THE LIMITS OF PROVABLE EFFICIENCY IN CRYPTOGRAPHY

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Andrew Spencer Morgan

May 2022

THE LIMITS OF PROVABLE EFFICIENCY IN CRYPTOGRAPHY

Andrew Spencer Morgan, Ph.D.

Cornell University 2022

In this work, we examine the science of proving formal security of primitives in cryptography through *security reductions*, with a focus on determining the achievability of various notions of efficiency. We work towards exploring the limits of provable efficiency through both feasibility and impossibility results, and summarize our findings in two distinct areas:

1. We first examine the notion of *concrete security loss* in reductions from the unforgeability of unique digital signature schemes and related primitives to standard cryptographic assumptions. Specifically, we show that such reductions are inherently *inefficient* in the level of concrete security they demonstrate for the primitive by ruling out a particular class of security-preserving reduction: *linear-preserving* reductions. Our first result proves the nonexistence of such reductions from unique digital signature schemes to any standard assumption with a polynomially bounded number of communication rounds, while our second result proves a similar impossibility for reductions from the adaptive multi-key security of message authentication codes or pseudorandom functions. Both results are proven via the *meta-reduction* paradigm, which has been applied to prove bounds on concrete security loss in the past, but all known meta-reductions in this space prior to our results have considerable technical limitations, usually only ruling out "simple" reductions that interact with their respective adversaries non-concurrently and reduce to a limited class of *non-interactive* assumptions. In contrast, our results present a novel rewinding technique and a "randomness-switching" lemma that allow

for the first concrete security bounds via meta-reductions that are not subject to these limitations.

2. We also prove two positive results in the area of *non-interactive secure two-party computation* (NISC), both of which improve on the efficiency of currently known constructions in different ways. The first result examines avenues of reducing the communication complexity of NISC protocols: we construct a NISC protocol which satisfies security with superpolynomial-time simulation (SPS) and also a notion of *succinctness*—where the communication complexity and receiver's running time are close to independent from the functionality f—and does so by leveraging a non-succinct NISC, an adaptive delegation scheme, and fully homomorphic encryption, all of which can be based on subexponential LWE. The second result constructs the first NISC protocol satisfying a concurrent and composable notion of "angel-based" *universally composable* (UC) security, demonstrating that we can achieve the optimal possible round complexity for such a protocol. This construction is based on a stand-alone secure NISC and non-interactive CCA-secure commitments; while the latter notion is only known based on complex and non-standard assumptions, we additionally demonstrate as a matching *negative* result that these assumptions are also *necessary* for UC-secure NISC (subject to perfect correctness), thus expanding on our final positive result to provide a nearly-tight characterization of the necessary and sufficient assumptions for this definition of security.

BIOGRAPHICAL SKETCH

Andrew Morgan completed his undergraduate education in 2013 at the University of Washington in Seattle, where he earned a B.S. in Computer Science with a minor in mathematics. Between completing that degree and starting at Cornell University, he spent two years divided between obtaining experience as a software engineer at Amazon and conducting independent research into game theory collaborating with faculty at the University of Washington. Since starting at Cornell in 2016 with the goal of obtaining an M.S. and Ph.D. in Computer Science, he has attended and presented at several cryptography conferences and, in Summer 2021, undertaken a research internship with Microsoft Research's Cryptography and Privacy group.

To the last steps of one journey...

...and the first steps into a new one.

ACKNOWLEDGEMENTS

Once upon a time, a more naïve me believed that, as many say, the first steps of a journey were the most difficult. But now I know that this couldn't be more wrong: the last steps—the slow crawl to the finish amidst fading dreams, decaying friendships, and an uncaring, isolated world—made this journey without a doubt the most difficult one that I have, and maybe that I ever *will* have, undertaken. There were days when I lost sight of the finish, when I wanted to give up, or when I couldn't even bring myself to get out of bed. Even in the last month while finishing my last paper and writing this dissertation, I remember late nights spent alone in despair, in the belief that all my carefully constructed research was falling apart and that I would be forced to withdraw. For the last two years, I didn't think I could make it to the end.

And yet—despite everything—here I am.

So, at the end of this terrible yet wonderful journey, I want to acknowledge, and thank from the bottom of my heart, all the people who not only were with me through these five and a half arduous years but also never gave up on me—those who stayed with me to the finish. There were many who did give up, from whom I've parted ways, and that's okay; we've moved on, and honestly I can hardly blame them for turning away from me at my worst. But I dedicate this work to those who remained, because when all's said and done this would not have been possible without them.

I'd like to thank the few friends I still have in Ithaca, especially Claire Liang and Kate Donahue, who not only were there to show their support at my defense, but also made time to see me this past summer when I finally moved out of Ithaca. It means a lot to me that I still have friends who care about me after everything that's happened, and I'll do my best to make it back to Ithaca and have a much-belated celebration when things settle down again.

I'd like to thank my out-of-committee collaborators, Muthu Venkitasubramaniam and

Antigoni Polychroniadou, for putting up with me for the past two years. I certainly didn't make collaboration easy, but I'm hoping that we can still finish the projects we were working on as I work on putting my life back together and preparing for whatever comes next. I'd also like to thank the Microsoft Research Cryptography and Privacy group, and especially my mentors there, Melissa Chase and Radames Cruz Moreno, for the summer I spent there; the circumstances of the internship certainly weren't ideal, but I definitely took a lot away from it, and I hope that the work I produced was useful.

I'd like to thank my committee members, Robert Kleinberg and Elaine Shi, and especially my advisor, Rafael Pass, for being an outstanding committee and three of the most brilliant and insightful researchers I have had the pleasure of working with. It continues to astonish me that all of these people could spark my interest in research problems and topics in theoretical computer science even at times when I could have sworn all my motivation was gone, and it's thanks in no small part to them that I could see this undertaking through. Even outside of research itself, I owe a lot to both Robert and Rafael for giving me a much-needed push in the right direction at times when I've felt lost or despondent. I'm honored to have gone through graduate school mentored by such an amazing trio and advised by one of the most motivated, understanding, and genuinely inspiring people I've ever known, and I sincerely hope that the end of my time at Cornell isn't the end of our correspondence or collaboration.

And finally, I'd like to thank my parents, William Morgan and Sandra Kaplan, for being there for me from start to finish, and for giving me not only a place to stay but also a home at a time when I have no other. We don't always get along or see eye to eye, but it means more to me than I can express in words that they, the only close family I have, are despite everything still trying their best to support me and be there for me at times when I truly have nobody else.

It's thanks to these people that I'm here, at the end of this journey—and thanks to them that I can yet hope that there's a brighter future waiting for me. It's time to find it.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Cryptography, the study of communication in the presence of untrusted or malicious parties (*adversaries*), has perhaps unsurprisingly existed as a concept for centuries if not millenia. Recently, with the advent of computing and the consequent need for both more widespread and more secure cryptographic protocols, cryptography has undergone massive advances as a science. Modern cryptography has not only introduced a vast array of new intuitive notions of secure communication and techniques for achieving those notions heuristically, but also placed a far greater emphasis on the mathematical foundations of cryptography and rigorous proofs of security. Current foundational cryptography, through a confluence of ideas from classical cryptography, information theory, mathematics, and theoretical computer science, allows us to formally describe a vast and ever-growing variety of notions of security, devise implementations of these notions, and prove their security in the presence of computationally described adversaries.

Central to this study, of course, is the notion of what is entailed by such a proof of security. As we currently lack the means to definitively prove the existence of "computationally hard" problems in the classical sense—i.e., problems that cannot be solved by a Turing machine in time polynomial in their input size—proofs of *unconditional* security in modern cryptography are virtually nonexistent; instead, cryptographic proofs of security are formulated as *security reductions*, where we prove that the computational hardness of some underlying assumptions[1] implies the computational hardness of breaking the security of a constructed cryptosystem. Security reductions provide strong security guarantees for applied algorithms as long as the assumptions on which they are based are not broken, and in addition allow us to work

[1]These are generally problems that are widely believed to be computationally hard, such as, to give a basic example, factoring a product of large primes.

towards a theoretical characterization of the overall strength or feasibility of cryptographic primitives or problems based on, for instance, the strength of the necessary or sufficient assumptions.

In this work, we explore techniques for proving both the existence and the non-existence of security reductions, and we present a series of key results that expands upon the previous state of the art of these techniques and in so doing furthers our understanding of the inherent *efficiency* achievable for constructions of two distinct classes of cryptographic primitives. We begin by studying the notion of *security preservation* in reductions, and demonstrate strong negative results showing that reductions from unique digital signatures and related primitives have inherent inefficiency in terms of the level of concrete security they preserve. We then turn to studying the *round, communication, and computation complexity* of protocols for secure two-party computation, and present two new protocols which improve on the efficiency of prior works in these areas. The ultimate intent of this research is to further our understanding of the limits of provable efficiency (or inefficiency) of cryptographic primitives, both by our results themselves and by advancing known techniques to guide and facilitate future research.

1.1 Security Reductions

To illustrate how to formally prove that a cryptosystem is secure based on one or more underlying assumptions, we begin by formally modeling security assumptions as "interactive security games" [103, 110]. Specifically, we can think of an assumption as a combination of an interactive Turing machine, or *challenger*, $\mathcal{C}$—which will either accept or reject after an interaction—and a "threshold" function $t(n)$, such that a probabilistic Turing machine *adversary* $\mathcal{A}$ is considered to be able to "break", or invalidate, the assumption if they are

able to interact with $\mathcal{C}$ and cause it to accept with probability significantly higher than $t(n)$.[2]

For instance, we could model the assumption that factoring products of large primes is a computationally hard problem using a challenger that, given some input size n (usually referred to as a *security parameter*), sends a product of two n-bit primes to $\mathcal{A}$ and accepts if and only if $\mathcal{A}$ returns the correct factoring of the product. Then, if there is no polynomial-time adversary $\mathcal{A}$ able to cause $\mathcal{C}$ to accept with probability significantly better than would be obtained via randomly guessing—specifically, "non-negligible" probability $1/p(n)$ for some polynomial p—we can consider the assumption to be "unbreakable" or secure.

This notion of a game-based assumption captures all standard cryptographic assumptions, ranging from the computational hardness of basic problems such as the above example to the security of complex primitives and algorithms. A security reduction, then, shows that one assumption Π (usually a primitive or cryptosystem) being secure is implied by a second assumption $\mathcal{C}$ being secure—or, by contrapositive, that if there exists an adversary $\mathcal{A}$ which can break Π, then there must also exist an adversary $\mathcal{A}'$ able to break $\mathcal{C}$. We call this *a reduction from the security of* Π *to the security of* $\mathcal{C}$.

Black-box reductions. In any meaningful reduction, the adversary $\mathcal{A}'$ constructed to break the underlying assumption $\mathcal{C}$ relies on the fact that the assumed adversary $\mathcal{A}$ breaks the security of the implied assumption Π. For most (but not all) security reductions in existence, that fact is sufficient, and $\mathcal{A}'$ can simply interact with A as though it were a "black box"—we refer to such a reduction as a *black-box reduction*. We can model black-box reductions as a separate interactive Turing machine $\mathcal{R}$ such that $\mathcal{R}^{\mathcal{A}}$ (denoting $\mathcal{R}$ given black-box access to $\mathcal{A}$) breaks $\mathcal{C}$. While there do exist reductions that are non-black-box in that

[2]When we mention "success probability" or "probability with which a reduction breaks $\mathcal{C}$", we refer to the difference between the actual probability and the threshold, as the threshold is usually the baseline probability with which $\mathcal{C}$ would accept if the adversary were to guess randomly.

the constructed adversary $\mathcal{A}'$ depends on the structure or implementation of the adversary $\mathcal{A}$, we remark that these are very rare and restrict our focus to black-box reductions in this work.

Efficiency and security loss. Often with security reductions we are content to prove *asymptotic* security—for instance, that the underlying assumption being secure against all polynomial-time adversaries implies that the primitive for which we prove security is secure against polynomial-time adversaries as well. This preserves security in a "polynomial" sense, but may often be unsatisfying from a practical standpoint if the reduction is "inefficient" compared to the adversary it uses as a black box. In particular, we can define the *security loss* [92], or inherent "inefficiency", of a reduction as the ratio between the "work" done by the reduction to break the underlying assumption and that done by the adversary to break the primitive, quantifying "work" as the ratio between a machine's running time and the probability with which it breaks an assumption. Then, the higher the security loss of a reduction, the easier the primitive will be to break compared to the underlying assumption in a concrete sense.

When designing algorithms for practical use, minimizing the security loss of a reduction is important for maximizing the provable security of the corresponding construction. In particular, as shown in works such as [102, 125], minimizing the dependence of a reduction's security loss on additional variables, such as dependence on the number of simultaneous instances in use for a primitive intended to be widely used, is critical for achieving a strong practical security guarantee. The best possible guarantees for concrete security are provided by *tight* reductions [92], or those with a constant-factor security loss, guaranteeing that the primitive inherits a level of security very close to that of the assumptions on which it is based and avoiding additional dependencies. There is a considerable amount of research

concerning tight and security-preserving reductions for a wide variety of primitives [20, 104, 105, 16, 18, 8, 9].

We formalize all of the definitions given in this section in Section 2.2.

1.2 Impossibility Results

Having discussed black-box reductions, we next turn to a discussion of methods for proving the *unprovability* of certain reductions or classes of reductions. While perhaps unsatisfying from a practical standpoint, impossibility results are of considerable theoretical interest as a means of "separating" cryptographic primitives and providing lower bounds on the strength of assumptions sufficient to prove a reduction from a primitive. These results can serve to not only direct the course of further research into positive results, but also to help characterize the relative strength of assumptions or primitives; in certain cases, an impossibility result with a matching feasibility result can provide a tight characterization and demonstrate that certain assumptions are both necessary *and* sufficient for a particular primitive.

The earliest results proving the impossibility of black-box reductions were proven with a paradigm known as *oracle separation* [78], which has since been used to prove many separations and impossibilities (e.g., [124, 77, 72, 73, 84, 5, 6]). At a high level, to prove impossibility of a reduction from a primitive Π to an assumption $\mathcal{C}$, one would begin by constructing an oracle $\mathcal{O}$ such that Π is *insecure* relative to $\mathcal{O}$—that is, there exists an adversary $\mathcal{A}$ that can break Π using $\mathcal{O}$—but also such that there exists an instantiation[3] $\mathcal{C}^*$ of $\mathcal{C}$ that can be proven unbreakable by even adversaries with access to $\mathcal{O}$. If this is the case, then given a hypothetical reduction $\mathcal{R}$ which uses $\mathcal{A}$ (and consequently the oracle $\mathcal{O}$) as a

[3]For instance, if $\mathcal{C}$ were the assumption that there existed secure families of one-way permutations, $\mathcal{C}^*$ might be a specific family of one-way permutations.

black box to break $\mathcal{C}$, then it should be able to break the specific instance $\mathcal{C}^*$ as well, a clear contradiction to $\mathcal{C}^*$ being unbreakable by adversaries with access to the oracle.

This approach is extremely general and flexible, hence its relative popularity; however, it has one key limitation in that it relies on Π being a *black-box construction*—that it must use the underlying assumption $\mathcal{C}$ as a black box independently of its implementation. While non-black-box *reductions* are rare, non-black-box constructions, such as those which might be based on a specific instantiation of a primitive and thus bypass an oracle separation relying on a different one, are far more common.

1.2.1 Meta-Reductions

The other widely-used technique for proving the impossibility of black-box reductions is known as the *meta-reduction* paradigm [25], and has the advantage that it can in theory be applied to rule out far more general classes of black-box reductions than oracle separations. While earlier meta-reductions [106, 107, 32, 74, 53, 2] ruled out reductions to very specific assumptions or classes of assumptions, later meta-reductions [59, 110, 13, 111] began to offer far more generality, ruling out reductions to much broader classes of assumptions, such as any bounded-round assumption or any falsifiable assumption. In practice, though, it requires considerable finesse to achieve this level of generality for reasons that we will discuss shortly.

To prove impossibility via a meta-reduction, one begins by constructing a "perfect" adversary $\mathcal{A}$ that breaks the primitive Π with probability 1, but generally does so inefficiently (e.g., by brute force search). Then, if we had a hypothetical black-box reduction $\mathcal{R}$ from the security of Π to an underlying assumption $\mathcal{C}$, $\mathcal{R}$ could use $\mathcal{A}$ to break the security of $\mathcal{C}$, albeit also inefficiently. The crux of the proof, then, is to produce a *meta-reduction* $\mathcal{B}$ which

is able to efficiently emulate the interaction between $\mathcal{R}$ and $\mathcal{A}$, generally by exploiting the structure of the interaction to efficiently "extract" information from $\mathcal{R}$ on behalf of $\mathcal{A}$ and thus circumvent needing to use brute force. If $\mathcal{B}$ emulates the interaction successfully with probability 1 (or close to probability 1), then it must break $\mathcal{C}$ roughly as often as $\mathcal{R}$ aided by $\mathcal{A}$ would, thus efficiently breaking $\mathcal{C}$ and demonstrating that, if $\mathcal{C}$ is a secure assumption in the first place, such a reduction $\mathcal{R}$ cannot exist.

Meta-reductions and efficiency bounds. An additional feature of meta-reductions, and one which will be highlighted by the results we are about to present, is the ability to prove a "partial" impossibility, or impossibility for reductions $\mathcal{R}$ having at least a certain probability of success $p(n)$. Specifically, if $\mathcal{B}$ emulates the interaction between $\mathcal{A}$ and the hypothetical reduction $\mathcal{R}$ with probability $1-p(n)$, and $\mathcal{R}$ when using the "perfect" adversary $\mathcal{A}$ to break $\mathcal{C}$ has probability of success $q(n)$ which is significantly (non-negligibly) higher than $p(n)$, then it follows that $\mathcal{B}$ will still have probability $q(n) - p(n)$ of breaking $\mathcal{C}$, thus still ruling out such a reduction $\mathcal{R}$. This type of bound on the success probability of a reduction can be used to prove lower bounds on security loss and rule out *efficient* classes of reductions. A considerable number of examples of meta-reductions proving concrete security bounds exist in literature [42, 106, 54, 82, 76, 122, 9, 7, 81, 126]; however, all of them deal with very limited classes of reductions relative to full meta-reductions such as [59, 110, 13, 111].

1.2.2 Limitations of Meta-Reductions

The difficulty in constructing and proving meta-reductions to rule out broad classes of reductions lies in the aforementioned fact that the meta-reduction must emulate the interaction between the adversary and a hypothetical reduction and usually does so by exploiting the

structure of the reduction. Specifically, virtually all meta-reductions work by *rewinding* the interaction between the adversary and the reduction (which can be done since the interaction is emulated internally) in order to attempt to extract the answer to a later "challenge" query (which determines whether security is broken) from the reduction via the results of queries earlier in the interaction. (We will explain this in more detail when we present our results.) However, reductions in general may have fairly complicated structures and interactions with the adversaries and challengers they communicate with, including some complications which interact very badly with rewinding, so the vast majority of full impossibility meta-reductions (and *all* concrete security meta-reductions prior to the ones we present here) have restricted their scope to consider only reductions and underlying assumptions that have simpler structures for technical reasons. We discuss two of the most common restrictions here.

Straight-line ("simple") reductions. In general, reductions may choose to run many instances of the adversary *concurrently*, and potentially may even choose to rewind or reset instances mid-execution. This presents an issue because the structure of such reductions can be highly dependent on the inputs they receive; hence, when a meta-reduction rewinds such a reduction in order to try different queries, the order in which the reduction makes its queries and invokes one or more instances of the adversary might dramatically change. A particularly troubling possibility is the case of "nesting", where during a rewound query made by the meta-reduction, the reduction could decide to start and finish an entirely different instance of the adversary, thus requiring the meta-reduction to respond to the challenge query of the internal instance by rewinding *that* instance, and potentially repeating recursively to the point where the running time of the meta-reduction could be increased exponentially by the rewinding process, thus invalidating the impossibility result as it no longer breaks the target assumption efficiently. Hence, it simplifies the process of constructing a meta-

reduction greatly to consider "simple" reductions which cannot run nested invocations of or rewind their respective adversaries; restricting to simple reductions ensures that rewinding preserves the structure of the reduction and thus greatly simplifies the analysis of both the success probability and running time of the reduction.

Non-interactive (two-round) assumptions. Another key issue with rewinding is that the structure of the interaction between a reduction and the challenger $\mathcal{C}$ for the underlying assumption might differ depending on rewinding and the adversary's inputs. Even if it does not change, rewinding a reduction that interacts with a challenger is potentially problematic because *the meta-reduction cannot rewind the challenger* as it can the internally emulated reduction. Hence, if the rewinding causes the reduction to send a message to the challenger and require a response before proceeding, it is impossible to properly emulate that. For this reason, many meta-reductions in literature deal with *non-interactive* assumptions whose respective challengers only send a single message at the beginning and receive a message at the end before accepting or rejecting; restricting to ruling out only reductions to non-interactive assumptions effectively removes the above problem by ensuring that no communication with the challenger can happen during the meta-reduction's emulation.

Bypassing the limitations. For the case of full impossibilities, [59, 110] were the first works to create meta-reductions featuring analysis that accounted for the above issues rather than restricting to cases where they could not occur. Specifically, noting that the issue of nested rewinding was already apparent in works concerning concurrent zero-knowledge [50], [110] presented a recursive rewinding strategy inspired by similar strategies in concurrent zero-knowledge research [119, 114, 44, 38] which, at a high level, rewinds recursively as normal but aborts when a certain limit of nested queries is reached, additionally reducing

this limit by a factor of n on every recursive level to ensure that the number of recursive levels is bounded by a constant and the running time of the meta-reduction thus remains polynomial. Advanced rewinding strategies such as this enabled meta-reductions that proved full impossibilities for *any* reductions (with an a priori polynomially bounded number of rounds of communication) and—since rewinding could likewise be aborted in the case of external communication with the challenger—for reductions to interactive assumptions as well.

However, for the case of meta-reductions as concrete security bounds, there was a key issue preventing such strategies from working. Specifically, if the meta-reduction has some probability for emulation to *fail* and not generate a correct response to a challenge query, this failure can occur not only in top-level emulation but also in the potentially recursive layers of rewinding. For the case of full impossibilities, where emulation succeeds with probability 1 (or extremely close to 1), the failure probability compounding over a polynomially bounded number of rewindings is not an issue; however, for the case of concrete security bounds, where, as we have explained, emulation fails with a significant (usually inverse polynomial) probability, this compounding can effectively reduce the success probability of the overall meta-reduction to zero. Hence, the problem of developing rewinding strategies that can allow meta-reductions for partial impossibilities to apply to general classes of reductions and assumptions has remained unsolved until recently.

1.3 Our Results: Impossibilities for Linear-Preserving Reductions

In Chapter 2, we present two results (fully published as [98] and [101]) that examine the concrete security loss of reductions from two classes of widely used cryptographic primi-

tives and show via meta-reduction that certain classes of efficient reductions—namely *tight* or *linear-preserving* reductions—are impossible. The first result, presented in Section 2.3, proves the impossibility of linear-preserving reductions from the unforgeability of unique digital signatures to any bounded-round assumption. The second result, presented in Section 2.4, proves the impossibility of linear-preserving reductions from the adaptive multi-user security of message authentication codes to any bounded-round assumption. Both results rule out rewinding or concurrent reductions to interactive assumptions, furthering the study of meta-reductions as used to prove concrete security bounds by circumventing the limitations in prior such works.

1.3.1 Linear-Preserving Reductions From Unique Signatures

Digital signatures are a primitive analogous to physical signatures: a signer can use a secret key to "sign" a message in such a way that the authenticity of the signature can be verified publicly by anyone with a corresponding public key, but an adversary without possession of the secret key is unable to forge a signature for a message, even after seeing valid signatures of potentially many other messages. More formally, the challenger $\mathcal{C}$ for unforgeability of a digital signature scheme generates a public key and a secret key, sends the public key to the adversary $\mathcal{A}$ and subsequently lets $\mathcal{A}$ send it any number of messages m to sign, responding to any such query with a signature σ generated using its secret key. Finally, in order for $\mathcal{C}$ to accept, $\mathcal{A}$ must send it a message and signature (m^*, σ^*) such that σ^* is a valid signature of m^* but the adversary has not previously asked for a signature of m^*.

The earliest constructions of digital signatures [120, 116, 123] were heuristically analyzed, but many provably secure constructions are known and used today, starting with [68]. In fact, many constructions are known that are provably secure via tight reductions to a variety

of underlying assumptions [20, 23, 1, 41, 40]. However, a commonly desired feature (see, e.g., [71, 113]) in signature schemes which none of these tightly secure constructions possess is a *uniqueness* property where there is a unique valid signature for any given message under a certain public key. There exist several constructions of unique signatures [95, 93] (see also [28, 80] for constructions of verifiable random functions, which imply unique signatures), but none of them have tightly secure reductions. In fact, none of these constructions even have reductions which are *linear-preserving* [92] in that they have security loss which is strictly polynomial in the security parameter n (and notably independent of additional variables such as, for instance, the number of signing queries made by the adversary).

Towards answering the question of whether this separation is inherent, Jean-Sebastien Coron [42] proved by a meta-reduction that for the specific case of simple reductions to non-interactive assumptions, dependence of the reduction's security loss on the number of signing queries (which we shall denote by ℓ) is inevitable and thus linear-preserving reductions are impossible. To explain how this meta-reduction works, consider an ideal adversary $\mathcal{A}$ that queries the challenger for ℓ signatures on random messages, selects another random message m^*, and signs it by brute force to break unforgeability. Then the meta-reduction $\mathcal{B}$ will internally run $\mathcal{R}$ and emulate $\mathcal{A}$ by making ℓ random queries as before; however, to attempt to find a valid forgery, $\mathcal{B}$ needs to rewind to attempt to extract the answer from $\mathcal{R}$. Specifically, $\mathcal{B}$ will pick a random forgery target m^* and start by rewinding the interaction to a randomly selected and querying $\mathcal{R}$ on m^*. If $\mathcal{R}$ gives a valid signature for m^*, then $\mathcal{B}$ has successfully extracted a forgery and can use that in the top-level (non-rewound) interaction to successfully emulate $\mathcal{A}$'s brute-force[4]; if it fails to provide a valid signature, then $\mathcal{B}$ will simply abort. Note that if $\mathcal{R}$ fails to provide valid responses to $\mathcal{A}$'s queries (or $\mathcal{B}$'s queries in the non-rewound execution), $\mathcal{A}$ (or $\mathcal{B}$) is not required to provide a forgery and may simply

[4]Note that this is where uniqueness is critical: without such a guarantee, we have no way of knowing whether the responses to queries given by $\mathcal{R}$ are distributed identically to the forgeries brute-forced by $\mathcal{A}$.

return the special symbol $\perp$ instead.

In this way, the meta-reduction $\mathcal{B}$ emulates the ideal adversary $\mathcal{A}$ by relying on the structure of the reduction $\mathcal{R}$. Coron then shows that $\mathcal{B}$ succeeds in this emulation with probability $1 - O(1)/\ell$—to give high-level intuition, this is because if $\mathcal{R}$ responds to fewer than a $1 - O(1)/\ell$ fraction of random signature queries (e.g., the rewound query to extract the forgery), then it is overwhelmingly likely that $\mathcal{R}$ will return an incorrect response to one of the queries in the execution of $\mathcal{B}$ before rewinding, thus causing a $\perp$ response. Given that, we can conclude that the meta-reduction rules out any reduction $\mathcal{R}$ running a single instance of $\mathcal{A}$ with a success probability of greater than $O(1)/\ell$—hence, a security loss of at least $O(\ell)$ when compared to the adversary with a success probability of 1. Even with a reduction $\mathcal{R}$ that runs M (non-concurrent) instances of $\mathcal{A}$, we obtain a success probability bound of M/ℓ by the union bound over instances, again giving us $O(\ell)$ security loss as the running time of $\mathcal{R}$ is now at least M times that of $\mathcal{A}$.

Our result: A new rewinding strategy. The result we present here builds considerably on both the construction and analysis of Coron's meta-reduction to prove a similar security loss bound for not only general reductions (admitting nesting and rewinding) but also reductions to any interactive assumptions with an a priori bounded number of rounds of communication[5]. Since, as we discussed, recursive rewinding such as that used in [110] is out of the question, we instead use an ostensibly far simpler strategy, inspired by similar techniques used for "bounded-concurrent" zero-knowledge arguments [90]. We begin by increasing the number of times $\mathcal{B}$ will attempt to extract a forgery: rather than selecting a single random query to rewind to as with Coron's meta-reduction, we instead start by

[5]We note that unforgeability assumptions, such as the security of unique signatures, do not fall into this category—this restriction is inherent as otherwise we would rule out reducing the security of a signature scheme to itself!

attempting to extract a forgery from the first query; if that fails, we try extracting a forgery by rewinding to the second query, and so on and so forth until either a forgery has been found or all ℓ queries have failed to extract.

We will need these additional rewindings to account for the possibility of concurrent reductions and interactive assumptions. Observing that we only ever need to rewind the meta-reduction $\mathcal{B}$ when we encounter an "end message"—that is, when $\mathcal{A}$ (or its emulation) must return a forgery to the reduction $\mathcal{R}$, we observe that the problematic rewindings occur during a rewound query where either (1) such an end message would occur for a separate instance of $\mathcal{A}$ before $\mathcal{R}$ responds, or (2) communication between $\mathcal{R}$ and $\mathcal{C}$ would happen before $\mathcal{R}$ responds (since we admit interactive assumptions $\mathcal{C}$, which as mentioned before we cannot rewind). Instead of attempting to rewind recursively, we simply abort the rewinding when this would happen and continue attempting to extract a forgery with the next query. This technique may seem at first glance like a fairly unrefined way to circumvent the issues at hand, but given the impossibility of rewinding $\mathcal{C}$ and the fact that recursive rewinding compounds the reduction's failure probability significantly, it bears the most thought to refine the analysis, rather than the construction, of the meta-reduction.

And indeed, the meta-reduction we present has a highly non-trivial analysis, especially compared to the relatively simple analysis of its counterparts in concurrent zero knowledge. In particular, we cannot rely on any sort of independence between rewindings as one might in zero-knowledge rewinding; if we queried a random m^* with each separate rewinding, the distribution of forgeries found by $\mathcal{B}$ would be biased based on which m^* $\mathcal{R}$ correctly responded to, which would be quite problematic as $\mathcal{B}$ needs to emulate the random forgery output by $\mathcal{A}$. So instead we must select a random forgery target m^* only once and query the same m^* every rewinding, creating a strong *dependence* between rewindings.

So, rather than using a more standard probabilistic analysis, we must instead rely on a "randomness switching" argument where we demonstrate that any "bad" sequence of queries (i.e., the ℓ queries in the non-rewound execution of $\mathcal{B}$ and the forgery m^*)—where $\mathcal{B}$ fails because all ℓ of the normal queries are answered correctly by $\mathcal{R}$ but no attempted extraction of m^* is successful due to either prohibited rewinding or to an incorrect response—can be permuted into many different "good" sequences where the above does not happen and thus $\mathcal{B}$ succeeds. By doing this, we can upper-bound the number of possible "bad" sequences of randomness and thus assert that a "bad" sequence happens sufficiently rarely that $\mathcal{B}$ will succeed with high probability. Ultimately, this bound on the failure probability of $\mathcal{B}$ lets us bound the success probability and security loss of a reduction $\mathcal{R}$, just as with Coron's result. Interestingly, unlike in Coron's simplified setting, we get a weaker lower bound on the security loss—$O(\sqrt{\ell})$ instead of $O(\ell)$—which raises an interesting question of whether more efficient reductions are possible via more structurally complex reductions or whether there is a better analysis to close the gap; either way, though, we prove that even in the more general case a dependence on the number of signature queries is inevitable.

1.3.2 Linear-Preserving Reductions From Adaptively Secure MACs

Our next result will prove an analogous impossibility for a related but distinct class of primitives. Message authentication codes (MACs) are a secret-key analogue of signatures (and as such are implied by signatures)—the main difference is that a single secret key is used for both tagging ("signing") and verification. However, for this result, we will consider a stronger notion of unforgeability than the one we previously considered: specifically, we investigate *adaptive multi-key unforgeability*, where a large number of instances (secret) keys

are generated and concurrently active. As before, the adversary may make tagging queries, this time to an instance of its choice, but it may also decide at any point to "open" the secret key for any instance of its choice. Security, then, is broken if the adversary can produce a valid forged tag for an instance whose key it has not yet opened and a message on which it has not yet requested a tag from that instance.

It is fairly straightforward to see that this is a strict strengthening of the standard single-instance notion of unforgeability, though the naive reduction does incur security loss proportional to the number of active instances [17, 8]. And indeed, with their use in widespread cryptographic protocols such as the TLS protocol [45, 46, 47] for secure key exchange, MACs' multi-user security under various notions has gathered considerable attention in recent works [17, 16, 15, 19, 102, 125, 8].

When investigating the security loss of reductions from multi-user or adaptive multi-user security of MACs, we perhaps unsurprisingly reveal a similar story to that of reductions from digital signatures: MACs with randomized, non-deterministic, or stateful tagging algorithms exist and have been shown to have reductions to either single-user security or other standard assumptions which are nearly tight [51, 8], whereas for the vast majority of *deterministic* MACs (or the similar but slightly stronger notion of *pseudorandom functions*, or PRFs) we know of no tight reductions. Even the natural reduction for transforming a randomized MAC to a deterministic one (exhibited in [61] for signatures) requires a PRF and thus is subject to the same security loss.

Our second major result [101] shows that, just as before, this gap is inherent, and any reduction proving the adaptive multi-user security of a deterministic MAC based on some bounded-round assumption will necessarily inherit a security loss of $\Omega(\sqrt{\ell})$, where ℓ now represents the number of active instances of the MAC, thus ruling out such linear-preserving

reductions. In addition, through a series of minor extensions via tight reductions between primitives, we obtain a similar result for multi-user adaptive security of deterministic digital signatures and pseudorandom functions.[6]

Our result: The importance of "key uniqueness". This meta-reduction is somewhat structurally similar to the previous one, and uses the same "randomness switching" argument as its crux; however, there is significant additional subtlety involved in ruling out reductions from adaptive multi-user security. In fact, we run into issues as soon as we attempt to construct a meta-reduction in the first place. The natural extension of our previous result would be to have the ideal adversary open all but one of the ℓ instances' keys and then rewind to try and open the target instance's key; however, we very quickly run into the issue that the meta-reduction $\mathcal{B}$ then has no way to verify whether the key given to it by $\mathcal{R}$ is actually the correct key. Worse still, even if we add some tagging queries to each instance and attempt to verify the respective tags with respect to the opened keys, we have no guarantee that the keys opened are not simply keys which coincidentally agree with the correct key on some, but not all, inputs!

Prior meta-reductions dealing with multi-user security, most notably [81] for reductions from multi-user authenticated encryption, dealt with this issue by requiring a "key uniqueness" property of the primitive—that any two keys which agree with one another on a certain number of inputs must continue to agree on *all* inputs. Our first observation towards removing this requirement is that this property is to a degree "naturally" upheld: if we make a very large number $q(n)$ of tagging queries on random messages to an instance, then if a

[6]Moreover, we can show that all of these results extend to the case of reductions to the single-user security of pseudorandom functions: by leveraging the observation that the challenger for this assumption is stateless, we can avoid having to abort in the case of needing to forward communication to the challenger by simply querying the challenger, and thus we can tolerate the lack of a bound on the assumption's communication rounds in the analysis.

pair of keys collides (produces the same tags) on smaller than a $1 - O(1)/q(n)$ fraction of messages, we are overwhelmingly likely to have queried a point on which the pair does not collide. In fact, by a union bound over all 2^{2n} pairs of keys, we demonstrate that, with overwhelming probability, if there exists a pair of keys that produce the same tags for all $q(n)$ of our random tagging queries, then those two keys must collide on at least a $1 - 2n/q(n)$ fraction of all messages.

Rewinding adversaries and a "hybrid meta-reduction". At this point it might seem that we are done: if we simply make a large polynomial number $q(n)$ of tagging queries to every instance, open all but one key, and then rewind, we should be able to verify that the key we extract is correct, or will at least produce the same forgery as the correct key on a new random message, "most" of the time. Unfortunately, this is insufficient for an extremely subtle reason: namely, because we are not restricted to "simple" reductions, the reduction $\mathcal{R}$ can rewind the adversary $\mathcal{A}$. In the prequel, this was not an issue for the specific reason that we dealt with unique signatures only: $\mathcal{A}$ was constructed so that all randomness would be determined at the start, and so, since the uniqueness of the forgery guaranteed that there would be a single possible accepting transcript, rewinding $\mathcal{A}$ was provably "pointless". Having the possibility of multiple different forgeries corresponding to a single instance of $\mathcal{A}$, even if it is relatively unlikely, opens up the possibility for $\mathcal{R}$ to rewind $\mathcal{A}$ to try and extract a different forgery, leading to the careful analysis from the prequel breaking down completely.

Instead of dealing with this possibility directly, we approach it from a different angle: we develop and analyze a *hybrid* meta-reduction $\mathcal{B}'$ that is "rewinding-proof" in the same way as the meta-reduction in the prequel, but runs inefficiently in order to ensure that. At a high level, the hybrid $\mathcal{B}'$ operates by rewinding identically to $\mathcal{B}$, but, if rewinding is

successful, $\mathcal{B}'$ will do an exhaustive search to determine whether there is a unique correct forgery for that instance. If there is a unique forgery, then $\mathcal{B}'$ will return it; on the other hand, if there are multiple possible forgeries, then $\mathcal{B}'$ will make a deterministic choice from those (and we design the ideal adversary $\mathcal{A}$ to choose the same way). Given this hybrid, we notice that $\mathcal{B}'$ behaves identically to $\mathcal{A}$ whenever rewinding is successful (and there is some forgery to return), whereas $\mathcal{B}'$ behaves identically to $\mathcal{B}$ whenever there is a unique forgery. However, as we can prove that $\mathcal{B}'$ is always rewinding-proof (since the forgery it returns will be deterministically chosen), we can now use the analysis from the prequel to compare $\mathcal{B}'$ and $\mathcal{A}$; comparing $\mathcal{B}'$ and $\mathcal{B}$ is also possible due to the "effective key uniqueness" lemma we sketched above. Thus, by a *hybrid argument*[7], we can show that the failure probability of $\mathcal{B}$ to emulate $\mathcal{A}$ is bounded by what amounts to (roughly) the failure probability in the prequel plus the additional $2n/q(n)$ from the key uniqueness argument, once again yielding a roughly $\sqrt{\ell}$ lower bound on the security loss of $\mathcal{R}$.

This result, similar to the last one, presents an interesting gap between the provably achievable and the provably unachievable; without using randomized or stateful tagging algorithms, we thus far have no constructions of adaptive multi-user secure MACs, signatures, or PRFs with reductions asymptotically outperforming $O(\ell)$ security loss. There are notably some negative results [16, 39] which deal with the tangential case of reducing multi-user to single-user security of MACs and demonstrate that *generic* reductions—that is, reductions that hold for an arbitrary MAC implementation—do in fact inherit this security loss. However, we consider the case of reducing multi-user security directly to standard assumptions without going through single-user security, as well as specific (non-generic) reductions, and very little is known towards a tight characterization of what we can achieve in this space. We believe that our results represent considerable progress towards being able to achieve

[7]Hybrid arguments, in which one compares two experiments by comparing each to one or more intermediate "hybrid" experiments, are very standard and widely used in cryptographic proofs.

tight concrete security bounds with meta-reductions in areas such as this, since we introduce novel techniques to bypass many of the limitations that were historically inherent in these meta-reductions; however, there is still a vast amount of refinement possible in both the construction and analysis of such meta-reductions, and it would be of great future interest to either build more advanced meta-reductions to (if possible) tighten our bounds proven here or prove similar bounds for even greater varieties of primitives and reductions.

1.4 Our Results: Feasibilities for Efficient Secure Computation

We next turn to studying different notions of efficiency: in Chapter 3, we prove another pair of results exploring the *communication and computation complexity* achievable in protocols for *secure non-interactive two-party computation*. The first result, presented in Section 3.3 (and fully published in [100]), demonstrates the feasibility of *succinct* non-interactive two-party secure computation, which features limited communication complexity and running time for the protocol, without strengthening the assumptions required from those that can be used to construct the non-succinct version of the primitive. The second, presented in Section 3.5 (see also [99]), demonstrates the feasibility of non-interactive two-party secure computation satisfying a vastly strengthened definition of security which holds both in *concurrent* settings with many instances active and under composition as part of a larger protocol; this result, being fully non-interactive (i.e., achieving the theoretical minimum of two communication rounds), improves vastly on the round complexity of prior concurrent and composable secure computation protocols, which required an unspecified constant number of rounds. As a compelling addendum to this result, we prove that the assumptions we use in this construction are essentially minimal, obtaining a tight characterization of the necessary and sufficient assumptions for this strengthened definition of security.

1.4.1 Secure Computation

Secure two-party computation is a cryptographic primitive, first considered in [127, 62], which allows two parties with private inputs x and y to jointly compute a function $f(x, y)$ of their inputs in such a way that *neither party learns anything other than the output of the function*. Our results focus on *non-interactive* protocols (or NISC), those with two rounds (the minimum possible) of communication where only one party (the *receiver*) receives the output. The definition of security in terms of "knowledge" is formalized by a *simulation* paradigm: roughly speaking, such a protocol is secure if an adversary corrupting one of the two parties is unable to efficiently distinguish between a transcript of the actual execution and a transcript of an "ideal" execution where the corrupted party communicates only with a simulator that does not know the other party's input but only learns the output of the computation. If this is the case, since the simulator's messages must be dependent only on the output of f (and specifically not on the honest party's input), this intuitively agrees with the notion that the adversary in the real execution should learn nothing aside from that output; were that not the case, then the adversary could use the learned information (or lack thereof from the simulator) to tell the difference between the "real world" and the "ideal world".

Interestingly, there are readily available negative results on secure two-party computation: in the plain model (i.e., without any additional trusted setup or "trusted third party" functionalities available) and restricting to simulators that run in polynomial time, [63] shows that constructing a secure protocol with two rounds of computation is impossible, and in fact [83] shows that four rounds are both necessary and sufficient with black-box simulation. There are several ways to circumvent this impossibility. First, one can relax the definition of security to *semi-honest*, or "honest but curious", security, which requires the adversary to

follow the protocol honestly and learn only what information they can from that honest execution, whereas a fully *malicious* adversary can deviate from the protocol and send whatever messages they wish to attempt to extract additional information. As it turns out, secure two-party computation under only semi-honest security is easily implemented through fully homomorphic encryption [58], so we focus on security against malicious adversaries.

One can also hope to circumvent the impossibility of [63] by adding trusted setup assumptions: indeed, there are several positive results that construct secure non-interactive two-party computation in models with trusted setup [69, 79, 3, 36, 75, 97, 12]. Malicious security in the plain model, however, requires a relaxed notion of simulatability for the definition; the standard such relaxation is *superpolynomial-time simulation* [109, 115], where we allow the simulator to run in "slightly" superpolynomial running time in the security parameter—for instance, time $n^{\log^c(n)}$ for some constant c—whereas an adversary $\mathcal{A}$ attempting to distinguish between the real and ideal worlds must still run in polynomial time. As it turns out, this slight relaxation allows for non-interactive two-party secure computation protocols, as shown in a number of works [109, 121, 10]; the most recent of these, [10], shows that non-interactive two-party secure simulation with superpolynomial-time simulation can be based on subexponential security (i.e., security against subexponential-time adversaries) of various standard hardness assumptions. Our results on secure computation will focus on constructing provably secure protocols with augmented notions of security in this setting.

1.4.2 Succinct Non-Interactive Secure Computation

The first result [100] focuses on adding a property to which we refer as *succinctness*. To motivate this property, consider an "outsourced computation" setting where a computationally weak client (the receiver) wishes to compute a potentially complex function f of its input x

jointly with a powerful server (the sender), possibly with its own input y, while maintaining privacy of its input. Ideally, for such a scenario, we would want the communication complexity of the protocol we design, as well as the receiver's running time, to be as close as possible to independent of the running time of the functionality f; formally, we will require these to be at most polynomial in the input length (i.e., in the security parameter) and *polylogarithmic* in the running time of f.

This setting was one of the original motivations behind *fully homomorphic encryption* (FHE) [58], which, at a high level, is an encryption primitive that allows arithmetic operations to be carried out on ciphertexts. As we briefly mentioned already, FHE provides a semi-honest solution to this problem: both parties encrypt their inputs, the receiver sends the ciphertext of x to the sender, and the sender homomorphically computes $f(x, y)$ and returns the result ciphertext to the receiver for decryption. As we briefly mentioned, FHE already provides a semi-honest approach to non-interactive secure computation, which as it turns out has the succinctness property we desire as well. However, FHE alone is insufficient for malicious security, since we would need a way to verify that the sender is computing the correct functionality on the respective inputs. Meanwhile, the protocols we mentioned above that do satisfy malicious security, even those in models with trusted setup, are not succinct to the best of our knowledge.

Recently, a primitive relevant to solving the above "verification of computation" issue with FHE has been studied: delegation of computation [65], a computationally bounded variant of interactive proofs [67], wherein a prover performs a computation and convinces a computationally weaker verifier that it indeed performed the correct computation. Most relevant to our work is a notion of delegation where the computation can be decided adaptively: in adaptive delegation [29], a verifier (the receiver) sends a public key to the prover (the sender); the prover then selects a statement s and a function g, computes the output

$z = g(s)$, and returns to the prover s, g, z, and a short proof π of the validity of the computation which can be verified efficiently (i.e., in time polylogarithmic in the running time of g) using the verifier's secret key. In fact, [30] constructs a protocol for "private delegation" that allows inputs for both parties and achieves a weaker notion of simulation-based security known as *witness indistinguishability*[8], albeit only for the sender, while the receiver's output is considered publicly known by both parties. This protocol does satisfy our notion of succinctness and as such as a notable step towards succinct non-interactive secure computation; however, its security falls short of what we would need for full simulation-based security.

As it turns out, the protocol we present which finally does achieve full security is constructed from all of the primitives we have just discussed: FHE, non-succinct NISC, and adaptive delegation. Interestingly, all three of these required primitives can be based on subexponential security of the Learning With Errors assumption [118], which is one of the standard assumptions on which the non-succinct NISC of [10] can be based; thus, we are able to add the succinctness property to a non-succinct NISC without requiring any additional assumptions.

Our results: Combining FHE and Delegation using NISC. To illustrate our construction, we begin by, as hinted above, trying to combine FHE and delegation in order to solve the issue of verifying the computation to prevent a malicious sender from performing arbitrary computations. Consider an initial attempt at succinct NISC where both parties homomorphically encrypt their respective inputs, the receiver sends the ciphertext ct_x for x to the sender, and the sender performs the homomorphic evaluation of f on ct_x and the ciphertext ct_y for its own input, uses delegation to generate a proof π that it computed the homomorphic evaluation correctly, and sends ct_x, ct_y, the result ciphertext $\mathsf{ct}_{\mathsf{out}}$, and π to

[8] [109] shows that witness indistinguishability is equivalent to a relaxation of superpolynomial-time security where the simulator's running time is unbounded rather than "slightly" superpolynomial.

the receiver so they can verify the delegation and subsequently decrypt the output of f from $\mathsf{ct}_{\mathsf{out}}$ if it verifies.

This is an interesting idea, but it falls flat when we realize that a malicious receiver can also just decrypt ct_y (which is required for the verification) to learn the sender's input. So, instead of performing the computation in the clear, we hide the inputs by moving the computation to an instance of a non-succinct NISC run in parallel with the messages from the above approach: the receiver provides the delegation secret key, the sender inputs ct_x, ct_y, $\mathsf{ct}_{\mathsf{out}}$, and π, and the NISC runs the delegation verifier, outputting $\mathsf{ct}_{\mathsf{out}}$ if verification succeeds and $\perp$ if not. This intuitively should allow the receiver to learn the output ciphertext without learning anything about ct_y or the sender's input y.

This approach manages to (mostly) provide security against a malicious receiver. In the case of a malicious sender, even though the sender cannot decrypt the receiver's input directly from ct_x, the sender can still break security by exploiting malleability of FHE ciphertexts; for instance, the sender can very easily use homomorphic evaluation and ct_x to produce a valid ciphertext of $x+1$ and use it in place of its own ciphertext ct_y. To prevent this, we add checks to the internal NISC that take x, y, the ciphertexts ct_x and ct_y, the public key pk, and the randomness r_x and r_y used in the respective encryptions, and verify that ct_x and ct_y are correctly generated with respect to the inputs, key, and randomness. Moreover, we have the *sender* provide the ciphertext ct_x of the *receiver's* input, as well as the public key, to ensure that the receiver sent the correct values in its first message. As an interesting aside, this approach means that we need to rely on an FHE scheme which satisfies *perfect correctness*—that is, that the ciphertexts are able to be correctly evaluated and decrypted regardless of the randomness used to generate them—to ensure that there is no "bad" randomness either party can use to produce a ciphertext that passes the check in the NISC but results in an incorrect output.

There is still an issue from a simulation perspective: considering the simulator in the definition of secure computation, we need a way to simulate the ciphertext $\mathsf{ct}_{\mathsf{out}}$ returned from the NISC when the verifications succeed. However, the simulator only knows the output of the functionality f, and we have no way to guarantee that the ciphertext that could be produced by (for instance) directly encrypting that output would be indistinguishable from the true ciphertext $\mathsf{ct}_{\mathsf{out}}$ produced from homomorphically evaluating f on the ciphertexts of the two inputs (which are not available to the simulator). Fortunately, this is fairly easily solved by adding yet more of the functionality—in this case, the final decryption of the output ciphertext—to the internal NISC. This ensures that the NISC simply returns the plaintext output from f, which is of course known by the simulator and can thus be trivially simulated.

A subtlety: implicit decryption. However, this ostensibly easy solution introduces a very subtle complication to the security proof. Notice that, in the case of a malicious sender, the simulator $\mathcal{S}$ needs to generate a simulated first message, including the ciphertext ct_x, independently of x. However, even though the malicious sender is effectively unable to decrypt the ciphertext and reveal information about x, the same is surprisingly not true of the functionality of the internal NISC; in order to decrypt the final output from $\mathsf{ct}_{\mathsf{out}}$, the NISC requires knowledge of the same secret key used to encrypt x. While intuitively this should not cause an issue since the internal NISC never *does* decrypt x directly, it is actually still theoretically possible for the malicious sender to manipulate their own inputs in such a way that the output becomes dependent on x and "implicitly" decrypts it, as shown in related work constructing delegation from FHE and related primitives [49].

Fortunately, in our result, we are able to carefully circumvent this issue by using a hybrid experiment with an alternate functionality for the underlying NISC that removes the implicit

decryption. Specifically, in the case of a malicious sender, the receiver must be honest, which means that in that specific case we can remove the checks to verify that the receiver's inputs are generated correctly; moreover, due to the security property (soundness) of the delegation scheme and the perfect correctness of the FHE, we can show that the decryption of $\mathsf{ct}_{\mathsf{out}}$ is with overwhelming probability equal to $f(x, y)$ whenever π is an accepting proof. Since x and y are inputs to the internal NISC, this means that we can replace the decryption of $\mathsf{ct}_{\mathsf{out}}$ in its functionality by simply returning $f(x, y)$. Crucially, such a functionality no longer needs to know how to decrypt, and in particular no longer needs the FHE secret key as an input. Thus, we prove simulation-based security against a malicious sender by a hybrid argument: first, we demonstrate that the hybrid experiment and the real protocol are indistinguishable since the output of the internal NISC must remain the same, and then we prove simulatability with respect to the hybrid experiment, which is now feasible since the hybrid experiment is not subject to the same issue of "implicit decryption" as the real one.

Succinctness. As a final note, while it is far from immediately obvious that we retain the succinctness of the original approach using FHE and delegation after transferring such a large part of the computation in our protocol to the internal NISC (which is not succinct itself[9]), recall that, intuitively, the bulk of the computation is present in the evaluation (or, respectively, homomorphic evaluation using the ciphertexts) of the functionality f. And indeed, this evaluation remains solely in the hands of the sender; the evaluations done by the NISC (and thus computed jointly between the receiver and the sender) include verifying the input ciphertexts (which depends only on the input length and not at all on f) and the efficient verification procedure of the delegation scheme (which is polylogarithmic in the running time

[9]However, even for non-succinct NISC, since the parties involved must run in polynomial time, we retain communication complexity and computation time polynomial in the input lengths and running time of the functionality evaluated.

of f). Hence, for complex functionalities, the sender does the overwhelming majority of the computation, and the receiver's computation and the communication complexity are limited as desired.

1.4.3 Concurrently Composable Non-Interactive Secure Computation

The process of leveraging the security of a known construction of a primitive to construct and prove a security reduction for a strengthened version of the primitive, such as we perform in the previous (and subsequent) result, or in general of constructing a new protocol through *composing* known and previously proven primitives, is quite common in modern cryptographic research. Thus, an extremely desirable notion of security that is conspicuously absent in all of the notions of security we have discussed so far is that of *composable security*, which provides security guarantees for a protocol even when it is used as a subprotocol or component in another protocol. This would have, for instance, been extremely useful in the prequel where we built our succinct NISC from a non-succinct one, but, as previously known constructions of NISC do not provide composability guarantees, we instead had to reduce to simulation-based security of the underlying assumption by hand in a rather tedious part of the proof.

There are classical definitions of simulation-based security that in fact do consider composability [96, 33]; however, the earliest definitions also only considered *stand-alone* security, or security of a single instance of the protocol, when realistically, and especially for practical protocol design, we would want security guarantees to hold even in a *concurrent* setting [52, 48, 50], where many instances of a protocol are executed, possibly at the same time,

and with an adversary which might control a large subset of the participants (possibly with different roles in different instances), adaptively use the results of one protocol to influence another, and reorder communication to or from corrupted parties at will.

Universally composable security. The first definition to provide both composability and concurrency guarantees is *universally composable* (UC) security [34]. UC security adds to the notion of the adversary an external "observer", or *environment* $\mathcal{Z}$, which runs and observes interactions between an adversary and various instances of the protocol, and can perform a wide variety of actions as described above (e.g., corrupting parties or controlling communication). We say that a protocol Π *UC-realizes* some functionality f if the environment is unable to distinguish between the transcripts of the real execution using Π and the "ideal" execution using a simulator $\mathcal{S}$ and an "ideal" version of f with a trusted third party (specifically, for two-party secure computation, both parties provide their inputs to a trusted third party that returns the output to the receiver, but all messages are simulated by $\mathcal{S}$ independently of the respective inputs); this is similar in spirit to the earlier notions of simulation-based security but of course gives the external observer far more power.

UC security provides an extremely strong concurrency and composability guarantee: specifically, if a protocol Π UC-realizes f and another protocol π exists that uses Π as a sub-protocol, then, for the purposes of proving security, the real protocol π is interchangeable with (UC-emulated by) the composed protocol π' where every instance of f is replaced with the idealized functionality. However, as a strict strengthening of standard simulation-based security, it stands to reason that this notion is extremely hard to achieve. For NISC realizing arbitrary polynomial-time functionalities f, the impossibilities of [63, 83] still hold for the case of the plain model with polynomial-time simulation; even considering computing restricted classes of functionalities, there exist impossibility results [37, 91] showing that

very few functionalities can even be computed with just concurrent security, let alone both concurrent and composable.

Oracle-aided universal composability. Hence, to achieve meaningful positive results in the plain model, we once again must consider a relaxed notion of simulatability such as superpolynomial-time simulation. SPS, however, does not translate directly to the case of composable security; intuitively, this is because SPS security only holds against polynomial-time adversaries, but composable security requires security against adversaries "simulated" by a superpolynomial-time simulator, which may themselves require superpolynomial time to run. So, to provide a relaxation of simulatability suitable for UC security, we consider a notion known as "angel-based" UC security [115], or *"UC security with a superpolynomial-time helper"* [38]. Under these definitions, simulatability is restricted to polynomial-time simulators (thus avoiding the "simulated adversary" issue above), but the environment and simulator are both given access to some specific superpolynomial-time oracle, or an "angel". This is a strictly stronger notion of security than SPS (since any oracle-aided simulator can be implemented in standard SPS), and additionally allows for concurrency and composability, albeit only among families of protocols secure relative to the same oracle.

Importantly, we *can* construct protocols for two-party computation that are provably secure under oracle-aided UC [115, 94, 38, 85, 86, 70]. However, in terms of round complexity, the state of the art leaves much to be desired; the best known UC-secure protocol runs in a large unspecified constant number of rounds, far from the two-round non-interactive protocols (i.e., NISC) one would hope to achieve. Even dropping composability and considering concurrent NISC with SPS security, 5 rounds [56], or 3 simultaneous messages [11], is the best known, with concurrent NISC only having been constructed in models with trusted setup in the form of a common reference string [21, 57, 22].

CCA-secure commitments. Returning to UC security, most of the recent constructions of UC-secure two-party computation protocols have been based on a notion of *CCA-secure commitments* [38]. Commitments are a primitive that allows a committer to "commit" to a certain value in such a way that an adversary is unable to determine the committed value from the commitment alone (*hiding*), but that the committer on opening the commitment later can open it only to the value to which they originally committed (*binding*). The CCA security property is a notion analogous to the classical notion of chosen ciphertext attack (CCA) security for encryption [117]: for commitments, it states that the hiding property should hold even against an adversary with access to a decommitment oracle $\mathcal{O}$. Namely, in a security game where the adversary selects two distinct values to commit and an identifier (or *tag*), receives a commitment to a random one of the two values under that identifier, and then may query an oracle $\mathcal{O}$ which, given any commitment on a *different* tag than the challenge tag, will return the corresponding value, the adversary should be unable to discern which of the two values was committed with probability significantly better than 1/2 (i.e., randomly guessing).

The earlier and more conventional known constructions of CCA-secure commitments [38, 87, 88] are *interactive*, such that committing to a value (and later revealing the commitment) are represented by multi-round protocols between a committer and a receiver. Interactive commitment schemes were sufficient to construct prior protocols for UC-secure two-party computation; however, the protocol we construct, as we seek to minimize round complexity, will require a *non-interactive* or single-round CCA-secure commitment, where committing to a value can be done through a single algorithm and requires no communication with the receiver aside from sending the final commitment. These have proven to be far more elusive and require far more complex and non-standard assumptions than their interactive counterparts. The initial construction in [108] is based on adaptive one-way permutations;

later, [24] construct a non-interactive commitment scheme, based on an earlier construction of [88][10], that satisfies a slightly weaker notion of "concurrent non-malleability" [112, 89] and is constructed from keyless multi-collision-resistant hash functions, injective one-way functions, non-interactive witness-indistinguishable arguments (NIWIs), and subexponentially-secure time-lock puzzles. We believe that the analysis of this construction can be modified to also provide full CCA security, but this has yet to be proven.

