}}}(\ddot{Y})$ accept.

This will give us the desired contradiction as it breaks the soundness of the protocol $\Pi^{\mathsf{O}\diamond Q^{\mathsf{Easy}}}$.

Toward the above goal, we first sample two random oracles O, O', a random string X, and compute $Y = G^{\mathsf{O}}(X)$. Let $Q = \{(q_1, \mathsf{O}(q_1)), \ldots, (q_t, \mathsf{O}(q_t))\}$ denote the query-answer pairs exchanged between G and its oracle O during computation $Y = G^{\mathsf{O}}(X)$. We now define the oracle $\mathsf{O}' \diamond Q(q) := \begin{cases} \mathsf{O}(q) & \text{if } (q, \mathsf{O}(q)) \in Q \\ \mathsf{O}'(q) & \text{otherwise} \end{cases}$. It is not hard to verify that $Y = G^{\mathsf{O}'\diamond Q}(X)$. By completeness, V will accept with probability $1 - \delta_c$ (where δ_c is the completeness error) in the execution $\mathsf{Exec}^{\mathsf{O}'\diamond Q}_{X,Y} = \langle P^{\mathsf{O}'\diamond Q}(X,Y), V^{\mathsf{O}'\diamond Q}(Y)\rangle$.

Note that during $\mathsf{Exec}^{\mathsf{O}'\diamond Q}_{X,Y}$, the verifier may make queries to its oracle $\mathsf{O}' \diamond Q$. We define a set of "easy" queries:

$$Q^{\mathsf{Easy}} := \{(q, \mathsf{O}(q)) \,\big|\, V \text{ queries } q \text{ with "high" probability during } \mathsf{Exec}^{\mathsf{O}'\diamond Q}_{X,Y}\}.$$

Let Q^{Hard} be the set difference $Q \setminus Q^{\mathsf{Easy}}$. It is not hard to see that $Y = G^{\mathsf{O}'\diamond(Q^{\mathsf{Easy}}\cup Q^{\mathsf{Hard}})}(X)$. By completeness, V will accepts with probability $1 - \delta_c$ in the execution $\mathsf{Exec}^{\mathsf{O}'\diamond(Q^{\mathsf{Easy}}\cup Q^{\mathsf{Hard}})}_{X,Y}$.

Now, consider the execution $\langle P^{\mathsf{O}'\diamond(Q^{\mathsf{Easy}}\cup Q^{\mathsf{Hard}})}(X,Y), V^{\mathsf{O}'\diamond Q^{\mathsf{Easy}}}(Y)\rangle$, which is identical to $\mathsf{Exec}^{\mathsf{O}'\diamond(Q^{\mathsf{Easy}}\cup Q^{\mathsf{Hard}})}_{X,Y}$, except that we remove the Q^{Hard} from the verifier's oracle. In this execution, the probability that V accepts will not differ too much from that in $\mathsf{Exec}^{\mathsf{O}'\diamond(Q^{\mathsf{Easy}}\cup Q^{\mathsf{Hard}})}_{X,Y}$, because the queries in Q^{Hard} are asked by V with only "low" probability.

We then prove that Y is in the range of $G^{\mathsf{O}'\diamond Q^{\mathsf{Easy}}}(\cdot)$ with probability at most 0.5 (up to negligible error). But the previous argument says that $V^{\mathsf{O}'\diamond Q^{\mathsf{Easy}}}(Y)$ accepts with probability close to 1. It then follows from an averaging argument that there exists "bad" $\ddot{\mathsf{O}}$, $\ddot{\mathsf{O}}'$ and $\ddot{X}$[3] such that $\ddot{Y} = G^{\ddot{\mathsf{O}}}(\ddot{X})$ is *not* in the range of $G^{\ddot{\mathsf{O}}' \diamond \ddot{Q}^{\mathsf{Easy}}}(\cdot)$, but $V^{\ddot{\mathsf{O}}' \diamond \ddot{Q}^{\mathsf{Easy}}}(\ddot{Y})$ can be convinced with probability close to 1, by the malicious prover $P^{\ddot{\mathsf{O}}' \diamond (\ddot{Q}^{\mathsf{Easy}} \cup \ddot{Q}^{\mathsf{Hard}})}(\ddot{X}, \ddot{Y})$ (which can be viewed as an oracle machine $\mathbb{P}^{\ddot{\mathsf{O}}' \diamond \ddot{Q}^{\mathsf{Easy}}}$ with non-uniform advice $\ddot{X}$, $\ddot{Y}$, and $\ddot{Q}^{\mathsf{Hard}}$). This breaks the soundness of $\Pi^{\ddot{\mathsf{O}}' \diamond \ddot{Q}^{\mathsf{Easy}}}$, thus completing the proof.

We remark that proving Y is in the range of $G^{\mathsf{O}' \diamond Q^{\mathsf{Easy}}}(\cdot)$ with probability ≤ 0.5 (up to negligible error) is the most involved part. And this is where the HVZK property of $\Pi^{(\cdot)}$ plays an essential role. Roughly, we will show that if this claim does not hold, then there exists an adversary $\mathcal{A}^{\mathsf{O}}_{\text{PRG}}$ that can break the pseudo-randomness of $G^{\mathsf{O}}(\cdot)$ by making *polynomially* many oracle queries. As we will explain later, this reduction requires $\mathcal{A}^{\mathsf{O}}_{\text{PRG}}$ to know the set Q^{Easy} w.r.t. the challenge string Y in the security game of PRG. But note that $\mathcal{A}^{\mathsf{O}}_{\text{PRG}}$ does not know the preimage X (if Y is indeed in the range), which is necessary to figure out Q^{Easy}. This is where the HVZK simulator comes to our rescue. We will run the simulator $\mathsf{Sim}^{\mathsf{O}}_{V}(Y)$ repeatedly for (polynomially) many times to get an estimate $\widetilde{Q}^{\mathsf{Easy}}$ for the set Q^{Easy}. This $\widetilde{Q}^{\mathsf{Easy}}$ will be good enough to finish our proof. A more detailed overview of this strategy is provided at the beginning of Section 5.3.4.

5.3.3 Proof of Theorem 5.3.1

Assume for contradiction that there exists an oracle machine $G^{(\cdot)}$ and a protocol $\langle P^{(\cdot)}, V^{(\cdot)} \rangle$ such that given the access to any one-way function $\{f_n\}_{n\in\mathbb{N}}$:

1. $G^{f_n} : \{0,1\}^{\ell} \to \{0,1\}^{\ell+1}$ is a PRG (ℓ and n are polynomially related); **and**
2. $\Pi = \langle P^{f_n}, V^{f_n} \rangle$ is a semi-honest zero-knowledge argument system for the Meta-FBB functionality $\mathcal{F}^{f_n}_{\text{ZK}}[G]$.

We first recall the following lemma, which says that the measure-one of randomly-sampled oracles is one-way.

Lemma 5.3.1 (One-Wayness of Random Oracles [IR89, Yer11])**.** *Let $\mathcal{O} = \{\mathsf{O}_n\}_{n\in\mathbb{N}}$ be a collection of oracles where each O_n is chosen uniformly from the space of functions from $\{0,1\}^n$ to $\{0,1\}^n$. With probability 1 over the choice of $\mathcal{O}$, $\mathcal{O}$ is one-way against unbounded adversaries that make only polynomially many oracle queries to $\mathcal{O}$.*

Let both $\mathcal{O} = \{\mathsf{O}_n\}_{n\in\mathbb{N}}$ and $\mathcal{O}' = \{\mathsf{O}'_n\}_{n\in\mathbb{N}}$ be defined (independently) as in Lemma 5.3.1. It follows from Lemma 5.3.1 that, with probability 1, both $\mathcal{O}$ and $\mathcal{O}'$ are one-way.

In the following, we show two hybrids. From the second hybrid, we will construct a malicious prover breaking the soundness of $\Pi^{(\cdot)}$ (with the oracle being instantiated by a special one-way oracle defined later). This will give us the desired contradiction, and thus will finish the proof of Theorem 5.3.1.

Notation. We first define some notation. For an oracle H and a set of tuples $\mathsf{S} = \{(q_1, a_1), \ldots, (q_t, a_t)\}$, we define a new oracle $\mathsf{H} \diamond \mathsf{S}$ as follows: if q equals some q_i for which there exists a pair (q_i, a_i) in the set S, the oracle $\mathsf{H} \diamond \mathsf{S}$ returns a_i; otherwise, it returns $\mathsf{H}(q)$.

[3]Note that these values already determine the sets $\ddot{Q}$, $\ddot{Q}^{\mathsf{Easy}}$, and $\ddot{Q}^{\mathsf{Hard}}$ as defined above.

Formally,

$$\mathsf{H} \diamond \mathsf{S}(q) = \begin{cases} \mathsf{H}(q) & \text{if } q \notin \{q_1, \dots, q_t\} \\ a_i & \text{if } q = q_i \in \{q_1, \dots, q_t\} \end{cases}.$$

Hybrid H_0. This hybrid samples $X_n \xleftarrow{\$} \{0,1\}^{\ell(n)}$, and computes $Y_n = G^{\mathsf{O}_n}(X_n)$. W.l.o.g., we assume that G on input X_n makes $t(n)$ *distinct* queries to its oracle O_n, where $t(n)$ is a polynomial of n. Let $Q_n = \big\{\big(q_1, \mathsf{O}_n(q_1)\big), \dots, \big(q_t, \mathsf{O}_n(q_t)\big)\big\}$ be the query-answer pairs during the computation $Y_n = G^{\mathsf{O}_n}(X_n)$.

Let $\mathsf{Exec}_{X_n,Y_n}^{\mathsf{O}'_n \diamond Q_n} = \langle P^{\mathsf{O}'_n \diamond Q_n}(X_n, Y_n), V^{\mathsf{O}'_n \diamond Q_n}(Y_n)\rangle$ denote the execution where P proves to V that there exists an X_n such that $Y_n = G^{\mathsf{O}'_n \diamond Q_n}(X_n)$. Note that during this execution, the verifier may query its oracle $\mathsf{O}'_n \diamond Q_n$. For each $q_i \in \{0,1\}^n$, let p_i denote the probability that q_i is queried by V during $\mathsf{Exec}_{X_n,Y_n}^{\mathsf{O}'_n \diamond Q_n}$. Let Q_n^{Easy} defines the set of "easy" queries and their corresponding answers:

$$Q_n^{\mathsf{Easy}} := \Big\{\big(q_i, \mathsf{O}'_n \diamond Q_n(q_i)\big) \;\big|\; p_i \geq \frac{1}{t(n)\cdot n} \text{ during } \mathsf{Exec}_{X_n,Y_n}^{\mathsf{O}'_n \diamond Q_n}\Big\}. \tag{5.1}$$

Let Q_n^{Hard} be the set difference $Q_n \setminus Q_n^{\mathsf{Easy}}$. We remark that Q_n and $Q_n^{\mathsf{Easy}} \cup Q_n^{\mathsf{Hard}}$ may not be the same, but it must hold that $Q_n \subseteq Q_n^{\mathsf{Easy}} \cup Q_n^{\mathsf{Hard}}$.

Looking ahead, we will instantiate $G^{(\cdot)}$ and $\Pi^{(\cdot)}$ with the oracle $\mathsf{O}'_n \diamond (Q_n^{\mathsf{Easy}} \cup Q_n^{\mathsf{Hard}})$. Note that $G^{(\cdot)}$ and $\Pi^{(\cdot)}$ will have the desired property only if they are instantiated with *one-way* functions. Therefore, we show in Claim 5.3.1 that the composed oracle $\mathsf{O}'_n \diamond (Q_n^{\mathsf{Easy}} \cup Q_n^{\mathsf{Hard}})$ is one-way. It is worth noting that the one-wayness of this composed oracle is independent of the choice of $\{X_n\}$, though the definition of Q_n, Q_n^{Easy} and Q_n^{Hard} depends on X_n.

Claim 5.3.1. *The collection of oracles $\big\{\mathsf{O}'_n \diamond (Q_n^{\mathsf{Easy}} \cup Q_n^{\mathsf{Hard}})\big\}_{n\in\mathbb{N}}$ defined above is one-way with probability 1, where the probability is taken over the sampling of $\mathcal{O} = \{\mathsf{O}_n\}_n$ and $\mathcal{O}' = \{\mathsf{O}'_n\}_n$, and is* independent *of the distribution of $\{X_n\}_{n\in\mathbb{N}}$.*

Proof. The query-answer pairs in Q_n^{Easy} and Q_n^{Hard} are of the form $\big(q, \mathsf{O}_n(q)\big)$ or $\big(q, \mathsf{O}'_n(q)\big)$. Although X_n decides which $(q,*)$[4] will be in Q_n^{Easy} and Q_n^{Hard}, the answer part $\mathsf{O}_n(q)$'s and $\mathsf{O}'_n(q)$'s are uniformly distributed, *independent* of X_n. That is, if O_n and O'_n are sampled randomly, then for any $X_n \in \{0,1\}^{\ell(n)}$, $\mathsf{O}'_n \diamond (Q_n^{\mathsf{Easy}} \cup Q_n^{\mathsf{Hard}})$ will also be a random oracle. Therefore, for *any* $\{X_n\}_{n\in\mathbb{N}}$ where $X_n \in \{0,1\}^{\ell(n)}$, the following holds

$$\big\{\mathsf{O}'_n \diamond (Q_n^{\mathsf{Easy}} \cup Q_n^{\mathsf{Hard}})\big\}_{n\in\mathbb{N}} \overset{\text{i.d.}}{=\!=} \{\mathsf{O}''_n\}_{n\in\mathbb{N}},$$

where each O''_n is sampled uniformly from the space of functions from $\{0,1\}^n$ to $\{0,1\}^n$. Since it follows from Lemma 5.3.1 that $\{\mathsf{O}''_n\}_{n\in\mathbb{N}}$ is one-way with probability 1, so is $\big\{\mathsf{O}'_n \diamond (Q_n^{\mathsf{Easy}} \cup Q_n^{\mathsf{Hard}})\big\}_{n\in\mathbb{N}}$. ☐

Claim 5.3.1 (together with our assumption) implies that, with probability 1 taken over the sampling of $\mathcal{O}$ and $\mathcal{O}'$:

[4]The symbol "$*$" denotes the wildcard that matches any answer to q.

– $G^{\mathsf{O}'_n \diamond (Q_n^{\mathsf{Easy}} \cup Q_n^{\mathsf{Hard}})} : \{0,1\}^{\ell(n)} \to \{0,1\}^{\ell(n)+1}$ is pseudo-random against all (unbounded) adversaries that make polynomially many queries to the oracle $\mathsf{O}'_n \diamond (Q_n^{\mathsf{Easy}} \cup Q_n^{\mathsf{Hard}})$; **and**

– $\Pi^{\mathsf{O}'_n \diamond (Q_n^{\mathsf{Easy}} \cup Q_n^{\mathsf{Hard}})}$ is a semi-honest zero-knowledge argument for the Meta-FBB functionality $\mathcal{F}_{\mathrm{ZK}}^{\mathsf{O}'_n \diamond (Q_n^{\mathsf{Easy}} \cup Q_n^{\mathsf{Hard}})}[G]$.

Let $\mathsf{Exec}_{X_n,Y_n}^{\mathsf{O}'_n \diamond (Q_n^{\mathsf{Easy}} \cup Q_n^{\mathsf{Hard}})}$ denote the following execution:

$$\langle P^{\mathsf{O}'_n \diamond (Q_n^{\mathsf{Easy}} \cup Q_n^{\mathsf{Hard}})}(X_n, Y_n), V^{\mathsf{O}'_n \diamond (Q_n^{\mathsf{Easy}} \cup Q_n^{\mathsf{Hard}})}(Y_n)\rangle.$$

Let $\mathsf{Exec}_{X_n,Y_n}^{\mathsf{O}'_n \diamond (Q_n^{\mathsf{Easy}} \cup Q_n^{\mathsf{Hard}})} = 1$ denote the event that the verifier accepts at the end of this execution. Then, it follows from Claim 5.3.1 and the completeness of $\Pi^{\mathsf{O}'_n \diamond (Q_n^{\mathsf{Easy}} \cup Q_n^{\mathsf{Hard}})}$ that:

$$\Pr_{\mathcal{O},\mathcal{O}'}\left[\begin{array}{l}\text{For sufficient large } n \in \mathbb{N}, \forall X_n \in \{0,1\}^{\ell(n)}, Y_n = G^{\mathsf{O}_n}(X_n),\\ \Pr\left[\mathsf{Exec}_{X_n,Y_n}^{\mathsf{O}'_n \diamond (Q_n^{\mathsf{Easy}} \cup Q_n^{\mathsf{Hard}})} = 1\right] \geq 1 - \delta_c(n)\end{array}\right] = 1, \tag{5.2}$$

where the inner probability is taken over the random coins of the prover and the verifier in the execution $\mathsf{Exec}_{X_n,Y_n}^{\mathsf{O}'_n \diamond (Q_n^{\mathsf{Easy}} \cup Q_n^{\mathsf{Hard}})}$, and $\delta_c(n)$ is the completeness error.

Hybrid H_1. This hybrid is identical to the previous one, except that H_1 executes the protocol

$$\langle P^{\mathsf{O}'_n \diamond (Q_n^{\mathsf{Easy}} \cup Q_n^{\mathsf{Hard}})}(X_n, Y_n), V^{\mathsf{O}'_n \diamond Q_n^{\mathsf{Easy}}}(Y_n)\rangle. \tag{5.3}$$

(Compared with the execution $\mathsf{Exec}_{X_n,Y_n}^{\mathsf{O}'_n \diamond (Q_n^{\mathsf{Easy}} \cup Q_n^{\mathsf{Hard}})}$ in H_0, the only difference is that H_1 remove Q_n^{Hard} from the *verifier's* oracle.)

As mentioned in the **Proof Sketch** of Theorem 5.3.1, we want to show that the verifier accepts in Execution (5.3) with probability close to that in the execution $\mathsf{Exec}_{X_n,Y_n}^{\mathsf{O}'_n \diamond (Q_n^{\mathsf{Easy}} \cup Q_n^{\mathsf{Hard}})}$. This is formalized as Claim 5.3.2.

Claim 5.3.2. *With probability 1 taken over the sampling of $\mathcal{O}$ and $\mathcal{O}'$, for sufficiently large $n \in \mathbb{N}$, it holds that $\forall X_n \in \{0,1\}^{\ell(n)}$ and $Y_n = G^{\mathsf{O}_n}(X_n)$,*

$$\Pr[\langle P^{\mathsf{O}'_n \diamond (Q_n^{\mathsf{Easy}} \cup Q_n^{\mathsf{Hard}})}(X_n, Y_n), V^{\mathsf{O}'_n \diamond Q_n^{\mathsf{Easy}}}(Y_n)\rangle = 1] \geq \Pr\left[\mathsf{Exec}_{X_n,Y_n}^{\mathsf{O}'_n \diamond (Q_n^{\mathsf{Easy}} \cup Q_n^{\mathsf{Hard}})} = 1\right] - \frac{1}{n}, \tag{5.4}$$

where the probabilities in the above inequality are taken over the random coins of the prover and the verifier during the corresponding executions.

Proof. First, we remark that the "with probability 1" part in this claim is to ensure that $\{\mathsf{O}'_n \diamond (Q_n^{\mathsf{Easy}} \cup Q_n^{\mathsf{Hard}})\}_n$ is one-way (see Claim 5.3.1). In the following, we proceed with $\{\mathsf{O}'_n \diamond (Q_n^{\mathsf{Easy}} \cup Q_n^{\mathsf{Hard}})\}_n$ being one-way (so the probabilities below are not taken over $\mathcal{O}$ and $\mathcal{O}'$).

By definition, any query[5] $q \in Q_n^{\mathsf{Hard}}$ is asked by V during $\mathsf{Exec}_{X_n,Y_n}^{\mathsf{O}'_n \diamond (Q_n^{\mathsf{Easy}} \cup Q_n^{\mathsf{Hard}})}$ with probability $< \frac{1}{t(n)\cdot n}$. Let us define the following event:

[5]Technically, elements in Q_n^{Hard} are query-answer pairs. From here on, we override the notation "$\in$" such that $q \in Q_n^{\mathsf{Hard}}$ also means that there exists a pair $(q, *)$ in Q_n^{Hard}.

– $\mathsf{Event}_{\mathsf{NoHard}}$: no $q \in Q_n^{\mathsf{Hard}}$ is asked by V in $\mathsf{Exec}_{X_n,Y_n}^{\mathsf{O}'_n \diamond (Q_n^{\mathsf{Easy}} \cup Q_n^{\mathsf{Hard}})}$.

It follows from the union bound that

$$\Pr\left[\mathsf{Event}_{\mathsf{NoHard}}\right] \geq 1 - \frac{1}{n}, \tag{5.5}$$

where the probability is taken over the random coins of P and V in the execution $\mathsf{Exec}_{X_n,Y_n}^{\mathsf{O}'_n \diamond (Q_n^{\mathsf{Easy}} \cup Q_n^{\mathsf{Hard}})}$.

Now, we prove Inequality (5.4). In the following, for succinctness, let

– Exec_0 denote the execution $\langle P^{\mathsf{O}'_n \diamond (Q_n^{\mathsf{Easy}} \cup Q_n^{\mathsf{Hard}})}(X_n, Y_n), V^{\mathsf{O}'_n \diamond Q_n^{\mathsf{Easy}}}(Y_n)\rangle$;

– Exec_1 denote the execution $\mathsf{Exec}_{X_n,Y_n}^{\mathsf{O}'_n \diamond (Q_n^{\mathsf{Easy}} \cup Q_n^{\mathsf{Hard}})}$.

Then, we have (probabilities below are taken over the random coins over P and V in the corresponding executions):

$$\begin{aligned}
\Pr[\mathsf{Exec}_1 = 1] &\geq \Pr\left[\mathsf{Exec}_1 = 1 \;\middle|\; \mathsf{Event}_{\mathsf{NoHard}}\right] \cdot \Pr\left[\mathsf{Event}_{\mathsf{NoHard}}\right] \\
&= \Pr[\mathsf{Exec}_0 = 1 \mid \mathsf{Event}_{\mathsf{NoHard}}] \cdot \Pr\left[\mathsf{Event}_{\mathsf{NoHard}}\right] && (5.6)\\
&\geq \Pr[\mathsf{Exec}_0 = 1] - \Pr[\neg\mathsf{Event}_{\mathsf{NoHard}}] && (5.7)\\
&\geq \Pr[\mathsf{Exec}_0 = 1] - \frac{1}{n} && (5.8)
\end{aligned}$$

where Step 5.6 is due to the fact that Exec_1 and Exec_0 are identical assuming V does not make any query $q \in Q_n^{\mathsf{Hard}}$, Step 5.7 follows from the basic probability inequality that $\Pr[A \mid B] \cdot \Pr[B] \geq \Pr[A] - \Pr[\neg B]$, and Step 5.8 follows from Inequality (5.5).

This finishes the proof of Claim 5.3.2. □

Claim 5.3.2 indicates that the verifier in Execution (5.3) accepts with "good" probability: at least as large as the accepting probability of $\mathsf{Exec}_{X_n,Y_n}^{\mathsf{O}'_n \diamond (Q_n^{\mathsf{Easy}} \cup Q_n^{\mathsf{Hard}})}$ minus $1/n$. Thus, we will have the desired contradiction if the Y_n in Execution (5.3) is a false statement, namely that Y_n is not in the range of $G^{\mathsf{O}'_n \diamond Q_n^{\mathsf{Easy}}}$ (i.e. $G^{(\cdot)}$ instantiated by the verifier's oracle in Execution (5.3)). This argument is formalized and proved in Claims 5.3.3 and 5.3.4, which will eventually finish the proof of Theorem 5.3.1.

Claim 5.3.3. *Let Q_n^{Easy} be defined as in Expression (5.1). For sufficiently large $n \in \mathbb{N}$, the following holds:*

$$\Pr_{\mathcal{O},\mathcal{O}',X_n}\left[G^{\mathsf{O}_n}(X_n) \in G^{\mathsf{O}'_n \diamond Q_n^{\mathsf{Easy}}}\left(\{0,1\}^{\ell(n)}\right)\right] \leq \frac{1}{2} + \mathsf{negl}(n). \tag{5.9}$$

Note that the above probability is taken (additionally) over $X_n \xleftarrow{\$} \{0,1\}^{\ell(n)}$.

Claim 5.3.4. *If Claim 5.3.3 holds, then Theorem 5.3.1 holds.*

The proof of Claim 5.3.3 is quite involved. It constitutes the main technical challenge of the current proof (of Theorem 5.3.1). Thus, we will deal with it in Section 5.3.4. In the following, we show the proof of Claim 5.3.4.

Proof of Claim 5.3.4. It follows from Expression (5.2) and Claim 5.3.2 that

$$\Pr_{\mathcal{O},\mathcal{O}'}\left[\begin{array}{l}\text{For sufficient large } n \in \mathbb{N}, \forall X_n \in \{0,1\}^{\ell(n)}, Y_n = G^{\mathsf{O}_n}(X_n),\\ \Pr\left[\langle P^{\mathsf{O}'_n \diamond (Q_n^{\mathsf{Easy}} \cup Q_n^{\mathsf{Hard}})}(X_n, Y_n), V^{\mathsf{O}'_n \diamond Q_n^{\mathsf{Easy}}}(Y_n)\rangle = 1\right] \geq 1 - \frac{1}{n} - \delta_c(n)\end{array}\right] = 1. \quad (5.10)$$

Following the same argument as for Claim 5.3.1, we can prove the one-wayness of the oracle $\{\mathsf{O}'_n \diamond Q_n^{\mathsf{Easy}}\}_n$ as follows. For each $(q, \mathsf{O}_n(q)) \in Q_n^{\mathsf{Easy}}$, the $\mathsf{O}_n(q)$ is a randomly sampled string from $\{0,1\}^n$. Therefore, no matter what X_n is, $\mathsf{O}'_n \diamond Q_n^{\mathsf{Easy}}$ is always a randomly sampled oracle (though Q_n^{Easy} is determined by X_n). It then follows from Lemma 5.3.1 that:

$$\Pr_{\mathcal{O},\mathcal{O}'}\left[\forall X_n \in \{0,1\}^{\ell(n)},\ \{\mathsf{O}'_n \diamond Q_n^{\mathsf{Easy}}\}_{n\in\mathbb{N}} \text{ is one-way}\right] = 1. \quad (5.11)$$

By an averaging argument over Expressions (5.9) to (5.11), it follows that there exists fixed sequences $\{\ddot{\mathsf{O}}_n\}_{n\in\mathbb{N}}$, $\{\ddot{\mathsf{O}}'_n\}_{n\in\mathbb{N}}$ and $\{\ddot{X}_n\}_{n\in\mathbb{N}}$[6] such that for sufficiently large $n \in \mathbb{N}$,

- $\{\ddot{\mathsf{O}}_n \diamond \ddot{Q}_n^{\mathsf{Easy}}\}_n$ is one-way; thus, $G^{\ddot{\mathsf{O}}_n \diamond \ddot{Q}_n^{\mathsf{Easy}}}$ is a PRG and $\Pi^{\ddot{\mathsf{O}}_n \diamond \ddot{Q}_n^{\mathsf{Easy}}}$ is an HVZK protocol for the membership of $G^{\ddot{\mathsf{O}}_n \diamond \ddot{Q}_n^{\mathsf{Easy}}}$; **and**
- $\ddot{Y}_n$ is not in the range of $G^{\ddot{\mathsf{O}}_n \diamond \ddot{Q}_n^{\mathsf{Easy}}}$; **and**
- $\Pr\left[\langle P^{\ddot{\mathsf{O}}'_n \diamond (\ddot{Q}_n^{\mathsf{Easy}} \cup \ddot{Q}_n^{\mathsf{Hard}})}(\ddot{X}_n, \ddot{Y}_n), V^{\ddot{\mathsf{O}}'_n \diamond \ddot{Q}_n^{\mathsf{Easy}}}(\ddot{Y}_n)\rangle = 1\right] \geq 1 - \frac{1}{n} - \delta_c(n)$, where the probability is taken over the random coins of P and V.

  Note that we can treat $P^{\ddot{\mathsf{O}}'_n \diamond (\ddot{Q}_n^{\mathsf{Easy}} \cup \ddot{Q}_n^{\mathsf{Hard}})}(\ddot{X}_n, \ddot{Y}_n)$ as an oracle machine $\mathbb{P}^{\ddot{\mathsf{O}}'_n \diamond \ddot{Q}_n^{\mathsf{Easy}}}$, which has $(\ddot{Q}_n^{\mathsf{Easy}}, \ddot{X}_n, \ddot{Y}_n)$ as non-uniform advice and makes only polynomially many queries to its oracle $\ddot{\mathsf{O}}'_n \diamond \ddot{Q}_n^{\mathsf{Easy}}$.

Since the completeness error $\delta_c(\cdot)$ is negligible, the above means that $\mathbb{P}^{\ddot{\mathsf{O}}'_n \diamond \ddot{Q}_n^{\mathsf{Easy}}}$ (with its non-uniform advice) convinces the verifier with non-negligible probability on the following *false* statement:

$$\ddot{Y}_n \in G^{\ddot{\mathsf{O}}'_n \diamond \ddot{Q}_n^{\mathsf{Easy}}}\left(\{0,1\}^{\ell(n)}\right).$$

This contradicts the soundness of $\Pi^{\ddot{\mathsf{O}}'_n \diamond \ddot{Q}_n^{\mathsf{Easy}}}$, thus finishing the proof of Claim 5.3.4. □

5.3.4 Proof of Claim 5.3.3

5.3.4.1 Proof Overview

Before diving into the proof, we first provide a high-level overview.

