On the Communication and Round Complexity of Secure Computation

Antigoni Polychroniadou

PhD Dissertation

Department of Computer Science
Aarhus University
Denmark

On the Communication and Round Complexity of Secure Computation

A Dissertation
Presented to the Faculty of Science and Technology
of Aarhus University
in Partial Fulfillment of the Requirements
for the PhD Degree

31. december 2016

Author:
Antigoni Polychroniadou

Supervisor:
Prof. Ivan Damgård

Abstract

Secure Multi-Party Computation (MPC) is a cryptographic technique allowing us to build distributed computer systems for private data. Such systems often operate in hostile environments where they are subjected to various attacks by adversarial parties. Communication efficiency is an important goal in the design of cryptographic protocols. However, the exact communication and computational complexity of general secure MPC protocols is not very well understood, despite the fact that they were introduced already in the 80s. Even though their efficiency has been significantly improved in recent years, we are still far from being able to apply MPC to large-scale computations. The research goal of this thesis is to advance the state of the art of MPC protocols by establishing new lower bounds in an effort to explore their efficiency limitations and by subsequently constructing secure MPC protocols with optimal complexities, measured in rounds of interaction, number of communicated bits and computational overhead.

More specifically, there are no known lower bounds on the communication complexity of general MPC protocols in the Information-Theoretic (IT) setting without the need of cryptographic assumptions. Even in the computational setting based on cryptographic assumptions there are also no known lower bounds on the round complexity of MPC protocols without the need of an initial trusted setup phase. Moreover, many feasibility results are prohibitively inefficient. In this thesis, we first establish lower bounds on the communication complexity of secure MPC protocols and finally construct secure MPC protocols with optimal complexities for both IT (Part III) and computationally secure protocols (Part II).

Resumé

Sikker distribueret beregning (SDB) er en kryptografisk teknik der tillader os at bygge distribuerede IT systemer der håndterer private data. Sådanne systemer opererer ofte i fjendtlige omgivelser hvor de udsættes for forskellige angreb. Kommunikationseffektivitet er et vigtigt designmål for kryptografiske protokoller. Men den eksakte kommunikations- og runde kompleksitet af generelle protokoller for SDB er ikke velforstået, påtrods af at de blev introduceret allerede i 80-erne. Selvom effektiviteten er blevet væsentligt forbedret i de senere år er vi stadig langt væk fra at kunne anvende SDB i store beregninger. Målet for denne afhandling er at forbedre status for SDB protokoller ved at etablere nye nedre grænser og dermed udforske grænserne for effektivitet, og dernæst at konstruere nye SBD protokoller med optimal kompleksitet målt i runder, kommunikation og beregningsmæssigt arbejde.

Mere specifikt, der er ingen kendte nedre grænser for kommunikationskompleksitet i den informations teoretiske (IT) model, hvor der ikke anvendes kryptografiske antagelser. Selv i den beregningsmæssige model er der heller ikke nogen grænser for rundekompleksiteten, når der ikke antages sikker opsætning. Endvidere giver mange af de kendte generelle resultater anledning til protokoller som er uanvendelige i praksis. I denne afhandling introducerer vi først nedre grænser for kommunikationskompleksiteten af sikre SDB protokoller og endelig konstruere vi sikre SDB protokoller med optimal kompleksitet for både IT modellen (Part III) og den beregningsmæssige model (See Part II).

Acknowledgments

A huge THANK YOU goes to my advisor, Ivan Damgård, who mentored me during my 4-year journey of my PhD studies. His immense experience in research taught me that patience is the epitome of strength. His kind and humble character create a literally perfect working environment. It has been a joyful journey of intense learning for me, not only on a research level, but also on a personal level. Needless to say, his guidance disseminates research freedom from posing the research questions to arranging research visit plans. Undoubtedly, I could not have imagined having a better advisor and mentor for my PhD studies.

I would also like to thank Jesper Nielsen. His continuous enthusiasm for research has been especially valuable and his enthusiasm for life helped make my time at Aarhus even more enjoyable. I am also thankful to Claudio Orlandi for his constant advices and suggestions. His personality has been a source of positive energy. I cannot think of the crypto group without Jesper's and Claudio's vibrant touches.

I was extremely fortunate to visit and collaborate with many researchers from other institutions as well. My first internship with Alon Rosen played a key role in the second year of my studies and boosted my confidence as well as my research. I would like to thank him for giving me the opportunity to interact not only with his group but also with other research groups in Israel. His direct and honest character steers constant motivation.

My collaboration with Sanjam Garg has taught me a lot, from thinking about new ideas to presenting them. His research serves as a constant source of inspiration and motivation. I would also like to thank him for giving me the opportunity to participate in the cryptography program at the Simons Institute for the Theory of Computing where I had the opportunity to interact with many people, among others with Vipul Goyal and Omkant Pandey. I would really like to thank them for our interesting discussions. Special thanks goes to Omkant for our intense research hours. I definitely had an amazing experience in Berkeley throughout my whole visit as well as during my other short visits there and I thank everyone who contributed to them.

I would like to express my deep appreciation to Tal Rabin who chose me for an amazing summer internship at the IBM crypto group. I had the great chance to interact with all the members of the group during the last months of my PhD. Without any doubt, the research experience with Craig Gentry and Shai Halevi was an instructive as well as creative journey. I extend my thanks to Fabrice Benhamouda, Eyal Kushilevitz and and Tal for all our insightful discussions. I am also thankful to Tal for all the help she constantly provides for my stays in New York.

I am grateful to Carmit Hazay and Muthu Venkitasubramaniam for our joyful collaboration. They are always available for my questions and generously spend time for our research, even at late hours via Skype. Their vast knowledge taught me a lot and influenced my technical abilities.

A big thanks to Yuval Ishai for giving me the chance to visit and work with him. The fact that he always has an answer to my questions is amazing. My interactions with him taught me a lot about research and helped me on how to view several research questions. Moreover, I would also like to thank him for our collaboration with Elette Boyle who I really enjoy working.

Special thanks to Ran Canetti and Daniel Wichs for accepting to referee this thesis. The lenses of their work are so broad that relate to multiple areas in cryptography. Needless to say that topics of this thesis are highly inspired by their work.

Many thanks to all current and past colleagues and members of Aarhus University whom I shared many wonderful experiences. Special thanks to Ignacio Cascudo, Nico Dottling, Sebastian Faust and Pratyay Mukherjee for our research collaborations and fun discussions. I also thank my colleagues Daniel Genkin, Divya Gupta, Peihan Miao and Dakshita Khurana from different research groups with whom I spent many hours of productive research.

The seed of this thesis was planted during my masters program in London and especially during an incredible collaboration with Kenny Paterson. He motivated me to go on with my studies and pursue a PhD. Moreover, I really appreciate my collaboration with my advisors in Greece, George Stephanides and Konstantinos Chalkias. I am really thankful to them.

Last but not the least, I would like to thank my family: my parents Nikolaos and Afroditi and my brothers Vasilis and Giannis for higly supporting me and bearing with me not only during my PhD studies but in each different phase of my life. I am thankful to all my friends in and outside Greece for the fun and supporting times. Special thanks to my three little nephews who really make my day! Finally, I would like to thank my other half for his constant patience and inspiration. His originality and encouragement are of a great importance to me.

Antigoni Polychroniadou,
Aarhus, December 31, 2016.

Contents

Part I

Introduction

Chapter 1

Motivation

A central as well as fundamental problem in cryptography is the study of secure Multi-Party Computation (MPC) cryptosystems. MPC is a cryptographic technique allowing us to build secure distributed computer systems over confidential data operating in hostile environments subject to various attacks by adversarial parties. In particular, such systems allow a set of mutually distrustful group of parties to compute a joint function on their inputs without the need of a trusted third party, while still maintaining both the privacy of the inputs and the correctness of the outputs. In the presence of a trusted third party the problem is trivial since all parties could privately send their inputs to the trusted third entity, which computes the function on behalf of the parties and announces the result. The feasibility MPC results from the '80s [Yao82, GMW87, BOGW88, CCD88], tell us that the parties can replace the trusted entity with a cryptographic MPC protocol while still preserving the same level of security that the trusted entity intuitively provides. There are countless applications of MPC such as electronic elections, electronic auctions, secure benchmarking and secure signal processing, to name just a few. Last but not least, distributed MPC systems seem to be one of the only viable defences against organised hacking and hostile intelligence services since data can only be stolen by breaking into almost all the computers.

Communication efficiency is an important goal in the design of MPC protocols. In particular, the exact communication complexity of general MPC protocols, since their inception in the '80s, is not very well understood neither in the Information-Theoretic (IT) setting without the need of cryptographic assumptions nor in the computational setting based on cryptographic assumptions. Moreover, many fundamental results in the area of MPC are prohibitively inefficient. Note that even though the efficiency of MPC protocols has been significantly improved in recent years, we are still very far from being able to apply MPC to large-scale, cloud computing and big data applications. All these issues are illustrated by the following shortcomings:

- Shortcoming-1 [Computational Setting]: Even in the computational setting based on cryptographic assumptions there are no known MPC protocols for general functionalities with optimal round complexity, while still getting the best possible security. In fact, there are no known lower bounds on the round complexity of MPC protocols without trusted setup.

- Shortcoming-2 [Information-Theoretic Setting]: The methods typically used in IT MPC protocols tend to be computationally much more efficient than the cryptographic machinery used in the computational setting but on the other hand, known IT constructions require excess communication. There are even no (non-trivial) known lower bounds on the communication/round complexity of general IT MPC protocols. Such lower bounds would enable us to focus our attention to where efficiency improvements are possible.

- Shortcoming-3 [Secure RAM Computation]: IT and computationally secure protocols have traditionally been developed in the circuit model of computation. However, a lot of recent work focuses on improving the overhead of such protocols when applied to realistic computations that are designed for accessing Random Access Machines (RAMs). However, there are no known protocols over large inputs in the RAM model with minimal complexities.

The goal of this thesis is to advance the state of the art in MPC protocols by the establishment of lower bounds on the communication and round complexity of secure MPC protocols as well as the instantiation of secure MPC protocols with optimal efficiency complexities. Important measures in this respect are the number of rounds we need to do, the communication complexity (the total number of bits sent) as well as the computational overhead. Obviously, achieving a constant number of rounds and *low* communication complexity, while still getting the best possible security, is an important research goal. To this end, this thesis focuses on the following three main objectives: (1) establish (non-trivial) lower bounds on the round complexity of *computationally secure* MPC protocols without setup (2) investigate the limitations of what we can achieve by showing lower bounds for *Information-Theoretic* MPC protocols, attacking a long-standing open problem and (3) construct new MPC protocols with optimal complexities based on different setup assumptions (or without setup) under the strongest security guarantees. Finally, achieve minimal overhead for cryptographic protocols in the RAM model.

Chapter 2

Contributions

We stress that current IT secure protocols, which are efficient in the circuit size of the evaluated function, are not constant round, and we highlight that it is a major open problem if it is even possible to have unconditional security and constant number of rounds for secure computation of any function. However, if we are willing to make computational assumptions, we can achieve constant round protocols. This state of affairs leads to the aforementioned objectives in Chapter 1. In this thesis, we make progress towards these goals and in most cases we completely achieve them. To this end, we state our contributions of secure computation in the computational setting, IT setting as well as mention our results in the RAM model of computation. Parts of the following sections are partially based on text from our work in [GMPP16, HPV16, HPV17, GP15, DPR16, DNPR16, GIP15].

2.1 Computational Setting

Unlike the IT setting, it is actually possible to achieve constant round protocols in the computational setting. The first example of such a constant round protocol is Yao's garbled circuits [Yao82, Yao86], a form of encrypted circuits, for secure Two-Party Computation (2PC). Since then secure protocols have been constructed in a long series of works in the *plain* model without trusted setup assumptions. In order to achieve strong concurrency security guarantees such as Universal Compossibility (UC) [Can01], secure protocols have also been constructed from trusted setup assumptions such as the *Common Reference String (CRS)* model where all parties receive as common input, in an initial setup phase, a string sampled from an a priori fixed distribution (from some trusted authority). Furthermore, secure protocols have been designed based on decentralized setup assumptions such as the *tamper-proof hardware token* model. In the following, we present our results in the plain, CRS and hardware token model.

2.1.1 Secure Computation in the Plain Model

The round complexity of secure computation is a fundamental question in the area of secure computation. In the past few years, we have seen tremendous progress on this question, culminating into constant round protocols for securely computing any MPC functionality [BMR90, KO04, Pas04, DI05, DI06, PPV08, Wee10, Goy11, LP11a, GLOV12]. These works essentially settle the question of *asymptotic* round complexity of this problem. The *exact* round complexity of secure computation, however, is still not very well understood. [1] For the special case of 2PC, Katz and Ostrovsky [KO04] proved that *five* rounds are necessary and sufficient. In particular, they proved that two-party coin-flipping cannot be achieved in *four* rounds, and pre-

[1]Our rough estimate for the exact round complexity of aforementioned MPC results in the computational setting is 20-30 rounds depending upon the underlying components and computational assumptions.

sented a five-round protocol for computing every 2PC functionality. The exact round complexity of MPC protocols has never been addressed before.

The standard model for MPC assumes that parties are connected via authenticated point-to-point channels as well as *simultaneous message exchange* channels in which all parties can simultaneously send messages at the same time. Therefore, in each round, all parties can simultaneously exchange messages.

This is in sharp contrast to the "standard" model for 2PC where, usually, a simultaneous message exchange framework is not considered. Due to this difference in the communication model, the negative result of Katz-Ostrovsky [KO04] of four rounds, does not apply to the multi-party setting. In particular, a four round multi-party coin-flipping protocol might still exist!

In other words, the results of Katz-Ostrovsky only hold for the special case of two parties *without* a simultaneous message exchange channel. The setting of 2PC *with a simultaneous message exchange channel* has not been addressed before. Therefore, in this work we address the following two questions in the plain model which do not consider any trusted initial setup or an initial trusted CRS distributed between the parties:

What is the exact *round complexity of secure MPC in the plain model?*

In the presence of a simultaneous message exchange *channel, what is the exact round complexity of secure 2PC in the plain model?*

These questions are intimately connected to the round complexity of *non-malleable commitments* [DDN91b]. Indeed, new results for non-malleable commitments have almost immediately translated to new results for secure computation. For example, the round complexity of coin-flipping was improved by Barak [Bar02], and of every multi-party functionality by Katz, Ostrovsky, and Smith [KOS03] based on techniques from non-malleable commitments. Likewise, black-box constructions for constant-round non-malleable commitments resulted in constant-round black-box constructions for secure computation [Wee10, Goy11] (see Section 3.1.1 for detailed related work). However, all of these results only focus on *asymptotic* improvements and do not try to resolve the exact round complexity, thereby leaving the following fundamental question unresolved:

What is the relationship between the exact *round complexities of non-malleable commitments and secure computation?*

This question is at the heart of understanding the exact round complexity of secure computation in both multi-party, and two-party with simultaneous message transmission.

In this thesis we try to resolve the questions mentioned above. We start by focusing on the simpler case of 2PC with a simultaneous message exchange channel, since it is a direct special case of the MPC setting.

2.1.1.1 Lower bounds for coin-flipping

We show that four *simultaneous message exchange rounds* are necessary for 2PC in the plain model. More specifically, we show that:

Theorem 2.1.1. [GMPP16] Let λ be the security parameter. Even in the simultaneous message model, there does not exist a *three*-round protocol for the two-party coin-flipping functionality for $\omega(\log \lambda)$ coins which can be proven secure via black-box simulation.

In fact, as a corollary all of the rounds must be "strictly simultaneous message transmissions", that is, both parties must *simultaneously* send messages in each of the four rounds. This is because in the simultaneous message exchange setting, the security is proven against the so called "rushing adversaries" who, in each round, can decide their message after seeing the messages of all honest parties in that round. Consequently, if only one party sends a message for example in the fourth round, this message can be "absorbed" within the third message of this party [2], resulting in a three round protocol.

2.1.1.2 Results in the 2PC setting with a *simultaneous message exchange* channel

Next, we consider the task of constructing a protocol for coin-flipping (or any general functionality) in four simultaneous message exchange rounds and obtain a positive result. In fact, we obtain our results by directly exploring the *exact relationship* between the round complexities of non-malleable commitments and secure computation. Specifically, we first prove the following result:

Theorem 2.1.2. [GMPP16] Assuming the existence of enhanced trapdoor permutations and a k-round protocol for parallel & 3-robust non-malleable commitment,[3] there exists a k'-round protocol for securely computing *every two-party functionality* with black-box simulation in the plain model against any malicious adversary under simultaneous message exchange channels, where $k' = \max(4, k + 1)$.

Instantiating this protocol with non-malleable commitments from [PPV08], we get a **four round** protocol for every two-party functionality in the presence of a simultaneous message exchange channel, albeit under a non-standard assumption (adaptive one-way function). However, a recent result by Goyal et al. [GPR16] constructs a non-malleable commitment protocol in *three* rounds from one-way function, although their protocol is not 3-robust and does not immediately extend to the parallel setting. Instantiating our protocol with an extention of [GPR16] presented in [COSV16b] based on one-way permutations secure w.r.t. sub-exponential time adversaries yields a *four round* 2PC protocol.

[2]Note that, such absorption is only possible when it maintains the mutual dependency among the messages, in particular does not affect the next-message functions.

[3]Parallel simply means that the man-in-the-middle receives κ non-malleable commitments in *parallel* from the left interaction and makes κ commitments on the right. Almost all known non-malleable commitment protocols satisfy this property. We describe the notion of a robust non-malleable commitment in Section 4.2.1; informally, traditional definitions of non-malleability consider a setting where a man-in-the middle adversary is participating in two (or more) executions of the *same* protocol. However, non-malleable commitments *robust* w.r.t. arbitrary protocols considers a class of adversaries that can participate in a left interaction of any *arbitrary* protocol. We consider 3-robust protocols secure against 3-round arbitrary protocols.