Our result: UC security from (nearly) minimal assumptions. The next and final result we present here is a construction of non-interactive UC-secure two party computation, improving on the known upper bound for the achievable round complexity of UC-secure, and even concurrently secure, two-party computation by showing that the theoretical minimum of two rounds is indeed feasible. The protocol we construct requires two major building blocks: a stand-alone SPS-secure NISC (such as that of [10]) and a non-interactive CCA-secure commitment.

Given the above discussion, it might be concerning that we require non-interactive CCA-secure commitments, as they rely on significantly more complex assumptions than the interactive analogue used to build prior secure computation protocols. However, we make two observations about this requirement. First, a slight weakening of CCA-secure commitments, which we call *weak* CCA-secure commitments[11], is actually sufficient for our result. The second observation is, rather more interestingly, that weakly CCA-secure commitments are not only sufficient, but also *necessary*, for constructions of UC-secure NISC. Specifically, we prove as an addendum to our result that UC-secure NISC actually implies, via reduction,

[10]This earlier construction gave non-interactive CCA-secure commitments secure against *uniform* adversaries, but this is insufficient as the overwhelming majority of cryptographic adversaries are *non-uniform*, meaning that, loosely speaking, they can receive an auxiliary input as "advice".

[11]Whereas standard CCA-secure commitments require the decommitment oracle in the security game to return not only the value committed but also the randomness used to commit, weak CCA-secure commitments exclude the latter requirement and allow the oracle to return only the value.

the existence of weakly CCA-secure commitments; as such, it stands to reason that any assumptions required for weakly CCA-secure commitments are, inevitably, likewise required to construct UC-secure NISC.

More than that, though, this implication gives us a nearly tight characterization of the assumptions necessary and sufficient for UC-secure NISC: our construction shows that UC-secure NISC is implied by the existence of weakly CCA-secure commitments and stand-alone secure NISC, whereas, conversely, UC-secure NISC also implies both the existence of weakly CCA-secure commitments and stand-alone secure NISC. The only gap is that, while our construction of UC-secure NISC requires a stand-alone secure NISC secure against superpolynomial-time adversaries, it only implies a stand-alone secure NISC secure against polynomial-time adversaries. Furthermore, we prove our result for only NISC satisfying *perfect correctness* (i.e., that an honestly executed protocol never fails, rather than potentially failing with small but negligible probability), and it is unclear how to adapt it to NISC protocols without that additional requirement. Nonetheless, the fact that we obtain a nearly bidirectional implication—an almost precise picture of the necessary and sufficient assumptions for the primitive we construct—is not only compelling in that our construction is provably constructed from a close to minimal set of assumptions, but also is theoretically interesting because the "equivalence" of these two sets of assumptions contributes greatly to our understanding of what implications and reductions can and cannot be proven. For instance, this can just as easily be interpreted as an impossibility result (that no set of assumptions strictly weaker than stand-alone secure NISC and weakly CCA-secure commitments can possibly imply UC-secure NISC) as it can be interpreted as a feasibility result for a strong and practically motivated definition of security.

Constructing UC-secure NISC. To build a UC-secure NISC from a stand-alone secure NISC, the obvious starting point is to have both parties send their inputs x and y to the internal NISC, which will compute f and return the result to the receiver. Of course, this is quite far from providing security on its own; we first need a way to ensure that the inputs given by a potentially malicious sender and receiver are well-formed, or technically that the simulator $\mathcal{S}$ we construct is able to extract the malicious sender's input from its message (for the purpose of sending it to the trusted third party to receive the correct output). We do this by leveraging our CCA-secure commitment scheme: the receiver and sender will commit to their respective inputs x and y with commitments c_x and c_y and send them with their respective messages. In order to extract from these commitments, we leverage the superpolynomial-time helper allowed by the security definition: specifically, we consider a helper $\mathcal{H}$ which acts as the decommitment oracle $\mathcal{O}$ from the definition of CCA security, but importantly will only open commitments generated using the identifiers of the party invoking it (that is, since only the environment and simulation have access to it, it will only open commitments from *corrupted* parties). So, when the simulator receives a message containing a commitment by a corrupted party, it can invoke $\mathcal{H}$ to extract the value, but notably honestly generated commitments are still secure from the adversary.

Of course, we still have no guarantee that a malicious adversary will commit to their correct input, or even to the same value they provide to the internal NISC. We will have to deal with this problem slightly differently depending on whether the receiver or sender is corrupted. For the case of a corrupted receiver, we can take an approach very similar to what we did in the prequel: specifically, we add a check to the NISC where the receiver inputs the randomness r_x used to generate c_x, the sender inputs c_x (to ensure that the receiver sends the correct commitment), and the NISC checks that c_x is correct with respect to x and r_x, returning $f(x, y)$ if so and $\perp$ if not. Hence, we can verify that the receiver's first message

is well-formed with respect to its inputs, and when it is we can always extract the correct input x from that message for the simulation.

Unfortunately, we cannot simply verify the commitment c_y the same way in the case of a corrupted sender, since the receiver provides its inputs to the internal NISC in the first round, before the sender even generates its commitment c_y. As an initial approach towards dealing with this, we use a second NISC to implement an SPS-secure two-round zero-knowledge (ZK) interactive argument[12], which we run in parallel with the internal NISC. Using this, the sender will prove that the commitment c_y *and the sender's message for the internal NISC* are correctly generated with respect to the input y and the respective randomness (the latter being the private witness for the proof); the receiver can then verify this without learning anything about the input y.

This alone is insufficient, since we cannot simulate the internal SPS-ZK in the case of a corrupted receiver without knowing the statement to prove—that is, the commitment c_y—and that would require knowledge of y. In fact, a similar problem holds for the original internal NISC as well. So, instead, we need to add a "fake witness" to the SPS-ZK: we add a "trapdoor" t that is generated at random by the receiver and committed to (and verified) in the same way as x. However, using the trapdoor t will provide simulatable functionalities for both the SPS-ZK *and* the internal NISC that do not require knowledge of the input y. Specifically, we modify the internal NISC to allow the sender to input t as well as a "hard-coded output" z^*, such that if t is provided then the NISC's functionality is effectively overwritten to output z^* if t matches the receiver's input t and $\perp$ if not. Then the SPS-ZK

[12]This is a protocol that allows a prover to prove to a verifier that a statement x is a member of some language $\mathcal{L}$ by using a witness w, but in such a way that the verifier only learns that x is true without learning anything about w—specifically, that proofs of a statement x can be (superpolynomially) simulated independently of the witness w. The zero-knowledge functionality can be easily implemented by a NISC: consider a functionality that takes (x, w) from the prover, tests for membership in $\mathcal{L}$, and returns either (x, Accept) or (x, Reject) depending on the result.

will prove that either (1) c_y and the sender's NISC message are correctly generated OR (2) the sender's NISC message was generated using the trapdoor t and no input y (with t, z^*, and the randomness as the private witness). The honest sender can provide a witness for (1), whereas the simulator communicating with a corrupted receiver can extract the trapdoor t from the receiver's first message using $\mathcal{H}$ and use that in conjunction with the output z^* from the trusted third party to simulate a NISC and ZK message using the second track.

Finally, with the addition of the "fake witness", we also need to add an "argument of knowledge" property to the ZK to ensure that a corrupted sender cannot somehow provide a valid witness to the second track of the SPS-ZK without actually knowing the trapdoor t (and thus having broken the commitment scheme by determining the value of the honest receiver's commitment). To do this, we rely on a technique from [109] where we add a commitment to the witness for each track of the ZK, and each track now additionally proves that the respective commitment is well-formed. Specifically, this does two things: first, it allows us to guarantee that a corrupted sender cannot use the second track of the protocol to break its security, as doing so would require it to provide a valid commitment to the witness, including the trapdoor t, in a way that is extractable and can thus be leveraged to contradict CCA security of the commitment. Second, this actually allows us to remove the commitment c_y from the protocol altogether, since y is already part of the witness to the "honest" first track of the ZK, and so we can already use that commitment to extract y with $\mathcal{H}$ in the case of a corrupted sender. With this, our construction is complete.

Proving minimality of assumptions. Finally, as discussed earlier, we prove that weakly CCA-secure commitments are not only sufficient but also *necessary* for constructing UC-secure NISC: we do this by constructing weakly CCA-secure commitments from a UC-secure NISC. Specifically, given a NISC that implements the *equality* functionality—that

is, $f(x, y) = 1$ if $x = y$ and 0 otherwise—we can commit to a value v by running the first-round receiver protocol of the NISC with input given by $v||p$ for some random padding p, and later open the commitment by verifying that it is valid with respect to the padding p and randomness r of the NISC.

Binding of the commitment scheme will hold due to the perfect correctness and security of the NISC: at a high level, if the same first message m of the NISC is valid with respect to two different inputs $v||p$ and $v'||p'$, this implies that the output of the respective protocol if the NISC were finished would, by perfect correctness, have to be 1 with probability 1 when the receiver's and sender's input were both $v||p$, and 0 with probability 1 when the receiver's input were $v||p$ but the sender's input were anything else. By security, then, the simulated interaction in the ideal world would have to output 1 with overwhelming probability when the sender's input is $v||p$ but 0 with overwhelming probability for any other input. But, since the simulator $\mathcal{S}$ only knows the result of the trusted third party which implements f—that is, whether the value it extracts from the receiver's first message is equal to the sender's input—the only feasible way for this to happen is for the input to be different only when f tests for equality with $v||p$, or if $\mathcal{S}$ extracts $v||p$ from the message m with overwhelming probability. But a symmetric argument holds to show that $\mathcal{S}$ also extracts $v'||p'$ from m with overwhelming probability, a clear contradiction.

For weak CCA security of the commitment scheme, we prove the reduction in two major steps. We begin by showing that, given any adversary $\mathcal{A}$ against CCA security, it can be implemented in polynomial time using the superpolynomial helper function $\mathcal{H}$ from the definition of UC security of the NISC instead of the decommitment oracle $\mathcal{O}$ (thus transforming it into an adversary against standard hiding rather than CCA security). This intuitively follows from the same extraction property as above: the simulator $\mathcal{S}$ must be able to extract the correct input from the receiver's first message, meaning that it must provide the same

functionality as the decommitment oracle $\mathcal{O}$, and so any polynomial-time adversary with access to $\mathcal{O}$ (i.e., $\mathcal{A}$) must be implementable by replacing invocations of $\mathcal{O}$ with invocations of $\mathcal{S}$, which is itself polynomial-time with access to $\mathcal{H}$.

As the second and final step, we show that the commitment scheme must satisfy hiding against polynomial-time adversaries even if they have access to $\mathcal{H}$. This follows from UC security: in an "ideal" version of the commitment scheme where we replace the NISC protocol with its simulated ideal-world counterpart, hiding holds trivially because the first message is simulated by $\mathcal{S}$ independently of the receiver's input. However, due to the guarantees given by UC security, it immediately follows that a polynomial-time adversary, even with access to $\mathcal{H}$, cannot distinguish the "ideal" version of the security game using the idealized NISC protocol from the *real* security game where we replace the idealized protocol with the *actual* NISC protocol, and so hiding holds in the real world as well. Thus, we have proven that, were there an adversary $\mathcal{A}$ that broke CCA security, it could be rewritten into a hiding adversary with access to $\mathcal{H}$, thus contradicting the fact that hiding still holds against such adversaries and completing the argument. This final proof, while simple, serves as an excellent conclusion to the results we present: not only does it demonstrate the power of UC security and better motivate the NISC protocols we construct, but it also represents a significant contribution to our overarching motivation of examining the limits of provable security and efficiency by providing a strong and nearly matching feasibility and impossibility result.

1.5 Overview

Chapter 2 contains both of the impossibility results for linear-preserving reductions that we discuss here in full. We present all relevant definitions in Section 2.2, as both of the results in

the chapter rely on very similar definitions; we then present the meta-reduction proving the impossibility of linear-preserving reductions from unique signatures to standard bounded-round assumptions in Section 2.3 and the meta-reduction proving the impossibility of linear-preserving reductions from adaptively secure multi-user MACs to standard bounded-round assumptions in Section 2.4.

Chapter 3 contains both of the feasibility results for efficient notions of non-interactive secure computation in full. The two results use very different sets of definitions due to the differences in both the definitions of security realized and the internal primitives needed for the construction, so we present the definitions needed for the construction of succinct non-interactive secure computation in Section 3.2 before the result itself in Section 3.3, and we present the additional definitions needed for the construction of concurrently composable non-interactive secure computation in Section 3.4 before the corresponding result in Section 3.5. Finally, Section 3.6 contains our discussion and proof concerning the minimality of the required assumptions for the latter construction.

CHAPTER 2

IMPOSSIBILITY RESULTS FOR LINEAR-PRESERVING REDUCTIONS

2.1 Introduction

We begin the technical section of this work by presenting our impossibility results demonstrating the inherent inefficiency of reductions from unique digital signatures and adaptively multi-user secure MACs to standard assumptions. First, we present formal definitions relevant to the two results. In Section 2.3, we state our theorem proving the impossibility of linear-preserving security reductions from unique digital signatures to bounded-round assumptions; after the theorem statement, we present a short technical overview in greater depth than the one presented in the introduction before proceeding to the complete proof. Section 2.4 follows the same format for our second result, proving the impossibility of linear-preserving security reductions from adaptive multi-user security of MACs to bounded-round assumptions. Both of these results present new techniques for constructing meta-reductions to prove concrete security bounds; they are the first meta-reductions in this space to bypass the limitations of prior results and rule out unrestricted notions of black-box reductions to extremely general classes of security assumptions.

2.2 Preliminaries and Definitions

Let $\mathbb{N} = \{1, 2, 3, \ldots\}$ and $[n] = \{m \in \mathbb{N} \mid m \leq n\} = \{1, 2, \ldots, n\}$. Take 1^n (given as an input to certain algorithms to represent the security parameter) to be the string of n ones. Let $\mathbf{1}_P$ be the *indicator function* of some equality or inequality statement P, defined as being 1 when P is true and 0 otherwise.

We say a statement is true for "sufficiently large $n \in \mathbb{N}$" if there exists $N \in \mathbb{N}$ such that the statement holds for all $n \geq N$. We shall call a function $\epsilon(\cdot)$ *negligible* if, for any polynomial $p(\cdot)$, $\epsilon(n) < 1/p(n)$ for sufficiently large $n \in N$; conversely, a function is *non-negligible* or *polynomial* if the above is not true.

Interactive Algorithms. We model interaction between algorithms by Turing machine interaction as given in [67]. For oracle-aided algorithms $\mathcal{R}^{(\cdot)}$ with deterministic oracles, we assume that $\mathcal{R}$'s oracle will take as input the current partial transcript of the interaction and return the result of applying the oracle's respective "next-message" function to the transcript. Using a stateless oracle in such a way allows us to consider the case where $\mathcal{R}$ can "rewind" its oracle to a previous point in the interaction by providing an earlier transcript. In fact, we note that even stateful oracles can be modeled in this way (by "replaying" or reading from the transcript to simulate the state of the oracle), as can non-deterministic oracles (by sampling a deterministic oracle randomly from a family of oracles with different fixed randomness); we take advantage of both of these observations in the proof of our main theorem.

Given oracle-aided algorithm $\mathcal{R}$ and interactive algorithms $\mathcal{A}$ and $\mathcal{C}$, let $\mathcal{R}^{\mathcal{A}}(x)$ denote the distribution over the output of $\mathcal{R}$ when interacting with oracle $\mathcal{A}$ on common input x, $\langle \mathcal{A}, \mathcal{C} \rangle(x)$ denote the distribution over the output of $\mathcal{C}$ after interaction between $\mathcal{A}$ and $\mathcal{C}$ on common input x (also provided to oracles if $\mathcal{A}$ or $\mathcal{C}$ are oracle-aided), and $[\mathcal{A} \leftrightarrow \mathcal{C}](x)$ denote the complete *view* (all messages sent, random coins used, and outputs) of the interaction between $\mathcal{A}$ and $\mathcal{C}$ on common input x.

If we wish to describe an interaction with certain randomness fixed, we write $\langle \mathcal{A}, \mathcal{C} \rangle_{y \leftarrow a}(x)$ (resp. $[\mathcal{A} \leftrightarrow \mathcal{C}]_{y \leftarrow a}(x)$) to denote that variable y is fixed to value a, or $\langle \mathcal{A}, \mathcal{C} \rangle_y(x)$ to simply

indicate that y is fixed when the value is clear from context.

2.2.1 Intractability Assumptions

We define a notion of "game-based security assumptions" as in [103, 110]. Informally, an assumption can be thought of as a pair of a challenger and a threshold function, where an adversary is able to "break" the assumption by causing the challenger to accept an interaction with probability non-negligibly greater than the given threshold.

Definition 1. For polynomial $r(\cdot)$, we call a pair $(\mathcal{C}, t(\cdot))$ an $r(\cdot)$**-round intractability assumption** if $t(\cdot) \in [0, 1]$ is a function and $\mathcal{C}$ is a (possibly randomized) interactive algorithm taking input 1^n and outputting either Accept or Reject after at most $r(n)$ rounds of external communication.

Given a probabilistic interactive algorithm $\mathcal{A}$ which interacts with $\mathcal{C}$, we say that $\mathcal{A}$ **breaks** the assumption $(\mathcal{C}, t(\cdot))$ with some non-negligible probability $p(\cdot)$ if, for infinitely many $n \in \mathbb{N}$: $\Pr\left[\langle \mathcal{A}, \mathcal{C} \rangle(1^n) = \mathsf{Accept}\right] \geq t(n) + p(n)$.

Conversely, we refer to $\mathcal{C}$ as **secure** if there exists no $\mathcal{A}$ which breaks $\mathcal{C}$ with non-negligible probability.

Lastly, we call an assumption $(\mathcal{C}, t(\cdot))$ a **bounded-round intractability assumption** if there exists some polynomial $r(\cdot)$ such that $(\mathcal{C}, t(\cdot))$ is an $r(\cdot)$-round intractability assumption.

The general notion of an intractability assumption captures any standard cryptographic assumption, including our earlier definition of adaptive multi-key unforgeability. Specifically, this would be the *unbounded-round* assumption $(\mathcal{C}_{\Pi}^{\ell(n)}, 0)$ (using the challenger defined in

Definition 9). Clearly, we cannot hope to rule out tight reductions from, say, adaptive multi-key unforgeability to itself; as such, we focus on ruling out only reductions to bounded-round assumptions, but we note that virtually all "standard" cryptographic assumptions fall into this category.[1]

2.2.2 Black-Box Reductions

We next formalize what it means to "base the security of one assumption ($\mathcal{C}_1$) on another assumption ($\mathcal{C}_2$)". Intuitively, this requires a proof that, if there exists an adversary breaking $\mathcal{C}_1$, then there likewise must exist an adversary breaking $\mathcal{C}_2$, which implies the desired result by contrapositive.

In practice, virtually all reductions are "black-box" reductions, where the adversary breaking $\mathcal{C}_2$ is given by an efficient oracle-aided machine $\mathcal{R}$ which interacts in a "black-box" manner with an adversary which breaks $\mathcal{C}_1$ and uses the view of the interaction to break $\mathcal{C}_2$. Formally:

Definition 2. Given a probabilistic polynomial-time oracle-aided algorithm $\mathcal{R}$, we say that $\mathcal{R}$ is a **black-box reduction for basing the hardness of assumption** $(\mathcal{C}_1, t_1(\cdot))$ **on that of** $(\mathcal{C}_2, t_2(\cdot))$ if, given any deterministic algorithm $\mathcal{A}$ that breaks $(\mathcal{C}_1, t_1(\cdot))$ with non-negligible probability $p_1(\cdot)$, $\mathcal{R}^{\mathcal{A}}$ breaks $(\mathcal{C}_2, t_2(\cdot))$ with non-negligible probability $p_2(\cdot)$.

Furthermore, if on common input 1^n $\mathcal{R}^{\mathcal{A}}$ queries $\mathcal{A}$ only on input 1^n, we refer to $\mathcal{R}$ as **fixed-parameter**.

We notably allow reductions to rewind their oracles (by sending a transcript from earlier

[1] An example of a "non-standard" assumption that does not fit this definition would be a non-falsifiable assumption, e.g., a "knowledge of exponent" assumption (see, e.g., [43]).

in the interaction) and even run multiple, potentially interleaved, instances of their oracle.

The restriction to deterministic oracles $\mathcal{A}$ may seem strange at first, but we stress that we can (and will) in fact simply model a randomized oracle by a family of deterministic oracles (where each deterministic oracle represents some fixed setting of the randomness). Using deterministic oracles enables us to reason about cases where the reduction $\mathcal{R}$ can rewind or restart the oracle. We also will restrict to fixed-parameter reductions: this is a restriction inherent to the meta-reduction paradigm, yet it is a natural one (since, as far as we know, all reductions in practice are indeed fixed-parameter).

Of course, we can apply the definition of a reduction to adaptive unforgeability as defined above, using the natural formulation as an intractability assumption:

Definition 3. We shall refer to a probabilistic polynomial-time oracle-aided algorithm $\mathcal{R}$ as a **fixed-parameter black-box reduction for basing adaptive $\ell(n)$-key unforgeability of a MAC Π on the hardness of an assumption** $(\mathcal{C}, t(\cdot))$ if it is a fixed-parameter black-box reduction for basing the hardness of assumption $(\mathcal{C}_{\Pi}^{\ell(n)}, 0)$ on that of $(\mathcal{C}, t(\cdot))$, where $\mathcal{C}_{\Pi}^{\ell(n)}$ is as given in Definition 9.

We refer to a probabilistic polynomial-time oracle-aided algorithm $\mathcal{R}$ as a **fixed-parameter black-box reduction for basing adaptively secure unforgeability of a MAC Π on the hardness of an assumption** $(\mathcal{C}, t(\cdot))$ if there exists polynomial $\ell(\cdot)$ for which $\mathcal{R}$ is a fixed-parameter black-box reduction for basing adaptively secure $\ell(n)$-key unforgeability of Π on the hardness of $(\mathcal{C}, t(\cdot))$.

2.2.3 Security Loss

We define a notion of the "inherent efficiency" of a reduction, or the *security loss*, intuitively representing a worst-case ratio between the "work" (expected time) needed to break the assumption $\mathcal{C}_2$ (i.e., the underlying assumption) and the "primitive" $\mathcal{C}_1$ (in our case, adaptive multi-key unforgeability). If the primitive is significantly easier to break than the underlying assumption, this indicates that the reduction is intuitively "less powerful" at guaranteeing security for the primitive, which corresponds to a higher security loss.

Definition 4. Let $\mathcal{R}$ be a black-box reduction for basing the hardness of assumption $(\mathcal{C}_1, t_1(\cdot))$ on that of $(\mathcal{C}_2, t_2(\cdot))$. Given any deterministic $\mathcal{A}$, we define the following, where $\tau_{\mathcal{M}}(x)$ denotes the time taken by an algorithm $\mathcal{M}$ in experiment x, $r_{\mathcal{A}}$ denotes all random coins used by $\mathcal{A}$ and $\mathcal{C}_1$ in the experiment $\langle \mathcal{A}, \mathcal{C}_1 \rangle$, and $r_{\mathcal{R}}$ denotes all random coins used by $\mathcal{A}$, $\mathcal{C}_2$, and $\mathcal{R}$ in the experiment $\langle \mathcal{R}^{\mathcal{A}}, \mathcal{C}_2 \rangle$:

- $\mathsf{Success}_{\mathcal{A}}(n) = \Pr_{r_{\mathcal{A}}}[\langle \mathcal{A}, \mathcal{C}_1 \rangle_{r_{\mathcal{A}}}(1^n) = \mathsf{Accept}] - t_1(n)$
- $\mathsf{Success}_{\mathcal{R}^{\mathcal{A}}}(n) = \Pr_{r_{\mathcal{R}}}[\langle \mathcal{R}^{\mathcal{A}}, \mathcal{C}_2 \rangle_{r_{\mathcal{R}}}(1^n) = \mathsf{Accept}] - t_2(n)$
- $\mathsf{Time}_{\mathcal{A}}(n) = \max_{r_{\mathcal{A}}}(\tau_{\mathcal{A}}([\mathcal{A} \leftrightarrow \mathcal{C}_1]_{r_{\mathcal{A}}}(1^n)))$
- $\mathsf{Time}_{\mathcal{R}^{\mathcal{A}}}(n) = \max_{r_{\mathcal{R}}}(\tau_{\mathcal{R}^{\mathcal{A}}}([\mathcal{R}^{\mathcal{A}} \leftrightarrow \mathcal{C}_2]_{r_{\mathcal{R}}}(1^n)))$.

Then the **security loss** [92] of $\mathcal{R}$ is defined as:

$$\lambda_{\mathcal{R}}(n) = \max_{\mathcal{A}} \left(\frac{\mathsf{Success}_{\mathcal{A}}(n)}{\mathsf{Success}_{\mathcal{R}^{\mathcal{A}}}(n)} \frac{\mathsf{Time}_{\mathcal{R}^{\mathcal{A}}}(n)}{\mathsf{Time}_{\mathcal{A}}(n)} \right)$$

If there exists polynomial $p(\cdot)$ for which $\lambda_{\mathcal{R}}(n) \leq p(n)$ given sufficiently large $n \in \mathbb{N}$, we call $\mathcal{R}$ **linear-preserving**. If there exists a constant c for which $\lambda_{\mathcal{R}}(n) \leq c$ given sufficiently large $n \in \mathbb{N}$, we call $\mathcal{R}$ **tight**.

2.2.4 Unique Signatures

We here define unique signature schemes, the first of the two primitives from which we rule out linear-preserving reductions. Recall that a signature scheme is a means by which a message can be signed with the signer's secret key and the signature can be verified using a public key. A unique signature scheme, then, is simply a signature scheme for which each message can only have one possible signature:

Definition 5. A **unique signature scheme** is a triple $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ of probabilistic polynomial-time algorithms such that, for every $n \in \mathbb{N}$:

- Gen, on input 1^n, produces a pair (pk, sk)
- Sign, on input (sk, m) for any $m \in \{0,1\}^n$, produces a signature σ. (We write $\sigma \leftarrow \mathsf{Sign}_{sk}(m)$.)
- Ver, on input (pk, m, σ), produces either Accept or Reject. (We write $out \leftarrow \mathsf{Ver}_{pk}(m, \sigma)$.)

and, in addition, the following properties hold:

- *Correctness*: For every $n \in \mathbb{N}$ and $m \in \{0,1\}^n$:

$$\Pr\left[(pk, sk) \leftarrow \mathsf{Gen}(1^n) : \mathsf{Ver}_{pk}(m, \mathsf{Sign}_{sk}(m)) = \mathsf{Accept}\right] = 1$$

- *Uniqueness*: For every $m \in \{0,1\}^*$, and $pk \in \{0,1\}^*$, there exists at most one $\sigma \in \{0,1\}^*$ for which $\mathsf{Ver}_{pk}(m, \sigma) = \mathsf{Accept}$.

We next turn to discussing what it means for such a scheme to be secure. A natural definition of security is the notion of *existential unforgeability against adaptive chosen-message*

attacks [68], which requires that an adversary knowing the public key, even if allowed to adaptively choose a bounded number of messages and observe their signatures, is unable to forge any signature for a message they have not yet queried. We formalize this by allowing the adversary access to an oracle for Sign, as follows:

Definition 6. We say that a signature scheme is **unforgeable** if, for every non-uniform probabilistic polynomial-time oracle-aided algorithm $\mathcal{A}$, there is some negligible function $\epsilon(\cdot)$ such that for all $n \in \mathbb{N}$:

$$\Pr\left[(pk, sk) \leftarrow \mathsf{Gen}(1^n); (m, \sigma) \leftarrow \mathcal{A}^{\mathsf{Sign}_{sk}(\cdot)}(1^n, pk) : \mathsf{Ver}_{pk}(m, \sigma) = \mathsf{Accept} \wedge \mathsf{Valid}\right] \leq \epsilon(n)$$

where Valid is the event that none of $\mathcal{A}$'s queries were for the signature of the output message m.

We will define a weaker notion of a signature scheme being $\ell(\cdot)$-**unforgeable** identically to the above, with the exception that Valid is the event that the following *two* conditions on $\mathcal{A}$ are true:

- $\mathcal{A}$ has queried its oracle at most $\ell(n)$ times.
- None of $\mathcal{A}$'s queries were for the signature of the output message m.

The bounded notion of $\ell(\cdot)$-unforgeability is primarily useful to prove our concrete security loss bound, whereas our main result applies to the general notion of unforgeability. Furthermore, for the purposes of the impossibility result, we weaken the definition of unforgeability to a worst-case definition, as this will strengthen our main theorem (by showing that basing even this weak notion of security on standard assumptions will incur a security loss):

Definition 7. We say that a signature scheme is **weakly unforgeable** (respectively, weakly

$\ell(\cdot)$-unforgeable) if, for every non-uniform probabilistic polynomial-time oracle-aided algorithm $\mathcal{A}$ and every $n \in \mathbb{N}$:

$$\Pr\left[(pk, sk) \leftarrow \mathsf{Gen}(1^n); (m, \sigma) \leftarrow \mathcal{A}^{\mathsf{Sign}_{sk}(\cdot)}(1^n, pk) : \mathsf{Ver}_{pk}(m, \sigma) = \mathsf{Accept} \wedge \mathsf{Valid}\right] < 1$$

where Valid is defined as above (and respectively for $\ell(\cdot)$-unforgeability). In particular, we say that a non-uniform probabilistic polynomial-time algorithm $\mathcal{A}$ *breaks weak unforgeability* of a signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ if the probability above is equal to 1.

2.2.5 Multi-User Secure MACs under Adaptive Corruption

Finally, towards defining the other primitive from which we rule out linear-preserving reductions, we define the notion of a *message authentication code*.

Definition 8. We refer to a tuple of efficient (poly(n)-time) algorithms $\Pi = (\mathsf{Gen}, \mathsf{Tag}, \mathsf{Ver})$, where:

- $\mathsf{Gen}(1^n) \to k$ takes as input a security parameter n and outputs a secret key $k \in \{0,1\}^n$,
- $\mathsf{Tag}_k(m) \to \sigma$ takes as input a secret key k and a message m from some message space $\mathcal{M}_n$ of size super-polynomial in n, and outputs a tag σ for the message, and
- $\mathsf{Ver}_k(m, \sigma) \to \{\mathsf{Accept}, \mathsf{Reject}\}$ takes as input a secret key k, a message m, and a tag σ, and outputs Accept or Reject denoting whether the tag σ is valid for the message m, specifically in such a manner that $\Pr[k \leftarrow \mathsf{Gen}(1^n); \mathsf{Ver}_k(m, \mathsf{Tag}_k(m)) \to \mathsf{Accept}] = 1$ for any valid message $m \in \mathcal{M}_n$,

as a **message authentication code** (MAC). If, in addition, the following properties hold:

- $\mathsf{Tag}_k(m)$ is a deterministic function, and

- $\mathsf{Ver}_k(m, \sigma) \to \mathsf{Accept}$ if and only if $\mathsf{Tag}_k(m) = \sigma$,

then we refer to Π as a **deterministic MAC**.

Note that we focus here on MACs having both an input (message) and output (tag) space superpolynomial in the length of a key (the security parameter n), a property which is satisfied by virtually all standard definitions and constructions.

The traditional notion of security for a MAC states that, given some instance of a MAC (i.e., a secret key $k \leftarrow \mathsf{Gen}(1^n)$), an efficient adversary given an oracle for the Tag algorithm is unable to forge a valid tag for a new message (i.e., return a pair (m, σ) where $\mathsf{Ver}_k(m, \sigma) \to \mathsf{Accept}$) without having queried a tag for that message using the oracle. Our definition of multi-user security with adaptive corruption expands this to a polynomial number $\ell(n)$ of instances of the MAC, and allows the adversary to make *key-opening* queries (i.e., to "corrupt" an instance and recover its key) in addition to tag queries; the adversary wins if they produce a valid forgery (m, σ) for some instance without having either queried the tag for m on that instance or corrupted the instance itself. Formally:

Definition 9. A MAC $\Pi = (\mathsf{Gen}, \mathsf{Tag}, \mathsf{Ver})$ is an $\ell(n)$-**key unforgeable MAC under adaptive corruption** (or **adaptively** $\ell(n)$-**key unforgeable**) if, for any interactive oracle-aided non-uniform probabilistic polynomial-time algorithm $\mathcal{A}$, there is a negligible function $\epsilon(\cdot)$ such that, for all $n \in \mathbb{N}$,

$$\Pr\left[\langle \mathcal{A}, \mathcal{C}_\Pi^{\ell(n)} \rangle(1^n) = \mathsf{Accept}\right] \leq \epsilon(n)$$

where $\mathcal{C}_\Pi^{\ell(n)}$ is the interactive challenger that does as follows on input 1^n:

- Let $(k_1, \ldots, k_{\ell(n)}) \leftarrow \mathsf{Gen}(1^n)$. Initialize empty transcript τ.

- Upon receiving a *tag* query (Query, i, m) for $i \in [\ell(n)]$, append $((\mathsf{Query}, i, m), \mathsf{Tag}_{k_i}(m))$ to τ and send τ.
- Upon receiving a *key-opening* query (Open, i) for $i \in [\ell(n)]$, append the tuple $((\mathsf{Open}, i), k_i)$ to τ and send τ.
- Upon receiving a forgery (m^*, σ^*, i^*) from $\mathcal{A}$, output Reject if one of the following three conditions is true:
  - τ contains a key opening query (Open, i^*).
  - τ contains an oracle query $(\mathsf{Query}, i^*, m^*)$.
  - $\mathsf{Ver}_{k_{i^*}}(m^*, \sigma^*) \to \mathsf{Reject}$.
- Otherwise, output Accept.

We call a MAC Π an **adaptively multi-key unforgeable MAC** if it is adaptively $\ell(n)$-key unforgeable for every polynomial $\ell(\cdot)$.

For syntactic clarity, we will assume that a machine interacting with a multi-key MAC adversary will begin interaction with a new instance of the adversary by sending a special message (Init, s), where s is the "identifier" for the instance, and communicate with the adversary by sending a partial transcript and receiving a next message as described above for oracle interaction.

2.3 Impossibility of Linear-Preserving Reductions from Unique Signatures

The first major impossibility result we show rules out linear-preserving black-box reductions from the unforgeability of unique signature schemes to bounded-round assumptions.

Formally:

Theorem 1. Let $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ be a unique signature scheme, and let $(\mathcal{C}, t(\cdot))$ be some $r(\cdot)$-round intractability assumption for polynomial $r(\cdot)$. If there exists some fixed-parameter black-box reduction $\mathcal{R}$ for basing weak unforgeability of Π on the hardness of $(\mathcal{C}, t(\cdot))$, then either:

(1) $\mathcal{R}$ is not a linear-preserving reduction, or

(2) there exists a polynomial-time adversary $\mathcal{B}$ that breaks the assumption $(\mathcal{C}, t(\cdot))$.

We note that this result also applies to the slightly more general notion of *rerandomizable* signatures through an almost identical argument; we refer the interested reader to the full version of the work [98]. Theorem 1 follows in a straightforward manner from the following lemma, which is a concrete security loss bound analogous to Coron's in [42], but generalized so that it handles arbitrary (i.e., not just simple) reductions:

Lemma 1. Let $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ be a unique signature scheme, and let $(\mathcal{C}, t(\cdot))$ be some $r(\cdot)$-round intractability assumption for polynomial $r(\cdot)$. If for some polynomial $\ell(\cdot)$ there exists some fixed-parameter black-box reduction $\mathcal{R}$ for basing weak $\ell(\cdot)$-unforgeability of Π on the hardness of $(\mathcal{C}, t(\cdot))$, then either $\mathcal{R}$'s security loss is at least

$$\lambda_{\mathcal{R}}(n) \geq \sqrt{\ell(n)} - (r(n) + 1)$$

for all sufficiently large $n \in \mathbb{N}$, or there exists a polynomial-time adversary $\mathcal{B}$ that breaks the assumption $(\mathcal{C}, t(\cdot))$.

First, we show that Lemma 1 implies Theorem 1:

Proof. Given Lemma 1, assume for the sake of contradiction that $(\mathcal{C}, t(\cdot))$ is secure, and that the $\mathcal{R}$ described in Theorem 1 is linear-preserving. Then there exists a polynomial $p(\cdot)$ such that $\lambda_{\mathcal{R}}(n) \leq p(n)$ for all sufficiently large n. Because $\mathcal{R}$ by assumption is a reduction from (unbounded) weak unforgeability, it must likewise be a reduction from the strictly more specific definition of weak $\ell(\cdot)$-unforgeability for any polynomial $\ell(\cdot)$, since every weak $\ell(\cdot)$-unforgeability adversary will trivially break unbounded weak unforgeability. So, let $\ell(n) \triangleq (p(n) + r(n) + 2)^2$; hence, by Lemma 1, either there exists a polynomial-time adversary $\mathcal{B}$ which breaks $(\mathcal{C}, t(\cdot))$—which is false by assumption—or the security loss of $\mathcal{R}$ is no less than

$$\sqrt{\ell(n)} - (r(n) + 1) = p(n) + 1 > p(n)$$

for all sufficiently large n, which is a contradiction and hence proves Theorem 1. □

Hence, the remainder of the section is dedicated to proving Lemma 1.

2.3.1 Technical Overview

Our proof of Lemma 1 follows four major steps, which we shall describe here at a high level before beginning the full argument.

Constructing an Ideal Adversary. First, we describe an "ideal" adversary $\mathcal{A}$ which is *guaranteed* to break the security of Π while sending $\ell(n)$ queries, but does so by brute force and hence does not run in polynomial time. Our objective then is to create a meta-reduction $\mathcal{B}$ that *almost* always emulates the interaction $\mathcal{R}^{\mathcal{A}}$ between $\mathcal{R}$ and $\mathcal{A}$. If it does so with, say, probability $1 - 1/p(n)$, then $\mathcal{B}$ will break the assumption $(\mathcal{C}, t(\cdot))$ with probability at least $\mathsf{Success}_{\mathcal{R}^{\mathcal{A}}}(n) - 1/p(n)$. However, this means that $\mathcal{R}^{\mathcal{A}}$ itself cannot have success

probability non-negligibly greater than $1/p(n)$; otherwise, $\mathcal{C}$ would be broken with non-negligible probability by $\mathcal{B}$.

The "ideal" $\mathcal{A}$ will pick $\ell(n)$ messages ($\overrightarrow{m}$) at random and query $\mathcal{R}$ for the signatures of each of these messages in turn, and will finally brute-force a secret key from the results and use that key to forge a signature for another random message m^*, which it will return. Crucially, $\mathcal{A}$ will also verify $\mathcal{R}$'s responses to its queries and return $\perp$ instead if not all are correct signatures. By construction $\mathcal{A}$ breaks $\ell(\cdot)$-weak unforgeability; however, due to the brute-force step, it (and consequently $\mathcal{R}^{\mathcal{A}}$) will not run in polynomial time.

Constructing a Meta-Reduction. Hence, to *efficiently* emulate $\mathcal{R}^{\mathcal{A}}$, we create the meta-reduction $\mathcal{B}$. $\mathcal{B}$ will run $\mathcal{R}$ and forward communicate with $\mathcal{C}$ as normal; when $\mathcal{R}$ would start a new instance of its adversary $\mathcal{A}$, $\mathcal{B}$ will generate messages $\overrightarrow{m}$ and m^* randomly (i.e., identically to $\mathcal{A}$) and forward queries to $\mathcal{R}$ in the same manner as $\mathcal{A}$. However, when $\mathcal{R}$ requires an instance to provide a forged signature, $\mathcal{B}$ will also "rewind" the simulated execution to the start of each query for that instance and try to query $\mathcal{R}$ with m^* instead of the message it would normally query. If $\mathcal{R}$ gives a response to the rewound query, then $\mathcal{B}$ has (efficiently) found a forgery for m^*, which it can return to $\mathcal{R}$ when it requests a forgery from the corresponding instance of $\mathcal{A}$.

$\mathcal{B}$, while rewinding, will abort (and try rewinding the next slot instead) if either $\mathcal{R}$ would communicate externally with $\mathcal{C}$ (which $\mathcal{B}$ of course cannot rewind) or $\mathcal{R}$ would request a forgery for some other simulated instance of $\mathcal{A}$ during the simulated execution of $\mathcal{R}^{\mathcal{A}}$ (i.e., before responding to the rewound query). In particular, this strategy ensures that recursive rewinding as in [110] will not be required, since $\mathcal{B}$ will never attempt to start rewinding some instance while rewinding a different one.

Furthermore, $\mathcal{B}$ will "verify" all of $\mathcal{R}$'s responses to its signature queries (in the non-rewound part of the execution) in the same manner as $\mathcal{A}$, likewise returning $\perp$ from the simulated instance of $\mathcal{A}$ if not all responses are valid. So, whenever either $\mathcal{R}$ gives a simulated instance one or more incorrect responses or $\mathcal{B}$ successfully extracts a forgery (noting that, by the uniqueness property, the forgeries they return must be identical, which is crucial), $\mathcal{A}$ and $\mathcal{B}$'s simulations of $\mathcal{A}$ will be identically distributed to one another.

Bounding the Failure Probability. So, to bound the probability with which $\mathcal{B}$ does *not* successfully emulate some instance of $\mathcal{A}$, we must bound the probability that all of $\mathcal{B}$'s queries to $\mathcal{R}$ ($\overrightarrow{m}$) are correctly answered, yet the rewinding of *every one* of the queries fails due to either $\mathcal{R}$ responding badly to m^*, $\mathcal{R}$ communicating externally with $\mathcal{C}$, or a forgery request for another simulated instance of $\mathcal{A}$ occurring before $\mathcal{R}$ responds.

We bound this probability by using a counting argument similar to that exhibited in the introduction. In particular, we consider the messages $\overrightarrow{m}$ and m^* for a particular instance, fixing the randomness outside of that instance arbitrarily. Then we show that, for any "bad" sequencing of these messages such that the non-rewound execution succeeds but every rewinding fails, many (though not all, because of the possibility for rewindings to fail due to an end message or external communication) of the rewindings of this sequence will correspond to "good" sequences where $\mathcal{B}$ returns $\perp$ due to receiving an incorrect response from $\mathcal{R}$.

What we intuitively show is that, in every set of $\ell(n) + 1$ sequences corresponding to a sequence and its various rewindings, at most $M(n) + r(n) + 1$ can be bad (since, informally, given a bad sequence, in expectation only $M(n) + r(n)$ of its rewindings can fail for reasons besides $\mathcal{R}$ responding incorrectly, i.e., due to nested end messages or external communication), where $M(n)$ is the maximum number of instances of $\mathcal{A}$ which $\mathcal{R}$ executes (which we

show by our construction of $\mathcal{A}$ must be no less than the number of successful end messages). Hence we obtain a bound of

$$\frac{M(n) + r(n) + 1}{\ell(n) + 1}$$

on the failure probability for each instance, which by the union bound over all $M(n)$ instances sums to an overall failure probability of less than

$$M(n) \left(\frac{M(n) + r(n) + 1}{\ell(n)} \right)$$

Bounding the Security Loss. This does not immediately imply a bound on the security loss $\lambda_{\mathcal{R}}(n)$, since $M(n)$ can be arbitrarily large. However, as in the technical overview, we bound the security loss by showing that, if $M(n)$ is large, this requires a large enough running time of $\mathcal{R}$ that we still obtain a non-trivial lower bound on the security loss. Specifically, recalling that

$$\lambda_{\mathcal{R}}(n) \geq \frac{\mathsf{Success}_{\mathcal{A}}(n)}{\mathsf{Success}_{\mathcal{R}^{\mathcal{A}}}(n)} \frac{\mathsf{Query}_{\mathcal{R}^{\mathcal{A}}}(n)}{\mathsf{Query}_{\mathcal{A}}(n)}$$

we notice first that $\mathsf{Query}_{\mathcal{R}^{\mathcal{A}}}(n)/\mathsf{Query}_{\mathcal{A}}(n) \geq M(n)$, which follows (with some subtleties which we defer to the main proof) from the fact that $\mathcal{R}$ will in the worst case run $M(n)$ instances of $\mathcal{A}$. So, since $\mathsf{Success}_{\mathcal{A}}(n) = 1$ by construction, and since, as we discussed previously, $\mathsf{Success}_{\mathcal{R}^{\mathcal{A}}}(n)$ cannot be non-negligibly larger than the failure probability of $\mathcal{B}$, we have

$$\lambda_{\mathcal{R}}(n) \geq \frac{\ell(n)}{M(n) + r(n) + 1}$$

which immediately implies the bound when $M(n) < \sqrt{\ell(n)} - (r(n) + 1)$ (and so $\lambda_{\mathcal{R}}(n) > \sqrt{\ell(n)}$). On the other hand, we also know that $\mathsf{Success}_{\mathcal{R}^{\mathcal{A}}}(n) \leq 1$ trivially, and so it is also the case that $\lambda_{\mathcal{R}}(n) \geq M(n)$, which implies the bound when $M(n) \geq \sqrt{\ell(n)} - (r(n) + 1)$, completing the proof of Lemma 1 and hence Theorem 1.

2.3.2 Proof: The "Ideal" Adversary

We now proceed to the formal proof of Lemma 1. First, we exhibit an inefficient adversary $\mathcal{A}$ that will break weak $\ell(\cdot)$-unforgeability, so that we can later construct an efficient $\mathcal{B}$ to simulate it while running $\mathcal{R}$ in order to break the assumption $(\mathcal{C}, t(\cdot))$.

Let $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ be a unique signature scheme, and let $(\mathcal{C}, t(\cdot))$ be some $r(\cdot)$-round intractability assumption for polynomial $r(\cdot)$. Assume that there exists some black-box reduction $\mathcal{R}$ for basing weak $\ell(\cdot)$-unforgeability of Π on the hardness of $(\mathcal{C}, t(\cdot))$ which, given an oracle breaking weak unforgeability, will break $(\mathcal{C}, t(\cdot))$ with probability $1/p(\cdot)$ for some polynomial $p(\cdot)$.

First, for any polynomial $\ell(n)$, we construct an inefficient but easily emulatable adversary $\mathcal{A}$ which sends at most $\ell(n)$ queries and is guaranteed to break weak unforgeability of Π. Since we will require $\mathcal{A}$ to be deterministic during execution yet generate random messages, we will assume that $\mathcal{A}$ is formally given by a deterministic interactive $\mathcal{A}^{\mathcal{O}}$ which has access to a random oracle $\mathcal{O}$ (of course, $\mathcal{O}$ is not needed for our actual constructions, as we shall emulate $\mathcal{A}$), which, as in [110], is given by a random variable which is uniformly distributed over functions $\{0,1\}^* \to \{0,1\}^\infty$. In particular, this ensures that the queries output by $\mathcal{A}^{\mathcal{O}}$ are uniformly distributed (i.e., over the randomness of $\mathcal{O}$), but are still preserved under rewinding.

We shall henceforth denote by $\mathcal{A}$ the specific adversary $\mathcal{A}^{\mathcal{O}}$ which, on input 1^n, behaves as described in Figure 2.1. Informally, $\mathcal{A}$ makes $\ell(n)$ signature queries, generating the message for each query by applying $\mathcal{O}$ to the current partial transcript. Finally, after receiving responses for each query, $\mathcal{A}$ returns a brute-forced forgery, but only if it successfully "verifies" the transcript by ensuring that each query's response is valid and that each query in

- Initially, receive a message pk, the public key; respond (i.e., generate m_1) according to the next step for $i = 1$.
- On receiving a message consisting of a partial transcript $\tau = (pk, m_1, \sigma_1, \cdots, m_{i-1}, \sigma_{i-1})$ for some $i \in [\ell(n)]$, do the following:
  - Generate m_i by taking the first n bits resulting from applying the oracle $\mathcal{O}$ to τ.
  - Return the new partial transcript $\tau || m_i$.
- On receiving a message consisting of a complete transcript $\tau = (pk, m_1, \sigma_1, \cdots, m_{\ell(n)}, \sigma_{\ell(n)})$ (we shall refer to such a message as a "forgery request" or "end message"), do the following:
  - Verify that, for each signature σ_i, $\mathsf{Ver}_{pk}(m_i, \sigma_i) = \mathsf{Accept}$. If not true for all i, return $\perp$.
  - Verify that, for each message m_i, m_i is equal to the first n bits resulting from applying the oracle $\mathcal{O}$ to the prefix transcript $\tau_{<i} = (pk, m_1, \sigma_1, \cdots, m_{i-1}, \sigma_{i-1})$. If not true for all i, then return $\perp$.
  - Finally, generate a random message m^* (distinct from each m_i in τ) by applying $\mathcal{O}$ to the transcript τ, use brute force to find a signature σ^* for which $\mathsf{Ver}_{pk}(m^*, \sigma^*) = \mathsf{Accept}$, and return the forgery (m^*, σ^*).

Figure 2.1: Formal description of the "ideal" adversary $\mathcal{A}^{\mathcal{O}}$.

the transcript was generated in the correct manner (i.e., by $\mathcal{O}$ applied to the prior partial transcript).

It is straightforward to see that $\mathcal{A}$, given any fixed oracle $\mathcal{O}$, will break weak $\ell(\cdot)$-unforgeability; given an honest signing oracle (which will always send the correct partial transcript), it will always return some (m, σ) such that $\mathsf{Ver}_{pk}(m, \sigma) = \mathsf{Accept}$, m was not queried (as m^* is not equal to any of the queries m_i), and only $\ell(n)$ queries were made.

However, when interacting with $\mathcal{R}$, which is not bound by the rules of an honest oracle, the transcript verification is necessary to prevent $\mathcal{R}$ from "cheating" in certain ways during its interaction. First, we wish to ensure that $\mathcal{R}$ will return valid signatures to queries as often as possible. Also, we wish to ensure that $\mathcal{R}$ is actually required to answer $\ell(n)$

signature queries generated randomly by $\mathcal{A}$ and cannot, for instance, immediately send $\mathcal{A}$ an end message with an artificially generated transcript; this is done by using the oracle $\mathcal{O}$ to generate $\mathcal{A}$'s messages and ensuring that the transcript is consistent with the oracle. Formally, we make the following claim, which will be useful later:

Claim 1. There exists a negligible function $\nu(\cdot)$ such that, for all $n \in \mathbb{N}$, the probability, over all randomness in the experiment $[\mathcal{R}^{\mathcal{A}^{\mathcal{O}}} \leftrightarrow \mathcal{C}](1^n)$, that some instance of $\mathcal{A}$ returns a forgery (i.e., something besides $\perp$) to $\mathcal{R}$ without having received $\ell(n)$ different responses to its signature queries from $\mathcal{R}$, is less than $\nu(n)$.

Proof. Assume that $\mathcal{A}$ returns a forgery to $\mathcal{R}$ after receiving fewer than $\ell(n)$ responses to its signature queries; we shall demonstrate that the only way this can happen is if $\mathcal{R}$ correctly predicts the output of the random oracle $\mathcal{O}$, which can happen with only negligible probability (specifically, no greater than $\nu(n) \triangleq \ell(n)2^{-n}$) due to the uniformly random choice of $\mathcal{O}$.

Let $i \in [\ell(n) - 1]$ (we exclude $\ell(n)$ because $\mathcal{R}$ by assumption makes a forgery request) be the index of the first signature query "skipped" by $\mathcal{R}$—that is, the first index for which $\mathcal{R}$ does *not* send $\mathcal{A}$ a partial transcript $\tau_{\leq i} = (pk, m_1, \sigma_1, \cdots, m_i, \sigma_i)$. Of course, there must exist such an i by the assumption that $\mathcal{R}$ sends $\mathcal{A}$ fewer than $\ell(n)$ messages. In order to receive a forgery from $\mathcal{A}$ (and not $\perp$), $\mathcal{R}$ must send a complete transcript $\tau = (pk, m_1, \sigma_1, \cdots, m_{\ell(n)}, \sigma_{\ell(n)})$, notably having the property that each message m_j is equivalent to the first n bits of the result of applying the random oracle $\mathcal{O}$ to the prior transcript $m_{\leq j-1}$.

In particular, since by assumption $\mathcal{R}$ did not send $\mathcal{A}$ any partial transcript $\tau_{\leq i}$, it is impossible for $\mathcal{R}$ to have received from $\mathcal{A}$ the correct message m_{i+1}, which can, by construction of $\mathcal{A}$, only be retrieved by sending $\mathcal{A}$ the (unique) partial transcript $\tau_{\leq i}$ for which all m_j are generated according to $\mathcal{O}$ and all σ_j are the (unique) accepting signature for the respective

m_j.

By our assumption, then, $\mathcal{R}$ will only be able to send the correct message m_{i+1} in its forgery request, and hence receive a forgery from $\mathcal{A}$, with probability 2^{-n}—that is, the chance that the output of the (uniformly distributed) random oracle $\mathcal{O}$ matches the sent m_{i+1} in the first n bits. And as, by our assumption, $\mathcal{R}$ sends fewer than $\ell(n)$ messages to $\mathcal{A}$, it cannot attempt to "guess" the correct m_{i+1} more than $\ell(n)$ times; thus the probability that $\mathcal{A}$ will return a forgery, taken over the randomly distributed $\mathcal{O}$, is by the union bound no more than $\nu(n) = \ell(n)2^{-n}$, which as desired is negligible. □

Furthermore, this construction of $\mathcal{A}$ (using the oracle $\mathcal{O}$) allows us to assume, without loss of generality, that the reduction $\mathcal{R}$ will never rewind an instance of $\mathcal{A}$—this is without loss of generality because there is a single accepting transcript for each choice of the oracle $\mathcal{O}$. Namely, given an oracle $\mathcal{O}$, if $\mathcal{R}$ always provides correct signatures, then $\mathcal{A}$'s messages (including the forgery it returns) and $\mathcal{R}$'s responses are fully determined by $\mathcal{O}$ and the uniqueness property of Π. Meanwhile, if $\mathcal{R}$ does not provide correct signatures, $\mathcal{A}$ will not return a forgery.

Because $\mathcal{A}$ breaks unforgeability, and by the assumed properties of $\mathcal{R}$ and the determinism of $\mathcal{A}^{\mathcal{O}}$ for any fixed oracle $\mathcal{O}$, it must be true that there exists polynomial $p(\cdot)$ such that

$$\Pr\left[\langle \mathcal{R}^{\mathcal{A}^{\mathcal{O}}}, \mathcal{C}\rangle(1^n) = \mathsf{Accept}\right] \geq t(n) + \frac{1}{p(n)}$$

for any oracle $\mathcal{O}$. As such, by the fact that $\mathcal{R}$ is fixed-parameter, we can observe that, for any n, averaging this probability over all possible oracles $\mathcal{O}$, we likewise have

$$\Pr\left[\langle \mathcal{R}^{\mathcal{A}}, \mathcal{C}\rangle(1^n) = \mathsf{Accept}\right] \geq t(n) + \frac{1}{p(n)}$$

even though $\mathcal{A}$ over a randomly-chosen $\mathcal{O}$ is not deterministic.

Of course, $\mathcal{A}$ is inefficient, so, in order to break the assumption $(\mathcal{C}, t(\cdot))$, we must construct an efficient $\mathcal{B}$ that is able to run $\mathcal{R}$ while emulating its interactions with $\mathcal{A}$ most of the time. Hence, the remainder of the proof will be dedicated to constructing this meta-reduction and analyzing the probability with which it succeeds in emulating the "ideal" $\mathcal{A}$. Intuitively, if $\mathcal{B}$ successfully emulates $\mathcal{A}$ at least $1 - 1/p'(n)$ of the time for some function $p'(\cdot)$, then:

$$\left|\Pr\left[\langle \mathcal{R}^{\mathcal{A}}, \mathcal{C}\rangle(1^n) = \mathsf{Accept}\right] - \Pr\left[\langle \mathcal{B}, \mathcal{C}\rangle(1^n) = \mathsf{Accept}\right]\right| \leq \frac{1}{p'(n)}$$

$$\Pr\left[\langle \mathcal{B}, \mathcal{C}\rangle(1^n) = \mathsf{Accept}\right] \geq t(n) + \frac{1}{p(n)} - \frac{1}{p'(n)}$$

meaning that $\mathcal{B}$ must break $\mathcal{C}$ with probability at least $1/p(n) - 1/p'(n)$, as desired. Hence, what we shall effectively show in the subsequent steps is that, unless the security loss of $\mathcal{R}$ is large, $1/p'(n)$ will be non-negligibly smaller than $1/p(n)$, and thus $\mathcal{B}$ will break the security of $(\mathcal{C}, t(\cdot))$.

Slots. As a notational aside, we shall for simplicity henceforth refer to the pair of a query made by $\mathcal{A}$ (or something, such as $\mathcal{B}$, which emulates $\mathcal{A}$) and its corresponding response by $\mathcal{R}$ as a *slot* (v_{open}, v_{close}). Such a slot is determined by two views: the "opening" of the slot, or the view v_{open} of the execution of $\mathcal{R}$ immediately before $\mathcal{A}$'s query to $\mathcal{R}$, and the "closing" of the slot, or the view v_{close} of the execution immediately after $\mathcal{R}$ responds to the respective query. (We will also often refer to the view of $\mathcal{R}$ immediately *after* the opening query of a message m, which we shall denote by the concatenation $v_{open}||m$.)

2.3.3 The Meta-Reduction

We next construct the meta-reduction $\mathcal{B}$ which will efficiently emulate $\mathcal{A}$. Let $\mathcal{B}$ be as described formally in Figure 2.2; informally, $\mathcal{B}$ will run $\mathcal{R}$ internally, forwarding communication

to $\mathcal{C}$ as $\mathcal{R}$ would while also internally simulating instances of $\mathcal{A}$ interacting with $\mathcal{R}$. The primary difference between $\mathcal{B}$ and the "ideal" execution of $\mathcal{A}$ interacting with $\mathcal{R}$ is that $\mathcal{B}$, being restricted to polynomial time, cannot brute-force forgeries as $\mathcal{A}$ does; instead, while simulating each instance of $\mathcal{A}$, $\mathcal{B}$ will select at random a message m^* for which to forge a signature and attempt to rewind each slot for that instance, substituting m^* for the original message.[2]

If $\mathcal{R}$ ever returns a valid signature σ^* for m^*, then $\mathcal{B}$ may store that signature and finally return (m^*, σ^*) when $\mathcal{R}$ requests a forgery for that instance. However, if one of the following "bad events" occurs:

- $\mathcal{R}$ fails to return a valid signature of m^*.
- $\mathcal{R}$ asks for a forgery for another instance before returning a signature.
- $\mathcal{R}$ requires external communication with $\mathcal{C}$ (which cannot be rewound) before returning a signature.

then $\mathcal{B}$ will abort and try the next slot. In this way we circumvent the issue of having to recursively rewind nested end messages as in [110].