We assume for contradiction that Claim 5.3.3 is false and try to break the pseudorandomness of G^{O_n}. First, observe that if $Y_n = G^{\mathsf{O}_n}(X_n)$ where $X_n \xleftarrow{\$} \{0,1\}^{\ell(n)}$, then our assumption implies that Y_n is in the range of $G^{\mathsf{O}'_n \diamond Q_n^{\mathsf{Easy}}}(\cdot)$ with probability noticeably larger than $1/2$. Therefore, on an input Y_n, if we can efficiently test if $Y_n \in G^{\mathsf{O}'_n \diamond Q_n^{\mathsf{Easy}}}\left(\{0,1\}^{\ell(n)}\right)$, we should have some advantage in the PRG game for $G^{\mathsf{O}_n}(\cdot)$. This strategy has the following potential problems:

[6]Note that these values also fix the corresponding $\{\ddot{Y}_n\}_{n\in\mathbb{N}}$, $\{\ddot{Q}_n^{\mathsf{Easy}}\}_{n\in\mathbb{N}}$ and $\{\ddot{Q}_n^{\mathsf{Hard}}\}_{n\in\mathbb{N}}$ as in the above.

1. Without the preimage X_n, we *cannot* compute the set Q_n^{Easy} (see Expression (5.1)) using only polynomially many queries to O_n;
2. If the input $Y_n \notin G^{\mathsf{O}_n}\left(\{0,1\}^{\ell(n)}\right)$, the set Q_n (thus Q_n^{Easy}) is not even well-defined, as there is no preimage X_n.

To avoid using X_n, we will run the HVZK simulator to obtain an estimate of the set Q_n^{Easy} in the following way. Recall that Q_n^{Easy} contains the "easy" queries made by the verifier during $\mathsf{Exec}_{X_n,Y_n}^{\mathsf{O}'_n \diamond Q_n}$. By the HVZK property of the protocol $\Pi^{\mathsf{O}'_n \diamond Q_n}$, each query in Q_n^{Easy} should be made with similar probability in the simulated execution $\mathsf{Sim}_V^{\mathsf{O}'_n \diamond Q_n}(Y_n)$. Therefore, repeating $\mathsf{Sim}_V^{\mathsf{O}'_n \diamond Q_n}(Y_n)$ (polynomially) many times will give us a good estimate to Q_n^{Easy}.

However, without X_n, we cannot figure out the set Q_n, which is necessary if we want to run $\mathsf{Sim}_V^{\mathsf{O}'_n \diamond Q_n}(Y_n)$. Fortunately, by a similar argument as that for Claim 5.3.1, we can prove that the oracle $\{\mathsf{O}_n\}_n$ and $\{\mathsf{O}'_n \diamond Q_n\}_n$ are identically distributed, even given X_n and $Y_n = G^{\mathsf{O}_n}(X_n)$. Therefore, running $\mathsf{Sim}_V^{\mathsf{O}_n}(Y_n)$ will be just as good as running $\mathsf{Sim}_V^{\mathsf{O}'_n \diamond Q_n}(Y_n)$. Note that this also solves Problem 2, because the simulator still works when invoked on false statements.

Now, we can construct the PRG distinguisher $\mathcal{A}_{\mathrm{PRG}}^{\mathsf{O}_n}(Y_n)$ as follows: on input Y_n, $\mathcal{A}_{\mathrm{PRG}}^{\mathsf{O}_n}(Y_n)$ obtains an estimate $\widetilde{Q}_n^{\mathsf{Easy}}$ to Q_n^{Easy} by running $\mathsf{Sim}_V^{\mathsf{O}_n}(Y_n)$ polynomially many times. It then samples a random function $\mathsf{O}'_n : \{0,1\}^n \to \{0,1\}^n$, and outputs 1 if $Y_n \in G^{\mathsf{O}'_n \diamond \widetilde{Q}_n^{\mathsf{Easy}}}\left(\{0,1\}^{\ell(n)}\right)$; otherwise, it outputs 0. Note that although sampling O' requires exponential time, $\mathcal{A}_{\mathrm{PRG}}^{\mathsf{O}_n}(Y_n)$ only makes *polynomially many* queries to the oracle O_n.

If $Y_n = G^{\mathsf{O}_n}(X_n)$ where $X_n \overset{\$}{\leftarrow} \{0,1\}^{\ell(n)}$, then by our assumption $\mathcal{A}^{\mathsf{O}_n}(Y_n)$ outputs 1 with probability noticeably larger than 1/2; if $Y_n \overset{\$}{\leftarrow} \{0,1\}^{\ell(n)+1}$, then Y_n is independent of O_n. Moreover, using a similar argument as for Claim 5.3.1, we can prove that Y_n is independent of the oracle $\widetilde{Q}_n^{\mathsf{Easy}}$ (thus $\mathsf{O}'_n \diamond \widetilde{Q}_n^{\mathsf{Easy}}$). Since the function $G^{\mathsf{O}'_n \diamond \widetilde{Q}_n^{\mathsf{Easy}}}(\cdot)$ stretch by 1 bit, the random Y_n will be in its range with probability 1/2. This means $\mathcal{A}^{\mathsf{O}_n}(Y_n)$ outputs 1 with probability exactly 1/2.

This gives us the desired contradiction.

5.3.4.2 The Formal Proof

We now present the formal proof for Claim 5.3.3. First, we describe in Algorithm 1 how to compute the set $\widetilde{Q}_n^{\mathsf{Easy}}$, which is the estimate to Q_n^{Easy} by running $\mathsf{Sim}_V^{\mathsf{O}_n}(Y_n)$ (see the above proof overview). In the following, we break Claim 5.3.3 into Claims 5.3.5 and 5.3.7, and prove them one-by-one.

Claim 5.3.5. *For sufficiently large $n \in \mathbb{N}$, it holds that*

$$\Pr_{\mathcal{O},\mathcal{O}',X_n}\left[G^{\mathsf{O}_n}(X_n) \in G^{\mathsf{O}'_n \diamond Q_n^{\mathsf{Easy}}}(\{0,1\}^{\ell(n)})\right] \leq \Pr_{\mathcal{O},\mathcal{O}',X_n,\widetilde{Q}_n^{\mathsf{Easy}}}\left[G^{\mathsf{O}_n}(X_n) \in G^{\mathsf{O}'_n \diamond \widetilde{Q}_n^{\mathsf{Easy}}}(\{0,1\}^{\ell(n)})\right] + \mathsf{negl}(n). \quad (5.12)$$

Proof. Let $Y_n = G^{\mathsf{O}_n}(X_n)$. Let $\mathsf{Exec}_{X_n,Y_n}^{\mathsf{O}_n}$ denote the execution $\langle P^{\mathsf{O}_n}(X_n, Y_n), V^{\mathsf{O}_n}(Y_n)\rangle$. For each $q_i \in \{0,1\}^n$, let p_i denote the probability that q_i is asked *by the verifier* during $\mathsf{Exec}_{X_n,Y_n}^{\mathsf{O}_n}$.

Algorithm 1: Sampling the Set $\widetilde{Q}_n^{\mathsf{Easy}}$

Input: a string $Y_n \in \{0,1\}^{\ell(n)+1}$.

Oracle: an oracle O_n mapping $\{0,1\}^n$ to $\{0,1\}^n$.

Let $0 < c < 1$ be a constant. This algorithm initializes a table $\mathcal{T}$ to the records of the form $\big(q_i, \mathsf{Count}[q_i] = 0\big)$ for all $q_i \in \{0,1\}^n$. $\mathcal{T}$ will be used to store the number that each query being asked.

This algorithm repeats the following procedure for $N = 3n/c^2$ times, using fresh randomness for each repetition:

– It invokes the HVZK simulator $\mathsf{Sim}_V^{\mathsf{O}_n}(Y_n)$. Note that the simulated view $\mathsf{View} \leftarrow \mathsf{Sim}_V^{\mathsf{O}_n}(Y_n)$ contains the query-answer pairs exchanged between V and O_n. For every query-answer pair $(q_i, \mathsf{O}_n(q_i))$ appeared in View, increase $\mathsf{Count}[q_i]$ by 1.

For any query $q_i \in \{0,1\}^n$, let $\widehat{p}_i$ denotes the frequency (i.e. the empirical mean) that q_i is asked by the verifier during the above N repetitions. The hybrid then computes the set $\widetilde{Q}_n^{\mathsf{Easy}}$ as follows:

$$\widetilde{Q}_n^{\mathsf{Easy}} := \left\{\big(q_i, \mathsf{O}_n(q_i)\big) \;\middle|\; \widehat{p}_i \geq \frac{1}{n \cdot t(n)} - 2c\right\}, \quad \text{where } \widehat{p}_i := \frac{\mathsf{Count}[q_i]}{N}. \tag{5.13}$$

We now define the following set:

$$\bar{Q}_n^{\mathsf{Easy}} := \left\{\big(q_i, \mathsf{O}_n(q_i)\big) \mid p_i \geq \frac{1}{t(n) \cdot n} \text{ during } \mathsf{Exec}_{X_n,Y_n}^{\mathsf{O}_n}\right\}. \tag{5.14}$$

We remark that $\bar{Q}_n^{\mathsf{Easy}}$ is the same as Q_n^{Easy} (Expression (5.1)) but is w.r.t. $\mathsf{Exec}_{X_n,Y_n}^{\mathsf{O}_n}$. In Claim 5.3.6, we show that $\bar{Q}_n^{\mathsf{Easy}}$ and Q_n^{Easy} are identically distributed. Looking ahead, this claim will allow us to replace Q_n^{Easy} with $\bar{Q}_n^{\mathsf{Easy}}$, for which we can obtain a "good" estimate (via the HVZK simulator) without knowing the preimage X_n.

Claim 5.3.6. *Let Q^{Easy} be as in Expression (5.1). Let $\bar{Q}^{\mathsf{Easy}}$ be as in Expression (5.14). The following holds:*

$$\Pr_{\mathcal{O},\mathcal{O}',X_n}\left[G^{\mathsf{O}_n}(X_n) \in G^{\mathsf{O}'_n \diamond Q_n^{\mathsf{Easy}}}\big(\{0,1\}^{\ell(n)}\big)\right] = \Pr_{\mathcal{O},\mathcal{O}',X_n}\left[G^{\mathsf{O}_n}(X_n) \in G^{\mathsf{O}'_n \diamond \bar{Q}_n^{\mathsf{Easy}}}\big(\{0,1\}^{\ell(n)}\big)\right] \tag{5.15}$$

Proof. We first claim that the following two ensembles are identically distributed:

$$\{X_n, G^{\mathsf{O}_n}(X_n), \mathsf{O}_n\}_{n\in\mathbb{N}} \overset{\text{i.d.}}{=\!=} \{X_n, G^{\mathsf{O}_n}(X_n), \mathsf{O}'_n \diamond Q_n\}_{n\in\mathbb{N}}, \tag{5.16}$$

where $X_n \xleftarrow{\$} \{0,1\}^{\ell(n)}$, O_n and O'_n are random functions mapping $\{0,1\}^n$ to $\{0,1\}^n$, and Q_n is defined as in hybrid H_0 (the set of query-answer pairs during the evaluation of $G^{\mathsf{O}_n}(X_n)$). To show Expression (5.16), we only need to show that given $\{X_n\}_n$, the oracles $\{\mathsf{O}_n\}_n$ and $\{\mathsf{O}'_n \diamond Q_n\}_n$ are identically distributed. This is true as they agree on all the query-answer pairs contained in Q_n, and the queries (and their answers) not in Q_n are identically distributed (i.e., uniform).

Then, note that $\{\bar{Q}^{\mathsf{Easy}}\}_n$ is determined (deterministically) by the left-hand side of Expression (5.16) *in the same way* as $\{Q\}_n$ is determined by the right-hand side of Expression (5.16). So $\{Q^{\mathsf{Easy}}\}_n$ and $\{\bar{Q}^{\mathsf{Easy}}\}_n$ are also identically distributed. Thus, Equation (5.15) follows. $\square$

Next, we prove that the $\widetilde{Q}_n^{\mathsf{Easy}}$ defined in Algorithm 1 is a good estimate to the set $\bar{Q}_n^{\mathsf{Easy}}$ in the sense that $\bar{Q}_n^{\mathsf{Easy}}$ is a subset of $\widetilde{Q}_n^{\mathsf{Easy}}$ except with negligible probability taken over the sampling of $\widetilde{Q}_n^{\mathsf{Easy}}$. For any $q_i \in \{0,1\}^n$, let $\widetilde{p}_i$ denote the probability that q_i is asked by the verifier in the simulated transcript $\mathsf{View} \leftarrow \mathsf{Sim}_V^{\mathsf{O}_n}(Y_n)$. Due to the HVZK property of Π^{O_n}, it holds that $\widetilde{p}_i \geq p_i - \mathsf{negl}(n)$. Recall the quantity $\widehat{p}_i$ from Expression (5.13), which is the empirical mean of the frequency that p_i is asked by the verifier in $3n/c^2$ repetitions of $\mathsf{Sim}_V^{\mathsf{O}_n}(Y)$. It then follows from the Chernoff bound that

$$\Pr_{\widehat{p}_i}\left[|\widehat{p}_i - \widetilde{p}_i| \geq c\right] \leq \frac{1}{2^n} = \mathsf{negl}(n). \tag{5.17}$$

By definition, each $q_i \in \bar{Q}_n^{\mathsf{Easy}}$ will be asked with probability $p_i \geq \frac{1}{n \cdot t(n)}$ during the execution $\mathsf{Exec}^{\mathsf{O}_n}_{X_n,Y_n}$. It then follows from Inequality (5.17) that for any $q_i \in \bar{Q}_n^{\mathsf{Easy}}$, the following holds except with negligible probability taken over the sampling of $\widehat{p}_i$:

$$\widehat{p}_i \geq \widetilde{p}_i - c = p_i - c - \mathsf{negl}(n) \geq \frac{1}{n \cdot t(n)} - c - \mathsf{negl}(n) > \frac{1}{n \cdot t(n)} - 2c,$$

which means $q_i \in \widetilde{Q}_n^{\mathsf{Easy}}$ by the definition of $\widetilde{Q}_n^{\mathsf{Easy}}$. Since $|\bar{Q}_n^{\mathsf{Easy}}|$ is a polynomial of n, it follows from union bound that

$$\Pr_{\widetilde{Q}_n^{\mathsf{Easy}}}\left[\bar{Q}_n^{\mathsf{Easy}} \subseteq \widetilde{Q}_n^{\mathsf{Easy}}\right] \geq 1 - \mathsf{negl}(n) \quad \Big(\Leftrightarrow \Pr_{\widetilde{Q}_n^{\mathsf{Easy}}}\left[\bar{Q}_n^{\mathsf{Easy}} \not\subseteq \widetilde{Q}_n^{\mathsf{Easy}}\right] \leq \mathsf{negl}(n)\Big). \tag{5.18}$$

Then we have

$$\begin{aligned}
&\Pr_{\mathcal{O},\mathcal{O}',X_n}\left[G^{\mathsf{O}_n}(X_n) \in G^{\mathsf{O}'_n \diamond \bar{Q}_n^{\mathsf{Easy}}}\left(\{0,1\}^{\ell(n)}\right)\right] \\
&\leq \Pr\left[G^{\mathsf{O}_n}(X_n) \in G^{\mathsf{O}'_n \diamond \bar{Q}_n^{\mathsf{Easy}}}\left(\{0,1\}^{\ell(n)}\right) \mid \bar{Q}_n^{\mathsf{Easy}} \subseteq \widetilde{Q}_n^{\mathsf{Easy}}\right] \cdot \Pr\left[\bar{Q}_n^{\mathsf{Easy}} \subseteq \widetilde{Q}_n^{\mathsf{Easy}}\right] + \Pr\left[\bar{Q}_n^{\mathsf{Easy}} \not\subseteq \widetilde{Q}_n^{\mathsf{Easy}}\right] \\
&\leq \Pr\left[G^{\mathsf{O}_n}(X_n) \in G^{\mathsf{O}'_n \diamond \widetilde{Q}_n^{\mathsf{Easy}}}\left(\{0,1\}^{\ell(n)}\right) \mid \bar{Q}_n^{\mathsf{Easy}} \subseteq \widetilde{Q}_n^{\mathsf{Easy}}\right] \cdot \Pr\left[\bar{Q}_n^{\mathsf{Easy}} \subseteq \widetilde{Q}_n^{\mathsf{Easy}}\right] + \Pr\left[\bar{Q}_n^{\mathsf{Easy}} \not\subseteq \widetilde{Q}_n^{\mathsf{Easy}}\right] \\
&\leq \Pr\left[G^{\mathsf{O}_n}(X_n) \in G^{\mathsf{O}'_n \diamond \widetilde{Q}_n^{\mathsf{Easy}}}\left(\{0,1\}^{\ell(n)}\right)\right] + \Pr\left[\bar{Q}_n^{\mathsf{Easy}} \not\subseteq \widetilde{Q}_n^{\mathsf{Easy}}\right] \\
&\leq \Pr_{\mathcal{O},\mathcal{O}',X_n,\widetilde{Q}_n^{\mathsf{Easy}}}\left[G^{\mathsf{O}_n}(X_n) \in G^{\mathsf{O}'_n \diamond \widetilde{Q}_n^{\mathsf{Easy}}}\left(\{0,1\}^{\ell(n)}\right)\right] + \mathsf{negl}(n)
\end{aligned} \tag{5.19}$$

where Inequality (5.19) follows from Inequality (5.18).

Equation (5.15) and Inequality (5.19) finish the proof of Claim 5.3.5. $\square$

Remark 5.3.1 (On the Probability Space)**.** *Algorithm 1 run the HVZK simulator for the protocol* Π^{O_n}*. We want to point out that* Π^{O_n} *is an HVZK protocol (i.e. the simulator exists)* *only if $\mathcal{O} = \{\mathsf{O}_n\}_n$ is one-way. Therefore, in the above proof, whenever we try to argue about some probability of the form*

$$\Pr_{\mathcal{O},\widetilde{Q}_n^{\mathsf{Easy}}}\left[\mathsf{Event}(\mathsf{O}_n, \widetilde{Q}_n^{\mathsf{Easy}})\right] = \mathsf{value} \tag{5.20}$$

with the probability taken over both *$\mathcal{O}$ and $\widetilde{Q}_n^{\mathsf{Easy}}$ (e.g. Inequalities (5.17), (5.18)* and *(5.19), and some expressions in the next claim), the technically correct way is to say: with probability 1 taken over $\mathcal{O} = \{\mathsf{O}_n\}_n$ (thus being one-way), it holds that*

$$\Pr_{\widetilde{Q}_n^{\mathsf{Easy}}}\left[\mathsf{Event}(\mathsf{O}_n, \widetilde{Q}_n^{\mathsf{Easy}})\right] = \mathsf{value}. \tag{5.21}$$

Or, put in another way:

1. $\Pr_{\mathcal{O}}[\mathcal{O}$ *is one-way*$] = 1$; **and**
2. $\Pr_{\widetilde{Q}_n^{\mathsf{Easy}}}\left[\mathsf{Event}(\mathsf{O}_n, \widetilde{Q}_n^{\mathsf{Easy}}) \mid \mathcal{O} \text{ is one-way}\right] = \mathsf{value}$.

However, the form of Expression (5.20) is also correct because it is actually a consequence of Expression (5.21): let A denote the event $\mathsf{Event}(\mathsf{O}_n, \widetilde{Q}_n^{\mathsf{Easy}})$, and B the event that $\mathcal{O}$ is one-way, then

$$\begin{aligned}\Pr_{\mathcal{O},\widetilde{Q}_n^{\mathsf{Easy}}}[A] &= \Pr_{\widetilde{Q}_n^{\mathsf{Easy}}}[A \mid B] \cdot \Pr_{\mathcal{O}}[B] + \Pr_{\widetilde{Q}_n^{\mathsf{Easy}}}[A \mid \neg B] \cdot \Pr_{\mathcal{O}}[\neg B] \\ &= \Pr_{\widetilde{Q}_n^{\mathsf{Easy}}}[A \mid B] \cdot 1 + \Pr_{\widetilde{Q}_n^{\mathsf{Easy}}}[A \mid \neg B] \cdot 0 = \Pr_{\widetilde{Q}_n^{\mathsf{Easy}}}[A \mid B] = \mathsf{value}.\end{aligned}$$

So, it is fine to use Expression (5.20).

Claim 5.3.7. *For sufficiently large $n \in \mathbb{N}$, it holds that*

$$\Pr_{\mathcal{O},\mathcal{O}',X_n,\widetilde{Q}_n^{\mathsf{Easy}}}\left[G^{\mathsf{O}_n}(X_n) \in G^{\mathsf{O}'_n \diamond \widetilde{Q}_n^{\mathsf{Easy}}}\left(\{0,1\}^{\ell(n)}\right)\right] \le \frac{1}{2} + \mathsf{negl}(n) \tag{5.22}$$

Proof. At a high level, this proof goes as follows. Assuming for contradiction that the claim does not hold, we show an adversary $\mathcal{A}_{\mathrm{PRG}}^{\mathsf{O}_n}$ that breaks the pseudo-randomness of $G^{\mathsf{O}_n}(\cdot)$. This gives us the desired contradiction, because $G^{\mathsf{O}_n}(\cdot)$ is a PRG given that $\mathcal{O} = \{\mathsf{O}_n\}_{n\in\mathbb{N}}$ is one-way.

Formally, we assume for contradiction that there exists a polynomial $\mathsf{poly}_{\mathrm{PRG}}(\cdot)$ such that, for infinitely many $n \in \mathbb{N}$, the following holds:

$$\Pr_{\mathcal{O},\mathcal{O}',X_n,\widetilde{Q}_n^{\mathsf{Easy}}}\left[G^{\mathsf{O}_n}(X_n) \in G^{\mathsf{O}'_n \diamond \widetilde{Q}_h^{\mathsf{Easy}}}\left(\{0,1\}^{\ell(n)}\right)\right] \ge \frac{1}{2} + \frac{1}{\mathsf{poly}_{\mathrm{PRG}}(n)}. \tag{5.23}$$

On input $Y_n \in \{0,1\}^{\ell(n)+1}$, $\mathcal{A}_{\mathrm{PRG}}^{\mathsf{O}_n}$ proceeds as follows:

1. Compute $\widetilde{Q}_n^{\mathsf{Easy}}$ using Algorithm 1;

2. $\mathcal{A}_{\mathrm{PRG}}^{\mathsf{O}_n}$ samples O'_n uniformly from all functions mapping $\{0,1\}^n$ to $\{0,1\}^n$. With the $\widetilde{Q}_n^{\mathsf{Easy}}$ from Step 1, $\mathcal{A}_{\mathrm{PRG}}^{\mathsf{O}_n}$ now has the full description of $\mathsf{O}'_n \diamond \widetilde{Q}_n^{\mathsf{Easy}}$. $\mathcal{A}_{\mathrm{PRG}}^{\mathsf{O}_n}$ then tests if $Y \in G^{\mathsf{O}'_n \diamond \widetilde{Q}_n^{\mathsf{Easy}}}\big(\{0,1\}^{\ell(n)}\big)$. We remark that it takes exponential computation to sample O'_n. However, this step does *not* incur any calls to the oracle O_n.
3. **Output:** If $Y_n \in G^{\mathsf{O}'_n \diamond \widetilde{Q}_n^{\mathsf{Easy}}}\big(\{0,1\}^{\ell(n)}\big)$, output 1; otherwise, output 0.

If Y_n is the output of $G^{\mathsf{O}_n}(\cdot)$ on a random X_n, then it follows from above description of $\mathcal{A}_{\mathrm{PRG}}^{\mathsf{O}_n}$ that

$$\begin{aligned}\Pr_{\mathcal{O},\mathcal{O}',X_n}\left[\mathcal{A}_{\mathrm{PRG}}^{\mathsf{O}_n}\big(G^{\mathsf{O}_n}(X_n)\big)=1\right] &= \Pr_{\mathcal{O},\mathcal{O}',X_n,\widetilde{Q}_n^{\mathsf{Easy}}}\left[G^{\mathsf{O}_n}(X_n)\in G^{\mathsf{O}'_n \diamond \widetilde{Q}_n^{\mathsf{Easy}}}\big(\{0,1\}^{\ell(n)}\big)\right]\\ &\geq \frac{1}{2}+\mathsf{negl}(n); \qquad (5.24)\end{aligned}$$

If $Y_n \xleftarrow{\$} \{0,1\}^{\ell(n)+1}$, then the oracle $\mathsf{O}'_n \diamond \widetilde{Q}_n^{\mathsf{Easy}}$ and Y_n are independently distributed. We emphasize that although $\widetilde{Q}_n^{\mathsf{Easy}}$ is obtained by running $\mathsf{Sim}_V^{\mathsf{O}_n}$ on Y_n (for $N = 3n/c^2$ times), for each $(q, \mathsf{O}_n(q)) \in \widetilde{Q}_n^{\mathsf{Easy}}$, the answer $\mathsf{O}_n(q)$ is independent of Y_n. Therefore, Y_n is independent of $\mathsf{O}'_n \diamond \widetilde{Q}_n^{\mathsf{Easy}}$. In this case, Y_n is the range of the PRG with probability exactly 1/2. Formally,

$$\begin{aligned}\Pr_{\mathcal{O},\mathcal{O}',Y_n}\left[\mathcal{A}_{\mathrm{PRG}}^{\mathsf{O}_n}(Y_n)=1\right] &= \Pr_{\mathcal{O},\mathcal{O}',Y_n,\widetilde{Q}_n^{\mathsf{Easy}}}\left[Y_n\in G^{\mathsf{O}'_n \diamond \widetilde{Q}_n^{\mathsf{Easy}}}\big(\{0,1\}^{\ell(n)}\big)\right]\\ &= \Pr_{Y_n,\mathsf{O}''_n}\left[Y_n \in G^{\mathsf{O}''_n}\big(\{0,1\}^{\ell(n)}\big)\right]=\frac{1}{2} \qquad (5.25)\end{aligned}$$

Expressions (5.24) and (5.25) imply that $\mathcal{A}_{\mathrm{PRG}}^{\mathsf{O}_n}$ breaks the pseudo-randomness of $G^{\mathsf{O}_n}(\cdot)$. This completes the proof of Claim 5.3.7. □

This completes the proof of Claim 5.3.3.

5.4 Proof-Based One-Way Functions

5.4.1 Definition

Definition 5.4.1 (Proof-Based OWFs). *Let $\lambda \in \mathbb{N}$ be the security parameter. Let $a(\cdot)$, $b(\cdot)$ and $c(\cdot)$ be polynomials. A* proof-based one-way function *consists of a function $F_\lambda : \{0,1\}^{a(\lambda)} \times \{0,1\}^{b(\lambda)} \to \{0,1\}^{c(\lambda)}$ and a protocol $\Pi = (S, R)$ of a pair of* PPT *machines. We use $(X, Y) \leftarrow \langle S(1^\lambda, \mathsf{x}), R(1^\lambda, \mathsf{r})\rangle$ to denote the execution of protocol Π where the security parameter is λ, the inputs to S and R are x and r respectively, and the outputs of S and R are X and Y respectively. Let $Y = \perp$ denote that R aborts in the execution. The following conditions hold:*

- **One-Wayness.** *The function $\{F_\lambda\}_\lambda$ is one-way in the following sense:*
  - Easy to compute: *for all $\lambda \in \mathbb{N}$ and all $(\mathsf{x}, \mathsf{r}) \in \{0,1\}^{a(\lambda)} \times \{0,1\}^{b(\lambda)}$, $F_\lambda(\mathsf{x}\|\mathsf{r})$ can be computed in polynomial time on λ.*

Figure 5.4.1: Functionality $\mathcal{F}_F$ for Proof-Based OWFs

The ideal functionality $\mathcal{F}_F$ interacts with a sender S and a receiver R. Upon receiving the input $\mathsf{x} \in \{0,1\}^{a(\lambda)}$ from S and $\mathsf{r} \in \{0,1\}^{b(\lambda)}$ from R, the functionality $\mathcal{F}_F$ sends $\mathsf{x}\|\mathsf{r}$ to S, and $F(\mathsf{x}\|\mathsf{r})$ to R.