2.1.1.3 Results in the multi-party setting

Next, we focus on the case of the multi-party *coin flipping* functionality. We show that a simpler version of our two-party protocol gives a result for multi-party *coin-flipping* presented in two versions:

Theorem 2.1.3. [GMPP16] Assuming the existence of enhanced trapdoor permutations and a k-round protocol for parallel & 3-robust non-malleable commitment, there exists a k'-round protocol for securely computing the *multi-party coin-flipping* functionality for polynomially many coins with black-box simulation in the plain model against any malicious adversary, where $k' = \max(4, 2k)$.

Theorem 2.1.4. [GMPP16] Assuming the existence of enhanced trapdoor permutations and a k-round protocol for parallel, input-delayed & 3-robust non-malleable commitment, there exists a k'-round protocol for securely computing the *multi-party coin-flipping* functionality for polynomially many coins with black-box simulation in the plain model against any malicious adversary, where $k' = \max(4, k + 1)$.

Combining this result with the two-round multi-party protocol of Mukherjee and Wichs [MW16] (based on LWE [Reg05]), we obtain a $k' + 2$ round protocol for computing *every multi-party* functionality. Instantiating these protocols with non-malleable commitments from [PPV08][4], we obtain a **four round** protocol for *coin-flipping* and a **six round** protocol for *every* functionality.

Finally, we show that the coin-flipping protocol for the multi-party setting can be extended to compute what we call "coin-flipping with committed inputs" functionality. Using this protocol with the two-round protocol of [GGHR14] based on indistinguishability obfuscation [GGH$^+$13b], we obtain a **five round** MPC protocol.

2.1.1.4 Subsequent and future work

The subsequent work of [BHP17] shows how to achieve an optimal (four) round MPC protocols based on the non-malleable commitment of [PPV08] and hardness of LWE.

A lot of work has been done recently on reducing the round complexity and computational assumptions of non-malleable commitments, see [COSV16b, GKS16, COSV16a]. In order to build four round protocols based on standard as well as minimal assumptions, one possible avenue is to construct a parallel (and 3-robust) three-round non-malleable commitment scheme based on standard assumptions.

2.1.2 Secure Computation in the CRS Model

We start by noting that two rounds of interaction are essential for realizing 2PC and MPC protocols in the CRS model. This is because a one-round protocol is inherently susceptible to the "residual function" attack in which a corrupted party could repeatedly evaluate the "residual

[4]The work of [PPV08] provides non-interactive and two-round non-malleable commitment schemes based on adaptive OWFs.

function", with the inputs of the honest parties fixed, on many different inputs of its own (e.g., see [HLP11]). This attack can be circumvented by having two rounds of interaction — where for example in the first round parties commit to their inputs and the output is generated only in the second round. The first round commitments guarantee that the "residual function" attack can not be performed in this setting.

The two-round lower bound holds for both the static and the adaptive setting. Unlike the static setting where the adversary corrupts parties on the onset of the protocol, adaptive security ensures that security holds against an adversary who can dynamically corrupt parities during the execution of the protocol. In general, security against static corruptions does not guarantee security against adaptive corruptions [CDD$^+$04]. Furthermore, adaptive security has been a notoriously difficult notion to achieve. For the static setting, Yao's original two-party secure computation protocol [Yao82] was already round-optimal. However, achieving similar results in the adaptive setting has remained open. Therefore, we focus on the adaptive setting in this section. See Section 3.1.2 for detailed related work.

Adaptive security is particularly hard to achieve in settings where arbitrary number of parties can be corrupted and honest parties are not trusted to properly erase their internal state. Specifically the round complexity of all known adaptively secure protocols secure against an arbitrary number of corruptions grows (see, e.g. [CLOS02, GS12, DMRV13]) linearly in the depth of the circuit that the parties are trying to compute. We stress that for this problem, this limitation holds for essentially every special case of interest — namely, even if we were to restrict to semi-honest/passive adversaries or to the special case of two-party protocols.

In this thesis we ask the following fundamental question:

Is it possible to construct a constant/optimal round MPC protocol secure against adaptive corruption of arbitrary number of parties?

2.1.2.1 Results on adaptive security for arbitrary corruptions

We answer the above question in the affirmative and show how to obtain a two-round adaptively secure MPC protocol which is round-optimal. Specifically:

Theorem 2.1. [GP15] Assuming sub-exponentially secure indistinguishability obfuscation and doubly enhanced trapdoor permutations, we show that *every multi-party functionality* can be UC-securely [Can01] computed in the CRS model against any adaptive, malicious adversary corrupting an arbitrary number of parties with just two rounds of broadcast.

We stress that in the above theorem we are in the setting where security holds against an adversary corrupting any arbitrary number of parties. Furthermore, honest parties in our case are not required to erase anything. Also note that our results are for the strongest notion of security, the UC security. This means that our protocol remains secure even when multiple instances of our protocol are executed simultaneously. Since it is impossible to achieve UC security for dishonest majority without assuming trusted setup assumptions [CF01, CKL03, Lin03a], we base our construction in the CRS model. In our results we consider an asynchronous multi-party network[5] where the communication is open (i.e. all the communication between the parties is seen by the adversary) and delivery of messages is not guaranteed. For simplicity, we assume that the delivered messages are authenticated. This can be achieved using standard methods.

2.1.2.2 Subsequent and future work for arbitrary corruptions

A subsequent work by [CP16] lift our protocol in the RAM model of computation and present a two-round protocol that does not require sub-exponential use of indistinguishability obfuscation only for the semi-honest case.

Since we can obtain two-round adaptive MPC protocols from indistinguishability obfuscation assumptions we leave open the question of two-round adaptive MPC protocols in the CRS model from standard assumptions.

2.1.2.3 Results on adaptive security for all-but-one corruptions

Restricting the adversary to all-but-one corruptions, there no known optimal-round adaptive protocols based on standard assumptions. In addition, as far as communication complexity is concerned, all known constant-round protocols [6], such as the work of Ishai et al. [IPS08], require communication complexity proportional to the circuit size of the evaluated function. Therefore, the question becomes:

> *Is it possible to construct constant/optimal adaptive MPC protocols secure against all-but-one corruptions with communication complexity independent of the circuit size?*

We answer the above question in the affirmative. More specifically, we achieve an adaptive UC-secure protocol that tolerates corruption of $n-1$ out of the n players with UC secure composition with protocols secure against $n-1$ corruptions. Our protocol requires a constant number of rounds and its communication complexity depends only on the length of inputs and outputs (and the security parameter), and not on the size of the evaluated and decryption circuits. The protocol is secure if the LWE problem is hard. Moreover, we *do not* consider the weaker model of secure erasures.

Theorem 2.2. [DPR16] Under the LWE assumption, we show that *every multi-party functionality* can be UC-securely computed in the CRS model [7] against any adaptive, malicious adversary corrupting all-but-one parties within a constant number of rounds and communication complexity dependant only on the length of the inputs and outputs of the functionality.

[5] The fact that the network is asynchronous means that the messages are not necessarily delivered in the order which they are sent.

[6] The adaptive protocols of [IPS08] would generically yield a larger constant for the number of rounds i.e. 30 rounds.

[7] In our MPC protocols secure against all-but-one adaptive corruptions the common string is sampled from a specific distribution. In particular, we assume that a public key has been distributed, and parties have been given shares of the corresponding secret key.

Assuming adaptively secure UC Non-Interactive Zero-Knowledge (NIZK), proposed by Groth et al. [GOS12], and LWE our adaptive protocols run in three rounds.

Theorem 2.3. [DPR16] Assuming hardness of LWE and the existence of adaptively secure UC NIZK, we show that *every multi-party functionality* can be UC-securely computed in the CRS model against any adaptive, malicious adversary corrupting all-but-one parties within three rounds of broadcast.

Note that in the above theorem the communication complexity of the derived adaptive protocol is also dependant only on the length of the inputs and outputs of the evaluated function. In our construction we assume a broadcast channel where encryption is performed using what we call Equivocal Fully Homomorphic Encryption (FHE), a notion weaker than Non-Committing Encryption (NCE)[CFGN96]. Using our equivocal scheme we also build adaptively secure UC commitments and UC Zero-Knowledge (ZK) proofs (of knowledge) based on hardness of LWE.

Last but not least, in the standard ZK-based decryption used by approaches based on FHE, all parties need to append a ZK proof with communication complexity dependant in the size of the decryption circuit in order to prove correctness of decryption. Using an AMD code mechanism [CDF$^+$08] we avoid the use of ZK and achieve communication complexity that does not scale with the decryption circuit. In particular, the total communication complexity of the decryption phase of our concrete protocol is $\mathcal{O}(n^2\lambda)$ where λ denotes the security parameter and n the number of parties. Instead, our rough estimate on the complexity of [IPS08], combined with the best known outer and inner protocols and optimized with the use of FHE based on NCE, yield an overhead of $\Omega(n^6\lambda^5)$. Note that we have tried to be optimistic on the part of the IPS compiler to not give our concrete protocol an unfair advantage.

Motivated by ruling out one possible approach to achieving adaptive security, Katz et al. [KTZ13] showed that FHE with security against adaptive corruption of the receiver is impossible. In our setting, we distribute the secret key of an FHE scheme among n parties; since we allow only $n-1$ of the parties to be corrupted, the impossibility result from [KTZ13] does not apply. Note that if an FHE scheme is to be of use in MPC, it seems to be necessary that the players are able to decrypt, if not by themselves, at least by collaborating. But if corruption of all n players was allowed, the adversary would necessarily learn all secret keys, and then the impossibility result of [KTZ13] would apply. This suggests that our result for multi-party functionalities with $n-1$ corruptions is the best we can achieve based *only* on FHE from standard assumptions.

2.1.2.4 Subsequent and future work for all-but-one corruptions

The authors in [LSS16] provide secure multi-party computation protocols based on our AMD code mechanism to avoid the use of ZK in their constructions.

The work of [GVW15] proposed equivocal homomorphic trapdoor functions. Such primitives could be used in conjunction with multi-key FHE to yield a two round MPC protocol, however, in such an approach the CRS will still have to be sampled from a specific distribution. Therefore, we leave open the question of proving or disproving the existence of adaptive two-round MPC in the common *random* string model for all-but-one corruptions.

2.1.3 Secure Computation in the Tamper-Proof Hardware Model

In this section, we first revisit secure computation for static corruptions in the tamper-proof hardware model since there no known optimal round protocols from minimal assumptions even against static corruptions. Recall that adaptive security has been a notoriously difficult notion to achieve. Based on our results on Oblivious Transfer (OT) in the static setting we attack the adaptive setting and we also present our contributions for protocols secure against adaptive corruptions.

2.1.3.1 Static corruptions

Traditional results prove security in the stand-alone model, where a *single* set of parties run a *single* execution of the protocol. However, the security of most cryptographic protocols proven in the stand-alone setting does not remain intact if many instances of the protocol are executed concurrently [Can01, CF01, Lin03a]. The strongest setting for concurrent security, known as *Universally Composable* (UC) security [Can01] considers the execution of an unbounded number of concurrent protocols in an arbitrary and adversarially controlled network environment. Unfortunately, stand-alone secure protocols typically fail to remain secure in the UC setting. In fact, as mentioned above, without assuming some *trusted help*, UC-security is impossible to achieve for most tasks [CF01, CKL03, Lin03a]. Consequently, UC-secure protocols have been constructed under various *trusted setup* assumptions in a long series of works; see [BCNP04, CDPW07, Kat07, KLP07, CPS07, LPV09, DMRV13] for few examples.

One such setup assumption and the focus of this section is the use of tamper-proof hardware tokens. Interestingly, the minimal assumption for secure computation based on tokens is one-way functions. In contrast, secure computation constructions even in the CRS model, necessarily require assumptions stronger than one-way functions [DNO10]. Furthermore, from a practical perspective, there has been tremendous interest in the hardware community to develop tamper-proof hardware. For instance, the next generation Intel chips, will provide a set of extensions, referred to as Software Guard Extensions (SGX), in the architecture that aims to provide integrity and privacy guarantees to security-sensitive computation performed on a computer to guard from privileged software (for e.g., kernel) which is potentially malicious.[8]

The first work to model tokens in the UC framework was by Katz in [Kat07] who introduced the $\mathcal{F}_{\text{WRAP}}$-functionality to capture such tokens and demonstrated feasibility of realizing general functionalities with UC-security. Most of the previous works in the tamper proof hardware [Kat07, CGS08, LPV09, GIS+10, DMRV13, CKS+14, DKMN15b] rely on this formulation. As we explain next, this formulation does not provide adequate composability guarantees. We begin by mentioning that any notion of composable security in an interactive setting should allow for multiple protocols to co-exist in the same system and interact with each other. We revisit the following desiderata put forth by Canetti, Lin and Pass [CLP10] for any notion of composable security:

[8]Disclaimer: We are neither advocating nor proving that SGX will serve as a platform to implement the protocols presented in the paper. We are merely observing the progress in the industry towards making tamper-proof hardware more viable.

Concurrent multi-instance security: The security properties related to local objects (including data and tokens) of the analyzed protocol itself should remain valid even when multiple instances of the protocol are executed concurrently and are susceptible to coordinated attacks against multiple instances. Almost all prior works in the tamper proof model do not specifically analyze their security in a concurrent setting. In other words, they only discuss UC-security of a single instance of the protocol. In particular, when executing protocols in the concurrent setting with tokens, an adversary could in fact transfer a token received from one execution to another and none of the previous works that are based on the $\mathcal{F}_{\mathrm{WRAP}}$-functionality accommodate transfers.

Modular analysis: Security of the larger overall protocols must be deducible from the security properties of its components. In other words, composing protocols should preserve security in a modular way. One of the main motivations and features in the UC-framework is the ability to analyze a protocol locally in isolation while guaranteeing global security. This does not only enable easier design but identifies the required security properties. The current framework proposed by Katz [Kat07] does not allow for such a mechanism.

The state-of-affairs regarding tamper-proof tokens leads us to ask the following question.

Does there exist a UC-formulation of tamper-proof hardware tokens that guarantees strong composability guarantees and allows for modular design?

Since the work of [Kat07], the power of hardware tokens has been explored extensively in a long series of works, especially in the context of achieving UC-security (for example, [CGS08, MS08, GIS+10, DKM11, DMMN13, DKMN15b, CKS+14]). While the work of Katz [Kat07] assumed the stronger stateful tokens, the work of Chandran, Goyal and Sahai [CGS08] was the first to achieve UC-security using only stateless tokens. In this work we will focus only on the weaker stateless token model. In the tamper-proof model with stateless tokens, as we argue below, the issue of minimal assumptions and round-complexity have been largely unaddressed. The work of Chandran et al. [CGS08] gives an $O(\lambda)$-round protocol (where λ is the security parameter) based on enhanced trapdoor permutations. Following that, Goyal et al. [GIS+10] provided a (problematic, as explained below) $O(1)$-round construction based on Collision-Resistant Hash Functions (CRHFs). The work of Choi et al. [CKS+14], extending the techniques of [GIS+10] and [DKM11], establishes the same result and provide a five-round construction based on CRHFs. See Section 3.1.3 for detailed related work.

All previous constructions require assumptions stronger than one-way functions (OWFs), namely either trapdoor permutations or CRHFs. Thus as a first question, we investigate the minimal assumptions required for token-based secure computation protocols. The works of [GIS+10] and [CKS+14] rely on CRHFs for realizing statistically-hiding commitment schemes. Towards minimizing assumptions, both these works, originally considered a variant of their respective protocols where they replace the construction of the statistically-hiding commitment scheme based on CRHFs to the one based on OWFs [HHRS15] to obtain UC-secure protocols under minimal assumptions (See Theorem 3 in [GIS+10] and Footnote 7 in [CKS+14]). While analyzing the proof of this variant in the work of [GIS+10], we found an issue in the original construction based on CRHFs.[9] More recently, the authors of [CKS+14] have removed this observation in their updated version of the original work (see [CKS+13]).[10] Given the state of affairs, our starting point is to address the following fundamental question regarding tokens that remains open.

> *Can we construct tamper-proof UC-secure protocols using stateless tokens assuming only one-way functions?*

A second important question that we address here is:

> *What is the round complexity of UC-secure two-party protocols using stateless tokens assuming only one-way functions?*

We remark here that relying on black-box techniques, it would be impossible to achieve non-interactive secure computation even in the tamper proof model as any such approach would be vulnerable to a residual function attack.[11] This holds even if we allow an initial token exchange phase, where the two parties exchange tokens (that are independent of their inputs). Hence, the best we could hope for is two rounds.

(G)UC-secure protocols in the multi-party setting. In the UC framework, it is possible to obtain UC-secure protocols in the MPC setting by first realizing the UC-secure oblivious transfer functionality (UC OT) in the two-party setting and then combining it with general compilation techniques (e.g., [Kil88, CLOS02, IPS08, LPV12] to obtain UC-secure multi-party computation protocols. First, we remark that specifically in the stateless tamper-proof token model, prior works fail to consider multi-versions of the OT-functionality while allowing transferability of tokens which is important in an MPC setting.[12] As such, none of the previous works explicitly study the round complexity of multi-party protocols in the tamper proof model (with stateless tokens), we thus address the following question.

> *Can we obtain round-optimal multi-party computation protocols with (G)UC-security in the tamper proof model?*

Unidirectional token exchange. Consider the scenario where companies such as Amazon or Google wish to provide an *email spam-detection* service and users of this service want to keep their emails private (so as to not have unwanted advertisements posted based on the content of their emails). In such a scenario, it is quite reasonable to assume that Amazon or Google have the infrastructure to create tamper-proof hardware tokens in large scale while the clients

[9] We remark that our observation *only* affects one particular result in [GIS+10], namely, realizing the UC-secure oblivious transfer functionality based on CRHFs and stateless tokens.