First, we can show that $\mathcal{B}$, unlike $\mathcal{A}$, is efficient:

Claim 2. There exists a polynomial $t(n)$ such that, for all $n \in \mathbb{N}$, $\mathsf{Real}(1^n)$ is guaranteed to run in time at most $t(n)$.

Proof. It suffices to show that $\mathcal{B}$ sends a polynomial number of simulated messages to $\mathcal{R}$, as $\mathcal{R}$ and $\mathcal{C}$ run in polynomial time by definition and all other operations performed by $\mathcal{B}$ are clearly

[2]That is, when "rewinding" a slot (v_{open}, v_{close}), $\mathcal{B}$ will simulate interaction with $\mathcal{R}$ starting from the view $v_{open}||m^*$.

- Set initial view $v \leftarrow \perp$ and set $k \leftarrow 1$. Execute $\mathcal{R}$, updating the current view v according to the following rules.
- When $\mathcal{R}$ begins a new instance of $\mathcal{A}$ and sends a public key pk, label this instance as instance k. Generate and store $\ell(n)$ random queries $\overrightarrow{m}_k = (m_{k,1}, \ldots, m_{k,\ell(n)})$ and a target forgery m_k^*. (Abort and return Fail if m_k^* is equal to a message in $\overrightarrow{m}_k$.) Also let $pk_k \leftarrow pk$ and initialize the forgery $f_k \leftarrow \{\}$. Lastly, respond with $\tau_k^* = pk_k || m_{k,1}$ and increment k.
- When $\mathcal{R}$ attempts to communicate externally with $\mathcal{C}$, forward the message, return $\mathcal{C}$'s response to $\mathcal{R}$, and update v accordingly.
- When $\mathcal{R}$ sends a transcript $\tau = (pk, m_{I,1}, \sigma_{I,1}, \cdots, m_{I,j}, \sigma_{I,j})$ to some simulated instance I of $\mathcal{A}$, store the signature $\sigma_{I,j}$ and do the following:
  - If $j = \ell(n)$ (i.e., this is an end message), then do the following:
    - If τ is an inconsistent transcript (i.e., $m_{I,i}$ or $\sigma_{I,i}$ in τ is different from the stored $m_{I,i}$ or $\sigma_{I,i}$ (respectively) for some $i \in [\ell(n)]$, or not all $\sigma_{I,i}$ have been stored) or $\mathcal{R}$'s response to some signature query j was invalid (i.e., $\mathsf{Ver}_{pk_I}(m_{I,i}, \sigma_{I,i}) = \mathsf{Reject}$ for some $i \in [\ell(n)]$), then return $\perp$.
    - Otherwise, if f_I is still empty (i.e., not $\perp$), run the procedure Rewind detailed below for the instance I.
    - If, at this point, there is a stored forgery $f_I = (m_I^*, \sigma_I^*)$, then return it and continue executing $\mathcal{R}$ as above. Otherwise, abort the entire execution of $\mathcal{B}$ and return Fail.
  - If the response is an invalid signature (i.e., $\mathsf{Ver}_{pk_I}(m_{I,j}, \sigma_{I,j}) = \mathsf{Reject}$), store $f_I \leftarrow \perp$.
  - Lastly, respond with $\tau || m_{I,j+1}$ and continue the execution of $\mathcal{R}$.

Rewind procedure:

- Given instance I, for $j \in [\ell(n)]$ let $(v_{open}^j, v_{close}^j)$ denote the slot corresponding to the j^{th} signature query for instance I.
- For each $j \in [\ell(n)]$, "rewind" the slot $(v_{open}^j, v_{close}^j)$ as follows: Let $k' \leftarrow k$, and begin executing $\mathcal{R}$ from the view $v' = v_{open}^j || m_I^*$ as in the main routine, with the following exceptions:
  - When $\mathcal{R}$ begins a new instance of $\mathcal{A}$, label this instance as instance k' and increment k'. (That is, continue creating new instances, but preserve the counter k in the outer execution for after the rewinding.)
  - When $\mathcal{R}$ attempts to communicate externally with $\mathcal{C}$, abort the rewinding and continue to the next j.
  - When $\mathcal{R}$ sends an end message for an instance $I' \neq I$ of $\mathcal{A}$, abort the rewinding and continue to the next j, *unless* $\mathcal{R}$ has not sent responses to $\ell(n)$ signature queries for that instance (in which case respond with $\perp$).
  - If v' ever contains a message whose transcript contains a response σ_I^* to the query for m_I^*, then, if it is the case that $\mathsf{Ver}_{pk_I}(m_I^*, \sigma_I^*) = \mathsf{Accept}$, store $f_i \leftarrow (m_I^*, \sigma_I^*)$ and end the Rewind procedure (i.e., return to the outer execution); otherwise, if $\mathsf{Ver}_{pk_I}(m_I^*, \sigma_I^*) = \mathsf{Reject}$, store nothing to f_I and continue to the next j.

Figure 2.2: Formal description of the meta-reduction $\mathcal{B}$.

polynomial-time. To that end, we observe that each instance I contains $O(\ell(n)+r(n))$ rounds of communication (including both communication with $\mathcal{R}$ and $\mathcal{R}$'s external communication with $\mathcal{C}$). Furthermore, there are at most a polynomial number of instances of $\mathcal{A}$ started by $\mathcal{R}$—call this number $M(n)$—and so disregarding rewinding we know that there must be $O((\ell(n) + r(n))M(n))$ total messages.

$\mathcal{B}$ will rewind $\mathcal{R}$ as described above; however, we know that there can be at most $\ell(n)M(n)$ rewound slots (as each slot is rewound once and $\mathcal{B}$ will never rewind recursively), each of which might contain $O((\ell(n) + r(n))M(n))$ nested messages; hence, the total number of messages is bounded by $O(\ell(n)(\ell(n) + r(n))M(n)^2)$, which is still polynomial. The claim follows from defining $t(n)$ accordingly.

(Note that, while we assume without loss of generality that $\mathcal{R}$ will never rewind $\mathcal{A}$, such rewinding even if it did occur could at most multiply the running time by a polynomial factor since $\mathcal{R}$ is restricted to polynomial time.) □

Next, to reason about the failure probability of $\mathcal{B}$ (and through it the success probability of $\mathcal{R}^{\mathcal{A}}$), let us define the following experiments:

- Let $\mathsf{Ideal}(1^n)$ denote $[\mathcal{R}^{\mathcal{A}} \leftrightarrow \mathcal{C}](1^n)$—that is, the experiment where $\mathcal{R}(1^n)$, using the "ideal" adversary $\mathcal{A}(1^n)$ as a black box, communicates with $\mathcal{C}(1^n)$.
  - When we refer to probabilities in the context of this experiment, they are taken over a uniform distribution over random oracles $\mathcal{O}$ (which results in uniformly distributed messages $\overrightarrow{m}_I$ and m_I^*) for each instance I of $\mathcal{A}$ started by $\mathcal{R}$.
  - When we wish to fix the randomness of a particular execution of Ideal, we will denote this with $\mathsf{Ideal}_{\{\mathcal{O}_I\}_{I\in[M(n)]},\mathcal{O}_{ext}}(1^n)$, or, for more clarity,

$\mathsf{Ideal}_{\{\overrightarrow{m}_I, m_I^*\}_{I \in [M(n)]}, \mathcal{O}_{ext}}(1^n)$. $\overrightarrow{m}_I$ and m_I^* are the messages and forgery generated for instance I by each oracle $\mathcal{O}_I$ given the (deterministic) prefix; $\mathcal{O}_{ext}$ is a random variable representing the random coins used by $\mathcal{R}$ and $\mathcal{C}$, containing a number of bits equal to the maximum number of coins needed (which must be polynomially many since $\mathcal{R}$ and $\mathcal{C}$ are polynomial-time). When all $\overrightarrow{m}_I$ and m_I^* are fixed, and $\mathcal{O}_{ext}$ is fixed, note that the execution of Ideal is deterministic for each instance.

- Let $\mathsf{Real}(1^n)$ denote $[\mathcal{B} \leftrightarrow \mathcal{C}](1^n)$—that is, the "real" experiment where $\mathcal{B}(1^n)$ communicates directly with $\mathcal{C}(1^n)$ by attempting to simulate the interaction between $\mathcal{A}(1^n)$ and $\mathcal{R}(1^n)$ while forwarding any external communications.

  - When we refer to probabilities in the context of this experiment, they are taken over uniformly distributed messages $\overrightarrow{m}_I$ and m_I^* for each simulated instance I of $\mathcal{A}$ started by $\mathcal{R}$ in the context of $\mathcal{B}$.
  - When we wish to fix the randomness of a particular execution of Real, we will again denote this with the notation $\mathsf{Real}_{\{\overrightarrow{m}_I, m_I^*\}_{I \in [M(n)]}, \mathcal{O}_{ext}}(1^n)$, where $\overrightarrow{m}_I$ and m_I^* are the messages and forgery for each simulated instance I of $\mathcal{A}$ and $\mathcal{O}_{ext}$ is again a random variable representing the random coins used by $\mathcal{R}$ and $\mathcal{C}$. When all $\overrightarrow{m}_I$ and m_I^* are fixed, and $\mathcal{O}_{ext}$ is fixed, note that the execution of Real is deterministic for each instance, just as with Ideal.
  - Furthermore, we may opt to isolate a particular simulated instance k by fixing all randomness *except for* that instance's; we denote this by $\mathsf{Real}^*_{\{\overrightarrow{m}_I, m_I^*\}_{-k}, \mathcal{O}_{ext}}(1^n)$ and note that the probability space in this altered experiment is over uniformly distributed $\overrightarrow{m}_k$ and m_k^*. Further note that in experiment Real^* the execution up to the start of the isolated instance k is deterministic, as is the execution for any choice of $\overrightarrow{m}_k$ and m_k^*.

- In all experiments, we will denote the *view*, or *execution*, as the transcript of all messages sent between, and all randomness consumed by, real or simulated machines (i.e., between $\mathcal{R}$, $\mathcal{A}$ or $\mathcal{B}$'s simulation of $\mathcal{A}$, and $\mathcal{C}$). We will notate this using just the notation for the experiment, e.g., $\mathsf{Real}(1^n)$.
- In all experiments, we will denote the *result*, or *output*, as either the final output of $\mathcal{C}$ (either Accept or Reject) if $\mathcal{C}$ finishes, or as Fail if $\mathcal{C}$ does not finish (i.e., when $\mathcal{B}$ aborts returning Fail). This will be notated by Output, e.g., $\mathsf{Output}[\mathsf{Real}(1^n)] = \mathsf{Accept}$.

2.3.4 Analyzing the Meta-Reduction

Using this terminology, we wish to show that $\mathsf{Real}(1^n)$ is identically distributed to $\mathsf{Ideal}(1^n)$ with high probability. To that end, we make the following claim:

Claim 3. For all $n \in \mathbb{N}$:

$$|\Pr[\mathsf{Output}[\mathsf{Real}(1^n)] = \mathsf{Accept}] - \Pr[\mathsf{Output}[\mathsf{Ideal}(1^n)] = \mathsf{Accept}]| \leq \Pr[\mathsf{Output}[\mathsf{Real}(1^n)] = \mathsf{Fail}]$$

Proof. The probability of each possible assignment of the random variables $\{\overrightarrow{m}_I, m_I^*\}_{I \in [M(n)]}$ and $\mathcal{O}_{ext}$ occurring is clearly identical between the Real and Ideal experiments due to the choice of uniformly random oracles $\mathcal{O}$ for the Ideal experiment. Thus, it suffices to show the following lemma:

Lemma 2. For all $n \in \mathbb{N}$ and for any assignment of $\{\overrightarrow{m}_I, m_I^*\}_{I \in [M(n)]}$ and $\mathcal{O}_{ext}$ such that

$$\mathsf{Output}[\mathsf{Real}_{\{\overrightarrow{m}_I, m_I^*\}_{I \in [M(n)]}, \mathcal{O}_{ext}}(1^n)] \neq \mathsf{Fail}$$

it holds that:

$$\mathsf{Ideal}_{\{\overrightarrow{m}_I, m_I^*\}_{I\in[M(n)]}, \mathcal{O}_{ext}}(1^n) = \mathsf{Real}_{\{\overrightarrow{m}_I, m_I^*\}_{I\in[M(n)]}, \mathcal{O}_{ext}}(1^n)$$

and thus

$$\mathsf{Output}[\mathsf{Ideal}_{\{\overrightarrow{m}_I, m_I^*\}_{I\in[M(n)]}, \mathcal{O}_{ext}}(1^n)] = \mathsf{Output}[\mathsf{Real}_{\{\overrightarrow{m}_I, m_I^*\}_{I\in[M(n)]}, \mathcal{O}_{ext}}(1^n)]$$

Proof. Intuitively, the lemma follows from the uniqueness property of Π; for any message m^*, there is only a single possible signature σ^*. Thus, unless the extraction fails for some m^*, the simulation in $\mathcal{B}$ will output the same forgery as in the actual execution of $\mathcal{A}$.

Formally, denote experiments $\mathsf{Real}_{\{\overrightarrow{m}_I, m_I^*\}_{I\in[M(n)]}, \mathcal{O}_{ext}}(1^n)$ and $\mathsf{Ideal}_{\{\overrightarrow{m}_I, m_I^*\}_{I\in[M(n)]}, \mathcal{O}_{ext}}(1^n)$ by Real and Ideal respectively. Because $\mathcal{A}$'s, $\mathcal{C}$'s, and $\mathcal{R}$'s randomness are fixed by assumption, it must be the case that the views of Ideal and Real are identical up to the first forgery request. At this point, if $\mathcal{B}$ in Real does not return Fail (indicating that $\mathcal{B}$ either found a valid forgery or will return $\perp$ due to $\mathcal{R}$ returning an incorrect signature), the forgeries (m_I^*, σ_I^*) returned by $\mathcal{A}$ in Ideal and by $\mathcal{B}$ in Real must also be identical. This holds because (1) the situations in which $\mathcal{A}$ or $\mathcal{B}$ return $\perp$ are the same, and (2) if, given certain randomness, $\mathcal{A}$ does not return $\perp$ and $\mathcal{B}$ does not return Fail, both $\mathcal{A}$ and $\mathcal{B}$ must return a correct signature of an identical message m_I^*, which by uniqueness must also be identical.

From here, we can inductively apply the same logic; if we assume that the first k forgery requests' responses are identical (i.e. that $\mathcal{B}$ has not yet failed), then, for each possible assignment of these responses (which must by assumption be equally likely between $\mathcal{A}$ and $\mathcal{B}$), the execution with fixed randomness proceeds identically up to the $(k+1)^{\text{st}}$ forgery request; hence, if $\mathcal{B}$ does not return Fail for that request as well, its responses are once again distributed identically in Real to $\mathcal{A}$'s in Ideal, meaning that the executions up to the $(k+1)^{\text{st}}$ forgery requests' responses must likewise be identical.

So, if we fix $\{\overrightarrow{m}_I, m^*_I\}_{I\in[M(n)]}$ and $\mathcal{O}_{ext}$ for both experiments, and if $\mathcal{B}$ does not return Fail, then all messages sent by $\mathcal{A}$ to $\mathcal{R}$ in Ideal must be identical to the simulated (non-rewound) messages from $\mathcal{B}$ to $\mathcal{R}$ in Ideal, ensuring that (since $\mathcal{R}$ and $\mathcal{C}$ are executed identically and with the same randomness in both experiments) the executions will likewise be identical. □

Thus, by this lemma, the probability that the outputs of the two experiments differ must be at most the probability that some assignment of the random variables occurs which causes $\mathsf{Real}(1^n)$ to return Fail (as in all other cases the views and outputs are identical), as desired. □

Next, we use this to bound $\mathcal{R}^{\mathcal{A}}$'s success probability by bounding the probability that $\mathcal{B}$ will return Fail for some simulated instance I of $\mathcal{A}$. Let $M(n)$ be the maximum, over all randomness of $\mathcal{A}$, $\mathcal{C}$, and $\mathcal{R}$, of the number of instances of $\mathcal{A}$ that $\mathcal{R}$ runs to completion (i.e., for which it responds to all $\ell(n)$ queries) during the experiment $\mathsf{Real}(1^n)$. Then we show the following:

Proposition 1. There exists a negligible function $\epsilon(\cdot)$ such that, for all $n \in \mathbb{N}$:

$$\Pr[\mathsf{Output}[\mathsf{Real}(1^n)] = \mathsf{Fail}] \leq M(n)\left(\frac{M(n)+r(n)+1}{\ell(n)+1}\right) + \epsilon(n)$$

Proof. We first prove the following claim for any execution $\mathsf{Real}^*_{\{\overrightarrow{m}_I, m^*_I\}_{-k}, \mathcal{O}_{ext}}(1^n)$ (i.e., for any fixed setting of all randomness aside from $\overrightarrow{m}_k$ and m^*_k), and notice that, since it applies to arbitrarily fixed randomness, it must thus apply over all possible randomness of the experiment $\mathsf{Real}(1^n)$:

Claim 4. There exists negligible $\nu(\cdot)$ such that, given any experiment $\mathsf{Real}^*_{\{\overrightarrow{m}_I, m^*_I\}_{-k}, \mathcal{O}_{ext}}(1^n)$, the probability, over the uniformly chosen messages $m_{k,1}, \ldots, m_{k,\ell(n)}, m^*_k$, that the simulated

instance k will return Fail is, for all $n \in \mathbb{N}$, at most

$$\frac{M(n) + r(n) + 1}{\ell(n) + 1} + \nu(n)$$

Proof. Let us begin by assuming that other simulated instances (besides k) of $\mathcal{A}$ in the experiment $\mathsf{Real}^*_{\{\overrightarrow{m}_I, m_I^*\}_{-k}, \mathcal{O}_{ext}}(1^n)$ will never return Fail (i.e., that they will "magically" produce a correct forgery in the case where they otherwise would return Fail). Clearly, this can only increase the probability that instance k will return Fail by ensuring that the experiment never aborts early.

Now let us consider the messages $\overrightarrow{m^*}_k \triangleq (m_{k,1}, \ldots, m_{k,\ell(n)}, m_k^*)$ in instance k; note that by the definition of $\mathsf{Real}^*_{\{\overrightarrow{m}_I, m_I^*\}_{-k}, \mathcal{O}_{ext}}(1^n)$ the execution is fully determined by $\overrightarrow{m^*}_k$. Let us also define for $i \in [\ell(n) - 1]$ the "rewound" sequence

$$\rho(\overrightarrow{m^*}_k, i) \triangleq (m_{k,1}, \ldots, m_{k,i-1}, m_k^*)$$

that is, the rewinding of $\overrightarrow{m}_k$ where the message in slot i is replaced by m_k^* to attempt to extract a forgery.

In order for $\mathcal{B}$ to return Fail, one of the following "bad events" must occur for each $i \in [\ell(n)]$:

- $E_1(\rho(\overrightarrow{m^*}_k, i))$: $\mathcal{R}$ fails to return a valid signature of m_I^* in the rewinding of the last slot in the sequence (slot i).
- $E_2(\rho(\overrightarrow{m^*}_k, i))$: During the rewinding of the last slot in the sequence (slot i), $\mathcal{R}$ asks for a forgery for another instance $k' \neq k$ before returning a signature for m_I^*, *or* $\mathcal{R}$ requires external communication with $\mathcal{C}$ (which cannot be rewound) before returning a signature for m_I^*.

In addition, for k to fail, the non-rewound execution of the instance must succeed, in that the event $E_1(\overrightarrow{m}_{k,\leq i})$ (where $\mathcal{R}$ fails to return a valid signature) *cannot* occur for any prefix $\overrightarrow{m}_{k,\leq i} = (m_{k,1}, \ldots, m_{k,i})$, where $i \in [\ell(n)]$.

Since, as we have noted, the behavior of k in the execution of $\mathsf{Real}^*_{\{\overrightarrow{m}_I, m^*_I\}_{-k}, \mathcal{O}_{ext}}(1^n)$ is fully determined by $\overrightarrow{m^*}_k$, every sequence $\overrightarrow{m^*}_k$ will deterministically either result in instance k returning something (either a forgery or $\perp$) or aborting and returning Fail; we shall refer to the former type of sequence (where k succeeds) as a "good" sequence, and the latter type as a "bad" sequence. To describe the relationship between these "good" and "bad" sequences, we first introduce the following terminology:

For any $k > 0$ and any arbitrary set $\overrightarrow{m}$ of k distinct messages in $[2^n]$, let $\Pi_k(\overrightarrow{m})$ denote the set of ordered permutations of the elements of $\overrightarrow{m}$. Given a sequence $\pi = (m_1, \ldots, m_{k-1}, m^*) \in \Pi_k(\overrightarrow{m})$, we let the "rewinding" operator ρ for $i \in [k-1]$ be defined as before—that is:

$$\rho(\pi, i) \triangleq (m_1, \ldots, m_{i-1}, m^*)$$

(Note that this is a sequence of length i.) We shall say that a sequence $a = (a_1, \ldots, a_{k-1}, a^*) \in \Pi_k(\overrightarrow{m})$ *blocks* a sequence $b = (b_1, \ldots, b_{k-1}, b^*) \in \Pi_k(\overrightarrow{m})$ with respect to some $i \in [k-1]$ if

$$\rho(a, i) = (b_1, \ldots, b_i)$$

that is, if a has a rewinding equivalent to a prefix of b. If we wish to denote that a blocks b with respect to a particular i, we shall say that a blocks b *in slot* i. Furthermore, we will say that sequences $a^1, a^2, \ldots a^c \in \Pi_k(\overrightarrow{m})$ *c-block* a sequence $b \in \Pi_k(\overrightarrow{m})$ if there exist *distinct* $i_1, \ldots, i_c \in [k-1]$ such that, for any $j \in [c]$, a^j blocks b in slot i_j.

We next formalize the relationship between the "blocking" property and good/bad sequences that will allow us to use this property to bound the number of bad sequences that may occur. Specifically, we prove the following lemma:

Lemma 3. Any sequence that is $(M(n) + r(n) + 1)$-blocked by bad sequences must be a good sequence.

Proof. Consider a sequence $\overrightarrow{m'}_k$ which is $(M(n) + r(n) + 1)$-blocked by bad sequences. This means that $\overrightarrow{m'}_k$ must have $M(n) + r(n) + 1$ distinct slots i_j for which E_1 or E_2 occurs in its (non-rewound) execution, as the execution is at each of those points identical to a rewinding of one of the bad sequences by the fact that the sequence blocks $\overrightarrow{m'}_k$ in slot i_j. However, because at most $M(n)$ end messages (to completed instances of $\mathcal{A}$, note that others are answered with $\perp$) and at most $r(n)$ rounds of external communication can occur in any given execution, we observe that E_2 can happen for at most $M(n) + r(n)$ of these slots, and thus that E_1 must happen for at least one slot. In this case, we can deduce that $\overrightarrow{m'}_k$ must be a *good* sequence, because it must contain some slot for which $\mathcal{R}$ fails to return a correct response (meaning that $\mathcal{B}$ can successfully emulate $\mathcal{A}$ by returning $\perp$)—that is, the event $E_1(\overrightarrow{m}_{k,\leq i})$ must occur for that slot, which we have previously stated cannot be the case for bad $\overrightarrow{m^*}_k$. □

Consider, then, a set S of "bad" sequences $\overrightarrow{m^*}_k$ which are permutations of any set of $\ell(n) + 1$ distinct messages (i.e., an unordered set containing $\overrightarrow{m}_{k,i}$ and m^*_k). The following lemma, combined with Lemma 3, allows us to bound the size of such a set S:

Lemma 4. Let $\overrightarrow{m}$ be an arbitrary set of $\ell+1$ distinct messages in $[2^n]$, and let $S \subset \Pi_{\ell+1}(\overrightarrow{m})$ be a set of permutations of $\overrightarrow{m}$. If it is the case that, for some $B \in \mathbb{N}$, any member of $\Pi_{\ell+1}(\overrightarrow{m})$ which is $(B+1)$-blocked by a subset of S cannot itself lie in S, then $|S| \leq (B+1)\ell!$

Proof. We begin with the following crucial claim:

Subclaim 1. No member of $\Pi_{\ell+1}(\overrightarrow{m})$ is $(B+2)$-blocked by a subset of S.

Proof. Assume for the sake of contradiction that there exists some $\pi \in \Pi_{\ell+1}(\overrightarrow{m})$, $B+2$ sequences $\pi^1, \ldots, \pi^{B+2} \in S$, and $B+2$ distinct integers $i_1, \ldots, i_{B+2} \in [\ell]$ such that each partial sequence $\rho(\pi^j, i_j)$ is equivalent to the first i_j elements of π.

Assume without loss of generality that the integers i_j are in strictly ascending order. Consider the last sequence $\pi^{B+2} = (\pi_1^{B+2}, \ldots, \pi_*^{B+2})$; we shall show that π^{B+2} is $(M+1)$-blocked by sequences in S, leading to a contradiction because by definition no element of S can be $(M+1)$-blocked by other members of S.

We know that, since by assumption the first i_{B+2} elements of π are equivalent to

$$\rho(\pi^{B+2}, i_{B+2}) = (\pi_1^{B+2}, \ldots, \pi_{i_{B+2}-1}^{B+2}, \pi_*^{B+2})$$

then the first $i_{B+2} - 1$ elements of π^{B+2} and π must be identical. However, notice that, for any $j < B+2$, we have that $\rho(\pi^j, i_j)$ is identical to π in the first i_j elements, which, since by assumption $i_j \leq i_{B+2} - 1$, also indicates that $\rho(\pi^j, i_j)$ is identical to π^{B+2} in the first i_j elements.

This in turn implies that π^{B+2} is $(B+1)$-blocked by the sequences $\pi^1, \ldots, \pi^{B+1} \in S$, contradicting that $\pi^{B+2} \in S$ by the requisite property of S. □

So, we know that any member of S can be at most B-blocked by a subset of S, while any non-member can be at most $(B+1)$-blocked by a subset of S. We will combine this fact (an effective upper bound on the number of blocked sequences) with the subsequent claim (a respective lower bound) to derive our final bound on $|S|$.

Subclaim 2. For each $i \in [\ell]$, there exist $|S|$ distinct sequences blocked in slot i by sequences in S.

Proof. Beginning with $i = 1$, we observe that sequences with at least $|S|/\ell!$ different last

elements m^* must occur in S (as there are only $\ell!$ sequences with any given last element). Furthermore, any sequence in S with a certain last element m^* must block in slot 1 a total of $\ell!$ different sequences (i.e., anything beginning with m^*), and different m^* will produce disjoint sets of sequences blocked. Thus, we conclude that the sequences in S will block in slot 1 at least $(|S|/\ell!)\ell! = |S|$ distinct sequences.

For the remaining slots $i > 1$, we can apply the same logic to the distinct arrangements of the elements $(m_1, ..., m_{i-1})$ and m^* . Among the sequences in S there must be a minimum of $|S|/(\ell+1-i)!$ such arrangements (since, given a fixed $(m_1, ..., m_{i-1}, m^*)$, there are $(\ell+1-i)!$ sequences possible), and sequences with each arrangement will block in slot i a total of $(\ell + 1 - i)!$ distinct sequences (i.e., any sequence beginning with $(m_1, ..., m_{i-1}, m^*)$). Hence, the sequences in S will block in slot i at least $(|S|/(\ell + 1 - i)!)(\ell + 1 - i)! = |S|$ distinct sequences. □

In total, we notice that at least $|S|$ distinct sequences are blocked in slot i for any $i \in [\ell]$, and so there are at least $|S|\ell$ distinct pairs (π, i) such that the sequence π is blocked in slot i by sequences in S. Furthermore, we recall that the sequences in S are each blocked in slot i by sequences in S for at most B different i, while the remaining $(\ell + 1)! - |S|$ elements are each blocked in slot i by sequences in S for at most $B + 1$ different i. This provides an upper bound of $B|S| + ((\ell + 1)! - |S|)(B + 1)$ on the number of "blocking" pairs (π, i). We lastly combine these lower and upper bounds (noting that, if the lower bound exceeded the upper bound, there would be a contradiction) to bound $|S|$:

$$|S|\ell \leq B|S| + ((\ell + 1)! - |S|)(B + 1) = B(\ell + 1)! + (\ell + 1)! - |S|$$

$$|S|(\ell + 1) \leq (B + 1)(\ell + 1)!$$

$$|S| \leq (B + 1)\ell!$$

□

Recall that, if S is the set of all bad sequences which are permutations of some set $\overrightarrow{m^*}$ of $\ell(n)+1$ distinct messages, we have by Lemma 3 that any sequence which is $(M(n)+r(n)+1)$-blocked by bad sequences in S must be good and thus lie outside of S. Hence, by Lemma 4, S has size at most

$$(M(n) + r(n) + 1)\,(\ell(n))!$$

Given any set of $\ell(n) + 1$ distinct messages, then, the above applies to show that at most an

$$\frac{(M(n) + r(n) + 1)\,(\ell(n))!}{(\ell(n) + 1)!} = \frac{M(n) + r(n) + 1}{\ell(n) + 1}$$

fraction of the sequences defined by the permutations of this set can be bad, and the remainder must be good. Applying this to every possible set of $\ell(n) + 1$ distinct messages, we get that at most the same fraction of *all* sequences of distinct messages can be bad. While the property that messages are distinct is not necessarily guaranteed, we note that the probability that they are not over uniformly randomly chosen messages is negligible—specifically, we notice that the probability of any pair of elements colliding is 2^{-n}, and so, by the union bound, the probability that any of the $\frac{\ell(n)(\ell(n)+1)}{2}$ pairs of elements can collide is smaller than $\nu(n) \triangleq \ell(n)^2 2^{-n}$ (which is negligible in n because $\ell(\cdot)$ is polynomial).

Hence, the chance that a sequence chosen at random is bad, which by definition is equal to the probability that a randomly chosen sequence of messages $\overrightarrow{m^*}_k = (m_{k,1}, \ldots, m_{k,\ell(n)}, m^*_k)$ will result in instance k returning Fail, can be at most the fraction of sequences without repeated elements which are bad plus the fraction of sequences with repeated elements, or:

$$\frac{M(n) + r(n) + 1}{\ell(n) + 1} + \nu(n)$$

as desired. □

Recall that, because this result holds for any execution of $\mathsf{Real}^*_{\{\vec{m}_I, m^*_I\}_{-k}, \mathcal{O}_{ext}}(1^n)$, it must also hold over a *random* such execution—i.e., the actual execution $\mathsf{Real}(1^n)$ of $\mathcal{B}$, where the messages for all instances are chosen uniformly at random. Furthermore, it holds for any instance k of $\mathcal{A}$.

To conclude the proof of the proposition, by Claim 1 we know that $\mathcal{R}$ must send a total of at least $\ell(n)$ messages to each instance of $\mathcal{A}$ in order for the failure probability of $\mathcal{B}$ to emulate that instance to be more than negligible; if not, then $\mathcal{B}$ will always respond with $\perp$ (having not received $\ell(n)$ signature query responses), while the claim shows that $\mathcal{A}$ will return $\perp$ with all but negligible probability. Recall that $M(n)$ is an upper bound to the number of instances of $\mathcal{A}$ to which $\mathcal{R}$ sends $\ell(n)$ messages. By Claim 4 and the union bound over all $M(n)$ completed instances of $\mathcal{A}$, the failure probability of $\mathcal{B}$ for those instances is at most

$$M(n)\left(\frac{M(n)+r(n)+1}{\ell(n)+1}+\nu(n)\right)$$

for negligible $\nu(\cdot)$, and the failure probability for all other instances (of which there can only be a polynomial number by the time constraint on $\mathcal{R}$) is negligible by the union bound applied to Claim 1. Hence the overall failure probability of Real (i.e., the execution of $\mathcal{B}$) must be bounded above by

$$\begin{aligned}\Pr[\mathsf{Output}[\mathsf{Real}(1^n)]=\mathsf{Fail}] &\leq M(n)\left(\frac{M(n)+r(n)+1}{\ell(n)+1}+\nu(n)\right)+\nu'(n)\\ &< M(n)\left(\frac{M(n)+r(n)+1}{\ell(n)+1}\right)+\epsilon(n)\end{aligned}$$

for some negligible functions $\nu'(\cdot)$ and $\epsilon(\cdot)$. □

Completing the Proof of Lemma 1. Finally, in order to bound the security loss, we note that, if the probability $\mathsf{Success}_{\mathcal{R}^{\mathcal{A}}}(n)$ (which specifically is by definition a lower bound to the probability $\Pr[\mathsf{Output}[\mathsf{Ideal}(1^n)]=\mathsf{Accept}]-t(n)$; recall that $t(\cdot)$ is the threshold for

the underlying assumption $\mathcal{C}$) is non-negligibly greater than the failure probability of Real, there exists a polynomial $p(\cdot)$ such that:

$$\Pr[\mathsf{Output}[\mathsf{Ideal}(1^n)] = \mathsf{Accept}] - t(n) \geq \Pr[\mathsf{Output}[\mathsf{Real}(1^n)] = \mathsf{Fail}] + \frac{1}{p(n)}$$

But, by Claim 3, this would imply that

$$\Pr[\mathsf{Output}[\mathsf{Real}(1^n)] = \mathsf{Accept}] - t(n) \geq \frac{1}{p(n)}$$

that is, that $\mathcal{B}$ breaks the security of $(\mathcal{C}, t(\cdot))$. So, by Proposition 1, unless $\mathcal{B}$ breaks the security of $(\mathcal{C}, t(\cdot))$, the above cannot be the case—that is, there must exist negligible $\epsilon(\cdot), \epsilon'(\cdot)$ such that, for sufficiently large n:

$$\begin{aligned}\mathsf{Success}_{\mathcal{R}^{\mathcal{A}}}(n) &\leq \Pr[\mathsf{Output}[\mathsf{Real}(1^n)] = \mathsf{Fail}] + \epsilon'(n) \\ < M(n)\left(\frac{M(n)+r(n)+1}{\ell(n)+1}\right) + (\epsilon(n)+\epsilon'(n)) &< M(n)\left(\frac{M(n)+r(n)+1}{\ell(n)}\right)\end{aligned}$$

Of course, $\mathsf{Success}_{\mathcal{R}^{\mathcal{A}}}(n)$, being a probability, is also trivially bounded above by 1. Furthermore, by the definition of $M(n)$, we know that $\mathsf{Query}_{\mathcal{R}^{\mathcal{A}}}(n) \geq M(n)\ell(n)$. Lastly, we consider two cases to derive our bound on the security loss.

Case 1. If $M(n) \geq \sqrt{\ell(n)} - (r(n)+1)$, then:

$$\begin{aligned}\lambda_{\mathcal{R}}(n) &\geq \frac{\mathsf{Success}_{\mathcal{A}}(n)}{\mathsf{Success}_{\mathcal{R}^{\mathcal{A}}}(n)} \frac{\mathsf{Query}_{\mathcal{R}^{\mathcal{A}}}(n)}{\mathsf{Query}_{\mathcal{A}}(n)} \geq \frac{1}{1} \frac{M(n)\ell(n)}{\ell(n)} \\ &= M(n) \geq \sqrt{\ell(n)} - (r(n)+1)\end{aligned}$$

Case 2. Otherwise, if $M(n) < \sqrt{\ell(n)} - (r(n)+1)$ we have $M(n)+r(n)+1 < \sqrt{\ell(n)}$, and so:

$$\lambda_{\mathcal{R}}(n) \geq \frac{\mathsf{Success}_{\mathcal{A}}(n)}{\mathsf{Success}_{\mathcal{R}^{\mathcal{A}}}(n)} \frac{\mathsf{Query}_{\mathcal{R}^{\mathcal{A}}}(n)}{\mathsf{Query}_{\mathcal{A}}(n)} \geq \frac{1}{M(n)\left(\frac{M(n)+r(n)+1}{\ell(n)}\right)} \frac{M(n)\ell(n)}{\ell(n)}$$

$$= \frac{\ell(n)}{M(n) + r(n) + 1} > \frac{\ell(n)}{\sqrt{\ell(n)}} = \sqrt{\ell(n)}$$

Either way, we observe that $\lambda_{\mathcal{R}}(n) \geq \sqrt{\ell(n)} - (r(n) + 1)$, thus completing the proof of both Lemma 1 and Theorem 1.

2.4 Impossibility of Linear-Preserving Reductions from Adaptively Secure MACs and PRFs

The second and final major impossibility result we present here rules out the possibility of a linear-preserving reduction from the security of any deterministic MAC to a bounded-round assumption. Formally:

Theorem 2. Let Π be a deterministic MAC. If there exists a fixed-parameter black-box reduction $\mathcal{R}$ for basing adaptive multi-key unforgeability of Π on some $r(\cdot)$-round intractability assumption $(\mathcal{C}, t(\cdot))$ (for polynomial $r(\cdot)$), then either:

(1) $\mathcal{R}$ is not a linear-preserving reduction, or

(2) there exists a polynomial-time adversary $\mathcal{B}$ breaking the assumption $(\mathcal{C}, t(\cdot))$.

Perhaps unsurprisingly given the similar structure, the crux of this theorem will be the same as that of its predecessor—that is, the counting lemma for rewindings that can be seen in Lemma 4. However, there are key differences and improvements to both the meta-reduction and the analysis that make this result very distinct and worth presenting alongside the previous one.

Theorem 2 can be generalized to apply as written to several other primitives besides simply deterministic MACs. Specifically, we can also apply it to rule out linear-preserving

reductions from either adaptive multi-key unforgeability of deterministic *digital signature schemes* or adaptive multi-key *pseudorandomness* of a family of functions (i.e., adaptive multi-key PRFs) to bounded-round assumptions. As these are fairly direct corollaries, we again refer the interested reader to the full version of the work [101].

To prove Theorem 2, we first present the following crucial lemma:

Lemma 5. Let Π be a deterministic MAC, and let $(\mathcal{C}, t(\cdot))$ be some $r(\cdot)$-round intractability assumption for polynomial $r(\cdot)$. If for some polynomial $\ell(\cdot)$ there exists a fixed-parameter black-box reduction $\mathcal{R}$ for basing adaptive $\ell(n)$-key unforgeability of Π on the hardness of $(\mathcal{C}, t(\cdot))$, then either $\mathcal{R}$'s security loss is at least

$$\lambda_{\mathcal{R}}(n) \geq \left(1 - \frac{1}{2\ell(n)^2}\right)\left(\sqrt{\ell(n)} - (r(n) + 2)\right)$$

for all sufficiently large $n \in \mathbb{N}$, or there exists a polynomial-time adversary $\mathcal{B}$ that breaks the assumption $(\mathcal{C}, t(\cdot))$.

Because $p(\cdot)$ in the definition of a linear-preserving reduction is an *a priori* fixed polynomial, and in particular cannot depend on $\ell(n)$, this lemma will prove Theorem 2, as follows:

Proof. Let $\mathcal{R}$ be a reduction from adaptive multi-key unforgeability of Π to the hardness of $(\mathcal{C}, t(\cdot))$. Assume for the sake of contradiction that Lemma 5 is true, yet $\mathcal{R}$ is linear-preserving and $(\mathcal{C}, t(\cdot))$ is secure. Because $\mathcal{R}$ is linear-preserving, there is some polynomial $p(\cdot)$ such that $\lambda_{\mathcal{R}}(n) \leq p(n)$ for sufficiently large n. Furthermore, $\mathcal{R}$ is by definition a reduction from adaptive $\ell(n)$-key unforgeability for *every* polynomial $\ell(n)$, including, say, $\ell(n) = (2p(n) + r(n) + 3)^2$, so by Lemma 5 we have:

$$\lambda_{\mathcal{R}}(n) \geq \left(1 - \frac{1}{2\ell(n)^2}\right)\left(\sqrt{\ell(n)} - (r(n) + 2)\right) \geq \frac{1}{2}(2p(n) + 1) > p(n)$$

which is a clear contradiction.

□

2.4.1 Technical Overview

Next, we shall explain the methodology for the proof of Lemma 5 at a high level.

The ideal adversary. We begin by constructing and investigating an "ideal" adversary $\mathcal{A}$. To summarize, $\mathcal{A}$ will first make $q(n)$ random tag queries (where $q(n)$ is a polynomial to be determined later) to each of the $\ell(n)$ instances of the MAC Π, continue by opening all but one of the keys in a random order (while also verifying that the challenger or $\mathcal{R}$ answered its queries consistently with the opened keys), and lastly, if it received correct responses for the opened instances, use the information gained from the queries for the remaining instance to attempt to brute-force a forgery for that instance. (On the other hand, if verification fails, $\mathcal{A}$ will "reject", returning $\perp$ instead of a forgery.)

In virtually all meta-reductions to date, the ideal adversary is able to perfectly brute-force the challenger's secret information and break the primitive with probability 1. Here, however, that is not the case; $\mathcal{A}$ is limited to a polynomial number of tag queries (which is necessary for simulatability) and furthermore has no way to publicly verify whether a certain key or forgery is correct. The most $\mathcal{A}$ can do, in fact, is brute-force the set of all keys consistent with the tag queries it makes for the unopened instance, pick one of those keys, and use it to generate a forgery in the hopes that it will match with the key the challenger has selected.

This is where the "key uniqueness" property discussed in the introduction will first factor in. We show that, since the key picked by the adversary agrees with the key picked by the

challenger on all $q(n)$ tag queries, then it must with overwhelming probability also agree on a large fraction $(1 - 2n/q(n))$ of possible messages. Hence, $\mathcal{A}$ will have a $1 - 2n/q(n)$ chance of producing a correct forgery when it evaluates the Tag function using the key it extracts on a random message m^* (i.e., the message it eventually will randomly select for its forgery)—that is, $\mathsf{Success}_{\mathcal{A}}(n) \geq 1 - 2n/q(n)$.

Before proceeding to discuss the meta-reduction, we need to address one final technical issue with the ideal adversary. Namely, since $\mathcal{A}$ works by returning the "next-message" function given a transcript of the interaction thus far, we need to ensure that $\mathcal{R}$ must actually complete the full interaction with $\mathcal{A}$ in order to cause $\mathcal{A}$ to accept and return a forgery, rather than potentially guessing a "fake" accepting transcript for a later point in the interaction to "skip" or avoid responding to certain queries from $\mathcal{A}$. In particular, a reduction $\mathcal{R}$ that skips key-opening queries would be extremely problematic in our analysis of the meta-reduction later on, since the meta-reduction will rely on $\mathcal{R}$'s responses to these queries to properly emulate the ideal adversary $\mathcal{A}$.

Unfortunately, it turns out that $\mathcal{A}$'s key-opening queries, since they convey no information besides the instance to open, have low entropy and thus are easy to predict (and skip) by $\mathcal{R}$. To fix this, we introduce additional "dummy" queries—specifically, random tag queries to instances whose keys have not yet been opened—made after each of the key-opening queries. These serve the purpose of increasing the entropy present in the key-opening phase of the transcript—which guarantees that $\mathcal{R}$ must answer all $\ell(n) - 1$ of $\mathcal{A}$'s key-opening queries to successfully complete the interaction (unless it can correctly guess the random input for the dummy query)—but are otherwise ignored.

The meta-reduction. In our discussion of $\mathcal{A}$, we were able to bound $\mathsf{Success}_{\mathcal{A}}(n)$; thus, we turn next to investigating $\mathsf{Success}_{\mathcal{R}^{\mathcal{A}}}(n)$. To do this, we construct a *meta-reduction* $\mathcal{B}$ which runs $\mathcal{R}$ while attempting to efficiently emulate the interaction between $\mathcal{R}$ and $\mathcal{A}$. $\mathcal{B}$ will simulate instances of $\mathcal{A}$ by, exactly as before, making $q(n)$ random tag queries to each instance, opening the key for all but one instance (in a random order and with the interleaved tag queries as above), and checking $\mathcal{R}$'s responses for consistency.

The key difference, of course, is that $\mathcal{B}$ cannot brute-force a forgery; instead, for the unopened instance, $\mathcal{B}$ will attempt to extract a correct key from $\mathcal{R}$ by rewinding the interaction to the key-opening queries and substituting the unopened instance for each other instance in turn. If $\mathcal{R}$ responds to any of the valid queries with a key that matches with the tag queries for that instance, then $\mathcal{B}$ will apply that key to a random message m^* to generate a forgery. If $\mathcal{B}$ does not receive a valid key in this fashion, then it will abort, returning Fail.

Notably, $\mathcal{B}$ will also have to ignore rewindings where, before returning its response to the key-opening query, either $\mathcal{R}$ attempts to communicate externally with $\mathcal{C}$ (which could change the state of the challenger if forwarded), $\mathcal{R}$ requests a forgery from another instance of $\mathcal{A}$ (as this would require additional "nested" rewinding which could grow exponentially), or $\mathcal{R}$ would rewind $\mathcal{A}$ (which precludes $\mathcal{R}$ returning a key); this will factor into the analysis of the failure probability later.

The main task in proving our lemma, then, reduces to that of bounding $\Pr[\langle \mathcal{R}^{\mathcal{A}}, \mathcal{C} \rangle \rightarrow \mathsf{Accept}] - \Pr[\langle \mathcal{B}, \mathcal{C} \rangle \rightarrow \mathsf{Accept}]$. Intuitively, if we come up with such a bound (call it $p(n)$), then, if $\mathsf{Success}_{\mathcal{R}^{\mathcal{A}}}$ is non-negligibly higher than $p(n)$—that is, $\langle \mathcal{R}^{\mathcal{A}}, \mathcal{C} \rangle$ accepts with such a probability—then $\langle \mathcal{B}, \mathcal{C} \rangle$ will accept with non-negligible probability, hence breaking $\mathcal{C}$. Bounding this probability $p(n)$ is in fact quite non-trivial, as one cannot, say, naïvely apply earlier techniques for meta-reduction analysis to the meta-reduction $\mathcal{B}$. Intuitively, this is

because we no longer have a strong "uniqueness" property characteristic of meta-reductions to date—that is, there is no longer a *unique* possible valid forgery $\mathcal{B}$ can extract from its rewinding. Not only does this make it difficult to guarantee that $\mathcal{A}$ and $\mathcal{B}$ produce close distributions of forgeries, but, in conjunction with $\mathcal{B}$'s rewinding strategy, this makes analyzing the failure probability problematic for more complex reasons. For example, consider a reduction $\mathcal{R}$ which might try to rewind $\mathcal{A}$ and change its responses to queries in order to attempt to change the forgery generated; it is straightforward to see that proof techniques such as that of the prequel [98] immediately fail (due to a potentially unbounded number of nested forgery requests) if $\mathcal{R}$ can theoretically expect to receive many different forgeries by repeatedly rewinding the same instance.

A "hybrid" meta-reduction. We present a way to effectively separate dealing with the issues of uniqueness and rewinding, namely by defining a "hybrid" meta-reduction $\mathcal{B}'$ which, while inefficient, is easy to compare to either $\mathcal{A}$ or $\mathcal{B}$.

At a high level, we construct $\mathcal{B}'$ so that it behaves identically to $\mathcal{B}$ as long as there is only a single possible forgery to return, and so that it behaves identically to $\mathcal{A}$ whenever rewinding succeeds. More specifically, it acts identically to $\mathcal{B}$ until after rewinding finishes, then, if it obtains a forgery, brute-forces one in the same manner as $\mathcal{A}$.

Clearly, $\mathcal{B}'$ can only diverge from $\mathcal{B}$ if the forgery $\mathcal{B}$ extracts is different from the one $\mathcal{B}'$ brute-forces. A straightforward application of our earlier "key uniqueness" lemma shows that this happens with at most $2n/q(n)$ probability per forgery returned by $\mathcal{B}'$.

On the other hand, $\mathcal{B}'$ will always return the same forgery as $\mathcal{A}$ *if* it returns a forgery, but we still need to determine the probability with which $\mathcal{B}'$ fails to return a forgery due to unsuccessful rewinding. Luckily, since $\mathcal{B}'$ now *does* have the uniqueness property, we

can proceed along the same lines as in [98] and bound the rewinding failure probability by effectively bounding the probability that a randomly chosen ordering of key-opening queries can result in rewinding failure (while assuming that the rest of the randomness in the interaction is fixed arbitrarily, as, if the bound applies to arbitrarily fixed randomness, it must likewise apply when taken over all possible assignments of the same randomness).

The intuition behind the argument is that, if we assume a bound of $W(n)$ on the number of times $\mathcal{R}$ will rewind past when $\mathcal{B}'$ generates the ordering π of the key-opening queries (and note that, due to uniqueness and careful construction, $W(n)$ will also be a bound on the number of distinct forgery requests $\mathcal{R}$ can make, as we show that any others will be internally simulatable and thus "pointless"), every sequence π that causes $\mathcal{B}'$ to fail must do so because all of its rewindings fail, and the rewindings specifically correspond to other sequences π that can occur. Furthermore, if a rewinding fails due to $\mathcal{R}$ responding to a query incorrectly (as opposed to, e.g., external communication or a nested forgery request), then this rewinding corresponds to a "good" sequence where $\mathcal{A}$ and $\mathcal{B}'$ return $\perp$ (and emulation is successful). So, if some sequence π contains more than $W(n) + r(n) + 1$ queries at which rewindings of other sequences fail, then, since we can have at most $W(n)$ (unique) forgery requests and $r(n)$ rounds of external communication, at least one query must fail due to an incorrect response, which shows that π is a "good" sequence. A counting argument then allows us to achieve a bound of $(W(n) + r(n) + 1)/\ell(n)$ on the failure probability of $\mathcal{B}'$ each time it performs rewinding, or $W(n)(W(n) + r(n) + 1)/\ell(n)$ overall failure probability.

Bounding security loss. Combining all of the facts so far, we know that the above quantity is equivalent to the probability with which $\mathcal{A}$ and $\mathcal{B}'$ diverge, while the probability with which $\mathcal{B}'$ and $\mathcal{B}$ diverge is $2nW(n)/q(n)$ (i.e., the probability that uniqueness fails for at least one of the $W(n)$ forgeries). Thus, $\mathsf{Success}_{\mathcal{R}^{\mathcal{A}}}(n)$, as we have argued, is bounded above by

the sum of these, which (taking $q(n)$ sufficiently large) is at most $W(n)(W(n)+r(n)+2)/\ell(n)$. Furthermore, $\mathsf{Time}_{\mathcal{R}^{\mathcal{A}}}(n)/\mathsf{Time}_{\mathcal{A}}(n) \geq W(n)$ by our assumption that in the worst case $\mathcal{R}$ runs $W(n)$ instances of $\mathcal{A}$. Lastly, $\mathsf{Success}_{\mathcal{A}}(n) \geq 1 - 2n/q(n)$ as we noted earlier.

Hence, either $(\mathcal{C}, t(\cdot))$ is insecure (and our bound for $\mathsf{Success}_{\mathcal{R}^{\mathcal{A}}}(n)$ does not apply), or, by the above facts and case analysis (to deal with the possibility that $W(n)$ might be arbitrarily large), we obtain the result:

$$\lambda_{\mathcal{R}}(n) \geq \left(1 - \frac{1}{2\ell(n)^2}\right)\left(\sqrt{\ell(n)} - (r(n)+2)\right)$$

2.4.2 Proof

We continue by formally proving Lemma 5. Assume a deterministic MAC Π, a reduction $\mathcal{R}$, and an assumption $(\mathcal{C}, t(\cdot))$ as defined in the statement of Lemma 5. Consider an ideal but inefficient adversary $\mathcal{A}$, which technically is given by a random selection from a *family* of inefficient adversaries $\mathcal{A} \leftarrow \{\mathcal{A}^{\mathcal{O}}\}$ (where $\mathcal{O}$ is a uniformly chosen random function) defined as in Figures 2.3 and 2.4; also consider an efficient meta-reduction $\mathcal{B}$ defined as in Figures 2.5 and 2.6.

Before analyzing the properties of $\mathcal{A}$ and $\mathcal{B}$, we verify that $\mathcal{B}$ runs efficiently through the following claim:

Claim 5. $\mathcal{B}(1^n)$ runs in time polynomial in n.

Proof. First consider the execution of $\mathcal{B}$ without rewinding. Because $\mathcal{R}$ is an efficient reduction by definition, $\mathcal{R}$ can begin at most a polynomial number of instances of the adversary $\mathcal{A}$, which means at most some polynomial number (henceforth we shall call this bound $M(n)$) of simulated instances of $\mathcal{A}$ are invoked during the execution of $\mathcal{B}$.

Now, for each instance of $\mathcal{A}$, $\mathcal{B}$ sends a total of $q(n)(\ell(n)+1)$ tag queries, $\ell(n)-1$ key opening queries, and a forgery, for a total of $(q(n)+2)\ell(n)$ total messages. It is clear that each message prior to the forgery takes at most a polynomial amount of time to compute (since no rewinding is involved until the forgery and all other computations, including the Valid predicate, are efficient). Furthermore, there are at most $r(n)$ rounds of communication between $\mathcal{B}$ and $\mathcal{C}$ that must be forwarded, but each of these is guaranteed to be efficiently computable as well because of $\mathcal{R}$'s efficiency.

To factor in rewinding, consider that $\mathcal{R}$ can rewind $\mathcal{A}$, but only up to a polynomial number of times; $\mathcal{B}$ also rewinds simulated instances of $\mathcal{A}$ while attempting to extract a valid forgery key from $\mathcal{R}$, but once again also only up to a polynomial number ($\ell(n)$) of times. Hence, rewinding will at most multiply the running time of $\mathcal{B}$ by a polynomial factor, which results in a polynomial total running time since $\mathcal{B}$ without rewinding is polynomial-time.

□

2.4.3 Analyzing the Ideal Adversary

In order to establish a bound to the security loss $\lambda_{\mathcal{R}}(n)$, we shall determine bounds for $\mathsf{Success}_{\mathcal{A}}(n)$ and $\mathsf{Success}_{\mathcal{R}^{\mathcal{A}}}(n)$; time analysis will follow naturally.

We begin by analyzing the probability $\mathsf{Success}_{\mathcal{A}}(n)$. This is fairly straightforward, following from the critical "key uniqueness" lemma which states that two keys agreeing on all of the $q(n)$ tag queries made by $\mathcal{A}$ are overwhelmingly likely to agree on "most" messages m. Hence, the key chosen by $\mathcal{A}$, even if not the same as that chosen by the challenger, is by definition consistent with it on all of the tag queries and thus should agree on a large fraction of the possible forgery inputs m^*. Formally:

- On receiving an initialization message (Init, s), let $m_{1,1}$ denote a uniformly random message in $\mathcal{M}_n$ generated by random coins resulting from applying the oracle $\mathcal{O}$ to the input $(s, 1, 1, 1)$, and send $(\mathsf{Query}, 1, m_{1,1})$.
- On receiving a transcript of the form

$$\tau = (q_{1,1}, q_{1,2}, \ldots, q_{1,\ell(n)}, q_{2,1}, \ldots, q_{i,j})$$

  where either $i < q(n)$ or $i = q(n)$ and $j < \ell(n)$, such that each $q_{u,v}$ is of the form $((\mathsf{Query}, v, m_{u,v}), \sigma_{u,v})$, do the following:
  - Let $j' = (j \bmod \ell(n)) + 1$.
  - Let $i' = i + 1$ if $j' = 1$ and $i' = i$ otherwise.
  - Let $m_{i',j'}$ be a uniformly random message in $\mathcal{M}_n$ generated by random coins resulting from applying the oracle $\mathcal{O}$ to the input $(s, i', j', 1)$.
  - Return $(\mathsf{Query}, j', m_{i',j'})$.
- On receiving a transcript of the form $\tau = \tau_1 || \tau_2$, where

$$\tau_1 = (q_{1,1}, q_{1,2}, \ldots, q_{1,\ell(n)}, q_{2,1}, \ldots, q_{q(n),\ell(n)})$$

  and where each $q_{u,v}$ is of the form $((\mathsf{Query}, v, m_{u,v}), \sigma_{u,v})$, do the following:
  - Let c be the number of Open queries that have so far appeared in τ_2.
  - Let $\pi = (\pi_1, \ldots, \pi_{\ell(n)})$ be a uniformly random permutation of $[\ell(n)]$, generated by random coins resulting from applying $\mathcal{O}$ to the input τ_1.
  - If τ_2 is empty or ends with messages of the form $((\mathsf{Open}, j), k_j)$, then:
    - Generate ω_{c+1} as a uniformly random message in $\mathcal{M}_n$ generated by random coins resulting from applying the oracle $\mathcal{O}$ to the input $\tau_1 || (s, q(n)+1, c+1, \mathsf{Valid}(\mathcal{O}, \tau^*, s))$ and return $(\mathsf{Query}, q, \omega_{c+1})$, where q is the lexicographically first instance for which τ_2 does not contain an Open query.
  - Otherwise, if $c < \ell(n) - 1$ and the last part of τ_2 contains messages of the form $((\mathsf{Query}, q, \omega_{c+1}), \cdot)$, then return $(\mathsf{Open}, \pi_{c+1})$.
  - Lastly, if τ_2 ends with $((\mathsf{Query}, q, \omega_{\ell(n)}), \cdot)$ and $c = \ell(n) - 1$, return a forgery as follows:
    - If $\mathsf{Valid}(\mathcal{O}, \tau, s) = 0$, return $\perp$.
    - Otherwise, use exhaustive search to find the set K^* of all keys k^* such that, given $j^* = \pi_{\ell(n)}$ as determined above, and for each $i \in [q(n)]$, $\mathsf{Ver}_{k^*}(m_{i,j^*}, \sigma_{i,j^*}) = \mathsf{Accept}$. If K^* is empty then return $\perp$.
    - Lastly, using random coins generated by applying $\mathcal{O}$ to a new input $\tau_1 || (s, q(n) + 2, 0, 1)$, generate a uniformly random message m^* (which will be distinct from all m_{i,j^*} with all-but-negligible probability) and take the *lexicographically first* element k^* of K^*. Return the forgery $(m^*, \mathsf{Tag}_{k^*}(m^*), j^*)$.

Figure 2.3: Formal description of the "ideal" adversary $\mathcal{A}^{\mathcal{O}}$ (1).

Let the predicate $\mathsf{Valid}(\mathcal{O}, \tau, s)$ be defined as follows:

- Parse τ as $\tau_1 || \tau_2$, where

$$\tau_1 = (q_{1,1}, q_{1,2}, \ldots, q_{1,\ell(n)}, q_{2,1}, \ldots, q_{q(n),\ell(n)})$$

  such that each $q_{u,v}$ is of the form $((\mathsf{Query}, v, m_{u,v}), \sigma_{u,v})$. If τ cannot be parsed as such, return 0.