∗ Hard to invert: *for any non-uniform PPT adversary $\mathcal{A}$, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that* $\forall \mathsf{r} \in \{0,1\}^{b(\lambda)}$,

$$\Pr[\mathsf{x} \xleftarrow{\$} \{0,1\}^{a(\lambda)}, X^* \leftarrow \mathcal{A}\left(1^\lambda, F_\lambda(\mathsf{x}\|\mathsf{r})\right) : F_\lambda(\mathsf{x}\|\mathsf{r}) = F_\lambda(X^*)] \leq \mathsf{negl}(\lambda),$$

– **Completeness.** *The protocol Π computes the ideal functionality $\mathcal{F}_F$ defined in Figure 5.4.1. Namely, $\forall \lambda \in \mathbb{N}$, $\forall \mathsf{x} \in \{0,1\}^{a(\lambda)}$ and $\forall \mathsf{r} \in \{0,1\}^{b(\lambda)}$, if $(X,Y) \leftarrow \langle S(1^\lambda,\mathsf{x}), R(1^\lambda,\mathsf{r})\rangle$, then $X = \mathsf{x}\|\mathsf{r}$ and $Y = F_\lambda(\mathsf{x}\|\mathsf{r})$.*

– **Soundness.** *For every* PPT *machine S^* and every auxiliary input $z \in \{0,1\}^*$, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that*

$$\Pr\left[\begin{array}{l} \mathsf{r} \xleftarrow{\$} \{0,1\}^{b(\lambda)}; \\ (\cdot, Y) \leftarrow \langle S^*(1^\lambda, z), R(1^\lambda, \mathsf{r})\rangle \end{array} : \begin{array}{l} Y \neq \bot \textbf{\textit{ and}} \\ \nexists \mathsf{x} \ s.t. \ F_\lambda(\mathsf{x}\|\mathsf{r}) = Y \end{array}\right] \leq \mathsf{negl}(\lambda),$$

– **Zero-Knowledge.** *This property is defined by requiring* only *security against corrupted R in the ideal-real paradigm for 2PC w.r.t. the ideal functionality $\mathcal{F}_F$ in Figure 5.4.1. Concretely, denote by $\mathsf{REAL}_{\Pi,\mathcal{A}(z)}(1^\lambda,\mathsf{x},\mathsf{r})$ the random variable consisting of the output of S and the output of the adversary $\mathcal{A}$ controlling R in an execution of Π, where x is the input to S and r to R. Similarly, denote by $\mathsf{IDEAL}_{\mathcal{F}_F,\mathsf{Sim}(z)}(1^\lambda,\mathsf{x},\mathsf{r})$ the corresponding output of S and Sim from the ideal execution.*[7] *Then there exist a PPT simulator Sim such that for any PPT adversary $\mathcal{A}$, $\forall \mathsf{x} \in \{0,1\}^{a(\lambda)}$, $\forall \mathsf{r} \in \{0,1\}^{b(\lambda)}$, and $\forall z \in \{0,1\}^*$, $\left\{\mathsf{REAL}_{\Pi,\mathcal{A}(z)}(1^\lambda,\mathsf{x},\mathsf{r})\right\}_{\lambda\in\mathbb{N}} \overset{c}{\approx} \left\{\mathsf{IDEAL}_{\mathcal{F}_F,\mathsf{Sim}(z)}(1^\lambda,\mathsf{x},\mathsf{r})\right\}_{\lambda\in\mathbb{N}}$.*

If the constructions of both F and Π makes only black-box access to other primitives, we call this a black-box PB-OWF.

5.4.2 Our Construction

Following the high-level idea described in Section 5.1.2, we show that PB-OWFs can be obtained assuming black-box access to OWFs.

Theorem 5.4.1 (Black-Box PB-OWFs from OWFs)**.** *There exists a PB-OWF that satisfies Definition 5.4.1 and makes only black-box use of OWFs.*

Our construction consists of a one-way function F^f (Construction 5.4.1) together with a protocol Π_F^f (Protocol 5.4.1). The construction relies on the following building blocks:

– a one-way function f;

[7] We refer the reader to [Gol04] for a detailed description of the ideal and real executions.

Construction 5.4.1: One-Way Function F^f

Let $m(\lambda)$ and $n(\lambda)$ be polynomials on λ. Let $0 < \delta < 1$ be a constant, and $k(\lambda) = \delta n(\lambda)$. Let $t(\lambda) = \log^2(\lambda)$ (see Remark 5.4.1). Assume that $f : \{0,1\}^\lambda \to \{0,1\}^{m(\lambda)}$ is a one-way function. On input $\mathsf{x} \in \{0,1\}^{n\lambda+(\log(n)+m)k}$ and $\mathsf{r} \in \{0,1\}^{t\log(n)}$, F^f parses them as

$$\mathsf{x} = (x_1,\ldots,x_n)\|(p_1, y'_{p_1}),\ldots,(p_k, y'_{p_k}), \text{ and } \mathsf{r} = (b_1,\ldots,b_t),$$

where $|x_i| = \lambda$, $|y'_{p_i}| = m$, $\{p_i\}_{i\in[k]}$ is a size-k subset of $[n]$, and $\{b_i\}_{i\in[t]}$ is a size-t subset of $[n]$. F^f computes via its oracle access to $f(\cdot)$ the values $(y_1,\ldots,y_n)$, where $y_i = f(x_i)$ for all $i \in [n]$. Then, it computes $\mathsf{s} = (s_1,\ldots,s_n)$ as follows:

1. if $\{p_1,\ldots,p_k\} \cap \{b_1,\ldots,b_t\} \neq \emptyset$, then let $s_i := y_i$ for all $i \in [n]$.
2. if $\{p_1,\ldots,p_k\} \cap \{b_1,\ldots,b_t\} = \emptyset$, then let $s_i := \begin{cases} y'_i & i \in \{p_1,\ldots,p_k\} \\ y_i & i \in [n] \setminus \{p_1,\ldots,p_k\} \end{cases}$.

It finally outputs $Y = (s_1,\ldots,s_n)\|(x_{b_1},\ldots,x_{b_t})\|(b_1,\ldots,b_t)$.

– a zero-knowledge commit-and-prove protocol $\Pi_{\mathsf{ZKCnP}} = (\mathsf{BBCom}, \mathsf{BBProve})$ as per Definition 5.2.2. Such protocols can also be constructed assuming only black-box access to f.

It follows immediately from the description that our construction makes only black-box access to OWFs.

Remark 5.4.1 (On the Parameters in Construction 5.4.1)**.** *The choice of $t(\lambda) = \log^2(\lambda)$ is somewhat arbitrary. In fact, any $t(\lambda) = \omega(\log\lambda)$ works as long as $(n-k-t)$ is some positive polynomial of λ for sufficiently large λ. This is to ensure that we can prove one-wayness in Lemma 5.4.1* and *$(1-\delta)^t$ is negligible on λ, which is needed when we prove soundness (Claim 5.4.1). We also remark that the role of r is to specify a size-t subset of $[n]$. The canonical way of mapping r to a size-t subset of $[n]$ may consume slightly less randomness than $|\mathsf{r}| = t\log(n)$. For simplicity, we forego further discussion and assume that there is a deterministic bijection between $\{0,1\}^{t\log(n)}$ and all size-t subsets of $[n]$. Similarly, the $\{p_1,\ldots,p_k\}$ are interpreted as a size-k subset of $[n]$, though we assign each p_i a length of* $\log(n)$.

Protocol 5.4.1: Protocol Π_F^f for Our Proof-Based One-Way Function

Let f, m, n, t, and k be as in Construction 5.4.1.

Input: the security parameter 1^λ is the common input. Sender S takes $\mathsf{x} \in \{0,1\}^{n\lambda+(\log(n)+m)k}$ as its private input; receiver R takes $\mathsf{r} \in \{0,1\}^{t\cdot\log(n)}$ as its private input.

1. S parses the input as $\mathsf{x} = (x_1,\ldots,x_n)\|(p_1, y'_{p_1}),\ldots,(p_k, y'_{p_k})$, where $|x_i| = \lambda$ for all $i \in [n]$, $|y'_{p_j}| = m$ for all $j \in [k]$, and $\{p_i\}_{i\in[k]}$ forms a size-k subset of $[n]$. S defines a

$2 \times n$ matrix $M = \begin{bmatrix} x_1 & \cdots & x_n \\ y_1 & \cdots & y_n \end{bmatrix}$, where $y_i = f(x_i)$ for all $i \in [n]$.

2. S and R execute $\mathsf{BBCom}(\alpha)$, the **Commit** stage of Π_{ZKCnP}, where S commits to the value
$$\alpha := M \| (p_1, y'_{p_1}), \ldots, (p_k, y'_{p_k}). \tag{5.26}$$
3. R sends r to S.
4. S interprets r as a size-t subset $(b_1, \ldots, b_t) \subseteq [n]$. S then defines $M_{\mathsf{r}} = \begin{bmatrix} x_{b_1} & \cdots & x_{b_t} \\ y_{b_1} & \cdots & y_{b_t} \end{bmatrix}$, i.e. the columns of M specified by r. S also computes $\mathsf{s} = (s_1, \ldots, s_n)$ in the way specified in Construction 5.4.1. S sends to R the values M_{r} and s.
5. With M_{r}, R checks (via its oracle access to $f(\cdot)$) if $f(x_{b_i}) = y_{b_i}$ holds for all $i \in [t]$; R also checks if $s_{b_i} = y_{b_i}$ holds for all $i \in [t]$. If all the checks pass, R proceeds to next step; otherwise, R halts and outputs $\bot$.
6. S and R execute $\mathsf{BBProve}$, the **Prove** stage of Π_{ZKCnP}, where S proves that it performs Stage 4 honestly. Namely, S proves that the α committed at Stage 2 satisfies the following conditions:

   (a) the values $\{p_1, \ldots, p_k\}$ contained in α form a size-k subset of $[n]$; **and**

   (b) the M_{r} does consist of the columns in M specified by r; **and**

   (c) The $\mathsf{s} = (s_1, \ldots, s_n)$ satisfies the following conditions:

   – if $\{p_1, \ldots, p_k\} \cap \{b_1, \ldots, b_t\} \neq \emptyset$, then $s_i = y_i$ for all $i \in [n]$.

   – if $\{p_1, \ldots, p_k\} \cap \{b_1, \ldots, b_t\} = \emptyset$, then $s_i = \begin{cases} y'_i & i \in \{p_1, \ldots, p_k\} \\ y_i & i \in [n] \setminus \{p_1, \ldots, p_k\} \end{cases}$.

   We remark that these conditions can indeed be expressed a predicate ϕ on the α committed at Stage 2. For completeness, we show the formal definition of ϕ in Figure 5.4.2. It is also worth noting that predicate ϕ needs to have the values r and s hard-wired, which are defined at Stages 3 and 4 respectively. This is why we need a Π_{ZKCnP} that allows us to defer the definition of the predicate until the **Prove** stage (Definition 5.2.2).
7. **(Receiver's Output).** R outputs $Y = (s_1, \ldots, s_n) \| (x_{b_1}, \ldots x_{b_t}) \| (b_1, \ldots, b_t)$.
8. **(Sender's Output).** S outputs $X = (x_1, \ldots, x_n) \| (p_1, y'_{p_1}), \ldots, (p_k, y'_{p_k}) \| (b_1, \ldots, b_t)$.

5.4.3 Security Proof for Our PB-OWFs

5.4.3.1 Proof Overview

Before presenting the formal proof of security, we first provide an overview.

One-wayness, completeness, and ZK follow from rather standard techniques. In the following, let us explain more about the soundness proof (shown formally as Lemma 5.4.2).

First, note that the $\mathsf{r} = \{b_1, \ldots, b_t\}$ sent by R in Stage 3 is a size-t random subset of

Figure 5.4.2: Predicate $\phi_{\lambda,m,t,n,k,\mathsf{r},\mathsf{s}}(\cdot)$

Predicate ϕ has the values $(\lambda, m, t, n, k, \mathsf{r}, \mathsf{s})$ (as defined in Protocol 5.4.1) hard-wired. On the input α, $\phi_{\lambda,m,t,n,k,r,s}(\alpha) = 1$ if and only if all of the following hold:

- the α can be parsed as $M\|(p_1, y'_{p_1}), \ldots, (p_k, y'_{p_k})$, where $M = \begin{bmatrix} x_1 & \cdots & x_n \\ y_1 & \cdots & y_n \end{bmatrix}$ such that $|x_j| = \lambda$ and $|y_j| = m\ \forall j \in [n]$, $|p_i| = \log(n)$ and $|y'_{p_i}| = m\ \forall i \in [k]$; **and**
- the values $\{p_1, \ldots, p_k\}$ form a size-k subset of $[n]$; **and**
- the M_r consists of the columns in M specified by r; **and**
- the $\mathsf{s} = (s_1, \ldots, s_n)$ satisfy the following requirement (recall that the $\{b_1, \ldots, b_t\}$ are from r):
  - $*$ if $\{p_1, \ldots, p_k\} \cap \{b_1, \ldots, b_t\} \neq \emptyset$, then $s_i = y_i$ for all $i \in [n]$.
  - $*$ if $\{p_1, \ldots, p_k\} \cap \{b_1, \ldots, b_t\} = \emptyset$, then $s_i = \begin{cases} y'_i & i \in \{p_1, \ldots, p_k\} \\ y_i & i \in [n] \setminus \{p_1, \ldots, p_k\} \end{cases}$.

$[n]$. It will overlap with $\{p_1, \ldots, p_k\}$ with negligible probability. Therefore, the **Editing** condition will almost never be triggered during a real execution of Protocol 5.4.1, thus can be safely ignored.

Stages 2 to 5 can be though as the following cut-and-choose procedure: the sender computes $\{y_i = f(x_i)\}_{i\in[n]}$; then the receiver checks t of them randomly. This ensures that a malicious S^* cannot cheat on more than $k = \delta n$ of the y_i's. We prove this statement formally in Claim 5.4.1, which requires us to handle extra technicalities due to the commit-and-prove structure and **Editing** condition. But this claim implies that a non-aborting Y output by an honest receiver contains at most $k = \delta n$ many s_i's that does *not* have a preimage under f (except with negligible probability). Let us assume w.l.o.g. that there are exactly k such "no-preimage" s_i's, which can be denoted as $\{s_{p_1}, \ldots, s_{p_k}\}$ (i.e. we denote the indices of these no-preimage s_i's by $\{p_1, \ldots, p_k\}$). Then, for each s_i where $i \in [n] \setminus \{p_1, \ldots, p_k\}$, this s_i must have (at least) one preimage under $f(\cdot)$. We denote an arbitrary preimage of such s_i as $f^{-1}(s_i)$. In particular, if i is equal to some $b_j \in \{b_1, \ldots, b_t\}$, the Y already contains the preimage for s_{b_j}, which is x_{b_j}.

We emphasize that, conditioned on $Y \neq \bot$, we have $\{p_1, \ldots, p_k\} \cap \{b_1, \ldots, b_t\} = \emptyset$. To see this, recall that R checks at Stage 5 that $y_{b_i} = f(x_{b_i})$ *and* $s_{b_i} = y_{b_i}$ for all $b_i \in \{b_1, \ldots, b_t\}$. If there is a p_i falling in the set $\{b_1, \ldots, b_t\}$, then s_{p_i} $(= y_{p_i})$ does not have a preimage under $f(\cdot)$. Then, R will output $Y = \bot$ at Stage 5.

With these observations, we show in the following how to construct x and r such that $F^f(\mathsf{x}\|\mathsf{r}) = Y$. At a high-level, we take advantage of Case 2. We will use the no-preimage s_{p_i}'s together with their indices as the (p_i, y'_{p_i}) part in x. We will set r to the $\{b_1, \ldots, b_t\}$ contained in Y. Since $\{b_1, \ldots, b_t\} \cap \{p_1, \ldots, p_k\} = \emptyset$, the function F^f will put the no-preimage s_{p_i}'s at the positions specified by p_i's (according to Case 2), which will give us Y. Concretely, we set:

$$\mathsf{x} = (x'_1, \ldots, x'_n)\|(p_1, s_{p_1}), \ldots, (p_k, s_{p_k}) \text{ and } \mathsf{r} = (b_1, \ldots, b_t),$$

where x'_i's are defined as follows: $\forall i \in [n],\ x'_i = \begin{cases} x_i & i \in \{b_1, \ldots, b_t\} \\ 0^\lambda & i \in \{p_1, \ldots, p_k\} \\ f^{-1}(s_i) & \text{otherwise} \end{cases}$.

We remark that $f^{-1}(s_i)$ may not be efficiently computable (indeed, f is a one-way function). But the above proof only relies on the *existence* of $f^{-1}(s_i)$. Also, we have $\{p_1 \ldots, p_k\} \cap \{b_1, \ldots, b_t\} = \emptyset$. It then follows from the description in Construction 5.4.1 (in particular, Case 2) that $F^f(\mathsf{x}\|\mathsf{r}) = Y$.

5.4.3.2 The Full Proof

We now formally prove that the function F^f in Construction 5.4.1 and Protocol 5.4.1 constitute a black-box OWF with proof (as per Definition 5.4.1). In Lemma 5.4.1, we prove that F^f satisfies the one-wayness requirement in Definition 5.4.1. Given Construction 5.4.1, the completeness of Protocol 5.4.1 follows immediately by construction. We then establish the soundness and zero-knowledge property for Protocol 5.4.1 in Lemmas 5.4.2 and 5.4.3 respectively.

Lemma 5.4.1 (One-Wayness)**.** *The function F^f in Construction 5.4.1 is one-way as defined in Definition 5.4.1.*

Proof. From the description of Construction 5.4.1, it is easy to see that F^f is efficiently computable. In the following, we show that it is also "hard to invert".

Assume for contradiction that F^f is not hard to invert, i.e. there exist a PPT $\mathcal{A}_F$ and $\mathsf{r} \in \{0,1\}^{t\log(n)}$ such that for $\mathsf{x} \xleftarrow{\$} \{0,1\}^{n\lambda+(\log(n)+m)k}$, $\mathcal{A}_F$ inverts $F^f(\mathsf{x}\|\mathsf{r})$ with non-negligible probability. We show how to construct a PPT $\mathcal{A}_f$ that inverts f with non-negligible probability.

On input y^*, $\mathcal{A}_f$ first samples a string $\mathsf{x} \xleftarrow{\$} \{0,1\}^{n\lambda+(\log(n)+m)k}$ and computes the following Y value as per Construction 5.4.1:

$$Y = F^f(\mathsf{x}\|\mathsf{r}) = (s_1, \ldots, s_n)\|(x_{b_1}, \ldots, x_{b_t})\|(b_1, \ldots, b_t).$$

It then samples $i^* \xleftarrow{\$} [n] \setminus \{b_1, \ldots, b_t\}$, and gets Y' by substituting the s_{i^*} in Y with y^*. $\mathcal{A}_f$ then feeds Y' to $\mathcal{A}_F$ and receives X', which is supposed to be the preimage of Y' under F^f. In the following, we argue that X' contains the preimage of y^* under f with non-negligible probability.

Suppose that $\mathcal{A}_F$ produces an X' satisfying $F^f(X') = Y'$. This X' must be of the following form:

$$X' = (x_1, \ldots, x_n)\|(p_1, y'_{p_1}), \ldots, (p_k, y'_{p_k})\|(b_1, \ldots, b_t).$$

Then, by Construction 5.4.1, we must have $f(x_{i^*}) = y^*$ except for the "bad case" where $\{p_1, \ldots, p_k\} \cap \{b_1, \ldots, b_t\} = \emptyset$ *and* there is a $p_j = i^*$ (in which case x_{i^*} could be arbitrary as long as $\mathcal{A}_F$ sets $y'_{p_j} = y^*$). However, since i^* is picked uniformly from $[n] \setminus \{b_1, \ldots, b_t\}$, the "bad case" happens with probability $\leq 1/(n-t-k)$, which is $1/\mathsf{poly}(\lambda)$ by our choice of n, t, and k. Thus, if $\mathcal{A}_F$ breaks the one-wayness of F^f with some non-negligible probability $\varepsilon(\lambda)$, $\mathcal{A}_f$ will break the one-wayness of f with probability $\geq (1 - 1/\mathsf{poly}(\lambda)) \cdot \varepsilon(\lambda)$, which is non-negligible. □

Lemma 5.4.2 (Soundness.). *For every* PPT *machine* S^* *and every auxiliary input* $z \in \{0,1\}^*$, *there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that*

$$\Pr\left[\begin{array}{l} \mathsf{r} \xleftarrow{\$} \{0,1\}^{t\cdot\log(n)}; \\ (\cdot, Y) \leftarrow \langle S^*(1^\lambda, z), R(1^\lambda, \mathsf{r})\rangle \end{array} : \begin{array}{l} Y \neq \bot \textbf{ and} \\ \nexists \mathsf{x} \; s.t. \; F^f(\mathsf{x}\|\mathsf{r}) = Y \end{array}\right] \leq \mathsf{negl}(\lambda). \tag{5.27}$$

Proof. Let us first define "non-trivial" S^*'s, which are the malicious provers that can make the honest receiver accepts with non-negligible probability.

Definition 5.4.2 (Non-Trivial S^*). *A PPT machine* S^* *is* non-trivial *if there exists some auxiliary input* $z \in \{0,1\}^*$ *such that the following holds: there exits a polynomial* $\mathsf{poly}(\cdot)$ *such that for infinitely many* $\lambda \in \mathbb{N}$,

$$\Pr\left[\mathsf{r} \xleftarrow{\$} \{0,1\}^{t\cdot\log(n)}; (\cdot, Y) \leftarrow \langle S^*(1^\lambda, z), R(1^\lambda, \mathsf{r})\rangle : Y \neq \bot\right] \geq \frac{1}{\mathsf{poly}(\lambda)}. \tag{5.28}$$

It is not hard to see that if a PPT machine $S^*(1^\lambda, z)$ is *not* non-trivial, then Inequality (5.27) holds immediately. Therefore, to prove Lemma 5.4.2, we only need to focus on the non-trivial S^*'s.

For any Y of the form $(s_1, \ldots, s_n)\|(x_{b_1}, \ldots, x_{b_t})\|(b_1, \ldots, b_t)$, we define the following event:

- Bad_Y: there are more than $k = \delta n$ number of s_i's such that $\nexists x$ s.t. $f(x) = s_i$.

In the following, we present a claim (Claim 5.4.1). We will first show how to prove Lemma 5.4.2 assuming that Claim 5.4.1 holds and then present its proof.

Claim 5.4.1. *For every non-trivial* S^* *with the* $z \in \{0,1\}^*$ *satisfying Inequality (5.28), there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that*

$$\Pr\left[\mathsf{r} \xleftarrow{\$} \{0,1\}^{t\cdot\log(n)}; (\cdot, Y) \leftarrow \langle S^*(1^\lambda, z), R(1^\lambda, \mathsf{r})\rangle : \mathsf{Bad}_Y \;\middle|\; Y \neq \bot\right] \leq \mathsf{negl}(\lambda). \tag{5.29}$$

Consider the Y output by R from $\langle S^*(1^\lambda, z), R(1^\lambda, \mathsf{r})\rangle$, where $S^*(1^\lambda, z)$ is non-trivial and $\mathsf{r} \xleftarrow{\$} \{0,1\}^{t\log(n)}$. If $Y \neq \bot$, it must be of the following form:

$$Y = (s_1, \ldots, s_n)\|(x_{b_1}, \ldots, x_{b_t})\|(b_1, \ldots, b_t),$$

where $s_i = f(x_i)$ for all $i \in \{b_1, \ldots, b_t\}$. Assuming that Claim 5.4.1 holds, to finish the proof of Lemma 5.4.2, it suffices to show the following claim:

- Conditioned on $Y \neq \bot$, if Bad_Y does not happen, then there exist an x and an r such that $F^f(\mathsf{x}\|\mathsf{r}) = Y$.

Conditioned on $Y \neq \bot$, Claim 5.4.1 implies that there are at most $k = \delta n$ many s_i's that do not have a preimage under f (except with negligible probability). In the following, we assume w.l.o.g. that there are exactly k such "no-preimage" s_i's. We denote them as $\{s_{p_1}, \ldots, s_{p_k}\}$ (i.e. we denote the indices of these no-preimage s_i's by $\{p_1, \ldots, p_k\}$). Then, for each s_i where $i \in [n] \setminus \{p_1, \ldots, p_k\}$, this s_i must have (at least) one preimage under $f(\cdot)$. We denote an arbitrary preimage of such s_i as $f^{-1}(s_i)$. In particular, if i is equal to some $b_j \in \{b_1, \ldots, b_t\}$, then Y already contains the preimage for s_{b_j}, which is x_{b_j}.

We emphasize that, conditioned on $Y \neq \bot$, $\{p_1, \ldots, p_k\} \cap \{b_1, \ldots, b_t\} = \emptyset$. To see this, recall that R checks at Stage 5 that $y_{b_i} = f(x_{b_i})$ *and* $s_{b_i} = y_{b_i}$ for all $b_i \in \{b_1, \ldots, b_t\}$. If there is a p_i falling in the set $\{b_1, \ldots, b_t\}$, then s_{p_i} $(= y_{p_i})$ does not have a preimage under $f(\cdot)$. Then, R will output $Y = \bot$ at Stage 5.

With these observations, we show in the following how to construct x and r such that $F^f(\mathsf{x}\|\mathsf{r}) = Y$, assuming that Bad_Y does not happen. At a high-level, we take advantage of Case 2. we will use the no-preimage s_{p_i}'s together with their indices as the (p_i, y'_{p_i}) part in x. We will set r to the $\{b_1, \ldots, b_t\}$ contained in Y. Since $\{b_1, \ldots, b_t\} \cap \{p_1, \ldots, p_k\} = \emptyset$, the function F^f will put the no-preimage s_{p_i}'s at the positions specified by p_i's (according to Case 2), which will give us Y. Concretely, we set:

$$\mathsf{x} = (x'_1, \ldots, x'_n)\|(p_1, s_{p_1}), \ldots, (p_k, s_{p_k}) \text{ and } \mathsf{r} = (b_1, \ldots, b_t),$$

where x'_i's are defined as follows: $\forall i \in [n], \quad x'_i = \begin{cases} x_i & i \in \{b_1, \ldots, b_t\} \\ 0^\lambda & i \in \{p_1, \ldots, p_k\} \\ f^{-1}(s_i) & \text{otherwise} \end{cases}$. We remark that $f^{-1}(s_i)$ may not be efficiently computable (indeed, f is a one-way function). But this proof only relies on the *existence* of $f^{-1}(s_i)$. Also, we have $\{p_1 \ldots, p_k\} \cap \{b_1, \ldots, b_t\} = \emptyset$. It then follows from the description in Construction 5.4.1 (in particular, Case 2) that $F^f(\mathsf{x}\|\mathsf{r}) = Y$.

In the following, we show the proof for Claim 5.4.1, which will complete the proof for Lemma 5.4.2.

Proof of Claim 5.4.1. All the probabilities appearing in this proof are taken over the following random procedure:

$$\mathsf{r} \xleftarrow{\$} \{0,1\}^{t \cdot \log(n)}; (\cdot, Y) \leftarrow \langle S^*(1^\lambda, z), R(1^\lambda, \mathsf{r})\rangle,$$

where S^* and z are as described in Claim 5.4.1.

First, note that $\Pr[\mathsf{Bad}_Y \mid Y \neq \bot] \cdot \Pr[Y \neq \bot] = \Pr[\mathsf{Bad}_Y \wedge (Y \neq \bot)]$. Since S^* is non-trivial, we know from Inequality (5.28) that $\Pr[Y \neq \bot]$ is non-negligible. Therefore, to prove Inequality (5.29), it suffices to show

$$\Pr[\mathsf{Bad}_Y \wedge (Y \neq \bot)] \leq \mathsf{negl}(\lambda), \tag{5.30}$$

which we prove in the following.

Consider the execution $(\cdot, Y) \leftarrow \langle S^*(1^\lambda, z), R(1^\lambda, \mathsf{r})\rangle$ where $Y \neq \bot$. Observe that S^* at Stage 2 commits a value α[8] of the form shown in Expression (5.26). For this execution, we define the following sequence of events:

- E_1: $\{p_1, \ldots, p_k\} \cap \{b_1, \ldots, b_t\} = \emptyset$. The $\{p_i\}_{i \in [k]}$ are contained in α and $\{b_i\}_{i \in [t]}$ are contained in Y.
- E_2: $\forall i \in [n]$, $y_i = s_i$. The $\{y_i\}_{i \in [n]}$ are contained in the second row of M (which is in turn

[8]This α is well-defined as BBCom is statistically-binding.

contained in α), and $\{s_i\}_{i\in[n]}$ are contained in Y.

- E_3: The M_r sent by S^* at Stage 4 indeed consists of the columns of M specified by $\mathsf{r} = \{b_1, \ldots, b_t\}$ (contained in Y).
- E_4: there are more than $k = \delta n$ number of y_i's (contained in the second row of M) that do not have a preimage under $f(\cdot)$.

We first claim:

$$\Pr[E_4 \wedge (Y \neq \bot)] \leq \mathsf{negl}(\lambda). \tag{5.31}$$

To see that, first notice the following:

$$\begin{aligned}\Pr[E_4 \wedge (Y \neq \bot)] &= \Pr[E_4 \wedge (Y \neq \bot) \wedge E_3] + \Pr[E_4 \wedge (Y \neq \bot) \wedge \overline{E}_3] \\ &\leq \underbrace{\Pr[(Y \neq \bot) \mid E_3 \wedge E_4]}_{\mathsf{P}_1} + \underbrace{\Pr[(Y \neq \bot) \wedge \overline{E}_3]}_{\mathsf{P}_2}.\end{aligned}$$

We now show that both P_1 and P_2 are negligible. First, recall that R checks at Stage 5 if $y_i = f(x_i)$ for all columns $[x_i\ y_i]^T$ contained in M_r. Thus, conditioned on E_3, the receiver does not abort only if the set $\mathsf{r} = \{b_1, \ldots, b_t\}$ *does not* select any "bad" column $[x_i\ y_i]^T$ in M s.t. $y_i \neq f(x_i)$. Moreover, E_4 ensures that there are more than $k = \delta n$ such "bad" columns in M. Therefore, the probability P_1 is smaller than $(1-\delta)^t$, which is negligible as $0 < \delta < 1$ is a constant and t is $\omega(\log \lambda)$.

Also, observe that E_3 was actually proved at Stage 6 (as Item 6b) by S^* via the commit-and-prove protocol Π_ZKCnP. The soundness of Π_ZKCnP implies that P_2 is negligible.