[10] In private communication, the authors of [CKS+13] explained that the variant that naively replaces the commitment with one based on one-way functions might be vulnerable to covert attacks.

[11] Recall that this attack allows the recipient of the (only) message to repeatedly evaluate the function on different inputs for a fixed sender's input.

[12] We remark that the work of [CKS+14] considers multiple sessions of OT between a single pair of parties. However, they do not consider multiple sessions between multiple pairs of parties which is required to realize UC-security in the multiparty setting.

cannot be expected to create tokens on their own. Most of the prior works assume (require) that both parties have the capability of constructing tokens. When relying on non-black-box techniques, the work of [CKS+14] shows how to construct UC-OT using a single stateless token and consequently requires only one of the parties to create the token. The work of Moran and Segev in [MS08] on the other hand shows how to construct UC-secure two-party computation via a black-box construction where tokens are required to be passed only in one direction, however, they require the stronger model of stateful tokens. It is desirable to obtain a black-box construction when relying on stateless tokens. Unfortunately, the work of [CKS+14] shows that this is impossible in the fully concurrent setting. More precisely, they show that UC-security is impossible to achieve for general functionalities via a black-box construction using stateless tokens if only one of the parties is expected to create tokens. In this work, we therefore wish to address the following question:

> *Is there a meaningful security notion that can be realized in a client-server setting relying on black-box techniques using stateless tokens where tokens are created only by the server?*

2.1.3.2 Results in the static setting

As our first contribution, we put forth a formulation of the tamper-proof hardware as a "global" functionality that provides strong composability guarantees. Towards addressing the various shortcomings of the composability guarantees of the UC-framework, Canetti et al. [CDPW07] introduced the Global Universal Composability (GUC) framework which among other things allows to consider global setup functionalities such as a global reference string, and more recently the global random oracle model [CJS14]. In this work, we put forth a new formulation of tokens in the GUC-framework that will satisfy all our desiderata for composition. Furthermore, in our formulation, we will be able to invoke the GUC composition theorem of [CDPW07] in a modular way. A formal description of the $\mathcal{F}_{\text{gWRAP}}$-functionality can be found in Figure 5.2 and more detailed discussion is presented in Section 5.2.

In the two-party setting we resolve both the round complexity and computational complexity required to realize GUC-secure protocols in the stronger $\mathcal{F}_{\text{gWRAP}}$-hybrid stated in the following theorem:

Theorem 2.1.5. [HPV16] Assuming the existence of OWFs, there exists a two-round protocol that GUC-securely realizes *every (well-formed) two-party functionality* in the global tamper-proof model assuming stateless tokens against any static, malicious adversary. Moreover, it only makes black-box use of the underlying OWF.

As mentioned earlier, any (black-box) non-interactive secure computation protocol is vulnerable to a residual function attack assuming stateless tokens. Therefore, the best round complexity we can hope for black box constructions assuming (stateless) tamper-proof tokens is two, achieved by our results. In concurrent work [DKMN15a], Dottling et al. show how to obtain UC-secure two-party computation protocols relying on OWFs via non-black-box techniques.

In the multi-party setting, our first theorem follows as a corollary of our results from the two-party setting.

Theorem 2.1.6. [HPV16] Assuming the existence of OWFs, there exists a $O(d_f)$-round protocol that GUC-securely realizes *every multi-party (well formed) functionality* f in the global tamper-proof model assuming stateless tokens against any static, malicious adversary, where d_f is the depth of any circuit implementing f.

Furthermore, this construction relies on the underlying one-way function in a black-box manner. Next, we improve the round complexity of our construction to obtain the following theorem:

Theorem 2.1.7. [HPV16] Assuming the existence of OWFs and stand-alone semi-honest MPC, there exists a three-round protocol that GUC-securely realizes *every multi-party (well formed) functionality* in the global tamper-proof model assuming stateless tokens against any static, malicious adversary.

We remark that our construction is black-box in the underlying one-way function but unlike our previous theorem it relies on the code of the MPC protocol in a non-black-box way. In particular, underlying MPC protocols typically rely on semi-honest oblivious transfer and our construction is non-black-box in this assumptions. We could build semi-honest MPC from our oblivious transfer protocol based on tokens from OWFs, constructed to achieve Theorem 2.1.5, and eliminate the semi-honest MPC assumption in Theorem 2.1.7. Moreover, it is conceivable that one can obtain a round-optimal construction if we do not require it to be black-box in the underlying primitives.

Finally, in the client-server setting, we prove the following theorem:

Theorem 2.1.8. [HPV16] Assuming the existence of OWFs, there exists a two-round protocol that securely realizes *every two-party functionality* assuming stateless tokens in a client-server setting against any static, malicious adversary where the tokens are created only by the server. We also provide an extension where we achieve UC-security against malicious clients and sequential and parallel composition security against malicious servers.

In more detail, we provide straight-line (UC) simulation of malicious clients and standard rewinding-based simulation against malicious servers. Our protocols guarantee security of the servers against arbitrary malicious coordinating clients and protects every individual client executing sequentially or in parallel against a corrupted server. We believe that this is a reasonable model in comparison to the Common Reference String (CRS) model where both parties require a trusted entity to sample the CRS. Furthermore, it guarantees meaningful concurrent security that is otherwise not achievable in the plain model in two rounds.

2.1.3.3 Adaptive corruptions

In the tamper-proof model, where the parties are assumed to have the capability of creating tamper-proof hardware tokens the work of Goyal et al. [GIS+10] shows how to realize unconditional (and hence, adaptive) UC-security in the tamper-proof model assuming stateful tokens. However, when we consider the weaker and more realistic model of stateless tokens, there is no known construction of adaptively secure protocols. As mentioned above and under strong assumptions, namely existence of (subexponentially-hard) indistinguishability obfuscation (iO) of circuits, the works of [GP15, CGP15, DKR15] provided the first constant-round adaptively secure protocols in the CRS model.[13]

As such, the best known adaptively secure protocols require strong assumptions and often higher round complexity. Given the state of affairs, in this section, we are motivated by the following natural question concerning adaptive security:

> *Can we construct adaptive GUC-secure constant-round protocols under standard polynomial-time assumptions from minimal setup?*

Recall that concurrent security cannot be achieved without assuming some form of trusted setup [CF01, CKL03, Lin03a]. However, in many scenarios, it is impossible to agree on a *trusted entity.* Specifically, protocols in the literature that rely on a trusted setup are rendered completely insecure if the setup is compromised. In the absence of setup, concurrently secure protocols have to rely on relaxed notions of security. The most popular notion in this line of work is security with super-polynomial simulators (SPS) [Pas03, BS05a, PS04, LPV09] which is a relaxation of the traditional simulation-based notion that allows the simulator to run in super-polynomial time. All these constructions require super-polynomial security of the underlying cryptographic primitives. Breakthrough work by Canetti, Lin and Pass showed how to obtain SPS security from standard polynomial time assumptions [CLP10]. In the adaptive setting, the works of [BS05a, DMRV13, Ven14] show how to obtain adaptive UC-secure protocols with SPS under super-polynomial time assumptions. More recently, the work of [HV16] shows how to obtain an $O(\lambda^\epsilon)$ (for any constant $0 < \epsilon < 1$) round adaptive UC-secure protocol with SPS under standard polynomial time assumptions.

In addition, motivated by designing practical protocols in the concurrent setting, another approach taken by Canetti, Jain and Scafuro [CJS14] considers the Random Oracle Model (ROM) of Bellare and Rogaway [BR93]. In order to provide strong compositional guarantees, they introduce the Global ROM and show how to obtain UC-secure protocols in the static setting. Their construction is based on the Decisional Diffie-Hellman assumption (DDH). In this section, we are interested in addressing the following questions that remain open:

> *Can we construct UC-secure protocols in the Global Random Oracle Model from minimal general assumptions?, and*
> *Can we construct adaptive UC-secure protocols in the Global Random Oracle Model?*

2.1.3.4 Results on adaptive security for arbitrary corruptions

We answer all our questions in the affirmative. Furthermore, all our results will be presented in the stronger GUC-setting of [CDPW07] that provide strong(-er) compositional guarantees. We will rely on the work [HPV16] where we model tokens for the GUC-setting. In order to incorporate adaptive corruptions we will have to determine the precise capabilities of the adversary when it can also corrupt the creator of the token post-execution and know the actual code embedded in the token. The $\mathcal{F}_{\text{gWRAP}}$-functionality introduced in [HPV16] will be sufficient to capture the adversary's capabilities.

[13] The work of [DKR15] assumes only polynomially-hard indistinguishability obfuscation.

Our first result shows how to construct constant-round adaptive GUC-secure protocols in the *tamper-proof hardware* model assuming *only* stateless tokens and the existence of OWFs. More precisely, we obtain the following theorem.

Theorem 2.1.9. [HPV17] Assuming the existence of OWFs, there exists a constant-round protocol that GUC-securely realizes *the commitment functionality* in the global tamper-proof model assuming stateless tokens against any adaptive, malicious adversary.

Next, we extend the ideas in this protocol to obtain an adaptive GUC-secure protocol for the oblivious transfer functionality from OWFs.

Theorem 2.1.10. [HPV17] Assuming the existence of OWFs, there exists a constant-round protocol that GUC-securely realizes *the oblivious transfer functionality* in the global tamper-proof model assuming stateless tokens against any adaptive, malicious adversary.

Combining this protocol with the adaptive UC-secure protocol in the OT-hybrid of Ishai et al. [IPS08], we can obtain as a corollary an adaptive GUC-secure protocol in the $\mathcal{F}_{\text{gWRAP}}$-hybrid assuming only OWFs. However, this protocol will require $O(d_f)$ rounds where d_f is the depth of the circuit computing the evaluated function f. Our main contribution in this section is to reduce the round complexity and show how to realize any well-formed functionality in $O(1)$-rounds independent of the complexity of the function. Below, we state this main theorem.

Theorem 2.1.11. [HPV17] Assuming the existence of OWFs, there exists a constant-round protocol that GUC-securely realizes *every (well-formed) two-party functionality* in the global tamper-proof model assuming stateless tokens against any adaptive, malicious adversary.

In order to achieve our results, at the heart of our construction we present a new technique that equivocates Yao's garbled circuits. On a high-level the idea is to use tokens to enable evaluation of the garbled circuit in Yao's garbling technique [Yao86]. That basic intuition here is that we view the garbling technique as system of labels where evaluation can be performed by "multiplexer" tokens (MPLX) where for each gate given labels corresponding to the inputs, the MPLX picks the corresponding output label for a gate. This basic idea can be made to work in the static setting to construct a secure computation protocol. However, in the adaptive setting things get problematic. The simulator in the garbling technique relies on a "fake" garbled circuit where only the "active keys" are correctly embedded in the garbled tables for the evaluator.[14] In the adaptive setting, if the garbled circuit evaluator is corrupted at the beginning and the generator is corrupted at the end of the execution the simulator needs to reveal the fake garbling as a real garbling. This is not possible in the $\mathcal{F}_{\text{gWRAP}}$ modelling of the tokens as the simulator is not allowed to "program" the token after creation.

Instead, we solve this problem differently. We will *not* alter the honest generator's strategy. Instead, we modify the simulation strategy in order to equivocate the garble circuit as follows:

- We embed a key K to a symmetric encryption scheme in each gate token.

[14] Using the terminology of [LP09b], active keys are observed by the evaluator while evaluating the garbled circuit, while inactive labels are the labels that remain hidden during the evaluation.

- The token will be hardwired with three labels, $\ell_{\omega_1}, \ell_{\omega_2}$ and ℓ_{ω_3}, which will be the supposed active labels for this gate, and a random string r.

- On input ℓ_1, ℓ_2, the token will behave as follows: If $\ell_1 = \ell_{\omega_1}$ and $\ell_2 = \ell_{\omega_2}$ it will output ℓ_{ω_3}. This corresponds to what the evaluator can obtain prior to corrupting the generator.

- If either ℓ_1 or ℓ_2 is different from the hardwired labels, it attempts to do the following. It decrypts the label that is different using the key K to obtain a string z that it reads as (x, y). The token then evaluates the circuit assuming the generator's input is x and the evaluator's input is y to obtain the actual values in the wires ω_1 and ω_2 that are the inputs to this gate, say b_1 and b_2. With this information, the token internally assigns the bit b_1 to label ℓ_{ω_1} and b_2 to label ℓ_{ω_2} and $G(b_1, b_2)$ to ℓ_{ω_3} where $G \in \{\text{AND}, \text{XOR}\}$ is the gate function. Next, it outputs based on the following strategy:

  1. If $\ell_1 = \ell_{\omega_1}$ and $\ell_2 \neq \ell_{\omega_2}$ output ℓ_{ω_3} if $G(b_1, 1 - b_2) = G(b_1, b_2)$, and output $\mathsf{Enc}(K, (x, y); r)$ otherwise.
  2. If $\ell_1 \neq \ell_{\omega_1}$ and $\ell_2 = \ell_{\omega_2}$ output ℓ_{ω_3} if $G(1 - b_1, b_2) = G(b_1, b_2)$, and output $\mathsf{Enc}(K, (x, y); r)$ otherwise.
  3. If $\ell_1 \neq \ell_{\omega_1}$ and $\ell_2 \neq \ell_{\omega_2}$ output ℓ_{ω_3} if $G(1 - b_1, 1 - b_2) = G(b_1, b_2)$, and output $\mathsf{Enc}(K, (x, y); r)$ otherwise.

In essence, this strategy figures out what bits the active labels should be associated with, and outputs the labels correctly. Furthermore, the information required to figure out the association is passed along. While this high-level idea allows to "equivocate" the circuit, we need the encryption to satisfy some additional properties such as non-malleability and evasiveness. Note that the above strategy followed by the simulator does provide a fake code to be embedded in the token, but once the sender is corrupted post-execution the simulator reveals an honest looking code to the adversary which does not include any information about the fake code e.g., the secret key K.

Last but not least, as a corollary to our techniques, we also present the first adaptively secure protocols in the ROM with round complexity proportional to the depth of the circuit implementing the functionality. Our protocols are secure in the global ROM [CJS14]. More precisely, we obtain an adaptively secure UC-commitment scheme in the global ROM assuming only OWFs. In comparison, the protocol of [CJS14] achieves only static security and relies on the specific assumption of DDH.

2.1.3.5 Subsequent and future work

Relying on our technique of equivocating Yao's garbled circuits based on stateless tokens, a subsequent beautiful work by Canetti et al. [CPV16] shows how to equivocate garbled circuits removing the reliance on tokens and thus construct the first two-round 2PC protocol secure against any adaptive semi-honest corruption of both parties based on semi-honest adaptive OT in the plain model. Therefore, their results also improve upon our work [GP15] which is based on indistinguishability obfuscation. The work of [CPV16] also constructs constant round MPC protocols tolerating adaptive malicious corruption of all parties (and UC-secure protocols in the

CRS model). As mentioned in Section 2.1.2.2 we still leave open the question of two-round adaptive MPC protocols in the CRS model from standard assumptions.

In the hardware token model, we leave open the question of constructing, two-round adaptive MPC protocols in the static as well as adaptive settings.

2.2 Information-Theoretic Setting

IT secure cryptography provides unconditional security without the need for unproven complexity assumptions. The techniques used in IT secure protocols tend to be computationally much more efficient than the cryptographic machinery needed for computational security. Therefore, IT secure protocols are attractive from a practical point of view, however they seem to require a lot of interaction. The protocols behind these results require a number of communication rounds that is proportional to d_f where d_f is the depth of any circuit implementing the evaluated functionality f. Moreover, these protocols offer communication complexity, which is proportional to the size of the circuit. Whether we can have constant round protocols and/or communication complexity much smaller than the size of the circuit and still be computationally efficient (polynomial-time) in the circuit size of the evaluated function is a long-standing open problem. Note also that if we give up on being efficient in the circuit size, then there are unconditionally secure and constant round protocols for any function [IK00] (which will, however, be very inefficient in general with respect to the computation). Moreover, there are works that apply to special classes of circuits (e.g., constant-depth circuits [BI05]) or protocols that require an exponential amount of computation [BFKR91, NN01] and exponential storage complexity [IKM$^+$13].

2.2.1 Gate-by-gate Design Pattern of Computation

The fact that existing IT secure protocols (which are efficient in the circuit size of the function) have large round and communication complexity is a consequence of the fact that all such protocols follow the same *typical* "gate-by-gate" design pattern: Initially all inputs are secret-shared among the players. Then, for each gate in the circuit, where both its inputs have been secret-shared, we execute a subprotocol that produces the output of the gate in a secret-shared form. The protocol maintains as an invariant that for all gates that have been processed so far, the secret-sharing of the output value is of the same form used for the inputs (so we can continue processing gates) and is appropriately randomised such that one could open this sharing while revealing only that output value. As a result, it is secure to reveal/open the final outputs from the circuit.

For all known constructions which are efficient in the circuit size of the function, it is the case that multiplication gates require communication to be processed (while addition/linear gates usually do not). The number of rounds is at least the (multiplicative) depth of the circuit, and the communication complexity is $\Omega(ns)$ for a circuit of size s (the size being measured as the number of multiplication gates) in the worst case for privacy threshold $t < n/3$ and $t < n/2$ in the works of [DN07, BTH08] and [BFO12, GIP$^+$14, GIP15], respectively. Note that protocols that tolerate a sub-optimal number of corrupted parties (e.g., $t < 0.49n$) and are based on packed secret-sharing techniques can reduce the amortised cost of multiplications if they can be parallelised [DIK$^+$08, IPS09, DIK10, GIP15]. These techniques do not apply to all circuits, in particular not to "tall and skinny" circuits whose multiplicative depth is comparable to their size. In addition, they can at best save an $O(n)$ factor in communication and computational work. The situation is essentially the same for recent protocols that are designed for dishonest majority in the preprocessing model [DPSZ12, NNOB12] (except that amortization based on packed secret-sharing does not apply here due to the dishonest majority setting). See Section 3.2 for a detailed related work.