- Let $\pi = (\pi_1, \ldots, \pi_{\ell(n)})$ be a permutation of $[\ell(n)]$ generated in the same manner as in $\mathcal{A}$, using random coins generated by applying $\mathcal{O}$ to the input τ_1.

- Parse

$$\tau_2 = (q_1^*, q_2^*, \ldots, q_c^*[, q_{c+1}^*])$$

  such that each q_i^* is of the form $((\mathsf{Query}, q_i, \omega_i), \cdot, (\mathsf{Open}, \pi_i), k_{\pi_i})$ and q_{c+1}^*, if present, is of the form $((\mathsf{Query}, q_i, \omega_{c+1}), \cdot)$. If τ_2 cannot be parsed as such, or if $c > \ell(n) - 1$, return 0.

- Verify that each q_i in τ_2 is equal to the lexicographically first instance $q \in [\ell(n)]$ such that q does not appear in an Open query earlier in τ_2. If not true, return 0.

- Verify that, for all $i \in [q(n)]$ and all $j \in \{\pi_1, \ldots, \pi_c\}$, $\mathsf{Ver}_{k_j}(m_{i,j}, \sigma_{i,j}) = \mathsf{Accept}$. (Do not verify the responses to queries ω_i in τ_2.) If not true, return 0.

- Verify that every $m_{i,j}$ parsed from the transcript is correctly generated by random coins resulting from applying $\mathcal{O}$ to the input $(s, i, j, 1)$ (for $i \in [q(n)]$) and that each ω_j is correctly generated by random coins resulting from applying $\mathcal{O}$ to the input $\tau_1 || (s, q(n) + 1, j, 1)$. If not true, return 0.

- Otherwise, return 1.

Figure 2.4: Formal description of the "ideal" adversary $\mathcal{A}^{\mathcal{O}}$ (2).

Claim 6. There exists a negligible function $\nu(\cdot)$ such that:

$$\mathsf{Success}_{\mathcal{A}}(n) \geq 1 - \frac{2n}{q(n)} - \nu(n)$$

Proof. The claim follows readily from the following lemma (and the fact that there is only a negligible chance that $\mathcal{A}$ generates an invalid m^*):

Lemma 6. There exists negligible $\nu(\cdot)$ such that, for any family of functions $\mathcal{U} = \{f_k : \mathcal{X}_n \to \mathcal{Y}_n\}_{k \in \{0,1\}^n, n \in \mathbb{N}}$, except with probability $\nu(n)$ over $q(n)$ uniformly random queries $(x_{1,j^*}, \ldots, x_{q(n),j^*}) \leftarrow (\mathcal{X}_n)^{q(n)}$, for any $k_1, k_2 \in (\{0,1\}^n)^2$ such that $f_{k_1}(x_{i,j^*}) = f_{k_2}(x_{i,j^*})$ for

all $i \in [q(n)]$, it is true that:

$$\Pr\left[x^* \leftarrow \mathcal{X}_n : f_{k_1}(x^*) = f_{k_2}(x^*)\right] \geq 1 - \frac{2n}{q(n)} \tag{2.1}$$

Proof. For any key pair (k_1, k_2), let

$$S_{k_1,k_2} \triangleq \{x^* \in \mathcal{X}_n : f_{k_1}(x^*) = f_{k_2}(x^*)\}$$

be the set of inputs where the two keys' outputs are identical.

So, if (2.1) is false for some pair (k_1, k_2), i.e., $|S_{k_1,k_2}| \leq |\mathcal{X}_n|\left(1 - \frac{2n}{q(n)}\right)$; then the probability over $\{x_{i,j^*}\}$ that both keys agree in all $q(n)$ queries to f made by $\mathcal{A}$, or equivalently the probability that $q(n)$ uniformly random queries $\{x_{i,j^*}\}$ lie in S_{k_1,k_2}, is bounded above by:

$$\left(1 - \frac{2n}{q(n)}\right)^{q(n)} = \left(\left(1 - \frac{2n}{q(n)}\right)^{q(n)/2n}\right)^{2n} < \left(\frac{1}{e}\right)^{2n} = \exp(-2n)$$

There exist no more than $(2^n)^2 = 2^{2n}$ possible key pairs $(k_1, k_2) \in (\{0,1\}^n)^2$, each of which by the above must either have the property (2.1) or be such that

$$\begin{aligned}
&\Pr\left[(x_{1,j^*}, \ldots, x_{q(n),j^*}) \leftarrow (\mathcal{X}_n)^{q(n)} : f_{k_1}(x_{i,j^*}) = f_{k_2}(x_{i,j^*}) \forall i \in [q(n)]\right] \\
&= \Pr\left[(x_{1,j^*}, \ldots, x_{q(n),j^*}) \leftarrow (\mathcal{X}_n)^{q(n)} : x_{i,j^*} \in S_{k_1,k_2} \forall i \in [q(n)]\right] \\
&\leq \exp(-2n)
\end{aligned}$$

Then the probability over $\{x_{i,j^*}\}$ that *some* key pair exists which does not have property (2.1) yet does have $f_{k_1}(x_{i,j^*}) = f_{k_2}(x_{i,j^*})$ for all x_{i,j^*} is, by a union bound, at most:

$$\begin{aligned}
\Pr\Big[&(x_{1,j^*}, \ldots, x_{q(n),j^*}) \leftarrow (\mathcal{X}_n)^{q(n)} : \exists (k_1, k_2) \in (\{0,1\}^n)^2 : \\
&x_{i,j^*} \in S_{k_1,k_2} \forall i \in [q(n)] \text{ and } |S_{k_1,k_2}| \leq |\mathcal{X}_n|\left(1 - \frac{2n}{q(n)}\right)\Big]
\end{aligned}$$

$$\leq \sum_{(k_1,k_2)\in(\{0,1\}^n)^2} \mathbf{1}_{|S_{k_1,k_2}|\leq|\mathcal{X}_n|(1-2n/q(n))} \Pr\Big[(x_{1,j^*},\ldots,x_{q(n),j^*}) \leftarrow (\mathcal{X}_n)^{q(n)} : x_{i,j^*} \in S_{k_1,k_2} \forall i \in [q(n)]\Big]$$

$$< 2^{2n}e^{-2n} = (2/e)^{2n}$$

which is clearly negligible in n. □

To prove the claim, we consider the above lemma, letting f_k be the deterministic function Tag_k. When interacting with an honest challenger, the responses to tag queries for each instance will always be consistent with the respective keys, and so $\mathcal{A}$ will never return $\perp$ due to the Valid predicate failing or K^* being empty. Furthermore, for the instance $\pi_{\ell(n)}$ for which $\mathcal{A}$ outputs a forgery, it is overwhelmingly likely (with probability $1-\nu(n)$), by Lemma 6, that all keys in the set K^* recovered by $\mathcal{A}$ will agree with the *correct* (challenger's) key k' for that instance on a large $(1-2n/q(n))$ fraction of random messages m^*. Specifically, this means that, given any choice of key k^* from K^*, $\mathcal{A}$ will produce a correct forgery (m^*, σ^*) (i.e., such that $\sigma^* = \mathsf{Tag}_{k'}(m^*)$, or equivalently $\mathsf{Ver}_{k'}(m^*, \sigma^*) = \mathsf{Accept}$) given random m^* with probability at least $1 - 2n/q(n)$.

Thus, $\mathcal{A}$ succeeds in the interaction in the event that Lemma 6 does not fail (i.e., property (2.1) holds for every key pair) and that $\mathcal{A}$ chooses a "good" m^* (i.e., one which does not repeat a previous query and produces the same tag under k^* as under the challenger's key k') given its choice of $k^* \leftarrow K^*$; the claim follows from the union bound over these events. □

We require one additional claim concerning the adversary, which states that the reduction $\mathcal{R}$ must have actually responded to all $\ell(n)-1$ key-opening queries to have a non-negligible chance of receiving a forgery. This will be important later, to ensure that $\mathcal{R}$ cannot "cheat" by sending a fake transcript while interacting with $\mathcal{B}$.

Claim 7. There exists a negligible function $\nu(\cdot)$ such that, for all $n \in \mathbb{N}$, the probability, over all randomness in the experiment $[\mathcal{R}^{\mathcal{A}^{\mathcal{O}}} \leftrightarrow \mathcal{C}](1^n)$, that some instance of $\mathcal{A}$ returns a forgery (i.e., something besides $\perp$) to $\mathcal{R}$ without having received responses to *all* $\ell(n) - 1$ (Open, i) (key-opening) queries from $\mathcal{R}$, is less than $\nu(n)$.

Proof. We demonstrate that, if $\mathcal{A}$ returns a forgery (i.e., not $\perp$) to $\mathcal{R}$ after $\mathcal{R}$ responds to strictly fewer than $\ell(n) - 1$ distinct key-opening queries from $\mathcal{A}$, then this requires $\mathcal{R}$ to guess a uniformly random message generated using the output of $\mathcal{A}$'s random oracle $\mathcal{O}$ on a new input, which can happen with at most probability $p(n)/|\mathcal{M}_n|$ for some polynomial $p(\cdot)$ due to $\mathcal{O}$ being uniformly random.

Assume that $\mathcal{R}$ responds to fewer than $\ell(n) - 1$ key-opening queries. Then there exists some $i \in [\ell(n) - 1]$ for which $\mathcal{R}$ does not send $\mathcal{A}$ a partial transcript ending with $((\mathsf{Open}, \pi_i), k_{\pi_i})$ (i.e., a response to $\mathcal{A}$'s i^{th} key-opening query). By the definition of the Valid predicate, in order for $\mathcal{R}$ to receive a final message from $\mathcal{A}$ that contains a forgery (and not $\perp$), $\mathcal{R}$ must send to $\mathcal{A}$ a complete transcript

$$\tau = \tau_1 || (\ldots, (\mathsf{Open}, \pi_i), k_{\pi_i}, (\mathsf{Query}, q, \omega_{i+1}), \ldots)$$

where ω_{i+1} is a uniformly random message generated by random coins resulting from applying $\mathcal{O}$ to $\tau_1 || (s, q(n) + 1, i + 1, 1)$.

By construction of $\mathcal{A}$ and the assumption that $\mathcal{R}$ does not send $\mathcal{A}$ a partial transcript ending with $((\mathsf{Open}, \pi_i), k_{\pi_i})$, however, $\mathcal{R}$ can never have received either ω_{i+1} or any message depending on the correct input $\tau_1 || (s, q(n) + 1, i + 1, 1)$ to $\mathcal{O}$. Hence, since ω_{i+1} is uniformly distributed and independent of any other message, we can conclude that $\mathcal{R}$ will send the correct ω_{i+1} in its final transcript with at most probability $1/|\mathcal{M}_n|$ (i.e., by guessing a random message correctly). While $\mathcal{R}$ can attempt to retrieve a forgery multiple times, it is restricted

to polynomial time, so the probability with which it can guess ω_{i+1} (which is necessary to receive a forgery from $\mathcal{A}$) is bounded above by $\nu(n) = p(n)/|\mathcal{M}_n|$ for polynomial $p(\cdot)$, which is negligible because we assume the message space to be super-polynomial (asymptotically greater than any polynomial) in n.

□

2.4.4 Analyzing the Meta-Reduction

The remaining part of the proof is devoted to analyzing the success probability $\mathsf{Success}_{\mathcal{R}^{\mathcal{A}}}(n)$. This, as previously discussed, involves investigating the probability with which the meta-reduction $\mathcal{B}$ and the ideal adversary $\mathcal{R}^{\mathcal{A}}$ diverge while interacting with $\mathcal{C}$. We formalize this with the following claim:

Claim 8. If $(\mathcal{C}, t(\cdot))$ is a secure assumption and we can bound

$$\Pr[\langle \mathcal{R}^{\mathcal{A}}, \mathcal{C} \rangle \rightarrow \mathsf{Accept}] - \Pr[\langle \mathcal{B}, \mathcal{C} \rangle \rightarrow \mathsf{Accept}] \leq p(n)$$

then there is a negligible $\epsilon(\cdot)$ such that $\mathsf{Success}_{\mathcal{R}^{\mathcal{A}}}(n) \leq p(n) + \epsilon(n)$.

Proof. Since $\mathcal{B}$ is efficient and $(\mathcal{C}, t(\cdot))$ is secure, there is a negligible $\epsilon(\cdot)$ such that $\Pr[\langle \mathcal{B}, \mathcal{C} \rangle \rightarrow \mathsf{Accept}] \leq t(n) + \epsilon(n)$.

So, given $\Pr[\langle \mathcal{R}^{\mathcal{A}}, \mathcal{C} \rangle \rightarrow \mathsf{Accept}] - \Pr[\langle \mathcal{B}, \mathcal{C} \rangle \rightarrow \mathsf{Accept}] \leq p(n)$, then we conclude that $\Pr[\langle \mathcal{R}^{\mathcal{A}}, \mathcal{C} \rangle \rightarrow \mathsf{Accept}] \leq t(n) + p(n) + \epsilon(n)$, and thus:

$$\mathsf{Success}_{\mathcal{R}^{\mathcal{A}}}(n) = \Pr[\langle \mathcal{R}^{\mathcal{A}}, \mathcal{C} \rangle \rightarrow \mathsf{Accept}] - t(n) \leq p(n) + \epsilon(n)$$

□

- Set initial view $v \leftarrow \perp$ and set $J \leftarrow 1$. Execute $\mathcal{R}$, updating the current view v according to the following rules.
- When $\mathcal{R}$ begins a new instance of $\mathcal{A}$ with some message (Init, s), label this instance as instance J. Generate and store $\ell(n)q(n)$ uniformly random queries $\overrightarrow{m}^1_J = (m^1_{J,1,1}, \ldots, m^1_{J,q(n),\ell(n)})$. Also initialize a variable $k_J \leftarrow \{\}$. Lastly, respond with $\tau^*_J = (\mathsf{Query}, 1, m^1_{J,1,1})$ and increment J.
- When $\mathcal{R}$ attempts to communicate externally with $\mathcal{C}$, forward the message, return $\mathcal{C}$'s response to $\mathcal{R}$, and update v accordingly.
- For any $i \in [q(n)]$, when $\mathcal{R}$ sends to some instance I of $\mathcal{A}$ a transcript of the form

  $$\tau = (q_{1,1}, q_{1,2}, \ldots, q_{1,\ell(n)}, q_{2,1}, \ldots, q_{i,j})$$

  where either $i < q(n)$ or $i = q(n)$ and $j < \ell(n)$, such that each $q_{u,v}$ is of the form $((\mathsf{Query}, v, m_{u,v}), \sigma_{u,v})$, do the following:
  - Let $j' = (j \bmod \ell(n)) + 1$.
  - Let $i' = i + 1$ if $j' = 1$ and $i' = i$ otherwise.
  - Return the response $(\mathsf{Query}, j', m^1_{I,i',j'})$.
- When $\mathcal{R}$ sends to some instance I of $\mathcal{A}$ a transcript of the form $\tau = \tau_1 || \tau_2$, where

  $$\tau_1 = (q_{1,1}, q_{1,2}, \ldots, q_{1,\ell(n)}, q_{2,1}, \ldots, q_{q(n),\ell(n)})$$

  and where each $q_{u,v}$ is of the form $((\mathsf{Query}, v, m_{u,v}), \sigma_{u,v})$, do the following:
  - Let c be the number of Open messages appearing in τ_2 so far.
  - If there is some tuple $(\tau_1, I, \pi, \overrightarrow{\omega}, m^*)$ stored, let π, $\overrightarrow{\omega}$, and m^* be as stored in the tuple. Otherwise, let $\pi = (\pi_1, \ldots, \pi_{\ell(n)})$ be a uniformly random permutation of $[\ell(n)]$, generate $2\ell(n)$ additional messages $\overrightarrow{\omega} = (\omega^0_1, \ldots, \omega^0_{\ell(n)}, \omega^1_1, \ldots, \omega^1_{\ell(n)})$ and a target forgery m^*, and store the tuple $(\tau_1, I, \pi, \overrightarrow{\omega}, m^*)$.
  - Consider the suffix transcript τ_2. If τ_2 is empty or ends with messages of the form $((\mathsf{Open}, j), k_j)$, then return $(\mathsf{Query}, q, \omega^{\mathsf{Valid}(\tau,I)}_{c+1})$, where q is the lexicographically first instance for which τ_2 does not contain an Open query.
  - Otherwise, if $c < \ell(n) - 1$ and τ_2 ends with messages of the form $((\mathsf{Query}, q, \omega_{c+1}), \cdot)$, then return $(\mathsf{Open}, \pi_{c+1})$.
  - Otherwise, if τ_2 ends with $((\mathsf{Query}, q, \omega_{\ell(n)}), \cdot)$ and $c = \ell(n) - 1$, generate a forgery as follows:
    - If $\mathsf{Valid}(\tau, I) = 0$, then return $\perp$.
    - Otherwise, run the procedure Rewind detailed below for the instance I.
    - If, after running Rewind, there is a stored key k_I, then return the forgery $(m^*, \mathsf{Tag}_{k_I}(m^*), \pi_{\ell(n)})$ and continue executing $\mathcal{R}$ as above. Otherwise, abort the entire execution of $\mathcal{B}$ and return Fail.

Figure 2.5: Formal description of the meta-reduction $\mathcal{B}$ (1).

Let the predicate $\mathsf{Valid}(\tau, I)$ be defined as follows:

- Parse τ as $\tau_1 || \tau_2$, where $\tau_1 = (q_{1,0}, q_{1,1}, \ldots, q_{1,\ell(n)-1}, q_{2,0}, \ldots, q_{q(n),\ell(n)})$ such that each $q_{u,v}$ is of the form $((\mathsf{Query}, v, m_{u,v}), \sigma_{u,v})$. If τ cannot be parsed as such, return 0.
- If there is a stored tuple $(\tau_1, I, \pi, \overrightarrow{\omega}, m^*)$ then set $\pi = (\pi_1, \ldots, \pi_{\ell(n)})$ equal to the third element of this tuple and set $\overrightarrow{\omega} = (\omega_1^0, \ldots, \omega_{\ell(n)}^0, \omega_1^1, \ldots, \omega_{\ell(n)}^1)$ equal to the fourth element. If there is no such tuple then return 0.
- Parse $\tau_2 = (q_1^*, q_2^*, \ldots, q_c^*[, q_{c+1}^*])$ such that each q_i^* is of the form $((\mathsf{Query}, q_i, \omega_i), \cdot, (\mathsf{Open}, \pi_i), k_{\pi_i})$ and q_{c+1}^*, if present, is of the form $((\mathsf{Query}, q_i, \omega_{c+1}), \cdot)$. If τ_2 cannot be parsed as such, or if $c > \ell(n) - 1$, return 0.
- Verify that each q_i in τ_2 is equal to the lexicographically first instance $q \in [\ell(n)]$ such that q does not appear in an Open query earlier in τ_2. If not true, return 0.
- Verify that, for all $i \in [q(n)]$ and all $j \in \{\pi_1, \ldots, \pi_c\}$, $\mathsf{Ver}_{k_j}(m_{i,j}, \sigma_{i,j}) = \mathsf{Accept}$. (Do not verify the responses to queries ω_i in τ_2.) If not true, return 0.
- Verify that every $m_{i,j}$ parsed from the transcript τ_1 is equal to the stored $m_{I,i,j}^1$, and that every ω_j parsed from τ_2 is equal to the respective ω_j^1. If not, then return 0. Otherwise, return 1.

Rewind procedure:

- Given instance I, for $j \in [\ell(n)]$ let V^j denote the view immediately before the query $(\omega_{\pi_j}, (\mathsf{Open}, \pi_j))$ for instance I (i.e., the query corresponding to the opening of the j^{th} instance after the order of instances π to open is randomized).
- For $j \in [\ell(n)]$, "rewind" the view to V^j as follows: Let $J' \leftarrow J$, let π' be identical to π except with $\pi_{\ell(n)}$ and π_j swapped (i.e., $\pi'_j = \pi_{\ell(n)}$ and $\pi'_{\ell(n)} = \pi_j$), and begin executing $\mathcal{R}$ from the view $V' \leftarrow V^j$ as in the main routine, with the following exceptions:
  - Replace any instances of π with π' (including in Valid).
  - When $\mathcal{R}$ begins a new instance of $\mathcal{A}$, label this instance as instance J' and increment J'.
  - When $\mathcal{R}$ attempts to communicate externally with $\mathcal{C}$ or "rewind" the current instance of $\mathcal{A}$ by sending a message corresponding to a point in the interaction before V^j, abort the rewinding and continue to the next repetition.
  - When $\mathcal{R}$ sends an end message for a valid instance $I' \neq I$ of $\mathcal{A}$ (i.e., a transcript τ such that $\mathcal{A}$'s next message would be a forgery (m^*, σ^*) for instance I'), abort the rewinding and continue to the next repetition. (If instead $\mathcal{A}$'s next message would be $\perp$ because $\mathsf{Valid}(\tau, I) = 0$, return $\perp$.)
  - If v' ever contains a message whose transcript contains a response k_I to any query for $(\mathsf{Open}, \pi_{\ell(n)})$ (i.e., (Open, π'_k)), then, if it is the case that $\mathsf{Ver}_{k_I}(m_{I,i,\pi_{\ell(n)}}, \sigma_{I,i,\pi_{\ell(n)}}) = \mathsf{Accept}$ for every $i \in [q(n)]$ (letting m and σ variables be defined as in the Valid predicate), store k_I and end the Rewind procedure (i.e., return to the outer execution); otherwise, if k_I is not a correct key, store nothing to k_I and continue to the next repetition.

Figure 2.6: Formal description of the meta-reduction $\mathcal{B}$ (2).

So it suffices to bound $\Pr[\langle \mathcal{R}^{\mathcal{A}}, \mathcal{C} \rangle \rightarrow \mathsf{Accept}] - \Pr[\langle \mathcal{B}, \mathcal{C} \rangle \rightarrow \mathsf{Accept}]$. In order to do so, we begin by defining an *inefficient* "hybrid" meta-reduction $\mathcal{B}'$ which acts identically to $\mathcal{B}$, with the sole exception that, during the Rewind procedure, if $\mathcal{B}'$ encounters a response k_I to a query for $(\mathsf{Open}, \pi_{\ell(n)})$ (i.e., a key for the instance for which $\mathcal{B}'$ must produce a forgery), and if the recovered k_I is valid (i.e., $\mathsf{Ver}_{k_I}(m_{I,i,\pi_{\ell(n)}}, \sigma_{I,i,\pi_{\ell(n)}}) = \mathsf{Accept}$ for every $i \in [q(n)]$), then $\mathcal{B}'$ will first determine, using brute force, whether there are any other keys k' such that $\mathsf{Ver}_{k'}(m_{I,i,\pi_{\ell(n)}}, \sigma_{I,i,\pi_{\ell(n)}}) = \mathsf{Accept}$ for every $i \in [q(n)]$ but $\mathsf{Tag}_{k'}(m_I^*) \neq \mathsf{Tag}_{k_I}(m_I^*)$. If not (i.e., either k_I is the only such key or there is a unique correct forgery (m_I^*, σ_I^*)), then $\mathcal{B}'$ stores k_I, identically to $\mathcal{B}$; otherwise, $\mathcal{B}'$ stores the *lexicographically first* such key k' and uses that key instead of k_I to produce the forgery (identically to $\mathcal{A}$).

For ease of notation, let us further define some experiments and variables:

- Let $\mathsf{Real}(1^n)$ denote the experiment $[\mathcal{B} \leftrightarrow \mathcal{C}](1^n)$, and $\mathsf{Output}[\mathsf{Real}(1^n)]$ the output distribution $\langle \mathcal{B}, \mathcal{C} \rangle(1^n)$. Let $\mathsf{Hyb}(1^n)$ and $\mathsf{Output}[\mathsf{Hyb}(1^n)]$ be defined analogously for the "hybrid" experiment $[\mathcal{B}' \leftrightarrow \mathcal{C}](1^n)$, and lastly $\mathsf{Ideal}(1^n)$ and $\mathsf{Output}[\mathsf{Ideal}(1^n)]$ for the "ideal" experiment $[\mathcal{R}^{\mathcal{A}} \leftrightarrow \mathcal{C}](1^n)$.
- For any such experiment, let $\{\overrightarrow{m}_I, \pi_I\}$ define the randomness used to generate, respectively, all query variables ($m_{(\cdot)}$ or $\omega_{(\cdot)}$) and the permutation π for an instance I (real or simulated) of $\mathcal{A}$ (including the case where a query or permutation might be regenerated after, e.g., rewinding). Let $\mathcal{O}_{ext}$ denote all other randomness. Furthermore, let $M(n)$ be an upper bound to the number of instances of $\mathcal{A}$ started by $\mathcal{R}$.
- For instance, an experiment $\mathsf{Real}_{\{\overrightarrow{m}_I, \pi_I\}_{I \in [M(n)] \setminus J}, \mathcal{O}_{ext}}(1^n)$ (which we henceforth abbreviate as $\mathsf{Real}_{\{\overrightarrow{m}_I, \pi_I\}_{-J}, \mathcal{O}_{ext}}(1^n)$) would indicate the interaction between $\mathcal{B}$ and $\mathcal{C}$ with all randomness fixed *except* for the variables m and π for a particular instance J of $\mathcal{A}$ (simulated by $\mathcal{B}$).

- Naturally, an experiment denoted by, e.g., $\mathsf{Real}_{\{\overrightarrow{m}_I,\pi_I\}_{I\in[M(n)]},\mathcal{O}_{ext}}(1^n)$, has all randomness fixed and hence is deterministic.

Let $\mathsf{Unique}(\{\overrightarrow{m}_I,\pi_I\}_{I\in[M(n)]},\mathcal{O}_{ext})$ be the "key-uniqueness" predicate on the randomness of Real (or Ideal) which is true if, during execution of the experiment $\mathsf{Real}_{\{\overrightarrow{m}_I,\pi_I\}_{I\in[M(n)]},\mathcal{O}_{ext}}(1^n)$, whenever $\mathcal{B}$ returns a forgery (m^*,σ^*), it is the case that $\sigma^* = \mathsf{Tag}_{k^*}(m^*)$, where k^* is the lexicographically first key k such that $\mathsf{Ver}_k(m_{I,i,\pi_{I,j}},\sigma_{I,i,\pi_{I,j}}) = \mathsf{Accept}$ for all $i \in [q(n)]$. That is, Unique is true whenever, given the randomness of an experiment, $\mathcal{B}$ (if rewinding succeeds) returns the same forgery as $\mathcal{A}$ would in the Ideal experiment. The occurrence of Unique is hence fully determined by the randomness ($\{\overrightarrow{m}_I,\pi_I\}_{I\in[M(n)]}$ and $\mathcal{O}_{ext}$) that fully determines the execution of Real or Ideal.

We must also deal with the fact that $\mathcal{R}$ may rewind $\mathcal{A}$. Let $W(n)$ be a polynomial upper bound to the number of times that $\mathcal{R}$ causes $\mathcal{A}$ to generate a permutation π (including by rewinding) in the experiment $\mathsf{Ideal}(1^n)$, and note that, trivially, $W(n) \geq M(n)$.

Now, with setup completed, we can proceed in two major steps. Our goal is to bound

$$|\Pr[\mathsf{Output}[\mathsf{Real}(1^n)] = \mathsf{Accept}] - \Pr[\mathsf{Output}[\mathsf{Ideal}(1^n)] = \mathsf{Accept}]|$$

which we can do by bounding

$$|\Pr[\mathsf{Output}[\mathsf{Real}(1^n)] = \mathsf{Accept}] - \Pr[\mathsf{Output}[\mathsf{Hyb}(1^n)] = \mathsf{Accept}]|$$

and

$$|\Pr[\mathsf{Output}[\mathsf{Hyb}(1^n)] = \mathsf{Accept}] - \Pr[\mathsf{Output}[\mathsf{Ideal}(1^n)] = \mathsf{Accept}]|$$

2.4.5 Comparing the Real and Hybrid Experiments

We begin with the first of these quantities, which is relatively straightforward to bound. Informally, whenever Unique holds (the probability of which is dictated by Lemma 6), $\mathcal{B}$ and $\mathcal{B}'$ behave identically by construction.

Claim 9. There exists negligible $\nu(\cdot)$ such that, for all $n \in \mathbb{N}$:

$$|\Pr[\mathsf{Output}[\mathsf{Real}(1^n)] = \mathsf{Accept}] - \Pr[\mathsf{Output}[\mathsf{Hyb}(1^n)] = \mathsf{Accept}]|$$
$$< \frac{2nW(n)}{q(n)} + \nu(n)$$

taken over the randomness of $\{\overrightarrow{m}_I, \pi_I\}_{I \in [M(n)]}$ and $\mathcal{O}_{ext}$.

Proof. This follows fairly easily from the construction of $\mathcal{B}'$ and definition of Unique. Recall that $\mathsf{Unique}(\{\overrightarrow{m}_I, \pi_I\}_{I \in [M(n)]}, \mathcal{O}_{ext})$ is true exactly when $\mathcal{B}$ would return the same forgery as $\mathcal{A}$ given successful rewinding; since $\mathcal{B}'$ by construction succeeds at rewinding whenever $\mathcal{B}$ does yet always returns the same forgery as $\mathcal{A}$, we can see that $\mathsf{Real}_{\{\overrightarrow{m}_I, \pi_I\}_{I \in [M(n)]}, \mathcal{O}_{ext}}(1^n)$ is identically distributed to $\mathsf{Hyb}_{\{\overrightarrow{m}_I, \pi_I\}_{I \in [M(n)]}, \mathcal{O}_{ext}}(1^n)$ whenever $\mathsf{Unique}(\{\overrightarrow{m}_I, \pi_I\}_{I \in [M(n)]}, \mathcal{O}_{ext})$ is true.

By Lemma 6, we can see that, whenever $\mathcal{B}$ (or $\mathcal{B}'$) generates a permutation π and forgery input m^*, $\mathsf{Unique}(\{\overrightarrow{m}_I, \pi_I\}_{I \in [M(n)]}, \mathcal{O}_{ext})$ has at most a $2n/q(n) + \nu'(n)$ chance (for negligible $\nu'(\cdot)$) to *not* hold for the respective forgery if it is eventually returned, since, by the lemma and the fact that the key retrieved by $\mathcal{B}$ must agree with the lexicographically first key k^* for every one of the $q(n)$ tag queries, it must (with probability $1 - \nu'(n)$) agree with k^* for at least a $1 - 2n/q(n)$ fraction of inputs m^*.

Since by assumption π (and m^*) are generated at most $W(n)$ times during the execution of Real or Hyb, the probability (over $\{\overrightarrow{m}_I, \pi_I\}_{I \in [M(n)]}$ and $\mathcal{O}_{ext}$) that

$\mathsf{Unique}(\{\overrightarrow{m}_I, \pi_I\}_{I\in[M(n)]}, \mathcal{O}_{ext})$ fails, and hence the probability that Real and Hyb diverge, is, by the union bound (and letting $\nu(n) = W(n)\nu'(n)$), at most

$$\frac{2nW(n)}{q(n)} + \nu(n)$$

as desired.

□

2.4.6 Comparing the Hybrid and Ideal Experiments

To relate the hybrid $\mathcal{B}'$ to the "ideal" interaction with $\mathcal{R}^{\mathcal{A}}$, we next present the following claim, which informally holds because, by construction, $\mathcal{B}'$ behaves identically to $\mathcal{A}$ as long as rewinding does not fail (in which case it would return Fail).

Claim 10.

$$|\Pr[\mathsf{Output}[\mathsf{Hyb}(1^n)] = \mathsf{Accept}] - \Pr[\mathsf{Output}[\mathsf{Ideal}(1^n)] = \mathsf{Accept}]|$$

$$\leq \Pr[\mathsf{Output}[\mathsf{Hyb}(1^n)] = \mathsf{Fail}]$$

taken over the randomness of $\{\overrightarrow{m}_I, \pi_I\}_{I\in[M(n)]}$ and $\mathcal{O}_{ext}$.

Proof. This follows immediately from the lemma below:

Lemma 7. For any assignment of $\{\overrightarrow{m}_I, \pi_I\}_{I\in[M(n)]}$ and $\mathcal{O}_{ext}$, either:

- $\mathsf{Output}[\mathsf{Hyb}_{\{\overrightarrow{m}_I,\pi_I\}_{I\in[M(n)]},\mathcal{O}_{ext}}(1^n)] = \mathsf{Fail}$, or
- $\mathsf{Ideal}_{\{\overrightarrow{m}_I,\pi_I\}_{I\in[M(n)]},\mathcal{O}_{ext}}(1^n) = \mathsf{Hyb}_{\{\overrightarrow{m}_I,\pi_I\}_{I\in[M(n)]},\mathcal{O}_{ext}}(1^n)$, and thus $\mathsf{Output}[\mathsf{Ideal}_{\{\overrightarrow{m}_I,\pi_I\}_{I\in[M(n)]},\mathcal{O}_{ext}}(1^n)] = \mathsf{Output}[\mathsf{Hyb}_{\{\overrightarrow{m}_I,\pi_I\}_{I\in[M(n)]},\mathcal{O}_{ext}}(1^n)]$

Proof. First, observe that, given the same (fixed) queries $m_{(\cdot)}$ and $\omega_{(\cdot)}$ and permutation π for each instance, and since $\mathcal{C}$ is the same in each case, the only points at which the views of Real (and hence Hyb) and Ideal can possibly diverge is when $\mathcal{B}$ (or $\mathcal{B}'$ or $\mathcal{A}$) generates a forgery. This is by construction; given a particular setting of $\{\overrightarrow{m}_I, \pi_I\}_{I\in[M(n)]}$ and $\mathcal{O}_{ext}$, an instance I of $\mathcal{A}$ and its simulation in $\mathcal{B}'$ are guaranteed to behave identically up to the point where a forgery is produced (assuming no other instance diverges before that point), as the queries they make to $\mathcal{C}$ are identical by fixing $\{\overrightarrow{m}_I, \pi_I\}$ and $\mathcal{C}$'s responses are identical by fixing $\mathcal{O}_{ext}$.

So, consider any point at which $\mathcal{B}'$ generates a forgery in Hyb, and assume that rewinding was successful (i.e., $\mathcal{B}'$ does not output Fail) and that Ideal and Hyb have not diverged yet. If there is a unique correct forgery to return at this point, $\mathcal{B}'$ will by construction return it, as will $\mathcal{A}$. Conversely, if there are multiple possible forgeries, then $\mathcal{B}'$ will in fact *also* behave identically to $\mathcal{A}$ by brute-forcing the lexicographically first key consistent with all tag queries and using that to generate a forgery guess. Either way, if $\mathcal{B}'$ does not return Fail (i.e., $\mathsf{Output}[\mathsf{Hyb}_{\{\overrightarrow{m}_I,\pi_I\}_{I\in[M(n)]},\mathcal{O}_{ext}}(1^n)] \neq \mathsf{Fail}$), then Ideal and Hyb will continue to behave identically, and thus, proceeding in this fashion over all forgeries produced, as long as $\mathcal{B}'$ never returns Fail we can conclude that

$$\mathsf{Ideal}_{\{\overrightarrow{m}_I,\pi_I\}_{I\in[M(n)]},\mathcal{O}_{ext}}(1^n) = \mathsf{Hyb}_{\{\overrightarrow{m}_I,\pi_I\}_{I\in[M(n)]},\mathcal{O}_{ext}}(1^n)$$

as desired. □

□

2.4.7 Bounding the Hybrid's Failure Probability

So all that remains is to investigate the probability of Hyb outputting Fail; to do this we can make a critical observation about rewinding in the context of our construction. Formally, we prove the following:

Proposition 2. There exists a negligible function $\epsilon(\cdot)$ such that, for all $n \in \mathbb{N}$, taken over the randomness of $\{\overrightarrow{m}_I, \pi_I\}_{I \in [M(n)]}$ and $\mathcal{O}_{ext}$:

$$\Pr[\mathsf{Output}[\mathsf{Hyb}(1^n)] = \mathsf{Fail}] \leq W(n) \left(\frac{W(n) + r(n) + 1}{\ell(n)} \right) + \epsilon(n)$$

Proof. First, we show that without loss of generality $\mathcal{R}$ can never rewind *except* from a point after π is generated to a point before π is generated; intuitively, this is because all of $\mathcal{A}$'s queries to $\mathcal{R}$ are dependent only on (1) the permutation π and (2) the validity of $\mathcal{R}$'s responses, and as such any rewinding that does not result in π being regenerated can in fact be internally simulated by $\mathcal{R}$. Formally, we state the following claim:

Claim 11. Given any $\mathcal{R}$ that rewinds any instance of $\mathcal{A}$ either (1) from a point before π is generated or (2) to a point after π is generated, there exists an $\mathcal{R}'$ with identical success probability that does not perform such rewinding.

Proof. Given any rewinding of type (1), we notice that $\mathcal{A}$'s or $\mathcal{B}$'s tag queries are completely determined on initialization; hence, we can simply let $\mathcal{R}'$ completely skip the pre-rewinding messages to effectively "collapse" this rewinding into a single execution thread, as rewinding in this case will not change any of the queries made by $\mathcal{A}$.

Similarly, given any rewinding of type (2), we can observe that the order π of key-opening

queries made by $\mathcal{A}$ (as well as the order of the instances q to which the tag queries are made, since this solely depends on π) is completely determined at the time that π is generated.

While the inputs ω to the tag queries and the final forgery output depend on the Valid predicate as well, one can additionally observe that, *as long as the* Valid *predicate holds*, these inputs and the forgery are also fully determined when π is generated, *even if* Unique *does not hold*. Specifically, this is because $\mathcal{A}$ and $\mathcal{B}'$ generate the forgery input m^* when π is generated, and because the key used to produce the forgery is uniquely determined then as well (by lexicographical ordering and by the fact that the responses to ω_i are never verified and hence cannot change the set of usable keys). Furthermore, the predicate Valid can be computed entirely by $\mathcal{R}$ given a particular transcript, since all information $(f, \pi, \overrightarrow{\omega})$ required to compute it is either public or given by $\mathcal{A}$ during the execution which produces the transcript.

Hence, $\mathcal{R}'$ can again collapse this latter type of rewinding by restricting to transcripts where Valid is true (as if Valid is false then it is guaranteed to receive $\perp$ instead of a forgery) and again skipping pre-rewinding messages (as they have no effect on $\mathcal{A}$'s queries and output given that Valid is true). $\square$

Hence, we assume without loss of generality that $\mathcal{R}$ sends at most $W(n)$ "end messages" (i.e., forgery requests) requiring rewinding, as π is by assumption generated no more than $W(n)$ times and the responses to any further end messages are effectively simulatable by $\mathcal{R}$. We disregard end messages sent for instances for which $\mathcal{R}$ has not answered all $\ell(n) - 1$ key opening queries, since, with all-but-negligible probability, $\mathcal{A}$ or $\mathcal{B}'$ can directly respond to these with $\perp$ (as Valid will evaluate to 0 unless $\mathcal{R}$ guesses a random and unknown ω_i correctly).

At this point, we have shown that our hybrid experiment gives us a setting with minimal rewinding and guaranteed key uniqueness, much like the setting discussed in [98] for the case of unique signatures. Hence, we can leverage this observation to prove the following claim, analogous to the key "rewinding lemma" therein. Consider the following for any possible execution $\mathsf{Hyb}_{\{\vec{m}_I,\pi_I\}_{-J},\vec{m}_J,\mathcal{O}_{ext}}(1^n)$ (i.e., for any fixed setting of all randomness aside from π_J), and notice that, since it applies to arbitrarily fixed randomness, it must thus apply over all possible randomness of the experiment $\mathsf{Hyb}(1^n)$:

Claim 12. Given any experiment $\mathsf{Hyb}_{\{\vec{m}_I,\pi_I\}_{-J},\vec{m}_J,\mathcal{O}_{ext}}(1^n)$, the probability, over the uniformly chosen permutation π_J, that the simulated instance J will return Fail when rewinding any end message, is, for all $n \in \mathbb{N}$, at most

$$\frac{W(n) + r(n) + 1}{\ell(n)}$$

Proof. First, let us assume that other simulated instances (besides J) of $\mathcal{A}$ in the experiment $\mathsf{Hyb}_{\{\vec{m}_I,\pi_I\}_{-J},\vec{m}_J,\mathcal{O}_{ext}}(1^n)$ will never return Fail (i.e., that they will "magically" produce a correct forgery instead of returning Fail). This can only increase the probability that instance J will return Fail, as it guarantees that the experiment never aborts early.

Next, consider the permutation $\pi_J = (\pi_J^1, \ldots, \pi_J^{\ell(n)})$ in instance J; note that by the definition of $\mathsf{Hyb}_{\{\vec{m}_I,\pi_I\}_{-J},\vec{m}_J,\mathcal{O}_{ext}}(1^n)$ the execution is fully determined by π_J. Let us also define for $i \in [\ell(n) - 1]$ the "rewound" sequence

$$\rho(\pi_J, i) \triangleq (\pi_J^1, \ldots, \pi_J^{i-1}, \pi_J^{\ell(n)})$$

corresponding to the rewound execution where the key-opening query in the i^{th} position is replaced by $\pi_J^{\ell(n)}$ to attempt to extract a correct key for that instance.

In order for $\mathcal{B}'$ to return Fail, one of the following "bad events" must occur for each $i \in [\ell(n)]$:

- $E_1(\rho(\pi_J, i))$: $\mathcal{R}$ fails to return a valid key (i.e., one consistent with all tag queries $m_{(\cdot)}$) for instance $\pi_{\ell(n)}$ in the respective rewinding, including the case where $\mathcal{R}$ attempts to rewind $\mathcal{A}$ before responding.
- $E_2(\rho(\pi_J, i))$: During the respective rewinding, $\mathcal{R}$ asks for a forgery for another instance $I \neq J$ (for which $\mathcal{B}'$ would be required to rewind) before returning a key for instance $\pi_J^{\ell(n)}$, *or* $\mathcal{R}$ requires external communication with $\mathcal{C}$ (which cannot be rewound) before returning a response.

In addition, for instance J to fail, the non-rewound execution of the instance must succeed (and finish executing), in that the event $E_1(\pi_{J,\leq i})$ (where $\mathcal{R}$ fails to return a valid tag or rewinds) *cannot* occur for any prefix $\pi_{J,\leq i} = (\pi_J^1, \ldots, \pi_J^i)$, where $i \in [\ell(n) - 1]$.

Since, as we have noted, the behavior of J during the hybrid execution $\mathsf{Hyb}_{\{\overrightarrow{m}_I, \pi_I\}_{-J}, \overrightarrow{m}_J, \mathcal{O}_{ext}}(1^n)$ is fully determined by π_J for any fixed $\overrightarrow{m}_J$, every sequence π_J will deterministically either result in instance J returning something (either a forgery or $\perp$) or aborting and returning Fail; we shall refer to the former type of sequence (where J succeeds) as a "good" sequence, and the latter type as a "bad" sequence. Notationally, let $\leq$ applied to two sequences denote the prefix operator—that is, $s \leq t$ if $s_i = t_i$ for all $i \in |s|$. Then:

Lemma 8. Let S be a set of permutations of $[\ell]$. Let $T = \{s'||n : s||n \in S, s' < s\}$ be the set of "rewound" prefixes including, for any permutation $p \in S$ and integer $i < \ell$, p's first i elements concatenated with its last element. For any $s \in S$, let $P_s = \{p : p \leq s\}$ be the set of prefixes of elements in S. If there exists integer B such that, for all $s \in S$, $|P_s \cap T| \leq B$

(i.e., no $s \in S$ can share more than B prefixes with T), then the size of S is bounded by $|S| \leq (B+1)(\ell-1)!$

Proof. Identical to the proof of Lemma 4 in Section 2.3. □

The relevance of this lemma stems intuitively from the fact that, if there exists a "bad" permutation $\pi_J = (\pi_J^1, \ldots, \pi_J^{\ell(n)})$, then any sequence aside from π_J itself which begins with an element of the set $\{s' || \pi_J^{\ell(n)} : s' \leq \pi_J\}$ will, by the fact that π_J is bad, fail to extract a valid key as a response to $(\mathsf{Open}, \pi_J^{\ell(n)})$ during the rewinding. Of course, this can occur due to either of the events E_1 or E_2 described above; we would like to conclude that these sequences are themselves good, but we can only conclude that a sequence is good if E_1 specifically occurs.

This is where the lemma helps us; letting T be the set of all of the prefixes guaranteed by some bad sequence in S to fail to extract a key in the rewinding, we know that a sequence containing strictly more than $W(n) + r(n)$ of these prefixes (namely, $W(n)$ at most which might contain end messages for other instances because of our earlier assumptions about rewinding, and $r(n)$ which might contain external communication) must be a good sequence, since at that point one of the prefixes is guaranteed to fail due to E_1 and not E_2.

So, applying Lemma 8 with $\ell = \ell(n)$, $B = W(n) + r(n)$, and S being the set of bad permutations, we deduce that S may have size at most

$$(W(n) + r(n) + 1)\,(\ell(n) - 1)!$$

and so at most an

$$\frac{(W(n) + r(n) + 1)\,(\ell(n) - 1)!}{\ell(n)!} = \frac{W(n) + r(n) + 1}{\ell(n)}$$

fraction of all permutations of $[\ell(n)]$ can be bad (for any setting of the randomness outside of the permutation π_J), and the remainder must be good.

Hence, the chance that a permutation chosen at random is bad, which by definition is equal to the probability that a randomly chosen permutation $\pi_J = (\pi_1, \ldots, \pi_{\ell(n)})$ will result in instance J returning Fail, can be at most the fraction of permutations which are bad, or:

$$\frac{W(n) + r(n) + 1}{\ell(n)}$$

As this holds for arbitrary $\overrightarrow{m}_J$, it likewise holds over *all* $\overrightarrow{m}_J$ and thus over all randomness of the execution $\mathsf{Hyb}_{\{\overrightarrow{m}_I, \pi_I\}_{-J}, \overrightarrow{m}_J, \mathcal{O}_{ext}}(1^n)$. □

We can conclude as desired that the probability of any forgery request causing $\mathcal{B}'$ to return Fail in the experiment Hyb is at most

$$W(n)\left(\frac{W(n) + r(n) + 1}{\ell(n)}\right) + \epsilon(n)$$

by combining Claim 12 (taken over all possible assignments of the fixed randomness) with the union bound over our bound of $W(n)$ possible (unique) forgery requests for which $\mathcal{R}$ has answered all key-opening queries. For any requests for which this is *not* the case, we know by Claim 7 that the probability of such requests causing $\mathcal{B}'$ to return anything besides $\perp$ is negligible, so, since $\mathcal{R}$ is polynomial-time, these requests add at most a negligible $\epsilon(n)$ to the probability of Hyb returning Fail.

□

2.4.8 Bounding the Security Loss

Finally, we must translate this bound on the failure probability of $\mathcal{B}'$ into a bound on the security loss of the reduction $\mathcal{R}$. This argument is quite similar to that presented in [98], albeit with the minor difference that the success probability of $\mathcal{A}$ is no longer 1 but instead very close to 1 for sufficiently large $q(n)$.

Take sufficiently large $q(n)$—say, $q(n) \geq 4n\ell(n)^2$. Recall that, by Claim 6, we know that, for sufficiently large n:

$$\mathsf{Success}_{\mathcal{A}}(n) \geq 1 - \frac{2n}{q(n)} - \nu(n) \geq 1 - \frac{1}{2\ell(n)^2}$$

By combining Claim 8, Claim 9, Claim 10, and Proposition 2, we can derive that, for negligible $\epsilon(\cdot), \nu(\cdot)$ and sufficiently large n:

$$\begin{aligned}\mathsf{Success}_{\mathcal{R}^{\mathcal{A}}}(n) &\leq \Pr[\mathsf{Output}[\mathsf{Hyb}(1^n)] = \mathsf{Fail}] + \frac{2nW(n)}{q(n)} + \nu(n) \\ &< W(n)\left(\frac{W(n) + r(n) + 1}{\ell(n)}\right) + \frac{2nW(n)}{q(n)} + (\epsilon(n) + \nu(n)) \\ &< W(n)\left(\frac{W(n) + r(n) + 2}{\ell(n)}\right)\end{aligned}$$

To deal with running time, Lemma 5 (Appendix B) of [98] shows that we can substitute the number of times a specific computation is performed (in this case, the number of times π is generated for some instance of $\mathcal{A}$) for Time and receive a lower bound to the time-based security loss; effectively, this follows because we can rewrite $\mathcal{A}$ so that the computation dominates the running time (for both $\mathcal{A}$ and $\mathcal{R}$). Formally, if we let $\mathsf{Query}^{\pi}_{\mathcal{A}}(n)$ denote the number of times π is generated during $\mathcal{A}$, and respectively for $\mathcal{R}^{\mathcal{A}}$, this lemma gives:

$$\lambda_{\mathcal{R}}(n) \geq \frac{\mathsf{Success}_{\mathcal{A}'}(n)}{\mathsf{Success}_{\mathcal{R}^{\mathcal{A}'}}(n)} \frac{\mathsf{Time}_{\mathcal{R}^{\mathcal{A}'}}(n)}{\mathsf{Time}_{\mathcal{A}'}(n)} \geq \frac{\mathsf{Success}_{\mathcal{A}}(n)}{\mathsf{Success}_{\mathcal{R}^{\mathcal{A}}}(n)} \frac{\mathsf{Query}^{\pi}_{\mathcal{R}^{\mathcal{A}}}(n)}{\mathsf{Query}^{\pi}_{\mathcal{A}}(n)}$$

By definition $\mathsf{Query}^{\pi}_{\mathcal{A}}(n) = 1$ and we may take $\mathsf{Query}^{\pi}_{\mathcal{R}^{\mathcal{A}}}(n) = W(n)$. We can separate our analysis into two cases to finish:

Case 1. If $W(n) \geq \sqrt{\ell(n)} - (r(n)+2)$, then:

$$\lambda_{\mathcal{R}}(n) \geq \frac{\mathsf{Success}_{\mathcal{A}}(n)}{\mathsf{Success}_{\mathcal{R}^{\mathcal{A}}}(n)} \frac{\mathsf{Query}^{\pi}_{\mathcal{R}^{\mathcal{A}}}(n)}{\mathsf{Query}^{\pi}_{\mathcal{A}}(n)} \geq \frac{1 - \frac{1}{2\ell(n)^2}}{1} \frac{W(n)}{1}$$

$$= \left(1 - \frac{1}{2\ell(n)^2}\right) W(n) \geq \left(1 - \frac{1}{2\ell(n)^2}\right) (\sqrt{\ell(n)} - (r(n)+2))$$

Case 2. Otherwise, $W(n) < \sqrt{\ell(n)} - (r(n)+2)$ then $W(n) + r(n) + 2 < \sqrt{\ell(n)}$, and we have:

$$\lambda_{\mathcal{R}}(n) \geq \frac{\mathsf{Success}_{\mathcal{A}}(n)}{\mathsf{Success}_{\mathcal{R}^{\mathcal{A}}}(n)} \frac{\mathsf{Query}^{\pi}_{\mathcal{R}^{\mathcal{A}}}(n)}{\mathsf{Query}^{\pi}_{\mathcal{A}}(n)} \geq \frac{1 - \frac{1}{2\ell(n)^2}}{W(n)\left(\frac{W(n)+r(n)+2}{\ell(n)}\right)} \frac{W(n)}{1}$$

$$= \frac{\ell(n)\left(1 - \frac{1}{2\ell(n)^2}\right)}{W(n)+r(n)+2} > \frac{\ell(n)\left(1 - \frac{1}{2\ell(n)^2}\right)}{\sqrt{\ell(n)}} = \left(1 - \frac{1}{2\ell(n)^2}\right)\sqrt{\ell(n)}$$

Either way, we conclude that, if $(\mathcal{C}, t(\cdot))$ is secure (and so our bound on the probability $\mathsf{Success}_{\mathcal{R}^{\mathcal{A}}}(n)$ holds), then:

$$\lambda_{\mathcal{R}}(n) \geq \left(1 - \frac{1}{2\ell(n)^2}\right) (\sqrt{\ell(n)} - (r(n)+2))$$

which finishes the proof of Lemma 5.

CHAPTER 3

FEASIBILITY RESULTS FOR SECURE COMPUTATION

3.1 Introduction

For the latter part of the technical section, we present our positive results; these results demonstrate feasibility of efficient notions of non-interactive secure two-party computation. As the sets of definitions for the two results are mostly independent from one another, we present the preliminaries for each result immediately prior to the result itself, in Sections 3.2 and 3.4. Section 3.3 contains our construction of NISC with a *succinctness* property, while Section 3.5 contains our construction of NISC satisfying (oracle-aided) universally composable security (which vastly improves on the best prior known round complexity of UC-secure two-party computation); both sections lead with a technical overview and high-level description of the respective protocols before stating the formal protocols and their respective theorems and proofs. Finally, in Section 3.6, we present the matching converse to our UC-secure NISC result, showing that it is based on essentially minimal assumptions and thus achieving a nearly tight characterization of the necessary and sufficient assumptions.

3.2 Preliminaries

Notation. Let $\mathbb{N}$ denote the set of natural numbers (positive integers), and let $[n]$ denote the set of natural numbers at most n, or $\{1, 2, \ldots, n\}$. For $n \in \mathbb{N}$, we denote by 1^n the string of n ones, which will be used to provide a security parameter as input to an algorithm (this is by convention, so that the input length is bounded below by the security parameter). We assume the reader is familiar with polynomial-time and probabilistic polynomial time (PPT)

algorithms. We say a function $\epsilon(\cdot)$ is *negligible* if, for any polynomial $p(\cdot)$, $\epsilon(n) < 1/p(n)$ for all sufficiently large $n \in \mathbb{N}$—that is, if $\epsilon(\cdot)$ is asymptotically smaller than any inverse polynomial.

3.2.1 Non-Interactive Secure Computation

Definition 10 (based on [127, 62, 10])**.** A **non-interactive two-party computation protocol** for computing some functionality $f(\cdot,\cdot)$ (where f is computable by a polynomial-time Turing machine) is given by three PPT algorithms $(\mathsf{NISC}_1, \mathsf{NISC}_2, \mathsf{NISC}_3)$ defining an interaction between a sender S and a receiver R, where only R will receive the final output. The protocol will have common input 1^n (the security parameter); the receiver R will have input x, and the sender will have input y. The algorithms $(\mathsf{NISC}_1, \mathsf{NISC}_2, \mathsf{NISC}_3)$ are such that:

- $(\mathsf{msg}_1, \sigma) \leftarrow \mathsf{NISC}_1(1^n, x)$ generates R's message msg_1 and persistent state σ (which is not sent to S) given the security parameter n and R's input x.
- $\mathsf{msg}_2 \leftarrow \mathsf{NISC}_2(\mathsf{msg}_1, y)$ generates S's message msg_2 given S's input y and R's message msg_1.
- $\mathit{out} \leftarrow \mathsf{NISC}_3(\sigma, \mathsf{msg}_2)$ generates R's output out given the state σ and S's message msg_2.

Furthermore, we require the following property:

- **Correctness.** For any parameter $n \in \mathbb{N}$ and inputs x, y:

$$\Pr\left[(\mathsf{msg}_1, \sigma) \leftarrow \mathsf{NISC}_1(1^n, x) : \mathsf{NISC}_3(\sigma, \mathsf{NISC}_2(\mathsf{msg}_1, y)) \neq f(x, y)\right] \leq \epsilon(n)$$

Defining non-interactive *secure* computation will require us to add a security definition, which we formalize as follows:

Security. We adopt a standard notion of *simulation-based security*, with the relaxation that we allow superpolynomial-time simulation (as pioneered by [109, 115]). We define security by comparing two experiments conducted between the sender and receiver, either of whom may be corrupted and act arbitrarily (while the other is honest and follows the protocol). In the *real* experiment, the two parties will perform the actual protocol; in the *ideal* experiment, the two parties will instead send their inputs to a "trusted third party" who performs the computation and returns the result only to, in this case (because the protocol is one-sided), the receiver. Informally, we say that a protocol is secure if, for any adversary $\mathcal{A}$ against the *real* experiment, acting either as the sender or receiver, there is a simulated adversary $\mathcal{S}$ in the *ideal* experiment which produces a near-identical (i.e., computationally indistinguishable) result; intuitively, if this is the case, we can assert that the real adversary cannot "learn" anything more than they could by interacting with a trusted intermediary. Let us formalize this notion for the case of SNISC:

- Let the *real* experiment be defined as an interaction between a sender S with input y and a receiver R with input x, defined as follows:
  - R computes $(\mathsf{msg}_1, \sigma) \leftarrow \mathsf{NISC}_1(1^n, x)$, stores σ, and sends msg_1 to S.
  - S, on receiving msg_1, computes $\mathsf{msg}_2 \leftarrow \mathsf{NISC}_2(\mathsf{msg}_1, y)$ and sends msg_2 to R.
  - R, on receiving msg_2 computes $out \leftarrow \mathsf{NISC}_3(\sigma, \mathsf{msg}_2)$ and outputs out.

  In this interaction, one party $I \in \{S, R\}$ is defined as the *corrupted* party; we additionally define an *adversary*, or a polynomial-time machine $\mathcal{A}$, which receives the security

parameter 1^n, an auxiliary input z, and the inputs of the corrupted party I, and sends messages (which it may determine arbitrarily) in place of I.

Letting Π denote the protocol to be proven secure, we shall denote by $\mathsf{Out}_{\Pi,\mathcal{A},I}(1^n, x, y, z)$ the random variable, taken over all randomness used by the honest party and the adversary, whose output is given by the outputs of the honest receiver (if $I = S$) and the adversary (which may output an arbitrary function of its view).

- Let the *ideal* experiment be defined as an interaction between a sender S, a receiver R, and a *trusted party* $\mathcal{T}_f$, defined as follows:

  - R sends x to $\mathcal{T}_f$, and S sends y to $\mathcal{T}_f$.
  - $\mathcal{T}_f$, on receiving x and y, computes $out = f(x, y)$ and returns it to R.
  - R, on receiving out, outputs it.

  As with the real experiment, we say that one party $I \in \{S, R\}$ is corrupted in that, as before, their behavior is controlled by an adversary $\mathcal{A}$. We shall denote by $\mathsf{Out}^{\mathcal{T}_f}_{\Pi_f,\mathcal{A},I}(1^n, x, y, z)$ the random variable, once again taken over all randomness used by the honest party and the adversary, whose output is again given by the outputs of the honest receiver (if $I = S$) and the adversary.