Next, we make another claim:

$$\Pr[\mathsf{Bad}_Y \wedge (Y \neq \bot) \wedge \overline{E}_2] \leq \mathsf{negl}(\lambda). \tag{5.32}$$

To prove this inequality, notice the following:

$$\begin{aligned}\Pr[\mathsf{Bad}_Y \wedge (Y \neq \bot) \wedge \overline{E}_2] &= \Pr[\mathsf{Bad}_Y \wedge (Y \neq \bot) \wedge \overline{E}_2 \wedge E_1] + \Pr[\mathsf{Bad}_Y \wedge (Y \neq \bot) \wedge \overline{E}_2 \wedge \overline{E}_1] \\ &\leq \underbrace{\Pr[E_1]}_{\mathsf{P}_3} + \underbrace{\Pr[(Y \neq \bot) \wedge \overline{E}_2 \mid \overline{E}_1]}_{\mathsf{P}_4}.\end{aligned}$$

We now show that both P_3 and P_4 are negligible. Recall that E_1 is the event that the set $\{p_1, \ldots, p_k\}$ contained in α does not overlap with $\mathsf{r} = \{b_1, \ldots, b_t\}$. First, note that the Π_ZKCnP proof at Stage 6 ensures that the set $\{p_1, \ldots, p_k\}$ is a size-k subset of $[n]$ (i.e. Item 6a) except with negligible probability. Also, observe that the r is a size-t random subset of $[n]$ that is sampled *independently* of $\{p_1, \ldots, p_k\}$. Therefore, E_1 happens with probability $\leq (1-\delta)^t + \mathsf{negl}(\lambda)$, which is negligible.

According to Construction 5.4.1, if $\{p_1, \ldots, p_k\} \cap \{b_1, \ldots, b_t\} \neq \emptyset$ (which is exactly the event $\overline{E}_1$), then it must hold that $y_i = s_i$ for all $i \in [n]$ (which is exactly E_2). Also, recall that this condition is enforced by the $\mathsf{BBProve}$ performed by S^* at Stage 6 (see Item 6c). The soundness of the Π_ZKCnP guarantees that, conditioned on $\overline{E}_1$, if E_2 does not hold, then R will abort with overwhelming probability. Therefore, P_4 is negligible.

We are now ready to derive Inequality (5.30):

$$\begin{aligned}\Pr[\mathsf{Bad}_Y \wedge (Y \neq \bot)] &= \Pr[\mathsf{Bad}_Y \wedge (Y \neq \bot) \wedge E_2] + \Pr[\mathsf{Bad}_Y \wedge (Y \neq \bot) \wedge \overline{E}_2] \\ &= \Pr[E_4 \wedge (Y \neq \bot) \wedge E_2] + \Pr[\mathsf{Bad}_Y \wedge (Y \neq \bot) \wedge \overline{E}_2] \qquad (5.33) \\ &\leq \Pr[E_4 \wedge (Y \neq \bot)] + \Pr[\mathsf{Bad}_Y \wedge (Y \neq \bot) \wedge \overline{E}_2] \\ &\leq \mathsf{negl}(\lambda) \qquad (5.34)\end{aligned}$$

where Equation (5.33) follows from the fact that, conditioned on E_2, the events Bad_Y and E_4 are identical. Also, note that Inequality (5.34) follows from Inequalities (5.31) and (5.32).

This finishes the proof of Claim 5.4.1 and thus also the proof for Lemma 5.4.2. □

Lemma 5.4.3 (Zero-knowledge)**.** *Protocol 5.4.1 satisfies the zero-knowledge property as in Definition 5.4.1.*

Proof. To prove the zero-knowledge property, we need to show a PPT ideal-world simulator Sim for any PPT malicious receiver R^*. At a high-level, such a simulator can be constructed as follows. Sim will use the simulator of the commit-and-prove protocol Π_{ZKCnP} at Stages 2 and 6. This allows Sim to finish the interaction without knowing the sender's input x.

Formally, we will show a sequence of hybrids starting from the real execution between the honest sender and R^*, and show that the last hybrid is essentially the simulator we want. We use Out_{H_i} to denote the output of hybrid H_i.

Hybrid $H_0(1^\lambda, \mathsf{x}, z)$**.** This hybrid uses the strategy of the honest $S(1^\lambda, \mathsf{x})$ to interact with the corrupted receiver $R^*(1^\lambda, z)$. At the end of the execution, H_0 outputs whatever R^* outputs. This hybrid is exactly the real execution.

Hybrid $H_1(1^\lambda, \mathsf{x}, z)$**.** This hybrid is identical to the previous one, except that

- At Stage 2, instead of executing BBCom, H_1 uses the strategy of Sim_1 in its communication with R^*, where Sim_1 is the simulator for the **Commit** stage of Π_{ZKCnP} (see Definition 5.2.2).
- At Stage 6, instead of doing the proof honestly, H_1 uses the strategy of Sim_2in its communication with R^*, where Sim_1 is the simulator for the **Prove** stage of Π_{ZKCnP} (see Definition 5.2.2).

$\mathsf{Out}_{H_0} \overset{c}{\approx} \mathsf{Out}_{H_1}$: This is due to the ZK property of Π_{ZKCnP}.

The simulator $\mathsf{Sim}(1^\lambda, z)$**.** We now describe the simulator Sim for the ideal execution. Sim is identical to H_1, except that

- Sim does not need to execute Stage 1;
- Upon receiving r at Stage 3, Sim sends it to the idea functionality $\mathcal{F}_{Ff}$, and receives back the value $Y = (s_1, \ldots, s_n) \| (x_{b_1}, \ldots, x_{b_t}) \| (b_1, \ldots, b_t)$;
- At Stage 4, Sim sets $\widetilde{M}_{\mathsf{r}} = \begin{bmatrix} x_{b_1} & \cdots & x_{b_t} \\ s_{b_1} & \cdots & s_{b_t} \end{bmatrix}$, and sends to R the values $\widetilde{M}_{\mathsf{r}}$ and $\mathsf{s} = (s_1, \ldots, s_n)$, where the s_i's are those contained in Y.

We remark that, unlike H_1, Sim does not need to know the input x to the sender; the Y that it obtained from $\mathcal{F}_{Ff}$ contains all the necessary information to finish its interaction with R^*. In particular, although the x_{b_i}'s in $\widetilde{M}_{\mathsf{r}}$ and the s now come from the Y that Sim obtained from the ideal functionality, they are identically distributed to the ones generated by the honest sender in a real execution. It then follows that the output of Sim is identical to H_1.

This finishes the proof for Lemma 5.4.3. □

5.5 Proof-Based Pseudo-Random Generators

We can also define proof-based pseudo-random generators (PB-PRGs) in a similar way as for PB-OWFs. It consists of a two-input function $G^g(\cdot,\cdot)$ and a protocol $\Pi^g_G = (S^g, R^g)$ such that for any PRG g, $G^g(\cdot,\mathsf{r})$ is a PRG for any choice of r, and Π^g_G satisfies the same completeness, soundness, and ZK requirements as in Definition 5.4.1 but w.r.t. G^g.

Our PB-PRG can be constructed by simply replacing the oracle OWF f with a PRG g in both Construction 5.4.1 and Protocol 5.4.1 (our PB-OWF construction). There is one caveat: the output Y of Construction 5.4.1 contains the preimage x_{b_i} for y_{b_i} (or s_{b_i}). While this is fine for one-wayness, such a Y will not be pseudo-random, because an adversary can always learn if Y is in the range of $G^g(\cdot,\mathsf{r})$ by testing whether $y_{b_i} = g(x_{b_i})$. To fix this, in the output Y, we will place x_{b_i} in the position where we originally put y_{b_i} (and we can drop the $(x_{b_1},\ldots,x_{b_t})$ part from Y). We will show that this modification lead to a valid PB-PRG.

In the following, we present the definition, construction, and the security proof for PB-PRGs.

5.5.1 Definition

Definition 5.5.1 (Proof-Based PRGs). *Let $a(\lambda)$, $b(\lambda)$ and $c(\lambda)$ be polynomials on λ. A proof-based pseudorandom generator consists of function $G_\lambda : \{0,1\}^{a(\lambda)+b(\lambda)} \to \{0,1\}^{c(\lambda)}$ and a protocol $\Pi = (S,R)$ involving a pair of* PPT *machines. We use $(X,Y) \leftarrow \langle S(1^\lambda,\mathsf{x}), R(1^\lambda,\mathsf{r})\rangle$ to denote the execution of protocol Π where the security parameter is λ, the inputs to S and R are x and r respectively, and the outputs of S and R are X and Y respectively. Let $Y = \bot$ denote that R aborts in the execution. The following conditions hold:*

- **Pseudo-randomness.** *For every $\mathsf{r} \in \{0,1\}^{b(\lambda)}$, $G_\lambda(\cdot\|\mathsf{r})$ is a PRG on its first input. That is, it is efficiently computable, length stretching, and*

$$\{\mathsf{x} \xleftarrow{\$} \{0,1\}^{a(\lambda)} : G_\lambda(\mathsf{x}\|\mathsf{r})\}_{\lambda\in\mathbb{N}} \quad \overset{\mathrm{c}}{\approx} \quad \{U_{c(\lambda)}\}_{\lambda\in\mathbb{N}}.$$

- **Completeness.** $\forall\lambda \in \mathbb{N}$, $\forall \mathsf{x} \in \{0,1\}^{a(\lambda)}$ *and* $\forall \mathsf{r} \in \{0,1\}^{b(\lambda)}$, *if* $(X,Y) \leftarrow \langle S(1^\lambda,\mathsf{x}), R(1^\lambda,\mathsf{r})\rangle$, *then $X = \mathsf{x}\|\mathsf{r}$ and $Y = G_\lambda(\mathsf{x}\|\mathsf{r})$.*
- **Soundness.** *For every* PPT *machine S^* and every auxiliary input $z \in \{0,1\}^*$, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that*

$$\Pr\left[\begin{array}{l} \mathsf{r} \xleftarrow{\$} \{0,1\}^{b(\lambda)}; \\ (\cdot, Y) \leftarrow \langle S^*(1^\lambda, z), R(1^\lambda,\mathsf{r})\rangle \end{array} : \begin{array}{l} Y \neq \bot \ \textbf{\textit{and}} \\ \nexists \mathsf{x}\ s.t.\ G_\lambda(\mathsf{x}\|\mathsf{r}) = Y \end{array}\right] \leq \mathsf{negl}(\lambda)$$

where the probability is taken over the random sampling of r*,and the randomness used by* S^* *and* R*.*

– **Zero-Knowledge.** *This property is defined by requiring the security against corrupted* R *in the ideal-real paradigm for 2PC w.r.t. the ideal functionality* $\mathcal{F}_G$*, which is obtained by replacing* F *with* G *in Figure 5.4.1. Namely, there exist a PPT simulator* Sim *such that for any PPT adversary* $\mathcal{A}$*,* $\forall \mathsf{x} \in \{0,1\}^{a(\lambda)}$*,* $\forall \mathsf{r} \in \{0,1\}^{b(\lambda)}$*, and* $\forall z \in \{0,1\}^*$*,*

$$\left\{\mathsf{REAL}_{\Pi,\mathcal{A}(z)}(1^\lambda, \mathsf{x}, \mathsf{r})\right\}_{\lambda\in\mathbb{N}} \overset{\text{c}}{\approx} \left\{\mathsf{IDEAL}_{\mathcal{F}_G,\mathsf{Sim}(z)}(1^\lambda, \mathsf{x}, \mathsf{r})\right\}_{\lambda\in\mathbb{N}}.$$

where $\mathsf{REAL}_{\Pi,\mathcal{A}(z)}(1^\lambda, \mathsf{x}, \mathsf{r})$ *and* $\mathsf{IDEAL}_{\mathcal{F}_G,\mathsf{Sim}(z)}(1^\lambda, \mathsf{x}, \mathsf{r})$ *are defined in the same way as in Definition 5.4.1.*

5.5.2 Our Construction

We now present our construction for proof-based PRGs, thus establishing the following theorem.

Theorem 5.5.1. *There exists a PB-PRG that satisfies Definition 5.5.1 and makes only black-box use of PRGs.*

Our construction consists of a PRG G^g (Construction 5.5.1) together with a black-box protocol (Protocol 5.5.1) that proves the membership for G^g. The construction relies on the following building blocks:

– A pseudo-random generator g;

– A zero-knowledge commit-and-prove protocol $\Pi_{\mathsf{ZKCnP}} = (\mathsf{BBCom}, \mathsf{BBProve})$ as per Definition 5.2.2. Such protocols can be constructed assuming only black-box access to g.

Construction 5.5.1: Pseudo-Random Generator G^g

Let $m(\lambda)$ and $n(\lambda)$ be polynomials on λ. Let $0 < \delta < 1$ be a constant, and $k(\lambda) = \delta n(\lambda)$. Let $t(\lambda) = \log^2(\lambda)$. Assume that $g : \{0,1\}^\lambda \to \{0,1\}^{m(\lambda)}$ is a PRG. On input $\mathsf{x} \in \{0,1\}^{n\lambda + k(\log(n)+m)}$ and $\mathsf{r} \in \{0,1\}^{t\log(n)}$, G^g parses them as

$$\mathsf{x} = (x_1, \ldots, x_n)\|(p_1, y'_{p_1}), \ldots, (p_k, y'_{p_k}), \text{ and } \mathsf{r} = (b_1, \ldots, b_t),$$

where $|x_i| = \lambda$, $|y'_{p_i}| = m$, $\{p_i\}_{i\in[k]}$ is a size-k subset of $[n]$, and $\{b_i\}_{i\in[t]}$ is a size-t subset of $[n]$. G^g outputs $Y = (z_1, \ldots, z_n)$, which are computed as follows:

1. if $\{p_1, \ldots, p_k\} \cap \{b_1, \ldots, b_t\} \neq \emptyset$, then $z_i = \begin{cases} x_i & i \in \{b_1, \ldots, b_t\} \\ g(x_i) & \text{otherwise} \end{cases}$;

2. if $\{p_1, \ldots, p_k\} \cap \{b_1, \ldots, b_t\} = \emptyset$, then $z_i := \begin{cases} x_i & i \in \{b_1, \ldots, b_t\} \\ y'_i & i \in \{p_1, \ldots, p_k\} \\ g(x_i) & \text{otherwise} \end{cases}$.

Protocol 5.5.1: Protocol Π_G^g for Our Proof-Based Pseudorandom Generator

Let g, m, n, k and t be as in Construction 5.5.1.

Input: both parties take the security parameter 1^λ as the common input. Sender S takes a string $\mathsf{x} \in \{0,1\}^{n\lambda+(\log(n)+m)k}$ as private input; receiver R takes a string $\mathsf{r} \in \{0,1\}^{t \cdot \log(n)}$ as private input.

1. S parses the input as $\mathsf{x} = (x_1, \ldots, x_n) \| (p_1, y'_{p_1}), \ldots, (p_k, y'_{p_k})$, where $|x_i| = \lambda$ for all $i \in [n]$, $|y'_{p_j}| = m$ for all $j \in [k]$, and $\{p_i\}_{i \in [k]}$ forms a size-k subset of $[n]$. S defines a $2 \times n$ matrix $M = \begin{bmatrix} x_1 & \cdots & x_n \\ y_1 & \cdots & y_n \end{bmatrix}$, where $y_i = g(x_i)$ for all $i \in [n]$.
2. S and R execute $\mathsf{BBCom}(\alpha)$, the **Commit** stage of Π_{ZKCnP}, where S commits to the value
$$\alpha \coloneqq M \| (p_1, y'_{p_1}), \ldots, (p_k, y'_{p_k}). \tag{5.35}$$
3. R sends r to S.
4. S interprets r as a size-t subset $\{b_1, \ldots, b_t\} \subseteq [n]$, and defines $M_\mathsf{r} = \begin{bmatrix} x_{b_1} & \cdots & x_{b_t} \\ y_{b_1} & \cdots & y_{b_t} \end{bmatrix}$, i.e. the columns of M specified by r. S also computes $(z_1, \ldots, z_n)$ in the way specified in Construction 5.5.1. S sends to R the values M_r and $(z_1, \ldots, z_n)$.
5. With M_r, R checks (via its oracle access to $g(\cdot)$) if $g(x_{b_i}) = y_{b_i}$ holds for all $i \in [t]$. If all the checks pass, R proceeds to next step; otherwise, R halts and outputs $\perp$.
6. S and R execute $\mathsf{BBProve}$, the **Prove** stage of Π_{ZKCnP}, where S proves that it performs Stage 4 honestly. Namely, S proves that the α committed at Stage 2 satisfies the following conditions:

   (a) the values $\{p_1, \ldots, p_k\}$ contained in α form a size-k subset of $[n]$; **and**

   (b) the M_r does consist of the columns in M specified by r; **and**

   (c) the $(z_1, \ldots, z_t)$ satisfy the following conditions:

   – if $\{p_1, \ldots, p_k\} \cap \{b_1, \ldots, b_t\} \neq \emptyset$, then $z_i = \begin{cases} x_i & i \in \{b_1, \ldots, b_t\} \\ y_i & \text{otherwise} \end{cases}$;

   – if $\{p_1, \ldots, p_k\} \cap \{b_1, \ldots, b_t\} = \emptyset$, then $z_i \coloneqq \begin{cases} x_i & i \in \{b_1, \ldots, b_t\} \\ y'_i & i \in \{p_1, \ldots, p_k\} \\ y_i & \text{otherwise} \end{cases}$.

   Similar as in Protocol 5.4.1, these conditions can be expressed a predicate on α.
7. **(Receiver's Output).** R outputs $Y = (z_1, \ldots, z_n)$.
8. **(Sender's Output).** S outputs $X = (x_1, \ldots, x_n) \| (p_1, y'_{p_1}), \ldots, (p_k, y'_{p_k}) \| (b_1, \ldots, b_t)$.

It follows immediately from the description that our construction makes only black-box access to PRGs. In Section 5.5.3, we show that it satisfies definition Definition 5.5.1.

5.5.3 Proof of Security

In this section, we prove that the function G^g in Construction 5.5.1 and Protocol 5.5.1 constitute a black-box PRG with proof (as per Definition 5.5.1). In Lemma 5.5.1, we prove that G^g satisfies the pseudo-randomness requirement in Definition 5.5.1. Given the description of Construction 5.5.1, the completeness of Protocol 5.5.1 follows immediately by construction. We then establish the soundness and zero-knowledge property in Lemmas 5.5.2 and 5.5.3 respectively.

Lemma 5.5.1 (Pseudo-randomness of G^g)**.** *Construction 5.5.1 satisfies the pseudo-randomness property defined in Definition 5.5.1.*

Proof (Sketch). We first argue that G^g is length-stretching. To see that, note that the input and output are of the following length respectively:

$$|X| = |\mathsf{x}| + |\mathsf{r}| = \delta nm + \delta n \log(n) + \lambda n + t\log(n), \text{ and } |Y| = nm - tm + \lambda t,$$

where $0 < \delta < 1$ is a constant, $t = \log^2(\lambda)$ (see also Remark 5.4.1), and m and n are polynomials on λ. Note that we have control over choice of m and n. For example, If we set $n = \Omega(\lambda^2)$ and $|m| = \Omega(\log n)$, then the dominating term for the input length will be δmn, and the dominating term for the output length will be mn. Since $0 < \delta < 1$, G^g is length-stretching.

The pseudo-randomness of G^g follows from standard hybrid arguments. The only point that requires extra attention is that the input to G^g has two parts x and r, and we need to prove the pseudo-randomness of its output for *all* $\mathsf{r} \in \{0,1\}^{b(\lambda)}$. Note that since x is sampled randomly, Case 1 in Construction 5.5.1 happens with overwhelming probability. Therefore, the pseudo-randomness of G^g (for all r) can be reduced to the pseudo-randomness of g by standard hybrid technique. We omit the details. □

Lemma 5.5.2 (Soundness)**.** *Protocol* Π_{PRGP} *in Protocol 5.5.1 satisfies the soundness property defined in Definition 5.5.1. Namely, for every* PPT *machine* S^* *and every auxiliary input* $z \in \{0,1\}^*$*, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that*

$$\Pr\left[\begin{array}{l} \mathsf{r} \xleftarrow{\$} \{0,1\}^{t\log(n)}; \\ (\cdot, Y) \leftarrow \langle S^*(1^\lambda, z), R(1^\lambda, \mathsf{r})\rangle \end{array} : \begin{array}{l} Y \neq \bot \ \textbf{\textit{and}} \\ \nexists \mathsf{x} \ s.t. \ G^g(\mathsf{x}\|\mathsf{r}) = Y \end{array}\right] \leq \mathsf{negl}(\lambda). \tag{5.36}$$

Proof. Let us first define "non-trivial" S^*'s, which are the malicious provers that can make the honest receiver accept with non-negligible probability.

Definition 5.5.2 (Non-Trivial S^*)**.** *A PPT machine S^* is* non-trivial *if there exists some auxiliary input $z \in \{0,1\}^*$ such that the following holds: there exits a polynomial* $\mathsf{poly}(\cdot)$ *such that for infinitely many $\lambda \in \mathbb{N}$,*

$$\Pr\left[\mathsf{r} \xleftarrow{\$} \{0,1\}^{t\cdot\log(n)}; (\cdot, Y) \leftarrow \langle S^*(1^\lambda, z), R(1^\lambda, \mathsf{r})\rangle : Y \neq \bot\right] \geq \frac{1}{\mathsf{poly}(\lambda)}. \tag{5.37}$$

It is not hard to see that if a PPT machine $S^*(1^\lambda, z)$ is *not* non-trivial for all $z \in \{0,1\}^*$, then Inequality (5.36) holds immediately. Therefore, to prove Lemma 5.5.2, we only need to focus on the non-trivial S^*'s.

For any $(\cdot, Y) \leftarrow \langle S^*(1^\lambda, z), R(1^\lambda, \mathsf{r})\rangle$ such that $Y \neq \bot$, the Y can then be parsed as $(z_1, \ldots, z_n)$ such that $|z_i| = \lambda$ for $i \in \{b_1, \ldots, b_t\}$ and $|z_i| = m$ for $i \in [n] \setminus \{b_1, \ldots, b_t\}$. Recall that $\{b_1, \ldots, b_t\}$ is the size-t subset of $[n]$ specified by r. Given such a Y and r, we say that a z_i is "no-preimage" if it satisfies the following requirements:

- z_i is in the set $\{z_1, \ldots, z_n\} \setminus \{z_{b_1}, \ldots, z_{b_t}\}$; **and**
- there does not exist any $x \in \{0,1\}^\lambda$ such that $g(x) = z_i$

We then define the following event

- $\mathsf{Bad}_{Y,\mathsf{r}}$: there are more than $k = \delta n$ number of "no-preimage" z_i's in Y.

In the following, we present a claim. We will first show how to prove Lemma 5.5.2 assuming that Claim 5.5.1 holds, and then present its proof.

Claim 5.5.1. *For every non-trivial S^* with the $z \in \{0,1\}^*$ satisfying Inequality (5.37), there exists a negligible function $\mathsf{negl}(\cdot)$ such that*

$$\Pr\left[\mathsf{r} \xleftarrow{\$} \{0,1\}^{t\cdot\log(n)}; (\cdot, Y) \leftarrow \langle S^*(1^\lambda, z), R(1^\lambda, \mathsf{r})\rangle : \mathsf{Bad}_{Y,\mathsf{r}} \;\middle|\; Y \neq \bot\right] \leq \mathsf{negl}(\lambda). \tag{5.38}$$

Consider the Y output by R from $\langle S^*(1^\lambda, z), R(1^\lambda, \mathsf{r})\rangle$, where $S^*(1^\lambda, z)$ is non-trivial and $\mathsf{r} \xleftarrow{\$} \{0,1\}^{t\log(n)}$. If $Y \neq \bot$, it must be of the form $(z_1, \ldots, z_n)$, where $|z_i| = \lambda$ for $i \in \{b_1, \ldots, b_t\}$ and $|z_i| = m$ for $i \in [n] \setminus \{b_1, \ldots, b_t\}$. Recall that $\{b_1, \ldots, b_t\}$ is the size-t subset of $[n]$ specified by r.

Assuming that Claim 5.5.1 holds, to finish the proof of Lemma 5.5.2, it suffices to show the following claim:

- Conditioned on $Y \neq \bot$, if $\mathsf{Bad}_{Y,\mathsf{r}}$ does not happen, then there exist an x and a r such that $G^g(\mathsf{x}\|\mathsf{r}) = Y$.

Conditioned on $Y \neq \bot$, Claim 5.5.1 implies that there are at most $k = \delta n$ no-preimage z_i's (except with negligible probability). In the following, we assume w.l.o.g. that there are exactly k no-preimage z_i's. We denote them as $\{z_{p_1}, \ldots, z_{p_k}\}$ (i.e. we denote the indices of these no-preimage z_i's by $\{p_1, \ldots, p_k\}$). Then, for each z_i where $i \in [n] \setminus \{p_1, \ldots, p_k\}$, this z_i must have (at least) one preimage under $g(\cdot)$. We denote an arbitrary preimage of such z_i as $g^{-1}(z_i)$. By definition, a no-preimage z_i is *not* in the set $\{z_{b_1}, \ldots, z_{b_t}\}$. Therefore, we must have that $\{p_1, \ldots, p_k\} \cap \{b_1, \ldots, b_t\} = \emptyset$.

With these observations, we show in the following how to construct x and r such that $G^g(\mathsf{x}\|\mathsf{r}) = (z_1, \ldots, z_n)$. At a high-level, we take advantage of Case 2. At a high-level, we will put those no-preimage z_i's together with their indices as the (p_i, y'_{p_i}) part of x. We will use the above $\mathsf{r} = \{b_1, \ldots, b_t\}$ from R. Since $\{b_1, \ldots, b_t\} \cap \{p_1, \ldots, p_k\} = \emptyset$, G^g will put the no-preimage z_{p_i}'s at position p_i's (are required by Case 2). This will give us the desired Y. Concretely, we set:

$$\mathsf{x} := (x'_1, \ldots, x'_n)\|(p_1, z_{p_1}), \ldots, (p_k, z_{p_k}), \quad \mathsf{r} := (b_1, \ldots, , b_t)$$

where x'_i's are defined as follows: $\forall i \in [n], \quad x'_i := \begin{cases} z_i & i \in \{b_1, \ldots, b_t\} \\ 0^\lambda & i \in \{p_1, \ldots, p_k\} \\ g^{-1}(z_i) & \text{otherwise} \end{cases}.$

We remark that $g^{-1}(z_i)$ may not be efficiently computable (indeed, g is a PRG). But this is fine since we only require the existence of $g^{-1}(z_i)$. Also, note that $\{p_1. \ldots, p_k\} \cap \{b_1, \ldots, b_t\} = \emptyset$. It then follows from the description in Construction 5.4.1 (in particular, Case 2) that $G^g(\mathsf{x} \| \mathsf{r}) = Y$.

In the following, we show the proof for Claim 5.5.1, which will complete the proof for Lemma 5.5.2.

Proof of Claim 5.5.1. All the probabilities appearing in this proof are taken over the following random procedure:

$$\mathsf{r} \xleftarrow{\$} \{0,1\}^{t \cdot \log(n)}; (\cdot, Y) \leftarrow \langle S^*(1^\lambda, z), R(1^\lambda, \mathsf{r}) \rangle,$$

where S^* and z are as described in Claim 5.5.1.

First, note that $\Pr[\mathsf{Bad}_{Y,\mathsf{r}} \mid (Y \neq \bot)] \cdot \Pr[Y \neq \bot] = \Pr[\mathsf{Bad}_{Y,\mathsf{r}} \wedge (Y \neq \bot)]$. Since S^* is non-trivial, we know from Inequality (5.37) that $\Pr[Y \neq \bot]$ is non-negligible. Therefore, to prove Inequality (5.38), it suffices to show

$$\Pr[\mathsf{Bad}_{Y,\mathsf{r}} \wedge (Y \neq \bot)] \leq \mathsf{negl}(\lambda), \tag{5.39}$$

which we prove in the following.

Consider the execution $(\cdot, Y) \leftarrow \langle S^*(1^\lambda, z), R(1^\lambda, \mathsf{r}) \rangle$ where $Y \neq \bot$. Observe that S^* at Stage 2 commits a value α[9] of the form shown in Expression (5.35). For this execution, we define the following sequence of events:

- E_1: $\{p_1, \ldots, p_k\} \cap \{b_1, \ldots, b_t\} = \emptyset$. The $\{p_i\}_{i \in [k]}$ are those contained in α and $\{b_i\}_{i \in [t]}$ are those specified by R's input r.
- E_2: $\forall i \in [n] \setminus \{b_1, \ldots, b_t\}$, $y_i = z_i$. The y_i's are contained in the second row of M (which is in turn contained in α), and z_i's are contained in Y.
- E_3: the M_r sent by S^* at Stage 4 does consist of the columns of M specified by $\mathsf{r} = \{b_1, \ldots, b_t\}$.
- E_4: there are more than $k = \delta n$ number of y_i's (contained in the second row of M) that satisfy the following requirement: there does not exist any $x \in \{0,1\}^\lambda$ such that $g(x) = y_i$.

We first claim:

$$\Pr[E_4 \wedge (Y \neq \bot)] \leq \mathsf{negl}(\lambda) \tag{5.40}$$

To see that, first notice the folloing:

$$\begin{aligned} \Pr[E_4 \wedge (Y \neq \bot)] &= \Pr[E_4 \wedge (Y \neq \bot) \wedge E_3] + \Pr[E_4 \wedge (Y \neq \bot) \wedge \overline{E}_3] \\ &\leq \underbrace{\Pr[(Y \neq \bot) \mid E_3 \wedge E_4]}_{\mathsf{P}_1} + \underbrace{\Pr[(Y \neq \bot) \wedge \overline{E}_3]}_{\mathsf{P}_2} \end{aligned}$$

[9]This α is well-defined as BBCom is statistically-binding.