In this paper, we ask a very natural question for unconditionally secure protocols:

> *Is it really inherent that the* typical *gate-by-gate approach to secure computation requires communication for each multiplication operation?*

2.2.1.1 Lower bounds for gate-by-gate protocols

We answer the question in the affirmative. In particular, we show the following result.

Theorem 2.2.1. [DNPR16] (Informal). Any gate-by-gate n-party protocol securely realizing any functionality f must communicate $\Omega(n)$ bits for every multiplication gate of the circuit implementing the evaluated functionality f in in the honest majority setting as well as in the dishonest majority setting with an initial preprocessed phase. [15]

In the honest majority setting, we also show the following supplementary bounds:

- We show that one cannot obtain a communication complexity lower bound that also grows with the *size* of the underlying field of the inputs of the evaluated function.
- For a constant number of players, amortization over several multiplication gates does not save on the computational work, and – in a restricted setting – we show that this also holds for communication.

All our lower bounds are met up to a constant factor by known protocols that follow the typical gate-by-gate paradigm. Our results imply that a fundamentally new approach must be found in order to improve the communication complexity of known protocols, such as [GMW87, BOGW88, CCD88, DPSZ12], to name just a few.

2.2.1.2 Subsequent and future work

It is highly interesting to investigate whether our lower bounds hold for any protocol, departing from the current gate-by-gate design of protocols in the literature. This question is long-standing and seems challenging since answering in the affirmative rules out the possibility of constructing something analogous to fully homomorphic encryption in the IT setting. On the other hand, if

[15] The result holds in the dishonest majority if the target output secret-sharing scheme is of a certain type that includes the simple additive secret-sharing scheme. If we put no restrictions on the target output secret-sharing scheme, the results get a bit more complicated (see details in Section 7.4.2).

it is indeed possible to construct constant round protocols in the IT setting, it is challenging to investigate novel approaches towards this goal.

As mentioned above, giving up on being efficient in the circuit size, there are IT secure and constant round protocols for any function [IK00], however they are extremely inefficient in general with respect to the computation. That said, they are efficient only for NC^1/ Logspace L computations. Without loss of generality, all these protocols fall into the IT Randomized Encoding (RE) model of computation. Note that there no known lower bounds on poly-size IT randomized encodings. For example, it is not known whether we can have poly-size IT randomized encoding for general poly-time functions (or efficient constant-round IT protocols). It is highly interesting to obtain lower bounds or construct poly-size IT randomized encodings.

Last but not least, we mention a subsequent work on secure computation over large inputs for specific functionalities. Secure computation on big data calls for MPC protocols that have sub-linear communication complexity in the input length. Quite unexpectedly, the works of [AMP10, BS05b, SV15] have shown that in several natural instances of secure computation on big data, such as computing the median and several combinatorial optimization problems, one can enjoy the best of both worlds: sub-linear communication complexity and low concrete computational overhead. In a recent work [BIP17], we put forward a framework for separating "practical" sublinear communication protocols from "impractical" ones. We show that while the previous protocols of [AMP10, BS05b, SV15] are indeed classified as being "practical" in this framework, this is provably not the case for some natural variations of the tasks realized by these protocols. For instance, while the median functionality enjoys sub-linear communication complexity [AMP10, SV15], we show that a variant of the median functionality – in which only one party learns the function output – requires linear communication complexity. This suggests that practical sub-linear cryptography is more sensitive to the precise formulation of the task at hand than it may seem.

2.2.2 Communication Complexity of Semi-honest vs. Malicious Gate-by-gate Protocols

The difficulty of designing MPC protocols depends largely on the power of the adversary. An important distinction is between MPC protocols that offer security against *passive* (or semi-honest) adversaries, who follow the protocol's specification but try to learn information from messages they receive, and security against *active* (or malicious) adversaries, who are allowed to deviate from the protocol's specification in arbitrary ways. A central problem in cryptography is that of converting protocols that offer security against passive (or semi-honest) adversaries into ones that offer security against active (or malicious) adversaries. This problem has been the topic of a large body of work in the area of secure MPC. Despite these efforts, there are still big efficiency gaps between the best protocols in these two settings.

A common paradigm for designing actively secure MPC protocols is to start with a passively secure protocol and then convert it into an actively secure protocol. Some relevant techniques include general-purpose "GMW-style" compilers that employ zero-knowledge proofs [GMW87, CLOS02], ad-hoc protocols for verifying the correct execution of subprotocols [BOGW88, CCD88], cut-and-choose techniques [LP07, LP11b], LEGO style cut-and-choose

[NO09, FJN+13, FJNT15], or "MPC in the head" [IKOS07, IPS08]. These techniques typically involve a significant overhead.

A different technique, which in some cases provides better results, was recently proposed independently by Genkin et al. [GIP+14] and Ikarashi et al. [IKHC14]. In particular, [GIP+14] suggested a new paradigm for efficiently transforming passive-secure MPC protocols into active-secure ones. The authors observe that in several known passively secure MPC protocols for evaluating arithmetic circuits, the effect of any arbitrary active attack on the protocol is limited to an *additive attack* on the evaluated circuit wires. That is, everything that an adversary can achieve by attacking the real protocol for evaluating C he could have also achieved by attacking directly the evaluated circuit C in which he can *blindly* add a field element of his choice to each wire. We refer to such protocols as *additively corruptible protocols*. To protect against such attacks, the original circuit is replaced by a so-called *AMD circuit* $\overline{\mathsf{C}}$ – a randomized circuit which is functionally equivalent to C but additionally offers resistance against additive attacks with constant multiplicative overhead to the size of C. [16]

Our motivating observation is that in the most efficient known IT MPC protocols that tolerate a slightly sub-optimal number of corrupted parties (e.g., $t < 0.49n$) based on packed secret sharing, it is *not* the case that general attacks reduce to additive attacks. As mentioned above, these protocols replace the standard secret sharing used in optimally resilient protocols by a more efficient packed secret sharing [FY92] technique, and as a result provide better asymptotic efficiency. We observe that the ideal attack corresponding to an active adversary attacking these protocols can include a limited form of linear combinations that combine multiple wire values of the evaluated circuit. In Section 7.5 we provide a simple example to illustrate the necessity of extending the attack model to linear. As a result, the techniques of [GIP+14, IKHC14] do not apply to such protocols. We refer to such protocols as *linearly corruptible protocols*.

A second disadvantage of the techniques of [GIP+14, IKHC14] is that they are tailored to specific protocols. In particular, the part of the analysis that maps general attacks to additive attacks is done in an ad-hoc way per protocol without a unified framework that captures all additively corruptible protocols.

In addition, we point out that the concrete efficiency of DIK-style protocols [DIK10, BELO14, IPS09] based on packed secret sharing involves prohibitively large constants when applied with near-optimal security threshold. Indeed, the threshold obtained directly by [DIK10] is $t < n/4$ which is quite far from the optimal bound of $n/2$. To improve on this threshold, a general technique due to Bracha [Bra87] is applied to boost the resilience. The basic idea is that a constant-size committee runs an optimally resilient protocol to emulate the role of each server in the low-threshold protocol. While this technique can be implemented with a constant multiplicative overhead, this constant is very large. See Table 3.1 for a detailed comparison of IT MPC protocols.

The above state of affairs leads to the following general question.

> *Can we achieve malicious IT MPC protocols based on packed secret sharing that match the communication complexity of their semi-honest variants?*

[16]The work of [IKHC14] does not explicitly construct AMD circuits, but implicitly relies on a simple construction of AMD circuits that tolerates a restricted class of additive attacks which suffices in some cases.

2.2.2.1 Results for malicious gate-by-gate protocols

We answer the above question in the affirmative and show how to extend the AMD circuit methodology to so-called *secure* SIMD *circuits*[17], which offer protection against the more general linear class of attacks[18] and extend the model considered in [GIP+14] from additive attacks to general algebraic attacks. In a nutshell we obtain the following general results [GIP15]:

- We construct SIMD circuits secure against linear attacks with constant multiplicative overhead to the size of the original circuit for arithmetic circuits.
- We apply secure SIMD circuits to obtain several asymptotic and concrete efficiency improvements over the current state of the art. In particular, we improve the additive per-layer overhead of the current best protocols from $O(n^2)$ to $O(n)$. Table 3.1 shows the achieved overheads. In a nutshell, in this work we close the efficiency gap between passively secure versus actively secure IT MPC protocols based on packed secret sharing.
- We obtain the first protocols based on packed secret sharing that "natively" achieve near-optimal security without incurring the high concrete cost of Bracha's committee-based security amplification method.
- Our analysis is based on a new modular framework for proving reductions from general attacks to algebraic attacks. This framework allows us to reprove previous results in a conceptually simpler and more unified way, as well as obtain our new results.

Maliciously Secure MPC Protocols from SIMD Circuits. In a nutshell, our approach for constructing actively secure MPC protocols is as follows. We present a general *framework* and prove that any passively secure protocol π, satisfying the framework's requirements is indeed additively or linearly corruptible depending on whether π uses packed secret sharing or not. Next, in order to transform any passively secure protocol for evaluating a circuit C, which meets the framework's requirement, into an actively secure protocol, we apply the *same* passive protocol on a different circuit $\overline{\mathsf{C}}$ which is essentially the secure version of C (either additive-attack secure or linear-attack secure, based on whether C is an SIMD circuit). We thus transfer the responsibility of handling the consequences resulting from an active adversary deviating from the protocol, to $\overline{\mathsf{C}}$. Since any deviations made by an active adversary correspond to an additive (or linear, for the case of SIMDcircuit) attack on C, $\overline{\mathsf{C}}$ is able to detect these deviations and abort the computation if necessary.

Our framework for proving that a passively secure protocol π is in fact additively or linearly corruptible, consists of three steps. We point out that while these steps modify the original protocols, are only a *thought-experiment* used to prove the main claim about the effect of an

[17]An SIMD circuit is a generalization of standard arithmetic circuits, composed by ℓ-gates which get as input two wire *bundles* of size ℓ and output a wire output *bundle* of size ℓ obtained by performing ℓ *point-wise* multiplications, additions and subtractions in parallel. Thus, SIMD circuits simultaneously evaluate ℓ copies of the same arithmetic circuit, on different inputs.

[18]A *linear* attack on an SIMD circuit changes the computation of a *multiplication* ℓ-gate by adding to each wire in the gate's output bundle a linear function $\phi : \mathbb{F}^{2\ell} \to \mathbb{F}^{\ell}$ of all the wires in the gate's two input bundles. We note that for the case where $\ell = 1$ linear attacks are equivalent to additive attacks.

active adversary on the underlying circuit that parties try to evaluate. Once a passively secure protocol is proven to satisfy the framework's requirements, the only *real* modification required to the protocol in order to transform it to an actively secure protocol, is to execute it on an additive-attack or linear-attack secure circuit.

Step 1: Protocol randomization. In order to convert an active adversary controlling a set of parties $\mathcal{T}$ to an additive attack, we first transform a protocol π to another protocol $\pi^{\mathcal{T}}$ such that all the messages $m_{\overline{\mathcal{T}},\mathcal{T}}$ sent by the parties in $\overline{\mathcal{T}}$ (the honest parties) to the parties in $\mathcal{T}$ (except during the last communication round) syntactically depend only on the randomness of π. In particular, we require that $m_{\overline{\mathcal{T}},\mathcal{T}}$ does not depend on the inputs $\mathbf{x}_{\overline{\mathcal{T}}}$ of the parties in $\overline{\mathcal{T}}$ or on the messages that the parties in $\overline{\mathcal{T}}$ receive during the protocol. In such case we say that $\pi^{\mathcal{T}}$ is $\mathcal{T}$-randomized.

We first show that for many natural MPC protocols, for any set of parties $\mathcal{T}$, such that $|\mathcal{T}|$ is below the privacy threshold of a protocol, it is possible to construct a $\mathcal{T}$-randomized protocol, $\pi^{\mathcal{T}}$, such that any deviation from π made by an active adversary has the same effect on the joint distribution of its view together with the output of the honest parties, as performing the same deviation from $\pi^{\mathcal{T}}$. Thus we proved that anything that the adversary achieves attacking π, he can also achieve attacking $\pi^{\mathcal{T}}$. In this case we say that $\pi^{\mathcal{T}}$ is $\mathcal{T}$-equivalent to π.

Notice that $\mathcal{T}$-randomization requirement is stronger than privacy. This is since $\mathcal{T}$-randomization requires that the *values* of $m_{\overline{\mathcal{T}},\mathcal{T}}$ do not depend on the inputs of the parties in $\overline{\mathcal{T}}$ or on messages that parties in $\overline{\mathcal{T}}$ received during the protocol as opposed to privacy which makes a similar requirement on the *distribution* of $m_{\overline{\mathcal{T}},\mathcal{T}}$. See Step 2 for the necessity of the $\mathcal{T}$-randomization requirement.

In the the full version we present the $\mathcal{T}$-randomized and $\mathcal{T}$-equivalent versions $\mathsf{GMW}^{\mathcal{T}}$, $\mathsf{DN}^{\mathcal{T}}$ and $\mathsf{DIK}^{\mathcal{T}}$ of the passively secure GMW, DN and DIK [GMW87, DN07, DIK10] protocols, respectively. In addition, we also prove that these protocols are $\mathcal{T}$-equivalent to their non-randomized versions.

Step 2: From general adversaries to additive attacks. In this step we reduce any general adversary controlling a set of parties $\mathcal{T}$ and attacking a $\mathcal{T}$-randomized protocol π, to an *additive* attack on the protocol circuit C_π. The circuit C_π is a direct implementation of the arithmetic operations performed by π. C_π gets as input the inputs $\mathbf{x}$ of the parties in π as well the randomness $\mathbf{r}$ used in π. It then evaluates π on inputs $(\mathbf{x};\mathbf{r})$ and outputs the outputs of all the parties following an execution of $\pi(\mathbf{x};\mathbf{r})$.

Since π is $\mathcal{T}$-randomized we can completely simulate from the randomness $\mathbf{r}$ for π and from the inputs $\mathbf{x}_{\mathcal{T}}$ of the parties in $\mathcal{T}$, the view $\widetilde{u}_{\mathcal{T}}$ (except during the last communication round) of the parties in $\mathcal{T}$. Next, we determine the additive attack on C_π corresponding to an adversary Adv controlling the parties in $\mathcal{T}$ as follows. We first honestly simulate the parties in $\mathcal{T}$ on their view $\widetilde{u}_T$ and obtain the messages $\widetilde{m}_{\mathcal{T},\overline{\mathcal{T}}}$ sent by the parties in $\mathcal{T}$ to the parties in $\overline{\mathcal{T}}$ during an honest execution of π. Next, we invoke Adv on the view $\widetilde{u}_{\mathcal{T}}$ and obtain the messages $\widetilde{m}^{\mathsf{Adv}}_{\mathcal{T},\overline{\mathcal{T}}}$ sent by the parties in $\mathcal{T}$ to the parties in $\overline{\mathcal{T}}$ during a real execution of π in the presence of Adv. Finally we determine the additive attack $\mathbf{A}$ on C_π by computing $\mathbf{A} \leftarrow \widetilde{m}^{\mathsf{Adv}}_{\mathcal{T},\overline{\mathcal{T}}} - \widetilde{m}_{\mathcal{T},\overline{\mathcal{T}}}$.

Since π is $\mathcal{T}$-randomized, it is the case that inside C_π under the additive attack $\mathbf{A}$ it holds that $\widetilde{m}^{\mathsf{Adv}}_{\mathcal{T},\overline{\mathcal{T}}} = \widetilde{m}_{\mathcal{T},\overline{\mathcal{T}}} + \mathbf{A}$, for any input $\mathbf{x}_{\overline{\mathcal{T}}}$ of the parties in $\overline{\mathcal{T}}$ as well as for any messages that these parties receive during π. We thus correctly simulate, inside C_π, the effect of Adv on π. Notice that this is not necessary true in case π is $\mathcal{T}$-private since for any selection of the randomness $\mathbf{r}$, the specific values of the messages sent by the parties in $\overline{\mathcal{T}}$ to Adv might depend on their inputs $\mathbf{x}_{\overline{\mathcal{T}}}$ to π. Since $\mathbf{x}_{\overline{\mathcal{T}}}$ is not known to the simulator, it cannot generate the correct view $\widetilde{u}_{\mathcal{T}}$ required in order to compute $\widetilde{m}^{\mathsf{Adv}}_{\mathcal{T},\overline{\mathcal{T}}}$ and $\widetilde{m}_{\mathcal{T},\overline{\mathcal{T}}}$.

In the full version we prove that $\mathbf{A}$ indeed correctly simulates in C_π any deviation made by Adv during a real execution of π. Combining the results of this step together with the results of the previous step, we obtain that any arbitrary behavior of an active adversary, controlling a set of parties $\mathcal{T}$ and attacking the passively secure $\mathsf{GMW}, \mathsf{DN}$ and DIK protocols, has the same effect as mounting additive attacks on the circuits $\mathsf{C}_{\mathsf{GMW}^{\mathcal{T}}}, \mathsf{C}_{\mathsf{DN}^{\mathcal{T}}}$ and $\mathsf{C}_{\mathsf{DIK}^{\mathcal{T}}}$, respectively.

Step 3: Translate attacks on C_π to attacks on C. We translate additive attacks mounted on C_π to equivalent linear or additive attack on C. In the full version we present the notion of *homomorphic circuits* and prove that if a circuit C' is homomorphic to a circuit C then for any additive attack $\mathbf{A}'$ on C' there exists an equivalent additive attack $\mathbf{A}$ on C such that $\mathsf{C}^{\mathbf{A}}(\mathbf{x}) = \mathsf{C}'^{\mathbf{A}'}(\mathbf{x})$, for all $\mathbf{x}$. Next, extending the notion of circuit homomorphism to SIMD circuits, in the full version we define the notion of ℓ-homomorphic circuits and prove that if a circuit C' is ℓ-homomorphic to an SIMD circuit C, then for any additive attack $\mathbf{A}'$ on C' there exists an equivalent *linear* attack on C such that $\mathsf{C}^{\mathbf{L}}(\mathbf{x}) = \mathsf{C}'^{\mathbf{A}'}(\mathbf{x})$ for all $\mathbf{x}$.