Given the above, we can now formally define non-interactive secure computation:

Definition 11 (based on [127, 62, 109, 115, 10])**.** Given a function $T(\cdot)$, a non-interactive two-party protocol $\Pi = (\mathsf{NISC}_1, \mathsf{NISC}_2, \mathsf{NISC}_3)$ between a sender S and a receiver R, and functionality $f(\cdot, \cdot)$ computable by a polynomial-time Turing machine, we say that Π **securely computes f with $T(\cdot)$-time simulation**, or that Π is a **non-interactive secure computation** (NISC) **protocol (with $T(\cdot)$-time simulation)** for computing f, if Π is a

non-interactive two-party computation protocol for computing f and, for any polynomial-time adversary $\mathcal{A}$ corrupting party $I \in \{S, R\}$, there exists a $T(n) \cdot \mathsf{poly}(n)$-time simulator $\mathcal{S}$ such that, for any $T(n) \cdot \mathsf{poly}(n)$-time algorithm $D : \{0,1\}^* \to \{0,1\}$, there exists negligible $\epsilon(\cdot)$ such that for any $n \in \mathbb{N}$ and any inputs $x, y \in \{0,1\}^n, z \in \{0,1\}^*$, we have:

$$\left| \Pr\left[D(\mathsf{Out}_{\Pi,\mathcal{A},I}(1^n, x, y, z)) = 1\right] - \Pr\left[D(\mathsf{Out}^{\mathcal{T}_f}_{\Pi_f,\mathcal{S},I}(1^n, x, y, z)) = 1\right]\right| < \epsilon(n)$$

where the experiments and distributions Out are as defined above.

Furthermore, if Π securely computes f with $T(\cdot)$-time simulation for $T(n) = n^{\log^c(n)}$ for some constant c, we say that Π **securely computes f with quasi-polynomial simulation**.

Succinctness. The defining feature of our construction will be a notion of *succinctness*; specifically, for functionality $f(\cdot,\cdot)$ with Turing machine description M and running time bounded by T_f, we show the existence of a NISC protocol $\Pi = (\mathsf{NISC}_1, \mathsf{NISC}_2, \mathsf{NISC}_3)$ for computing f whose message length (i.e., the combined output length of NISC_1 and NISC_2) and total receiver running time on input 1^n are relatively short and essentially independent of the running time of f. Formally:

Definition 12. We say that a NISC protocol $\Pi = (\mathsf{NISC}_1, \mathsf{NISC}_2, \mathsf{NISC}_3)$ has **communication complexity** $\rho(\cdot)$ if, for any $n \in \mathbb{N}$, $x, y \in \{0,1\}^n$, and $z \in \{0,1\}^*$, the outputs of $\mathsf{NISC}_1(1^n, x)$ and $\mathsf{NISC}_2(1^n, y, z)$ contain at most $\rho(n)$ bits.

We shall define a NISC protocol which, given functionality $f : \{0,1\}^n \times \{0,1\}^n \leftarrow \{0,1\}^{\ell(n)}$ computable by a Turing machine M with running time $T_f(n)$, features communication complexity and receiver running time bounded above by $p(n, \log(T_f(n)), |M|, \ell(n))$ for an *a priori* fixed polynomial p.

There exist *non-succinct* non-interactive secure computation protocols in the standard model based on a notion of "weak oblivious transfer" ([10]), which in turn can be based on subexponential security of the Learning With Errors assumption [26]:

Theorem 3 ([10, 26, 100])**.** Assuming subexponential hardness of the Learning With Errors assumption, for any functionality $f(\cdot,\cdot)$ computable by a polynomial-time Turing machine there exists a (non-succinct) non-interactive secure computation protocol for computing f with quasi-polynomial simulation.

We note that this theorem essentially follows from [10, 26]; however, [10] required as an additional assumption the existence of an *onto* one-way function. In the full version of this result [100], we present a variant which demonstrates how to prove Theorem 3 without this added assumption.

3.2.2 Fully Homomorphic Encryption

Intuitively, a fully homomorphic encryption (FHE) scheme [58] is an encryption scheme with the additional property that computations may, using the public key, be performed on ciphertexts for respective inputs in such a way that the result will be a ciphertext for the correct output. We formalize this as follows:

Definition 13 (based on [58])**.** A **fully homomorphic encryption** (FHE) **scheme** consists of a tuple of algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$, where Gen, Enc are PPT and Eval, Dec are (deterministic) polynomial-time algorithms, such that:

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^n; \rho)$: takes the security parameter n as input and outputs a public key pk and secret key sk.

- $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, m; \rho)$: takes as input a public key pk and a message $m \in \{0,1\}$, and outputs a ciphertext ct. (For multi-bit messages $\overrightarrow{m} \in \{0,1\}^{p(n)}$, we let $\overrightarrow{\mathsf{ct}} \leftarrow \mathsf{Enc}(\mathsf{pk}, \overrightarrow{m})$ be such that $\mathsf{ct}_i = \mathsf{Enc}(\mathsf{pk}, m_i)$.)
- $\mathsf{ct}' = \mathsf{Eval}(\mathsf{pk}, C, \overrightarrow{\mathsf{ct}})$: takes as input a list of ciphertexts $\overrightarrow{\mathsf{ct}}$ and a circuit description C of some function to evaluate and outputs a ciphertext ct'.
- $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$: takes as input a ciphertext ct and outputs a message m'.

We furthermore require that the following properties are satisfied:

1. **Full homomorphism:** There exist sets of boolean circuits $\{\mathcal{C}_n\}_{n\in\mathbb{N}}$, negligible function $\epsilon(n)$, and polynomial $p(\cdot)$ such that $\mathcal{C} = \bigcup_n \mathcal{C}_n$ includes the set of all arithmetic circuits over $\mathsf{GF}(2)$[1], and, for any $n \in \mathbb{N}$, we have that, for all $C \in \mathcal{C}_n$ and $\overrightarrow{m} \in \{0,1\}^{p(n)}$:

$$\Pr[z \neq C(\overrightarrow{m}) : (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^n), \overrightarrow{\mathsf{ct}} \leftarrow \mathsf{Enc}(\mathsf{pk}, \overrightarrow{m}), \\ z \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{Eval}(C, \mathsf{pk}, \overrightarrow{\mathsf{ct}}))] < \epsilon(n),$$

   Furthermore, if this probability is identically zero, we refer to the scheme as having **perfect correctness**.

2. **Compactness:** There exists a polynomial $q(\cdot)$ such that the output length of Eval given (any number of) inputs generated with security parameter n is at most $q(n)$.

Definition 14 (based on [66])**.** We say that an FHE $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ is **secure** if, for all non-uniform PPT D, there exists a negligible $\epsilon(\cdot)$ such that for any $n \in \mathbb{N}$:

$$|\Pr[D(1^n, \mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, 0)) = 1] - \Pr[D(1^n, \mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, 1)) = 1]| < \epsilon(n)$$

over $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^n)$. If this condition holds also with respect to subexponential-size distinguishers D (i.e., algorithms implemented by circuits of size $poly(2^{n^\epsilon})$ for some $\epsilon > 0$), we refer to the scheme as being **subexponentially secure**.

[1] $\mathsf{GF}(2)$ is the set of arithmetic circuits consisting only of $+$ and $\times$ gates over the field $\mathbb{F}_2$.

We have the following consequence for encryptions of $\mathsf{poly}(n)$-bit messages $\overrightarrow{m_0}, \overrightarrow{m_1}$:

Fact 1. If an FHE scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ is secure (resp., subexponentially secure), then, for any polynomial $p(\cdot)$ and for any non-uniform PPT (resp., subexponential-size) $(\mathcal{A}, D)$ where $\mathcal{A}$ outputs messages $\overrightarrow{m_0}, \overrightarrow{m_1} \in \{0,1\}^{p(n)}$ for polynomial $p(\cdot)$, there exists a negligible $\epsilon(\cdot)$ such that for any $n \in \mathbb{N}$:

$$|\Pr[D(1^n, \mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, \overrightarrow{m_0})) = 1] - \Pr[D(1^n, \mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, \overrightarrow{m_1})) = 1]| < \epsilon(n)$$

where

$$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^n), (\overrightarrow{m_0}, \overrightarrow{m_1}) \leftarrow \mathcal{A}(1^n, \mathsf{pk})$$

We can construct an FHE scheme with all of the above properties based on the Learning With Errors (LWE) assumption:

Theorem 4 ([27, 64, 4])**.** Based on computational (resp., subexponential) hardness of the Learning With Errors assumption, there exists a secure (resp., subexponentially secure) fully homomorphic encryption scheme satisfying perfect correctness.

3.2.3 Adaptive Delegation Schemes

A delegation scheme allows for the effective "outsourcing" of computation from one party to another; that is, using delegation, the sender can compute both the correct result of some (possibly expensive) computation on a receiver's input and a (short) proof which can convince the receiver of the correctness of the computation without requiring the receiver to perform the computation themselves. We consider a notion of delegation with the additional property, formalized in [29], that the functionality $f(\cdot)$ whose computation is to be delegated can be decided *adaptively* after the keys pk, sk are computed (i.e., the key-generation algorithm Gen is independent from f). Formally:

Definition 15 (based on [29])**.** An **adaptive delegation scheme** is given by a triple of algorithms $(\mathsf{Gen}, \mathsf{Comp}, \mathsf{Ver})$, where Comp and Ver are (deterministic) polynomial-time algorithms and Gen is PPT, such that:

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^n; \rho)$ takes as input a security parameter n and probabilistically outputs a public key pk and secret key sk.
- $(y, \pi, 1^T) \leftarrow \mathsf{Comp}(\mathsf{pk}, f, \overrightarrow{x})$ takes as input a Turing machine description of the functionality f to be computed, as well as the inputs $\overrightarrow{x}$ to f, and produces a result y which the sender claims to be the result of the computation, a $\mathsf{poly}(n)$-size proof π of its correctness, and the running time T of the computation in unary.
- $\{\mathsf{Accept}, \mathsf{Reject}\} \leftarrow \mathsf{Ver}(\mathsf{sk}, f, \overrightarrow{x}, y, \pi, T)$ takes as input the functionality f to be computed, inputs $\overrightarrow{x}$, result y, proof π, and running time T, and returns Accept or Reject depending on whether π is a valid proof of $f(\overrightarrow{x}) = y$.

Furthermore, we require the following properties:

1. **Completeness:** There exists a negligible function $\epsilon(\cdot)$ such that, for any $n \in \mathbb{N}$, any f computable by a Turing machine that runs in time at most 2^n, and any $\overrightarrow{x}$ in the domain of f:

   $$\Pr\left[(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^n); (\pi, y, 1^T) = \mathsf{Comp}(\mathsf{pk}, f, \overrightarrow{x}) : \mathsf{Ver}(\mathsf{sk}, f, \overrightarrow{x}, \pi, y, T) = \mathsf{Reject}\right] < \epsilon(n)$$

   In addition, if the above probability is identically zero, we say that the adaptive delegation scheme satisfies **perfect completeness**.

2. **Correctness:** For any $n \in \mathbb{N}$, any f computable by a Turing machine that runs in time at most 2^n, and any $\overrightarrow{x}$ in the domain of f:

   $$\Pr\left[(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^n) : \mathsf{Comp}(\mathsf{pk}, f, \overrightarrow{x}) = (f(\overrightarrow{x}), \cdot, \cdot)\right] = 1$$

3. **Soundness:** For any non-uniform PPT adversary $\mathcal{A}$, there exists a negligible function $\epsilon(\cdot)$ such that, for any $n \in \mathbb{N}$:

$$\Pr\left[(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Gen}(1^n), (f, \overrightarrow{x}, y_1, y_2, \pi_1, \pi_2, 1^{T_1}, 1^{T_2}) \leftarrow \mathcal{A}(1^n, \mathsf{pk}) : \right.$$
$$T < 2^n \wedge \mathsf{Ver}(\mathsf{sk}, f, \overrightarrow{x}, y_1, \pi_1, T_1) = \mathsf{Accept}$$
$$\left. \wedge \mathsf{Ver}(\mathsf{sk}, f, \overrightarrow{x}, y_2, \pi_2, T_2) = \mathsf{Accept} \wedge y_1 \neq y_2\right] < \epsilon(n)$$

Furthermore, if this condition holds with respect to subexponential-size adversaries, we say that the scheme is **subexponentially sound**.

A construction of an adaptive delegation scheme with perfect completeness can be found in the work of Brakerski et al. [29], and is based on a secure private information retrieval (PIR) scheme, which in turn can be constructed based on a leveled FHE scheme (including the one presented in Theorem 4). Hence:

Theorem 5 ([27, 64, 29, 4])**.** Based on computational (resp., subexponential) hardness of the Learning With Errors assumption, there exists a sound (resp., subexponentially sound) adaptive delegation scheme satisfying perfect completeness.

3.3 Succinct Non-Interactive Secure Computation

In this section, we present our first feasibility result, which constructs a succinct non-interactive secure computation protocol. We lead with a technical overview before presenting the formal theorem and proof.

3.3.1 Technical Overview

At a high level, our approach begins with the semi-honestly secure approach of using FHE (which we detailed in the introduction) and attempts to compile it to become secure with respect to malicious attackers. Instead of using ZK-SNARKs (which rely on non-standard assumptions and trusted setup), we will instead use an adaptive delegation scheme and a non-succinct NISC. For our approach to work, we will strongly rely on *perfect correctness/completeness* properties of both the FHE and the delegation scheme; as far as we know, perfect correctness of these types of primitives has not previously been used to enable applications (where the goal itself isn't perfect correctness).[2]. Despite this, though, recent constructions (or slight variants) of both FHE and delegation protocols fortunately do provide these guarantees.

Adaptive Delegation: A Starting Point. To explain the approach, we shall start from a (flawed) candidate which simply combines an FHE scheme and an adaptive delegation scheme. In an adaptive delegation scheme (as given in [29]), a verifier generates a public/secret key-pair $(\mathsf{pk}, \mathsf{sk})$ and sends pk to the prover. The prover next picks some statement $\tilde{x}$ and function g, computes the output $\tilde{y} = g(\tilde{x})$, and produces a "short" proof π of the validity of the statement that $\tilde{y} = g(\tilde{x})$. The prover finally sends $(\tilde{x}, g, \tilde{y}, \pi)$ to the verifier, who can use its secret key sk to check the validity of the proof. We will rely on an adaptive delegation scheme satisfying *perfect completeness*—that is, for all public keys in the range of the key generation algorithm, the prover can convince the verifier with probability 1.

The candidate SNISC leverages delegation to "outsource" the computation of the homomorphic evaluation to the sender: specifically, the receiver first generates a public/secret

[2]The only work we are aware that uses perfect correctness of a FHE is a very recent work [4] which uses perfectly correct FHE as a tool to get perfectly correct iO.

key-pair $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{sk}_{\mathsf{FHE}})$ for the FHE, encrypts its input x using the FHE (obtaining a ciphertext ct_x), generates a public/secret key pair $(\mathsf{pk}_{\mathsf{Del}}, \mathsf{sk}_{\mathsf{Del}})$ for the delegation scheme, and finally sends $(\mathsf{ct}_x, \mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}})$ to the sender. The sender in turn encrypts its input y, obtaining a ciphertext ct_y; next, it lets g be the function for homomorphically evaluating f on two ciphertexts, computes $g(\mathsf{ct}_x, \mathsf{ct}_y)$ (i.e., homomorphically evaluates f on ct_x and ct_y) to obtain a ciphertext $\mathsf{ct}_{\mathsf{out}}$, and computes a delegation proof π (with respect to $\mathsf{pk}_{\mathsf{Del}}$) of the validity of the computation of g. Finally, the sender sends $(\mathsf{ct}_y, \mathsf{ct}_{\mathsf{out}}, \pi)$ to the receiver, who verifies the proof and, if the proof is accepting, decrypts $\mathsf{ct}_{\mathsf{out}}$ and outputs it.

Intuitively, this approach hides the input x of the receiver, but clearly fails to hide the input y of the sender, as the receiver can simply decrypt ct_y to obtain y. So, rather than providing ct_y and π in the clear (as even just the proof π could leak things about ct_y), we instead use the (non-succinct) NISC to run the verification procedure of the delegation scheme. That is, we can add to the protocol a NISC instance where the receiver inputs $\mathsf{sk}_{\mathsf{Del}}$, the sender inputs $\mathsf{ct}_x, \mathsf{ct}_y, \mathsf{ct}_{\mathsf{out}}, \pi$, and the functionality runs the verification algorithm for the delegation scheme, outputting either $\perp$ if verification fails or, otherwise, $\mathsf{ct}_{\mathsf{out}}$ (which can be decrypted by the receiver).

Input Independence: Leveraging Perfect Correctness of FHE. The above approach intuitively hides the inputs of both players, and also ensures that the function is computed correctly. But there are many problems with it. For instance, while it guarantees that the sender does not learn the receiver's input x, it does *not* guarantee "input independence", or that the sender's input does not depend on the receiver's somehow: for instance, the sender can easily maul ct_x into, say, an encryption ct_y of $x + 1$ and use that as its input. On a more technical level, simulation-based security requires the simulator to be able to extract the inputs of malicious players, but it is not clear how this can be done here—in fact, a

simulator *cannot* extract the sender's input y due to the above malleability attack.

To overcome this issue, we again leverage the non-succinct NISC to enable extractability: we add x and the randomness, r_x, needed to generate ct_x as an input from the receiver, and we add ct_x (i.e., the ciphertext obtained from the receiver), y, and the randomness needed to generate ct_y as input from the sender. The functionality additionally checks that the ciphertexts $\mathsf{ct}_x, \mathsf{ct}_y$ respectively are valid encryptions of the inputs x, y using the given randomness. (It is actually essential that the *sender* includes the ciphertext ct_x from the receiver as part of its input, as opposed to having the receiver input it, as otherwise we could not guarantee that the receiver is sending the same ciphertext to the sender as it is inputting to the NISC). If we have perfect correctness for the underlying FHE scheme with respect to the public-keys selected by the receiver, this approach guarantees that we can correctly extract the inputs of the players. The reason that we need perfect correctness is that the NISC only guarantees that the ciphertexts have been honestly generated using *some* randomness, but we have no guarantees that the randomness is honestly generated. Perfect correctness ensures that all randomness is "good" and will result in a "well-formed" ciphertext on which homomorphic computation, and subsequently decryption, will always lead to the correct output.

Dealing with a Malicious Receiver: Interactive Witness Encryption and Perfectly Correct Delegation. While the above protocol suffices to deal with a malicious sender (although, as we shall discuss later on, even this is not trivial due to the potential for "spooky interactions" [49]), it still does not allow us to deal with a malicious receiver. The problem is that the receiver could send invalid public keys, either for the FHE or for the delegation scheme. For instance, if the public key for the FHE is invalid, perfect correctness may no longer hold, and we may not be able to extract a correct input for the receiver. Likewise,

if the public key for the delegation scheme is invalid, we will not be able to determine whether the verification algorithm of the delegation scheme will be accepting, and thus cannot carry out a simulation. Typically, dealing with a malicious receiver would require adding a zero-knowledge proof of well-formedness of its messages; however, given that the receiver is sending the first message, this seems problematic since, even with SPS-security, one-message ZK is impossible (with respect to non-uniform attackers [109, 14]).

To explain our solution to this problem, let us first assume that we have access to a *witness encryption scheme* [55]. Recall that a witness encryption scheme enables encrypting a message m with a statement $\tilde{x}$ so that anyone having a witness w to $\tilde{x}$ can decrypt the message; if the statement is false, however, the encryption scheme conceals the message m. If we had access to such a witness encryption scheme, we could have the functionality in the NISC compute a witness encryption of $\mathsf{ct}_{\mathsf{out}}$ with the statement being that the public keys have been correctly generated. This method ensures that the receiver does not get any meaningful output unless it actually generated the public keys correctly. Of course, it may still use "bad" randomness—we can only verify that the public keys are in the range of the key generating function. But, if the delegation scheme *also* satisfies a "perfect correctness" property (specifically, both correctness of the computation and *perfect completeness* of the generated proof), this enables us to simulate the verification of the delegation scheme (as once again, in this case, perfect correctness guarantees that there is no "bad" randomness).

We still have an issue: perfect correctness of the FHE will ensure that the decryption of the output is correct, but we also need to ensure that we can simulate the ciphertext output by the NISC. While this can be handled using an FHE satisfying an appropriate rerandomizability/simulatability property (also with respect to maliciously selected ciphertext), doing so introduces additional complications. Furthermore, while we motivated the above modification using witness encryption, currently known constructions of witness encryption

rely on non-standard, and less understood, hardness assumptions; as such, we would like to altogether avoid using it as an underlying primitive.

So, to circumvent the use of witness encryption—while at the same time ensuring that the output of the NISC is simulatable—we realize that in our context, it in fact suffices to use a *two-round version of witness encryption*, where the receiver of the encryption chooses the statement and can first send a message corresponding to the statement. And such a non-interactive version of witness encryption can be readily implemented using a NISC! As we are already running an instance of a NISC, we can simply have the NISC also implement this interactive witness encryption. More precisely, we now additionally require the receiver to provide its witness—i.e., the randomness for the key generation algorithms—as an input to the NISC, while the sender additionally provides the public keys $\mathsf{pk}_{\mathsf{FHE}}$ and $\mathsf{pk}_{\mathsf{Del}}$ which it receives. The functionality will now only release the output $\mathsf{ct}_{\mathsf{out}}$ if it verifies that the keys input by the sender are correctly generated from the respective randomness input by the receiver. Better still, since the randomness used to generate the public/secret key-pair is now an input to the functionality, the functionality can *also* recover the secret key for the FHE, and next also decrypt $\mathsf{ct}_{\mathsf{out}}$ and simply output plain text corresponding to $\mathsf{ct}_{\mathsf{out}}$. This prevents the need for rerandomizing $\mathsf{ct}_{\mathsf{out}}$, since it is now internal to the NISC instance (and is no longer output). With all of the above modifications, we can now prove that the protocol satisfies SPS security.

The Final Protocol. For clarity, let us summarize the final protocol.

- The Receiver generates $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{sk}_{\mathsf{FHE}})$ and $(\mathsf{pk}_{\mathsf{Del}}, \mathsf{sk}_{\mathsf{Del}})$ using randomness r_{FHE} and r_{Del} (respectively) and generates an encryption ct_x of its input x using randomness r_x. It then sends $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x)$ and the first message msg_1 of a NISC using the input

$x' = (x, r_{\mathsf{FHE}}, r_{\mathsf{Del}}, r_x)$ (for a functionality to be specified shortly).

- The Sender, upon receiving $\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{msg}_1$ generates an encryption ct_y of its input y using randomness r_y, applies the homomorphic evaluation of f to ct_x and ct_y to obtain a ciphertext $\mathsf{ct}_{\mathsf{out}} = g(\mathsf{ct}_x, \mathsf{ct}_y)$, generates a proof π using the delegation scheme (w.r.t. $\mathsf{pk}_{\mathsf{Del}}$) of the correctness of the computation that $\mathsf{ct}_{\mathsf{out}} = g(\mathsf{ct}_x, \mathsf{ct}_y)$, and finally sends the second message msg_2 of the NISC using the input $y' = (y, \mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x, \mathsf{ct}_y, \mathsf{ct}_{\mathsf{out}}, \pi, r_y)$.
- Finally, the receiver, upon getting msg_2, computes the output z of the NISC protocol and outputs it.
- The functionality computed by the NISC on input $x' = (x, r_{\mathsf{FHE}}, r_{\mathsf{Del}}, r_x)$ and $y' = (y, \mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x, \mathsf{ct}_y, \mathsf{ct}_{\mathsf{out}}, \pi, r_y)$ does the following: it checks that:

  1. the public keys $\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}$ were respectively generated using randomness $r_{\mathsf{FHE}}, r_{\mathsf{Del}}$;
  2. the ciphertexts $\mathsf{ct}_x, \mathsf{ct}_y$ are respectively encryptions of x, y using randomness r_x, r_y; and,
  3. π is a valid proof of $\mathsf{ct}_{\mathsf{out}} = g(\mathsf{ct}_x, \mathsf{ct}_y)$ w.r.t. $(\mathsf{pk}_{\mathsf{Del}}, \mathsf{sk}_{\mathsf{Del}})$ (as generated from r_{Del}).

  If the checks pass, it decrypts $\mathsf{ct}_{\mathsf{out}}$ (by first generating $\mathsf{sk}_{\mathsf{FHE}}$ from r_{FHE}), obtaining the plaintext z, and finally outputs z. (If any of the checks fail, it instead outputs $\perp$.)

A summary of the message flow can be found in Figure 3.1.

A Subtlety in the Security Proof One subtle point that arises in the proof of security is that, to simulate a malicious sender, we need to simulate the ciphertext ct_x without knowledge of x. But the functionality of the underlying NISC takes as input the randomness

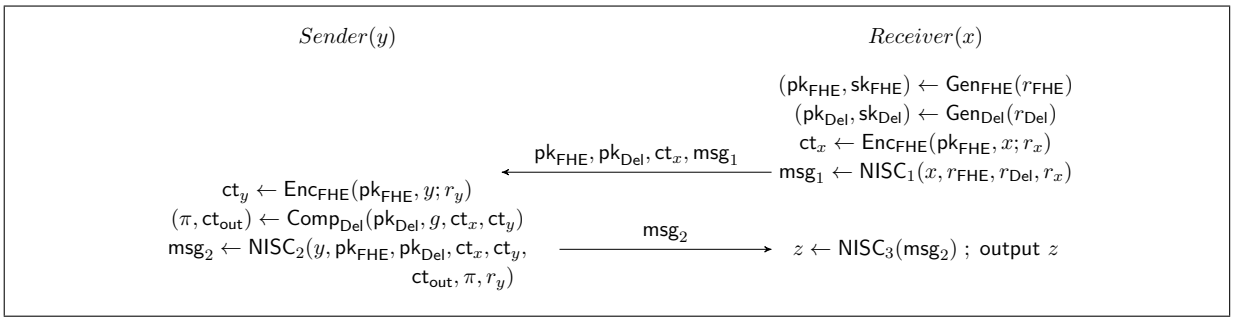

Figure 3.1: The final SNISC protocol. $(\mathsf{NISC}_1, \mathsf{NISC}_2, \mathsf{NISC}_3)$ denotes the underlying (non-succinct) NISC protocol and and the functionality g denotes the homomorphic evaluation $g(c_1, c_2) = \mathsf{Eval}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, f, c_1, c_2)$.

used for both the key generation of $\mathsf{pk}_{\mathsf{FHE}}$ and for encrypting ct_x, and thus the functionality implicitly knows how to decrypt ct_x. A similar issue has arisen in the related context of constructing delegation schemes from FHE and related primitives (see [49]), where it was shown that so-called "spooky interactions" can arise, where a malicious sender (even though it does not how to decrypt the ciphertext) can in fact use this dependence to make the receiver output values that correlate in undesirable ways with the input x (in particular, in ways that would not have been possible if using an "idealized" FHE). Fortunately, in our context, we are able to overcome this issue by using the perfect correctness of the FHE scheme and soundness of our underlying delegation scheme to perform a carefully designed hybrid argument.

A bit more precisely, the key point is that when simulating a malicious sender in communication with an *honest* receiver, the receiver's public key and ciphertext ct_x will always be correctly generated (as such, we do not have to perform the checks involving the receiver to simulate the underlying NISC's output); furthermore, by soundness of delegation and the perfect correctness of the FHE, the decryption of $\mathsf{ct}_{\mathsf{out}}$ must equal $f(x, y)$ (with overwhelming probability) if π is accepting, so we can use this fact to show that decrypting $\mathsf{ct}_{\mathsf{out}}$ is actually *also* unnecessary. As such, we do not need to use either r_{FHE} or r_x to emulate the experiment for a malicious sender, and we can create (and prove security in) a hybrid functionality for

the underlying NISC which is independent of this randomness (and only depends on $\mathsf{pk}_{\mathsf{FHE}}$).

3.3.2 Protocol and Proof

We next state and prove our formal theorem and construction:

Theorem 6. Assuming subexponential hardness of the Learning With Errors assumption, there exists polynomial $p(\cdot,\cdot,\cdot,\cdot)$ such that, for any polynomials $T_f(\cdot)$ and $\ell(\cdot)$ and any Turing machine M with running time bounded by $T_f(\cdot)$ computing functionality $f(\cdot,\cdot) : \{0,1\}^n \times \{0,1\}^n \leftarrow \{0,1\}^{\ell(n)}$, there exists a non-interactive secure computation protocol for computing f with quasi-polynomial simulation which is additionally *succinct* in that both its communication complexity and the running time of the honest receiver are at most $p(n, \log(T_f(n)), |M|, \ell(n))$.

We propose the protocol Π given in Figure 3.2 for secure non-interactive secure computation of a function $f(x,y)$ given a receiver input x and sender input y, where Π shall use the following primitives:

- Let $\pi = (\mathsf{NISC}_1, \mathsf{NISC}_2, \mathsf{NISC}_3)$ be a non-succinct NISC protocol with $T(n)$-time simulation for $T(n) = n^{\log^c(n)}$ (i.e., quasi-polynomial simulation), whose functionality h will be determined in the first round of the protocol. (The existence of such a primitive is guaranteed by Theorem 3 under subexponential LWE.)
- Let $(\mathsf{Gen}_{\mathsf{FHE}}, \mathsf{Enc}_{\mathsf{FHE}}, \mathsf{Dec}_{\mathsf{FHE}}, \mathsf{Eval}_{\mathsf{FHE}})$ be a fully homomorphic encryption scheme satisfying perfect correctness, compactness, and subexponential security (in particular, with respect to $T(n) \cdot \mathsf{poly}(n)$-time adversaries). (The existence of such a primitive is guaranteed by Theorem 4 under subexponential LWE.)

Input: The receiver R and the sender S are given input $x, y \in \{0,1\}^n$, respectively, and both parties have common input 1^n.
Output: R receives $f(x, y)$.

Round 1: R proceeds as follows:

1. Generate random coins $r_{\mathsf{FHE}} \leftarrow \{0,1\}^*$ and compute $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{sk}_{\mathsf{FHE}}) = \mathsf{Gen}_{\mathsf{FHE}}(1^n; r_{\mathsf{FHE}})$.
2. Let T_g denote the running time of the functionality $g(c_1, c_2) = \mathsf{Eval}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, f, c_1, c_2)$, and let $\lambda = \max(n, \log(T_g))$. Generate random coins $r_{\mathsf{Del}} \leftarrow \{0,1\}^*$ and compute $(\mathsf{pk}_{\mathsf{Del}}, \mathsf{sk}_{\mathsf{Del}}) = \mathsf{Gen}_{\mathsf{Del}}(1^\lambda; r_{\mathsf{Del}})$.
3. Generate random coins $r_{\mathsf{Enc}(x)} \leftarrow \{0,1\}^*$ and compute $\mathsf{ct}_x = \mathsf{Enc}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, x; r_{\mathsf{Enc}(x)})$.
4. Generate message $\mathsf{msg}_1 \leftarrow \mathsf{NISC}_1(x, r_{\mathsf{FHE}}, r_{\mathsf{Del}}, r_{\mathsf{Enc}(x)})$ to compute the functionality h described in Figure 3.3.
5. Send $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x, \mathsf{msg}_1)$ to S.

Round 2: S proceeds as follows:

1. Generate random coins $r_{\mathsf{Enc}(y)} \leftarrow \{0,1\}^*$ and compute $\mathsf{ct}_y = \mathsf{Enc}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, y; r_{\mathsf{Enc}(y)})$.
2. Compute $(\mathsf{ct}_{\mathsf{out}}, \pi_{\mathsf{Del}}, 1^T) = \mathsf{Comp}_{\mathsf{Del}}(\mathsf{pk}_{\mathsf{Del}}, g, \mathsf{ct}_x, \mathsf{ct}_y)$ for the functionality $g(c_1, c_2) = \mathsf{Eval}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, f, c_1, c_2)$.
3. Generate message $\mathsf{msg}_2 \leftarrow \mathsf{NISC}_2(y, \mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x, \mathsf{ct}_y, \mathsf{ct}_{\mathsf{out}}, \pi_{\mathsf{Del}}, r_{\mathsf{Enc}(y)}, T)$ to compute the functionality h described in Figure 3.3.
4. Send msg_2 to R.

Output phase: R proceeds as follows:

1. Compute $\mathsf{out} = \mathsf{NISC}_3(\mathsf{msg}_2)$ and return the result.

Figure 3.2: Protocol Π for succinct non-interactive secure computation.

- Let $(\mathsf{Gen}_{\mathsf{Del}}, \mathsf{Comp}_{\mathsf{Del}}, \mathsf{Ver}_{\mathsf{Del}})$ be an adaptive delegation scheme with perfect completeness, correctness, and subexponential soundness (in particular, with respect to $T(n) \cdot \mathsf{poly}(n)$-time adversaries). (The existence of such a primitive is guaranteed by Theorem 5 under subexponential LWE.)

Input: The receiver R has input $(x, r_{\mathsf{FHE}}, r_{\mathsf{Del}}, r_{\mathsf{Enc}(x)})$, and the sender S has input $(y, \mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x, \mathsf{ct}_y, \mathsf{ct}_{\mathsf{out}}, \pi_{\mathsf{Del}}, r_{\mathsf{Enc}(y)}, T)$
Output: Either a message out or the special symbol $\perp$.

Functionality:

1. Verify that all of the following checks hold. If any fail, return $\perp$.
   (a) $(\mathsf{pk}_{\mathsf{FHE}}, \cdot) = \mathsf{Gen}_{\mathsf{FHE}}(1^n; r_{\mathsf{FHE}})$
   (b) $(\mathsf{pk}_{\mathsf{Del}}, \cdot) = \mathsf{Gen}_{\mathsf{Del}}(1^\lambda; r_{\mathsf{Del}})$
   (c) $\mathsf{ct}_x = \mathsf{Enc}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, x; r_{\mathsf{Enc}(x)})$
   (d) $\mathsf{ct}_y = \mathsf{Enc}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, y; r_{\mathsf{Enc}(y)})$
2. Compute $(\cdot, \mathsf{sk}_{\mathsf{FHE}}) = \mathsf{Gen}_{\mathsf{FHE}}(1^n; r_{\mathsf{FHE}})$ and $(\cdot, \mathsf{sk}_{\mathsf{Del}}) = \mathsf{Gen}_{\mathsf{Del}}(1^\lambda; r_{\mathsf{Del}})$.
3. If $\mathsf{Ver}_{\mathsf{Del}}(\mathsf{sk}_{\mathsf{Del}}, g, \mathsf{ct}_x, \mathsf{ct}_y, \mathsf{ct}_{\mathsf{out}}, \pi_{\mathsf{Del}}, T) = \mathsf{Reject}$ for the functionality $g(c_1, c_2) = \mathsf{Eval}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, f, c_1, c_2)$, then return $\perp$.
4. Compute $\mathsf{out} = \mathsf{Dec}_{\mathsf{FHE}}(\mathsf{sk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{out}})$ and return the result.

Figure 3.3: Functionality h used for the underlying 2PC protocol π.

Overview. After first proving the succinctness and correctness of the protocol, we turn to proving its security. We do this in two steps. In the first step, we consider a "hybrid" model in which the underlying NISC protocol is replaced by an "ideal" third party $\mathcal{T}_h$. If the underlying protocol were universally composable [34], this step would be trivial; unfortunately, it is not, so we need to take care to formally reduce this transformation to the simulation-based security of the underlying protocol. Crucially, this will rely on the fact that we restrict our attention to two-round protocols.

Next, in the second step, we can create and prove the respective simulators for a corrupted sender and corrupted receiver in the $\mathcal{T}_h$-hybrid model. The corrupted receiver case follows in a fairly straightforward way, relying on the perfect correctness and completeness of the delegation and FHE schemes. The corrupted sender case, however, has some interesting subtleties in the reduction, and in fact will require another hybrid with a slightly different third party $\mathcal{T}_{h'}$ to complete; we discuss these subtleties in more detail when they arise during

the proof.

We begin the formal proof by proving that the protocol Π is *succinct*:

Lemma 9. There exists polynomial $p(\cdot,\cdot,\cdot,\cdot)$ such that, for any polynomials $T_f(\cdot)$ and $\ell(\cdot)$ and any Turing machine M with running time bounded by $T_f(\cdot)$ computing functionality $f(\cdot,\cdot) : \{0,1\}^n \times \{0,1\}^n \leftarrow \{0,1\}^{\ell(n)}$, the respective non-interactive secure computation protocol Π has communication complexity and honest receiver running time bounded above by $p(n, \log(T_f(n)), |M|, \ell(n))$.

Proof. To lead, we point out that, while $T_f(n)$ and $\ell(n)$ are given to be $\mathsf{poly}(n)$, we deliberately quantify them after $p(n)$ in order to treat them separately in the analysis below, as we specifically wish to show that the communication complexity of Π is at most polylogarithmic in the running time $T_f(n)$ for any possible polynomial-time Turing machine computable functionality $f(\cdot,\cdot)$. Furthermore, we assume without loss of generality that the input lengths provided to sub-algorithms are correct, as in the event that an adversary provides incorrectly sized inputs the algorithms may simply abort.

We begin by analyzing the communication complexity, as succinctness of the receiver's running time will follow immediately from this analysis. Aside from messages msg_1 and msg_2 for the underlying NISC π, the only communication consists of the public keys $\mathsf{pk}_{\mathsf{FHE}}$ and $\mathsf{pk}_{\mathsf{Del}}$ and the ciphertext ct_x. $\mathsf{pk}_{\mathsf{FHE}}$ has length $\mathsf{poly}(n)$ since $\mathsf{Gen}_{\mathsf{FHE}}$ is a polynomial-time algorithm running on input 1^n, and the ciphertext ct_x (which consists of a ciphertext for each bit in $x \in \{0,1\}^n$) also has length $\mathsf{poly}(n)$ since $\mathsf{Enc}_{\mathsf{FHE}}$ is polynomial-time and is run on inputs of length $\mathsf{poly}(n)$. $\mathsf{pk}_{\mathsf{Del}}$ will have length $\mathsf{poly}(n, \log(T_f))$; specifically, its length is given to be $\mathsf{poly}(\lambda) = \mathsf{poly}(n, \log(T_g))$, where T_g is the running time of the functionality $g(c_1, c_2) = \mathsf{Eval}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, f, c_1, c_2)$ with inputs generated from common input 1^n. However, since $\mathsf{pk}_{\mathsf{FHE}}$ has $\mathsf{poly}(n)$ length, the input ciphertexts both have $\mathsf{poly}(n)$ length by the efficiency of $\mathsf{Enc}_{\mathsf{FHE}}$,

and f in this case is given as a *circuit* description, which will have size $\mathsf{poly}(T_f(n))$, we have by the efficiency of $\mathsf{Eval}_{\mathsf{FHE}}$ that $T_g = \mathsf{poly}(n, T_f(n))$, implying $\mathsf{poly}(\lambda) = \mathsf{poly}(n, \log(T_f(n)))$.

So it suffices now to bound the length of the NISC messages msg_1 and msg_2. Specifically, even for a *non-succinct* NISC protocol π, the honest sender and receiver must be efficient, and so the message length is still at most polynomial in the input length and running time of the functionality h. We argue that these are $\mathsf{poly}(n, \log(T_f(n)), |M|, \ell(n))$ to complete the proof of the claim:

- The input length to π is given as the size of the inputs $(x, r_{\mathsf{FHE}}, r_{\mathsf{Del}}, r_{\mathsf{Enc}(x)})$ from the receiver and $(y, \mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x, \mathsf{ct}_y, \mathsf{ct}_{\mathsf{out}}, \pi_{\mathsf{Del}}, r_{\mathsf{Enc}(y)}, T)$ from the sender. x and y have length n by assumption. $\mathsf{pk}_{\mathsf{FHE}}$, ct_x, and ct_y have length $\mathsf{poly}(n)$ as argued above, and $\mathsf{ct}_{\mathsf{out}}$ (which consists of a ciphertext output from $\mathsf{Eval}_{\mathsf{FHE}}$ for each bit of $f(x, y) \in \{0,1\}^{\ell(n)}$) has length $\mathsf{poly}(n, \ell(n))$ by the compactness of the underlying FHE scheme. $\mathsf{pk}_{\mathsf{Del}}$ has length $\mathsf{poly}(n, \log(T_f(n)))$ as argued above, and π_{Del} also has length $\mathsf{poly}(\lambda) = \mathsf{poly}(n, \log(T_f(n)))$; T will have size $\lambda = \mathsf{poly}(n, \log(T_f(n)))$ as $T \leq 2^\lambda$ is required by the properties of the delegation scheme. Lastly, the randomness $r_{\mathsf{FHE}}, r_{\mathsf{Del}}, r_{\mathsf{Enc}(x)}, r_{\mathsf{Enc}(y)}$ cannot have length greater than the running times of the respective algorithms $\mathsf{Gen}_{\mathsf{FHE}}, \mathsf{Gen}_{\mathsf{Del}}, \mathsf{Enc}_{\mathsf{FHE}}$, all of which we have already noted are at most $\mathsf{poly}(n, \log(T_f(n)))$.
- To bound the running time of the functionality h, notice that it consists of the following:
  - $\mathsf{Gen}_{\mathsf{FHE}}$ (run twice), $\mathsf{Enc}_{\mathsf{FHE}}$ (run $2n$ times, once for each bit of x and y), $\mathsf{Eval}_{\mathsf{FHE}}$ (run $\ell(n)$ times, once for each bit of out), all of which are efficient algorithms run on inputs of at most length $\mathsf{poly}(n)$ (and hence have running time $\mathsf{poly}(n)$);
  - $\mathsf{Dec}_{\mathsf{FHE}}$ (run $\ell(n)$ times), which has inputs $\mathsf{sk}_{\mathsf{FHE}}$ with size $\mathsf{poly}(n)$ and $\mathsf{ct}_{\mathsf{out}}$ with size $\mathsf{poly}(n, \ell(n))$, and hence has running time $\mathsf{poly}(n, \ell(n))$;

- $\mathsf{Gen}_{\mathsf{Del}}$ (run twice), which runs in time $\mathsf{poly}(\lambda) = \mathsf{poly}(n, \log(T_f(n)))$;
- $\mathsf{Ver}_{\mathsf{Del}}$ (run once), which, given inputs $\mathsf{sk}_{\mathsf{Del}}, \pi_{\mathsf{Del}}$ of size $\mathsf{poly}(\lambda) = \mathsf{poly}(n, \log(T_f(n)))$, $\mathsf{ct}_x, \mathsf{ct}_y$ of size $\mathsf{poly}(n)$, $\mathsf{ct}_{\mathsf{out}}$ of size $\mathsf{poly}(n, \ell(n))$, g (the description of $g(c_1, c_2) = \mathsf{Eval}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, f, c_1, c_2)$, where we here interpret f as the Turing machine M) of size $\mathsf{poly}(|M|)$, and $T \leq 2^\lambda$ of size at most $\lambda = \mathsf{poly}(n, \log(T_f(n)))$, has running time which is at most $\mathsf{poly}(n, \log(T_f(n)), |M|, \ell(n))$;

and a $\mathsf{poly}(n)$ number of comparisons between input values and function outputs which have already been established to have at most $\mathsf{poly}(n, \log(T_f(n)))$ length.

The above shows that the communication complexity of Π is succinct. Furthermore, as the honest receiver runs only $\mathsf{Gen}_{\mathsf{FHE}}$, $\mathsf{Gen}_{\mathsf{Del}}$, $\mathsf{Enc}_{\mathsf{FHE}}$, and the (efficient) receiver protocol for the underlying NISC on the aforementioned inputs, and as we have already established that all of these algorithms have running time $\mathsf{poly}(n, \log(T_f(n)), |M|, \ell(n))$, the receiver will inherit the same running time bound. □

Towards proving security for Π, let $\mathsf{Out}_{\Pi,\mathcal{A},I}(1^n, x, y, z)$ denote the random variable, taken over all randomness used by the honest party and the adversary, of the outputs of the honest receiver (if $I = S$) and the adversary in the execution of protocol Π given adversary $\mathcal{A}$ controlling corrupted party $I \in \{S, R\}$, receiver input x, sender input y, and adversary auxiliary input z. Let $\mathsf{Exec}_{\Pi,\mathcal{A},I}(1^n, x, y, z)$ denote the respective experiment.

Let us also define the "ideal" execution by letting $\mathcal{T}_f$ denote the ideal functionality corresponding to the computation target $f(x, y)$ and letting Π_f be the "ideal" version of the protocol where R sends x to $\mathcal{T}_f$, S sends y to $\mathcal{T}_f$, and then R finally outputs the result out output by $\mathcal{T}_f$. We want to show the following theorem:

Theorem 7. Assume, given functionality $f(\cdot,\cdot)$, the respective protocol Π described in Figure 3.2 and the assumptions required in Theorem 6, and let $T(\cdot)$ be such that the underlying NISC π is secure with $T(\cdot)$-time simulation. For any efficient adversary $\mathcal{A}$ corrupting party $I \in \{S, R\}$, there exists a $T(n) \cdot \mathsf{poly}(n)$-time simulator $\mathcal{S}$ such that, for any non-uniform polynomial-time distinguisher D, there exists a negligible function $\epsilon(\cdot)$ such that, for all $n \in \mathbb{N}$, $x, y \in \{0,1\}^n$, and auxiliary input z, D distinguishes the distributions $\mathsf{Out}_{\Pi,\mathcal{A},I}(1^n, x, y, z)$ and $\mathsf{Out}^{\mathcal{T}_f}_{\Pi_f,\mathcal{S},I}(1^n, x, y, z)$ with at most probability $\epsilon(n)$.

Notice that correctness of Π holds trivially from the perfect correctness of the underlying FHE, the correctness and perfect completeness of the underlying adaptive delegation scheme, and the correctness of the underlying NISC protocol π; hence, Theorem 7, which proves security, and Lemma 9, which proves succinctness, will in conjunction directly imply Theorem 6 (where quasi-polynomial simulation results from our use of an underlying NISC protocol with quasi-polynomial simulation, as given in Theorem 3). The remainder of the section, then, is devoted to proving Theorem 7.

3.3.3 Comparing Real and Hybrid Executions

We begin by defining a "trusted third party" $\mathcal{T}_h$ which executes the ideal functionality for h—that is, given the corresponding sender and receiver inputs, $\mathcal{T}_h$ outputs the correct value of h computed on those inputs. Our first task is to show, then, that the "real" experiment's outputs $\mathsf{Out}_{\Pi,\mathcal{A},I}(1^n, x, y, z)$ cannot be distinguished from those of a "hybrid" experiment, which we shall denote by $\mathsf{Out}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',I}(1^n, x, y, z)$.

Formally, we let Π_h denote a protocol which is identical to Π with the exception that, in rounds 1 and 2, rather than generating msg_1 and msg_2, R and S instead send the respective

inputs to $\mathcal{T}_h$, and, in the output phase, R receives and returns the output from $\mathcal{T}_h$ rather than unpacking msg_2. We then prove the following lemma:

Lemma 10. For any efficient adversary $\mathcal{A}$ corrupting party $I \in \{S, R\}$, there exists a $T(n)\cdot\mathsf{poly}(n)$-time adversary $\mathcal{A}'$ such that, for any non-uniform polynomial-time distinguisher D, there exists a negligible function $\epsilon(\cdot)$ such that, for all $n \in \mathbb{N}$, $x, y \in \{0,1\}^n$, and auxiliary input z, D distinguishes the distributions $\mathsf{Out}_{\Pi,\mathcal{A},I}(1^n, x, y, z)$ and $\mathsf{Out}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',I}(1^n, x, y, z)$ with at most probability $\epsilon(n)$.

Proof. We separate into two cases, based on whether $I = R$ (the receiver is corrupted) or $I = S$ (the sender is corrupted).

Corrupted Receiver. In this case we begin by, given some adversary $\mathcal{A}$ against the real experiment $\mathsf{Exec}_{\Pi,\mathcal{A},R}(1^n, x, y, z)$, defining an adversary $\mathcal{A}_h$ against the underlying 2PC protocol π. Without loss of generality, let $\mathcal{A}$ be a deterministic algorithm which uses the auxiliary input z as its source of randomness for r_{FHE}, r_{Del}, and $r_{\mathsf{Enc}(x)}$. $\mathcal{A}_h(1^n, z)$ will behave identically to $\mathcal{A}(1^n, z)$ (acting as the corrupted receiver), with the exception that $\mathcal{A}_h$ will only send msg_1 in round 1. On receiving msg_2 from the honest sender, $\mathcal{A}_h$ will run the output phase of $\mathcal{A}$ once again, using msg_2 as the input from the second round, to determine $\mathcal{A}$'s final output $\mathsf{out}_{\mathcal{A}}$ and return the result.

Now, consider the following adversary $\mathcal{A}'_{\mathsf{Real}}(1^n, z)$ in the real experiment, which runs $\mathcal{A}_h$:

1. Run $\mathcal{A}_h(1^n, z)$, which will start by producing a message msg_1 for π. Also run $\mathcal{A}(1^n, z)$ to produce a message $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x, \cdot)$, and send $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x, \mathsf{msg}_1)$ to the sender S.
2. S will return a message msg_2 for π. Run the output phase of $\mathcal{A}_h$ on this message.

3. $\mathcal{A}_h$ will output $\mathsf{out}_{\mathcal{A}_h}$; return it.

Also define a $T(n)\cdot\mathsf{poly}(n)$-time adversary $\mathcal{A}'$ in the hybrid experiment $\mathsf{Exec}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',R}(1^n,x,y,z)$ which runs the $(T(n)\cdot\mathsf{poly}(n)$-time) simulator $\mathcal{S}_h$ corresponding to the adversary $\mathcal{A}_h$ (as guaranteed by the definition of simulation-based security for π). $\mathcal{A}'(1^n,z)$ will do as follows:

1. Run $\mathcal{S}_h(1^n,z)$, which will start by producing a message m_1 to send to the ideal functionality $\mathcal{T}_h$; forward this message. Also run $\mathcal{A}(1^n,z)$ to produce a message $(\mathsf{pk}_{\mathsf{FHE}},\mathsf{pk}_{\mathsf{Del}},\mathsf{ct}_x,\mathsf{msg}_1)$, and send $(\mathsf{pk}_{\mathsf{FHE}},\mathsf{pk}_{\mathsf{Del}},\mathsf{ct}_x)$ to the sender S.
2. S will provide its input to $\mathcal{T}_h$, and subsequently $\mathcal{T}_h$ will return a result out. Forward out to $\mathcal{S}_h$.
3. $\mathcal{S}_h$ will return a simulated message $\mathsf{out}_{\mathcal{S}}$; return it.

We can use these adversaries to show that $\mathsf{Out}_{\Pi,\mathcal{A},R}(1^n,x,y,z)$ and $\mathsf{Out}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',R}(1^n,x,y,z)$ cannot be distinguished with non-negligible probability, based on the following two claims:

Claim 13. $\mathsf{Out}_{\Pi,\mathcal{A},R}(1^n,x,y,z)$ and $\mathsf{Out}_{\Pi,\mathcal{A}'_{\mathsf{Real}},R}(1^n,x,y,z)$ are identically distributed.

Proof. Importantly, recall that the adversaries $\mathcal{A}$ and $\mathcal{A}_h$ are deterministic and use z as their source of randomness. We begin by reproducing the experiment $\mathsf{Exec}_{\Pi,\mathcal{A},R}(1^n,x,y,z)$ for clarity:

1. Run $\mathcal{A}(1^n,z)$ to produce a message $(\mathsf{pk}_{\mathsf{FHE}},\mathsf{pk}_{\mathsf{Del}},\mathsf{ct}_x,\mathsf{msg}_1)$, and send it to the sender S.
2. S will return a message msg_2 for π. Run the output phase of $\mathcal{A}$ on this message.
3. $\mathcal{A}$ will output $\mathsf{out}_{\mathcal{A}}$; return it.

There are two semantic differences between the two experiments. Namely, $\mathcal{A}'_{\mathsf{Real}}$ uses $\mathcal{A}_h(1^n, z)$ rather than $\mathcal{A}(1^n, z)$ to produce msg_1, as well as to produce the final output. However, by the definition of $\mathcal{A}_h$, for any auxiliary input z it is clearly the case that $\mathcal{A}_h(1^n, z)$ and $\mathcal{A}(1^n, z)$ compute msg_1 and their final output in an identical way; hence, the full experiments are completely identical. □

So it is equivalent to compare $\mathsf{Out}_{\Pi,\mathcal{A}'_{\mathsf{Real}},R}(1^n, x, y, z)$ and $\mathsf{Out}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',R}(1^n, x, y, z)$. The following claim, then, yields the desired conclusion:

Claim 14. For any polynomial-time non-uniform distinguisher D, there exists negligible $\epsilon(\cdot)$ such that, for any $n \in \mathbb{N}$ and inputs x, y, z, the distributions $\mathsf{Out}_{\Pi,\mathcal{A}'_{\mathsf{Real}},R}(1^n, x, y, z)$ and $\mathsf{Out}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',R}(1^n, x, y, z)$ cannot be distinguished by D with probability greater than $\epsilon(n)$.

Proof. This will intuitively follow from the simulation-based security of the underlying protocol π.

Formally, assume for contradiction that there exists a non-uniform polynomial-time distinguisher D and polynomial $p(\cdot)$ such that, for infinitely many $n \in \mathbb{N}$, there are inputs x, y, z such that D is able to distinguish the two distributions $\mathsf{Out}_{\Pi,\mathcal{A}'_{\mathsf{Real}},R}(1^n, x, y, z)$ and $\mathsf{Out}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',R}(1^n, x, y, z)$ with probability $1/p(n)$. In this case, for each such n, there must exist some assignment r^* of the (honest) sender's randomness $r_{\mathsf{Enc}(y)}$ such that D distinguishes $\mathsf{Out}_{\Pi,\mathcal{A}'_{\mathsf{Real}},R}(1^n, x, y, z)|_{r^*}$ and $\mathsf{Out}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',R}(1^n, x, y, z)|_{r^*}$ (which denote the respective experiments with $r_{\mathsf{Enc}(y)}$ fixed to r^*) with probability at least $1/p(n)$.

Given fixed x, y, z and fixed $r_{\mathsf{Enc}(y)} = r^*$, recall the inputs $x' = (x, r_{\mathsf{FHE}}, r_{\mathsf{Del}}, r_{\mathsf{Enc}(x)})$ and $y' = (y, \mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x, \mathsf{ct}_y, \mathsf{ct}_{\mathsf{out}}, \pi_{\mathsf{Del}}, r_{\mathsf{Enc}(y)}, T)$ provided by the receiver and sender (respectively) to the protocol π; all randomness used to generate these inputs is now fixed

(since $r_{\mathsf{FHE}}, r_{\mathsf{Del}}, r_{\mathsf{Enc}(x)}$ are given by z), which means that each assignment of inputs x, y, z, r^* for Π corresponds uniquely to fixed inputs x', y', z for the underlying protocol π.

This implies that we can use the distinguisher D directly to break the simulation-based security of the underlying protocol π. Specifically, given the previously fixed x, y, z, r^*, consider the distributions $\mathsf{Out}_{\pi,\mathcal{A}_h,R}(1^n, x', y', z)$ and $\mathsf{Out}_{\pi_h,\mathcal{S}_h,R}(1^n, x', y', z)$, where π_h is the idealized version of π (as executed in Π_h) where both parties send their respective inputs x', y' to the functionality $\mathcal{T}_h$, which computes $h(x', y')$.

Since, by definition, the outputs of $\mathcal{A}'_{\mathsf{Real}}$ and $\mathcal{A}'$ (fixing inputs x, y, z, r^*) are given by the outputs of $\mathcal{A}_h$ and $\mathcal{S}_h$ (fixing the corresponding inputs x', y', z) in the respective experiments π and π_h, we have that

$$\mathsf{Out}_{\Pi,\mathcal{A}'_{\mathsf{Real}},R}(1^n, x, y, z)|_{r^*} = \mathsf{Out}_{\pi,\mathcal{A}_h,R}(1^n, x', y', z)$$

and

$$\mathsf{Out}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',R}(1^n, x, y, z)|_{r^*} = \mathsf{Out}_{\pi_h,\mathcal{S}_h,R}(1^n, x', y', z)$$

But, since D distinguishes $\mathsf{Out}_{\Pi,\mathcal{A}'_{\mathsf{Real}},R}(1^n, x, y, z)|_{r^*}$ and $\mathsf{Out}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',R}(1^n, x, y, z)|_{r^*}$ with probability $1/p(n)$, it must also distinguish $\mathsf{Out}_{\pi,\mathcal{A}_h,R}(1^n, x', y', z)$ and $\mathsf{Out}_{\pi_h,\mathcal{S}_h,R}(1^n, x', y', z)$ with the same probability. And, as this holds for infinitely many $n \in \mathbb{N}$, it follows that D contradicts the simulation-based security of the underlying protocol π (w.r.t. non-uniform polynomial-time distinguishers), which is a contradiction. □

Corrupted Sender. Given adversary $\mathcal{A}$ against the real experiment $\mathsf{Exec}_{\Pi,\mathcal{A},S}(1^n, x, y, z)$, let $\mathcal{A}_h$ as before be the respective adversary against simulation-based security of the 2PC protocol π. Now $\mathcal{A}_h$ receives message msg_1 from the honest receiver and must generate msg_2 using $\mathcal{A}$; however, $\mathcal{A}$ also requires the public parameters $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x)$ from the receiver's first-round message as input. As such, we concatenate these parameters with the auxiliary

input z provided to $\mathcal{A}_h$. Formally, we let $\mathcal{A}_h(1^n, (\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x, z))$ given message msg_1 run $\mathcal{A}(1^n, z)$ given message $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x, \mathsf{msg}_1)$. Then let $\mathcal{A}'_{\mathsf{Real}}(1^n, z)$ in the real experiment proceed as follows:

1. Receive a message $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x, \mathsf{msg}_1)$ from the receiver R.
2. Run $\mathcal{A}_h(1^n, z' = (\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x, z))$ with msg_1 as the input from the receiver; forward the message msg_2 from $\mathcal{A}_h$ to the sender.
3. Output whatever $\mathcal{A}_h$ outputs.

By simulation-based security, then, there is also a simulator $\mathcal{S}_h$ in the idealized experiment for π which sends the relevant inputs to the ideal functionality $\mathcal{T}_h$ but returns no output to the receiver. Let $\mathcal{A}'(1^n, z)$ be a $T(n) \cdot \mathsf{poly}(n)$-time adversary in the hybrid experiment $\mathsf{Exec}^{\mathcal{T}_h}_{\Pi_h, \mathcal{A}', S}(1^n, x, y, z)$ which does the following:

1. Receive a message $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x)$ from the receiver R.
2. Run $\mathcal{S}_h(1^n, z' = (\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x, z))$. $\mathcal{S}_h$ will produce a message m_2 to send to the ideal functionality $\mathcal{T}_h$; forward it to $\mathcal{T}_h$.
3. Output whatever $\mathcal{S}_h$ outputs.