We now show that both P_1 and P_2 are negligible. First, recall that R checks at Stage 5 if $y_i = g(x_i)$ for all columns $[x_i\ y_i]^T$ contained in M_{r}. Thus, conditioned on E_3, the receiver does not abort only if the set $\mathsf{r} = \{b_1, \ldots, b_t\}$ *does not* select any "bad" column $[x_i\ y_i]^T$ in M s.t. $y_i \neq f(x_i)$. Moreover, E_4 ensures that there are more than $k = \delta n$ such "bad" columns in M. Therefore, the probability P_1 is smaller than $(1-\delta)^t$, which is negligible as $0 < \delta < 1$ is a constant and t is $\omega(\log \lambda)$.

Also, observe that E_3 was actually proved at Stage 6 (as Item 6b) by S^* via the commit-and-prove protocol Π_{ZKCnP}. The soundness of Π_{ZKCnP} implies that P_2 is negligible.

Next, we make another claim:

$$\Pr[\mathsf{Bad}_{Y,\mathsf{r}} \wedge (Y \neq \bot) \wedge \overline{E}_2] \leq \mathsf{negl}(\lambda) \tag{5.41}$$

To prove this inequality, notice the following:

$$\begin{aligned}\Pr[\mathsf{Bad}_{Y,\mathsf{r}} \wedge (Y \neq \bot) \wedge \overline{E}_2] &= \Pr[\mathsf{Bad}_{Y,\mathsf{r}} \wedge (Y \neq \bot) \wedge \overline{E}_2 \wedge E_1] + \Pr[\mathsf{Bad}_{Y,\mathsf{r}} \wedge (Y \neq \bot) \wedge \overline{E}_2 \wedge \overline{E}_1] \\ &\leq \underbrace{\Pr[E_1]}_{\mathsf{P}_3} + \underbrace{\Pr[(Y \neq \bot) \wedge \overline{E}_2 \mid \overline{E}_1]}_{\mathsf{P}_4}\end{aligned}$$

We now show that both P_3 and P_4 are negligible. Recall that E_1 is the event that the set $\{p_1, \ldots, p_k\}$ contained in α does not overlap with $\mathsf{r} = \{b_1, \ldots, b_t\}$. First, note that the Π_{ZKCnP} proof at Stage 6 ensures that, except with negligible probability, the set $\{p_1, \ldots, p_k\}$ is a size-k subset of $[n]$ (i.e. Item 6a). Also, observe that the r is a size-t random subset of $[n]$ that is sampled *independently* of $\{p_1, \ldots, p_k\}$. Therefore, E_1 happens with probability $\leq (1-\delta)^t + \mathsf{negl}(\lambda)$, which is negligible.

According to Construction 5.4.1, if $\{p_1, \ldots, p_k\} \cap \{b_1, \ldots, b_t\} \neq \emptyset$ (which is exactly the event $\overline{E}_1$), then it must hold that $y_i = z_i$ for all $i \in [n] \setminus \{b_1, \ldots, b_t\}$ (which is exactly E_2). Also, recall that this condition is enforced by the $\mathsf{BBProve}$ performed by S^* at Stage 6 (see Item 6c). The soundness of the Π_{ZKCnP} guarantees that, conditioned on $\overline{E}_1$, if E_2 does not hold, then R will abort with overwhelming probability. Therefore, P_4 is negligible.

Before proving Inequality (5.39), we need one more claim:

$$\Pr[\mathsf{Bad}_{Y,\mathsf{r}} \mid E_2] \leq \Pr[E_4 \mid E_2] \tag{5.42}$$

To prove the above inequality, recall that if $\mathsf{Bad}_{Y,\mathsf{r}}$ happens, then there are more than $k = \delta n$ many z_i's in $\{z_1, \ldots, z_n\} \setminus \{z_{b_1}, \ldots, z_{b_t}\}$ that do not have any preimage under $g(\cdot)$. Moreover, conditioned on E_2, we known that $z_i = y_i$ for all $i \in [n] \setminus \{b_1, \ldots, b_t\}$. In this case, $\mathsf{Bad}_{Y,\mathsf{r}}$ implies that there are more than δn many y_i's, among all the y_i's whose index lies in $[n] \setminus \{b_1, \ldots, b_t\}$, that do not have any preimage under $g(\cdot)$. Thus, there are definitely more than $k = \delta n$ many y_i's (among all the y_i's contained in the second row of M) that do not have any preimage, which is exactly the event E_4. Therefore, conditioned on E_2, $\mathsf{Bad}_{Y,\mathsf{r}}$ implies E_4. This gives us Inequality (5.42).

We are now ready to derive Inequality (5.39):

$$\begin{aligned}
\Pr[\mathsf{Bad}_Y \wedge (Y \neq \bot)] &= \Pr[\mathsf{Bad}_Y \wedge (Y \neq \bot) \wedge E_2] + \Pr[\mathsf{Bad}_Y \wedge (Y \neq \bot) \wedge \overline{E}_2] \\
&\leq \Pr[E_4 \wedge (Y \neq \bot) \wedge E_2] + \Pr[\mathsf{Bad}_Y \wedge (Y \neq \bot) \wedge \overline{E}_2] && (5.43) \\
&\leq \Pr[E_4 \wedge (Y \neq \bot)] + \Pr[\mathsf{Bad}_Y \wedge (Y \neq \bot) \wedge \overline{E}_2] \\
&\leq \mathsf{negl}(\lambda) && (5.44)
\end{aligned}$$

where Inequality (5.43) follows from Inequality (5.42). Also, note that Inequality (5.44) follows from Inequalities (5.40) and (5.41). This finishes the proof of Claim 5.5.1.

This finishes the proof for Lemma 5.5.2. □

Lemma 5.5.3 (Zero-Knowledge)**.** *Protocol Π^g_G in Protocol 5.5.1 satisfies the zero-knowledge property defined in Definition 5.5.1.*

Proof. To prove the zero-knowledge property, we need to show a PPT ideal-world simulator Sim for any PPT malicious receiver R^*. At a high-level, such a simulator can be constructed as follows. Sim will use the simulator of the commit-and-prove protocol Π_{ZKCnP} at Stages 2 and 6. This allows Sim to finish the interaction without knowing the sender's input x.

Formally, we will show two hybrids H_0 and H_1, where the H_0 is the real execution between the honest sender and R^*, and show that H_1 is essentially the simulator we want. We use Out_{H_i} to denote the output of hybrid H_i.

Hybrid $H_0(1^\lambda, \mathsf{x}, z)$**.** This hybrid uses the strategy of the honest $S(1^\lambda, \mathsf{x})$ to interact with the corrupted receiver $R^*(1^\lambda, z)$. At the end of the execution, H_0 outputs whatever R^* outputs. This hybrid is exactly the real execution.

Hybrid $H_1(1^\lambda, \mathsf{x}, z)$**.** This hybrid is identical to the previous one, except that

- At Stage 2, instead of executing BBCom, H_1 uses the strategy of Sim_1 in its communication with R^*, where Sim_1 is the simulator for the **Commit** stage of Π_{ZKCnP} (see Definition 5.2.2).
- At Stage 6, instead of doing the proof honestly, H_1 uses the strategy of Sim_2 in its communication with R^*, where Sim_1 is the simulator for the **Prove** stage of Π_{ZKCnP} (see Definition 5.2.2).

$\mathsf{Out}_{H_0} \overset{\text{c}}{\approx} \mathsf{Out}_{H_1}$: This is due to the ZK property of Π_{ZKCnP}.

The Simulator $\mathsf{Sim}(1^\lambda, z)$**.** We now describe the simulator Sim for the ideal execution. Sim is identical to H_1 except for the following changes:

- Sim does not need to execute Stage 1;
- Upon receiving $\mathsf{r} = (b_1, \ldots, b_t)$ at Stage 3, Sim sends it to the idea functionality $\mathcal{F}_{G^g}$, and receives back $Y = (z_1, \ldots, z_n)$.
- At Stage 4, Sim (with its oracle access to $g(\cdot)$) sets $\widetilde{M}_{\mathsf{r}} = \begin{bmatrix} z_{b_1} & \cdots & z_{b_t} \\ g(z_{b_1}) & \cdots & g(z_{b_t}) \end{bmatrix}$, and sends $\widetilde{M}_{\mathsf{r}}$ and $(z_1, \ldots, z_n)$ to R^*.

We remark that, unlike H_1, Sim does not need to know the input x to the sender; the values $(z_1, \ldots, z_n)$ it obtains from $\mathcal{F}_{G^g}$ contain all the information to finish its execution with R^*. In particular, although the z_i's now come from the ideal functionality, they are identically distributed to the ones generated by the honest sender. Thus, the $\widetilde{M}_{\mathsf{r}}$ is also identically distributed to the M_{r} in H_1. It Then follows that the output of Sim is identical to H_1.

This finishes the proof for Lemma 5.5.3. □

5.6 Proof-Based Collision-Resistant Hash Families

We now discuss proof-based collision-resistant hash families (PB-CRHFs). As mentioned in Section 5.1.3, the definition and construction of PB-CRHF follow the same template as our PB-OWFs, except that we need to handle the **Editing** condition differently.

5.6.1 Definition

Our PB-CRHF consists of an oracle machine $H^{(\cdot)}$ and an oracle protocol $\Pi_H^{(\cdot)}$. As mentioned in Section 5.1.3, the $H^{(\cdot)}$ will be instantiated as a hash *family*. That is, given a collision-resistant hash family $\mathcal{H}'$, we first run its KGen' to sample a function $h_i \in \mathcal{H}'$, and then instantiate $H^{(\cdot)}$'s oracle as h_i. Therefore, H^{h_i} is also a hash family whose KGen simply runs KGen' for $\mathcal{H}'$ (and samples a random string z that we will describe later).

Once $H^{(\cdot)}$ and $\Pi_H^{(\cdot)}$ are instantiated with an $h_i \xleftarrow{\$} \mathsf{KGen}'(1^\lambda)$, we can start talking about security. Same as in Definition 5.4.1, H^{h_i} takes two inputs x and r. We require that, for all r, $H^{h_i}(\cdot, \mathsf{r})$ is collision-resistant on its first input. The protocol $\Pi_H^{h_i}$ satisfies similar completeness, soundness, and ZK requirements as those in Definition 5.4.1. We provide the formal definition in Definition 5.6.1.

Definition 5.6.1 (Proof-Based CRHF)**.** *Let $a(\lambda)$, $b(\lambda)$ and $c(\lambda)$ be polynomials on λ such that $a(\lambda) + b(\lambda) < c(\lambda)$. A proof-based collision-resistant hash family (PB-CRHF) is a function family $\mathcal{H} = \{H_i\}_{i \in I}$ for some index set I, where $H_i : \{0,1\}^{a(\lambda)} \times \{0,1\}^{b(\lambda)} \to \{0,1\}^{c(\lambda)}$. For each $H_i \in \mathcal{H}$, there exists a protocol $\Pi_i = (S, R)_i$ consisting of a pair of* PPT *machines. $\mathcal{H}$ is a collision-resistant hash family in the following sense.:*

- **Collision-Resistance.** *$\mathcal{H}$ has PPT algorithms KGen and Eval, which satisfy the same requirements as in Definition 5.2.1; it further satisfies the following collision-resistant requirement: for any non-uniform PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\forall \mathsf{r} \in \{0,1\}^{b(\lambda)}$,*

$$\Pr[i \xleftarrow{\$} \mathsf{KGen}(1^\lambda), (\mathsf{x}, \mathsf{x}') \leftarrow \mathcal{A}(1^\lambda, i) : \mathsf{x} \neq \mathsf{x}' \wedge H_i(\mathsf{x} \| \mathsf{r}) = H_i(\mathsf{x}' \| \mathsf{r})] \leq \mathsf{negl}(\lambda).$$

Let $(X, Y) \leftarrow \langle S(1^\lambda, \mathsf{x}), R(1^\lambda, \mathsf{r})\rangle_i$ denote the execution of the protocol Π_i, where the security parameter is λ, the input to S and R are x and r respectively, and the output of S and R are X and Y respectively. Let $Y = \bot$ denote the event that R aborts in the execution. The following properties for protocol $\Pi_i = (S, R)_i$ hold with overwhelming probability over the choice of $i \xleftarrow{\$} \mathsf{KGen}(1^\lambda)$ (see also Remark 5.6.1):

Figure 5.6.1: Functionality $\mathcal{F}_{H_i}$ for Proof-Based CRHFs

The ideal functionality $\mathcal{F}_{H_i}$ interacts with a sender S and a receiver R. Upon receiving the input $\mathsf{x} \in \{0,1\}^{a(\lambda)}$ from S and $\mathsf{r} \in \{0,1\}^{b(\lambda)}$ from R, the functionality $\mathcal{F}_{H_i}$ sends $\mathsf{x}\|\mathsf{r}$ to S, and $H_i(\mathsf{x}\|\mathsf{r})$ to R.

- **Completeness.** $\forall \lambda \in \mathbb{N}$, $\forall i \in \{0,1\}^{\lambda}$, $\forall \mathsf{x} \in \{0,1\}^{a(\lambda)}$ *and* $\forall \mathsf{r} \in \{0,1\}^{b(\lambda)}$,

$$\Pr\left[(X,Y) \leftarrow \langle S(1^\lambda, \mathsf{x}), R(1^\lambda, \mathsf{r})\rangle_i : X = \mathsf{x}\|\mathsf{r} \textbf{ and } Y = H_i(\mathsf{x}\|\mathsf{r})\right] = 1,$$

  where the probability is taken over the the randomness used by S and R.

- **Soundness.** *For every* PPT *machine S^* and every auxiliary input* $\mathsf{aux} \in \{0,1\}^*$, *there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that*

$$\Pr\left[\begin{array}{l} \mathsf{r} \xleftarrow{\$} \{0,1\}^{b(\lambda)}; \\ (\cdot, Y) \leftarrow \langle S^*(1^\lambda, \mathsf{aux}), R(1^\lambda, \mathsf{r})\rangle_i \end{array} : \begin{array}{l} Y \neq \bot \textbf{ and} \\ \nexists \mathsf{x} \text{ s.t. } H_i(\mathsf{x}\|\mathsf{r}) = Y \end{array}\right] \leq \mathsf{negl}(\lambda)$$

  where the probability is taken over the random sampling of i and r,*and the randomness used by S^* and R.*

- **Zero-Knowledge.** *This property is defined as* only *requiring security against corrupted R in the ideal-real paradigm for 2PC w.r.t. the ideal functionality $\mathcal{F}_{H_i}$ in Figure 5.6.1. Namely, there exist a PPT simulator* Sim *such that for any PPT adversary $\mathcal{A}$,* $\forall \mathsf{aux} \in \{0,1\}^*$, $\forall \mathsf{x} \in \{0,1\}^{a(\lambda)}$ *and* $\forall \mathsf{r} \in \{0,1\}^{b(\lambda)}$,

$$\mathsf{REAL}_{\Pi_i, \mathcal{A}(\mathsf{aux})}(\mathsf{x}, \mathsf{r}) \overset{\text{c}}{\approx} \mathsf{IDEAL}_{\mathcal{F}_{H_i}, \mathsf{Sim}(\mathsf{aux})}(\mathsf{x}, \mathsf{r}),$$

  where $\mathsf{REAL}_{\Pi_i, \mathcal{A}(\mathsf{aux})}(\mathsf{x}, \mathsf{r})$ *and* $\mathsf{IDEAL}_{\mathcal{F}_{H_i}, \mathsf{Sim}(\mathsf{aux})}(\mathsf{x}, \mathsf{r})$ *are defined in the same way as in Definition 5.4.1.*

If both the constructions of $\mathcal{H}$ and Π only involve black-box access to other primitives, we obtain black-box *PB-CRHF.*

Remark 5.6.1. *Similar to the case of PB-OWFs, the protocol here is meant to be a ZK system for the range-membership of the function. But note that $\mathcal{H}$ in the above PB-CRHF is a* family *of functions, and the concrete function H_i is sampled by running the key generation algorithm. So, the range-membership is well-defined only when H_i is sampled and fixed. Therefore, the security properties of protocol Π_i will depend on the sampling of H_i by running* KGen. *This idea already appears explicitly in the definition of collision-resistant hash* families*—the adversary's winning probability in the collision-resistant game depends on the random procedure of executing* KGen.

5.6.2 Merkle Tree Related Notation

As mentioned in the technical overview (Section 5.1.3), our construction makes use of the Merkle hashing tree [Mer90]. In this part, we setup related notation.

Notation for Perfect Binary Trees. A *perfect* binary tree is a binary tree in which all interior nodes have two children and all leaves have the same depth. A perfect binary tree of hight ℓ has $2^{\ell+1}-1$ nodes, where 2^ℓ of the nodes are leaves. There exist canonical methods (e.g., array representation) to index the nodes in the tree, which forms a bijection between the nodes and the set $[2^{\ell+1}-1]$. We will use the indices to refer to the corresponding nodes under this bijection. In particular, we require that the first 2^ℓ indices (i.e. $[2^\ell]$) represent the leaves.

For each node $k \in [2^{\ell+1}-1]$, v_k denotes the content/value of this node. For a leaf node $i \in 2^\ell$, the *sibling path* of i consists of the value v_i together with the values of all the siblings of nodes on the path from i to the root. We use P_i to denote the sibling path of leaf i, and $\mathsf{Ind}(\mathsf{P}_i)$ denotes the collection of indices of the nodes on path P_i. For k sibling paths $(\mathsf{P}_{i_1}, \ldots, \mathsf{P}_{i_k})$, we use $\mathsf{Ind}(i_1, \ldots, i_k)$ to denote the collection of indices of the nodes on all these paths, i.e.,

$$\mathsf{Ind}(i_1, \ldots, i_k) \coloneqq \mathsf{Ind}(\mathsf{P}_{i_1}) \cup \ldots \cup \mathsf{Ind}(\mathsf{P}_{i_1}). \tag{5.45}$$

We remark that $\mathsf{Ind}(i_1, \ldots, i_k)$ is a set of indices for nodes. It does not depend on the contents/values of nodes. Once the structure of the tree is fixed, this set is then fixed, even if the tree contains only dummy nodes.

Merkle Trees. A Merkle hashing tree [Mer90] is a special type of perfect binary tree constructed as follows. Let $n = 2^l$ for some integer l. Given a hash function $h : \{0,1\}^{2m} \to \{0,1\}^m$ and a length-mn string X, the Merkle tree $MT_{h,m}(X)$ is a size-$(2n-1)$ perfect binary tree of the following form:

– X is parsed as n blocks $(x_1, \ldots, x_n)$, each x_i is of length m. These x_i's are the contents of the n leaves;

– **(Merkle Consistency.)** For any non-leaf node k and its left child ℓ and right child r, their contents satisfy the equation $h(v_k) = h(v_\ell \| v_r)$.

A sibling path P_i $(i \in [n])$ is said to be *Merkle-consistent* if all the nodes on this path satisfy the above Merkle consistency condition.

5.6.3 Our Construction

The formal construction is provided in Construction 5.6.1 and Protocol 5.6.1. We follow the high-level idea described in Section 5.1.3 with the following modifications. Instead of hashing the $(x_1, \ldots, x_n)$ (contained in x) separately, we build a Merkle tree using them as the leaves. In Construction 5.6.1, P_i denotes the sibling path from leaf x_i to the root; $\mathsf{Ind}(b_1, \ldots, b_t)$ denotes the set of *indices* of the nodes on path $\mathsf{P}_{b_1}, \ldots, \mathsf{P}_{b_t}$. (See Section 5.6.2 for relevant notation.) In Protocol 5.6.1, the receiver checks t leaves and their corresponding sibling paths. This ensures that there are at least $(n-k)$ "good" leaves, in the sense that there are valid sibling paths from the Merkle root to them. In the **Editing** case, this will allow us to perform preimage editing by planting the v_{p_i} values on the k "bad" paths to obtain a (partial) tree consistent with the root t_1 contained in Y. Note that we also hash the Λ in Step 1c. As explained in Section 5.1.3, this is to prevent the adversary from taking advantage of preimage editing to find collisions.

Construction 5.6.1: Collision-Resistant Hash Family $H_z^{h_i}$

Let $m(\lambda)$ and $n(\lambda)$ be polynomials of λ. Assume w.l.o.g. that is n a power of 2 (i.e., $n = 2^\ell$ for some ℓ). Let $0 < \delta < 1$ be a constant, and $k(\lambda) = \delta n(\lambda)$. Let $t(\lambda) = \log^2(\lambda)$ (see also Remark 5.4.1). Let $\mathcal{H}' = \{h_i\}_{i \in I}$ be a collision-resistant hash family where $h_i : \{0,1\}^{2m(\lambda)} \to \{0,1\}^{m(\lambda)}$. Denote its key generation as KGen'.

– **Key Generation.** On input 1^λ, sample a function from $\mathcal{H}'$ by running $h_i \leftarrow \mathsf{KGen}'(1^\lambda)$; sample a random string $z \overset{\$}{\leftarrow} \{0,1\}^{m(\lambda)}$; outputs (i, z) as the hash key.

– **Evaluation.** On input $\mathsf{x} \in \{0,1\}^{nm+k(\log(2n)+m)+3m}$ and $\mathsf{r} \in \{0,1\}^{t\log(n)}$, the evaluation algorithm parses them as:

$$\mathsf{x} = (x_1, \ldots, x_n)\|(p_1, v_{p_1}), \ldots, (p_k, v_{p_k})\|\tau\|\mu, \quad \mathsf{r} = (b_1, \ldots, b_t), \tag{5.46}$$

where $|x_i| = |v_{p_i}| = m$, $\{p_i\}_{i\in[k]}$ is a size-k subset of $[2n-1]$, and $\{b_i\}_{i\in[t]}$ is a size-t subset of $[n]$. The set $\{b_i\}_{i\in[t]}$ specifies t leaves out of all the n leaves.

The algorithm builds a perfect binary tree T that has n leaves, where all the nodes are dummies. As discussed in Section 5.6.2, the indices of the nodes in T are well-defined, even though T now contains only dummy nodes. The evaluation procedure outputs $Y = \mathsf{t}_1\|\mathsf{t}_2\|(\mathsf{P}_{b_1}, \ldots, \mathsf{P}_{b_t})\|(b_1, \ldots, b_t)$, which is computed as follows:

1. **Non-Editing:** If $\tau = z$ **or** $h_i(\tau) \neq h_i(z)$ **or** $\mathsf{Ind}(b_1, \ldots, b_t) \cap \{p_1, \ldots, p_k\} \neq \emptyset$:
   (a) It fills the tree T as follows. It places $(x_1, \ldots, x_n)$ at the n leaves. For any other node in T, its content is the hash value under h_i on the concatenation of its left child and right child. Denote the root value as t_1.
   (b) For $i \in [t]$, P_{b_i} is the sibling path of leaf x_{b_i} in the above tree T;
   (c) Use h_i to hash[a] the following Λ value to a length-m string denoted as t_2:

$$\Lambda = (p_1, v_{p_1}), \ldots, (p_k, v_{p_k})\|\tau\|\mu.$$

2. **Editing:** if $\tau \neq z$ **and** $h_i(\tau) = h_i(z)$ **and** $\mathsf{Ind}(b_1, \ldots, b_t) \cap \{p_1, \ldots, p_k\} = \emptyset$:
   (a) It fills the tree T as follows. It places $(x_1, \ldots, x_n)$ at the n leaf positions in T. Then, fill the tree bottom up, following the rule for Merkle tree (i.e. the hashing of two children nodes' contents is the parent node's content), with the following exception: for node $p_i \in \{p_1, \ldots, p_k\}$, it fills node p_i with the v_{p_i} contained in x instead of the hash of the children of node p_i. Denote the root value as t_1.
   (b) For $i \in [t]$, P_{b_i} is defined as the sibling path of leaf x_{b_i} in the tree T;
   (c) Set $\mathsf{t}_2 = \mu$ (recall that μ is contained in x);

[a]Note that the input to h_i should have length $2m$. But $|\Lambda| > 2m$. This can be handled using domain-extension techniques, e.g., the Merkle-Damgård transformation [Mer90, Dam90].

It is also worth noting that Construction 5.6.1 and Protocol 5.6.1 work for an x of fixed length. But since we hash the $\{x_i\}_{i\in[n]}$ part using a Merkle tree, we can handle x with a

Protocol 5.6.1: Protocol $\Pi_z^{h_i}$ for Our PB-CHRF

Let $\mathcal{H}'$, m, n, δ, t and k be as in Construction 5.6.1. Let $\Pi_{\mathsf{ZKCnP}} = (\mathsf{BBCom}, \mathsf{BBProve})$ be a black-box commit-and-prove protocol. For a function defined by (i, z) from the PB-CRHF in Construction 5.6.1, this protocol proceeds as follows. Both parties take the security parameter 1^λ as the common input. Sender S takes a string $\mathsf{x} \in \{0,1\}^{nm+k(\log(n)+m)+3m}$ as private input; receiver R takes a string $\mathsf{r} \in \{0,1\}^{t\log(n)}$ as private input.

1. S parses x as $(x_1, \ldots, x_n) \| (p_1, v_{p_1}), \ldots, (p_k, v_{p_k}) \| \tau \| \mu$ (in the same manner as in Expression (5.46)). S build a Merkle tree $MT'_{h,m}(x)$ using $(x_1, \ldots, x_n)$ as the leaves (this is identical to Step 1a). Denote the root of this tree as t_x.
2. S and R execute $\mathsf{BBCom}(\nu)$, the **Commit** stage of Π_{ZKCnP}, where S commits to the following value
$$\nu := \mathsf{t}_\mathsf{x} \| (p_1, \ldots, p_k). \tag{5.47}$$
3. R sends the value r.
4. S parses r as $(b_1, \ldots, b_t)$ where each b_i is of length $\log(n)$. With the values x and r, S evaluates the function $H_z^{h_i}$ as per Construction 5.6.1 to compute the following Y, which it sends to R:
$$Y = \mathsf{t}_1 \| \mathsf{t}_2 \| (\mathsf{P}_{b_1}, \ldots, \mathsf{P}_{b_t}) \| (b_1, \ldots, b_t).$$
5. R checks if P_{b_i} is Merkle-consistent for all $i \in [t]$. R aborts if any of the check fails.
6. S and R execute $\mathsf{BBProve}$, the **Prove** stage of Π_{ZKCnP}, where S proves that the ν committed in Stage 2 satisfies the following conditions:
   (a) the $\{p_1, \ldots, p_k\}$ in ν form a size-k subset of $[2n-1]$, where $k = \delta n$; **and**
   (b) the t_x contained in τ is equal to t_1, **or** $\mathsf{Ind}(b_1, \ldots, b_t) \cap \{p_1, \ldots, p_k\} = \emptyset$.
7. **(Receiver's Output).** R outputs $Y = \mathsf{t}_1 \| \mathsf{t}_2 \| (\mathsf{P}_{b_1}, \ldots, \mathsf{P}_{b_t}) \| (b_1, \ldots, b_t)$.
8. **(Sender's Output).** S outputs $X = (x_1, \ldots, x_n) \| (p_1, v_{p_1}), \ldots, (p_k, v_{p_k}) \| \tau \| \mu \| (b_1, \ldots, b_t)$.

various-length $\{x_i\}_{i\in[n]}$ part (which dominates the length of x). To maintain security, we simply include the height of the Merkle tree in Y.

5.6.4 Proof of Security

In this section, we prove the following theorem.

Theorem 5.6.1. *The construction shown in Construction 5.6.1 and Protocol 5.6.1 is a PB-CRHF (as per Definition 5.6.1) that makes only black-box use of a CRHF $\mathcal{H}'$. Moreover, the PB-OWF is private-coin (resp. public-coin) if $\mathcal{H}'$ is private-coin (resp. public-coin).*

It follows immediately from the description that our construction makes only black-box use of $\mathcal{H}'$, and it also satisfies the complement requirement as per Definition 5.6.1. In the following, we first prove the collision resistance of Construction 5.6.1 in Lemma 5.6.1. Then, we establish the soundness and zero-knowledge property for Protocol 5.6.1 in Lemma 5.6.2 and Lemma 5.6.3, respectively.

Lemma 5.6.1 (Collision Resistance)**.** *Construction 5.6.1 satisfies the collision-resistant requirement as in Definition 5.6.1. Moreover, Construction 5.6.1 is private-coin (resp. public-coin) if $\mathcal{H}'$ is private-coin (resp. public-coin).*

Proof. We first show that the function is input-compressing. According to Construction 5.6.1. the input is of length:

$$|X| = |\mathsf{x}| + |\mathsf{r}| = nm + k(\log(2n) + m) + 3m + t\log(n),$$

and the output is the length

$$|Y| = 2m + 2\ell m + t\log(n),$$

where $\ell = \log(n)$ is the height of the Merkle tree T. Thus, we have $|X| > |Y|$.

Public-Coin v.s. Private-Coin. Let us recall the key generation algorithm in Construction 5.6.1. In addition to running the KGen' of $\mathcal{H}'$, it only sample a random string $z \in \{0,1\}^{m(\lambda)}$, which is a public-key operation. Therefore, this key generation procedure is private-coin (resp. public-coin) if $\mathcal{H}'$ is private-coin (resp. public-coin).