In the full version of [GIP15] we apply the above transformations on the arithmetic version of the passively secure GMW protocol, proving that it is additively corruptible. Next, we apply the above transformations to the passively secure DN and DIK protocols, proving that these protocols are additively and linear corruptible, respectively. In the full version, we also present our techniques for additive-attack and and linear-attack security for securing circuits against additive and linear attacks.

2.2.2.2 Subsequent and future work

Our work [GIP15] closes the communication efficiency gap between passively secure versus actively secure IT MPC gate-by-gate protocols by constructing SIMD circuits secure against linear attacks with *constant* multiplicative overhead to the size of the original evaluated circuit for arithmetic circuits. We leave open the question of constructing secure AMD circuits with constant multiplicative overhead to the size of the original evaluated circuit for *boolean circuit*. Such a result would close the communication efficiency gap also for protocols based on RE and Yao's garbled circuits even in the computational setting. Current constructions require polylogarithmic (in the security parameter) overhead [IKL+13].

A recent work by [GIW16] presented AMD circuits secure against additive attacks for *boolean circuits* but with polylogarithmic (in the security parameter) multiplicative overhead to the size of the original circuit.

2.3 Secure RAM Computation

The tremendous growth of cloud computing and outsourced computation has made the protection of computations executed over untrusted agents a central focus in cryptography. Cryptographic protocols that address these issues have traditionally been developed in the circuit model of computation. However, a lot of recent work focuses on improving the overhead of such protocols when applied to realistic computations that are (as most programs are) designed for machines with RAM, departing from the circuit model. A major source of inefficiency that arises in the context of secure RAM computation is that of converting a program into a circuit. There are a lot of works on secure cloud computing but most notably Garbled RAM (GRAM) schemes [LO13b], the RAM analogue of Yao's garbled circuits, overcome the above considerations by garbling a RAM program directly without converting it into a circuit.

Traditional cryptographic protocols for secure "big data" computations require communication complexity and computational overhead that scale with the size of the input dataset. Works on oblivious RAM [Gol87, Ost90], fully-homomorphic encryption (FHE) [Gen09, BV11b, BV11a, GSW13] and secure RAM computation [OS97, GKK+12, LO13b, LO13a, GHL+14, WHC+14, GLOS15, GLO15, GGMP16] offer cryptographic solutions in an effort to reduce these overheads. Non-interactive protocols based on FHE generally have minimal communication complexity but incur a prohibitively large computational overhead. However, protocols for the RAM model generally have minimal computational overhead, but lack in terms of communication efficiency, especially in the multi-party setting. Is it possible to achieve the best of both worlds? In our work [CDG+17] we introduce the new notion of Laconic Oblivious transfer that helps to strike a balance between two seeming opposing goals.

Results in Secure RAM computation. In this section we only mention our results in [CDG+17]. In particular, we introduce a novel technique for secure computation over large inputs. Based on the Decisional Diffie-Hellman (DDH) assumption, we provide a new Oblivious Transfer protocol with a laconic receiver. In particular, the laconic OT allows a receiver to commit to a large database input D (of length m) via a short message. Subsequently, a single short message by a sender allows the receiver to learn s_{D_i}, where s_0, s_1 and $i \in [m]$ are dynamically chosen by the sender. All prior constructions of OT required the receiver's message to grow with m. At the technical core of this construction is a novel use of somewhere statistically binding hashing in conjunction with hash proof systems.

Such an OT is suited to secure computation over large data. More specifically, we show applications of laconic OT to non-interactive secure computation and homomorphic encryption for RAM programs.

- *Non-Interactive Secure Computation à la [IKO+11] on Large Inputs:* Can a receiver publish a (small) encoding of her large confidential database D so that any sender, holding a secret input x, can reveal the output $f(x, D)$ to the receiver by sending her a single message? Using laconic OT, we present the first solution to this problem. In our construction, the size of receiver's published message is independent of the size of her database D.

Furthermore, in case f can be computed using a RAM program P, then the size of the sender's message (and computational cost of both the sender and the receiver) in our construction grows only with the running time of the RAM program. In this case, the receiver R learns nothing more than the output $P^D(x)$ and the locations in D touched during the computation.

- *Multi-Hop Homomorphic Encryption à la [GHV10] for RAM programs:* Consider a party Alice with a private input x and a sequence of clients $Q_1, Q_2, \ldots$ with private databases $D_1, D_2, \ldots$, respectively. The clients ship encrypted versions of their private databases to Alice. Alice encrypts her private input x under her public key and sends the obtained ciphertext to Q_1. Next, clients $Q_1, Q_2, \ldots$ in an arbitrary order compute arbitrary RAM programs on the provided ciphertext, while accessing their private databases. Finally, Alice wants to be able to decrypt the evaluated ciphertext using her secret key, while preserving the persistent nature of the database that gets dynamically updated as multiple programs are sequentially executed. Note that computation itself happens at decryption time, and that is when Alice uses the encrypted versions of the clients databases. Using laconic OT, we obtain a construction of such a homomorphic encryption scheme for which the computational costs and the size of the ciphertexts grow only with the running time of the programs computed by the clients.

2.4 Roadmap

In Chapter 3 we provide a detailed survey on the communication complexity of secure computation works in the literature in the computational setting as well as in the information-theoretic setting.

In Part II we present our results in the computational setting. More specifically, secure computation has been achieved under various models including no trusted setup (i.e. the plain model), trusted setup (i.e. the CRS model) and decentralized setup (i.e. the tamper-proof hardware model). In Chapter 4 we present our results [GMPP16] in the plain model which are updated to include the robustness requirement for the non-malleable commitments in order to achieve our two-party protocols. Due to space constraints we refer to the full version of [GMPP16] for our multi-party protocols. In Chapter 5 we present our two-party protocols in the tamper-proof hardware model. More specifically, we present our results as stated in [HPV16] with minor modifications for the static setting. Due to space constraints we refer to the full version of [HPV16] for our results in the multi-party setting and to [HPV17] for our results in the adaptive setting. In Chapter 6 we present our adaptively secure multi-party protocols in the CRS model as stated in our work in [GP15] with minor modifications. Moreover, in Section 6.2 we present our general construction of equivocal FHE as stated in [DPR16] with minor modifications. For our multi-party computation protocols secure against all-but-one corruptions we refer to the full version of [DPR16].

In Part III we present our lower bounds in the information-theoretic setting. In particular, we present our results as presented in [DNPR16] with minor modifications and due to space constraints we refer to [GIP15] for our constructions of information-theoretic multi-party computation protocols.

Chapter 3

Related Work

3.1 Computational Setting

The round complexity of secure computation has a rich and long history. We mention the results in the computational setting (that are most relevant to this thesis) in the plain model, CRS model and tamper-proof hardware model. Note that our works in [GMPP16, HPV16, GP15, DPR16, DNPR16] provide related work sections however we expand these sections and provide a detailed overview of the state-of-the-art in this thesis.

3.1.1 Related Work in the Plain Model

For the computational setting and the special case of two party computation, the semi-honest secure protocol of Yao [Yao82, Yao86, LP11b] consists of only three rounds (see Section 6.1.3). An alternative approach using randomized polynomials was also given by [IK00, AIK05]. For malicious security [1], the first constant round protocol based on GMW was presented by Lindell [Lin01]. [2] Ishai, Prabhakaran, and Sahai [IPS08] presented a different approach which also results in a constant round protocol.

The problem of the exact round complexity of two party computation was studied in the beautiful work of Katz and Ostrovsky [KO04] who provided a 5 round protocol for computing any two-party functionality. They also ruled out the possibility of a four round protocol for coin-flipping, thus completely resolving the case of two party. Recently Ostrovsky, Richelson and Scafuro [ORS15] constructed a different 5-round protocol for the general two-party computation by only relying on *black-box* usage of the underlying trapdoor one-way permutation.

The standard setting for two-party computation does not consider simultaneous message exchange channels, and hence the negative results for the two-party setting do not apply to the multi-party setting where simultaneous message exchange channels are standard. Prior to our work [GMPP16], the case of the two-party setting in the presence of a simultaneous message exchange channel was not explored in the context of the exact round complexity of secure computation.

For the multi-party setting, the exact round complexity has remained open for a long time. The work of [BMR90] gave the first constant-round non black-box protocol for honest majority (improved by the black-box protocols of [DI05, DI06]). Katz, Ostrovsky, and Smith [KOS03], adapted techniques from [DDN91b, Bar02, BMR90, CLOS02] to construct the first asymptotically round-optimal protocols for any multi-party functionality for the dishonest majority case based on sub-exponential hardness assumptions ([KOS03] achieves logarithmic round-complexity based on polynomial-time assumptions). More specifically, apart from enhanced

[1]From here on, unless specified otherwise, we are always in the malicious setting by default.

[2]Versions of [LP07, LP11b, Lin13, HKE13] in the plain model construct two-party constant round malicious protocols based on cut-and-choose techniques.

trapdoor permutations their protocol relies on the assumption of sub-exponentially secure dense cryptosystems as well as sub-exponentially secure collision resistant hash functions. Pass [Pas04] constructed a constant-round bounded-concurrent protocol based on standard polynomial-time assumptions, i.e. enhanced trapdoor permutations and collision resistant hash functions. The constant-round protocols of [KOS03, Pas04] relied on non-black-box use of the adversary's algorithm [Bar01]. However, constant-round protocols making black-box use of the adversary were constructed by [PPV08, LP11a, Goy11], and making black-box use of one-way functions by Wee in $\omega(1)$ rounds [Wee10] and by Goyal in constant rounds [Goy11]. Furthermore, based on the non-malleable commitment scheme of [Goy11], the work of [GLOV12] constructs a constant-round multi-party coin-tossing protocol. Lin, Pass, and Venkitasubramanian[LPV09] presented a unified approach to construct UC-secure protocols from non-malleable commitments. However, as mentioned earlier, none of the aforementioned works focused on the *exact* round complexity of secure computation based on the round-complexity of non-malleable commitments.

Our recent work [GMPP16] examined the simultaneous-message model (in both two-party and multi-party settings), and proved a lower bound of four rounds in that model for general functionalities. More precisely, we show that in the simultaneous message exchange model, two-party coin-flipping (of $\omega(\log k)$ bits) requires at least four rounds (where both parties receive the output). On the positive side, starting from any k-round parallel and (3-robust) non-malleable commitment we show how to obtain (1) a $\max(4, t+1)$-round protocol for arbitrary functionalities in the two-party setting, and (2) a $\max(4, t+1)$-round protocol for the specific coin-tossing functionality in the multi-party setting.

3.1.2 Related Work in the CRS Model

Static Corruptions. In the CRS model, much effort has already been made in providing round efficient UC secure protocols. The works of Jarecki and Shmatikov [JS07] and Horvitz and Katz [HK07] present *two-party* protocols where the former is constant-round and the latter is two-round, which is the optimal. Asharov et al. [AJL$^+$12] first show a three-round *multi-party* computation protocol in the CRS model and a two-round multi-party computation protocol in the reusable public-key infrastructure setup model based on LWE. The authors in [AJL$^+$12] construct threshold FHE schemes based on the FHE schemes of [BV11a, BGV12]. The works of [BD10, MSS11] also present threshold FHE schemes but their protocols required more than two rounds of interaction. The work of Garg et al. [GGHR14] gives a two-round multi-party protocol under strong assumptions, namely, the existence of indistinguishability obfuscation for polynomial circuits and statistically-sound NIZKs.

More recently, the work of Mukherjee and Wichs [MW16], and its extensions [BP16, PS16], based on multi-key FHE [LTV12, CM15], shows how to obtain optimal 2-round constructions based on LWE and NIZKs in the CRS model. [3]

An alternative approach using randomized polynomials [AIK05] combined with [GMW87] yields a four-round multi-party computation protocol based on the assumption of semi-honest OT. Furthermore, we can generically compile any semi-honest protocol into one that is (UC)

[3]The protocol of [MW16] only assumes a common random string (as opposed to a common reference string which is sampled form a specific distribution.

secure against malicious adversaries using coin-tossing and (UC) NIZKs [DDO$^+$01], at the cost of adding two extra rounds for the coin-tossing.

Note that there is a large body of work that constructs secure computation protocols *in the preprocessing model* which require an initial offline computation phase independent of the inputs of the evaluated function. In particular, the works of [FH96, JJ00, CDN01, DN03, BDOZ11, DPSZ12, NNOB12, DZ13, DKL$^+$13] present multi-party computation protocols with round complexity proportional to the multiplicative depth of the evaluated circuit and the works of [LR14, LR15, NO16] present constant round protocols. Recent works tailored to the special case of two parties are presented in the works of [LR14, LR15, NO16] in the preprocessing model.

Adaptive Corruptions. Unconditionally secure protocols such as [BOGW88, CCD88] are typically adaptively secure. But these protocols are not constant round, and it is a major open problem if it is even possible to have unconditional security and constant number of rounds for secure computation of any function, see [DNPR16] for a detailed discussion and Section 2.2.

Recall that based on computational assumptions, we can achieve constant round protocols, with the first work of Yao's garbled circuits for two players, but on the other hand this does not give us adaptive security. Another class of protocols based on FHE also naturally leads to constant round protocols, where we can tolerate that a majority of players is corrupted. Here we also get low communication complexity, that depends only on the length of the inputs and outputs of the evaluated function. But again, these protocols achieve only static security as mentioned in the previous Section (see for instance [Gen09, AJL$^+$12, MW16]).

We can in fact get adaptive security in the computational setting, as shown in [CFGN96] by introducing the notion of Non-Commiting Encryption (NCE). Moreover, in [DN03], adaptive security was obtained as well, but much more efficiently using additively homomorphic encryption. However, neither [CFGN96] nor [DN03] run in a constant number of rounds.

If we assume honest majority, we can get both constant round and adaptive security but the communication complexity will be propositional to the size of the evaluated circuit. This was shown in several papers [DI05, DI06, DIK$^+$08, IPS08]. The idea here is to use an unconditionally secure protocol to compute, for instance, a Yao garbled circuit, that is then used to compute the desired function in a constant number of rounds. Since the computation leading to the Yao circuit is easy to parallelise, this can be constant round as well and we inherit adaptive security from the unconditionally secure preprocessing. In the dishonest majority setting, the work of [DPR16] shows an adaptively secure 3-round protocol based on LWE. The result of [KTZ13] suggests that the result of [DPR16] secure against $n-1$ corruptions out of the n parties is the best we can achieve based only on FHE.

Security against arbitrary corruptions. For the setting where all parties can be corrupted the round complexity of all known adaptively secure protocols secure against n corruptions grows (see, e.g. [CLOS02, GS12, DMRV13, Ven14]) linearly in the depth of the evaluated circuit. Recent independent works [GP15, CGP15, DKR15, CP16], have been shown that MPC protocols with security against n corruptions in a constant number of rounds can be achieved using indistinguishability obfuscation [GGH$^+$13b]. The work of [GP15] achieves two rounds of interactions which is the optimal. The work of Garg and Sahai [GS12] shows that a linear

number of rounds is required (in the multi-party setting) in the plain model with black-box simulation. However, this impossibility result does not apply here since the results mentioned above require a CRS.
Subsequent work. Relying on the technique of equivocating Yao's garbled circuits based on stateless tokens in the work of [HPV17] (see Section 2.1.3.3), a subsequent beautiful work by Canetti et al. [CPV16] shows how to equivocate garbled circuits removing the reliance on tokens in [HPV17] and thus construct the first two-round two-party protocol secure against any adaptive semi-honest corruption of both parties based on semi-honest adaptive OT in the plain model. [CPV16] also construct constant round multi-party computation protocols tolerating adaptive malicious corruption of all parties (and UC-secure protocols in the CRS model).

3.1.3 Related Work in the Tamper-Proof Hardware Model

Static Corruptions. The work of Goldreich and Ostrovsky [GO96] first considered the use of hardware tokens in the context of *software obfuscation* via Oblivious RAMs. A decade later, Katz in [Kat07] demonstrated the feasibility of achieving UC-secure protocols for arbitrary functionalities assuming tamper-proof tokens under static corruptions. In his formulation, the parties can create a token that computes arbitrary functionalities such that any adversary that is given access to the token can *only* observe the input/output behaviour of the token. In the UC framework, Katz described an ideal functionality $\mathcal{F}_{\text{WRAP}}$ that captures this model. Note that tokens can either be stateful or stateless, depending on whether the tokens are allowed to maintain some state between invocations (where stateless tokens are easier to implement). Following [Kat07], Goldwasser et al. [GKR08] investigated the use of *one-time programs*, that allow a semi-honest sender to create simple stateful tokens where a potentially malicious receiver executes them exactly once (or a bounded number of times). Their work considered concrete applications such as zero-knowledge proofs and focused on minimizing the number of required tokens. Our work [HPV16] shows that the standard (and most popular) formalization of the tamper proof hardware tokens (namely the $\mathcal{F}_{\text{WRAP}}$-functionality due to Katz [Kat07],) does not fully capture the power of the adversary in a concurrent setting. In particular, the formulation in [Kat07] does not capture a man-in-the-middle attack where an adversary can transfer the tokens received from one session to another. In [HPV16], a new formulation of tamper-proof hardware in the Global Universal Composable (GUC) framework was introduced that addressed these shortcomings.