We must show that, given this $\mathcal{A}'$, both (1) the outputs of the adversary and (2) the outputs of the honest receiver R between $\mathsf{Exec}_{\Pi, \mathcal{A}, S}(1^n, x, y, z)$ and $\mathsf{Exec}^{\mathcal{T}_h}_{\Pi_h, \mathcal{A}', S}(1^n, x, y, z)$ cannot be distinguished. The following claims imply the desired conclusion:

Claim 15. $\mathsf{Out}_{\Pi, \mathcal{A}, S}(1^n, x, y, z)$ and $\mathsf{Out}_{\Pi, \mathcal{A}'_{\mathsf{Real}}, S}(1^n, x, y, z)$ are identically distributed.

Proof. This follows directly from the fact that the adversaries $\mathcal{A}$ and $\mathcal{A}_h$ are deterministic and use z as their source of randomness, and the fact that $\mathcal{A}_h(1^n, (\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x, z))$ given

message msg_1 behaves identically to $\mathcal{A}(1^n, z)$ given message $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x, \mathsf{msg}_1)$, implying that msg_2 and the output of $\mathcal{A}$ (resp. $\mathcal{A}_h$) are identically distributed between the experiments.

□

So it suffices to compare $\mathsf{Out}_{\Pi,\mathcal{A}'_{\mathsf{Real}},S}(1^n, x, y, z)$ and $\mathsf{Out}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',S}(1^n, x, y, z)$, which we do as follows:

Claim 16. For any polynomial-time non-uniform distinguisher D, there exists negligible $\epsilon(\cdot)$ such that, for any $n \in \mathbb{N}$ and inputs x, y, z, the distributions $\mathsf{Out}_{\Pi,\mathcal{A}'_{\mathsf{Real}},S}(1^n, x, y, z)$ and $\mathsf{Out}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',S}(1^n, x, y, z)$ cannot be distinguished by D with probability greater than $\epsilon(n)$.

Proof. Once again this will intuitively follow from the simulation-based security of the underlying protocol π.

Formally, assume for contradiction that there exists a non-uniform polynomial-time distinguisher D and polynomial $p(\cdot)$ such that, for infinitely many $n \in \mathbb{N}$, there are inputs x, y, z such that D is able to distinguish the two distributions $\mathsf{Out}_{\Pi,\mathcal{A}'_{\mathsf{Real}},S}(1^n, x, y, z)$ and $\mathsf{Out}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',S}(1^n, x, y, z)$ with probability $1/p(n)$. As with the corrupted receiver case, for each such n, there must exist some assignment r^* of the (honest) receiver's randomness $r_R = (r_{\mathsf{FHE}}, r_{\mathsf{Del}}, r_{\mathsf{Enc}(x)})$ such that D distinguishes $\mathsf{Out}_{\Pi,\mathcal{A}'_{\mathsf{Real}},S}(1^n, x, y, z)|_{r^*}$ and $\mathsf{Out}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',S}(1^n, x, y, z)|_{r^*}$ (which once again denote the respective experiments with r_S fixed to r^*) with probability at least $1/p(n)$.

Given x, y, z, and $r_R = r^*$ fixed, we must argue as before that this corresponds uniquely to fixed inputs x', y', z' for the underlying protocol π. In this case, this will follow from the fact that $\mathcal{A}'_{\mathsf{Real}}$ and $\mathcal{A}'$ run $\mathcal{A}_h$ and $\mathcal{S}_h$ (respectively) on the input $(1^n, z')$, where $z' = (\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x, z)$ is fully determined by z, x, and the randomness r_R consumed by the receiver in the first round (notice that this holds because $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x)$ is part of the

honest receiver's first message in the protocol Π). Furthermore, x' is uniquely determined by x and r_R, and y', the sender's input, is uniquely determined by z', y, and the sender's randomness (which, since the sender is malicious, is given by z).

As before, consider the distributions $\mathsf{Out}_{\pi,\mathcal{A}_h,R}(1^n, x', y', z')$ and $\mathsf{Out}_{\pi_h,\mathcal{S}_h,R}(1^n, x', y', z')$, where π_h is the idealized version of π (as executed in Π_h) where both parties send their respective inputs x', y' to the functionality $\mathcal{T}_h$. As before, the outputs of $\mathcal{A}'_{\mathsf{Real}}$ and $\mathcal{A}'$ (fixing inputs x, y, z, r^*) are given by the outputs of $\mathcal{A}_h$ and $\mathcal{S}_h$ (fixing the corresponding inputs x', y', z') in the respective experiments π and π_h; furthermore, by the definition of the protocols Π and Pi_h, the (honest) receiver's outputs are given by the outputs of π and π_h (respectively). So we once again determine that

$$\mathsf{Out}_{\Pi,\mathcal{A}'_{\mathsf{Real}},S}(1^n, x, y, z)|_{r^*} = \mathsf{Out}_{\pi,\mathcal{A}_h,S}(1^n, x', y', z')$$

and

$$\mathsf{Out}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',S}(1^n, x, y, z)|_{r^*} = \mathsf{Out}_{\pi_h,\mathcal{S}_h,S}(1^n, x', y', z')$$

which, since D distinguishes $\mathsf{Out}_{\Pi,\mathcal{A}'_{\mathsf{Real}},S}(1^n, x, y, z)|_{r^*}$ and $\mathsf{Out}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',S}(1^n, x, y, z)|_{r^*}$ with probability $1/p(n)$, implies that it must also distinguish $\mathsf{Out}_{\pi,\mathcal{A}_h,S}(1^n, x', y', z')$ and $\mathsf{Out}_{\pi_h,\mathcal{S}_h,S}(1^n, x', y', z')$ with the same probability. Thus, since the above implies that there exist x', y', z' for infinitely many $n \in \mathbb{N}$ such that D distinguishes the respective distributions, it follows that D contradicts the simulation-based security of π (w.r.t. non-uniform polynomial-time distinguishers), a contradiction. □

□

3.3.4 Comparing Hybrid and Ideal Executions

Next, we need to compare the hybrid execution $\mathsf{Exec}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',I}(1^n, x, y, z)$ to the "ideal" execution $\mathsf{Exec}^{\mathcal{T}_f}_{\Pi_f,\mathcal{S},I}(1^n, x, y, z)$ to finish the proof of Theorem 7.

Lemma 11. For any $T(n) \cdot \mathsf{poly}(n)$-time adversary $\mathcal{A}'$ corrupting party $I \in \{S, R\}$, there exists a $T(n) \cdot \mathsf{poly}(n)$-time simulator $\mathcal{S}$ such that, for any non-uniform polynomial-time distinguisher D, there exists a negligible function $\epsilon(\cdot)$ such that, for all $n \in \mathbb{N}$, $x, y \in \{0,1\}^n$, and auxiliary input z, D distinguishes the distributions $\mathsf{Out}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',I}(1^n, x, y, z)$ and $\mathsf{Out}^{\mathcal{T}_f}_{\Pi_f,\mathcal{S},I}(1^n, x, y, z)$ with at most probability $\epsilon(n)$.

Proof. We again separate into two cases, based on whether $I = R$ (the receiver is corrupted) or $I = S$ (the sender is corrupted).

Corrupted Receiver. In this case, define a $T(n) \cdot \mathsf{poly}(n)$-time simulator $\mathcal{S}_R$ which does as follows:

1. Run the corrupted receiver $\mathcal{A}'$. $\mathcal{A}'$, in the first round, will output a message $(x, r_{\mathsf{FHE}}, r_{\mathsf{Del}}, r_{\mathsf{Enc}})$ to be sent to $\mathcal{T}_h$. Send x to the ideal functionality $\mathcal{T}_f$.
2. Receive an output message out from the ideal functionality $\mathcal{T}_f$. If out is $\perp$, return $\perp$ to $\mathcal{A}'$ (as the output of $\mathcal{T}_h$).
3. Verify the following. If any checks fail, return $\perp$ to $\mathcal{A}'$.
   (a) $(\mathsf{pk}_{\mathsf{FHE}}, \cdot) = \mathsf{Gen}_{\mathsf{FHE}}(1^n; r_{\mathsf{FHE}})$
   (b) $(\mathsf{pk}_{\mathsf{Del}}, \cdot) = \mathsf{Gen}_{\mathsf{Del}}(1^\lambda; r_{\mathsf{Del}})$
   (c) $\mathsf{ct}_x = \mathsf{Enc}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, x; r_{\mathsf{Enc}(x)})$

4. If all checks in the previous step pass, return out to $\mathcal{A}'$. Finally, output whatever $\mathcal{A}'$ outputs.

It suffices here to argue that the output which $\mathcal{S}_R$ returns to $\mathcal{A}'$ in the ideal experiment is identically distributed to the output which $\mathcal{T}_h$ would return to $\mathcal{A}'$ in the hybrid experiment, as this, combined with the observation that the only input $\mathcal{A}'$ receives (aside from the auxiliary input z) is the output from $\mathcal{T}_h$, allows us to conclude that $\mathcal{A}'$'s views in $\mathsf{Exec}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',R}(1^n, x, y, z)$ and $\mathsf{Exec}^{\mathcal{T}_f}_{\Pi_f,\mathcal{S}_R,R}(1^n, x, y, z)$ (and hence $\mathcal{A}'$'s outputs) are likewise identically distributed. We can argue this using the following claims:

Claim 17. If S is honest, then, given messages $(x, r_{\mathsf{FHE}}, r_{\mathsf{Del}}, r_{\mathsf{Enc}})$ and $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x)$ from $\mathcal{A}'$, step (4) of $\mathcal{S}_R$ succeeds (i.e., does not return $\perp$) in Π_f if and only if all checks in step (1) of the functionality h described in Figure 3.3 succeed in the respective instance of Π_h.

Proof. The "if" direction is trivial since the checks in step (4) of $\mathcal{S}_R$ are a strict subset of the checks in step (1) of h.

The "only if" direction follows from the assumption that S is honest, and will hence compute $\mathsf{ct}_y = \mathsf{Enc}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, y; r_{\mathsf{Enc}(y)})$ correctly using the correct inputs. □

Claim 18. If S is honest and all checks in step (1) of the functionality h described in Figure 3.3 succeed in Π_h, then, with probability 1, step (3) of the functionality h will not return $\perp$.

Proof. Since step (1) is successful, we know that $(\mathsf{pk}_{\mathsf{Del}}, \mathsf{sk}_{\mathsf{Del}}) = \mathsf{Gen}_{\mathsf{Del}}(1^\lambda, r_{\mathsf{Del}})$; moreover, since S is honest, we know that it must have computed $(\mathsf{ct}_{\mathsf{out}}, \pi_{\mathsf{Del}}, 1^T) = \mathsf{Comp}_{\mathsf{Del}}(\mathsf{pk}_{\mathsf{Del}}, g, \mathsf{ct}_x, \mathsf{ct}_y)$ correctly (and using the correct $\mathsf{pk}_{\mathsf{Del}}$ and ct_x, since the checks

in step (1) passed). It follows by perfect completeness of the delegation scheme $(\mathsf{Gen}_{\mathsf{Del}}, \mathsf{Comp}_{\mathsf{Del}}, \mathsf{Ver}_{\mathsf{Del}})$ that

$$\mathsf{Ver}_{\mathsf{Del}}(\mathsf{sk}_{\mathsf{Del}}, g, \mathsf{ct}_x, \mathsf{ct}_y, \mathsf{ct}_{\mathsf{out}}, \pi_{\mathsf{Del}}, T) = \mathsf{Accept}$$

as desired. □

Claim 19. If S is honest and, in Π_h, all checks in step (1) of the functionality h described in Figure 3.3 succeed, and step (3) of the functionality h does not return $\perp$, then the value of out returned by step (4) of h will be equal to $f(x, y)$ with probability 1.

Proof. Since S is honest and step (1) is successful, we know, as in the previous claim, that $(\mathsf{pk}_{\mathsf{Del}}, \mathsf{sk}_{\mathsf{Del}}) = \mathsf{Gen}_{\mathsf{Del}}(1^\lambda, r_{\mathsf{Del}})$ and $(\mathsf{ct}_{\mathsf{out}}, \pi_{\mathsf{Del}}, 1^T) = \mathsf{Comp}_{\mathsf{Del}}(\mathsf{pk}_{\mathsf{Del}}, g, \mathsf{ct}_x, \mathsf{ct}_y)$. It follows by correctness of the delegation scheme $(\mathsf{Gen}_{\mathsf{Del}}, \mathsf{Comp}_{\mathsf{Del}}, \mathsf{Ver}_{\mathsf{Del}})$ that

$$\mathsf{ct}_{\mathsf{out}} = g(\mathsf{ct}_x, \mathsf{ct}_y) = \mathsf{Eval}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, f, \mathsf{ct}_x, \mathsf{ct}_y)$$

It suffices to show that this will decrypt to the correct output $\mathsf{out} = f(x, y)$. This holds due to perfect correctness of the FHE scheme $(\mathsf{Gen}_{\mathsf{FHE}}, \mathsf{Enc}_{\mathsf{FHE}}, \mathsf{Dec}_{\mathsf{FHE}}, \mathsf{Eval}_{\mathsf{FHE}})$; specifically, since ct_x and ct_y are encryptions of x and y, respectively:

$$\mathsf{Dec}_{\mathsf{FHE}}(\mathsf{sk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{out}}) = \mathsf{Dec}_{\mathsf{FHE}}(\mathsf{sk}_{\mathsf{FHE}}, \mathsf{Eval}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, f, \mathsf{ct}_x, \mathsf{ct}_y)) = f(x, y)$$

□

Chaining together Claims 17, 18, and 19 leads us to the conclusion that (by Claim 17), $\mathcal{S}_R$ returns $\perp$ in $\mathsf{Exec}^{\mathcal{T}_f}_{\Pi_f, \mathcal{S}_R, R}(1^n, x, y, z)$ if and only if $\mathcal{T}_h$ would return $\perp$ (from step (1)) in the respective execution of $\mathsf{Exec}^{\mathcal{T}_h}_{\Pi_h, \mathcal{A}', R}(1^n, x, y, z)$, and furthermore, if this event does not occur, then (by Claims 18 and 19 as well as the definition of $\mathcal{S}_R$) both $\mathcal{S}_R$ (in $\mathsf{Exec}^{\mathcal{T}_f}_{\Pi_f, \mathcal{S}_R, R}(1^n, x, y, z)$) and $\mathcal{T}_h$ (in the respective execution of $\mathsf{Exec}^{\mathcal{T}_h}_{\Pi_h, \mathcal{A}', R}(1^n, x, y, z)$) will return an output out that

is precisely equal to $f(x, y)$, where x is the value sent by the adversary to $\mathcal{T}_h$ and y is the (honest) sender's input. This completes the argument for the case $I = R$.

Corrupted Sender. In the case $I = S$, define a $T(n) \cdot \mathsf{poly}(n)$-time simulator $\mathcal{S}_S$ which does as follows:

1. Generate $r_{\mathsf{FHE}}, r_{\mathsf{Del}}, r_{\mathsf{Enc}(x)} \leftarrow \{0,1\}^*$ and $(\mathsf{pk}_{\mathsf{FHE}}, \cdot) = \mathsf{Gen}_{\mathsf{FHE}}(1^n; r_{\mathsf{FHE}}), (\mathsf{pk}_{\mathsf{Del}}, \cdot) = \mathsf{Gen}_{\mathsf{Del}}(1^\lambda; r_{\mathsf{Del}}), \mathsf{ct}_x = \mathsf{Enc}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, 0; r_{\mathsf{Enc}(x)})$
2. Run the corrupted sender $\mathcal{A}'$ using input $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x)$. $\mathcal{A}'$ will generate a message $(y', \mathsf{pk}'_{\mathsf{FHE}}, \mathsf{pk}'_{\mathsf{Del}}, \mathsf{ct}'_x, \mathsf{ct}'_y, \mathsf{ct}'_{\mathsf{out}}, \pi'_{\mathsf{Del}}, r'_{\mathsf{Enc}(y)}, T')$ to send to $\mathcal{T}_h$. Perform the following checks to verify this message, and return $\perp$ to $\mathcal{T}_f$ (causing it to output $\perp$) if any of them fail.
   (a) $\mathsf{pk}_{\mathsf{FHE}} = \mathsf{pk}'_{\mathsf{FHE}}$, $\mathsf{pk}_{\mathsf{Del}} = \mathsf{pk}'_{\mathsf{Del}}$, $\mathsf{ct}_x = \mathsf{ct}'_x$.
   (b) $\mathsf{ct}'_y = \mathsf{Enc}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, y'; r'_{\mathsf{Enc}(y)})$
   (c) $\mathsf{Ver}_{\mathsf{Del}}(\mathsf{sk}_{\mathsf{Del}}, g, \mathsf{ct}_x, \mathsf{ct}_y, \mathsf{ct}'_{\mathsf{out}}, \pi'_{\mathsf{Del}}, T') = \mathsf{Accept}$ for the functionality $g(c_1, c_2) = \mathsf{Eval}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, f, c_1, c_2)$.
3. Otherwise (if the above checks pass), send y' to $\mathcal{T}_f$. Finally, output whatever $\mathcal{A}'$ outputs.

As this case has interesting subtleties, we lead the formal proof with a brief overview. Recall that, for this case, we need not only to verify that the adversary $\mathcal{A}'$'s views in the experiments $\mathsf{Exec}^{\mathcal{T}_h}_{\Pi_h, \mathcal{A}', S}(1^n, x, y, z)$ and $\mathsf{Exec}^{\mathcal{T}_f}_{\Pi_f, \mathcal{S}_S, S}(1^n, x, y, z)$ (and hence $\mathcal{A}'$'s outputs) cannot be distinguished, but also that the honest receiver R's outputs cannot be distinguished between the two experiments.

The natural way to do this would be to begin by creating a hybrid protocol Π'_h where the receiver, instead of sending a ciphertext of their input x in the first round, sends the corresponding ciphertext of 0 (as the simulator does when running $\mathcal{A}'$ in Π_f). Ostensibly, this would allow us to show that the output distributions between Π_h and Π'_h are close by using the CPA-security of the underlying FHE protocol to assert that the ciphertexts, and hence the views of $\mathcal{A}'$, are indistinguishable between the two experiments. And while this does indeed directly imply that the *adversary's* outputs are close, we run into an issue the moment we consider the *receiver's* output; specifically, the receiver's output is the output from the ideal functionality $\mathcal{T}_h$, which among other things depends on *the secret key* $\mathsf{sk}_{\mathsf{FHE}}$ *and the randomness* r_{FHE} *used to generate it.* In fact, this makes a reduction from Π'_h to the security of the FHE scheme impossible (using current techniques), since a hypothetical adversary simulating this functionality would only know $\mathsf{pk}_{\mathsf{FHE}}$.

Instead we will have to consider an alternate functionality h' which only depends on the public key $\mathsf{pk}_{\mathsf{FHE}}$ and does not use the randomness or secret key. Specifically, rather than decrypting the final result $\mathsf{ct}_{\mathsf{out}}$, h' will instead simply return $f(x, y')$. We then show that the output distribution of $\Pi_{h'}$ is *statistically close* to that of Π_h. Specifically, they are identical except when the adversary $\mathcal{A}'$ can force the ideal functionality h' to verify a proof π_{Del} of an incorrect ciphertext $\mathsf{ct}_{\mathsf{Out}}$—this implies that their statistical distance must be at most the (negligible) soundness error of delegation.[3] Now, given $\Pi_{h'}$, we can finally consider a protocol $\Pi'_{h'}$ where the receiver uses a ciphertext of 0; now that h' no longer depends on $\mathsf{sk}_{\mathsf{FHE}}$, the reduction to the CPA-security will go through (for both the adversary's and receiver's outputs), and we can lastly compare $\mathsf{Exec}^{\mathcal{T}_h}_{\Pi'_{h'},\mathcal{A}',S}(1^n, x, y, z)$ and $\mathsf{Exec}^{\mathcal{T}_f}_{\Pi_f,\mathcal{S}_S,S}(1^n, x, y, z)$ to

[3] An attentive reader might wonder at this point why, in doing this, we are not simply backing ourselves into the same corner, since indeed $\mathcal{T}_h$ and even $\mathcal{T}_{h'}$ are very much dependent on the randomness r_{Del} and secret key $\mathsf{sk}_{\mathsf{Del}}$. The intuitive answer is that, unlike with the reduction to FHE, we are able to "outsource" the dependence on $\mathsf{sk}_{\mathsf{Del}}$ in $\mathcal{T}_{h'}$ to the security game for the soundness of delegation, allowing us to effectively *emulate* h' without said secret key in the adversary we construct.

show that, actually, the output distributions are identically distributed.

We continue to the formal proof. Let h' be the functionality defined as h, but with four key differences:

- h', instead of taking input r_{FHE} from the receiver, takes input $\mathsf{pk}_{\mathsf{FHE}}$.
- In step (1), instead of verifying that $(\mathsf{pk}_{\mathsf{FHE}}, \cdot) = \mathsf{Gen}_{\mathsf{FHE}}(1^n, r_{\mathsf{FHE}})$, h' verifies that the sender's and receiver's inputs $\mathsf{pk}_{\mathsf{FHE}}$ match.
- In step (2), h' no longer computes $(\cdot, \mathsf{sk}_{\mathsf{FHE}}) = \mathsf{Gen}_{\mathsf{FHE}}(1^n; r_{\mathsf{FHE}})$.
- In step (4), h' returns $f(x, y)$ rather than $\mathsf{Dec}_{\mathsf{FHE}}(\mathsf{sk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{out}})$.

Let $\Pi_{h'}$ be defined identically to Π_h except that both parties use the ideal functionality $\mathcal{T}_{h'}$ in place of $\mathcal{T}_h$ and the receiver inputs $\mathsf{pk}_{\mathsf{FHE}}$ to $\mathcal{T}_{h'}$ instead of r_{FHE} as specified above. We state the following claim:

Claim 20. There exists negligible $\epsilon(\cdot)$ such that, for all $n \in \mathbb{N}$ and inputs x, y, z, the output distributions $\mathsf{Out}^{\mathcal{T}_h}_{\Pi_h, \mathcal{A}', S}(1^n, x, y, z)$ and $\mathsf{Out}^{\mathcal{T}_{h'}}_{\Pi_{h'}, \mathcal{A}', S}(1^n, x, y, z)$ are $\epsilon(n)$-statistically close.

Proof. Intuitively, this will follow from the soundness of the delegation scheme $(\mathsf{Gen}_{\mathsf{Del}}, \mathsf{Comp}_{\mathsf{Del}}, \mathsf{Ver}_{\mathsf{Del}})$. First, observe that the adversary's views in $\mathsf{Exec}^{\mathcal{T}_h}_{\Pi_h, \mathcal{A}', S}(1^n, x, y, z)$ and $\mathsf{Exec}^{\mathcal{T}_{h'}}_{\Pi_{h'}, \mathcal{A}', S}(1^n, x, y, z)$, and thus the adversary's outputs, are identically distributed; hence, it suffices to argue about the honest receiver's output, i.e., the output of $\mathcal{T}_h$ or $\mathcal{T}_{h'}$.

Second, since the receiver R is honest, the fact that h' verifies that the sender's and receiver's inputs $\mathsf{pk}_{\mathsf{FHE}}$ match is equivalent to the verification in h of the sender's $\mathsf{pk}_{\mathsf{FHE}}$ (that $(\mathsf{pk}_{\mathsf{FHE}}, \cdot) = \mathsf{Gen}_{\mathsf{FHE}}(1^n, r_{\mathsf{FHE}})$), since the receiver's input $\mathsf{pk}_{\mathsf{FHE}}$ will always be equal to $\mathsf{Gen}_{\mathsf{FHE}}(1^n, r_{\mathsf{FHE}})$. So the only change that can possibly affect the output of $\mathcal{T}_{h'}$ compared to $\mathcal{T}_h$ in the corrupted sender case is the fact that h' returns $f(x, y)$ rather than $\mathsf{Dec}_{\mathsf{FHE}}(\mathsf{sk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{out}})$.

Now, assume for the sake of contradiction that there is some polynomial $p(\cdot)$ such that, for infinitely many $n \in \mathbb{N}$, there exist x, y, z so that the ideal functionality's output is different between $\mathsf{Exec}^{\mathcal{T}_h}_{\Pi_h, \mathcal{A}', S}(1^n, x, y, z)$ and $\mathsf{Exec}^{\mathcal{T}_{h'}}_{\Pi_{h'}, \mathcal{A}', S}(1^n, x, y, z)$ with probability $1/p(n)$. We shall use this to construct a $T(n) \cdot \mathsf{poly}(n)$-time adversary $\mathcal{A}_{\mathsf{Del}}$ to break the soundness of the delegation scheme with probability $1/p(n)$.

Specifically, let $\mathcal{A}_{\mathsf{Del}}$ do as follows on input $(1^n, \mathsf{pk}_{\mathsf{Del}})$:

1. Generate $r_{\mathsf{FHE}}, r_{\mathsf{Enc}(x)} \leftarrow \{0,1\}^*$ and $(\mathsf{pk}_{\mathsf{FHE}}, \cdot) = \mathsf{Gen}_{\mathsf{FHE}}(1^n; r_{\mathsf{FHE}}), \mathsf{ct}_x = \mathsf{Enc}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, x; r_{\mathsf{Enc}(x)})$.
2. Run the corrupted sender $\mathcal{A}'$ with sender input y, auxiliary input z, and first-round message $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x)$. $\mathcal{A}'$ will generate a message $(y', \mathsf{pk}'_{\mathsf{FHE}}, \mathsf{pk}'_{\mathsf{Del}}, \mathsf{ct}'_x, \mathsf{ct}'_y, \mathsf{ct}'_{\mathsf{out}}, \pi'_{\mathsf{Del}}, r'_{\mathsf{Enc}(y)}, T')$ to send to the ideal functionality ($\mathcal{T}_h$ or $\mathcal{T}_{h'}$).
3. Run $(\mathsf{ct}_{\mathsf{out}}, \pi_{\mathsf{Del}}, 1^T) \leftarrow \mathsf{Comp}_{\mathsf{Del}}(\mathsf{pk}_{\mathsf{Del}}, g, \mathsf{ct}_x, \mathsf{ct}'_y)$ for the functionality $g(c_1, c_2) = \mathsf{Eval}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, f, c_1, c_2)$
4. Verify the following and abort if any are false.
   (a) $\mathsf{pk}_{\mathsf{FHE}} = \mathsf{pk}'_{\mathsf{FHE}}$, $\mathsf{pk}_{\mathsf{Del}} = \mathsf{pk}'_{\mathsf{Del}}$, $\mathsf{ct}_x = \mathsf{ct}'_x$
   (b) $\mathsf{ct}'_y = \mathsf{Enc}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, y'; r'_{\mathsf{Enc}(y)})$
5. Otherwise, return $(g, \mathsf{ct}_x, \mathsf{ct}'_y, \mathsf{ct}_{\mathsf{out}}, \mathsf{ct}'_{\mathsf{out}}, \pi_{\mathsf{Del}}, \pi'_{\mathsf{Del}}, 1^T, 1^{T'})$.

We claim that $\mathcal{A}_{\mathsf{Del}}$ returns a tuple $(g, \mathsf{ct}_x, \mathsf{ct}'_y, \mathsf{ct}_{\mathsf{out}}, \mathsf{ct}'_{\mathsf{out}}, \pi_{\mathsf{Del}}, \pi'_{\mathsf{Del}}, 1^T, 1^{T'})$ such that $\mathsf{ct}_{\mathsf{out}} \neq \mathsf{ct}'_{\mathsf{out}}$ but $\mathsf{Ver}_{\mathsf{Del}}(\mathsf{sk}_{\mathsf{Del}}, g, \mathsf{ct}_x, \mathsf{ct}'_y, \mathsf{ct}_{\mathsf{out}}, \pi_{\mathsf{Del}}, T) = \mathsf{Ver}_{\mathsf{Del}}(\mathsf{sk}_{\mathsf{Del}}, g, \mathsf{ct}_x, \mathsf{ct}'_y, \mathsf{ct}'_{\mathsf{out}}, \pi'_{\mathsf{Del}}, T') = \mathsf{Accept}$—that is, $\mathcal{A}_{\mathsf{Del}}$ breaks soundness of the delegation scheme—precisely when h decrypts

a ciphertext that is not equal to $\mathsf{ct}_{\mathsf{out}}$ as returned by $\mathsf{Comp}_{\mathsf{Del}}(\mathsf{pk}_{\mathsf{Del}}, g, \mathsf{ct}_x, \mathsf{ct}_y)$ for the corresponding functionality and inputs; furthermore, we claim that this is the only case where h and h' may not be identically distributed.

To verify this, we start by observing that the input to $\mathcal{A}'$ in step (2) of $\mathcal{A}_{\mathsf{Del}}$ is identically distributed to the inputs in the experiments $\mathsf{Exec}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',S}(1^n, x, y, z)$ and $\mathsf{Exec}^{\mathcal{T}_{h'}}_{\Pi_{h'},\mathcal{A}',S}(1^n, x, y, z)$, since $\mathsf{pk}_{\mathsf{Del}}$ is honestly generated and the receiver is honest. Furthermore, given the message from $\mathcal{A}'$ to the ideal functionality, as well as the fact that R is honest, we can assert that the checks in step (4) of $\mathcal{A}_{\mathsf{Del}}$ are equivalent to the checks in step (1) of h or h', since the receiver's inputs $\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x$ are guaranteed to be honestly generated. So, comparing $\mathcal{T}_h$ and $\mathcal{T}_{h'}$ for a particular interaction, there are four possible outcomes, which we shall analyze:

1. Step (1) of h or h' fails, in which case both return $\perp$ (and $\mathcal{A}_{\mathsf{Del}}$ will abort).
2. Step (1) succeeds, but the verification in step (3) fails, in which case both will return $\perp$ (and $\mathcal{A}_{\mathsf{Del}}$ will produce output $(g, \mathsf{ct}_x, \mathsf{ct}'_y, \mathsf{ct}_{\mathsf{out}}, \mathsf{ct}'_{\mathsf{out}}, \pi_{\mathsf{Del}}, \pi'_{\mathsf{Del}}, 1^T, 1^{T'})$ which is *rejected* because $(\mathsf{ct}'_{\mathsf{out}}, \pi'_{\mathsf{Del}})$ fails to verify).
3. Steps (1) and (3) succeed, and $\mathsf{ct}'_{\mathsf{out}}$ given by the adversary is the same as the correct $(\mathsf{ct}_{\mathsf{out}}, \cdot, \cdot) = \mathsf{Comp}_{\mathsf{Del}}(\mathsf{pk}_{\mathsf{Del}}, g, \mathsf{ct}_x, \mathsf{ct}'_y)$, in which case the outputs of h and h' are identical and not $\perp$ by perfect correctness of Enc and Eval, as well as correctness of the delegation scheme.

   Specifically, considering the inputs to h, we know by correctness of delegation that, since $(\mathsf{ct}'_{\mathsf{out}}, \cdot, \cdot) = \mathsf{Comp}_{\mathsf{Del}}(\mathsf{pk}_{\mathsf{Del}}, g, \mathsf{ct}_x, \mathsf{ct}'_y)$, $\mathsf{ct}'_{\mathsf{out}} = g(\mathsf{ct}_x, \mathsf{ct}'_y) = \mathsf{Eval}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, f, \mathsf{ct}_x, \mathsf{ct}'_y)$. Furthermore, by perfect correctness of the FHE scheme and the fact that ct_x and ct'_y are encryptions of x and y, respectively:

$$\mathsf{Dec}_{\mathsf{FHE}}(\mathsf{sk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{out}}) = \mathsf{Dec}_{\mathsf{FHE}}(\mathsf{sk}_{\mathsf{FHE}}, \mathsf{Eval}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, f, \mathsf{ct}_x, \mathsf{ct}'_y)) = f(x, y')$$

that is, the output of h will be identical to the output $f(x, y')$ of h'. In this case, $\mathcal{A}_{\mathsf{Del}}$ will produce output $(g, \mathsf{ct}_x, \mathsf{ct}'_y, \mathsf{ct}_{\mathsf{out}}, \mathsf{ct}'_{\mathsf{out}}, \pi_{\mathsf{Del}}, \pi'_{\mathsf{Del}}, 1^T, 1^{T'})$ which is *rejected* because $\mathsf{ct}'_{\mathsf{out}} = \mathsf{ct}_{\mathsf{out}}$.

4. Steps (1) and (3) succeed, and $\mathsf{ct}'_{\mathsf{out}}$ given by the adversary is *not* the same as the correct $(\mathsf{ct}_{\mathsf{out}}, \cdot, \cdot) = \mathsf{Comp}_{\mathsf{Del}}(\mathsf{pk}_{\mathsf{Del}}, g, \mathsf{ct}_x, \mathsf{ct}'_y)$, in which case the outputs of h and h' *may be different* (and $\mathcal{A}_{\mathsf{Del}}$ will produce output $(g, \mathsf{ct}_x, \mathsf{ct}'_y, \mathsf{ct}_{\mathsf{out}}, \mathsf{ct}'_{\mathsf{out}}, \pi_{\mathsf{Del}}, \pi'_{\mathsf{Del}}, 1^T, 1^{T'})$ which is *accepted* because $\mathsf{ct}'_{\mathsf{out}} \neq \mathsf{ct}_{\mathsf{out}}$ and $(\mathsf{ct}'_{\mathsf{out}}, \pi'_{\mathsf{Del}}, 1^{T'})$ verifies successfully).

The above implies that the probability over possible interactions that the outputs of h and h' are different—which, as we have argued above, is equal to the statistical distance between the distributions $\mathsf{Out}^{\mathcal{T}_h}_{\Pi_h, \mathcal{A}', S}(1^n, x, y, z)$ and $\mathsf{Out}^{\mathcal{T}_{h'}}_{\Pi_{h'}, \mathcal{A}', S}(1^n, x, y, z)$—is no greater[4] than the probability with which $\mathcal{A}_{\mathsf{Del}}$'s output is accepted. In particular, by our assumption that, for infinitely many $n \in \mathbb{N}$, there were x, y, z such that this statistical distance was greater than $1/p(n)$, this implies that the probability that $\mathcal{A}_{\mathsf{Del}}$'s output is accepted (for the corresponding inputs) must be greater than $1/p(n)$ for infinitely many $n \in \mathbb{N}$. But this contradicts the soundness of delegation, so the claim is proven. □

Now let $\Pi'_{h'}$ be identical to $\Pi_{h'}$, with the sole exception that the receiver's first-round message to the sender replaces the correctly generated $\mathsf{ct}_x = \mathsf{Enc}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, x; r_{\mathsf{Enc}(x)})$ with the corresponding encryption $\mathsf{ct}_x = \mathsf{Enc}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, 0; r_{\mathsf{Enc}(x)})$ of 0. We present the following claim comparing $\mathsf{Exec}^{\mathcal{T}_{h'}}_{\Pi_{h'}, \mathcal{A}', S}(1^n, x, y, z)$ and $\mathsf{Exec}^{\mathcal{T}_{h'}}_{\Pi'_{h'}, \mathcal{A}', S}(1^n, x, y, z)$:

Claim 21. For any polynomial-time non-uniform distinguisher D, there exists negligible $\epsilon(\cdot)$ such that, for any $n \in \mathbb{N}$ and inputs x, y, z, the distributions $\mathsf{Out}^{\mathcal{T}_{h'}}_{\Pi_{h'}, \mathcal{A}', S}(1^n, x, y, z)$ and $\mathsf{Out}^{\mathcal{T}_{h'}}_{\Pi'_{h'}, \mathcal{A}', S}(1^n, x, y, z)$ cannot be distinguished by D with probability greater than $\epsilon(n)$.

[4]Note that equality is not guaranteed, as h could possibly accept a ciphertext $\mathsf{ct}'_{\mathsf{out}} \neq \mathsf{ct}_{\mathsf{out}}$ that still decrypts to $f(x, y)$.

Proof. Intuitively, this follows from the CPA-security of the FHE scheme with respect to $T(n) \cdot \mathsf{poly}(n)$-time adversaries and the fact that both h' and the view of $\mathcal{A}'$ are independent of r_{FHE} and $\mathsf{sk}_{\mathsf{FHE}}$.

Formally, assume for contradiction that there exists a non-uniform polynomial-time distinguisher D and polynomial $p(\cdot)$ such that, for infinitely many $n \in \mathbb{N}$, there are inputs x, y, z such that D is able to distinguish $\mathsf{Out}^{\mathcal{T}_{h'}}_{\Pi_{h'},\mathcal{A}',S}(1^n, x, y, z)$ and $\mathsf{Out}^{\mathcal{T}_{h'}}_{\Pi'_{h'},\mathcal{A}',S}(1^n, x, y, z)$ with probability $1/p(n)$. We define a tuple of $T(n) \cdot \mathsf{poly}(n)$-time algorithms $(\mathcal{A}_{\mathsf{FHE}}, D')$ that can break the CPA-security of the FHE scheme $(\mathsf{Gen}_{\mathsf{FHE}}, \mathsf{Enc}_{\mathsf{FHE}}, \mathsf{Eval}_{\mathsf{FHE}}, \mathsf{Dec}_{\mathsf{FHE}})$ with probability $1/p(n)$ as follows:

- $\mathcal{A}_{\mathsf{FHE}}$, on input 1^n, outputs $(0, x)$.
- D', on input $(1^n, \mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_x)$, where c is either $\mathsf{ct}^0_x = \mathsf{Enc}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, 0)$ or $\mathsf{ct}^1_x = \mathsf{Enc}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, x)$, does the following:
  1. Generate $r_{\mathsf{Del}} \leftarrow \{0,1\}^*$ and $(\mathsf{pk}_{\mathsf{Del}}, \mathsf{sk}_{\mathsf{Del}}) = \mathsf{Gen}_{\mathsf{Del}}(1^\lambda; r_{\mathsf{Del}})$.
  2. Run the corrupted sender $\mathcal{A}'$ with sender input y, auxiliary input z, and first-round message $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x)$. $\mathcal{A}'$ will generate a message $(y', \mathsf{pk}'_{\mathsf{FHE}}, \mathsf{pk}'_{\mathsf{Del}}, \mathsf{ct}'_x, \mathsf{ct}'_y, \mathsf{ct}'_{\mathsf{out}}, \pi'_{\mathsf{Del}}, r'_{\mathsf{Enc}(y)}, T')$ to send to $\mathcal{T}_{h'}$ and output $\mathsf{out}_{\mathcal{A}'}$. Store $\mathsf{out}_{\mathcal{A}'}$.
  3. Verify the following and set $\mathsf{out}_R = \perp$ if any are false. Otherwise, set $\mathsf{out}_R = f(x, y')$.
     (a) $\mathsf{pk}_{\mathsf{FHE}} = \mathsf{pk}'_{\mathsf{FHE}}$, $\mathsf{pk}_{\mathsf{Del}} = \mathsf{pk}'_{\mathsf{Del}}$, $\mathsf{ct}_x = \mathsf{ct}'_x$
     (b) $\mathsf{ct}'_y = \mathsf{Enc}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, y'; r'_{\mathsf{Enc}(y)})$
     (c) $\mathsf{Ver}_{\mathsf{Del}}(\mathsf{sk}_{\mathsf{Del}}, g, \mathsf{ct}_x, \mathsf{ct}'_y, \mathsf{ct}'_{\mathsf{out}}, \pi'_{\mathsf{Del}}, T') = \mathsf{Accept}$ for the functionality $g(c_1, c_2) = \mathsf{Eval}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, f, c_1, c_2)$

4. Return $D(1^n, (\mathsf{out}_{\mathcal{A}'}, \mathsf{out}_R))$.

First, notice that (given that the inputs $\mathsf{pk}_{\mathsf{FHE}}$ and $\mathsf{ct}_x = \mathsf{Enc}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, m)$ for either $m = 0$ or $m = x$ are generated correctly) the inputs to $\mathcal{A}'$ in step (2) of D' are identically distributed to either the inputs in $\mathsf{Exec}^{\mathcal{T}_{h'}}_{\Pi_{h'},\mathcal{A}',S}(1^n, x, y, z)$ (if $m = x$) or the inputs in $\mathsf{Exec}^{\mathcal{T}_{h'}}_{\Pi'_{h'},\mathcal{A}',S}(1^n, x, y, z)$ (if $m = 0$). Hence, the view of $\mathcal{A}'$ in D' is identically distributed to the corresponding view in the respective experiment, which implies that the output $\mathsf{out}_{\mathcal{A}'}$ must be as well, as must the message sent to $\mathcal{T}_{h'}$.

It remains to argue about the receiver's output out_R; recall that the honest receiver's output in either experiment is given by the output of the ideal functionality $\mathcal{T}_{h'}$. However, out_R as defined in step (3) of D' can easily be seen to be identically distributed to the output of h' in the respective experiment $\mathsf{Exec}^{\mathcal{T}_{h'}}_{\Pi_{h'},\mathcal{A}',S}(1^n, x, y, z)$ (if $m = x$) or $\mathsf{Exec}^{\mathcal{T}_{h'}}_{\Pi'_{h'},\mathcal{A}',S}(1^n, x, y, z)$ (if $m = 0$). This holds because, since R is honest, R's inputs $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x)$ are honestly generated and so the verifications in steps (3a) and (3b) are identical to the respective checks in step (1) of h. Furthermore, the verification in step (3c) of D' is identical to the verification in step (3) of h, so it follows that $\mathsf{out}_R = \perp$ exactly when h' in the respective experiment would return $\perp$, and that, otherwise, $\mathsf{out}_R = f(x, y')$, which by the definition of h' is identical to what h' would return if not outputting $\perp$.

So we have argued that the distribution $(\mathsf{out}_{\mathcal{A}'}, \mathsf{out}_R)$ is identical to $\mathsf{Out}^{\mathcal{T}_{h'}}_{\Pi_{h'},\mathcal{A}',S}(1^n, x, y, z)$ when $m = x$ and to $\mathsf{Out}^{\mathcal{T}_{h'}}_{\Pi'_{h'},\mathcal{A}',S}(1^n, x, y, z)$ when $m = 0$. But we have assumed that for infinitely many $n \in \mathbb{N}$ there exist x, y, z so that D can distinguish $\mathsf{Out}^{\mathcal{T}_{h'}}_{\Pi_{h'},\mathcal{A}',S}(1^n, x, y, z)$ and $\mathsf{Out}^{\mathcal{T}_{h'}}_{\Pi'_{h'},\mathcal{A}',S}(1^n, x, y, z)$ with probability $1/p(n)$, i.e., that there is at least a $1/p(n)$ difference between the probability that $D(1^n, (\mathsf{out}_{\mathcal{A}'}, \mathsf{out}_R))$ returns 1 in the $m = x$ case and the respective probability in the $m = 0$ case. But, since D' returns precisely $D(1^n, (\mathsf{out}_{\mathcal{A}'}, \mathsf{out}_R))$,

this gives us

$$|\Pr[D(1^n, \mathsf{pk}_{\mathsf{FHE}}, \mathsf{Enc}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, 0)) = 1] - \Pr[D(1^n, \mathsf{pk}_{\mathsf{FHE}}, \mathsf{Enc}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, x)) = 1]| \geq 1/p(n)$$

which, since $\mathcal{A}_{\mathsf{FHE}}$ always returns $(0, x)$, means that $(\mathcal{A}_{\mathsf{FHE}}, D')$ is able to break the CPA-security of the underlying FHE scheme (w.r.t. $T(n) \cdot \mathsf{poly}(n)$-time adversaries) with probability $1/p(n)$ for infinitely many $n \in \mathbb{N}$, a contradiction. □

It remains to compare $\mathsf{Out}^{\mathcal{T}_{h'}}_{\Pi'_{h'}, \mathcal{A}', S}(1^n, x, y, z)$ and $\mathsf{Out}^{\mathcal{T}_f}_{\Pi_f, \mathcal{S}_S, S}(1^n, x, y, z)$; however, we claim that in fact these distributions are already identical. First, observe that the input provided to $\mathcal{A}'$ in $\mathcal{S}_S$ is identically distributed to the input provided to $\mathcal{A}'$ in $\mathsf{Exec}^{\mathcal{T}_{h'}}_{\Pi'_{h'}, \mathcal{A}', S}(1^n, x, y, z)$; in both cases this consists of an honestly generated $\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x$ such that ct_x is the respective encryption of 0. So it follows that the adversary's output, as well as the message sent by the adversary to the ideal functionality, must be identically distributed between the two experiments. Demonstrating that the receiver's outputs are identical—that is, that the output of h' in $\mathsf{Exec}^{\mathcal{T}_{h'}}_{\Pi'_{h'}, \mathcal{A}', S}(1^n, x, y, z)$ is always equal to the output $f(x, y)$ in $\mathsf{Exec}^{\mathcal{T}_f}_{\Pi_f, \mathcal{S}_S, S}(1^n, x, y, z)$—will follow from the following claim, to which we have already alluded in the previous two reductions:

Claim 22. If R is honest, then, given messages $(x, \mathsf{pk}_{\mathsf{FHE}}, r_{\mathsf{Del}}, r_{\mathsf{Enc}})$ sent to $\mathcal{T}_{h'}$, $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x)$ sent to $\mathcal{A}'$, and $(y', \mathsf{pk}'_{\mathsf{FHE}}, \mathsf{pk}'_{\mathsf{Del}}, \mathsf{ct}'_x, \mathsf{ct}'_y, \mathsf{ct}'_{\mathsf{out}}, \pi'_{\mathsf{Del}}, r'_{\mathsf{Enc}(y)}, T')$ sent by $\mathcal{A}'$ to $\mathcal{T}_{h'}$, the checks in step (2) of $\mathcal{S}_S$ succeed if and only if all checks in steps (1) and (3) of the functionality h' succeed.

Proof. If R is honest, it must be the case that $(\mathsf{pk}_{\mathsf{Del}}, \cdot) = \mathsf{Gen}_{\mathsf{Del}}(1^\lambda; r_{\mathsf{Del}})$ and $\mathsf{ct}_x = \mathsf{Enc}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, x; r_{\mathsf{Enc}(x)})$; hence step (2a) of $\mathcal{S}_S$ is equivalent to verifying $\mathsf{pk}'_{\mathsf{FHE}} = \mathsf{pk}_{\mathsf{FHE}}$, $(\mathsf{pk}'_{\mathsf{Del}}, \cdot) = \mathsf{Gen}_{\mathsf{Del}}(1^\lambda; r_{\mathsf{Del}})$, and $\mathsf{ct}'_x = \mathsf{Enc}_{\mathsf{FHE}}(\mathsf{pk}'_{\mathsf{FHE}}, x; r_{\mathsf{Enc}(x)})$, i.e., the first three checks of

step (1) of h'. Step (2b) is trivially equivalent to the last check in step (1) of h' and step (2c) is trivially equivalent to the check in step (3) of h', completing the argument. $\square$

This implies that the receiver in $\mathsf{Exec}^{\mathcal{T}_{h'}}_{\Pi'_{h'},\mathcal{A}',S}(1^n,x,y,z)$ will return $\perp$ as the output from h' precisely when $\mathcal{S}_S$ will return $\perp$ to the ideal functionality (based on the checks in step (2)) and cause the receiver in $\mathsf{Exec}^{\mathcal{T}_f}_{\Pi_f,\mathcal{S}_S,S}(1^n,x,y,z)$ to return $\perp$. However, when $\mathcal{T}_f$ does not output $\perp$, it will always output $f(x,y')$ on the respective inputs x from the honest receiver and y' from $\mathcal{S}_S$; similarly, when $\mathcal{T}_{h'}$ does not return $\perp$, it will, by definition, *also* always output $f(x,y')$ on the respective input x from the honest receiver and y' from $\mathcal{A}'$. The above, then, is sufficient to conclude that the distributions $\mathsf{Out}^{\mathcal{T}_{h'}}_{\Pi'_{h'},\mathcal{A}',S}(1^n,x,y,z)$ and $\mathsf{Out}^{\mathcal{T}_f}_{\Pi_f,\mathcal{S}_S,S}(1^n,x,y,z)$ are identical.

We conclude the proof of the lemma with a standard hybrid argument; specifically, if there exists some non-uniform polynomial-time distinguisher D and polynomial $p(\cdot)$ such that, for infinitely many $n\in\mathbb{N}$, there are inputs x,y,z so that D can distinguish $\mathsf{Out}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',S}(1^n,x,y,z)$ and $\mathsf{Out}^{\mathcal{T}_f}_{\Pi_f,\mathcal{S},S}(1^n,x,y,z)$ with probability $1/p(n)$, then D must likewise be able to distinguish one of the following pairs with probability $1/p'(n)$ for some polynomial $p'(\cdot)$:

- $\mathsf{Out}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',S}(1^n,x,y,z)$ and $\mathsf{Out}^{\mathcal{T}_{h'}}_{\Pi_{h'},\mathcal{A}',S}(1^n,x,y,z)$
- $\mathsf{Out}^{\mathcal{T}_{h'}}_{\Pi_{h'},\mathcal{A}',S}(1^n,x,y,z)$ and $\mathsf{Out}^{\mathcal{T}_{h'}}_{\Pi'_{h'},\mathcal{A}',S}(1^n,x,y,z)$
- $\mathsf{Out}^{\mathcal{T}_{h'}}_{\Pi'_{h'},\mathcal{A}',S}(1^n,x,y,z)$ and $\mathsf{Out}^{\mathcal{T}_f}_{\Pi_f,\mathcal{S},S}(1^n,x,y,z)$

The first case would contradict Claim 20, the second case would contradict Claim 21, and the third case is impossible because we showed the distributions to be identical. Therefore, such a distinguisher D cannot exist. $\square$

By the same logic, a standard hybrid argument shows that Lemmas 10 and 11 imply

Theorem 7: if there were some non-uniform polynomial-time distinguisher D and polynomial $p(\cdot)$ such that, for infinitely many $n \in \mathbb{N}$, there were inputs x, y, z so that D could distinguish $\mathsf{Out}_{\Pi,\mathcal{A},I}(1^n, x, y, z)$ and $\mathsf{Out}^{\mathcal{T}_f}_{\Pi_f,\mathcal{S},I}(1^n, x, y, z)$ with probability $1/p(n)$, then D would be able to distinguish either:

- $\mathsf{Out}_{\Pi,\mathcal{A},I}(1^n, x, y, z)$ and $\mathsf{Out}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',I}(1^n, x, y, z)$, or
- $\mathsf{Out}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',I}(1^n, x, y, z)$ and $\mathsf{Out}^{\mathcal{T}_f}_{\Pi_f,\mathcal{S},I}(1^n, x, y, z)$

with probability $1/p'(n)$ for some polynomial $p'(\cdot)$. The first case would contradict Lemma 10 and the second Lemma 11; hence, Theorem 7 is proven.

3.4 Additional Definitions

3.4.1 Non-Interactive Secure Computation, Continued

We start by defining a stronger notion of non-interactive secure computation, which is the definition achieved in the second feasibility result we present. Recall the definition of non-interactive two-party computation given in Definition 10 of Section 3.2; we begin with this definition and the additional restriction of **perfect correctness** (i.e., that correctness should hold with no error rather than a negligible $\epsilon(n)$ error).

Externalized Universally Composable Security. To define the notion of security proven in our main theorem, we use the framework of *universally composable security* [33, 34], extended to include access to superpolynomial "helper functionalities" [35, 38]. Specifically,

we prove UC security in the presence of an external helper which allows the adversary to break the commitments of corrupted parties.

Model of execution. We recall the discussion of UC security with external helper functionalities provided in [38]. Consider parties represented by polynomial-time interactive Turing machines [67]; the model contains a number of parties running instances of the protocol Π, as well as an *adversary* $\mathcal{A}$ and an *environment* $\mathcal{Z}$. The environment begins by invoking the adversary on an arbitrary input, and afterwards can proceed by invoking parties which participate in single instances of the protocol Π by providing them with their respective inputs, as well as a *session identifier* (which is unique for each instance of the protocol Π) and a *party identifier* (which is unique among the participants in each session). The environment can furthermore read the output of any party involved in some execution of Π, as well as any output provided by the adversary.

For the purposes of UC security, we will restrict our attention to environments which may only invoke a single session of the protocol Π—that is, any instances invoked must have the same session identifier. Concurrent and composable security (i.e., against more generalized environments) will follow from this via a *universal composition theorem*, which we will state later in this section.

The adversary, on the other hand, is able to control all communication between the various parties involved in executions of Π, and to furthermore modify the outputs of certain *corrupted* parties (which we here assume are decided non-adaptively, i.e., every party is either invoked as permanently corrupted or permanently uncorrupted). Uncorrupted parties will always act according to the protocol Π, and we assume that the adversary only delivers messages from uncorrupted parties that were actually intended to be sent (i.e., authenticated

communication); the adversary can, on the other hand, deliver any message on behalf of a corrupted party. The adversary can also send messages to and receive them from the environment at any point.

We will furthermore assume a notion of security using an "imaginary angel" [115], which can be formalized in the *externalized UC* (EUC) setting [35]; both the corrupted parties and environment will have access to an external *helper functionality* $\mathcal{H}$, also defined as an interactive Turing machine—unlike the participants, adversary, or environment, however, $\mathcal{H}$ is not restricted to polynomial running time. $\mathcal{H}$ is persistent throughout the execution and is invoked by the environment immediately after the adversary is; furthermore, $\mathcal{H}$ must be immediately informed of the identity of all corrupted parties when parties are determined by the environment to be corrupted.

Finally, while honest players can only be invoked on a single session identifier, we allow the adversary to invoke $\mathcal{H}$ on behalf of corrupt parties using potentially *arbitrary* session identifiers; this is needed to prove the composition theorem.

The execution ends when the environment halts, and we assume the output to be the output of the environment. We let $\mathsf{Exec}_{\Pi,\mathcal{A},\mathcal{Z}}(1^n, z)$ denote the distribution of the environment's output, taken over the random tape given to $\mathcal{A}$, $\mathcal{Z}$, and all participants, in the execution above (with a single session of Π), where the environment originally gets as input security parameter 1^n and auxiliary input z. We say that Π *securely emulates* some other protocol Π' if, for any adversary $\mathcal{A}$, there exists a simulator $\mathcal{S}$ such that the environment $\mathcal{Z}$ is unable to tell the difference between the execution of Π with $\mathcal{A}$ and the execution of Π' with $\mathcal{S}$—that is, intuitively, the environment gains the same information in each of the two executions. Formally:

Definition 16 (based on [38])**.** For some (superpolynomial-time) interactive Turing machine

$\mathcal{H}$, we say a protocol Π **$\mathcal{H}$-EUC-emulates** some protocol Π' if, for any polynomial-time adversary $\mathcal{A}$, there exists some simulated polynomial-time adversary $\mathcal{S}$ such that, for any non-uniform polynomial-time environment $\mathcal{Z}$ and polynomial-time distinguisher D, there exists negligible $\nu(\cdot)$ such that, for any $n \in \mathbb{N}$ and $z \in \{0,1\}^*$:

$$|\Pr\left[D(\mathsf{Exec}_{\Pi,\mathcal{A},\mathcal{Z}}(1^n, z)) = 1\right] - \Pr\left[D(\mathsf{Exec}_{\Pi',\mathcal{S},\mathcal{Z}}(1^n, z)) = 1\right]| \leq \nu(n)$$

To prove that a protocol Π *securely realizes* an ideal functionality $\mathcal{T}$, we wish to show that it securely emulates an "ideal" protocol $\Pi(\mathcal{T})$ in which all parties send their respective inputs to an instance of $\mathcal{T}$ with the same session identifier and receive the respective output; note that the adversary does not receive the messages to or from each instance of $\mathcal{T}$.

Definition 17 (based on [38])**.** For some (superpolynomial-time) interactive Turing machine $\mathcal{H}$, we say a protocol Π **$\mathcal{H}$-EUC-realizes** some functionality $\mathcal{T}$ if it $\mathcal{H}$-EUC-emulates the protocol $\Pi(\mathcal{T})$ given above.

In the case of two-party computation for functionality f, $\mathcal{T}$ will simply receive inputs x from the receiver and y from the sender and return $f(x, y)$ to the receiver:

Definition 18. For some (superpolynomial-time) interactive Turing machine $\mathcal{H}$, we refer to a non-interactive two-party computation protocol Π for some functionality $f(\cdot, \cdot)$ as **$\mathcal{H}$-EUC-secure** if it $\mathcal{H}$-EUC-realizes the functionality $\mathcal{T}_f$, which, on input x from a receiver R and input y from a sender S, returns $f(x, y)$ to R.

Remarks. Notice that, since $\mathcal{Z}$'s output is a (randomized) function of its view, it suffices to show that $\mathcal{Z}$'s view cannot be distinguished by any polynomial-time distinguisher D between the respective experiments. We can also without loss of generality assume that the environment $\mathcal{Z}$ in the real execution effectively runs the adversary $\mathcal{A}$ internally and forwards

all of $\mathcal{A}$'s messages to and from other parties by using a "dummy adversary" $\mathcal{D}$ which simply forwards communication from $\mathcal{Z}$ to the respective party. This allows us to effectively view the environment $\mathcal{Z}$ and adversary $\mathcal{A}$ as a single entity.

Furthermore, observe that we use a *polynomial-time* simulator $\mathcal{S}$ in our definition of security. [63] shows that two-round secure computation protocols cannot be proven secure with standard polynomial-time simulation; hence, many protocols are proven secure using superpolynomial-time simulators (a technique originally proposed by [109, 115]). Indeed, we note that, if $\mathcal{H}$ runs in time $T(\cdot)$, then a protocol that $\mathcal{H}$-EUC-realizes some functionality $\mathcal{T}$ with polynomial-time simulation will also UC-realize $\mathcal{T}$ with $\mathsf{poly}(T(\cdot))$-time simulation; hence, in a way, the simulator $\mathcal{S}$ we propose in our security definition can still be considered to do a superpolynomial-time amount of "work".

Universal composition. The chief advantage of the UC security paradigm is the notion of *universal composition*; intuitively, if a protocol ρ UC-realizes (or, respectively, $\mathcal{H}$-EUC-realizes) an ideal functionality $\mathcal{T}$, then it is "composable" in the sense that any protocol that uses the functionality $\mathcal{T}$ as a primitive derives the same security guarantees from the protocol ρ as they would the ideal functionality.

More formally, given an ideal functionality $\mathcal{T}$, let us define a *$\mathcal{T}$-hybrid protocol* as one where the participating parties have access to an unbounded number of copies of the functionality $\mathcal{T}$ and may communicate directly with these copies as in an "ideal" execution (i.e., without communication being intercepted by the adversary). Each copy of $\mathcal{T}$ will have a unique session identifier, and their inputs and outputs are required to contain the respective identifier.

Then, if Π is a $\mathcal{T}$-hybrid protocol, and ρ is a protocol which realizes $\mathcal{T}$, then we can

define a *composed protocol* Π^ρ by modifying Π so that the first message sent to $\mathcal{T}$ is instead an invocation of a new instance of ρ with the same session identifier and the respective message as input, and so that further messages are likewise relayed to the same instance of ρ instead, again with their contents as the respective input. Any output from an instance of ρ is substituted for the respective output of the corresponding instance of $\mathcal{T}$. The following powerful theorem, then, states the notion of composability intuitively described above.