Collision Resistance. Assume for contradiction that there is a PPT adversary $\mathcal{A}$ that breaks the collision-resistance property of Construction 5.6.1.

Consider a function $H_{h_i,z}$ sampled from the family, where $h_i \leftarrow \mathsf{KGen}(1^\lambda)$ and $z \xleftarrow{\$} \{0,1\}^{2m}$. First, we claim that $\mathcal{A}$ cannot output an X that contains a τ satisfying $\tau \neq z$ and $h_i(\tau) = h_i(z)$; otherwise, $\mathcal{A}$ can be used to break the collision resistance of h. Therefore, we assume in the following that the purported collision pair X, X' output by $\mathcal{A}$ will not trigger the **Editing** condition.

Conditioned on the event that the **Editing** condition is not triggered, if $H_{h_i,z}(X) = H_{h_i,z}(X')$, X and X' must differ in the $(x_1, \ldots, x_n)$ part. This is because the output of $H_{h_i,z}$ contains the values $(b_1, \ldots, b_t)$ (thus, the collision will not happen here); it also contains a hash of the value $\Lambda = (p_1, v_{p_1}), \ldots, (p_k, v_{p_k}) \| \tau \| \mu$. If X and X' have different Λ's, $\mathcal{A}$ breaks the collision-resistant property of h_i.

The above argument implies that X and X' contain different $(x_1, \ldots, x_n)$ values, but $H^{h_i}(X)$ and $H^{h_i}(X')$ contain the same t_1, the root of the Merkle tree on $(x_1, \ldots, x_n)$. Therefore, $\mathcal{A}$ can be converted to a collision-finder that breaks the collision-resistant of h_i (in the Step 1a Merkle tree). ☐

Completeness follows immediately from the description of Construction 5.6.1 and Protocol 5.6.1. In the following, we prove soundness (Lemma 5.6.2) and zero-knowledge (Lemma 5.6.3).

Lemma 5.6.2 (Soundness)**.** *Protocol 5.6.1 satisfies the soundness requirement defined in Definition 5.6.1.*

Proof. First, let $\mathsf{Bad}_{\mathsf{t}_1}$ denote the following event: there does not exist an n-leaf full and complete binary tree rooted at t_1 (contained in Y) such that at least $(1-\delta)n$ leaves with their corresponding sibling paths satisfy the Merkle consistency requirement.

At a high-level, the proof for Lemma 5.6.2 follows the approach of that for Lemma 5.4.2. First, we assert in Claim 5.6.1 that it is impossible (except with negligible probability) that R accepts the execution *and* the event $\mathsf{Bad}_{\mathsf{t}_1}$ happens. Second, we show that whenever R accepts and $\mathsf{Bad}_{\mathsf{t}_1}$ does not happen, there must exist a valid preimage for the Y learned by R. These two claims together imply Lemma 5.6.2.

Claim 5.6.1. *For every* PPT *machine* S^* *and every auxiliary input* $\mathsf{aux} \in \{0,1\}^*$, *there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that*

$$\Pr\left[\mathsf{r} \xleftarrow{\$} \{0,1\}^{t\log(n)}; (\cdot, Y) \leftarrow \langle S^*(1^\lambda, \mathsf{aux}), R(1^\lambda, \mathsf{r})\rangle : (Y \neq \bot) \wedge \mathsf{Bad}_{\mathsf{t}_1}\right] \leq \mathsf{negl}(\lambda),$$

where the probability is taken over the random sampling of r,*and the randomness used by* S^* *and* R.

Proof. We claim that, except with negligible probability, the t_1 in Y that S^* sends in Stage 4 is equal to the t_x in ν committed in Stage 2. Due to the soundness of the $\mathsf{BBProve}$ performed in Stage 6, the following holds with overwhelming probability

$$\mathsf{t}_1 = \mathsf{t}_\mathsf{x}, \quad \textbf{or} \quad \mathsf{Ind}(b_1, \ldots, b_t) \cap \{p_1, \ldots, p_k\} = \emptyset.$$

Moreover, since the values $\{b_i\}_{i=1}^t$ form a *random* size-t subset of $[n]$, the event $\mathsf{Ind}(b_1, \ldots, b_t) \cap \{p_1, \ldots, p_k\} = \emptyset$ will only happen with probability $\leq (1-\delta)^t$ (recall that $k = \delta n$), which is negligible as $0 < \delta < 1$ is a constant and $t = \omega(\log \lambda)$. This implies that $\mathsf{t}_1 = \mathsf{t}_\mathsf{x}$ must happen with overwhelming probability. Therefore, the following holds:

$$\begin{aligned}
&\Pr\left[(Y \neq \bot) \wedge \mathsf{Bad}_{\mathsf{t}_1}\right] \\
&= \Pr\left[(Y \neq \bot) \wedge \mathsf{Bad}_{\mathsf{t}_1} \wedge (\mathsf{t}_1 = \mathsf{t}_\mathsf{x})\right] + \Pr\left[(Y \neq \bot) \wedge \mathsf{Bad}_{\mathsf{t}_1} \wedge (\mathsf{t}_1 \neq \mathsf{t}_\mathsf{x})\right] \\
&= \Pr[\mathsf{Bad}_{\mathsf{t}_1} \wedge (\mathsf{t}_1 = \mathsf{t}_\mathsf{x})] \cdot \Pr\left[(Y \neq \bot) \;\middle|\; \mathsf{Bad}_{\mathsf{t}_1} \wedge (\mathsf{t}_1 = \mathsf{t}_\mathsf{x})\right] + \mathsf{negl}(\lambda),
\end{aligned}$$

where the probability is taken over the same random procedure as in Claim 5.6.1. To prove Claim 5.6.1, it now suffices to show the following

$$\Pr\left[(Y \neq \bot) \;\middle|\; \mathsf{Bad}_{\mathsf{t}_1} \wedge (\mathsf{t}_1 = \mathsf{t}_\mathsf{x})\right] \leq \mathsf{negl}(\lambda).$$

Note that $Y \neq \bot$ represents the event that R does not abort *until the end of the protocol*. Therefore, we only need to prove that

$$\Pr\left[R \text{ does not abort until Stage 5 (inclusively)} \;\middle|\; \mathsf{Bad}_{\mathsf{t}_1} \wedge \mathsf{t}_1 = \mathsf{t}_\mathsf{x}\right] \leq \mathsf{negl}(\lambda). \tag{5.48}$$

Assuming that $\mathsf{Bad}_{\mathsf{t}_1}$ happens and that $\mathsf{t}_1 = \mathsf{t}_\mathsf{x}$, R does not abort in Stage 5 only if its challenge $\mathsf{r} = (b_1, \ldots, b_t)$ (in Stage 3) hits the leaves that have sibling paths consistent with the root t_1 ($= \mathsf{t}_\mathsf{x}$). Since t_x is fixed before r (a random subset of $[n]$), this event happens with probability $< (1-\delta)^t$, which is negligible as $0 < \delta < 1$ is a constant and $t = \omega(\lambda)$. Therefore, Inequality (5.48) holds.

This finishes the proof for Claim 5.6.1. □

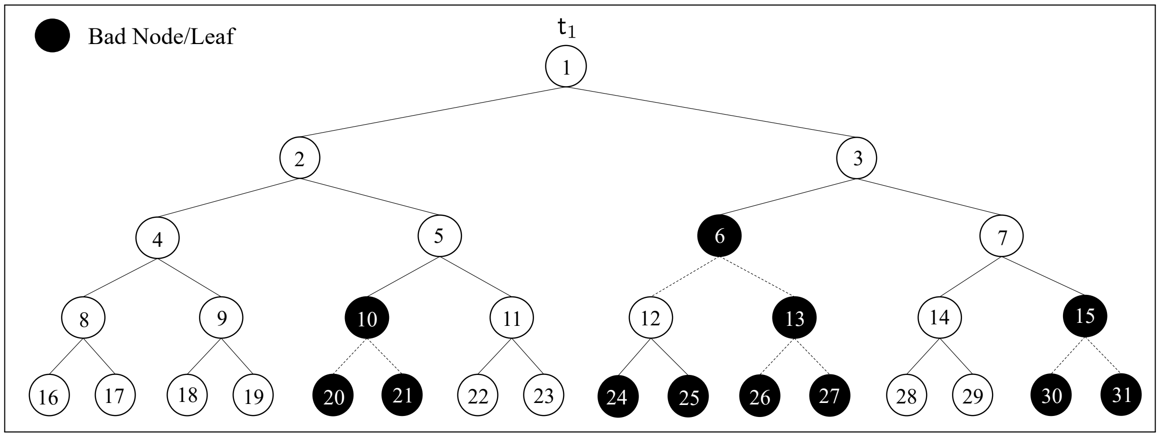


Figure 5.1: A tree with 16 leaves, 8 Bad leaves, and 4 Bad (non-leaf) nodes. Solid lines indicate consistent hash relation and dashed lines indicate inconsistent hash relation.

In the following, we show the existence of a preimage for Y given that $Y \neq \bot$ and that $\mathsf{Bad}_{\mathsf{t}_1}$ does not happen. Note that Y can be parsed as

$$Y = \mathsf{t}_1 \| \mathsf{t}_2 \| (\mathsf{P}_{b_1}, \ldots, \mathsf{P}_{b_t}) \| (b_1, \ldots, b_t),$$

where $(\mathsf{P}_{b_1}, \ldots, \mathsf{P}_{b_t})$ are consistent with t_1. Since $\mathsf{Bad}_{\mathsf{t}_1}$ does not happen, there is a perfect binary tree consistent with the root t_1 that has at most δn "Bad" leaves. On the path from the root t_1 to each Bad leaf, there exists a node of minimal depth (i.e. closest to the root) for which the hash consistency breaks (i.e. this node is not equal to the hash value of the concatenation of its two children). We call such nodes Bad nodes. Note that the number of Bad nodes cannot exceed δn. (See Figure 5.1 for an example.) W.l.o.g., we assume that there are exact $k = \delta n$ Bad nodes, and denote their indices in the tree as $\{p_1, \ldots, p_k\}$. We refer to the content of node p_i as v_{p_i} for all $i \in [k]$.

Now we describe how to find a preimage $X = \mathsf{x} \| \mathsf{r}$ for Y. We set r as the $(b_1, \ldots, b_t)$ part in Y. The x_i's in x are defined in the following way: if the i-th leaf in the above binary tree is not Bad, set x_i to be the contents of this leaf; otherwise, set x_i to be a dummy string (e.g. 0^λ). The (p_i, v_{p_i})'s consist of the indices and corresponding contents Bad nodes. We set τ such that it triggers the **Editing** condition in Construction 5.6.1 (i.e. $\tau \neq z$ and $h(\tau) = h(z)$)[10]. And μ is set to the value t_2 in Y.

To see why the above $\mathsf{x} \| \mathsf{r}$ is a valid preimage for Y under $H_z^{h_i}$, just follow the evaluation procedure in Construction 5.6.1. The **Editing** condition is triggered due to the way we set τ and the fact that the set of Bad indices $\{p_1, \ldots, p_k\}$ does not overlap with $\mathsf{Ind}(b_1, \ldots, b_t)$ (i.e. the indices of nodes on the sibling paths of leaves $\{x_{b_1}, \ldots, x_{b_t}\}$). Therefore, when building the Merkle tree, we will place v_{p_i} once we reach node p_i. Since all the non-Bad nodes in this Merkle tree are identical to the above binary tree, they necessarily share the same root value t_1. Finally, t_2 value will also be put in the correct position in Y as $\mathsf{t}_2 = \mu$

[10]Note that such a τ exists with overwhelming probability. Because h_i maps length-$2m$ strings to length-m ones, it cannot be injective on more than 2^m strings in its domain. Since we pick $z \xleftarrow{\$} \{0,1\}^{2m}$, with probability at least $(1 - \frac{1}{2^m})$, there exists a $\tau \neq z$ s.t. $h(z) = h(\tau)$.

in the **Editing** case. □

Lemma 5.6.3 (Zero-Knowledge)**.** *Protocol 5.6.1 satisfies the zero-knowledge requirement defined in Definition 5.6.1.*

Proof (Sketch). This lemma follows from an argument similar to that for Lemma 5.4.3. The ideal-world simulator Sim use S_1 (the **Commit**-stage simulator for Π_{ZKCnP}) and S_2 (the **Prove**-stage simulator for Π_{ZKCnP}) to go through Stage 2 and Stage 6 respectively. Note that Sim receives R's challenge r in Stage 3. It sends this r to the ideal functionality to learn Y. Similar as in the proof of Lemma 5.4.3, this Y value contains all the information to finish the remainder of the simulated interaction with R. □

5.7 Proof-Based One-Way Functions Supporting Predicates

Following the approach discussed in Section 5.1.4, we now describe how to extend our PB-OWF construction from Section 5.4 to support a predicate.

5.7.1 Definition

Definition 5.7.1 (PB-OWFs Supporting Predicates)**.** *Let $a(\lambda)$, $b(\lambda)$ and $c(\lambda)$ be polynomials on λ. Let ϕ be an efficiently computable predicate. A* proof-based one-way function supporting ϕ*, denoted as (F, Π_ϕ), consists of a function $F_\lambda : \{0,1\}^{a(\lambda)} \times \{0,1\}^{b(\lambda)} \to \{0,1\}^{c(\lambda)}$ and a protocol $\Pi_\phi = \langle S, R\rangle_\phi$ of a pair of* PPT *machines. The function F satisfies the following one-wayness requirement:*

- **One-Wayness.** *F_λ satisfies the same one-way requirement as in Definition 5.4.1. That is, for all $\mathsf{r} \in \{0,1\}^{b(\lambda)}$, $F_\lambda(\cdot, \mathsf{r})$ is one-way.*

In the protocol Π, both parties take the security parameter (which we omit henceforth for simplicity) and a predicate ϕ as public input. The private input for S and R are x and r, respectively. At the end of the protocol, S outputs X, and R outputs $Y\|u$. The protocol satisfies the following requirements:

- **Completeness.** *The protocol Π computes the ideal functionality $\mathcal{F}_F$ defined in Figure 5.7.1. Namely, $\forall \mathsf{x} \in \{0,1\}^{a(\lambda)}$ and $\forall \mathsf{r} \in \{0,1\}^{b(\lambda)}$, if $(X, Y\|u) \leftarrow \langle S(\mathsf{x}), R(\mathsf{r})\rangle_\phi$, then $X = \mathsf{x}\|\mathsf{r}$, $Y = F_\lambda(X)$ and $u = \phi(\alpha)$.*
- **Soundness.** *For every* PPT *machine S^* and every auxiliary input $z \in \{0,1\}^*$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that*

$$\Pr\left[\begin{array}{l} \mathsf{r} \xleftarrow{\$} \{0,1\}^{b(\lambda)}; \\ (\cdot, Y\|u) \leftarrow \langle S^*(z), R(\mathsf{r})\rangle_\phi \end{array} : \begin{array}{l} Y \neq \bot \textbf{ and} \\ \nexists \mathsf{x} \text{ s.t. } (F_\lambda(\mathsf{x}\|\mathsf{r}) = Y \wedge \phi(\alpha) = u) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

where the probability is taken over the random sampling of r, and the randomness used by S^* and R.

- **Zero-Knowledge.** *This property is defined in the same way as the ZK property of Definition 5.4.1, but w.r.t. the ideal functionality specified in Figure 5.7.1.*

Figure 5.7.1: Functionality $\mathcal{F}_{F,\phi}$ for Proof-Based OWFs Supporting Predicates

The ideal functionality $\mathcal{F}_F$ interacts with a sender S and a receiver R. Upon receiving the input $\mathsf{x} \in \{0,1\}^{a(\lambda)}$ from S and $\mathsf{r} \in \{0,1\}^{b(\lambda)}$ from R, the functionality $\mathcal{F}_F$ sends $\mathsf{x} \| \mathsf{r}$ to S, and $(F(\mathsf{x}\|\mathsf{r}), \phi(\alpha))$ to R, where α is the prefix of x with the length satisfying ϕ's input requirement.

5.7.2 Our Construction

To show how we can modify the constructions in Section 5.4 to satisfy the above requirements, we first review Construction 5.4.1 and Protocol 5.4.1 from a new perspective.

A New Interpretation of Our PB-OWF Construction. Recall that the input to F^f in Construction 5.4.1 takes the following form:

$$\mathsf{x} = \underbrace{(x_1, \ldots, x_n)}_{\alpha} \| \underbrace{(p_1, y'_{p_1}), \ldots, (p_k, y'_{p_k})}_{\beta}, \text{ and } \mathsf{r} = (b_1, \ldots, b_t).$$

On an input $\mathsf{x} = \alpha \| \beta \| \mathsf{r}$, we can think of Construction 5.4.1 as applying the following 3 steps:

- **Encoding:** apply some encoding Enc on α. In Construction 5.4.1, $\mathsf{Enc}(\alpha)$ simply partitions α to x_i's;
- **Hardness-Inducing:** perform some one-way operation on $\mathsf{Enc}(\alpha)$ to ensure that the output is hard to invert. In Construction 5.4.1, this one-way operation is just applying the oracle OWF f to each element of $\mathsf{Enc}(\alpha) = (x_1, \ldots, x_n)$. In Protocol 5.4.1, the sender reveals some portion of $\mathsf{Enc}(\alpha)$ specified by the receiver's challenge r. These openings allow R to check if S performs honestly. However, this can only ensure that S^* does not cheat for a large portion of $\mathsf{Enc}(\alpha)$; there is still a small part (say, a δ fraction) of $\mathsf{Enc}(\alpha)$ on which S^* can cheat. Therefore, we need the following **Editing** step;
- **Editing:** when β and r satisfy some predefined condition, we will edit the output of last step using the values contained in β. This step is critical to ensure the existence of a preimage under F^f for the Y learned by the receiver: as mentioned in last step, there is a small portion of $\mathsf{Enc}(\alpha)$ on which S^* may cheat. This **Editing** step essentially extends the pre-image set of Y from a single $\mathsf{Enc}(\alpha)$ to all the strings within a δ fractional distance to the valid $\mathsf{Enc}(\alpha)$. In this way, $Y \neq \bot$ will always have a preimage even if S^* can cheat on a δ fraction of $\mathsf{Enc}(\alpha)$ without being caught by R.

In light of the above perspective, we now discuss how to make the construction support ZK proofs for an arbitrary predicate $\phi(\cdot)$. Very roughly, we will pick a more robust **Encoding** approach that binds the sender to a unique α after the execution of the **Compute** stage. Moreover, it should be flexible enough to allow the sender to prove that this α satisfies some predicate $\phi(\cdot)$ later in the **Prove** stage. The **Hardness-Inducing** step will also need some change to accommodate our new **Encoding** method (such that we do not hurt the one-wayness). In the following, we elaborate on our construction.

The New Function F^f. We use a (n,t)-perfectly secure verifiable secret sharing VSS as our new Enc. Note that VSS is a randomized procedure. Fortunately, x is a random string. So we will draw randomness from some part (denoted as η) of x. That is, we encode α as $\mathsf{Enc}(\alpha;\eta) \coloneqq \mathsf{VSS}_{\mathsf{Share}}(\alpha;\eta) = ([\alpha]_1,\ldots,[\alpha]_n)$, where $\{[\alpha]_i\}$ are the VSS shares. Now, we want to apply some **Hardness-Inducing** on $\mathsf{Enc}(\alpha;\eta)$. Note that it does not suffice anymore to apply the oracle OWF f on these shares (like what we did in Construction 5.4.1), because these VSS shares are correlated (thus, α may be recoverable from $\{f([\alpha]_i)\}_{i\in[n]}$). Therefore, we instead apply Naor's commitment to these shares, which can be built from OWF in black-box. Note that Naor's commitment also requires randomness, which will be obtained from other parts of x. For the **Editing** step, we use the same approach as in Construction 5.4.1. We formalize the above intuition by showing the complete description of our new F^f in Construction 5.7.1.

The New Protocol Π^f. The protocol that computes our new F^f follows the same template as Protocol 5.4.1. The formal description is given in Protocol 5.7.1. We now explain it by comparing it with Protocol 5.4.1. The R in Protocol 5.7.1 additionally takes a string ρ as input. Looking ahead, this ρ will be used as the first message for Naor's commitment. The sender parses its input x as:

$$\mathsf{x} = \alpha\|\eta\|\underbrace{(\gamma_1,\ldots,\gamma_n)}_{\gamma}\|(p_1,c'_{p_1}),\ldots,(p_k,c'_{p_k}).$$

Same as before, S first encodes α using a (n,t)-perfectly secure VSS scheme with randomness η. Denote the resulting shares as $\{[\alpha]_i\}_{i\in[n]}$. S then commits to these shares (in parallel) using Naor's commitment, using ρ as the first message and $\{\gamma\}_{i\in[n]}$ as the respective randomness. That is, S define the following values:

$$c_1 = \mathsf{Com}_\rho([\alpha]_1;\gamma_1),\ldots,c_n = \mathsf{Com}_\rho([\alpha]_n;\gamma_n).$$

Note that S has not sent to R these values yet. These $\{c_i\}_{i\in[n]}$ values should be viewed as the analogs of $\{y_i\}_{i=[n]}$ in Stage 1 of Protocol 5.4.1. One can think of them as the result of our new **Encoding** and **Hardness-Inducing** performed on the α part in x.

Recall that the protocol Π^f needs to additionally let R learn $\phi(\alpha)$. To do that, we use the "MPC-in-the-head" technique [IKOS07, GLOV12]. The sender emulates "in his head" n parties $\{P_i\}_{i\in[n]}$, where P_i's input is the i-th share $[\alpha]_i$. These parties execute a (n,t)-perfectly secure MPC protocol for computing $\phi(\alpha)$. Let $\{\mathsf{v}_1,\ldots,\mathsf{v}_n\}$ denote the views of the n parties during the MPC execution. The sender appends these views to the ν committed in BBCom, and later reveals those specified by receiver's r (see below). From now on, Protocol 5.7.1 proceeds in an identical way as Protocol 5.4.1 except for the "consistency-checking" step:

- in Stage 5 of Protocol 5.4.1, R only checks $y_i = f(x_i)$ for the (x_i,y_i) pairs revealed by S according to r;
- while in Stage 7 of Protocol 5.7.1, the revealed $\{c_{b_i},[\alpha]_{b_i},\mathsf{v}_{b_i}\}_{i\in[t]}$ values (according to r)

have a slightly more complex relation due to our new **Encoding** and **Hardness-Inducing** method. Here, R needs to verify that c_{b_i} is indeed a valid Naor's commitment to $[\alpha]_{b_i}$, and that the revealed $\{[\alpha]_{b_i}, \mathsf{v}_{b_i}\}_{i\in[t]}$ values are consistent MPC inputs and views for the corresponding parties. This is crucial for us to obtain soundness in this setting (proved formally in Lemma 5.7.2).

Other parts of the **Compute** phase (especially, how the **Editing** is handled) are done in the same way as Protocol 5.4.1. The ZK property can be proved as before, plus the (n,t)-privacy of the VSS and MPC protocol.

Formal Description. We present the formal construction in Construction 5.7.1 and Protocol 5.7.1, relying on the following building blocks (in addition to the f and Π_{ZKCnP} for the construction in Section 5.4).

- Naor's commitment scheme Com [Nao90], which is a two-round statistically-binding commitment making only black-box use of f.
- $(n+1,t)$-perfectly secure VSS scheme $\mathsf{VSS} = (\mathsf{VSS_{Share}}, \mathsf{VSS_{Recon}})$ (see Definition 2.6.1);
- A (n,t)-perfectly secure MPC protocol (see Definition 2.7.1 and Remark 2.7.1);

For the VSS and MPC protocols, we require that t is a constant fraction of n such that $t \le n/3$. There are known constructions satisfying these properties [BGW88, CDD+99].

Construction 5.7.1: The New One-Way Function $F^f(\cdot)$

Let $n(\lambda)$ be a polynomial on λ. Let both t and k be a constant fraction of n such that $k < t \le n/3$. On input X, $F^f(X)$ is be computed as follows:

1. Parse the input X as the following two parts:

$$\mathsf{x} = \alpha \| \eta \| \gamma \| (p_1, c'_{p_1}), \ldots, (p_k, c'_{p_k}), \text{ and } \mathsf{r} \| \rho \text{ where } \mathsf{r} = (b_1, \ldots, b_t),$$

   where $|\alpha| = \lambda$, $|\rho| = 3\lambda$, and $\{p_i\}_{i\in[k]}$ and $\{b_i\}_{i\in[t]}$ should be long enough such that they form a size-k subset and a size-t subset of $[n]$, respectively (also see Remark 5.4.1).

   Remark 5.7.1. *(On the length of η, γ and c'_{p_i}'s) The η should be long enough such that it can be used as the random tape in the* VSS *execution. The γ should be long enough such that it can be used as the randomness used in Step 3 to commit to the* VSS *shares of α. Each c'_{p_i} is of the same length as a commitment to a* VSS *share (i.e., the length of c_i in Step 3). Given a concrete* VSS *scheme, the length of these values can be determined accordingly.*

2. Emulate $n+1$ (virtual) players $\{P_i\}_{i\in[n+1]}$ to execute the VSS protocol, where the input to P_{n+1} (i.e., the Dealer) is α. At the end of the execution, each player P_i ($i \in [n]$) obtains a share of α denoted as $[\alpha]_i$. The randomness for this part is taken from η.
3. For $i \in [n]$, apply Naor's commitment Com on each $[\alpha]_i$, where the first message for Com is set to ρ, and the randomness used by the committer is from $\gamma = (\gamma_1, \ldots, \gamma_n)$.

Namely, it generates:

$$c_1 = \mathsf{Com}_\rho([\alpha]_1; \gamma_1), \ldots, c_n = \mathsf{Com}_\rho([\alpha]_n; \gamma_n).$$

4. Compute $(s_1, \ldots, s_n)$ as follows:
   (a) if $\{p_1, \ldots, p_k\} \cap \{b_1, \ldots, b_t\} \neq \emptyset$, then let $s_i := c_i$ for all $i \in [n]$.
   (b) if $\{p_1, \ldots, p_k\} \cap \{b_1, \ldots, b_t\} = \emptyset$, then let $s_i := \begin{cases} c'_i & i \in \{p_1, \ldots, p_k\} \\ c_i & i \in [n] \setminus \{p_1, \ldots, p_k\} \end{cases}$.
5. Output $Y = (s_1, \ldots, s_n) \| ([\alpha]_{b_1}, \gamma_{b_1}), \ldots, ([\alpha]_{b_t}, \gamma_{b_t}) \| \rho \| (b_1, \ldots, b_t)$.

Protocol 5.7.1: Protocol $\Pi^f_{F,\phi}$

Input: the security parameter 1^λ is the common input. The sender S takes x as its private input; the receiver R takes (r, ρ) as its private input.

1. R sends ρ to S;
2. S parses the input as $\mathsf{x} = \alpha \| \eta \| \gamma \| (p_1, c'_{p_1}), \ldots, (p_k, c'_{p_k})$. S computes the $(c_1, \ldots, c_n)$ as defined in Step 3 of Construction 5.7.1. Note that $\{c_i\}_{i \in [n]}$ are supposed to be the commitments to the VSS shares $\{[\alpha]_i\}_{i \in [n]}$.
3. S emulates in its head n (virtual) players $\{P_i\}_{i \in [n]}$, where P_i's input is $[\alpha]_i$. These n parties execute the the (n, t)-perfectly secure MPC protocol for the following functionality: the functionality reconstructs α from $\{[\alpha]_i\}_{i \in [n]}$ (collected from each party), and sends $\phi(\alpha)$ to all the parties as their output. For $i \in [n]$, let v_i be the view of party P_i during the MPC execution.
4. S and R executes $\mathsf{BBCom}(\nu)$, the **Commit** stage of Π_{ZKCnP}, where S commits to the value
$$\nu := (c_1, \ldots, c_n) \| (\mathsf{v}_1, \ldots, \mathsf{v}_n) \| (p_1, c'_{p_1}), \ldots, (p_k, c'_{p_k}). \tag{5.49}$$
5. R sends r to S.
6. S interprets r as a size-t subset $\{b_1, \ldots, b_t\} \subseteq [n]$. S computes the $\mathsf{s} = (s_1, \ldots, s_n)$ as defined in Step 4 of Construction 5.7.1. S sends the values s and $\{(c_{b_i}, [\alpha]_{b_i}, \gamma_{b_i}, \mathsf{v}_{b_i})\}_{i \in [t]}$. (Recall that $[\alpha]_{b_i}$ is the value committed in c_{b_i} using randomness γ_{b_i}.)
7. Upon receiving the values S sends in last stage, R checks:
   (a) $\mathsf{Com}_\rho([\alpha]_{b_i}; \gamma_{b_i}) = c_{b_i}$ holds for all $i \in [t]$; **and**
   (b) $\{[\alpha]_{b_1}, \ldots, [\alpha]_{b_t}\}$ are consistent w.r.t. the VSS procedure; **and**
   (c) $\{\mathsf{v}_{b_1}, \ldots, \mathsf{v}_{b_t}\}$ constitute consistent (as per Definition 2.7.2) views w.r.t. the MPC execution as described in Stage 3. We remark that this includes checking that $[\alpha]_{b_i}$ is the prefix of v_{b_i}.

If all the checks pass, R proceeds to next step; otherwise, R halts and outputs $\perp$.