The construction of [Kat07] relied on stateful tokens based on the DDH assumption, and was later improved by Lin et al. [LPV09] to rely on the minimal assumption of one-way functions. Goyal et al. [GIS+10] resolved the power of stateful tokens and showed how to obtain unconditionally secure protocols using stateful tokens. The work of Chandran, Goyal and Sahai [CGS08] was the first to achieve UC-security using only stateless tokens. Choi et al. [CKS+14] gave the first constant-round UC-secure protocols using stateless tokens assuming collision-resistant hash-functions. The works of [Nil15, MMN16] consider a GUC-like formulation of the tokens for the two-party setting where the parties have fixed roles. The focus in [Nil15, MMN16] was to obtain a formulation that accommodates reusability of a single token for several independent protocols in the UC-setting for the specific two-party case. In contrast to the work in [HPV16] , [Nil15, MMN16] does not explicitly model or discuss adversarial transferability of tokens. Finally, our work in [HPV16] resolved the question of identifying the minimal assumptions to construct UC-secure protocols under static corruptions with stateless tokens, namely, we show how to realize constant-round two-party and multi-party UC-secure protocols assuming only the existence of one-way functions. Besides these works, there have been several works in the tamper-proof token model [CGS08, MS08, LPV09, GIS$^+$10, DKM11, DMRV13, DMMN13, DKMN15b, CKS$^+$14]) addressing various efficiency parameters.

Adaptive Corruptions. In the adaptive setting, the works of [DMRV13, Ven14] and [GIS$^+$10] construct adaptive UC-secure protocols in the tamper-proof model using stateful tokens with round complexity proportional to the depth of the evaluated circuit. While the works of [DMRV13, Ven14] rely on simulatable public-key encryption schemes, Goyal et al. in [GIS$^+$10] provide unconditionally secure protocols (which in particular imply adaptive UC-security). The first work to address the feasibility of adaptive security using stateless tokens is our work [HPV17] which presents the first constant-round two-party protocol secure against malicious and adaptive adversaries who can adaptively corrupt all parties using stateless tokens. Our two-party adaptive protocol is based only on OWFs.

3.2 Information-Theoretic Setting

For all known constructions which are efficient in the circuit size of the evaluated function, it is the case that multiplication gates require communication to be processed (while addition/linear gates usually do not). That said, the number of rounds is at least the (multiplicative) depth of the circuit, and the communication complexity is proportional to the size of the evaluated circuit [GMW87, BOGW88, FY92, DN07, BTH08, IPS09, DIK10, BFO12, GIP$^+$14, GIP15] (see Table 3.1 for a detailed comparison on the communication complexity of IT MPC protocols). Whether this is inherent is an open problem. However, our work [DNPR16] shows that it is indeed inherent for protocols that follow the typical "gate-by-gate" design pattern, followed by the current protocols in the literature which are efficient in the evaluated function. In particular, [DNPR16] shows that any protocol that follows the typical gate-by-gate design pattern of secure computation must have communication complexity dependant on the circuit size and number of rounds dependant on the depth of the evaluated circuit (even in the dishonest majority setting with preprocessing). More specifically, the message complexity per multiplication gate must be equal to the privacy threshold. Furthermore, for arithmetic circuits over large fields one might wonder whether the communication must grow with the field size. Surprisingly, as shown in [DNPR16], each message can be independent of the field size of the inputs. The impossibility result of [DNPR16] implies that a fundamental new approach must be found in order to construct protocols, that are efficient in the circuit size of the evaluated function, with reduced communication complexity and constant round complexity that beat the complexities of [GMW87, BOGW88, CCD88, DPSZ12] etc.

We stress that there is a special case of IT protocols that enjoys constant round complexity. More specifically, the work of [GIKR02] constructs a two-round IT protocol which is efficient in the circuit size only if a *single* party holds the entire input.

Giving up on being efficient in the circuit size, there are IT secure and constant round protocols for any function [IK00], however they are extremely inefficient in general with respect to the computation. That said, they are efficient only for NC^1/ Logspace L computations. Without loss of generality, all these protocols fall into the IT Randomized Encoding (RE) model of computation. In particular, RE can be constructed for any language in L-$\mathsf{parity/poly}$, and also for some specific languages like quadratic residuosity (see [App14]). Note that there no known lower bounds on poly-size IT randomized encodings. For example, it is not known whether we can have poly-size IT randomized encoding for general poly-time functions (or efficient constant-round IT protocols). It is highly interesting to obtain lower bounds or construct poly-size IT randomized encodings.

Last but not least, there are two-round constructions in the computational setting (assuming only OWFs) that consider only a *single* corrupted party such as the work of [IKP10] for $n >= 5$ where n denotes the number of parties. The work of [IKKP15] complement the work of [IKP10] by presenting results for $n = 3$ and $n = 4$. However, lifting these protocols to the IT setting yield protocols that are only efficient in the branching program size of the evaluated function.

Comparison of [DNPR16] to related work. There is a lot of prior work on lower bounding communication in interactive protocols, see for instance [Kus92, FY92, CK93, FKN94, KM97, KR94, BSPV99, GR03] (see [DPP14] for an overview of these results). They typically provide lower bounds for very specific functions such as modular addition, and are not applicable to our situation. Probably the most relevant previous work is [DPP14]. Their model does not match ours, as they consider three parties where only two have input and only the third party gets output. Hence we cannot use their results directly, but it is instructive to consider their techniques as it shows why our problem is more tricky than it may seem at first. One important idea used in [DPP14] is to make a "cut", i.e., one considers a (small) subset $\mathcal{C}$ of the parties and then argue that either the communication between $\mathcal{C}$ and the rest of the world must be large enough to determine their inputs, since otherwise other players could not compute the output; or that $\mathcal{C}$ must receive information of sufficient size to be able to compute its own outputs.

It turns out that these ideas are not sufficient for us: recall that we start from a situation where players already have shares of the input values a, b. Now, if $\mathcal{C}$ is large enough to be qualified in the input secret sharing scheme, then $\mathcal{C}$ already has information enough to determine a, b (and for some secret sharing schemes even the shares of all players). So $\mathcal{C}$ can in principle compute correct shares of $c = ab$ by itself without communicating with anyone. On the other hand, if $\mathcal{C}$ is unqualified, then the complement of $\mathcal{C}$ is typically qualified, and therefore does not need information from $\mathcal{C}$ to compute output. But one might think that $\mathcal{C}$ needs to *receive* information to determine its output, in particular, the output shares must be properly coordinated to form a consistent sharing of c. Remember, however, that players already have properly coordinated shares of *the inputs*, and they might be able to use those to form a correct output sharing while communicating less. Indeed, this is what happens for addition gates, where there is no communication, players just add their shares locally.

It follows that the idea of a cut is not enough, one must exploit in some non-trivial way that we are handling a *multiplication* gate, which is exactly what we do. It is possible that one could use the fact that we do multiplication together with the concept of residual information which was also used in [DPP14], to get better bounds than we achieve here, but this remains a speculation.

Note that our model does not count communication needed to construct the shares that are input, nor does it count any communication needed to reconstruct results from the output shares. This does count in the standard model and makes lower bounds easier to prove. For instance, in [DNOR16] lower bounds were recently proved on the message complexity of computing a large class of functions securely, primarily by showing that a significant number of messages must be sent before the inputs are uniquely determined. In fact, if we included a secret sharing phase before the multiplication protocol and a reconstruction phase after it, these would entail so much communication that the bounds obtained from existing results would leave nothing to explain why the *privacy preserving* multiplication step is communication intensive.

It is also easy to see that one cannot get bounds in our model based only on correctness, for instance by methods from communication complexity. If parties have shares in a and b, no communication is needed to produce some set of *correct* shares in ab: one can simply consider the shares in a and b together as a (redundant) sharing of ab. Indeed this satisfies all our demands to a multiplication gate protocol except privacy: the output threshold is the same and we can correctly reconstruct ab, but privacy is of course violated because reconstruction would tell us more than ab. So, our bounds arguably require privacy.

Last but not least, the work of [FDB93] provides impossibility results for homomorphic monotone sharing schemes. Their result restricts the output sharing scheme to be of the same structure as the input sharing scheme. Moreover, the work of [FDB93] does not yield implications to secure computation in the dishonest majority setting (with preprocessing). The result of [DNPR16] does not pose the restriction on the sharing schemes and provides impossibility results in the dishonest majority setting.

[4]In the client-server model, where m is the number of clients and n the number of servers.

Ref.	Adv.	Copies	Resilience	Communication complexity
[GMW87]	passive	1	$\lvert\mathcal{T}\rvert < n$	$O(n^2\lvert\mathsf{C}\rvert)$ for boolean circuits
[IPS09]	passive	1	$\lvert\mathcal{T}\rvert < n$	$O(n^2\lvert\mathsf{C}\rvert)$
[BOGW88]	passive	1	$\lvert\mathcal{T}\rvert < n/2$	$O(n^2\lvert\mathsf{C}\rvert)$
[DN07]	passive	1	$\lvert\mathcal{T}\rvert < n/2$	$O(n\lvert\mathsf{C}\rvert + n^2)$
[FY92]	passive	$\Theta(n)$	$\lvert\mathcal{T}\rvert < (1/2-\epsilon)n$	$O(n\lvert\mathsf{C}\rvert)$
[DIK10]a	passive	$\Theta(n)$	$\lvert\mathcal{T}\rvert < (1/2-\epsilon)n$	$O(\lvert\mathsf{C}\rvert + n)$
[DIK10]b	passive	1	$\lvert\mathcal{T}\rvert < (1/2-\epsilon)n$	$\tilde{O}(\lvert\mathsf{C}\rvert + n \cdot d_{\mathsf{C}} + n^2)$
[IPS09]a	active	1	$\lvert\mathcal{T}\rvert < n$	$O(n^2\lvert\mathsf{C}\rvert + \log\lvert\mathbb{F}\rvert \cdot d_{\mathsf{C}})$
[GIP+14]	active	1	$\lvert\mathcal{T}\rvert < n$	$O(n^2\lvert\mathsf{C}\rvert)$
[GIP15]	active	1	$\lvert\mathcal{T}\rvert < n$	$O(n^2\lvert\mathsf{C}\rvert)$
[BFO12]	active	1	$\lvert\mathcal{T}\rvert < n/2$	$O(n\lvert\mathsf{C}\rvert + n^2 \log n \cdot d_{\mathsf{C}}) +$ poly(n)
[GIP+14]	active	1	$\lvert\mathcal{T}\rvert < n/2$	$O(n\lvert\mathsf{C}\rvert + n^2)$
[GIP15]	active	1	$\lvert\mathcal{T}\rvert < n/2$	$O(n\lvert\mathsf{C}\rvert + n^2)$
[IPS09]b	active	$\Theta(n)$	$\lvert\mathcal{T}\rvert < (1/2-\epsilon)n$	$O(\lvert\mathsf{C}\rvert + n \cdot d_{\mathsf{C}})$
[IPS09]c	active	$\Theta(n)$	$\lvert\mathcal{T}\rvert < (1/2-\epsilon)n$	$O(\lvert\mathsf{C}\rvert + m \cdot d_{\mathsf{C}})$[4]
[GIP15]	active	$\Theta(n)$	$\lvert\mathcal{T}\rvert < (1/2-\epsilon)n$	$O(\lvert\mathsf{C}\rvert + n)$
[DIK10]c	active	1	$\lvert\mathcal{T}\rvert < (1/2-\epsilon)n$	$\tilde{O}(\lvert\mathsf{C}\rvert + n^2 \cdot d_{\mathsf{C}})$
[GIP15]	active	1	$\lvert\mathcal{T}\rvert < (1/2-\epsilon)n$	$\tilde{O}(\lvert\mathsf{C}\rvert + n \cdot d_{\mathsf{C}} + n^2)$

Table 3.1: [GIP15] Comparison of information-theoretic MPC protocols for arithmetic circuits. In the above, n is the number of parties, ϵ is an arbitrary small positive constant, C is an arithmetic circuit or an SIMD circuit over a finite field $\mathbb{F}$, d_{C} is the multiplicative depth of C, and $\mathcal{T}$ is the set of corrupted parties such that $\lvert\mathcal{T}\rvert \leq t$ for privacy threshold t. We consider two regimes for such protocols: the single input, single circuit regime and the Franklin and Yung (FY) [FY92] regime for simultaneously evaluating ℓ copies of the circuit on different inputs. Notice that the latter is a special case of the former that allows for simpler and more efficient solutions. The copies column indicates the number of simultaneously evaluated circuit copies. Passively secure protocols achieve perfect security while actively secure protocols realize C (with abort) with at most $O(1/\lvert\mathbb{F}\rvert)$ simulation error. The communication complexity column counts the total number of field elements exchanged between the parties. For the case of simultaneous evaluation of multiple copies, we count the amortized cost for evaluating a single copy of C. The protocols having resilience $\lvert\mathcal{T}\rvert < n$ are constructed on the OT or OLE hybrid model. An OLE oracle (an arithmetic generalization of OT) is a generalization of an OT oracle, it receives $a, b \in \mathbb{F}$ from one party and x from another, and returns $ax + b$ to the latter. Note that the $\tilde{O}$ notation suppresses logarithmic factors.

Part II

Results in the Computational Setting

Chapter 4

Secure Computation in the Plain Model

In this chapter we tackle the *exact* round complexity of secure computation in the multi-party and two-party settings. In particular, we present our results in [GMPP16]. See Section 2.1.1 for a detailed overview of our contributions.

Overview. For the special case of two-parties *without* a simultaneous message exchange channel, this question has been extensively studied and resolved. In particular, as mentioned in Section 2.1.1, [KO04] proved that five rounds are necessary and sufficient for securely realizing every two-party functionality where both parties receive the output. However, the exact round complexity of general multi-party computation, as well as two-party computation *with* a simultaneous message exchange channel, is not very well understood.

These questions are intimately connected to the round complexity of non-malleable commitments. Indeed, the *exact* relationship between the round complexities of non-malleable commitments and secure multi-party computation has also not been explored.

In this work, we revisit these questions and obtain several new results. First, we establish the following main results. Suppose that there exists a k-round non-malleable commitment scheme, and let $k' = \max(4, k+1)$; then,

- **(Two-party setting with simultaneous message transmission):** there exists a k'-round protocol for securely realizing *every* two-party functionality;
- **(Multi-party setting):** there exists a k'-round protocol for securely realizing the *multi-party coin-flipping* functionality.

As a corollary of the above results, by instantiating them with existing non-malleable commitment protocols (from the literature), we establish that **four** rounds are both necessary and sufficient for both the results above. Furthermore, we establish that, for *every multi-party functionality five* rounds are sufficient. We actually obtain a variety of results offering trade-offs between rounds and the cryptographic assumptions used, depending upon the particular instantiations of underlying protocols.

Due to space constraints we refer to the full version for our multi-party coin-flipping protocol [GMPP16]. In this thesis, we present the lower bound and the two-party protocol.

4.1 Techniques

We now provide an overview of our approach. As discussed earlier, we first focus on the two-party setting with a simultaneous message exchange channel.

The starting point of our construction is the Katz-Ostrovsky (KO) protocol [KO04] which is a *four round* protocol for *one-sided* functionalities, i.e., in that only one party gets the output.

Recall that, this protocol does not assume the presence of a simultaneous message exchange channel. At the cost of an extra round, the KO two-party protocol can be converted to a *complete* (i.e. both-sided) protocol where both parties get their corresponding outputs via a standard trick [Gol03] as follows: parties compute a modified functionality in which the first party P_1 learns its output as well as the output of the second party P_2 in an "encrypted and authenticated"[1] form. It then sends the encrypted value to P_2 who can decrypt and verify its output.

A natural first attempt is to adapt this simple and elegant approach to the setting of simultaneous message exchange channel, so that the "encrypted/authenticated output" can somehow be communicated to P_2 simultaneously at the same time when P_2 sends its last message, thereby removing the additional round.

It is not hard to see that any such approach would not work. Indeed, in the presence of malicious adversaries while dealing with a simultaneous message exchange channel, the protocol must be proven secure against "rushing adversaries" who can send their messages after looking at the messages sent by the other party. This implies that, if P_1 could indeed send the "encrypted/authenticated output" message simultaneously with last message from P_2, it could have sent it earlier as well. Now, applying this argument repeatedly, one can conclude that any protocol which does not use the simultaneous message exchange channel necessarily in all of the four rounds, is bound to fail (see Section 4.3). In particular, any such protocol can be transformed, by simple rescheduling, into a 3-round protocol contradicting our lower bound[2].

This means that we must think of an approach which must use the simultaneous message exchange channel in each round. In light of this, a natural second attempt is to run two executions of a 4-round protocol (in which only one party learns the output) in "opposite" directions. This would allow both parties to learn the output. Unfortunately, such approaches do not work in general since there is no guarantee that an adversarial party would use the same input in both protocol executions. Furthermore, another problem with this approach is that of "non-malleability" where a cheating party can make its input dependent on the honest party's input: for example, it can simply "replay" back the messages it receives. A natural approach to prevent such attacks is to deploy non-malleable commitments, as we discuss below.

Simultaneous executions + non-malleable commitments. Following the approach discussed above we observe that:

1. A natural direction is to use two simultaneous executions of the KO protocol (or any other similar 4-round protocol) over the simultaneous message exchange channel in opposite directions. Since we have only 4 rounds, a different protocol (such as some form of 2-round semi-honest protocol based on Yao) is not a choice.

2. We must use non-malleable commitments to prevent replay/mauling attacks.

[1] In particular, the encryption prevents P_1 to know P_2's output ensuring output privacy whereas the authentication does not allow P_1 to send P_2 a wrong output.

[2] Recall that we show that (see Thoerem 4.2 for a formal statement) 4 rounds are necessary even with simultaneous message exchange channels.

We remark that, the fact that non-malleable commitments come up as a natural tool is not a coincidence. As noted earlier, the multi-party case is well known to be *inherently connected* to non-malleable commitments. Even though our current focus is solely on the two-party case, this setting is essentially (a special case of) the multi-party setting due to the use of the simultaneous message exchange channel. Prior to our work, non-malleable commitments have been used extensively to design multi-party protocols [Goy11, LP11b, LPTV10, GLOV12]. However, all of these works result in rather poor round complexity because of their focus on asymptotic, as opposed to exact, number of rounds.