Theorem 8 (Relativized Universal Composition [33, 38])**.** For some ideal functionality $\mathcal{T}$ and helper functionality $\mathcal{H}$, if Π is a $\mathcal{T}$-hybrid protocol, and ρ is a protocol that $\mathcal{H}$-EUC-realizes $\mathcal{T}$, then Π^ρ $\mathcal{H}$-EUC-emulates Π.

Stand-alone Security. As one of the key building blocks of our UC-secure protocol, we use a non-interactive secure computation protocol which satisfies the strictly weaker notion of *stand-alone security with superpolynomial-time simulation* as given in Definition 11 of Section 3.2, and recall Theorem 3 which demonstrates that stand-alone secure NISC protocols with quasi-polynomial simulation exist assuming the existence of a notion of "weak OT", which in turn can be based on subexponential versions of standard assumptions [10, 26].

3.4.2 SPS-ZK Arguments

We proceed to recalling the definition of interactive arguments.

Definition 19 ([31, 67, 60])**.** We refer to an interactive protocol (P, V) between a probabilistic *prover* P and a *verifier* V as an **interactive argument** for some language $\mathcal{L} \subseteq \{0,1\}^*$ if the following conditions hold:

1. **Completeness.** There exists a negligible function $\nu(\cdot)$ such that, for any $x \in \mathcal{L}$:

$$\Pr\left[\langle P, V\rangle(x) = \mathsf{Accept}\right] \geq 1 - \nu(|x|)$$

2. $T(\cdot)$**-time soundness.** For *any* non-uniform probabilistic $T(\cdot)$-time prover P^* (not necessarily honest), there exists a negligible function $\nu(\cdot)$ such that, for any $x \notin \mathcal{L}$:

$$\Pr\left[\langle P^*, V\rangle(x) = \mathsf{Accept}\right] \leq \nu(|x|)$$

   Furthermore, if the above holds even if the statement $x \notin \mathcal{L}$ can be adaptively chosen by the cheating prover anytime prior to sending its last message, we call such a protocol ($T(\cdot)$-time) **adaptively sound**.

We also require a notion of *zero-knowledge* [67] with superpolynomial simulation (SPS-ZK) [109], which states that the prover's witness w should be "hidden" from the verifier in the sense that proofs of a particular statement $x \in L$ should be simulatable in a manner independent of w:

Definition 20 ([109])**.** We refer to an interactive argument for some NP language L (with witness relation R_L) as $T'(\cdot)$**-time simulatable zero-knowledge with** $T(\cdot)$**-time security** (or $(T(\cdot), T'(\cdot))$-simulatable zero-knowledge) if, for any $T(\cdot)$-time cheating verifier V^* (which can output an arbitrary function of its view), there exists a $T'(\cdot)$-time simulator Sim and negligible function $\nu(\cdot)$ such that, for any $T(\cdot)$-time non-uniform distinguisher D, given any statement $x \in L$, any witness $w \in R_L(x)$, and any auxiliary input $z \in \{0,1\}^*$, it holds that:

$$\left|\Pr\left[D(x, \langle P(w), V^*(z)\rangle(x)) = 1\right] - \Pr\left[D(x, \mathsf{Sim}(x, z)) = 1\right]\right| \leq \nu(|x|)$$

Our construction will use a two-round adaptively sound zero-knowledge argument consisting of three polynomial-time algorithms, $(\mathsf{ZK}_1, \mathsf{ZK}_2, \mathsf{ZK}_3)$, defining the following interaction $\langle P, V\rangle$:

- V runs $(\mathsf{zk}_1, \sigma) \leftarrow \mathsf{ZK}_1(1^n)$, which takes as input the security parameter n and generates a first message zk_1 and persistent state σ.
- P runs $\mathsf{zk}_2 \leftarrow \mathsf{ZK}_2(\mathsf{wi}_1, x, w)$, which takes as input the first message wi_1, a statement x, and a witness w, and returns a second message zk_2.
- V runs $\{\mathsf{Accept}, \mathsf{Reject}\} \leftarrow \mathsf{ZK}_3(\mathsf{zk}_2, x, \sigma)$, which takes as input a second message zk_2, a statement x, and the persistent state σ, and returns Accept if zk_2 contains an accepting proof that $x \in L$ and Reject otherwise.

We observe that, in fact, this primitive is implied by the existence of a *stand-alone secure NISC* (see Definition 11).

Theorem 9. For any constants $c < c'$, letting subexponential functions $T(n) = n^{log^c(n)}$ and $T'(n) = n^{log^{c'}(n)}$, then, if there exists a subexponentially stand-alone secure non-interactive two-party computation protocol for any polynomial-time Turing-computable functionality $f(\cdot, \cdot)$ with $T(\cdot)$-time security and $T'(\cdot)$-time simulation, then there exists a two-round interactive argument with $T(\cdot)$-time adaptive soundness and $(T(\cdot), T'(\cdot))$-simulatable zero-knowledge.

The construction and its proof of security is straightforward, but the interested reader may find it in the full version of this work [99].

3.4.3 Non-Interactive CCA-secure Commitments

Our construction will rely on non-interactive (single-message) tag-based commitment schemes satisfying the notion of CCA security [108, 38, 88].

Definition 21 (based on [88])**.** A **non-interactive tag-based commitment scheme** (with $t(\cdot)$-bit tags) consists of a pair of polynomial-time algorithms $(\mathsf{Com}, \mathsf{Open})$ such that:

- $c \leftarrow \mathsf{Com}(1^n, \mathsf{id}, v; r)$ (alternately denoted $\mathsf{Com}_{\mathsf{id}}(1^n, v; r)$) takes as input an identifier (tag) $\mathsf{id} \in \{0,1\}^{t(n)}$, a value v, randomness r, and a security parameter n, and outputs a commitment c. We assume without loss of generality that the commitment c includes the respective tag id.
- $\{\mathsf{Accept}, \mathsf{Reject}\} \leftarrow \mathsf{Open}(c, v, r)$ takes as input a commitment c, a value v, and randomness r, and returns either Accept (if c is a valid commitment for v under randomness r) or Reject (if not).

We consider commitment schemes having the following properties:

1. **Correctness:** For any security parameter $n \in \mathbb{N}$, any $v, r \in \{0,1\}^*$, and any $\mathsf{id} \in \{0,1\}^{t(n)}$:
$$\Pr[c \leftarrow \mathsf{Com}(1^n, \mathsf{id}, v; r) : \mathsf{Open}(c, v, r) = \mathsf{Accept}] = 1$$
2. **Perfect binding:** For any commitment string c, values v, v', and randomness r, r', if it is true that $\mathsf{Open}(c, v, r) = 1$ and $\mathsf{Open}(c, v', r') = 1$, then $v = v'$.
3. **$T(\cdot)$-time hiding:** For any $T(\cdot)$-time non-uniform distinguisher D and fixed polynomial $p(\cdot)$, there exists a negligible function $\nu(\cdot)$ such that, for any $n \in \mathbb{N}$, any $\mathsf{id} \in \{0,1\}^{t(n)}$ and any values $v, v' \in \{0,1\}^{p(n)}$:
$$|\Pr[D(\mathsf{Com}(1^n, \mathsf{id}, v)) = 1] - \Pr[D(\mathsf{Com}(1^n, \mathsf{id}, v')) = 1]| \leq \nu(n)$$

For our construction, we require a strictly stronger property than just hiding: hiding should hold even against an adversary with access to a "decommitment oracle". This property is known as *CCA security* due to its similarity to the analogous notion for encryption

schemes [117]. We introduce a weakening of CCA security, to which we shall refer as "weak CCA security", which is nonetheless sufficient for our proof of security, and, as we shall prove in Section 3.6, is *necessary* for our proof of security as well. We define this as follows:

Definition 22. Let $\mathcal{O}^*$ be an oracle which, given a commitment c, returns a valid committed value v—that is, such that there exists some randomness r for which $\mathsf{Open}(c, v, r) = \mathsf{Accept}$.

A tag-based commitment scheme $(\mathsf{Com}, \mathsf{Open})$ is $T(\cdot)$**-time weakly CCA-secure** with respect to $\mathcal{O}^*$ if, for any polynomial-time adversary $\mathcal{A}$, letting $\mathsf{Exp}_b(\mathcal{O}^*, \mathcal{A}, n, z)$ (for $b \in \{0,1\}$) denote $\mathcal{A}$'s output in the following interactive experiment:

- $\mathcal{A}$, on input $(1^n, z)$, is given oracle access to $\mathcal{O}^*$, and adaptively chooses values v_0, v_1 and tag id.
- $\mathcal{A}$ receives $\mathsf{Com}(1^n, \mathsf{id}, v_b)$ and returns an arbitrary output; however, $\mathcal{A}$'s output is replaced with $\perp$ if $\mathcal{O}^*$ was ever queried on any commitment c with tag id.

then, for any $T(\cdot)$-time distinguisher D, there exists negligible $\nu(\cdot)$ such that, for any $n \in \mathbb{N}$ and any $z \in \{0,1\}^*$, it holds that:

$$|\Pr[D(\mathsf{Exp}_0(\mathcal{O}^*, \mathcal{A}, n, z)) = 1] - \Pr[D(\mathsf{Exp}_1(\mathcal{O}^*, \mathcal{A}, n, z)) = 1]| \leq \nu(n)$$

We remark that the only difference from the "standard" notion of CCA security is that the CCA oracle, given a commitment c, rather than returning both the value v committed to and the randomness r used in the commitment, instead returns just the value v. This is similar to the definition of CCA security commonly used for encryption schemes [117].

3.5 Concurrently Composable Non-Interactive Secure Computation

In this section, we state our final result, which constructively proves the feasibility of externalized UC-secure non-interactive two-party computation, and describe and prove the respective protocol. As before, we lead with a technical overview before the formal theorem statement and protocol description.

3.5.1 Technical Overview

We will demonstrate a strong and composable notion of concurrent security using the *externalized UC model* [33, 35], where we assume the adversary, the environment, and the simulator are strictly *polynomial-time* but have access to an "imaginary angel", or a global "helper" entity $\mathcal{H}$ that implements some superpolynomial-time functionality. (This notion was first considered in [115] for the case of non-interactive, stateless, angels) In our case (as in [38]) $\mathcal{H}$ will implement the CCA decommitment oracle $\mathcal{O}$ for a CCA secure commitment; while interacting with a party P, $\mathcal{H}$ will send a valid decommitment in response to any commitments made using that party's identity as the tag. (Since the adversary controls corrupted parties, this effectively means that $\mathcal{H}$ will decommit any commitments with a corrupted party's identifier, but none with an honest party's identifier). CCA security guarantees, then, that an adversary will never be able to break an honest party's commitment; on the other hand, the presence of the helper $\mathcal{H}$ makes it relatively easy for the simulator $\mathcal{S}$ we construct for the definition of UC security to extract information necessary for simulation from corrupted parties' commitments.

Aside from the commitment scheme, our protocol consists of two major sub-components. First, in order to evaluate the functionality $f(x, y)$, we begin with a NISC protocol that satisfies *stand-alone* security with superpolynomial-time simulation. In order to build this into a protocol satisfying full UC security, however, we will need to leverage the CCA-secure commitment scheme in order to allow the simulator to extract the malicious party's input from their message; since the simulator is restricted to polynomial time (with access to the CCA helper $\mathcal{H}$), this cannot be done by simply leveraging the superpolynomial-time simulator of the underlying NISC. Instead, if both parties commit to their respective inputs and send the commitments alongside the messages of the underlying NISC, the simulator can easily use the CCA helper to extract the inputs from the commitments. This, however, presents another issue: namely, there must be a way to verify that a potentially malicious party commits to the correct input (i.e., the same one they provided to the NISC). For the case of a corrupted sender, this will require the other major component of our protocol: a 2-round zero-knowledge (ZK) interactive argument with SPS security; unsurprisingly, we remark that an appropriate such SPS-ZK protocol can be obtained from an SPS-NISC.

Towards intuitively describing our protocol, we now briefly describe how we deal with extracting from a malicious receiver and sender before presenting the complete protocol.

Dealing with a malicious receiver: using "interactive witness encryption". As suggested above, the first step towards extracting a malicious receiver's input x is to have the receiver commit to their input x and send the commitment c_x with their first-round message. This way, when the receiver is corrupted, the simulator can extract x using the decommitment helper $\mathcal{H}$. Of course, we require a way to verify that the commitment sent by the receiver is indeed a commitment to the correct value of x (i.e., the same as the receiver's input to the NISC which computes $f(x, y)$). We deal with this using a technique reminiscent of the

recent non-concurrent NISC protocol of [100], by using the underlying NISC to implement an "interactive witness encryption scheme".[5] The receiver will, in addition to their input x for f, input the randomness r_x used to generate the commitment c_x, as well as the corresponding decommitment d_x, to the NISC; the sender will input c_x in addition to y, and the NISC will return $f(x, y)$ *if and only if* (c_x, d_x) is a valid commitment of x using randomness r_x. Hence, if the receiver sends an invalid commitment to x to the sender, they receive $\perp$ from the NISC instead of the correct output; otherwise, if it is valid, the simulator can always extract the correct value of x from the commitment using $\mathcal{H}$.

Dealing with a malicious sender: using a "two-track" functionality For the case where the sender is corrupted, we begin by having the sender produce a commitment c_y to their input y and send it along with their second message. Unlike with c_x, however, we cannot verify c_y within the underlying NISC protocol, since the receiver does not receive the commitment c_y until after they have given their inputs to the NISC (and hence cannot provide it as an input to verify that the commitment they received from the sender is correct). Instead, we rely on the SPS-ZK argument, which the sender uses to prove that c_y is a valid commitment to the same input y used to produce their NISC message. However, this in turn creates issues for simulatability in the corrupted-receiver case, since the simulator does not know y and cannot simulate the underlying NISC or the ZK argument in only polynomial time.

To deal with this, we switch to what is effectively a *two-track* functionality for the underlying NISC and ZK argument. We add a trapdoor t, chosen at random and committed to by the receiver simultaneously with x. Furthermore, to ensure that the corrupted-receiver sim-

[5]Recall that *witness encryption* [55] is a primitive where a message m can be encrypted with a statement x so that anyone with a witness w to x can decrypt m, but m cannot be recovered if x is false. Here, we would like c_x to be the "statement" that the commitment is correctly generated, and the randomness r_x and decommitment d_x the "witness".

ulator can properly simulate the output of the NISC, we "fix" the output when the trapdoor is used; that is, we augment the NISC's functionality yet again to take inputs t' and z^* from the sender and output z^* if the sender provides t' which matches the receiver's trapdoor t. More explicitly, the sender can *program the output* of the computation in case it can recover the trapdoor t selected by the receiver.

The ZK argument will then prove that either (1) there exists a witness w_1 demonstrating that c_y and the sender's NISC message are correctly generated (i.e., using the same y) given the receiver's first message, OR (2) there exists a witness w_2 which demonstrates that the sender's NISC message was generated using the trapdoor t and no input y (which, in particular, means that the NISC will output $\perp$ if the trapdoor is *incorrect*). The honest sender can provide a witness for statement (1), while the simulator in the malicious receiver case can decommit t using $\mathcal{H}$ to obtain the trapdoor and generate a witness for statement (2). In fact, this is not quite sufficient to simulate for a corrupted sender; we furthermore need an *extractability*, or "argument of knowledge", property such that the sender not only proves that there exists such a witness but also demonstrates that it *knows* such a witness—in other words, such a witness should be extractable from the prover's message in superpolynomial time. This will be necessary to show that a corrupted sender cannot provide a valid witness w_2 to the trapdoor without having recovered the correct trapdoor t and thus broken the security of the commitment scheme.

In our case, since the only extractor available to us is the decommitment oracle $\mathcal{H}$, we implement extractability by using a technique from [109] which adds a *commitment to the witness* to the statement of the proof. The sender provides a witness (w_1, w_2) and two commitments c_1 and c_2, and the proof accepts either if c_1 is a valid commitment to w_1 and w_1 is a valid witness to statement (1) above, or if the respective statement holds for c_2, w_2, and statement (2). This way, a corrupted sender must with overwhelming probability use a

Input: A commitment c, which without loss of generality contains identity id and was sent by party P in session S.
Output: A value v or the special symbol $\perp$.

Functionality:

1. Verify that $\mathsf{id} = (S, P)$ and return $\perp$ if not.
2. Otherwise, run the oracle $\mathcal{O}$ (from the definition of weak CCA security) to find a valid decommitment v (i.e., such that, for some randomness r $\mathsf{Open}(c, v, r) = \mathsf{Accept}$), and return it, or return $\perp$ if there is no valid decommitment (i.e., $\mathcal{O}$ returns $\perp$).

Figure 3.4: Decommitment helper $\mathcal{H}$ for a weakly CCA-secure commitment scheme $(\mathsf{Com}, \mathsf{Open})$.

witness for statement (1) in its proof (implying that its NISC messages and commitment to y are correctly generated), as, otherwise, a commitment of a correct witness for statement (2) would reveal the trapdoor t when decommitted and thus clearly break CCA security of the commitment scheme.

Finally, we note at this point that the commitment c_y is actually redundant, since the sender already commits to y as part of their commitment to the witness w_1, and, as we observed, the corrupted sender must (unless they can correctly guess the trapdoor t) use the witness w_1 to produce an accepting SPS-ZK proof. Hence, we can remove c_y entirely and have the simulator extract the corrupted sender's input from c_1 using the helper $\mathcal{H}$ instead.

3.5.2 Protocol and Proof

We next state and prove the formal theorem:

Theorem 10. If there exist superpolynomial-time functions $T_{\mathsf{Com}}(\cdot) = n^{\log^{c_0}(n)}$, $T_{\mathsf{ZK}}(\cdot) = n^{\log^{c_1}(n)}$, $T_{\mathsf{Sim}}(\cdot) = n^{\log^{c_2}(n)}$, and $T_{\pi}(\cdot) = n^{\log^{c_3}(n)}$ for constants $0 < c_0 < c_1 < c_2 < c_3$

so that there exist (1) a non-interactive weakly CCA-secure commitment scheme with respect to a $T_{\mathsf{Com}}(n)$-time oracle $\mathcal{O}$, (2) a non-interactive computation protocol for general polynomial-time Turing-computable functionalities satisfying $T_{\mathsf{ZK}}(\cdot)$-time stand-alone security and $T_{\mathsf{Sim}}(\cdot)$-time simulation, and (3) a non-interactive computation protocol for general polynomial-time Turing-computable functionalities satisfying $T_{\pi}(\cdot)$-time stand-alone security (and $T'(\cdot)$-time simulation for some $T'(\cdot) \gg T_{\pi}(\cdot)$), then, for any polynomial-time Turing-computable functionality $f(\cdot,\cdot)$, the protocol Π given in Figure 3.5 for computing f is an $\mathcal{H}$-EUC-secure non-interactive secure computation protocol with respect to the helper $\mathcal{H}$ in Figure 3.4.

Let $T_{\mathsf{Com}}(\cdot), T_{\mathsf{ZK}}(\cdot), T_{\mathsf{Sim}}(\cdot), T_{\pi}(\cdot)$ be as given in the theorem. Π will use the following primitives:

- $(\mathsf{Com}, \mathsf{Open})$, a secure commitment scheme satisfying weak CCA security with respect to some oracle $\mathcal{O}$ having running time $T_{\mathsf{Com}}(n)$. This is primitive (1) given in the theorem.
- $(\mathsf{ZK}_1, \mathsf{ZK}_2, \mathsf{ZK}_3)$, a two-message interactive argument which satisfies $T_{\mathsf{ZK}}(n)$-time adaptive soundness and $(T_{\mathsf{ZK}}(\cdot), T_{\mathsf{Sim}}(\cdot))$-simulatable zero-knowledge (with respective $T_{\mathsf{Sim}}(\cdot)$-time simulator $\mathsf{Sim}_{\mathsf{ZK}}$). By Theorem 9, this can be constructed from the primitive (2) given in the theorem.
- $\pi = (\mathsf{NISC}_1, \mathsf{NISC}_2, \mathsf{NISC}_3)$, a *stand-alone secure* non-interactive two-party computation protocol for the functionality h given in Figure 3.6 satisfying $T_{\pi}(\cdot)$-time security and $T'(\cdot)$-time simulation for some $T'(n) \gg T_{\pi}(n)$. This is implied by primitive (3) in the theorem.

We provide the complete proof, which constructs the polynomial-time simulator $\mathcal{S}$ (aided

by $\mathcal{H}$) required for the definition of $\mathcal{H}$-EUC-security.

Correctness of Π follows trivially from the correctness of π and $(\mathsf{Com}, \mathsf{Open})$. To prove security, we state the following lemma, which amounts to showing that Π $\mathcal{H}$-EUC-realizes the ideal functionality $\mathcal{T}_f$ for secure two-party computation of $f(\cdot, \cdot)$:

Lemma 12. Let $\Pi(\mathcal{T}_f)$ be the protocol where the sender and the receiver send their inputs x and y to the ideal functionality $\mathcal{T}_f$, which, given those inputs, computes $f(x, y)$ and sends the result to the receiver.

Then, for any polynomial-time adversary $\mathcal{A}$, there exists a polynomial-time simulator $\mathcal{S}$ such that, for any non-uniform polynomial-time environment $\mathcal{Z}$ and polynomial-time distinguisher D, there exists a negligible function $\nu(\cdot)$ such that, for any $n \in \mathbb{N}$ and $\mathsf{aux} \in \{0,1\}^*$:

$$\left|\Pr\left[D(\mathsf{Exec}_{\Pi,\mathcal{A},\mathcal{Z}}(1^n, \mathsf{aux})) = 1\right] - \Pr\left[D(\mathsf{Exec}_{\Pi(\mathcal{T}_f),\mathcal{S},\mathcal{Z}}(1^n, \mathsf{aux})) = 1\right]\right| \leq \nu(n)$$

Proof. Recall that the environment and adversary will have access to the helper $\mathcal{H}$ in both the real and ideal experiments, as will the simulator $\mathcal{S}$ which we now construct. $\mathcal{S}$ will forward communication directly between the environment $\mathcal{Z}$ and the helper $\mathcal{H}$, while simulating the session of Π started by $\mathcal{Z}$ by running one of the simulators $\mathcal{S}_R$ (see Figure 3.7), $\mathcal{S}_S$ (see Figure 3.8), or $\mathcal{S}_N$ (see Figure 3.9), depending on whether (respectively) the receiver, the sender, or neither is corrupted. We can then demonstrate that the environment's interaction with the simulator $\mathcal{S}$ in the ideal world is indistinguishable from the environment's interaction with the adversary $\mathcal{A}$ (which, without loss of generality, we can assume to be controlled by the environment) in the real world.

Input: The receiver R (with identity P_R) and the sender S (with identity P_S) are given input $x, y \in \{0,1\}^n$, respectively, and both parties have common input 1^n and session ID id.
Output: R outputs $f(x,y)$.

Round 1: R proceeds as follows:

1. Generate trapdoor $t \leftarrow \{0,1\}^n$ and randomness $r_x \leftarrow \{0,1\}^*$.
2. Compute $c_x = \mathsf{Com}(1^n, (\mathsf{id}, P_R), x||t; r_x)$.
3. Compute $(\mathsf{msg}_1, \sigma_{\mathsf{NISC}}) \leftarrow \mathsf{NISC}_1(1^n, (x, r_x, t))$, where the protocol $\pi = (\mathsf{NISC}_1, \mathsf{NISC}_2, \mathsf{NISC}_3)$ computes the functionality h given in Figure 3.6.
4. Compute $(\mathsf{zk}_1, \sigma_{\mathsf{ZK}}) \leftarrow \mathsf{ZK}_1(1^n)$.
5. Send $(\mathsf{msg}_1, \mathsf{zk}_1, c_x)$ to S.

Round 2: S proceeds as follows:

1. Generate randomness $r_1, r_2, r_{\mathsf{NISC}} \leftarrow \{0,1\}^*$.
2. Compute $\mathsf{msg}_2 = \mathsf{NISC}_2(\mathsf{msg}_1, (c_x, y, \perp, \perp); r_{\mathsf{NISC}})$.
3. Let $v = (\mathsf{msg}_1, \mathsf{msg}_2, c_x)$, $w_1 = (r_{\mathsf{NISC}}, y)$, and $w_2 = (\perp, \perp, \perp)$. Compute $c_1 = \mathsf{Com}(1^n, (\mathsf{id}, P_S), w_1; r_1)$ and $c_2 = \mathsf{Com}(1^n, (\mathsf{id}, P_S), 0; r_2)$.
4. Compute $\mathsf{zk}_2 \leftarrow \mathsf{ZK}_2(1^n, \mathsf{zk}_1, (v, c_1, c_2), (w_1, r_1, w_2, \perp))$ for the language L consisting of tuples (v, c_1, c_2), where $v = (\mathsf{msg}_1, \mathsf{msg}_2, c_x)$, such that there exists a witness (w_1, r_1, w_2, r_2) so that either:
   (a) $c_1 = \mathsf{Com}(1^n, (\mathsf{id}, P_S), w_1; r_1)$, and $w_1 = (r_{\mathsf{NISC}}, y)$ satisfies $\mathsf{msg}_2 = \mathsf{NISC}_2(\mathsf{msg}_1, (c_x, y, \perp, \perp); r_{\mathsf{NISC}})$.
   OR:
   (b) $c_2 = \mathsf{Com}(1^n, (\mathsf{id}, P_S), w_2; r_2)$, and $w_2 = (r_{\mathsf{NISC}}, t, z^*)$ satisfies $\mathsf{msg}_2 = \mathsf{NISC}_2(\mathsf{msg}_1, (c_x, \perp, t, z^*); r_{\mathsf{NISC}})$.
5. Send $(\mathsf{msg}_2, \mathsf{zk}_2, c_1, c_2)$ to R.

Output phase: R proceeds as follows:

1. Let $v = (\mathsf{msg}_1, \mathsf{msg}_2, c_x)$. If $\mathsf{ZK}_3(\mathsf{zk}_2, (v, c_1, c_2), \sigma_{\mathsf{ZK}}) \neq \mathsf{Accept}$, terminate with output $\perp$.
2. Compute $z = \mathsf{NISC}_3(\mathsf{msg}_2, \sigma_{\mathsf{NISC}})$. If $z = \perp$, terminate with output $\perp$; otherwise return z.

Figure 3.5: Protocol Π for non-interactive secure computation.

Input: The receiver R has input (x, r_x, t), and the sender S has input (c_x, y, t', z^*)
Output: Either $f(x, y)$, z^*, or the special symbol $\perp$.

Functionality:

1. If $c_x \neq \mathsf{Com}(1^n, (\mathsf{id}, P_R), x||t; r_x)$, return $\perp$.
2. If $t = t'$, then return z^*.
3. Otherwise, return $f(x, y)$ (or $\perp$ if either x or y is $\perp$).

Figure 3.6: Functionality h used for the underlying 2PC protocol π.

So, assume for the sake of contradiction that there exist some environment $\mathcal{Z}$ and distinguisher D such that, for infinitely many $n \in \mathbb{N}$, there is auxiliary input $\mathsf{aux} \in \{0,1\}^n$ such that D distinguishes the distributions $\mathsf{Exec}_{\Pi,\mathcal{A},\mathcal{Z}}(1^n, \mathsf{aux})$ and $\mathsf{Exec}_{\Pi(\mathcal{T}_f),\mathcal{S},\mathcal{Z}}(1^n, \mathsf{aux})$ with some non-negligible probability $1/p(n)$ (for polynomial $p(\cdot)$). We henceforth denote the experiments which produce these distributions by $\mathsf{Exp}_{\mathsf{Real}}$ and $\mathsf{Exp}_{\mathsf{Sim}}$, respectively; it then suffices to show that their respective views $\mathsf{View}_{\mathsf{Real}}$ and $\mathsf{View}_{\mathsf{Sim}}$ cannot be distinguished by any distinguisher running in polynomial time with access to the helper $\mathcal{H}$. We handle this in three separate cases, depending on whether the receiver, sender, or neither party is corrupted in the session of Π.

3.5.3 Corrupted Receiver

In this case, the simulator $\mathcal{S}$ will run $\mathcal{S}_R$ as given in Figure 3.7 in the experiment $\mathsf{Exp}_{\mathsf{Sim}}$; we prove that there exists no distinguisher between the views $\mathsf{View}_{\mathsf{Real}}$ and $\mathsf{View}_{\mathsf{Sim}}$ by a sequence of hybrids:

- Let H_0 be identical to $\mathsf{Exp}_{\mathsf{Real}}$, with the exception that the honest sender sends y to an instance of the ideal functionality $\mathcal{T}_f$, uses the decommitment helper $\mathcal{H}$ to retrieve the

Simulator $\mathcal{S}_R$

1. Receive the adversary's first-round message $(\mathsf{msg}_1, \mathsf{zk}_1, c_x)$.
2. Generate randomness $r_{\mathsf{NISC}} \leftarrow \{0,1\}^*$.
3. Use the helper $\mathcal{H}$ to compute $x^*||t \leftarrow \mathcal{H}(c_x)$.
4. Send x^* to the ideal functionality $\mathcal{T}_f$ and receive the output z.
5. Compute $\mathsf{msg}_2 \leftarrow \mathsf{NISC}_2(\mathsf{msg}_1, (c_x, \perp, t, z); r_{\mathsf{NISC}})$.
6. Let $v = (\mathsf{msg}_1, \mathsf{msg}_2, c_x)$, $w_1 = (\perp, \perp)$, and $w_2 = (r_{\mathsf{NISC}}, t, z)$. Compute $c_1 = \mathsf{Com}(1^n, (\mathsf{id}, P_S), 0; r_1)$ and $c_2 = \mathsf{Com}(1^n, (\mathsf{id}, P_S), w_2; r_2)$.
7. Compute $\mathsf{zk}_2 \leftarrow \mathsf{ZK}_2(1^n, \mathsf{zk}_1, (v, c_1, c_2), (w_1, \perp, w_2, r_2))$.
8. Send $(\mathsf{msg}_2, \mathsf{zk}_2, c_1, c_2)$ to the adversary.

Figure 3.7: Simulator for a corrupted receiver.

values $x^*||t \leftarrow \mathcal{H}(c_x)$, sends x^* to $\mathcal{T}_f$ on behalf of the receiver, and retrieves the output z from $\mathcal{T}_f$.

Views are identically distributed.

- Let H_1 be identical to H_0, with the exception that the sender computes the commitment c_2 as $c_2 = \mathsf{Com}(1^n, (\mathsf{id}, P_S), w_2; r_2)$ (where $w_2 = (r_{\mathsf{NISC}}, t, z)$), rather than as $c_2 = \mathsf{Com}(1^n, (\mathsf{id}, P_S), 0; r_2)$.

  Follows from weak CCA security of $(\mathsf{Com}, \mathsf{Open})$.

- Let H_2 be identical to H_1, with the exception that the sender computes the second ZK message as $\mathsf{zk}_2 \leftarrow \mathsf{Sim}_{\mathsf{ZK}}(\mathsf{zk}_1, (v, c_1, c_2))$ rather than computing it as $\mathsf{zk}_2 \leftarrow \mathsf{ZK}_2(\mathsf{zk}_1, (v, c_1, c_2), (w_1, r_1, w_2, \perp))$.

  Follows from simulation-based security of the ZK proof, specifically indistinguishability against $T_{\mathsf{ZK}}(\cdot)$*-time distinguishers and* $T_{\mathsf{ZK}}(\cdot) \gg T_{\mathsf{Com}}(\cdot)$.

- Let H_3 be identical to H_2, with the exception that the sender computes the second NISC message using the corrupted-receiver simulator $\mathcal{S}'_R$ for the underlying NISC π, as follows:

  - Forward the corrupted receiver's first-round message msg_1 for π to $\mathcal{S}'_R$, which will produce a message (x, r_x, t) to send to the ideal functionality $\mathcal{T}_h$.

– Verify that the corrupted receiver's commitments c_x and c_t satisfy $c_x = \mathsf{Com}(1^n, (\mathsf{id}, P_R), x||t; r_x)$; if not, forward $\perp$ to $\mathcal{S}'_R$ as the result from $\mathcal{T}_h$.

– Forward $f(x^*, y)$ to $\mathcal{S}'_R$ as the result from $\mathcal{T}_h$.

– $\mathcal{S}'_R$ will produce a message msg'_2 to send to the corrupted receiver for π; send $(\mathsf{msg}'_2, \mathsf{zk}_2, c_1, c_2)$ to the corrupted receiver for Π.

rather than computing it as $\mathsf{msg}_2 \leftarrow \mathsf{NISC}_2(\mathsf{msg}_1, (c_x, y, \perp, \perp))$.

Follows from simulation-based security of the NISC protocol π, specifically indistinguishability against $T_\pi(\cdot)$-time distinguishers and $T_\pi(\cdot) \gg T_{\mathsf{Sim}}(\cdot) + T_{\mathsf{Com}}(\cdot)$.

- Let H_4 be identical to H_3, with the exception that the sender computes the second NISC message as $\mathsf{msg}_2 \leftarrow \mathsf{NISC}_2(\mathsf{msg}_1, (c_x, \perp, t, z))$—that is, using the trapdoor—rather than computing it using the simulator $\mathcal{S}'_R$ as in H_3.

  Follows again from simulation-based security of the NISC protocol π, and additionally the fact that the simulator $\mathcal{S}'_R$ depends only on the adversary and not the inputs to the NISC (hence, the same simulator can be used for the definition of security in this hybrid as in the last).

- Let H_5 be identical to H_4, with the exception that the sender computes the second ZK message as $\mathsf{zk}_2 \leftarrow \mathsf{ZK}_2(\mathsf{zk}_1, (v, c_1, c_2), (w_1, \perp, w_2, r_2))$, where $w_1 = (\perp, \perp)$ and $w_2 = (r_{\mathsf{NISC}}, t, z)$, rather than computing it as $\mathsf{zk}_2 \leftarrow \mathsf{Sim}_{\mathsf{ZK}}(\mathsf{zk}_1, (v, c_1, c_2))$.

  Follows once again from simulation-based security of the ZK proof.

- Let H_6 be identical to H_5, with the exception that the sender computes the commitment c_1 as $c_1 = \mathsf{Com}(1^n, (\mathsf{id}, P_S), 0; r_1)$, rather than as $c_1 = \mathsf{Com}(1^n, (\mathsf{id}, P_S), w_1; r_1)$ (where $w_1 = (r_{\mathsf{NISC}}, y)$). Note that H_6 is now identical to the experiment $\mathsf{Exp}_{\mathsf{Sim}}$ where the adversary interacts with the simulator $\mathcal{S}_R$.

  Follows from weak CCA security of $(\mathsf{Com}, \mathsf{Open})$.

We set out to prove:

Claim 23. For the case where the receiver is corrupted in the session of Π, there exists no polynomial-time distinguisher D^* with oracle access to $\mathcal{H}$ such that, for some polynomial $p^*(\cdot)$:

$$|\Pr[D^*(\mathsf{View}_{\mathsf{Real}}) = 1] - \Pr[D^*(\mathsf{View}_{\mathsf{Sim}}) = 1]| \geq 1/p^*(n)$$

Proof. Observe that, if the views $\mathsf{View}_{\mathsf{Real}}$ and $\mathsf{View}_{\mathsf{Sim}}$ are distinguishable (i.e., the claim is not true), there must be some sequence of randomness r for the experiments prior to the session of Π (recall that the experiments prior to Π are identical) such that the views $\mathsf{View}_{\mathsf{Real}}|_r$ and $\mathsf{View}_{\mathsf{Sim}}|_r$, henceforth denoting the respective views with randomness fixed to r when applicable, are similarly distinguishable—that is:

$$|\Pr[D^*(\mathsf{View}_{\mathsf{Real}}|_r) = 1] - \Pr[D^*(\mathsf{View}_{\mathsf{Sim}}|_r) = 1]| \geq 1/p^*(n)$$

We thus proceed by proving the respective hybrids are indistinguishable for any such fixed randomness r.

To start, the views of H_0 and $\mathsf{Exp}_{\mathsf{Real}}$ are trivially identically distributed as the sender computes all of its messages in the same way. We continue with the following claims to complete the proof:

Subclaim 3. There exists no polynomial-time distinguisher D with access to the decommitment helper $\mathcal{H}$ such that, for some polynomial $p'(\cdot)$ and some fixed randomness r prior to the session of Π:

$$|\Pr[D(\mathsf{View}[H_0]|_r) = 1] - \Pr[D(\mathsf{View}[H_1]|_r) = 1]| \geq 1/p'(n)$$

Proof. This will follow from the CCA security of $(\mathsf{Com}, \mathsf{Open})$. Assuming for the sake of contradiction that there exists a polynomial-time distinguisher D which distinguishes $\mathsf{View}[H_0]|_r$ and $\mathsf{View}[H_1]|_r$ with non-negligible probability $1/p'(n)$, we can use this to construct a polynomial-time adversary $\mathcal{A}_{\mathsf{Com}}$ which breaks the CCA security of the commitment scheme. $\mathcal{A}_{\mathsf{Com}}$ does as follows:

- Run the experiment given by H_0 with the following differences:
  - Use the fixed randomness r prior to the session of Π.
  - Use CCA oracle queries in place of queries to the decommitment helper $\mathcal{H}$ for commitments using corrupted parties' identifiers.
  - Respond with $\perp$ to commitment queries using honest parties' identifiers.
  - When the honest sender generates the witness $w_2 = (r_{\mathsf{NISC}}, t, z)$, send the values w_2 and 0 and the tag (id, P_S) to the challenger to receive a commitment c^* of a randomly chosen one of the two values.
  - Substitute c^* for c_2 whenever it occurs in the session of Π.
- Once the experiment finishes, run the distinguisher D on the final view and output the result.

Since S is honest, $\mathcal{A}_{\mathsf{Com}}$ never makes a CCA oracle query using the challenge commitment's tag (id, P_S). Furthermore, we note that the ZK proof zk_2 is independent of the value

of w_2 (as it is computed using $w_2 = (\perp, \perp, \perp)$). Hence, if c^* is a commitment to 0, the view of the experiment is identically distributed to $\mathsf{View}[H_0]|_r$; if it is instead a commitment to w_2, the view is identically distributed to $\mathsf{View}[H_1]|_r$. Hence, if D distinguishes between $\mathsf{View}[H_0]|_r$ and $\mathsf{View}[H_1]|_r$ with probability $1/p'(n)$, $\mathcal{A}_{\mathsf{Com}}$ can distinguish between the experiments Exp_0 and Exp_1 in the CCA security game with probability $1/p'(n)$ and without making queries to commitments with the challenge tag, contradicting the CCA security of $(\mathsf{Com}, \mathsf{Open})$. □

Subclaim 4. There exists no $T_{\mathsf{Com}}(n) \cdot \mathsf{poly}(n)$-time distinguisher D such that, for some polynomial $p'(\cdot)$ and some fixed randomness r prior to the session of Π:

$$|\Pr[D(\mathsf{View}[H_1]|_r) = 1] - \Pr[D(\mathsf{View}[H_2]|_r) = 1]| \geq 1/p'(n)$$

Proof. This will follow from the simulation-based security of the ZK proof $(\mathsf{ZK}_1, \mathsf{ZK}_2, \mathsf{ZK}_3)$. Assume for the sake of contradiction that there exists some $T_{\mathsf{Com}}(n) \cdot \mathsf{poly}(n)$-time distinguisher D which distinguishes the respective views with some non-negligible probability $1/p'(n)$; in that case, we can construct a $T_{\mathsf{Com}}(n) \cdot \mathsf{poly}(n)$-time distinguisher D' which can distinguish the real ZK proof zk_2 from the output of the simulator $\mathsf{Sim}_{\mathsf{ZK}}$ given a certain statement $v \in L$.

There must exist some assignment of the randomness r' for both the corrupted receiver and honest sender prior to the point where the ZK proof zk_2 is generated such that D distinguishes the distributions $\mathsf{View}[H_1]|_{r||r'}$ and $\mathsf{View}[H_2]|_{r||r'}$ with probability $1/p'(n)$. Moreover, the randomness r' defines a unique statement $v = (\mathsf{msg}_1, \mathsf{msg}_2, c_x)$ and commitments c_1, c_2, where we know $(v, c_1, c_2) \in L$ because it was generated honestly by the sender, as well as

specific witnesses $w_1 = (r_{\mathsf{NISC}}, y)$ and $w_2 = (\perp, \perp, \perp)$ and a specific first message zk_1. We can then construct the $T_{\mathsf{Com}}(n)\cdot\mathsf{poly}(n)$-time distinguisher D' which distinguishes between the distributions $\mathsf{ZK}_2(1^n, \mathsf{zk}_1, (v, c_1, c_2), (w_1, r_1, w_2, \perp))$ and $\mathsf{Sim}_{\mathsf{ZK}}(\mathsf{zk}_1, (v, c_1, c_2))$. Specifically, given a sample zk_2^* from one of the two distributions, D' does as follows:

- Run the experiment given by H_1 until the session of Π, using the fixed randomness r. Internally run the $T_{\mathsf{Com}}(n)$-time oracle $\mathcal{O}$ in place of the decommitment helper $\mathcal{H}$ throughout the experiment. (This will require time $T_{\mathsf{Com}}(n) \cdot \mathsf{poly}(n)$.)
- For the session of Π, let both parties use the fixed randomness given by r', and replace the ZK proof zk_2 by the given zk_2^*.
- Once Π finishes, run D on the resulting view and output the result.

If zk_2^* is an honestly generated proof $\mathsf{zk}_2^* \leftarrow \mathsf{ZK}_2(1^n, \mathsf{zk}_1, (v, c_1, c_2), (w_1, r_1, w_2, r_2))$, then the view given to D is identically distributed to $\mathsf{View}[H_1]|_{r||r'}$; meanwhile, if it is a simulated proof $\mathsf{zk}_2^* \leftarrow \mathsf{Sim}_{\mathsf{ZK}}(\mathsf{zk}_1, (v, c_1, c_2))$, then the view will be identically distributed to $\mathsf{View}[H_2]|_{r||r'}$. Hence D' distinguishes the real and simulated proofs with the same probability $1/p'(n)$ as with which D distinguishes the respective views of the hybrids, contradicting the definition of simulation-based security of the ZK proof since it runs in time $T_{\mathsf{Com}}(n)\cdot\mathsf{poly}(n) \ll T_{\mathsf{ZK}}(n)$. $\square$

Notably, the above claim (in addition to future claims which prove non-existence of a $T_{\mathsf{Com}}(n) \cdot \mathsf{poly}(n)$-time distinguisher) applies identically to any polynomial-time distinguisher with oracle access to $\mathcal{H}$, since any such distinguisher can trivially be made into a $T_{\mathsf{Com}}(n) \cdot \mathsf{poly}(n)$-time distinguisher by running $\mathcal{O}$ internally in place of $\mathcal{H}$.

Subclaim 5. There exists no $T_{\mathsf{Com}}(n) \cdot \mathsf{poly}(n)$-time distinguisher D such that, for some polynomial $p'(\cdot)$ and some fixed randomness r prior to the session of Π:

$$|\Pr[D(\mathsf{View}[H_2]|_r) = 1] - \Pr[D(\mathsf{View}[H_3]|_r) = 1]| \geq 1/p'(n)$$

Proof. To prove this, we leverage the stand-alone security of the internal NISC protocol π. First, assume the opposite for the sake of contradiction—that is, that there exists such a $T_{\mathsf{Com}}(n) \cdot \mathsf{poly}(n)$-time distinguisher D able to distinguish the views with probability $1/p'(n)$. The definition of stand-alone security guarantees that, for every adversary $\mathcal{A}'$ against π, there exists a simulator $\mathcal{S}'_R$ whose interaction with a corrupted receiver in an idealized experiment with the ideal functionality $\mathcal{T}_h$ is indistinguishable from the real interaction where parties use π. Notably, the simulator $\mathcal{S}'_R$ depends *only* on the adversary and not on the inputs to the NISC, so the same simulator applies to the adversary (i.e., the corrupted receiver) in H_2 as to the adversary in H_3, and, critically, later to the adversary in H_4.

Now, assuming without loss of generality that the adversaries are deterministic and take their randomness as part of the auxiliary input aux, there must exist auxiliary input aux such that D distinguishes $\mathsf{View}[H_2]|_{r||\mathsf{aux}}$ from $\mathsf{View}[H_3]|_{r||\mathsf{aux}}$ with probability $1/p'(n)$. In particular, we note that aux, in conjunction with the inputs x and y, fixes the corrupted receiver's first NISC message msg_1, as well as both inputs (x, r_x, t) and $(c_x, y, \perp, \perp)$ to π.

We construct a D' that runs in time $T_{\mathsf{Sim}}(n) \cdot \mathsf{poly}(n)$ and distinguishes the real and simulated executions of π (with inputs x and y and auxiliary input aux) on the respective inputs (x, r_x, t) and $(c_x, y, \perp, \perp)$. D', given some view (m_1, m_2, out) of the messages and receiver's output from either the real interaction (between the corrupted receiver and honest sender) or the ideal interaction (between the corrupted receiver and simulator $\mathcal{S}'_R$), does as follows:

- Run the experiment given by H_2 until the session of Π, using the fixed randomness r. Internally run the $T_{\mathsf{Com}}(n)$-time oracle $\mathcal{O}$ in place of the decommitment helper $\mathcal{H}$ throughout the experiment. (Since the experiment runs the ZK simulator $\mathsf{Sim}_{\mathsf{ZK}}$, this will require time $T_{\mathsf{Sim}}(n) + T_{\mathsf{Com}}(n) \cdot \mathsf{poly}(n)$.)
- For the execution of Π, let the adversary use the fixed randomness given by aux (note that this will fix their first NISC message msg_1 to be identical to m_1), and replace the honest sender's second NISC message msg_2 by m_2.
- Once Π finishes, run D on the resulting view and output the result.

If the given view is the real execution of the NISC protocol π, the view given to D will be identically distributed to $\mathsf{View}[H_2]|_{r||\mathsf{aux}}$; if the view is the simulated execution using $\mathcal{S}'_R$, then the view given to D will be identically distributed to $\mathsf{View}[H_3]|_{r||\mathsf{aux}}$. Hence, D' must distinguish the views of the real and simulated interactions with probability $1/p'(n)$.

We remark that it is critical that the output of h run on the respective inputs be the same between H_2 and H_3 so that the output of the (real or simulated) NISC in the views distinguished by D matches the respective output in the views distinguished by D'. Indeed, it can easily be verified that the output of h in both experiments is always $f(x, y)$ when the commitments c_x and c_t are validly generated and always $\perp$ if not.

Hence, the existence of a distinguisher D which distinguishes $\mathsf{View}[H_0]|_r$ and $\mathsf{View}[H_1]|_r$ with non-negligible probability implies a D' that contradicts the stand-alone security of π (since D' runs in time $T_{\mathsf{Sim}}(n) \cdot \mathsf{poly}(n) \ll T_\pi(n)$), completing the argument. □

Subclaim 6. There exists no $T_{\mathsf{Com}}(n) \cdot \mathsf{poly}(n)$-time distinguisher D such that, for some polynomial $p'(\cdot)$ and some fixed randomness r prior to the session of Π:

$$|\Pr[D(\mathsf{View}[H_3]|_r) = 1] - \Pr[D(\mathsf{View}[H_4]|_r) = 1]| \geq 1/p'(n)$$

Proof. Identical to Subclaim 5, except that D' now differentiates between the real and simulated executions of π on inputs (x, r_x, t) and $(c_x, \perp, t, z)$. We note that, as with the above experiment, the output of h is always identical between H_3 and H_4; furthermore, since the adversary $\mathcal{A}'$ remains the same throughout all of our hybrids, the same simulator $\mathcal{S}'_R$ satisfies the definition of security for the proof here as it does for the proof in the previous subclaim. □

Subclaim 7. There exists no $T_{\mathsf{Com}}(n) \cdot \mathsf{poly}(n)$-time distinguisher D such that, for some polynomial $p'(\cdot)$ and some fixed randomness r prior to the session of Π:

$$|\Pr[D(\mathsf{View}[H_4]|_r) = 1] - \Pr[D(\mathsf{View}[H_5]|_r) = 1]| \geq 1/p'(n)$$

Proof. Identical to Subclaim 4, but using different witnesses $w_1 = (\perp, \perp)$ and $w_2 = (r_{\mathsf{NISC}}, t, z)$, though, importantly, the statement (v, c_1, c_2) is the same. □

Subclaim 8. There exists no polynomial-time distinguisher D with access to the decommitment helper $\mathcal{H}$ such that, for some polynomial $p'(\cdot)$ and some fixed randomness r prior to the session of Π:

$$|\Pr[D(\mathsf{View}[H_5]|_r) = 1] - \Pr[D(\mathsf{View}[H_6]|_r) = 1]| \geq 1/p'(n)$$

Simulator $\mathcal{S}_S$

1. Generate trapdoor $t \leftarrow \{0,1\}^n$ and randomness $r_x \leftarrow \{0,1\}^*$.
2. Compute $c_x = \mathsf{Com}(1^n, (\mathsf{id}, P_R), 0||t; r_x)$.
3. Compute $(\mathsf{msg}_1, \sigma_{\mathsf{NISC}}) \leftarrow \mathsf{NISC}_1(1^n, (0, r_x, t))$.
4. Compute $(\mathsf{zk}_1, \sigma_{\mathsf{ZK}}) \leftarrow \mathsf{ZK}_1(1^n)$.
5. Send $(\mathsf{msg}_1, \mathsf{zk}_1, c_x)$ to the adversary.
6. Receive output $(\mathsf{msg}_2, \mathsf{zk}_2, c_1, c_2)$ from the adversary.
7. Let $v = (\mathsf{msg}_1, \mathsf{msg}_2, c_x)$. If $\mathsf{ZK}_3(\mathsf{zk}_2, (v, c_1, c_2), \sigma_{\mathsf{ZK}}) \neq \mathsf{Accept}$, terminate with output $\perp$.
8. Use the helper to decommit $w_1 \leftarrow \mathcal{H}(c_1)$. Parse $w_1 = (r_{\mathsf{NISC}}, y^*)$.
9. Verify that $\mathsf{msg}_2 = \mathsf{NISC}_2(\mathsf{msg}_1, (c_x, y^*, \perp, \perp); r_{\mathsf{NISC}})$; if not, terminate with output $\perp$.
10. Send y^* to the ideal functionality $\mathcal{T}_f$.

Figure 3.8: Simulator for a corrupted sender.

Proof. Identical to Subclaim 3, but using the values w_1 and 0 for the commitment c_1. Note that the ZK proof zk_2 is independent of the value of w_1 (as it is computed using $w_1 = (\perp, \perp)$). □

Since the view of H_6 is by inspection identically distributed to the simulated experiment $\mathsf{Exp}_{\mathsf{Sim}}$, this suffices to complete the proof of the overall claim by a standard hybrid argument (i.e., if there were a D^* contradicting the claim, it would necessarily also contradict one of the above subclaims 3-8 by distinguishing the respective distributions with probability $1/6p^*(n)$).

□

3.5.4 Corrupted Sender

In this case, the simulator $\mathcal{S}$ will run $\mathcal{S}_S$ as given in Figure 3.8 in the experiment $\mathsf{Exp}_{\mathsf{Sim}}$; we again use a sequence of hybrids:

- Let H_0 be identical to $\mathsf{Exp}_{\mathsf{Real}}$, with the exception that the honest receiver sends x to an instance of the ideal functionality $\mathcal{T}_f$, uses the decommitment helper $\mathcal{H}$ to retrieve

the value $w_1 \leftarrow \mathcal{H}(c_1)$, parses $w_1 = (r_{\mathsf{NISC}}, y^*)$ and sends y^* to $\mathcal{T}_f$ on behalf of the sender (or $\perp$ if w_1 fails to parse).

Views are identically distributed.

- Let H_1 be identical to H_0, with the exception that the receiver, rather than using NISC_1 and NISC_3 to honestly generate the NISC messages and output, instead uses the corrupted sender simulator $\mathcal{S}'_S$ for the underlying NISC π and the corrupted sender $\mathcal{A}$. Specifically, in H_1, the receiver:

  - Sends the simulated message msg'_1 from $\mathcal{S}'_S$ to the corrupted sender instead of msg_1.
  - Upon receiving the corrupted sender's message, verifies that:
    * $w_1 \leftarrow \mathcal{H}(c_1)$ parses as $w_1 = (r_{\mathsf{NISC}}, y^*)$,
    * $\mathsf{ZK}_3(\mathsf{zk}_2, (v, c_1, c_2), \sigma_{\mathsf{ZK}}) = \mathsf{Accept}$, and
    * $\mathsf{msg}_2 = \mathsf{NISC}_2(\mathsf{msg}_1, (c_x, y^*, \perp, \perp); r_{\mathsf{NISC}})$.

    Terminates with output $\perp$ if not true.
  - Otherwise, returns the output z from $\mathcal{T}_f$.

  The real and simulated first messages are indistinguishable by the simulation-based security of π with respect to $T_\pi(\cdot)$-time distinguishers. Furthermore, by soundness of the ZK proof (with respect to $T_{\mathsf{ZK}}(\cdot)$-time adversaries) and weak CCA security of the commitment scheme, the corrupted sender, if it provides an accepting proof, must with overwhelming probability provide a valid witness to part (a) of the ZK language; in that case, we can show that the outputs are identical.

- Let H_2 be identical to H_1, with the exception that the receiver computes the commitment to x as $c_x = \mathsf{Com}(1^n, (\mathsf{id}, P_R), 0||t; r_x)$ rather than $c_x = \mathsf{Com}(1^n, (\mathsf{id}, P_R), x||t; r_x)$.

Follows from weak CCA security of the commitment scheme $(\mathsf{Com}, \mathsf{Open})$*; however, the CCA security adversary cannot run the full simulator* $\mathcal{S}'_S$*, so instead the simulated first message of the NISC is hard-coded as non-uniform advice. Note that, since the first message of the NISC is simulated, the receiver's first message is no longer dependent on* x*.*

- Let H_3 be identical to H_2, with the exception that the receiver no longer uses the simulator $\mathcal{S}'_S$ and instead computes $(\mathsf{msg}_1, \sigma_{\mathsf{NISC}}) \leftarrow \mathsf{NISC}_1(1^n, (0, r_x, t))$. Note that H_3 is now identical to the experiment $\mathsf{Exp}_{\mathsf{Sim}}$ where the adversary interacts with the simulator $\mathcal{S}_S$ and the honest receiver outputs the result z from the ideal functionality $\mathcal{T}_f$.

  Follows from the simulation-based security of π*.*

We claim the following:

Claim 24. For the case where the sender is corrupted in the session of Π, there exists no polynomial-time distinguisher D^* with oracle access to $\mathcal{H}$ such that, for some polynomial $p^*(\cdot)$:

$$|\Pr[D^*(\mathsf{View}_{\mathsf{Real}}) = 1] - \Pr[D^*(\mathsf{View}_{\mathsf{Sim}}) = 1]| \geq 1/p^*(n)$$

Proof. As before, this will follow from a series of subclaims proven by fixing the randomness r of the experiment prior to the session of Π. First, observe that $\mathsf{View}[H_0]$ is trivially identically distributed to $\mathsf{View}_{\mathsf{Real}}$, since the way the receiver generates its messages and output is unchanged. Next:

Subclaim 9. There exists no $T_{\mathsf{Com}}(n) \cdot \mathsf{poly}(n)$-time distinguisher D such that, for some polynomial $p'(\cdot)$ and some fixed randomness r prior to the session of Π:

$$|\Pr[D(\mathsf{View}[H_0]|_r) = 1] - \Pr[D(\mathsf{View}[H_1]|_r) = 1]| \geq 1/p'(n)$$

Proof. We begin by analyzing the views through the first message sent by the (honest) receiver. Letting $\mathsf{View}^*_i[H_0]|_r$ denote $\mathsf{View}[H_0]|_r$ with all messages after the receiver's first message in Π removed (and respectively for H_1), we show:

Subclaim 10. There exists no $T_{\mathsf{Com}}(n) \cdot \mathsf{poly}(n)$-time distinguisher D such that, for some polynomial $p'(\cdot)$ and some fixed randomness r prior to the session of Π:

$$|\Pr[D(\mathsf{View}^*_i[H_0]|_r) = 1] - \Pr[D(\mathsf{View}^*_i[H_1]|_r) = 1]| \geq 1/p'(n)$$

Proof. This will follow from the simulation-based security of the NISC protocol π. Assuming for the sake of contradiction that there exists some $T_{\mathsf{Com}}(n) \cdot \mathsf{poly}(n)$-time distinguisher D which distinguishes the respective truncated views with non-negligible probability $1/p'(n)$, we can construct a distinguisher D' which distinguishes between the real and simulated executions of π on the same inputs x, y, aux as those given to the session of Π.

D', given some view (m_1, m_2, out) of the messages and receiver's output from either the real interaction (between the corrupted sender and honest receiver) or the ideal interaction (between the corrupted sender and simulator $\mathcal{S}'_S$), does as follows:

- Run the experiment given by H_0 until the session of Π, using the fixed randomness r. Internally run the $T_{\mathsf{Com}}(n)$-time oracle $\mathcal{O}$ in place of the decommitment helper $\mathcal{H}$ throughout the experiment. (This will require time $T_{\mathsf{Com}}(n) \cdot \mathsf{poly}(n)$.)
- During Π, replace the honest receiver's first NISC message msg_1 by m_1.
- After the receiver sends their first message, run D on the resulting view and output the result.

If the given view is the real execution of the NISC protocol π, the view given to D will be identically distributed to $\mathsf{View}_i^*[H_0]|_{r||\mathsf{aux}}$; if the view is the simulated execution using $\mathcal{S}'_R$, then the view given to D will be identically distributed to $\mathsf{View}_i^*[H_1]|_{r||\mathsf{aux}}$. Hence, D' must distinguish the views of the real and simulated interactions with probability $1/p'(n)$, contradicting the definition of simulation-based security of π. □

It suffices, then, to compare the honest receiver's output between the two hybrids. In particular, notice that the verification performed in H_1 is precisely the verification of the ZK proof with the addition that $\perp$ will be returned if the malicious sender provides an invalid witness w_1 (i.e., it either provides a witness w_2 or both invalid witnesses).