8. S and R execute $\mathsf{BBProve}$, the **Prove** stage of Π_{ZKCnP}, where S proves that it performed Stage 6 honestly. That is, S proves that the ν committed at Stage 4 satisfies the following conditions:

   (a) the values $\{p_1, \ldots, p_k\}$ contained in ν form a size-k subset of $[n]$; **and**

   (b) the $(c_{b_1}, \ldots, c_{b_t})$ revealed by S in Stage 6 do consist of the subset of $\{c_i\}_{i \in [n]}$ (contained in ν) specified by r (sent by R in Stage 5); **and**

   (c) the $(\mathsf{v}_{b_1}, \ldots, \mathsf{v}_{b_t})$ revealed by S in Stage 6 do consist of the subset of $\{\mathsf{v}_i\}_{i \in [n]}$ specified by r; **and**

   (d) The $\mathsf{s} = (s_1, \ldots, s_n)$ satisfies the following conditions:
      - if $\{p_1, \ldots, p_k\} \cap \{b_1, \ldots, b_t\} \neq \emptyset$, then $s_i = c_i$ for all $i \in [n]$.
      - if $\{p_1, \ldots, p_k\} \cap \{b_1, \ldots, b_t\} = \emptyset$, then $s_i = \begin{cases} c'_i & i \in \{p_1, \ldots, p_k\} \\ c_i & i \in [n] \setminus \{p_1, \ldots, p_k\} \end{cases}$.

9. **(Receiver's Output.)** R outputs

$$Y = (s_1, \ldots, s_n) \| ([\alpha]_{b_1}, \gamma_{b_1}), \ldots, ([\alpha]_{b_t}, \gamma_{b_t}) \| \rho \| (b_1, \ldots, b_t)$$

   and $u = \phi(\alpha)$ (this value can be obtained from the MPC views revealed to R).

10. **(Sender's Output.)** S outputs $X = \alpha \| \eta \| \gamma \| (p_1, c'_{p_1}), \ldots, (p_k, c'_{p_k}) \| \rho \| (b_1, \ldots, b_t)$.

5.7.3 Security Proof

Theorem 5.7.1. *The constructions shown in Construction 5.7.1 and Protocol 5.7.1 is a PB-OWF that supports predicates (as per Definition 5.7.1). Moreover, it only makes black-box use of OWFs.*

It follows immediately from the description that our construction makes only black-box use of the OWF f, and it also satisfies the complement requirement as per Definition 5.7.1. In the following, we establish the one-wayness in Lemma 5.7.1, soundness in Lemma 5.7.2, and zero-knowledge property in Lemma 5.7.3.

Lemma 5.7.1 (One-Wayness). *The function F^f in Construction 5.7.1 is one-way as defined in Definition 5.7.1.*

Proof. We reduce one-wayness to the computationally-hiding property of Naor's commitment Com. Concretely, given a PPT machine $\mathcal{A}_{\mathrm{ow}}$ breaking one-wayness of F^f, we will construct a PPT machine $\mathcal{A}_{\mathsf{Com}}$ that breaks the security of Com. The machine $\mathcal{A}_{\mathsf{Com}}$ works in the following way:

- It samples randomly the $\mathsf{x} = \alpha \| \eta \| \gamma \| (p_1, c'_{p_1}), \ldots, (p_k, c'_{p_k})$, $\mathsf{r} = \{b_1, \ldots, b_t\}$ and ρ defined in Construction 5.7.1.
- It computes $\{[\alpha]_1, \ldots, [\alpha]_n\}$ as specified in Step 2 of Construction 5.7.1.

- $\mathcal{A}_{\mathsf{Com}}$ forwards ρ (the first-round message for Naor's commitment) and $\overline{m}_0 = \{0^{|[\alpha]_i|}\}_{i\in[n]\setminus\mathsf{r}}$ and $\overline{m}_1 = \{[\alpha]_i\}_{i\in[n]\setminus\mathsf{r}}$ to the Com challenger.
- The challenger picks a random bit $b \xleftarrow{\$} \{0,1\}$ and commits to each element in $\overline{m}_b$ in parallel, using ρ as the first message. Denote the commitments from the challenger as $\{c^*_i\}_{i\in[n]\setminus\mathsf{r}}$.
- $\mathcal{A}_{\mathsf{Com}}$ then computes $\{s_i\}_{i\in[n]}$ where $s_i = \begin{cases} c^*_i & i \in [n]\setminus\mathsf{r} \\ \mathsf{Com}_\rho([\alpha]_i;\gamma_i) & i \in \mathsf{r} \end{cases}$.
- It internally invokes $\mathcal{A}_{\mathrm{ow}}$ on the input

$$Y^* := (s_1,\ldots,s_n)\|([\alpha]_{b_1},\gamma_{b_1}),\ldots,([\alpha]_{b_t},\gamma_{b_t})\|\rho\|(b_1,\ldots,b_t),$$

  and in turn receives a value X^*.
- $\mathcal{A}_{\mathsf{Com}}$ outputs 1 if and only if $F^f(X^*) = Y^*$.

We then argue that $\mathcal{A}_{\mathsf{Com}}$ wins the hiding game with non-negligible probability. There are two possible cases depending on the Com challenger's choice of b:

1. the challenger commits to $\overline{m}_1$: the Y^* is identically distributed as a $F^f(\cdot)$ evaluation on a random input. Thus, $\mathcal{A}_{\mathrm{ow}}$ will find a valid preimage X^* with non-negligible probability, which implies that $\mathcal{A}_{\mathsf{Com}}$ will guess $b = 1$ correctly with non-negligible probability. We remark that there is a negligible chance that $\mathsf{r} \cap \{p_1,\ldots,p_k\} \neq \emptyset$. But this can be safely ignored without affecting the current proof.
2. the challenger commits to $\overline{m}_0$: note that $\overline{m}_0 \cup \{\mathsf{v}_i\}_{i\in[n]\setminus\mathsf{r}}$ cannot be consistent VSS shares as $\overline{m}_0$ contains only 0 strings. Moreover, it follows from the statistically-binding property that $\{c^*_{b_1}\}_{i\in[t]}$ (the commitments to $\overline{m}_0$) cannot be interpreted as commitments to values other than $\overline{m}_0$ (except with negligible probability). Since these $\{c^*_{b_1}\}_{i\in[t]}$ values are contained in Y^*, even an unbounded adversary cannot find and X^* such that $F^f(X^*) = Y^*$ (except with negligible probability). Thus, in this case, $\mathcal{A}_{\mathsf{Com}}$ will output 1 with negligible probability.

The above analysis shows that $\Big|\Pr[\mathcal{A}_{\mathsf{Com}} = 1|b = 1] - Pr[\mathcal{A}_{\mathsf{Com}} = 1|b = 0]\Big|$ is non-negligible. Therefore, $\mathcal{A}_{\mathsf{Com}}$ breaks the hiding property of Com. □

Lemma 5.7.2 (Soundness.). *Protocol 5.7.1 satisfies the soundness property specified in Definition 5.7.1.*

Proof. From our new perspective described at the beginning of Section 5.7.2, it is not hard to see that the soundness follows from the same argument used for Lemma 5.4.2 except for the following two caveats:

1. As mentioned earlier, the **Encoding** and **Hardness-Inducing** methods used in Protocol 5.7.1 (and Construction 5.7.1) are different from those in Protocol 5.4.1. As a result, the receiver in Protocol 5.7.1 needs to check the consistency of the revealed VSS shares. We need to argue that such checks suffice to ensure the existence of a preimage.

2. Lemma 5.4.2 ensures the existence of a preimage X for the non-aborting Y learned by R. In Protocol 5.7.1, R additionally output a value u; so we need to (additionally) argue that u is the result of ϕ evaluated on the prefix α of X.

In the following, we address the above two points in order.

Addressing the First Point. To prove soundness for the current construction, we need to argue formally that if S^* cheats on the $(c_1, \ldots, c_n)$ values, R will abort with overwhelming probability. Recall that the counterpart of c_i's in Protocol 5.4.1 are $(y_1, \ldots, y_n)$, where y_i is supposed to be $f(x_i)$ for some underlying x_i; and in the proof of Lemma 5.4.2, we need to prove Claim 5.4.1, which says that there are at most δn many "bad" y_i's (i.e., there are no pre-images for them). In the current proof, we need an analog of this claim: here, the c_i's are commitments to VSS shares; so we need to argue that if S^* commits to more than k "bad" shares (defined later), R will abort with overwhelming probability by checking t (out of n) shares.

Formally, let $([\alpha]^*_1, \ldots, [\alpha]^*_n)$ be the values committed in $\{c_i\}_{i \in [n]}$ by S^*. We denote the following event, which should be considered as an analog of the event "Bad_Y" defined in the proof of Lemma 5.4.2:

- Bad_Y: there are at least $k = \delta n$ shares in $\{[\alpha]^*_i\}_{i \in [n]}$ that are "bad", i.e., it is *impossible* to convert these $\{[\alpha]^*_i\}_{i \in [n]}$ values to consistent VSS shares by modifying less than $k = \delta n$ of them (where $0 < \delta < 1$ is a constant).

We now show that, by checking a size-t random subset of them, R will catch at least one *pair* of inconsistent shares with overwhelming probability. In the following, we prove it relying on the "inconsistency graph" technique from [IKOS07].

Define a graph G with n vertices (corresponding to the n views). Assign an edge between node i and j in G if the views $[\alpha]^*_i$ and $[\alpha]^*_j$ are inconsistent w.r.t. VSS. When Bad_Y happens, it must be that the *minimum vertex cover* of G has size at least $k = \delta n$. We would like to argue that a random choice of t (a constant fraction of n) vertices will hit an edge with overwhelming probability. For this, we use the well-known connection between the size of a minimum vertex cover to the size of a *maximum matching*. Concretely, the graph G must have a matching $\mathcal{M}$ of size at least $\frac{\delta n}{2}$. (Otherwise, if the maximum matching contains less than $\frac{\delta n}{2}$ edges, the vertices of this matching form a vertex cover of size less than δn.) If the receiver R picks at least one edge of G, he will reject. The probability that the t vertices (shares) that the receiver picks miss all the edges of G is smaller than the probability that he misses all edges in $\mathcal{M}$. As shown in the following, the latter is negligible.

We use P to denote the number of distinct pairs of vertices in G, i.e. $P \coloneqq \binom{n}{2}$. We use T to denote the distinct pairs of t different vertices, i.e. $T \coloneqq \binom{t}{2}$. We use K to denote the size of $\mathcal{M}$, i.e. $K \coloneqq \frac{\delta n}{2}$. Then, the following holds:

$$
\begin{aligned}
\Pr[\text{all edges in } \mathcal{M} \text{ are missed}] &= \frac{\binom{P-K}{T}}{\binom{P}{T}} = \frac{(P-K-T+1)(P-K-T+2)\cdots(P-K)}{(P-T+1)(P-T+2)\cdots P} \\
&= \left(1 - \frac{K}{P-T+1}\right)\left(1 - \frac{K}{P-T+2}\right)\cdots\left(1 - \frac{K}{P}\right) \\
&\leq \left(1 - \frac{K}{P}\right)^{T} = \left(1 - \frac{\delta}{n-1}\right)^{\frac{t(t-1)}{2}} \\
&= e^{-\Omega(\frac{t^2}{n})} \qquad (5.50)
\end{aligned}
$$

By our choice of parameters, t is a constant fraction of n. Therefore, the above probability is negligible.

Remark 5.7.2 (Binding to α)**.** *It is worth noting that the above argument shows that there are at least $(n-k)$ consistent* VSS *shares. Since the parameter k is no larger than the* VSS *threshold t (by our choice of parameter), these $(n-k)$ consistent shares statistically bind the sender to a unique α.*

Addressing the Second Point. To handle this issue, Protocol 5.7.1 follows the black-box commit-and-prove technique in [IKOS07, GLOV12, GOSV14]: in Stage 3, the sender performs the MPC-in-the-head execution to compute $\phi(\alpha)$; later in Stage 5 and Stage 7c, the receiver checks a size-t random subset of the views of the MPC-in-the-head parties. Following the same "inconsistency graph" argument as before, it is not hard to see that at least $(n-k)$ parties have consistent views w.r.t. the MPC execution for $\phi(\alpha)$. Since $k \leq t$ and the MPC protocol used in our construction is (n,t)-perfectly secure[11], it follows that $\phi(\alpha)$ is computed honestly. More accurately, this means that at least $(n-k)$ parties receive the same $\phi(\alpha)$ value as the output, where the α is the (unique) value reconstructed from the VSS shares from these $(n-k)$ parties' input (i.e., the value to which S was bound as described in Remark 5.7.2).

This finishes the proof for soundness. $\square$

Lemma 5.7.3 (Zero-Knowledge)**.** *Protocol 5.7.1 satisfies the zero-knowledge property defined in Definition 5.7.1.*

Proof. To prove the zero-knowledge property, we need to show a PPT ideal-world simulator Sim for any PPT malicious receiver R^*. At a high-level, such a simulator can be constructed as follows. Sim will use the simulator of the commit-and-prove protocol Π_{ZKCnP} at Stages 4 and 8. We will show how this allows Sim to finish the interaction without knowing the sender's input x.

Formally, we will build 2 hybrids starting from the real execution between the honest sender and R^*, and show that the second hybrid is essentially the simulator we want. We use Out_{H_i} ($i \in \{0,1\}$) to denote the output of hybrid H_i.

[11]Here, (n,t)-perfect robustness (Definition 2.7.5) will suffice.

Hybrid $H_0(1^\lambda, \mathsf{x}, z)$**.** This hybrid uses the strategy of the honest $S(1^\lambda, \mathsf{x})$ to interact with the corrupted receiver $R^*(1^\lambda, z)$. At the end of the execution, H_0 outputs whatever R^* outputs. This hybrid is exactly the real execution.

Hybrid $H_1(1^\lambda, \mathsf{x}, z)$**.** This hybrid is identical to the previous one, except that

– At Stage 4, instead of executing BBCom, H_1 uses the strategy of Sim_1 in its communication with R^*, where Sim_1 is the simulator for the **Commit** stage of Π_{ZKCnP} (see Definition 5.2.2).

– At Stage 8, instead of doing the proof honestly, H_1 uses the strategy of Sim_2 in its communication with R^*, where Sim_1 is the simulator for the **Prove** stage of Π_{ZKCnP} (see Definition 5.2.2).

$\mathsf{Out}_{H_0} \overset{\text{c}}{\approx} \mathsf{Out}_{H_1}$: This is due to the ZK property of Π_{ZKCnP}.

The Simulator $\mathsf{Sim}(1^\lambda, z)$**.** We now describe the simulator Sim for the ideal execution. Sim is identical to H_1 except for the following changes:

– Sim does not need to execute Stage 2;

– Upon receiving $\mathsf{r} = (b_1, \ldots, b_t)$ at Stage 5, Sim sends r and ρ (received from Stage 1) to the idea functionality $\mathcal{F}_{F,\phi}$, and receives back

$$Y = (s_1, \ldots, s_n) \| ([\alpha]_{b_1}, \gamma_{b_1}), \ldots, ([\alpha]_{b_t}, \gamma_{b_t}) \| \rho \| (b_1, \ldots, b_t), \quad \text{and} \quad \phi(\alpha).$$

– With $\{[\alpha]_{b_i}\}_{i \in [t]}$ (the input to P_{b_i}'s) and $\phi(x)$ (the output of P_{b_i}'s), Sim run the MPC simulator to generate the simulated views $\{\tilde{\mathsf{v}}_{b_i}\}_{i \in [t]}$ for parties $\{P_{b_i}\}_{i \in [t]}$. Since the MPC protocol is (n,t)-perfectly secure[12], the simulated views are identically distributed to the views in the real execution.

– At Stage 6, Sim sends to R^* the values $\mathsf{s} = (s_1, \ldots, s_n)$ and $\{(c_{b_i}, [\alpha]_{b_i}, \gamma_{b_i}, \tilde{\mathsf{v}}_{b_i})\}_{i \in [t]}$ (Note that s and $\{(c_{b_i}, [\alpha]_{b_i}, \gamma_{b_i})\}_{i \in [t]}$ are contained in Y^*.)

We remark that, unlike H_1, Sim does not need to know the input x to the sender; the Y^* it obtained from $\mathcal{F}_{F,\phi}$ contains all the information it needs to finish its execution with R^*. In particular, although the s and $\{(c_{b_i}, [\alpha]_{b_i}, \gamma_{b_i})\}_{i \in [t]}$ values now come from the ideal functionality $\mathcal{F}_{F,\phi}$, they are identically distributed to the ones generated by the honest sender. As mentioned earlier, the simulated views $\{\tilde{\mathsf{v}}_{b_i}\}_{i \in [t]}$ are also identically distributed to the views in the real execution. It then follows that the output of Sim is identical to H_1.

This finishes the proof for Lemma 5.7.3. □

[12]Here, semi-honest (n,t)-computational privacy (Definition 2.7.3) will suffice.

Bibliography

[AB09] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009. 13

[ABG$^+$20] Benny Applebaum, Zvika Brakerski, Sanjam Garg, Yuval Ishai, and Akshayaram Srinivasan. Separating two-round secure computation from oblivious transfer. In *11th Innovations in Theoretical Computer Science Conference (ITCS 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020. 3

[AL11] Gilad Asharov and Yehuda Lindell. A full proof of the BGW protocol for perfectly-secure multiparty computation. Cryptology ePrint Archive, Report 2011/136, 2011. `https://eprint.iacr.org/2011/136`. 19

[Bar01] Boaz Barak. How to go beyond the black-box simulation barrier. In *42nd Annual Symposium on Foundations of Computer Science*, pages 106–115, Las Vegas, NV, USA, October 14–17, 2001. IEEE Computer Society Press. 7, 15, 25, 27, 29, 35, 36, 39, 40

[Bar02] Boaz Barak. Constant-round coin-tossing with a man in the middle or realizing the shared random string model. In *43rd Annual Symposium on Foundations of Computer Science*, pages 345–355, Vancouver, BC, Canada, November 16–19, 2002. IEEE Computer Society Press. 4, 24

[BBF13] Paul Baecher, Christina Brzuska, and Marc Fischlin. Notions of black-box reductions, revisited. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology – ASIACRYPT 2013, Part I*, volume 8269 of *Lecture Notes in Computer Science*, pages 296–315, Bengalore, India, December 1–5, 2013. Springer, Heidelberg, Germany. 1

[BCL$^+$05] Boaz Barak, Ran Canetti, Yehuda Lindell, Rafael Pass, and Tal Rabin. Secure computation without authentication. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 361–377, Santa Barbara, CA, USA, August 14–18, 2005. Springer, Heidelberg, Germany. 91

[BCNP04] Boaz Barak, Ran Canetti, Jesper Buus Nielsen, and Rafael Pass. Universally composable protocols with relaxed set-up assumptions. In *45th Annual Symposium on Foundations of Computer Science*, pages 186–195, Rome, Italy, October 17–19, 2004. IEEE Computer Society Press. 10

[BDH$^+$17] Brandon Broadnax, Nico Döttling, Gunnar Hartung, Jörn Müller-Quade, and Matthias Nagel. Concurrently composable security with shielded super-polynomial simulators. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017, Part I*, volume 10210 of *Lecture Notes in Computer Science*, pages 351–381, Paris, France, April 30 – May 4, 2017. Springer, Heidelberg, Germany. 4, 5

[Bea92] Donald Beaver. Foundations of secure interactive computing. In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 377–391, Santa Barbara, CA, USA, August 11–15, 1992. Springer, Heidelberg, Germany. 31

[BG93] Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In Ernest F. Brickell, editor, *Advances in Cryptology – CRYPTO'92*, volume 740 of *Lecture Notes in Computer Science*, pages 390–420, Santa Barbara, CA, USA, August 16–20, 1993. Springer, Heidelberg, Germany. 110

[BGH$^+$04] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. Robust PCPs of proximity, shorter PCPs and applications to coding. In László Babai, editor, *36th Annual ACM Symposium on Theory of Computing*, pages 1–10, Chicago, IL, USA, June 13–16, 2004. ACM Press. 35

[BGJ$^+$17] Saikrishna Badrinarayanan, Vipul Goyal, Abhishek Jain, Dakshita Khurana, and Amit Sahai. Round optimal concurrent MPC via strong simulation. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part I*, volume 10677 of *Lecture Notes in Computer Science*, pages 743–775, Baltimore, MD, USA, November 12–15, 2017. Springer, Heidelberg, Germany. 8

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *20th Annual ACM Symposium on Theory of Computing*, pages 1–10, Chicago, IL, USA, May 2–4, 1988. ACM Press. 21, 22, 164

[BL02] Boaz Barak and Yehuda Lindell. Strict polynomial-time in simulation and extraction. In *34th Annual ACM Symposium on Theory of Computing*, pages 484–493, Montréal, Québec, Canada, May 19–21, 2002. ACM Press. 26, 29

[Blu82] Manuel Blum. Coin flipping by telephone. In *Proc. IEEE Spring COMPCOM*, pages 133–137, 1982. 108

[BM07] Boaz Barak and Mohammad Mahmoody-Ghidary. Lower bounds on signatures from symmetric primitives. In *48th Annual Symposium on Foundations of Computer Science*, pages 680–688, Providence, RI, USA, October 20–23, 2007. IEEE Computer Society Press. 1

[BM09] Boaz Barak and Mohammad Mahmoody-Ghidary. Merkle puzzles are optimal - an $O(n^2)$-query attack on any key exchange from a random oracle. In

Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 374–390, Santa Barbara, CA, USA, August 16–20, 2009. Springer, Heidelberg, Germany. 1

[BM17] Boaz Barak and Mohammad Mahmoody-Ghidary. Merkle's key agreement protocol is optimal: An $O(n^2)$ attack on any key agreement from random oracles. *Journal of Cryptology*, 30(3):699–734, July 2017. 123

[BP12] Nir Bitansky and Omer Paneth. From the impossibility of obfuscation to a new non-black-box simulation technique. In *53rd Annual Symposium on Foundations of Computer Science*, pages 223–232, New Brunswick, NJ, USA, October 20–23, 2012. IEEE Computer Society Press. 24

[BP13] Nir Bitansky and Omer Paneth. On the impossibility of approximate obfuscation and applications to resettable cryptography. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th Annual ACM Symposium on Theory of Computing*, pages 241–250, Palo Alto, CA, USA, June 1–4, 2013. ACM Press. 24

[BPS06] Boaz Barak, Manoj Prabhakaran, and Amit Sahai. Concurrent non-malleable zero knowledge. In *47th Annual Symposium on Foundations of Computer Science*, pages 345–354, Berkeley, CA, USA, October 21–24, 2006. IEEE Computer Society Press. 86

[BS05] Boaz Barak and Amit Sahai. How to play almost any mental game over the net - concurrent composition via super-polynomial simulation. In *46th Annual Symposium on Foundations of Computer Science*, pages 543–552, Pittsburgh, PA, USA, October 23–25, 2005. IEEE Computer Society Press. 4, 5, 93

[Can00a] Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, January 2000. 20, 31

[Can00b] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2000. `https://eprint.iacr.org/2000/067`. 89

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science*, pages 136–145, Las Vegas, NV, USA, October 14–17, 2001. IEEE Computer Society Press. 3, 8, 29, 38, 90, 92, 93, 113

[Can04] Ran Canetti. Universally composable signature, certification, and authentication. In *17th IEEE Computer Security Foundations Workshop, (CSFW-17 2004), 28-30 June 2004, Pacific Grove, CA, USA*, page 219. IEEE Computer Society, 2004. 91

[CCOV19] Nishanth Chandran, Wutichai Chongchitmate, Rafail Ostrovsky, and Ivan Visconti. Universally composable secure computation with corrupted tokens. In

Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 432–461, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany. 10

[CDD$^+$99] Ronald Cramer, Ivan Damgård, Stefan Dziembowski, Martin Hirt, and Tal Rabin. Efficient multiparty computations secure against an adaptive adversary. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 311–326, Prague, Czech Republic, May 2–6, 1999. Springer, Heidelberg, Germany. 21, 164

[CDMW08] Seung Geol Choi, Dana Dachman-Soled, Tal Malkin, and Hoeteck Wee. Black-box construction of a non-malleable encryption scheme from any semantically secure one. In Ran Canetti, editor, *TCC 2008: 5th Theory of Cryptography Conference*, volume 4948 of *Lecture Notes in Computer Science*, pages 427–444, San Francisco, CA, USA, March 19–21, 2008. Springer, Heidelberg, Germany. 9

[CDMW09] Seung Geol Choi, Dana Dachman-Soled, Tal Malkin, and Hoeteck Wee. Simple, black-box constructions of adaptively secure protocols. In Omer Reingold, editor, *TCC 2009: 6th Theory of Cryptography Conference*, volume 5444 of *Lecture Notes in Computer Science*, pages 387–402. Springer, Heidelberg, Germany, March 15–17, 2009. 25

[CDPW07] Ran Canetti, Yevgeniy Dodis, Rafael Pass, and Shabsi Walfish. Universally composable security with global setup. In Salil P. Vadhan, editor, *TCC 2007: 4th Theory of Cryptography Conference*, volume 4392 of *Lecture Notes in Computer Science*, pages 61–85, Amsterdam, The Netherlands, February 21–24, 2007. Springer, Heidelberg, Germany. 93

[CF01] Ran Canetti and Marc Fischlin. Universally composable commitments. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 19–40, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Heidelberg, Germany. 3, 10

[CGGM00] Ran Canetti, Oded Goldreich, Shafi Goldwasser, and Silvio Micali. Resettable zero-knowledge (extended abstract). In *32nd Annual ACM Symposium on Theory of Computing*, pages 235–244, Portland, OR, USA, May 21–23, 2000. ACM Press. 86

[CGMA85] Benny Chor, Shafi Goldwasser, Silvio Micali, and Baruch Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults (extended abstract). In *26th Annual Symposium on Foundations of Computer Science*, pages 383–395, Portland, Oregon, October 21–23, 1985. IEEE Computer Society Press. 20

[CHH$^+$07] Ronald Cramer, Goichiro Hanaoka, Dennis Hofheinz, Hideki Imai, Eike Kiltz, Rafael Pass, abhi shelat, and Vinod Vaikuntanathan. Bounded CCA2-secure encryption. In Kaoru Kurosawa, editor, *Advances in Cryptology – ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 502–518, Kuching, Malaysia, December 2–6, 2007. Springer, Heidelberg, Germany. 9

[CK88] Claude Crépeau and Joe Kilian. Achieving oblivious transfer using weakened security assumptions (extended abstract). In *29th Annual Symposium on Foundations of Computer Science*, pages 42–52, White Plains, NY, USA, October 24–26, 1988. IEEE Computer Society Press. 1, 6

[CKL03] Ran Canetti, Eyal Kushilevitz, and Yehuda Lindell. On the limitations of universally composable two-party computation without set-up assumptions. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 68–86, Warsaw, Poland, May 4–8, 2003. Springer, Heidelberg, Germany. 3, 10

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *34th Annual ACM Symposium on Theory of Computing*, pages 494–503, Montréal, Québec, Canada, May 19–21, 2002. ACM Press. 3, 10

[CLP10] Ran Canetti, Huijia Lin, and Rafael Pass. Adaptive hardness and composable security in the plain model from standard assumptions. In *51st Annual Symposium on Foundations of Computer Science*, pages 541–550, Las Vegas, NV, USA, October 23–26, 2010. IEEE Computer Society Press. 5, 9, 86, 89, 113, 114

[CLP16] Ran Canetti, Huijia Lin, and Rafael Pass. Adaptive hardness and composable security in the plain model from standard assumptions. *SIAM J. Comput.*, 45(5):1793–1834, 2016. 114

[CLP20a] Rohit Chatterjee, Xiao Liang, and Omkant Pandey. Improved black-box constructions of composable secure computation. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *ICALP 2020: 47th International Colloquium on Automata, Languages and Programming*, volume 168 of *LIPIcs*, pages 28:1–28:20, Saarbrücken, Germany, July 8–11, 2020. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik. 6, 36, 122

[CLP20b] Rohit Chatterjee, Xiao Liang, and Omkant Pandey. Improved black-box constructions of composable secure computation. *IACR Cryptol. ePrint Arch.*, 2020:494, 2020. 36

[Coo71] Stephen A Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158, 1971. 6

[CPS13] Kai-Min Chung, Rafael Pass, and Karn Seth. Non-black-box simulation from one-way functions and applications to resettable security. In Dan Boneh, Tim

Roughgarden, and Joan Feigenbaum, editors, *45th Annual ACM Symposium on Theory of Computing*, pages 231–240, Palo Alto, CA, USA, June 1–4, 2013. ACM Press. 24

[Dam90] Ivan Damgård. A design principle for hash functions. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 416–427, Santa Barbara, CA, USA, August 20–24, 1990. Springer, Heidelberg, Germany. 156

[DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography (extended abstract). In *23rd Annual ACM Symposium on Theory of Computing*, pages 542–552, New Orleans, LA, USA, May 6–8, 1991. ACM Press. 9, 85

[DI05] Ivan Damgård and Yuval Ishai. Constant-round multiparty computation using a black-box pseudorandom generator. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 378–394, Santa Barbara, CA, USA, August 14–18, 2005. Springer, Heidelberg, Germany. 6

[DMRV13] Dana Dachman-Soled, Tal Malkin, Mariana Raykova, and Muthuramakrishnan Venkitasubramaniam. Adaptive and concurrent secure computation from new adaptive, non-malleable commitments. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology – ASIACRYPT 2013, Part I*, volume 8269 of *Lecture Notes in Computer Science*, pages 316–336, Bengalore, India, December 1–5, 2013. Springer, Heidelberg, Germany. 8