To obtain our protocol, we put the above two ideas together, modifying several components of KO[3] to use non-malleable commitments. These components are then put together in a way such that, even though there are essentially two simultaneous executions of the protocol in opposite directions, messages of one protocol cannot be maliciously used to affect the other messages. In the following, we highlight the main ideas of our construction:

1. The first change we make is to the proof systems used by KO. Recall that KO uses the Fiege-Shamir (FS) protocol as a mechanism to "force the output" in the simulation. Our first crucial modification is to consider a variant of the FS protocol in which the verifier gives two non-malleable commitments (nmcom) to two strings σ_1, σ_2 and gives a *witness indistinguishable proof-of-knowledge* (WIPOK) that it knows one of them. These are essentially the simulation trapdoors, but implemented through nmcom instead of a one-way function. This change is actually crucial, and as such, brings in an effect similar to "simulation sound" zero-knowledge.

2. The oblivious transfer protocol based on trapdoor permutations and coin-tossing now performs coin-tossing with the help of nmcom instead of simple commitments. This is a crucial change since this allows us to slowly get rid of the honest party's input in the simulation and still argue that the distribution of the adversary's input does not change as a result of this.

   We note that there are many parallel executions on nmcom that take place at this stage, and therefore, we require that nmcom should be non-malleable under many parallel executions. This is indeed true for most nmcom.

3. Finally, we introduce a mechanism to ensure that the two parties use the exact same input in both executions. Roughly speaking, this is done by requiring the parties to prove consistency of messages "across" protocols.

4. To keep the number of rounds to $k + 1$ (or 4 if $k < 3$), many of the messages discussed above are "absorbed" with other rounds by running in parallel.

[3]The KO protocol uses a clever combination of garble circuits, semi-honest oblivious transfer, coin-tossing, and WIPOK to ensure that the protocol is executed with a fixed input (allowing at the same time simulation extractability of the input), and relies on the zero-knowledge property of a modified Fiege-Shamir proof to achieve output simulation.

Multi-party setting. The above protocol does not directly extend to the multi-party settings. Nevertheless, for the special case of *coin flipping*, we show that a (simplified) version of the above protocol works for the multi-party case. This is because the coin-tossing functionality does not really require any computation, and therefore, we can get rid of components such as oblivious transfer. In fact, this can be extended "slightly more" to also realize the "coin-flipping with committed inputs" since committing the input does not depend on inputs of other parties.

Next, to obtain our result for general functionalities, we simply invoke known results: using [MW16] with coin-flipping gives us a six round protocol, and using [GGHR14] gives a five round result.

4.2 Preliminaries

Notation. We denote the security parameter by κ. We say that a function $\mu : \mathbb{N} \to \mathbb{N}$ is *negligible* if for every positive polynomial $p(\cdot)$ and all sufficiently large κ's it holds that $\mu(\kappa) < \frac{1}{p(\kappa)}$. We use the abbreviation PPT to denote probabilistic polynomial-time. We often use $[n]$ to denote the set $\{1, ..., n\}$. Moreover, we use $d \leftarrow \mathcal{D}$ to denote the process of sampling d from the distribution $\mathcal{D}$ or, if $\mathcal{D}$ is a set, a uniform choice from it. If $\mathcal{D}_1$ and $\mathcal{D}_2$ are two distributions, then we denote that they are statistically close by $\mathcal{D}_1 \approx_{\mathrm{s}} \mathcal{D}_2$; we denote that they are computationally indistinguishable by $\mathcal{D}_1 \approx_{\mathrm{c}} \mathcal{D}_2$; and we denote that they are identical by $\mathcal{D}_1 \equiv \mathcal{D}_2$. Let V be a random variable corresponding to the distribution $\mathcal{D}$. Sometimes we abuse notation by using V to denote the corresponding distribution $\mathcal{D}$.

We assume familiarity with several standard cryptographic primitives. For notational purposes, we recall here the basic working definitions for some of them. We skip the well-known formal definitions for secure two-party and multi-party computations (see Appendix for a formal description). It will be sufficient to have notation for the two-party setting. We denote a two party functionality by $F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^* \times \{0,1\}^*$ where $F = (F_1, F_2)$. For every pair of inputs (x, y), the output-pair is a random variable $(F_1(x, y), F_2(x, y))$ ranging over pairs of strings. The first party (with input x) should obtain $F_1(x, y)$ and the second party (with input y) should obtain $F_2(x, y)$. Without loss of generality, we assume that F is deterministic. The security is defined through the ideal/real world paradigm where for adversary $\mathcal{A}$ participating in the real world protocol, there exists an ideal world simulator $\mathcal{S}$ such that for every (x, y), the output of $\mathcal{S}$ is indistinguishable from that of $\mathcal{A}$. See Appendix for an extended discussion.

We now recall the definitions for non-malleable commitments as well as some components from the work of Katz-Ostrovsky [KO04].

4.2.1 Tag Based Mon-Malleable Commitments

Let $\mathsf{nmcom} = \langle C, R \rangle$ be a k-round commitment protocol where C and R represent (randomized) committer and receiver algorithms, respectively. Denote the messages exchanged by $(\mathsf{nm}_1, \ldots, \mathsf{nm}_k)$ where nm_i denotes the message in the i-th round.

For some string $u \in \{0,1\}^\kappa$, tag $\mathsf{id} \in \{0,1\}^t$, non-uniform PPT algorithm M with "advice" string $z \in \{0,1\}^*$, and security parameter κ, consider the following experiment: M on input $(1^\kappa, z)$, interacts with C who commits to u with tag id; simultaneously, M interacts with $R(1^\kappa, \widetilde{\mathsf{id}})$

attempting to commit to a related value $\widetilde{u}$, again using identity $\widetilde{\mathsf{id}}$ of its choice (M's interaction with C is called the left interaction, and its interaction with R is called the right interaction); M controls the scheduling of messages; the output of the experiment is denoted by a random variable $\mathsf{nmc}^M_{\langle C,R\rangle}(u,z)$ that describes the view of M in both interactions and the value $\widetilde{u}$ which M commits to R in the right execution unless $\widetilde{\mathsf{id}} = \mathsf{id}$ in which case $\widetilde{u} = \perp$, i.e., a commitment where the adversary copies the identity of the left interaction is considered invalid.

Definition 4.1 (Tag based non-malleable commitments)**.** A commitment scheme $\mathsf{nmcom} = \langle C, R\rangle$ is said to be non-malleable *with respect to commitments* if for every non-uniform PPT algorithm M (man-in-the-middle), for every pair of strings $(u^0, u^1) \in \{0,1\}^\kappa \times \{0,1\}^\kappa$, every tag-string $\mathsf{id} \in \{0,1\}^t$, every $\kappa \in \mathbb{N}$, every (advice) string $z \in \{0,1\}^*$, the following two distributions are computationally indistinguishable:

$$\mathsf{nmc}^M_{\langle C,R\rangle}(u^0,z) \overset{c}{\approx} \mathsf{nmc}^M_{\langle C,R\rangle}(u^1,z)$$

Parallel Non-Malleable Commitments. We consider a strengthening of nmcom in which M can receive commitments to m strings on the "left", say $(u_1, \ldots, u_m)$, with tags $(\mathsf{id}_1, \ldots, \mathsf{id}_m)$ and makes m commitments on the "right" with tags $(\widetilde{\mathsf{id}}_1, \ldots, \widetilde{\mathsf{id}}_m)$. We assume that m is a fixed, possibly a-priori bounded, polynomial in the security parameter κ. In the following let $i \in [m], b \in \{0,1\}$: We say that a nmcom is an m-bounded parallel non-malleable commitment if for every pair of sequences $\{u_i^b\}$ the random variables $\mathsf{nmc}^M_{\langle C,R\rangle}(\{u_i^0\},z)$ and $\mathsf{nmc}^M_{\langle C,R\rangle}(\{u_i^1\},z)$ are computationally indistinguishable where $\mathsf{nmc}^M_{\langle C,R\rangle}(\{u_i^b\},z)$ describes the view of M and the values $\{\widetilde{u}_i^b\}$ committed by M in the m sessions on the right with tags $\{\widetilde{\mathsf{id}}_i\}$ while receiving parallel commitments to $\{u_i^b\}$ on left with tags $\{\mathsf{id}_i\}$.

First message binding property. It will be convenient in the notation to assume that the first message nm_1 of the non-malleable commitment scheme nmcom statistically determines the message being committed. This can be relaxed to only require that the message is fixed before the last round if $k \geq 3$.

Non-Malleable Commitment Robust w.r.t. k-round Protocols. Lin and Pass [LP09a] introduced the notion of non-malleability w.r.t. arbitrary k-round protocols. Traditional definitions of non-malleability consider a setting where a man-in-the middle adversary is participating in two (or more) executions of the *same* protocol. However, non-malleability w.r.t. arbitrary protocols considers a class of adversaries that can participate in a left interaction of any arbitrary protocol.

Consider a one-many man-in-the-middle adversary M that participates in one left interaction – communicating with a machine B –and one right interaction– acting as a committer using the commitment scheme $\langle C, R\rangle$. As in the standard definition of non-malleability, M can adaptively choose the identity in the right interaction. We denote by $\mathsf{nmc}^{B,M}_{\langle C,R\rangle}(y,z)$ the random variable consisting of the view of $M(z)$ in a man-in-the-middle execution when communicating with $B(y)$ on the left and an honest receiver on the right, combined with the value M commits to on the right. Intuitively, we say that $\langle C, R\rangle$ is non-malleable w.r.t. B if $\mathsf{nmc}^{B,M}_{\langle C,R\rangle}(y^1,z)$ and

$\mathsf{nmc}^{B,M}_{\langle C,R\rangle}(y^2, z)$ are indistinguishable, whenever interactions with $B(y^1)$ and $B(y^2)$ cannot be distinguished.

Definition 4.2 (Robust non-malleable commitments)**.** Let B a probabilistic polynomial-time machine. A commitment scheme $\mathsf{nmcom} = \langle C, R\rangle$ is said to be *non-malleable w.r.t.* B, if for every two sequences $\{y^1_\kappa\}_{\kappa\in\mathbb{N}}$ and $\{y^2_\kappa\}_{\kappa\in\mathbb{N}}$ where $y^1_\kappa, y^2_\kappa \in \{0,1\}^\kappa$, every $\kappa \in \mathbb{N}$, every (advice) string $z \in \{0,1\}^*$, the following two distributions are computationally indistinguishable for all non-uniform PPT algorithms $\tilde{M}$:

$$\mathsf{view}_{\tilde{M}}[\langle B(y^1_\kappa), \tilde{M}(z)\rangle(1^\kappa)] \overset{c}{\approx} \mathsf{view}_{\tilde{M}}[\langle B(y^2_\kappa), \tilde{M}(z)\rangle(1^\kappa)].$$

where $\mathsf{view}_{\tilde{M}}[\langle B(y), M(z)\rangle(1^\kappa)]$ denotes the view of $\tilde{M}$ in interaction with B on common input 1^κ, and private inputs z and y respectively, then it also holds that for every non-uniform PPT man-in-the-middle adversary M the following two distributions are computationally indistinguishable:

$$\mathsf{nmc}^{B,M}_{\langle C,R\rangle}(y^1_\kappa, z) \overset{c}{\approx} \mathsf{nmc}^{B,M}_{\langle C,R\rangle}(y^2_\kappa, z)$$

We say that nmcom is non-malleable w.r.t. k-round protocols if nmcom is non-malleable w.r.t. any machine B that interacts with the man-in-the-middle adversary in k rounds. Any commitment scheme that is "extractable" and has more than k "rewinding slots" is directly one-many non-malleable w.r.t. k-round protocols [LP09a]. In this work, we focus on non-malleability w.r.t 3-round protocols and in particular 3-round WI proof of knowledge protocols; we call such non malleable commitments 3-*robust* (or simply *robust*).

4.2.2 Components of our Protocol

In this section, we recall some components from the KO protocol [KO04]. These are mostly standard and recalled here for a better exposition. The only (minor but crucial) change needed in our protocol is to the FLS proof system [FLS99, FS90a, Fei90] where a *non-malleable* commitment protocol is used by the verifier. For concreteness, let us discuss how to fix these proof systems first.

Modified Feige-Shamir proof systems. We use two proof systems: Π_{WIPOK} and Π_{FS}. Protocol Π_{WIPOK} is the 3-round, public-coin, witness-indistinguishable proof-of-knowledge based on the work of Feige, Lapidot, Shamir [FLS99] for proving graph Hamiltonicity. This proof system proves statements of the form $\mathsf{st}_1 \wedge \mathsf{st}_2$ where st_1 is fixed at the first round of the protocol, but st_2 is determined only in the last round of the protocol.[4] For concreteness, this proof system is given in the Appendix.

Protocol Π_{FS} is the 4-round zero-knowledge argument-of-knowledge protocol of Feige and Shamir [FS90a], which allows the prover to prove statement thm, with the modification that the protocol from verifier's side is implemented using nmcom. More specifically,

[4]Typically, st_1 is a empty statement and not usually mentioned; but KO [KO04] uses a specific, non-empty, statement and so does this work.

- Recall that the Feige-Shamir protocol consists of two executions of Π_{WIPOK} in reverse directions. In the first execution, the verifier selects a one-way function f and sets $x_1 = f(w_1)$, $x_2 = f(w_2)$ and proves the knowledge of a witness for $x_1 \vee x_2$. In the second execution, prover proves the knowledge of a witness to the statement $\mathsf{thm} \vee (x_1 \vee x_2)$ where thm is the statement to be proven. The rounds of these systems can be somewhat parallelized to obtain a 4-round protocol.
- Our modified system, simply replaces the function f and x_1, x_2 with two executions of nmcom. For convenience, suppose that nmcom has only 3 rounds. Then, our protocol creates the first message of two independent executions of nmcom to strings σ_1, σ_2, denoted by $\mathsf{nm}_1^{\sigma_1}, \mathsf{nm}_1^{\sigma_2}$ respectively, and sets $x_1 = \mathsf{nm}_1^{\sigma_1}, x_2 = \mathsf{nm}_1^{\sigma_2}$. The second and third messages of nmcom are sent with the second and third messages of the original FS protocol.

  If nmcom has more than 3 rounds, simply complete the first $k - 3$ rounds of the two executions before the 4 messages of the proof system above are exchanged.
- As before, although Π_{FS} proves statement thm, as noted in [KO04], it actually proves statements of the form $\mathsf{thm} \wedge \mathsf{thm}'$ where thm can be fixed in the second round, and thm' in the fourth round. Usually thm is empty and not mentioned. Indeed, this is compatible with the second Π_{WIPOK} which proves statement of the form $\mathsf{st}_1 \wedge \mathsf{st}_2$, just set $\mathsf{st}_1 = \mathsf{thm}, \mathsf{st}_2 = \mathsf{thm}'$.

For completeness, we describe the full Π_{FS} protocol in the Appendix.

4.2.2.1 Components of Katz-Ostrovsky Protocol

The remainder of this section is largely taken from [KO04] where we provide basic notations and ideas for semi-honest secure two-party computation based on Yao's garbled circuits and semi-honest oblivious transfer (based on trapdoor one-way permutations). Readers familiar with [KO04] can skip this part without loss in readability.

Semi-Honest Secure Two-party Computation. We view Yao's garbled circuit scheme [Yao82, LP09b] as a tuple of PPT algorithms ($\mathsf{GenGC}, \mathsf{EvalGC}$), where GenGC is the "generation procedure" which generates a garbled circuit for a circuit C along with "labels," and EvalGC is the "evaluation procedure" which evaluates the circuit on the "correct" labels. Each individual wire i of the circuit is assigned two labels, namely $Z_{i,0}, Z_{i,1}$. More specifically, the two algorithms have the following format (here $i \in [\kappa], b \in \{0,1\}$):

- $(\{Z_{i,b}\}, \mathsf{GC}_y) \leftarrow \mathsf{GenGC}(1^\kappa, F, y)$: GenGC takes as input a security parameter κ, a circuit F and a string $y \in \{0,1\}^\kappa$. It outputs a *garbled circuit* GC_y along with the set of all *input-wire labels* $\{Z_{i,b}\}$. *The* garbled circuit *may be viewed as representing the function* $F(\cdot, y)$.
- $v = \mathsf{EvalGC}(\mathsf{GC}_y, \{Z_{i,x_i}\})$: *Given a garbled circuit* GC_y *and a set of input-wire labels* $\{Z_{i,x_i}\}$ *where* $x \in \{0,1\}^\kappa$, EvalGC *outputs either an invalid symbol* $\perp$, *or a value* $v = F(x, y)$.

The following properties are required:

Correctness. $\Pr[F(x,y) = \mathsf{EvalGC}(\mathsf{GC}_y, \{Z_{i,x_i}\})] = 1$ for all F, x, y, taken over the correct generation of $\mathsf{GC}_y, \{Z_{i,b}\}$ by GenGC.

Security. There exists a PPT simulator SimGC such that for any (F, x) and uniformly random labels $\{Z_{i,b}\}$, we have that:

$$(\mathsf{GC}_y, \{Z_{i,x_i}\}) \overset{c}{\approx} \mathsf{SimGC}\left(1^\kappa, F, v\right)$$

where $(\{Z_{i,b}\}, \mathsf{GC}_y) \leftarrow \mathsf{GenGC}\left(1^\kappa, F, y\right)$ and $v = F(x, y)$.