So, if the ZK proof zk_2 fails to verify given the malicious sender's inputs, the receiver will return $\perp$ in both cases; if the ZK proof accepts and the sender used a statement (v, c_1, c_2) and witness $w_1 = (r_{\mathsf{NISC}}, y)$ satisfying part (a) of the language L (which, in particular, proves that c_1 is a valid commitment to w_1 and that the NISC was correctly generated with respect to the same y^* as in w_1), then both experiments will return $\perp$ if c_x is not correctly input by the sender and otherwise return $f(x, y^*)$, where y^* is the value committed to in c_1 (which, by perfect binding of $(\mathsf{Com}, \mathsf{Open})$, must be unique). Hence, there are two cases where it is possible for the outputs of the receiver to differ:

1. The malicious sender provides a proof zk_2 for a statement $(v, c_1, c_2) \notin L$ which accepts. (In this case, the commitments c_1 and c_2 to the witnesses w_1 and w_2 may be invalid.)
2. The malicious sender provides a witness $w_2 = (r_{\mathsf{NISC}}, t, z)$ proving that (v, c_1, c_2) satisfies part (b) of the language L with respect to the trapdoor t chosen by the receiver, in such a way that the NISC in H_0 outputs something besides $\perp$. (Note that H_1 will always result in output $\perp$ if the sender provides a witness w_2.)

The following two subclaims, then, complete the proof by demonstrating that the receiver's outputs are in fact statistically closely distributed between the two hybrids:

Subclaim 11. Case (1) above can occur with probability at most negligible in n during the session of Π comprising $\mathsf{View}[H_0]|_r$.

Proof. This follows from adaptive soundness of the ZK protocol $(\mathsf{ZK}_1, \mathsf{ZK}_2, \mathsf{ZK}_3)$; specifically, assuming that case (1) does occur with some non-negligible probability $1/p''(n)$, we can use this to construct a cheating prover P^* which runs in time $T_{\mathsf{Com}}(n) \cdot \mathsf{poly}(n)$ and breaks adaptive soundness by simply running the experiment H_0 with randomness r fixed (and internally running the $T_{\mathsf{Com}}(n)$-time oracle $\mathcal{O}$ in place of the CCA helper $\mathcal{H}$) and, in the session of Π, selecting the statement (v, c_1, c_2) returned by the malicious sender and the proof zk_2. Since we assumed that, with probability $1/p''(n)$, $(v, c_1, c_2) \notin L$ and $\mathsf{ZK}_3(\mathsf{zk}_2, (v, c_1, c_2), \sigma_{\mathsf{ZK}}) = \mathsf{Accept}$, this directly implies that the cheating prover P^* will provide an accepting proof of a statement $(v, c_1, c_2) \notin L$ with non-negligible probability. This contradicts the $T_{\mathsf{ZK}}(\cdot)$-time adaptive soundness of $(\mathsf{ZK}_1, \mathsf{ZK}_2, \mathsf{ZK}_3)$, since the cheating prover runs in time $T_{\mathsf{Com}}(n) \cdot \mathsf{poly}(n) \ll T_{\mathsf{ZK}}(n)$. □

Subclaim 12. Case (2) above can occur with probability at most negligible in n during the session of Π comprising $\mathsf{View}[H_0]|_r$.

Proof. This follows from CCA security of the commitment scheme $(\mathsf{Com}, \mathsf{Open})$. Assume towards a contradiction that, in the experiment H_0, the malicious sender, with some probability $1/p''(n)$, provides a statement (v, c_1, c_2) and a witness $w_2 = (r_{\mathsf{NISC}}, t, z)$ which proves that the NISC message msg_2 was correctly generated with respect to the trapdoor t, and in addition that this trapdoor t is the same trapdoor chosen by the receiver, so that the NISC in H_0 outputs something besides $\perp$. (Notice that, if t is the *incorrect* trapdoor, the NISC in H_0 will output $\perp$, since it must receive $y = \perp$ as an input for w_2 to be valid.) Given this, we can construct a polynomial-time adversary $\mathcal{A}_{\mathsf{Com}}$ that breaks CCA security of the underlying commitment scheme, as follows:

- Run the experiment given by H_0 with the following differences:
  - Use the fixed randomness r prior to the session of Π.
  - Use CCA oracle queries in place of queries to the decommitment helper $\mathcal{H}$ for commitments using corrupted parties' identifiers.
  - Respond with $\perp$ to commitment queries using honest parties' identifiers.
  - When the trapdoor t is generated, send values t and t' (for an arbitrary $t' \neq t$) and tag (id, P_R) to obtain a commitment c^* of a randomly chosen one of the two values under the respective tag. $\mathcal{A}_{\mathsf{Com}}$ continues the experiment as before, but substitutes c^* for c_t.
  - Substitute c^* for c_2 whenever it occurs in the session of Π.
- On receiving a second message $(\mathsf{msg}_2, \mathsf{zk}_2, c_1, c_2)$ from the corrupted sender, verify zk_2, and return 0 if it fails.
- Otherwise, extract a witness w_2 by running the CCA oracle on c_2. If w_2 is a tuple $(r_{\mathsf{NISC}}, t^*, z)$ such that $t^* = t$, then return 1; otherwise return 0.

Notice that c_2 (in order to verify) must be generated with the tag (id, P_S) and the receiver is honest, so the condition that the CCA oracle is never called on the tag (id, P_R) is satisfied. If c^* is a commitment of t, then the experiment is identically distributed to H_0; hence, by the assumption that case (2) occurs with probability $1/p''(n)$, $\mathcal{A}_{\mathsf{Com}}$ returns 1 with probability at least $1/p''(n)$. Otherwise, if c^* is a commitment of a random $t' \neq t$, the verification will only succeed either if w_1 and c_1 are valid or if w_2 is a decommitment of c^*, which, by perfect binding, can only be true for some witness $(r_{\mathsf{NISC}}, t', z)$. Since $t' \neq t$, $\mathcal{A}_{\mathsf{Com}}$ will always return 0 in this case, unless the sender returns both a valid witness w_1 and a witness w_2 that does not verify but commits to t; this latter case can happen with at most probability negligible in n since the input to the sender is completely independent of t. Hence, $\mathcal{A}_{\mathsf{Com}}$ distinguishes between commitments of t and t' with probability $1/p''(n) - \nu(n)$ (for some negligible $\nu(\cdot)$), thus breaking CCA security of the commitment scheme. □

□

Subclaim 13. There exists no polynomial-time distinguisher D with oracle access to the decommitment helper $\mathcal{H}$ such that, for some polynomial $p'(\cdot)$ and some fixed randomness r prior to the session of Π:

$$|\Pr\left[D(\mathsf{View}[H_1]|_r) = 1\right] - \Pr\left[D(\mathsf{View}[H_2]|_r) = 1\right]| \geq 1/p'(n)$$

Proof. We can prove this by using the CCA security of $(\mathsf{Com}, \mathsf{Open})$. Assume for the sake of contradiction that there exists a distinguisher D, running in polynomial time but with oracle access to $\mathcal{H}$, that distinguishes between $\mathsf{View}[H_1]|_r$ and $\mathsf{View}[H_2]|_r$ with probability

$1/p'(n)$. In particular, this means that there exists some sequence r' of the receiver's first-round randomness (including the randomness used to generate the simulated NISC message msg_1) such that D distinguishes between $\mathsf{View}[H_1]|_{r||r'}$ and $\mathsf{View}[H_2]|_{r||r'}$ with probability $1/p'(n)$. We can use this to construct a polynomial-time adversary $\mathcal{A}_{\mathsf{Com}}$ which breaks the CCA security of $(\mathsf{Com}, \mathsf{Open})$. $\mathcal{A}$ will receive as non-uniform advice the first-round message msg_1 generated from the randomness r' (since it cannot run the $T_\pi(n)$-time NISC simulator), and proceeds as follows:

- Send the values $x||t$ and $0||t$ and the tag (id, P_R) to the challenger to receive a commitment c^* of a randomly chosen one of the two values.
- Run the experiment given by H_1 with the following differences:
  - Use the fixed randomness r prior to the session of Π.
  - Use the randomness r' during Π, and, rather than running the simulator $\mathcal{S}'_S$, send msg_1 as the first NISC message.
  - Use CCA oracle queries in place of queries to the decommitment helper $\mathcal{H}$ for commitments using corrupted parties' identifiers.
  - Respond with $\perp$ to commitment queries using honest parties' identifiers.
  - Substitute c^* for c_x whenever it occurs in the session of Π.
- Once the experiment finishes, run the distinguisher D on the final view and output the result.

Since R is honest, $\mathcal{A}_{\mathsf{Com}}$ never makes a CCA oracle query using the challenge commitment's tag (id, P_R). When c^* commits to $x||t$, the view given to D is identically distributed to $\mathsf{View}[H_1]|_r$; meanwhile, when c^* commits to $0||t$, the view is identically distributed to

$\mathsf{View}[H_2]|_r$. Hence, if D distinguishes between $\mathsf{View}[H_1]|_r$ and $\mathsf{View}[H_2]|_r$ with probability $1/p'(n)$, $\mathcal{A}_{\mathsf{Com}}$ can distinguish between the experiments Exp_0 and Exp_1 in the CCA security game with probability $1/p'(n)$ and without making queries to commitments with the challenge tag, contradicting the CCA security of $(\mathsf{Com}, \mathsf{Open})$. (Furthermore, since $\mathcal{A}_{\mathsf{Com}}$ is given the hard-coded simulated NISC message msg_1 instead of running the NISC simulator, it runs in polynomial time with access to the CCA oracle.) □

Subclaim 14. There exists no $T_{\mathsf{Com}}(n) \cdot \mathsf{poly}(n)$-time distinguisher D such that, for some polynomial $p'(\cdot)$ and some fixed randomness r prior to the session of Π:

$$|\Pr[D(\mathsf{View}[H_2]|_r) = 1] - \Pr[D(\mathsf{View}[H_3]|_r) = 1]| \geq 1/p'(n)$$

Proof. Identical to Subclaim 10, since only the first message changes between the two hybrids. □

Since the view of H_3 is by inspection identically distributed to the simulated experiment $\mathsf{Exp}_{\mathsf{Sim}}$, this suffices to complete the proof of the overall claim by a standard hybrid argument (i.e., if there were a D^* contradicting the claim, it would necessarily also contradict one of the above subclaims 9-14 by distinguishing the respective distributions with probability $1/3p^*(n)$).

□

Simulator $\mathcal{S}_N$

1. Generate trapdoor $t \leftarrow \{0,1\}^n$ and randomness $r_x \leftarrow \{0,1\}^*$.
2. Compute $c_x = \mathsf{Com}(1^n, (\mathsf{id}, P_R), 0||t; r_x)$.
3. Compute $(\mathsf{msg}_1, \sigma_{\mathsf{NISC}}) \leftarrow \mathsf{NISC}_1(1^n, (0, r_x, t))$.
4. Compute $(\mathsf{zk}_1, \sigma_{\mathsf{ZK}}) \leftarrow \mathsf{ZK}_1(1^n)$.
5. Send $(\mathsf{msg}_1, \mathsf{zk}_1, c_x)$ to the adversary.
6. Receive the output z from the ideal functionality $\mathcal{T}_f$ (after the honest receiver and sender have sent their respective inputs).
7. Generate randomness $r_{\mathsf{NISC}} \leftarrow \{0,1\}^*$.
8. Compute $\mathsf{msg}_2 \leftarrow \mathsf{NISC}_2(\mathsf{msg}_1, (c_x, \bot, t, z); r_{\mathsf{NISC}})$.
9. Let $v = (\mathsf{msg}_1, \mathsf{msg}_2, c_x)$, $w_1 = (\bot, \bot)$, and $w_2 = (r_{\mathsf{NISC}}, t, z)$. Compute $c_1 = \mathsf{Com}(1^n, (\mathsf{id}, P_S), 0; r_1)$ and $c_2 = \mathsf{Com}(1^n, (\mathsf{id}, P_S), w_2; r_2)$.
10. Compute $\mathsf{zk}_2 \leftarrow \mathsf{ZK}_2(1^n, \mathsf{zk}_1, (v, c_1, c_2), (w_1, \bot, w_2, r_2))$.
11. Send $(\mathsf{msg}_2, \mathsf{zk}_2, c_1, c_2)$ on behalf of the sender. Send z as the receiver's final output.

Figure 3.9: Simulator for the case where neither party is corrupted.

3.5.5 Honest Receiver and Sender

Lastly, if both parties are honest in the session of Π, the simulator $\mathcal{S}$ will run $\mathcal{S}_N$ as given in Figure 3.9 in the experiment $\mathsf{Exp}_{\mathsf{Sim}}$. We omit the proofs for the following hybrids, as they all mirror their counterparts in the corrupted sender or receiver case (with the exception that the constructed adversaries will run the honest protocol instead of the corrupted sender or receiver).

- Let H_0 be identical to $\mathsf{Exp}_{\mathsf{Real}}$, with the exception that both parties send their respective inputs x and y to an instance of the ideal functionality $\mathcal{T}_f$ and the receiver outputs the result from $\mathcal{T}_f$.

  Views are statistically close by correctness of Π.

- Let H_1 be identical to H_0, with the exception that the sender computes the commitment c_2 as $c_2 = \mathsf{Com}(1^n, (\mathsf{id}, P_S), w_2; r_2)$ (where $w_2 = (r_{\mathsf{NISC}}, t, z)$), rather than as $c_2 = \mathsf{Com}(1^n, (\mathsf{id}, P_S), 0; r_2)$.

  See Subclaim 3.

- Let H_2 be identical to H_1, with the exception that the sender computes the second ZK message as $\mathsf{zk}_2 \leftarrow \mathsf{Sim}_{\mathsf{ZK}}(\mathsf{zk}_1, (v, c_1, c_2))$ rather than computing it as $\mathsf{zk}_2 \leftarrow \mathsf{ZK}_2(\mathsf{zk}_1, (v, c_1, c_2), (w_1, r_1, w_2, \perp))$.

  See Subclaim 4.

- Let H_3 be identical to H_2, with the exception that both parties compute their respective NISC messages using the honest sender/receiver simulator $\mathcal{S}'_N$ for the underlying NISC π.

  See Subclaim 5; note that the simulator $\mathcal{S}'_N$ generates both the sender's and receiver's messages in this case.

- Let H_4 be identical to H_3, with the exception that the receiver computes the commitment to x as $c_x = \mathsf{Com}(1^n, (\mathsf{id}, P_R), 0||t; r_x)$ rather than $c_x = \mathsf{Com}(1^n, (\mathsf{id}, P_R), x||t; r_x)$.

  See Subclaim 13.

- Let H_5 be identical to H_4, with the exception that the sender computes the second NISC message as $\mathsf{msg}_2 \leftarrow \mathsf{NISC}_2(\mathsf{msg}_1, (c_x, \perp, t, z))$—that is, using the trapdoor—rather than computing it using the simulator $\mathcal{S}'_R$ as in H_3.

  See Subclaim 6.

- Let H_6 be identical to H_5, with the exception that the sender computes the second ZK message as $\mathsf{zk}_2 \leftarrow \mathsf{ZK}_2(\mathsf{zk}_1, (v, c_1, c_2), (w_1, \perp, w_2, r_2))$, where $w_1 = (\perp, \perp)$ and $w_2 = (r_{\mathsf{NISC}}, t, z)$, rather than computing it as $\mathsf{zk}_2 \leftarrow \mathsf{Sim}_{\mathsf{ZK}}(\mathsf{zk}_1, (v, c_1, c_2))$.

  See Subclaim 7.

- Let H_7 be identical to H_6, with the exception that the sender computes the commitment c_1 as $c_1 = \mathsf{Com}(1^n, (\mathsf{id}, P_S), 0; r_1)$, rather than as $c_1 = \mathsf{Com}(1^n, (\mathsf{id}, P_S), w_1; r_1)$. Note that H_7 is now identical to the experiment $\mathsf{Exp}_{\mathsf{Sim}}$ where the adversary observes messages from the simulator $\mathcal{S}_N$.

See Subclaim 8.

The above sequence of hybrids is sufficient to prove the following claim, which completes the proof of Lemma 12 and in turn Theorem 10:

Claim 25. For the case where neither party is corrupted in the session of Π, there exists no polynomial-time distinguisher D^* with oracle access to $\mathcal{H}$ such that, for some polynomial $p^*(\cdot)$:

$$|\Pr[D^*(\mathsf{View}_{\mathsf{Real}}) = 1] - \Pr[D^*(\mathsf{View}_{\mathsf{Sim}}) = 1]| \geq 1/p^*(n)$$

□

3.6 Minimality of Assumptions

In this section, we prove an interesting negative result as a counterpart to Theorem 10; specifically, that the assumptions on which we base the respective protocol are minimal. We show this by proving the converse of Theorem 10—that a NISC protocol satisfying externalized UC security implies both a (polynomial-time) stand-alone secure NISC protocol with superpolynomial-time simulation and weakly CCA-secure commitments. Thus, these primitives are not only sufficient but also *necessary* for the existence of an externalized UC-secure NISC. The only gap between the sufficient and necessary conditions is that Theorem 10 requires a stand-alone NISC having simulation-based security with respect to subexponential-time distinguishers, whereas one can only construct a polynomial-time secure stand-alone NISC from our definition of UC security.

Theorem 11. Assume the existence of a protocol $\Pi = (\pi_1, \pi_2, \pi_3)$ for non-interactive computation of any polynomial-time Turing-computable functionality $f(\cdot, \cdot)$; further assume that Π satisfies the notion of UC security with respect to some superpolynomial-time helper $\mathcal{H}$. Then there exist both a stand-alone secure non-interactive two-party computation protocol (for any polynomial-time Turing-computable functionality $h(\cdot, \cdot)$) with superpolynomial-time simulation and a non-interactive weakly CCA-secure commitment scheme.

Proof. The first implication is immediate; since stand-alone SPS security is strictly weaker than externalized UC security, any NISC protocol satisfying externalized UC security is already stand-alone secure with SPS.

So, it suffices to prove that externalized UC-secure NISC implies weakly CCA-secure commitments; formally, we prove the following:

Lemma 13. Assume a protocol $\Pi = (\pi_1, \pi_2, \pi_3)$ for non-interactive computation of the functionality which, on inputs x and y, returns $f(x, y) = 1$ if $x = y$ and $f(x, y) = 0$ otherwise; further assume that Π satisfies the notion of UC security with a superpolynomial-time helper. Then there exists a commitment scheme $(\mathsf{Com}, \mathsf{Open})$ which satisfies correctness, perfect binding, and weak CCA security.

Proof. We define the weakly CCA secure commitment scheme $(\mathsf{Com}, \mathsf{Open})$ as follows:

- $\mathsf{Com}(1^n, \mathsf{id}, x)$ generates random padding $p \leftarrow \{0,1\}^n$ and outputs $c \leftarrow \pi_1(1^n, (\mathsf{id}, 1), x||p)$ as well as the session identifier id.

  That is, c is the first (receiver's) message of a new instance of Π with receiver input x, padded by the random p, and session identifier id.

Note: We shall assume throughout that the player identifiers in any instance of Π are equal to 1 for the sender and 2 for the receiver.

- $\mathsf{Open}(c, x, (p, r))$ outputs Reject if $c \neq \pi_1(1^n, (\mathsf{id}, 1), x||p; r)$, and otherwise recovers the receiver's state σ after π_1 and outputs $b \leftarrow \pi_3(\pi_2(c, x), \sigma)$.

  That is, Open *first verifies that the commitment* c *is validly generated with respect to the value* x *and the receiver's randomness; if not, it returns* Reject*. Otherwise, it returns the result (*Accept *if 1,* Reject *if 0) of running the sender of* Π *given the initial message* c *and sender's input* x *to produce a message* m*, and finally running the receiver of* Π *given* m *as the sender's message.*

Correctness of $(\mathsf{Com}, \mathsf{Open})$ will follow directly from the correctness of Π. For the other two properties, we prove the following claims:

Claim 26. $(\mathsf{Com}, \mathsf{Open})$ satisfies perfect binding.

Proof. Perfect binding will follow from the correctness and security of Π. Fix the simulator $\mathcal{S}$ (and superpolynomial-time helper $\mathcal{H}$) given by the definition of $\mathcal{H}$-EUC security for Π as a secure implementation of the equality functionality. Then assume for the sake of contradiction that there exists a commitment c and two pairs $(x, (p, r))$ and $(x', (p', r'))$ such that $x \neq x'$ but $\mathsf{Open}(c, x, (p, r)) = \mathsf{Accept}$ and $\mathsf{Open}(c, x', (p', r')) = \mathsf{Accept}$ both with non-zero probability.

First, note that this implies that c is both a correctly generated commitment to x under (p, r) and a correctly generated commitment to x' under (p', r'), as otherwise the respective opening will return Reject with probability 1. And, given that the commitments are correctly generated (and thus honestly generated first-round messages of Π), correctness[6] of Π

[6]Note that our definition specifies *perfect* correctness, as is indeed satisfied by our construction in Theorem 10.

implies that, in fact, $\mathsf{Open}(c, x, (p, r)) = \mathsf{Accept}$ and $\mathsf{Open}(c, x', (p', r')) = \mathsf{Accept}$ both with probability 1; this follows since both of these are the result of running the honest protocol Π with the valid first message c, and thus must return the correct result from Π (either Accept or Reject) with probability 1, which must be Accept due to our earlier assumption that Open returns Accept with non-zero probability on both of these commitments.

Towards our contradiction, we examine the security of Π by constructing an environment $\mathcal{Z}$ for any sufficiently large security parameter $n \in \mathbb{N}$ (i.e., any n such that $x||p$ and $x'||p'$ are valid inputs). Letting id be the identity corresponding to c, $\mathcal{Z}$, on input x^*, will do as follows:

- Start an instance of Π with a corrupted receiver, session identifier id (and player identifiers 1 for the receiver and 2 for the sender), and input $x||p$ for the receiver and x^* for the sender.
- Substitute c for the receiver's first message to the honest sender, and receive the sender's response m.
- Run the standard final round π_3 of the receiver's protocol using m as the sender's message and r as the randomness to produce an output $\pi_3(m)|_r$.

Considering the real execution of the above, on input $x^* = x||p$ for the sender, we notice that by perfect correctness the output must be 1 with probability 1, whereas if instead we provide input $x^* \neq x||p$ the output must be 0 with probability 1. So, by the security of Π, it must be the case that the final output of the ideal execution is 1 except with negligible probability when the sender's input is $x||p$ and 0 except with negligible probability when the input is anything else. However, in the ideal version of the execution, notice that the only

input to the simulator $\mathcal{S}$ that is dependent on the sender's input is the output from the ideal functionality $\mathcal{T}_f$.

We can in fact use this to deduce that the input extracted by the simulator $\mathcal{S}$ from c and sent to the ideal functionality on behalf of the corrupted sender is $x||p$ with overwhelming probability. If not, then there exists a proper subset X of all x^* not containing $x||p$ such that $\mathcal{S}$ extracts a member of X with some non-negligible probability $1/p(n)$. But this means that, comparing the case where the sender's input is $x||p$ to a case where the sender's input is $x^* \neq x||p$ but also is a non-member of X, the inputs to the simulator are identically distributed with non-negligible probability (i.e., when $\mathcal{T}_f$ returns 0 because a member of X was selected), and thus it is impossible for the output of the ideal interaction in the former case to be 1 except with negligible probability and the output of the ideal interaction in the latter case to be 0 except with negligible probability as is required for security, as for that to be true the distributions would have to share only a negligible fraction of probability mass.

This in itself is not a contradiction; however, if we consider a similar experiment to the above but using $x'||p'$ as the receiver's input rather than $x||p$ (and r' as the respective randomness), we can use the same logic to arrive at the conclusion that the input extracted by the simulator $\mathcal{S}$ from c and sent to the ideal functionality on behalf of the corrupted sender is $x'||p'$ with overwhelming probability. Clearly, this cannot be true simultaneously with the above fact; thus, by contradiction, $(\mathsf{Com}, \mathsf{Open})$ must satisfy perfect binding. □

Claim 27. $(\mathsf{Com}, \mathsf{Open})$ satisfies weak CCA security.

Proof. Fix the simulator $\mathcal{S}$ and superpolynomial-time helper $\mathcal{H}$ implied by the definition of $\mathcal{H}$-EUC security of the protocol Π. Assume for the sake of contradiction that there exists an adversary $\mathcal{A}$ which can contradict the definition of weak CCA security (Definition 22). We first show that $\mathcal{A}$, which is by definition polynomial-time with oracle access to a weak

CCA decommitment oracle $\mathcal{O}^*$, can also be effectively implemented in polynomial time with oracle access to the helper functionality $\mathcal{H}$.

Subclaim 15. Any polynomial-time adversary $\mathcal{A}$ against weak CCA security with oracle access to the oracle $\mathcal{O}^*$ defined in Definition 22 can also be implemented in polynomial time using oracle access to the helper functionality $\mathcal{H}$ instead, with error at most negligible in the security parameter n of Π[7], and with the additional property that $\mathcal{H}$ will never be queried using a session identifier sid that is the same as the identifier used in $\mathcal{A}$'s challenge commitment.

Proof. Consider replacing each of $\mathcal{A}$'s queries to $\mathcal{O}^*$ by the following process, which runs in polynomial time given oracle access to $\mathcal{H}$:

- Receive a commitment c to decommit, with tag id.
- Start a new instance of Π with a corrupted receiver and session identifier id (and player identifiers 1 for the receiver and 2 for the sender).
- Run the simulator $\mathcal{S}$ (which uses the helper $\mathcal{H}$) on the respective instance of Π, substituting c for the corrupted receiver's message. $\mathcal{S}$ will generate an input $x^*||p$ to send to the ideal functionality; return x^* to $\mathcal{A}$.

We claim that, if the above process does not generate correct responses to all oracle queries with overwhelming probability (i.e., $1 - \nu(n)$ for some negligible $\nu(\cdot)$), then there exists an environment $\mathcal{Z}$ able to distinguish between the real and simulated executions with non-negligible probability.

[7]We comment that, while the implementation of $\mathcal{O}^*$ does not decommit successfully with probability 1, decommitting with overwhelming probability is sufficient as it creates at most a negligible error in the adversary's output in the CCA security game.

First, we consider a number of "hybrid" oracles $\mathcal{O}_0, \mathcal{O}_1, \ldots$, where in $\mathcal{O}_i$ the first i queries are answered by the true oracle $\mathcal{O}^*$ and all other queries are answered by the procedure above. Assume then for the sake of contradiction that there exists some fixed randomness r for the CCA security adversary such that, in the respective instance of the security game, the poly-time implementation of $\mathcal{O}^*$ gives at least one incorrect decommitment with some non-negligible probability $1/p(n)$. Then there necessarily exists some $i \in \mathbb{N}$ such that the oracle's outputs in $\mathcal{O}_i$ and $\mathcal{O}_{i-1}$ differ with non-negligible probability $1/q(n)$ (since the adversary in the CCA security game is restricted to at most a polynomial number of oracle queries).

We use this fact to construct our distinguishing environment $\mathcal{Z}$. Specifically, $\mathcal{Z}$ receives as non-uniform advice the i^{th} query c and (padded) decommitment $x||p$ (which can be $\perp$ if c is an invalid commitment), which are deterministic given fixed randomness r and the responses from the true CCA oracle to the first $i-1$ queries, and does as follows:

- Start a single instance of Π with a corrupted receiver, session identifier given by the tag of c (and player identifiers 1 for the receiver and 2 for the sender), and receiver and sender input both equal to $x||p$.
- Replace the receiver's first message with c, and return the output of the protocol.

By perfect correctness of Π, and the assumption that c is a valid first-round message on input $x||p$, $\mathcal{Z}$ outputs 1 in the real interaction with probability 1; however, by our assumption that the responses to oracle queries in $\mathcal{O}_i$ and $\mathcal{O}_{i-1}$ differ with non-negligible probability $1/q(n)$, we know that in the ideal interaction $\mathcal{S}$ must send some $x'||p'$ with $x' \neq x$ to the ideal functionality on behalf of the corrupted receiver with at least probability $1/q(n)$. Therefore, since the honest sender's input to the ideal functionality is always $x||p$, we observe that $\mathcal{Z}$ outputs 0 in the ideal interaction with probability $1/q(n)$, thus contradicting security of Π by distinguishing the real and ideal interactions and completing our argument.

Lastly, we note that, during the $\mathcal{H}$-aided reimplementation of the adversary $\mathcal{A}$, $\mathcal{H}$ will never be queried using a session identifier sid that is the same as the identifier used in the challenge commitment. This follows from the restriction that the simulator $\mathcal{S}$ may never query $\mathcal{H}$ using an honest party's identifiers $(\mathsf{sid}, \mathsf{pid})$: the only corrupted parties are those with sid equal to the tags of the queried commitments, which by the definition of weak CCA security may never be identical to the tag of the challenge. □

We also show the following, which together with the previous claim will provide a contradiction:

Subclaim 16. $(\mathsf{Com}, \mathsf{Open})$ satisfies hiding against any polynomial-time adversary $\mathcal{A}$, even if the adversary is given oracle access to the helper functionality $\mathcal{H}$, as long as $\mathcal{A}$ never queries $\mathcal{H}$ using a session identifier sid that is the same as the identifier used in the challenge commitment.

Proof. First, let us consider an "ideal" commitment scheme $(\mathsf{Com}', \mathsf{Open}')$, defined identically to $(\mathsf{Com}, \mathsf{Open})$ except that, rather than using the protocol Π, $(\mathsf{Com}', \mathsf{Open}')$ runs the idealized version of the protocol, which we shall call Π', where the sender's and receiver's inputs in π_1 and π_2 are sent to an ideal functionality $\mathcal{T}_f$ that computes and outputs the result of the function f (i.e., the equality function) and messages are generated independently of the respective inputs by using the simulator $\mathcal{S}$.

We claim that $(\mathsf{Com}', \mathsf{Open}')$ trivially satisfies hiding, even against unbounded-time adversaries; this follows since, given a randomly generated commitment c to some value x, the respective receiver input to the idealized protocol Π' is $x||p$ for some random padding $p \leftarrow \{0,1\}^n$. Furthermore, since the commitment c is given by the simulated first-round message of the idealized protocol, it is generated completely independently of x or p. So,

given an adversary $\mathcal{A}$ which picks two values x_0 and x_1 and receives a commitment to a random one of the two, the only way $\mathcal{A}$ can receive any information about the committed value is by attempting to open the commitment—i.e., by interacting with the ideal functionality. However, since the receiver input contains not only the value x committed but also the random padding p, the ideal functionality will return 0 unless the adversary manages to guess both x and p correctly. And since the ideal functionality computes f only once, and the commitment c by construction is independent of $x||p$, $\mathcal{A}$ can guess p with probability at most 2^{-n}. The remainder of the time, it receives inputs which are entirely independent of the committed value, meaning that it clearly cannot distinguish the two commitments with non-negligible probability even if given unbounded time.

This is precisely what we need to prove the claim, since the remainder follows by the relativized universal composition theorem and the $\mathcal{H}$-EUC security of Π. Consider the following protocol G defining the hiding security game between the adversary and the challenger:

- The adversary, given input 1^n and some auxiliary input z, selects values x_0 and x_1 and sends them to the challenger.
- The challenger, given input 1^n, session identifier id, and $b \in \{0,1\}$, receives the input from the adversary and generates a commitment $c \leftarrow \mathsf{Com}(1^n, \mathsf{id}, x_b)$, which it sends to the adversary.
- The adversary receives c and produces an arbitrary output.

Let G' be defined identically to G except that G' will use the idealized Com' in place of Com. G' is clearly a $\mathcal{T}_f$-hybrid protocol (recall that $\mathcal{T}_f$ is the ideal equality functionality implemented by Π), and Π $\mathcal{H}$-EUC-realizes $\mathcal{T}_f$ by assumption; therefore, G, which is simply the composed protocol where Π replaces $\mathcal{T}_f$, $\mathcal{H}$-EUC-emulates G'. This means that, given

any polynomial-time adversary $\mathcal{A}$ in the hiding security game—even if $\mathcal{A}$ has access to the superpolynomial-time helper $\mathcal{H}$—and any inputs $1^n, \mathsf{id}, z, b$, no polynomial-time distinguisher D can distinguish the distribution of the adversary's output in the ideal execution of G' from the distribution of the adversary's output in the real execution of G. That is, if for $b \in \{0,1\}$ we let $\mathsf{Exp}_b(\mathcal{A}, n, z)$ be the adversary's output distribution in G with inputs $1^n, z, b$, as in Definition 22, and $\mathsf{Exp}'_b(\mathcal{A}, n, z)$ defined respectively for G', we are guaranteed that there exists negligible $\nu(\cdot)$ such that, for any polynomial-time distinguisher D:

$$|\Pr[D(\mathsf{Exp}_b(\mathcal{A}, n, z)) = 1] - \Pr[D(\mathsf{Exp}'_b(\mathcal{A}, n, z)) = 1]| \leq \nu(n)$$

But hiding in the ideal world provides that:

$$|\Pr[D(\mathsf{Exp}'_0(\mathcal{A}, n, z)) = 1] - \Pr[D(\mathsf{Exp}'_1(\mathcal{A}, n, z)) = 1]| \leq 2^{-n}$$

and so we have:

$$|\Pr[D(\mathsf{Exp}_0(\mathcal{A}, n, z)) = 1] - \Pr[D(\mathsf{Exp}_1(\mathcal{A}, n, z)) = 1]| \leq 2\nu(n) + 2^{-n}$$

which is sufficient to prove hiding as desired. Notice, however, that $\mathcal{H}$-EUC security requires the environment $\mathcal{Z}$ to query the helper $\mathcal{H}$ only on behalf of corrupted parties. Since the challenge commitment c, given its tag id, requires an instance of Π to be started with honest parties and session identifier id, the above holds only if the adversary $\mathcal{A}$ does not query $\mathcal{H}$ with the same session identifier. □

So, given an adversary $\mathcal{A}$ that contradicts weak CCA security using polynomial time and oracle access to the CCA oracle $\mathcal{O}^*$, Subclaim 15 implies that there is a reimplemented adversary $\mathcal{A}'$ that likewise contradicts weak CCA security and uses polynomial time and oracle access to the superpolynomial-time helper functionality $\mathcal{H}$ without invoking the helper using a session identifier equal to the tag of the challenge commitment. But this directly

contradicts Subclaim 16, since weak CCA security without access to the CCA oracle is equivalent to hiding, and the subclaim shows that $\mathcal{A}'$ cannot break the hiding property of $(\mathsf{Com}, \mathsf{Open})$ without invoking $\mathcal{H}$ using the challenge commitment's tag. Therefore, by this contradiction, $(\mathsf{Com}, \mathsf{Open})$ satisfies weak CCA security, as desired.

□

□

□

BIBLIOGRAPHY

[1] Abdalla, M., Fouque, P.A., Lyubashevsky, V., Tibouchi, M.: Tightly-secure signatures from lossy identification schemes. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 572–590. Springer, Heidelberg (Apr 2012)

[2] Abe, M., Groth, J., Ohkubo, M.: Separating short structure-preserving signatures from non-interactive assumptions. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 628–646. Springer, Heidelberg (Dec 2011)

[3] Afshar, A., Mohassel, P., Pinkas, B., Riva, B.: Non-interactive secure computation based on cut-and-choose. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 387–404. Springer, Heidelberg (May 2014)

[4] Asharov, G., Ephraim, N., Komargodski, I., Pass, R.: On perfect correctness without derandomization. Cryptology ePrint Archive, Report 2019/1025 (2019), `https://eprint.iacr.org/2019/1025`

[5] Asharov, G., Segev, G.: Limits on the power of indistinguishability obfuscation and functional encryption. In: Guruswami, V. (ed.) 56th FOCS. pp. 191–209. IEEE Computer Society Press (Oct 2015)

[6] Asharov, G., Segev, G.: On constructing one-way permutations from indistinguishability obfuscation. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016-A, Part II. LNCS, vol. 9563, pp. 512–541. Springer, Heidelberg (Jan 2016)

[7] Auerbach, B., Cash, D., Fersch, M., Kiltz, E.: Memory-tight reductions. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part I. LNCS, vol. 10401, pp. 101–132. Springer, Heidelberg (Aug 2017)

[8] Bader, C., Hofheinz, D., Jager, T., Kiltz, E., Li, Y.: Tightly-secure authenticated key exchange. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part I. LNCS, vol. 9014, pp. 629–658. Springer, Heidelberg (Mar 2015)

[9] Bader, C., Jager, T., Li, Y., Schäge, S.: On the impossibility of tight cryptographic reductions. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 273–304. Springer, Heidelberg (May 2016)

[10] Badrinarayanan, S., Garg, S., Ishai, Y., Sahai, A., Wadia, A.: Two-message witness indistinguishability and secure computation in the plain model from new assumptions.

In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017, Part III. LNCS, vol. 10626, pp. 275–303. Springer, Heidelberg (Dec 2017)

[11] Badrinarayanan, S., Goyal, V., Jain, A., Khurana, D., Sahai, A.: Round optimal concurrent MPC via strong simulation. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017, Part I. LNCS, vol. 10677, pp. 743–775. Springer, Heidelberg (Nov 2017)

[12] Badrinarayanan, S., Jain, A., Ostrovsky, R., Visconti, I.: Non-interactive secure computation from one-way functions. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part III. LNCS, vol. 11274, pp. 118–138. Springer, Heidelberg (Dec 2018)

[13] Baldimtsi, F., Lysyanskaya, A.: On the security of one-witness blind signature schemes. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part II. LNCS, vol. 8270, pp. 82–99. Springer, Heidelberg (Dec 2013)

[14] Barak, B., Pass, R.: On the possibility of one-message weak zero-knowledge. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 121–132. Springer, Heidelberg (Feb 2004)

[15] Baudron, O., Pointcheval, D., Stern, J.: Extended notions of security for multicast public key cryptosystems. In: Montanari, U., Rolim, J.D.P., Welzl, E. (eds.) ICALP 2000. LNCS, vol. 1853, pp. 499–511. Springer, Heidelberg (Jul 2000)

[16] Bellare, M., Boldyreva, A., Micali, S.: Public-key encryption in a multi-user setting: Security proofs and improvements. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 259–274. Springer, Heidelberg (May 2000)

[17] Bellare, M., Canetti, R., Krawczyk, H.: Pseudorandom functions revisited: The cascade construction and its concrete security. In: 37th FOCS. pp. 514–523. IEEE Computer Society Press (Oct 1996)

[18] Bellare, M., Ristenpart, T.: Simulation without the artificial abort: Simplified proof and improved concrete security for Waters' IBE scheme. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 407–424. Springer, Heidelberg (Apr 2009)

[19] Bellare, M., Ristenpart, T., Tessaro, S.: Multi-instance security and its application to password-based cryptography. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 312–329. Springer, Heidelberg (Aug 2012)

[20] Bellare, M., Rogaway, P.: The exact security of digital signatures: How to sign with

RSA and Rabin. In: Maurer, U.M. (ed.) EUROCRYPT'96. LNCS, vol. 1070, pp. 399–416. Springer, Heidelberg (May 1996)

[21] Benhamouda, F., Lin, H.: k-round multiparty computation from k-round oblivious transfer via garbled interactive circuits. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part II. LNCS, vol. 10821, pp. 500–532. Springer, Heidelberg (Apr / May 2018)

[22] Benhamouda, F., Lin, H., Polychroniadou, A., Venkitasubramaniam, M.: Two-round adaptively secure multiparty computation from standard assumptions. In: Beimel, A., Dziembowski, S. (eds.) TCC 2018, Part I. LNCS, vol. 11239, pp. 175–205. Springer, Heidelberg (Nov 2018)

[23] Bernstein, D.J.: Proving tight security for Rabin-Williams signatures. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 70–87. Springer, Heidelberg (Apr 2008)

[24] Bitansky, N., Lin, H.: One-message zero knowledge and non-malleable commitments. In: Beimel, A., Dziembowski, S. (eds.) TCC 2018, Part I. LNCS, vol. 11239, pp. 209–234. Springer, Heidelberg (Nov 2018)

[25] Boneh, D., Venkatesan, R.: Breaking RSA may not be equivalent to factoring. In: Nyberg, K. (ed.) EUROCRYPT'98. LNCS, vol. 1403, pp. 59–71. Springer, Heidelberg (May / Jun 1998)

[26] Brakerski, Z., Döttling, N.: Two-message statistically sender-private OT from LWE. In: Beimel, A., Dziembowski, S. (eds.) TCC 2018, Part II. LNCS, vol. 11240, pp. 370–390. Springer, Heidelberg (Nov 2018)

[27] Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. In: Goldwasser, S. (ed.) ITCS 2012. pp. 309–325. ACM (Jan 2012)

[28] Brakerski, Z., Goldwasser, S., Rothblum, G.N., Vaikuntanathan, V.: Weak verifiable random functions. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 558–576. Springer, Heidelberg (Mar 2009)

[29] Brakerski, Z., Holmgren, J., Kalai, Y.T.: Non-interactive delegation and batch NP verification from standard computational assumptions. In: Hatami, H., McKenzie, P., King, V. (eds.) 49th ACM STOC. pp. 474–482. ACM Press (Jun 2017)

[30] Brakerski, Z., Kalai, Y.T.: Monotone batch NP-delegation with applications to access

control. Cryptology ePrint Archive, Report 2018/375 (2018), https://eprint.iacr.org/2018/375

[31] Brassard, G., Chaum, D., Crépeau, C.: Minimum Disclosure Proofs of Knowledge. J. Comput. Syst. Sci. 37(2), 156–189 (Oct 1988)

[32] Bresson, E., Monnerat, J., Vergnaud, D.: Separation results on the "one-more" computational problems. In: Malkin, T. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 71–87. Springer, Heidelberg (Apr 2008)

[33] Canetti, R.: Security and composition of multiparty cryptographic protocols. Journal of Cryptology 13(1), 143–202 (Jan 2000)

[34] Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42nd FOCS. pp. 136–145. IEEE Computer Society Press (Oct 2001)

[35] Canetti, R., Dodis, Y., Pass, R., Walfish, S.: Universally composable security with global setup. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 61–85. Springer, Heidelberg (Feb 2007)

[36] Canetti, R., Jain, A., Scafuro, A.: Practical UC security with a global random oracle. In: Ahn, G.J., Yung, M., Li, N. (eds.) ACM CCS 2014. pp. 597–608. ACM Press (Nov 2014)

[37] Canetti, R., Kushilevitz, E., Lindell, Y.: On the limitations of universally composable two-party computation without set-up assumptions. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 68–86. Springer, Heidelberg (May 2003)

[38] Canetti, R., Lin, H., Pass, R.: Adaptive hardness and composable security in the plain model from standard assumptions. In: 51st FOCS. pp. 541–550. IEEE Computer Society Press (Oct 2010)

[39] Chatterjee, S., Menezes, A., Sarkar, P.: Another look at tightness. In: Miri, A., Vaudenay, S. (eds.) SAC 2011. LNCS, vol. 7118, pp. 293–319. Springer, Heidelberg (Aug 2012)

[40] Chen, J., Gong, J., Weng, J.: Tightly secure IBE under constant-size master public key. In: Fehr, S. (ed.) PKC 2017, Part I. LNCS, vol. 10174, pp. 207–231. Springer, Heidelberg (Mar 2017)

[41] Chen, J., Wee, H.: Fully, (almost) tightly secure IBE and dual system groups. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 435–460. Springer, Heidelberg (Aug 2013)

[42] Coron, J.S.: Optimal security proofs for PSS and other signature schemes. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 272–287. Springer, Heidelberg (Apr / May 2002)

[43] Damgård, I.: Towards practical public key systems secure against chosen ciphertext attacks. In: Feigenbaum, J. (ed.) CRYPTO'91. LNCS, vol. 576, pp. 445–456. Springer, Heidelberg (Aug 1992)

[44] Deng, Y., Goyal, V., Sahai, A.: Resolving the simultaneous resettability conjecture and a new non-black-box simulation strategy. In: 50th FOCS. pp. 251–260. IEEE Computer Society Press (Oct 2009)

[45] Dierks, T., Allen, C.: The TLS Protocol Version 1.0. RFC 2246 (Jan 1999), `https://rfc-editor.org/rfc/rfc2246.txt`

[46] Dierks, T., Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.1. RFC 4346 (Apr 2006), `https://rfc-editor.org/rfc/rfc4346.txt`

[47] Dierks, T., Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Aug 2008), `https://rfc-editor.org/rfc/rfc5246.txt`

[48] Dolev, D., Dwork, C., Naor, M.: Non-malleable cryptography (1998), manuscript

[49] Dwork, C., Langberg, M., Naor, M., Nissim, K., Reingold, O.: Succinct Proofs for NP and Spooky Interactions (2004)

[50] Dwork, C., Naor, M., Sahai, A.: Concurrent zero-knowledge. In: 30th ACM STOC. pp. 409–418. ACM Press (May 1998)

[51] Dworkin, M.: Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC. NIST Special Publication 800-38D (2007)

[52] Feige, U.: Alternative Models for Zero-Knowledge Interactive Proofs. Ph.D. Thesis, Weizmann Institute of Science (1990)

[53] Fischlin, M., Schröder, D.: On the impossibility of three-move blind signature schemes.

In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 197–215. Springer, Heidelberg (May / Jun 2010)

[54] Garg, S., Bhaskar, R., Lokam, S.V.: Improved bounds on security reductions for discrete log based signatures. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 93–107. Springer, Heidelberg (Aug 2008)

[55] Garg, S., Gentry, C., Sahai, A., Waters, B.: Witness encryption and its applications. In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) 45th ACM STOC. pp. 467–476. ACM Press (Jun 2013)

[56] Garg, S., Kiyoshima, S., Pandey, O.: On the exact round complexity of self-composable two-party computation. In: Coron, J.S., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part II. LNCS, vol. 10211, pp. 194–224. Springer, Heidelberg (Apr / May 2017)

[57] Garg, S., Srinivasan, A.: Two-round multiparty secure computation from minimal assumptions. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part II. LNCS, vol. 10821, pp. 468–499. Springer, Heidelberg (Apr / May 2018)

[58] Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Mitzenmacher, M. (ed.) 41st ACM STOC. pp. 169–178. ACM Press (May / Jun 2009)

[59] Gentry, C., Wichs, D.: Separating succinct non-interactive arguments from all falsifiable assumptions. In: Fortnow, L., Vadhan, S.P. (eds.) 43rd ACM STOC. pp. 99–108. ACM Press (Jun 2011)

[60] Goldreich, O.: Foundations of Cryptography: Basic Tools, vol. 1. Cambridge University Press, Cambridge, UK (2001)

[61] Goldreich, O.: Foundations of Cryptography: Volume 2, Basic Applications. Cambridge University Press, New York, NY, USA, 1st edn. (2009)

[62] Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or A completeness theorem for protocols with honest majority. In: Aho, A. (ed.) 19th ACM STOC. pp. 218–229. ACM Press (May 1987)

[63] Goldreich, O., Oren, Y.: Definitions and properties of zero-knowledge proof systems. Journal of Cryptology 7(1), 1–32 (Dec 1994)

[64] Goldwasser, S., Kalai, Y.T., Popa, R.A., Vaikuntanathan, V., Zeldovich, N.: Reusable

garbled circuits and succinct functional encryption. In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) 45th ACM STOC. pp. 555–564. ACM Press (Jun 2013)

[65] Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: Delegating computation: interactive proofs for muggles. In: Ladner, R.E., Dwork, C. (eds.) 40th ACM STOC. pp. 113–122. ACM Press (May 2008)

[66] Goldwasser, S., Micali, S.: Probabilistic Encryption. Journal of computer and system sciences 28(2), 270–299 (1984)

[67] Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems (extended abstract). In: 17th ACM STOC. pp. 291–304. ACM Press (May 1985)

[68] Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM Journal on Computing 17(2), 281–308 (Apr 1988)

[69] Goyal, V., Ishai, Y., Sahai, A., Venkatesan, R., Wadia, A.: Founding cryptography on tamper-proof hardware tokens. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 308–326. Springer, Heidelberg (Feb 2010)

[70] Goyal, V., Lin, H., Pandey, O., Pass, R., Sahai, A.: Round-efficient concurrently composable secure computation via a robust extraction lemma. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part I. LNCS, vol. 9014, pp. 260–289. Springer, Heidelberg (Mar 2015)

[71] Guo, F., Chen, R., Susilo, W., Lai, J., Yang, G., Mu, Y.: Optimal Security Reductions for Unique Signatures: Bypassing Impossibilities with a Counterexample. In: Katz, J., Shacham, H. (eds.) Advances in Cryptology – CRYPTO 2017: 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20–24, 2017, Proceedings, Part II. pp. 517–547. Springer International Publishing, Cham (2017), https://doi.org/10.1007/978-3-319-63715-0_18

[72] Haitner, I., Hoch, J.J., Reingold, O., Segev, G.: Finding collisions in interactive protocols - a tight lower bound on the round complexity of statistically-hiding commitments. In: 48th FOCS. pp. 669–679. IEEE Computer Society Press (Oct 2007)

[73] Haitner, I., Holenstein, T.: On the (im)possibility of key dependent encryption. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 202–219. Springer, Heidelberg (Mar 2009)

[74] Haitner, I., Rosen, A., Shaltiel, R.: On the (im)possibility of Arthur-Merlin witness hiding protocols. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 220–237. Springer, Heidelberg (Mar 2009)

[75] Hazay, C., Polychroniadou, A., Venkitasubramaniam, M.: Composable security in the tamper-proof hardware model under minimal complexity. In: Hirt, M., Smith, A.D. (eds.) TCC 2016-B, Part I. LNCS, vol. 9985, pp. 367–399. Springer, Heidelberg (Oct / Nov 2016)

[76] Hofheinz, D., Jager, T., Knapp, E.: Waters signatures with optimal security reduction. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 66–83. Springer, Heidelberg (May 2012)

[77] Hsiao, C.Y., Reyzin, L.: Finding collisions on a public road, or do secure hash functions need secret coins? In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 92–105. Springer, Heidelberg (Aug 2004)

[78] Impagliazzo, R., Rudich, S.: Limits on the provable consequences of one-way permutations. In: 21st ACM STOC. pp. 44–61. ACM Press (May 1989)

[79] Ishai, Y., Kushilevitz, E., Ostrovsky, R., Prabhakaran, M., Sahai, A.: Efficient non-interactive secure computation. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 406–425. Springer, Heidelberg (May 2011)

[80] Jager, T.: Verifiable random functions from weaker assumptions. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part II. LNCS, vol. 9015, pp. 121–143. Springer, Heidelberg (Mar 2015)

[81] Jager, T., Stam, M., Stanley-Oakes, R., Warinschi, B.: Multi-key authenticated encryption with corruptions: Reductions are lossy. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017, Part I. LNCS, vol. 10677, pp. 409–441. Springer, Heidelberg (Nov 2017)

[82] Kakvi, S.A., Kiltz, E.: Optimal security proofs for full domain hash, revisited. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 537–553. Springer, Heidelberg (Apr 2012)

[83] Katz, J., Ostrovsky, R.: Round-optimal secure two-party computation. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 335–354. Springer, Heidelberg (Aug 2004)

[84] Katz, J., Schröder, D., Yerukhimovich, A.: Impossibility of blind signatures from

one-way permutations. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 615–629. Springer, Heidelberg (Mar 2011)

[85] Kiyoshima, S.: Round-efficient black-box construction of composable multi-party computation. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part II. LNCS, vol. 8617, pp. 351–368. Springer, Heidelberg (Aug 2014)

[86] Kiyoshima, S., Manabe, Y., Okamoto, T.: Constant-round black-box construction of composable multi-party computation protocol. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 343–367. Springer, Heidelberg (Feb 2014)

[87] Lin, H., Pass, R.: Black-box constructions of composable protocols without set-up. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 461–478. Springer, Heidelberg (Aug 2012)

[88] Lin, H., Pass, R., Soni, P.: Two-round and non-interactive concurrent non-malleable commitments from time-lock puzzles. In: Umans, C. (ed.) 58th FOCS. pp. 576–587. IEEE Computer Society Press (Oct 2017)

[89] Lin, H., Pass, R., Venkitasubramaniam, M.: Concurrent non-malleable commitments from any one-way function. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 571–588. Springer, Heidelberg (Mar 2008)

[90] Lindell, Y.: Bounded-concurrent secure two-party computation without setup assumptions. In: 35th ACM STOC. pp. 683–692. ACM Press (Jun 2003)

[91] Lindell, Y.: Lower bounds for concurrent self composition. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 203–222. Springer, Heidelberg (Feb 2004)

[92] Luby, M.: Pseudorandomness and cryptographic applications. Princeton computer science notes, Princeton University Press, Princeton, NJ, USA (1996)

[93] Lysyanskaya, A.: Unique signatures and verifiable random functions from the DH-DDH separation. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 597–612. Springer, Heidelberg (Aug 2002)

[94] Malkin, T., Moriarty, R., Yakovenko, N.: Generalized environmental security from number theoretic assumptions. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 343–359. Springer, Heidelberg (Mar 2006)

[95] Micali, S., Rabin, M.O., Vadhan, S.P.: Verifiable random functions. In: 40th FOCS. pp. 120–130. IEEE Computer Society Press (Oct 1999)

[96] Micali, S., Rogaway, P.: Secure computation (abstract). In: Feigenbaum, J. (ed.) CRYPTO'91. LNCS, vol. 576, pp. 392–404. Springer, Heidelberg (Aug 1992)

[97] Mohassel, P., Rosulek, M.: Non-interactive secure 2PC in the offline/online and batch settings. In: Coron, J.S., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part III. LNCS, vol. 10212, pp. 425–455. Springer, Heidelberg (Apr / May 2017)

[98] Morgan, A., Pass, R.: On the security loss of unique signatures. In: Beimel, A., Dziembowski, S. (eds.) TCC 2018, Part I. LNCS, vol. 11239, pp. 507–536. Springer, Heidelberg (Nov 2018)

[99] Morgan, A., Pass, R.: Concurrently composable non-interactive secure computation (2021), https://eprint.iacr.org/2021/1562

[100] Morgan, A., Pass, R., Polychroniadou, A.: Succinct non-interactive secure computation. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part II. LNCS, vol. 12106, pp. 216–245. Springer, Heidelberg (May 2020)

[101] Morgan, A., Pass, R., Shi, E.: On the adaptive security of MACs and PRFs. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part I. LNCS, vol. 12491, pp. 724–753. Springer, Heidelberg (Dec 2020)

[102] Mouha, N., Luykx, A.: Multi-key security: The Even-Mansour construction revisited. In: Gennaro, R., Robshaw, M.J.B. (eds.) CRYPTO 2015, Part I. LNCS, vol. 9215, pp. 209–223. Springer, Heidelberg (Aug 2015)

[103] Naor, M.: On cryptographic assumptions and challenges (invited talk). In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 96–109. Springer, Heidelberg (Aug 2003)

[104] Naor, M., Reingold, O.: Number-theoretic constructions of efficient pseudo-random functions. In: 38th FOCS. pp. 458–467. IEEE Computer Society Press (Oct 1997)

[105] Naor, M., Reingold, O.: On the construction of pseudorandom permutations: Luby-Rackoff revisited. Journal of Cryptology 12(1), 29–66 (Jan 1999)

[106] Paillier, P., Vergnaud, D.: Discrete-log-based signatures may not be equivalent to discrete log. In: Roy, B.K. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 1–20. Springer, Heidelberg (Dec 2005)

[107] Paillier, P., Villar, J.L.: Trading one-wayness against chosen-ciphertext security in factoring-based encryption. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 252–266. Springer, Heidelberg (Dec 2006)

[108] Pandey, O., Pass, R., Vaikuntanathan, V.: Adaptive one-way functions and applications. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 57–74. Springer, Heidelberg (Aug 2008)

[109] Pass, R.: Simulation in quasi-polynomial time, and its application to protocol composition. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 160–176. Springer, Heidelberg (May 2003)

[110] Pass, R.: Limits of provable security from standard assumptions. In: Fortnow, L., Vadhan, S.P. (eds.) 43rd ACM STOC. pp. 109–118. ACM Press (Jun 2011)

[111] Pass, R.: Unprovable security of perfect NIZK and non-interactive non-malleable commitments. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 334–354. Springer, Heidelberg (Mar 2013)

[112] Pass, R., Rosen, A.: Concurrent non-malleable commitments. In: 46th FOCS. pp. 563–572. IEEE Computer Society Press (Oct 2005)

[113] Pass, R., Shi, E.: The sleepy model of consensus. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017, Part II. LNCS, vol. 10625, pp. 380–409. Springer, Heidelberg (Dec 2017)

[114] Pass, R., Venkitasubramaniam, M.: On constant-round concurrent zero-knowledge. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 553–570. Springer, Heidelberg (Mar 2008)

[115] Prabhakaran, M., Sahai, A.: New notions of security: Achieving universal composability without trusted setup. In: Babai, L. (ed.) 36th ACM STOC. pp. 242–251. ACM Press (Jun 2004)

[116] Rabin, M.O.: Digital signatures and public key functions as intractable as factorization. Technical Report MIT/LCS/TR-212, Massachusetts Institute of Technology (Jan 1979)

[117] Rackoff, C., Simon, D.R.: Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In: Feigenbaum, J. (ed.) CRYPTO'91. LNCS, vol. 576, pp. 433–444. Springer, Heidelberg (Aug 1992)

[118] Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Gabow, H.N., Fagin, R. (eds.) 37th ACM STOC. pp. 84–93. ACM Press (May 2005)

[119] Richardson, R., Kilian, J.: On the concurrent composition of zero-knowledge proofs. In: Stern, J. (ed.) EUROCRYPT'99. LNCS, vol. 1592, pp. 415–431. Springer, Heidelberg (May 1999)

[120] Rivest, R.L., Shamir, A., Adleman, L.M.: A method for obtaining digital signatures and public-key cryptosystems. Communications of the Association for Computing Machinery 21(2), 120–126 (1978)

[121] Schröder, D., Unruh, D.: Round optimal blind signatures. Cryptology ePrint Archive, Report 2011/264 (2011), `https://eprint.iacr.org/2011/264`

[122] Seurin, Y.: On the exact security of Schnorr-type signatures in the random oracle model. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 554–571. Springer, Heidelberg (Apr 2012)

[123] Shamir, A.: A fast signature scheme. Tech. rep., Cambridge, MA, USA (1978)

[124] Simon, D.R.: Finding collisions on a one-way street: Can secure hash functions be based on general assumptions? In: Nyberg, K. (ed.) EUROCRYPT'98. LNCS, vol. 1403, pp. 334–345. Springer, Heidelberg (May / Jun 1998)

[125] Tessaro, S.: Optimally secure block ciphers from ideal primitives. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015, Part II. LNCS, vol. 9453, pp. 437–462. Springer, Heidelberg (Nov / Dec 2015)

[126] Wang, Y., Matsuda, T., Hanaoka, G., Tanaka, K.: Memory lower bounds of reductions revisited. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part I. LNCS, vol. 10820, pp. 61–90. Springer, Heidelberg (Apr / May 2018)

[127] Yao, A.C.: Protocols for secure computations (extended abstract). In: FOCS. pp. 160–164 (1982)