[DNS98] Cynthia Dwork, Moni Naor, and Amit Sahai. Concurrent zero-knowledge. In *30th Annual ACM Symposium on Theory of Computing*, pages 409–418, Dallas, TX, USA, May 23–26, 1998. ACM Press. 47, 85

[DPP98] Ivan B Damgard, Torben P Pedersen, and Birgit Pfitzmann. Statistical secrecy and multibit commitments. *IEEE Transactions on Information Theory*, 44(3):1143–1151, 1998. 16

[FS90] Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *22nd Annual ACM Symposium on Theory of Computing*, pages 416–426, Baltimore, MD, USA, May 14–16, 1990. ACM Press. 3, 6, 88

[Gay19] Romain Gay. *Public-Key Encryption, Revisited: Tight Security and Richer Functionalities*. PhD thesis, Université Paris Sciences et Lettres, 2019. viii

[GGJ13] Vipul Goyal, Divya Gupta, and Abhishek Jain. What information is leaked under concurrent composition? In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 220–238, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany. 4

[GGJS11] Sanjam Garg, Vipul Goyal, Abhishek Jain, and Amit Sahai. Bringing people of different beliefs together to do UC. In Yuval Ishai, editor, *TCC 2011: 8th Theory of Cryptography Conference*, volume 6597 of *Lecture Notes in Computer Science*, pages 311–328, Providence, RI, USA, March 28–30, 2011. Springer, Heidelberg, Germany. 10

[GGJS12] Sanjam Garg, Vipul Goyal, Abhishek Jain, and Amit Sahai. Concurrently secure computation in constant rounds. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 99–116, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany. 16, 26, 89

[GGK03] Rosario Gennaro, Yael Gertner, and Jonathan Katz. Lower bounds on the efficiency of encryption and digital signature schemes. In *35th Annual ACM Symposium on Theory of Computing*, pages 417–425, San Diego, CA, USA, June 9–11, 2003. ACM Press. 1

[GGMP16] Sanjam Garg, Divya Gupta, Peihan Miao, and Omkant Pandey. Secure multiparty RAM computation in constant rounds. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B: 14th Theory of Cryptography Conference, Part I*, volume 9985 of *Lecture Notes in Computer Science*, pages 491–520, Beijing, China, October 31 – November 3, 2016. Springer, Heidelberg, Germany. 6

[GGS15] Vipul Goyal, Divya Gupta, and Amit Sahai. Concurrent secure computation via non-black box simulation. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 23–42, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany. 31

[GHMM18] Sanjam Garg, Mohammad Hajiabadi, Mohammad Mahmoody, and Ameer Mohammed. Limits on the power of garbling techniques for public-key encryption. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part III*, volume 10993 of *Lecture Notes in Computer Science*, pages 335–364, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Heidelberg, Germany. 1

[GIKR01] Rosario Gennaro, Yuval Ishai, Eyal Kushilevitz, and Tal Rabin. The round complexity of verifiable secret sharing and secure multicast. In *33rd Annual ACM Symposium on Theory of Computing*, pages 580–589, Crete, Greece, July 6–8, 2001. ACM Press. 21

[GJ13] Vipul Goyal and Abhishek Jain. On concurrently secure computation in the multiple ideal query model. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 684–701, Athens, Greece, May 26–30, 2013. Springer, Heidelberg, Germany. 4, 9

[GJO10] Vipul Goyal, Abhishek Jain, and Rafail Ostrovsky. Password-authenticated session-key generation on the internet in the plain model. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 277–294, Santa Barbara, CA, USA, August 15–19, 2010. Springer, Heidelberg, Germany. 4

[GK90] Oded Goldreich and Hugo Krawczyk. On the composition of zero-knowledge proof systems. In *Automata, Languages and Programming, 17th International Colloquium, ICALP*, pages 268–282, 1990. 38

[GKM$^+$00] Yael Gertner, Sampath Kannan, Tal Malkin, Omer Reingold, and Mahesh Viswanathan. The relationship between public key encryption and oblivious transfer. In *41st Annual Symposium on Foundations of Computer Science*, pages 325–335, Redondo Beach, CA, USA, November 12–14, 2000. IEEE Computer Society Press. 1

[GKOV12] Sanjam Garg, Abishek Kumarasubramanian, Rafail Ostrovsky, and Ivan Visconti. Impossibility results for static input secure computation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 424–442, Santa Barbara, CA, USA, August 19–23, 2012. Springer, Heidelberg, Germany. 9

[GKP17] Sanjam Garg, Susumu Kiyoshima, and Omkant Pandey. On the exact round complexity of self-composable two-party computation. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017, Part II*, volume 10211 of *Lecture Notes in Computer Science*, pages 194–224, Paris, France, April 30 – May 4, 2017. Springer, Heidelberg, Germany. 16

[GKP18] Sanjam Garg, Susumu Kiyoshima, and Omkant Pandey. A new approach to black-box concurrent secure computation. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 566–599, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany. 6, 8, 24, 25, 26, 28, 29, 31, 47, 48, 49, 51, 52, 53, 56, 59, 64

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *21st Annual ACM Symposium on Theory of Computing*, pages 25–32, Seattle, WA, USA, May 15–17, 1989. ACM Press. 10, 11

[GL91] Shafi Goldwasser and Leonid A. Levin. Fair computation of general functions in presence of immoral majority. In Alfred J. Menezes and Scott A. Vanstone, editors, *Advances in Cryptology – CRYPTO'90*, volume 537 of *Lecture Notes in Computer Science*, pages 77–93, Santa Barbara, CA, USA, August 11–15, 1991. Springer, Heidelberg, Germany. 31

[GL02] Shafi Goldwasser and Yehuda Lindell. Secure computation without agreement. In Dahlia Malkhi, editor, *Distributed Computing, 16th International Conference,* *DISC 2002, Toulouse, France, October 28-30, 2002 Proceedings*, volume 2508 of *Lecture Notes in Computer Science*, pages 17–32. Springer, 2002. 31

[GLM$^+$04] Rosario Gennaro, Anna Lysyanskaya, Tal Malkin, Silvio Micali, and Tal Rabin. Algorithmic tamper-proof (ATP) security: Theoretical foundations for security against hardware tampering. In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 258–277, Cambridge, MA, USA, February 19–21, 2004. Springer, Heidelberg, Germany. 1, 6, 10

[GLOV12] Vipul Goyal, Chen-Kuei Lee, Rafail Ostrovsky, and Ivan Visconti. Constructing non-malleable commitments: A black-box approach. In *53rd Annual Symposium on Foundations of Computer Science*, pages 51–60, New Brunswick, NJ, USA, October 20–23, 2012. IEEE Computer Society Press. 6, 7, 9, 12, 23, 31, 36, 87, 108, 122, 163, 169

[GLP$^+$15] Vipul Goyal, Huijia Lin, Omkant Pandey, Rafael Pass, and Amit Sahai. Round-efficient concurrently composable secure computation via a robust extraction lemma. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part I*, volume 9014 of *Lecture Notes in Computer Science*, pages 260–289, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany. 5, 36, 86

[GLPV20] Sanjam Garg, Xiao Liang, Omkant Pandey, and Ivan Visconti. Black-box constructions of bounded-concurrent secure computation. In Clemente Galdi and Vladimir Kolesnikov, editors, *SCN 20: 12th International Conference on Security in Communication Networks*, volume 12238 of *Lecture Notes in Computer Science*, pages 87–107, Amalfi, Italy, September 14–16, 2020. Springer, Heidelberg, Germany. 6

[GM00] Juan A. Garay and Philip D. MacKenzie. Concurrent oblivious transfer. In *41st Annual Symposium on Foundations of Computer Science*, pages 314–324, Redondo Beach, CA, USA, November 12–14, 2000. IEEE Computer Society Press. 9

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *17th Annual ACM Symposium on Theory of Computing*, pages 291–304, Providence, RI, USA, May 6–8, 1985. ACM Press. 2

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989. 2, 89

[GMR01] Yael Gertner, Tal Malkin, and Omer Reingold. On the impossibility of basing trapdoor functions on trapdoor predicates. In *42nd Annual Symposium on Foundations of Computer Science*, pages 126–135, Las Vegas, NV, USA, October 14–17, 2001. IEEE Computer Society Press. 1

[GMW86] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design (extended abstract). In *27th Annual Symposium on Foundations of Computer Science*, pages 174–187, Toronto, Ontario, Canada, October 27–29, 1986. IEEE Computer Society Press. 6

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th Annual ACM Symposium on Theory of Computing*, pages 218–229, New York City, NY, USA, May 25–27, 1987. ACM Press. 1, 2

[GO07] Jens Groth and Rafail Ostrovsky. Cryptography in the multi-string model. In Alfred Menezes, editor, *Advances in Cryptology – CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 323–341, Santa Barbara, CA, USA, August 19–23, 2007. Springer, Heidelberg, Germany. 10

[Gol01] Oded Goldreich. *Foundations of Cryptography: Basic Tools*, volume 1. Cambridge University Press, Cambridge, UK, 2001. 13, 89, 110

[Gol04] Oded Goldreich. *Foundations of Cryptography: Basic Applications*, volume 2. Cambridge University Press, Cambridge, UK, 2004. 136

[Gol08] O. Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, 2008. 13

[Gol09] Oded Goldreich. *Foundations of cryptography: volume 2, basic applications*. Cambridge university press, 2009. 19, 20

[GOSV14] Vipul Goyal, Rafail Ostrovsky, Alessandra Scafuro, and Ivan Visconti. Black-box non-black-box zero knowledge. In David B. Shmoys, editor, *46th Annual ACM Symposium on Theory of Computing*, pages 515–524, New York, NY, USA, May 31 – June 3, 2014. ACM Press. 1, 7, 8, 9, 25, 26, 27, 28, 29, 34, 35, 36, 40, 42, 87, 108, 169

[Goy11] Vipul Goyal. Constant round non-malleable protocols using one way functions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd Annual ACM Symposium on Theory of Computing*, pages 695–704, San Jose, CA, USA, June 6–8, 2011. ACM Press. 3, 6

[Goy12] Vipul Goyal. Positive results for concurrently secure computation in the plain model. In *53rd Annual Symposium on Foundations of Computer Science*, pages 41–50, New Brunswick, NJ, USA, October 20–23, 2012. IEEE Computer Society Press. 24

[GPR16] Vipul Goyal, Omkant Pandey, and Silas Richelson. Textbook non-malleable commitments. In Daniel Wichs and Yishay Mansour, editors, *48th Annual ACM Symposium on Theory of Computing*, pages 1128–1141, Cambridge, MA, USA, June 18–21, 2016. ACM Press. 88, 94

[GT00] Rosario Gennaro and Luca Trevisan. Lower bounds on the efficiency of generic cryptographic constructions. In *41st Annual Symposium on Foundations of Computer Science*, pages 305–313, Redondo Beach, CA, USA, November 12–14, 2000. IEEE Computer Society Press. 1

[Hai08] Iftach Haitner. Semi-honest to malicious oblivious transfer - the black-box way. In Ran Canetti, editor, *TCC 2008: 5th Theory of Cryptography Conference*, volume 4948 of *Lecture Notes in Computer Science*, pages 412–426, San Francisco, CA, USA, March 19–21, 2008. Springer, Heidelberg, Germany. 3, 6, 41

[Haj18] Mohammad Hajiabadi. Enhancements are blackbox non-trivial: Impossibility of enhanced trapdoor permutations from standard trapdoor permutations. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018: 16th Theory of Cryptography Conference, Part I*, volume 11239 of *Lecture Notes in Computer Science*, pages 448–475, Panaji, India, November 11–14, 2018. Springer, Heidelberg, Germany. 1

[HHNZ19] Marcella Hastings, Brett Hemenway, Daniel Noble, and Steve Zdancewic. Sok: General purpose compilers for secure multi-party computation. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1220–1237. IEEE, 2019. 1

[HIK$^+$11] Iftach Haitner, Yuval Ishai, Eyal Kushilevitz, Yehuda Lindell, and Erez Petrank. Black-box constructions of protocols for secure computation. *SIAM Journal on Computing*, 40(2):225–266, 2011. 25, 48

[HILL99a] Johan Håstad, Russell Impagliazzo, Leonid A Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999. 10

[HILL99b] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999. 48

[HL10] Carmit Hazay and Yehuda Lindell. *Efficient secure two-party protocols: Techniques and constructions*. Springer Science & Business Media, 2010. 19

[Hof11] Dennis Hofheinz. Possibility and impossibility results for selective decommitments. *J. Cryptol.*, 24(3):470–516, 2011. 36

[HR04] Chun-Yuan Hsiao and Leonid Reyzin. Finding collisions on a public road, or do secure hash functions need secret coins? In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 92–105, Santa Barbara, CA, USA, August 15–19, 2004. Springer, Heidelberg, Germany. 1, 121

[HR07] Iftach Haitner and Omer Reingold. Statistically-hiding commitment from any one-way function. In David S. Johnson and Uriel Feige, editors, *39th Annual ACM Symposium on Theory of Computing*, pages 1–10, San Diego, CA, USA, June 11–13, 2007. ACM Press. 16

[HV15] Carmit Hazay and Muthuramakrishnan Venkitasubramaniam. On black-box complexity of universally composable security in the CRS model. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology – ASIACRYPT 2015, Part II*, volume 9453 of *Lecture Notes in Computer Science*, pages 183–209, Auckland, New Zealand, November 30 – December 3, 2015. Springer, Heidelberg, Germany. 5

[HV16a] Carmit Hazay and Muthuramakrishnan Venkitasubramaniam. Composable adaptive secure protocols without setup under polytime assumptions. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B: 14th Theory of Cryptography Conference, Part I*, volume 9985 of *Lecture Notes in Computer Science*, pages 400–432, Beijing, China, October 31 – November 3, 2016. Springer, Heidelberg, Germany. 8, 9

[HV16b] Carmit Hazay and Muthuramakrishnan Venkitasubramaniam. On the power of secure two-party computation. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 397–429, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany. 6

[HV18] Carmit Hazay and Muthuramakrishnan Venkitasubramaniam. Round-optimal fully black-box zero-knowledge arguments from one-way permutations. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018: 16th Theory of Cryptography Conference, Part I*, volume 11239 of *Lecture Notes in Computer Science*, pages 263–285, Panaji, India, November 11–14, 2018. Springer, Heidelberg, Germany. 7

[IKLP06] Yuval Ishai, Eyal Kushilevitz, Yehuda Lindell, and Erez Petrank. Black-box constructions for secure computation. In Jon M. Kleinberg, editor, *38th Annual ACM Symposium on Theory of Computing*, pages 99–108, Seattle, WA, USA, May 21–23, 2006. ACM Press. 3, 6

[IKOS07] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In David S. Johnson and Uriel Feige, editors, *39th Annual ACM Symposium on Theory of Computing*, pages 21–30, San Diego, CA, USA, June 11–13, 2007. ACM Press. 1, 6, 7, 12, 23, 86, 107, 108, 110, 122, 163, 168, 169

[ILL89] Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudo-random generation from one-way functions (extended abstracts). In *21st Annual ACM Symposium on Theory of Computing*, pages 12–24, Seattle, WA, USA, May 15–17, 1989. ACM Press. 10

[IPS08] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 572–591, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Heidelberg, Germany. 1, 6, 53

[IR89] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *21st Annual ACM Symposium on Theory of Computing*, pages 44–61, Seattle, WA, USA, May 15–17, 1989. ACM Press. 1, 10, 123, 126

[JP14] Abhishek Jain and Omkant Pandey. Non-malleable zero knowledge: Black-box constructions and definitional relationships. In Michel Abdalla and Roberto De Prisco, editors, *SCN 14: 9th International Conference on Security in Communication Networks*, volume 8642 of *Lecture Notes in Computer Science*, pages 435–454, Amalfi, Italy, September 3–5, 2014. Springer, Heidelberg, Germany. 87

[Kar72] Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972. 6

[Kat07] Jonathan Katz. Universally composable multi-party computation using tamper-proof hardware. In Moni Naor, editor, *Advances in Cryptology – EUROCRYPT 2007*, volume 4515 of *Lecture Notes in Computer Science*, pages 115–128, Barcelona, Spain, May 20–24, 2007. Springer, Heidelberg, Germany. 10

[Kil88] Joe Kilian. Founding cryptography on oblivious transfer. In *20th Annual ACM Symposium on Theory of Computing*, pages 20–31, Chicago, IL, USA, May 2–4, 1988. ACM Press. 6

[Kiy14] Susumu Kiyoshima. Round-efficient black-box construction of composable multi-party computation. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part II*, volume 8617 of *Lecture Notes in Computer Science*, pages 351–368, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany. 5, 6, 9, 36, 86, 113, 114

[Kiy20] Susumu Kiyoshima. Round-optimal black-box commit-and-prove with succinct communication. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020, Part II*, volume 12171 of *Lecture Notes in Computer Science*, pages 533–561, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Heidelberg, Germany. 7

[KLP05] Yael Tauman Kalai, Yehuda Lindell, and Manoj Prabhakaran. Concurrent general composition of secure protocols in the timing model. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 644–653, Baltimore, MA, USA, May 22–24, 2005. ACM Press. 10

[KLV17] Susumu Kiyoshima, Huijia Lin, and Muthuramakrishnan Venkitasubramaniam. A unified approach to constructing black-box UC protocols in trusted setup models. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part I*, volume 10677 of *Lecture Notes in Computer Science*, pages 776–809, Baltimore, MD, USA, November 12–15, 2017. Springer, Heidelberg, Germany. 10

[KMO14] Susumu Kiyoshima, Yoshifumi Manabe, and Tatsuaki Okamoto. Constant-round black-box construction of composable multi-party computation protocol. In Yehuda Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, pages 343–367, San Diego, CA, USA, February 24–26, 2014. Springer, Heidelberg, Germany. 8

[KO04] Jonathan Katz and Rafail Ostrovsky. Round-optimal secure two-party computation. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 335–354, Santa Barbara, CA, USA, August 15–19, 2004. Springer, Heidelberg, Germany. 3

[KOS18] Dakshita Khurana, Rafail Ostrovsky, and Akshayaram Srinivasan. Round optimal black-box "commit-and-prove". In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018: 16th Theory of Cryptography Conference, Part I*, volume 11239 of *Lecture Notes in Computer Science*, pages 286–313, Panaji, India, November 11–14, 2018. Springer, Heidelberg, Germany. 1, 7, 12, 87, 107

[Lep19] Tancrède Lepoint. Personal communication, 2019. viii

[Lev73] Leonid Anatolevich Levin. Universal sequential search problems. *Problemy peredachi informatsii*, 9(3):115–116, 1973. 6

[Lin03a] Yehuda Lindell. Bounded-concurrent secure two-party computation without setup assumptions. In *35th Annual ACM Symposium on Theory of Computing*, pages 683–692, San Diego, CA, USA, June 9–11, 2003. ACM Press. 3, 4

[Lin03b] Yehuda Lindell. General composition and universal composability in secure multi-party computation. In *44th Annual Symposium on Foundations of Computer Science*, pages 394–403, Cambridge, MA, USA, October 11–14, 2003. IEEE Computer Society Press. 3

[Lin04] Yehuda Lindell. Lower bounds for concurrent self composition. In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 203–222, Cambridge, MA, USA, February 19–21, 2004. Springer, Heidelberg, Germany. 3, 31

[Lin12] Huijia Rachel Lin. *Concurrent Security.* PhD thesis, Cornell University, 2012. 89

[Lin13] Yehuda Lindell. A note on constant-round zero-knowledge proofs of knowledge. *Journal of Cryptology*, 26(4):638–654, October 2013. 36, 109

[Lin16] Yehuda Lindell. How to simulate it - A tutorial on the simulation proof technique. Cryptology ePrint Archive, Report 2016/046, 2016. `https://eprint.iacr.org/2016/046`. 19

[LP09] Huijia Lin and Rafael Pass. Non-malleability amplification. In Michael Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 189–198, Bethesda, MD, USA, May 31 – June 2, 2009. ACM Press. 8, 30, 88

[LP11] Huijia Lin and Rafael Pass. Concurrent non-malleable zero knowledge with adaptive inputs. In Yuval Ishai, editor, *TCC 2011: 8th Theory of Cryptography Conference*, volume 6597 of *Lecture Notes in Computer Science*, pages 274–292, Providence, RI, USA, March 28–30, 2011. Springer, Heidelberg, Germany. 86

[LP12] Huijia Lin and Rafael Pass. Black-box constructions of composable protocols without set-up. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 461–478, Santa Barbara, CA, USA, August 19–23, 2012. Springer, Heidelberg, Germany. 5, 6, 9, 25, 49, 51, 86, 113, 114

[LPTV10] Huijia Lin, Rafael Pass, Wei-Lung Dustin Tseng, and Muthuramakrishnan Venkitasubramaniam. Concurrent non-malleable zero knowledge proofs. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 429–446, Santa Barbara, CA, USA, August 15–19, 2010. Springer, Heidelberg, Germany. 86

[LPV08] Huijia Lin, Rafael Pass, and Muthuramakrishnan Venkitasubramaniam. Concurrent non-malleable commitments from any one-way function. In Ran Canetti, editor, *TCC 2008: 5th Theory of Cryptography Conference*, volume 4948 of *Lecture Notes in Computer Science*, pages 571–588, San Francisco, CA, USA, March 19–21, 2008. Springer, Heidelberg, Germany. 30, 85

[LPV09] Huijia Lin, Rafael Pass, and Muthuramakrishnan Venkitasubramaniam. A unified framework for concurrent security: universal composability from stand-alone non-malleability. In Michael Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 179–188, Bethesda, MD, USA, May 31 – June 2, 2009. ACM Press. 10, 36

[Mer90] Ralph C. Merkle. A certified digital signature. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 218–238, Santa Barbara, CA, USA, August 20–24, 1990. Springer, Heidelberg, Germany. 154, 155, 156

[MMY06] Tal Malkin, Ryan Moriarty, and Nikolai Yakovenko. Generalized environmental security from number theoretic assumptions. In Shai Halevi and Tal Rabin, editors, *TCC 2006: 3rd Theory of Cryptography Conference*, volume 3876 of *Lecture Notes in Computer Science*, pages 343–359, New York, NY, USA, March 4–7, 2006. Springer, Heidelberg, Germany. 5, 93

[MOSV06] Daniele Micciancio, Shien Jin Ong, Amit Sahai, and Salil P. Vadhan. Concurrent zero knowledge without complexity assumptions. In Shai Halevi and Tal Rabin, editors, *TCC 2006: 3rd Theory of Cryptography Conference*, volume 3876 of *Lecture Notes in Computer Science*, pages 1–20, New York, NY, USA, March 4–7, 2006. Springer, Heidelberg, Germany. 16, 113

[MPR06] Silvio Micali, Rafael Pass, and Alon Rosen. Input-indistinguishable computation. In *47th Annual Symposium on Foundations of Computer Science*, pages

367–378, Berkeley, CA, USA, October 21–24, 2006. IEEE Computer Society Press. 4, 9

[MR92] Silvio Micali and Phillip Rogaway. Secure computation (abstract). In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 392–404, Santa Barbara, CA, USA, August 11–15, 1992. Springer, Heidelberg, Germany. 31

[MR04] Silvio Micali and Leonid Reyzin. Physically observable cryptography (extended abstract). In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 278–296, Cambridge, MA, USA, February 19–21, 2004. Springer, Heidelberg, Germany. 10

[Nao90] Moni Naor. Bit commitment using pseudo-randomness. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 128–136, Santa Barbara, CA, USA, August 20–24, 1990. Springer, Heidelberg, Germany. 16, 94, 108, 120, 121, 164

[Nao91] Moni Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, January 1991. 48

[NOVY98] Moni Naor, Rafail Ostrovsky, Ramarathnam Venkatesan, and Moti Yung. Perfect zero-knowledge arguments for NP using any one-way permutation. *Journal of Cryptology*, 11(2):87–108, March 1998. 16

[NY89] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *21st Annual ACM Symposium on Theory of Computing*, pages 33–43, Seattle, WA, USA, May 15–17, 1989. ACM Press. 16

[OPV10] Rafail Ostrovsky, Omkant Pandey, and Ivan Visconti. Efficiency preserving transformations for concurrent non-malleable zero knowledge. In Daniele Micciancio, editor, *TCC 2010: 7th Theory of Cryptography Conference*, volume 5978 of *Lecture Notes in Computer Science*, pages 535–552, Zurich, Switzerland, February 9–11, 2010. Springer, Heidelberg, Germany. 86

[ORS15] Rafail Ostrovsky, Silas Richelson, and Alessandra Scafuro. Round-optimal black-box two-party computation. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 339–358, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany. 3

[OSV15] Rafail Ostrovsky, Alessandra Scafuro, and Muthuramakrishnan Venkitasubramaniam. Resettably sound zero-knowledge arguments from OWFs - the (semi) black-box way. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part I*, volume 9014 of *Lecture Notes in Computer Science*, pages 345–374, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany. 9, 24, 28

[Pas03] Rafael Pass. Simulation in quasi-polynomial time, and its application to protocol composition. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 160–176, Warsaw, Poland, May 4–8, 2003. Springer, Heidelberg, Germany. 4, 5, 93

[Pas04a] Rafael Pass. Alternative variants of zero-knowledge proofs. Master's thesis, Royal Institute of Technology, Stockholm, Sweden, 2004. 14

[Pas04b] Rafael Pass. Bounded-concurrent secure multi-party computation with a dishonest majority. In László Babai, editor, *36th Annual ACM Symposium on Theory of Computing*, pages 232–241, Chicago, IL, USA, June 13–16, 2004. ACM Press. 4, 7, 8, 9, 25, 29, 31

[PLV12] Rafael Pass, Huijia Lin, and Muthuramakrishnan Venkitasubramaniam. A unified framework for UC from only OT. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology – ASIACRYPT 2012*, volume 7658 of *Lecture Notes in Computer Science*, pages 699–717, Beijing, China, December 2–6, 2012. Springer, Heidelberg, Germany. 8, 10

[PR03] Rafael Pass and Alon Rosen. Bounded-concurrent secure two-party computation in a constant number of rounds. In *44th Annual Symposium on Foundations of Computer Science*, pages 404–415, Cambridge, MA, USA, October 11–14, 2003. IEEE Computer Society Press. 4, 29, 36

[PRS02] Manoj Prabhakaran, Alon Rosen, and Amit Sahai. Concurrent zero knowledge with logarithmic round-complexity. In *43rd Annual Symposium on Foundations of Computer Science*, pages 366–375, Vancouver, BC, Canada, November 16–19, 2002. IEEE Computer Society Press. 16

[PS04] Manoj Prabhakaran and Amit Sahai. New notions of security: Achieving universal composability without trusted setup. In László Babai, editor, *36th Annual ACM Symposium on Theory of Computing*, pages 242–251, Chicago, IL, USA, June 13–16, 2004. ACM Press. 4, 5, 93

[PW08] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In Richard E. Ladner and Cynthia Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 187–196, Victoria, BC, Canada, May 17–20, 2008. ACM Press. 9

[PW09] Rafael Pass and Hoeteck Wee. Black-box constructions of two-party protocols from one-way functions. In Omer Reingold, editor, *TCC 2009: 6th Theory of Cryptography Conference*, volume 5444 of *Lecture Notes in Computer Science*, pages 403–418. Springer, Heidelberg, Germany, March 15–17, 2009. 3, 6, 16, 36, 48

[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, Prague, Czech Republic, May 2–6, 1999. Springer, Heidelberg, Germany. 86

[Ros04] Alon Rosen. A note on constant-round zero-knowledge proofs for NP. In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 191–202, Cambridge, MA, USA, February 19–21, 2004. Springer, Heidelberg, Germany. 16

[Ros12] Mike Rosulek. Must you know the code of f to securely compute f? In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 87–104, Santa Barbara, CA, USA, August 19–23, 2012. Springer, Heidelberg, Germany. 6, 7, 10, 115, 123, 124

[RTV04] Omer Reingold, Luca Trevisan, and Salil P. Vadhan. Notions of reducibility between cryptographic primitives. In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 1–20, Cambridge, MA, USA, February 19–21, 2004. Springer, Heidelberg, Germany. 1, 10, 123

[Sim98] Daniel R. Simon. Finding collisions on a one-way street: Can secure hash functions be based on general assumptions? In Kaisa Nyberg, editor, *Advances in Cryptology – EUROCRYPT'98*, volume 1403 of *Lecture Notes in Computer Science*, pages 334–345, Espoo, Finland, May 31 – June 4, 1998. Springer, Heidelberg, Germany. 1

[Ven14] Muthuramakrishnan Venkitasubramaniam. On adaptively secure protocols. In Michel Abdalla and Roberto De Prisco, editors, *SCN 14: 9th International Conference on Security in Communication Networks*, volume 8642 of *Lecture Notes in Computer Science*, pages 455–475, Amalfi, Italy, September 3–5, 2014. Springer, Heidelberg, Germany. 8

[Wee10] Hoeteck Wee. Black-box, round-efficient secure computation via non-malleability amplification. In *51st Annual Symposium on Foundations of Computer Science*, pages 531–540, Las Vegas, NV, USA, October 23–26, 2010. IEEE Computer Society Press. 3, 6, 25

[Wyn75] Aaron D Wyner. The wire-tap channel. *Bell system technical journal*, 54(8):1355–1387, 1975. 1, 6

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science*, pages 162–167, Toronto, Ontario, Canada, October 27–29, 1986. IEEE Computer Society Press. 1

[Yer11] Arkady Boris Yerukhimovich. *A study of separations in cryptography: new results and new models*. PhD thesis, University of Maryland, College Park, Maryland, USA, 2011. 126