In the semi-honest setting, two parties can compute a function F of their inputs, in which only *one* party, say P_1, learns the output, as follows. Let x, y be the inputs of P_1, P_2, respectively. First, P_2 computes $(\{Z_{i,b}\}, \mathsf{GC}_y) \leftarrow \mathsf{GenGC}(1^\kappa, F, y)$ and sends GC_y to P_1. Then, the two parties engage in κ parallel instances of OT. In particular, in the i-th instance, P_1 inputs x_i, P_2 inputs $(Z_{i,0}, Z_{i,1})$ to the OT protocol, and P_1 learns the "output" Z_{i,x_i}. Then, P_1 computes $v = \mathsf{EvalGC}(\mathsf{GC}_y, \{Z_{i,x_i}\})$ and outputs $v = F(x, y)$.

A 3-round, semi-honest, OT protocol can be constructed from enhanced trapdoor permutations (TDP). For notational purposes, define TDP as follows:

Definition 4.3 (Trapdoor permutations)**.** Let $\mathcal{F}$ be a triple of PPT algorithms $(\mathsf{Gen}, \mathrm{Eval}, \mathrm{Invert})$ such that if $\mathsf{Gen}(1^\kappa)$ outputs a pair $(f, \mathsf{t}d)$, then $\mathrm{Eval}(f, \cdot)$ is a permutation over $\{0,1\}^\kappa$ and $\mathrm{Invert}(f, \mathsf{t}d, \cdot)$ is its inverse. $\mathcal{F}$ is a trapdoor permutation such that for all PPT adversaries A:

$$\Pr[(f, \mathsf{t}d) \leftarrow \mathsf{Gen}(1^\kappa); y \leftarrow \{0,1\}^\kappa; x \leftarrow A(f, y) : \mathrm{Eval}(f, x) = y] \leq \mu(\kappa).$$

For convenience, we drop $(f, \mathsf{t}d)$ from the notation, and write $f(\cdot), f^{-1}(\cdot)$ to denote algorithms $\mathrm{Eval}(f, \cdot), \mathrm{Invert}(f, \mathsf{t}d, \cdot)$ respectively, when $f, \mathsf{t}d$ are clear from the context. We assume that $\mathcal{F}$ satisfies (a weak variant of) "certifiability": namely, given some f it is possible to decide in polynomial time whether $\mathrm{Eval}(f, \cdot)$ is a permutation over $\{0,1\}^\kappa$.

Let H be the *hardcore bit function for κ bits* for the family $\mathcal{F}$; κ hardcore bits are obtained from a single-bit hardcore function h and $f \in \mathcal{F}$ as follows: $\mathsf{H}(z) = h(z) \| h(f(z)) \| \ldots \| h(f^{\kappa-1}(z))$. Informally, $\mathsf{H}(z)$ looks pseudorandom given $f^\kappa(z)$.

The semi-honest OT protocol based on TDP is constructed as follows. Let P_2 hold two strings $Z_0, Z_1 \in \{0,1\}^\kappa$ and P_1 hold a bit b. In the first round, P_2 chooses trapdoor permutation $(f, f^{-1}) \leftarrow \mathsf{Gen}(1^\kappa)$ and sends f to P_1. Then P_1 chooses two random string $z_0', z_1' \leftarrow \{0,1\}^\kappa$, computes $z_b = f^\kappa(z_b')$ and $z_{1-b} = z_{1-b}'$ and sends (z_0, z_1) to P_2. In the last round P_2 computes $W_a = Z_a + \mathsf{H}(f^{-\kappa}(z_a))$ where $a \in \{0,1\}$, H is the hardcore bit function and sends (W_0, W_1) to P_1. Finally, P_2 can recover Z_b by computing $Z_b = W_b + \mathsf{H}(z_b)$.

Putting it altogether, we obtain the following 3-round, semi-honest secure two-party protocol for the single-output functionality F (here only P_1 receives the output):

Protocol $\Pi_{\mathsf{S}H}$. P_1 holds input $x \in \{0,1\}^\kappa$ and P_2 holds inputs $y \in \{0,1\}^\kappa$. Let $\mathcal{F}$ be a family of trapdoor permutations and let H be a hardcore bit function. For all $i \in [\kappa]$ and $b \in \{0,1\}$ the following steps are executed:

Round-1 : P_2 computes $(\{Z_{i,b}\}, \mathsf{GC}_y) \leftarrow \mathsf{GenGC}(1^\kappa, F, y)$ and chooses trapdoor permutation $(f_{i,b}, f_{i,b}^{-1}) \leftarrow \mathsf{Gen}(1^\kappa)$ and sends $(\mathsf{GC}_y, \{f_{i,b}\})$ to P_2.

Round-2 : P_1 chooses random strings $\{z'_{i,b}\}$, computes $z_{i,b} = f^\kappa(z'_{i,b})$ and $z_{i,1-b} = z'_{i,1-b}$ and sends $\{z_{i,b}\}$ to P_2.

Round-3 : P_2 computes $W_{i,b} = Z_{i,b} + \mathsf{H}(f_{i,b}^{-\kappa}(z_{i,b}))$ and sends $\{W_{i,b}\}$ to P_2.

Output : P_1 recovers the labels $Z_{i,x_i} = W_{i,x_i} + \mathsf{H}(z_{i,x_i})$ and computes $v = \mathsf{EvalGC}(\mathsf{GC}_y, \{Z_{i,x_i}\})$ where $v = F(x,y)$

Equivocal Commitment scheme Eqcom. We assume familiarity with equivocal commitments, and use the following equivocal commitment scheme Eqcom based on any (standard) non-interactive, perfectly binding, commitment scheme Com: to commit to a bit x, the sender chooses coins ζ_1, ζ_2 and computes $\mathsf{Eqcom}(x; \zeta_1, \zeta_2) \overset{\text{def}}{=} \mathsf{Com}(x; \zeta_1) || \mathsf{Com}(x; \zeta_2)$. It sends $\mathsf{C}_x = \mathsf{Eqcom}(x; \zeta_1, \zeta_2)$ to the receiver along with a zero-knowledge proof that C_x was constructed correctly (i.e., that there exist x, ζ_1, ζ_2 such that $\mathsf{C}_x = \mathsf{Eqcom}(x; \zeta_1, \zeta_2)$.

To decommit, the sender chooses a bit b at random and reveals x, ζ_b, denoted by $\mathsf{open}_{\mathsf{C}_x}$. Note that a simulator can "equivocate" the commitment by setting $C = \mathsf{Com}(x; \zeta_1) || \mathsf{Com}(\overline{x}; \zeta_2)$ for a random bit x, simulating the zero-knowledge proof and then revealing ζ_1 or ζ_2 depending on x and the bit to be revealed. This extends to strings by committing bitwise.

Sketch of the Two-Party KO Protocol. The main component of the two-party KO protocol is Yao's 3-round protocol $\Pi_{\mathsf{S}H}$, described above, secure against semi-honest adversaries. In order to achieve security against a malicious adversary their protocol proceeds as follows. Both parties commit to their inputs; run (modified) coin-tossing protocols to guarantee that each party obtains random coins which are committed to the other party (note that coin flipping for the side of the garbler P_2 is not needed since a malicious garbler P_2 gains nothing by using non-uniform coins. To force P_1 to use random coins the authors use a 3-round sub-protocol which is based on the work of [BL02]); and run the $\Pi_{\mathsf{S}H}$ protocol together with ZK arguments to avoid adversarial inconsistencies in each round. Then, simulation extractability is guaranteed by the use of WI proof of knowledge and output simulation by the Feige-Shamir ZK argument of knowledge.

However, since even a ZK argument for the first round of the protocol alone will already require 4 rounds, the authors use specific proof systems to achieve in total a 4-round protocol. In particular, the KO protocol uses a specific WI proof of knowledge system with the property that the statement to be proven need not be known until the last round of the protocol, yet soundness, completeness, and witness-indistinguishability still hold. Also, this proof system has the property that the first message from the prover is computed independently of the statement being proved. Note that their 4-round ZK argument of knowledge enjoys the same properties. Furthermore, their protocol uses an equivocal commitment scheme to commit to the garble circuit for the following reason. Party P_1 may send his round-two message before the proof of correctness for round one given by P_2 is complete. Therefore, the protocol has to be constructed in a way that the proof of correctness for round one completes in round three and that party

P_2 reveals the garbled circuit in the third round. But since the proof of security requires P_2 to commit to a garble circuit at the end of the first round, P_2 does so using an equivocal commitment scheme.

4.3 The Exact Round Complexity of Coin Tossing

In this section we first show that it is impossible to construct two-party (simulatable) coin-flipping for a super-logarithmic number of coins in 3 simultaneous message exchange rounds. We first recall the definition of a simulatable coin flipping protocol using the real/ideal paradigm from [KOS03].

Definition 4.4 ([KOS03])**.** An n-party protocol Π is a simulatable coin-flipping protocol if it is an $(n-1)$-secure protocol realizing the coin-flipping functionality. That is, for every PPT adversary $\mathcal{A}$ corrupting at most $n-1$ parties there exists an expected PPT simulator $\mathcal{S}$ such that the (output of the) following experiments are indistinguishable. Here we parse the result of

$\mathrm{REAL}(1^\kappa, 1^\lambda)$	$\mathrm{IDEAL}(1^\kappa, 1^\lambda)$
$c, \mathsf{view}_\mathcal{A} \leftarrow \mathrm{REAL}_{\Pi,\mathcal{A}}(1^\kappa, 1^\lambda)$	$c' \leftarrow \{0,1\}^\lambda$
	$\widetilde{c}, \mathsf{view}_\mathcal{S} \leftarrow \mathcal{S}^\mathcal{A}(c', 1^\kappa, 1^\lambda)$
Output $(c, \mathsf{view}_\mathcal{A})$	If $\widetilde{c} = \{c', \perp\}$ then Output $(\widetilde{c}, \mathsf{view}_\mathcal{S})$
	Else output `fail`

running protocol Π with adversary $\mathcal{A}$ (denoted by $\mathrm{REAL}_{\Pi,\mathcal{A}}(1^\kappa, 1^\lambda)$) as a pair $(c, \mathsf{view}_\mathcal{A})$ where $c \in \{0,1\}^\lambda \cup \{\perp\}$ is the outcome and $\mathsf{view}_\mathcal{A}$ is the view of the adversary $\mathcal{A}$.

We restrict ourselves to the case of two parties ($n = 2$), which can be extended to any $n > 2$. Below we denote messages in protocol Π which are sent by party P_i to party P_j in the ρ-th round by $\mathsf{m}_{i,j}^{\Pi[\rho]}$.

As mentioned earlier, Katz and Ostrovsky [KO04] showed that simulatable coin-flipping protocol is impossible in 4 rounds without simultaneous message exchange. Since we will use the result for our proofs in this section, we state their result below without giving their proof.

Lemma 4.1. [KO04, Theorem 1] Let $p(\kappa) = \omega(\log \kappa)$, where κ is the security parameter. Then there does not exist a 4-round protocol *without simultaneous message transmission* for tossing $p(\kappa)$ coins which can be proven secure via black-box simulation.

In the following, we state our impossibility result for coin-fliping in 3 rounds of simultaneous message exchange.

Lemma 4.2. Let $p(\kappa) = \omega(log\kappa)$, where κ is the security parameter. Then there does not exist a 3-round protocol with simultaneous message transmission for tossing $p(\kappa)$ coins which can be proven secure via black-box simulation.

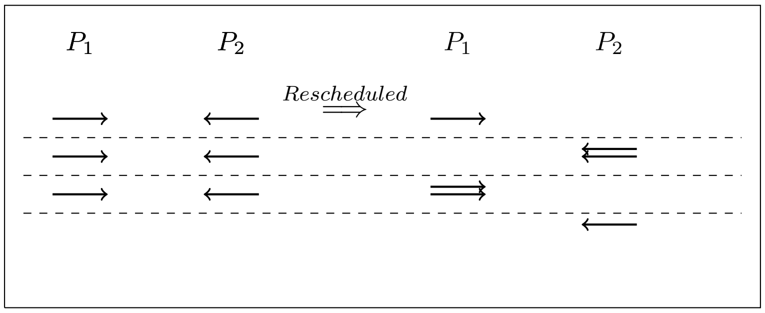


Figure 4.1: [GMPP16] A 3-round simultaneous protocol rescheduled to a 4-round non-simultaneous protocol.

Proof. We prove the above statement by showing that a 3-round simultaneous message exchange protocol can be "rescheduled" to a 4-round *non-simultaneous* protocol which contradicts the impossibility of [KO04]. Here by rescheduling we mean rearrangement of the messages without violating mutual dependencies among them, in particular without altering the next-message functions.

For the sake of contradiction, assume that there exists a protocol $\Pi_{\mathsf{flip}}^{\Leftrightarrow}$ which realizes simulatable coin-flipping in 3 simultaneous message exchange rounds, then we can reschedule it in order to construct a protocol $\Pi_{\mathsf{flip}}^{\leftrightarrows}$ which realizes simulatable coin-flipping in 4 rounds[5] without simultaneous message exchange as follows:

Protocol $\Pi_{\mathsf{flip}}^{\leftrightarrows}$

Round-1: P_1 sends the first message $\mathsf{m}_{1,2}^{\Pi_{\mathsf{flip}}^{\leftrightarrows}[1]} := \mathsf{m}_{1,2}^{\Pi_{\mathsf{flip}}^{\Leftrightarrow}[1]}$ to P_2.

Round-2: Party P_2 sends to P_1 the second message

$$\mathsf{m}_{2,1}^{\Pi_{\mathsf{flip}}^{\leftrightarrows}[2]} := (\mathsf{m}_{2,1}^{\Pi_{\mathsf{flip}}^{\Leftrightarrow}[1]}, \mathsf{m}_{2,1}^{\Pi_{\mathsf{flip}}^{\Leftrightarrow}[2]}).$$

Round-3: Party P_1 sends to P_2 the third message

$$\mathsf{m}_{1,2}^{\Pi_{\mathsf{flip}}^{\leftrightarrows}[3]} := (\mathsf{m}_{1,2}^{\Pi_{\mathsf{flip}}^{\Leftrightarrow}[2]}, \mathsf{m}_{1,2}^{\Pi_{\mathsf{flip}}^{\Leftrightarrow}[3]}).$$

Round-4: Finally P_2 sends to P_1 the last message $\mathsf{m}_{2,1}^{\Pi_{\mathsf{flip}}^{\leftrightarrows}[4]} := \mathsf{m}_{2,1}^{\Pi_{\mathsf{flip}}^{\Leftrightarrow}[3]}$.

We provide a pictorial presentation of the above rescheduling in Fig. 4.1 for better illustration.

Now, without loss of generality assume that P_1 is corrupted. Then we need to build an expected PPT simulator $\mathcal{S}_{P_1}$ (or simply $\mathcal{S}$) meeting the adequate requirements (according to Def. 4.4). First note that, since by assumption the protocol $\Pi_{\mathsf{flip}}^{\Leftrightarrow}$ is secure (i.e. achieves Def. 4.4)

[5]The superscript $\Leftrightarrow$ stands for the simultaneous message exchange setting and $\leftrightarrows$ for the setting without simultaneous message exchange

the following holds: for any corrupt $P_1^{\Leftrightarrow}$ executing the simultaneous message exchange protocol $\Pi_{\mathsf{flip}}^{\Leftrightarrow}$ there exists an expected PPT simulator $\mathcal{S}^{\Leftrightarrow}$ (let us call it the "inner" simulator and $\mathcal{S}$ the "outer" simulator) in the ideal world. So, $\mathcal{S}$ can be constructed using $\mathcal{S}^{\Leftrightarrow}$ for a corrupted party $P_1^{\Leftrightarrow}$ which can be *emulated* by $\mathcal{S}$ based on P_1. Finally, $\mathcal{S}$ just outputs whatever $\mathcal{S}^{\Leftrightarrow}$ returns. $\mathcal{S}$ emulates the interaction between $\mathcal{S}^{\Leftrightarrow}$ and $P_1^{\Leftrightarrow}$ as follows:

1. On receiving a value $c' \in \{0,1\}^{\lambda}$ from the ideal functionality, $\mathcal{S}$ runs the inner simulator $\mathcal{S}^{\Leftrightarrow}(c', 1^{\kappa}, 1^{\lambda})$ to get the first message $\mathsf{m}_{2,1}^{\Pi_{\mathsf{flip}}^{\Leftrightarrow}[1]}$. Notice that in protocol $\Pi_{\mathsf{flip}}^{\Leftrightarrow}$ the first message from (honest) party $P_2^{\Leftrightarrow}$ does not depend on the first message of the corrupted party $P_1^{\Leftrightarrow}$. So, the inner simulator must be able to produce the first message even before seeing the first message of party P_1 (or the emulated party $P_1^{\Leftrightarrow}$)[6]. Then it runs P_1 to receive the first message $\mathsf{m}_{1,2}^{\Pi_{\mathsf{flip}}^{\Leftrightarrow}[1]}$.

2. Then $\mathcal{S}$ forwards $\mathsf{m}_{1,2}^{\Pi_{\mathsf{flip}}^{\Leftrightarrow}[1]}$ to the inner simulator which then returns the second simulated message $\mathsf{m}_{2,1}^{\Pi_{\mathsf{flip}}^{\Leftrightarrow}[2]}$. Now $\mathcal{S}$ can construct the simulated message $\mathsf{m}_{2,1}^{\Pi_{\mathsf{flip}}^{\overset{\leftarrow}{\rightarrow}}[2]}$ by combining $\mathsf{m}_{2,1}^{\Pi_{\mathsf{flip}}^{\Leftrightarrow}[2]}$ and $\mathsf{m}_{2,1}^{\Pi_{\mathsf{flip}}^{\Leftrightarrow}[1]}$ received earlier (see above) which $\mathcal{S}$ then forwards to P_1.

3. In the next step, $\mathcal{S}$ gets back messages $\mathsf{m}_{1,2}^{\Pi_{\mathsf{flip}}^{\overset{\leftarrow}{\rightarrow}}[3]} = (\mathsf{m}_{1,2}^{\Pi_{\mathsf{flip}}^{\Leftrightarrow}[2]}, \mathsf{m}_{1,2}^{\Pi_{\mathsf{flip}}^{\Leftrightarrow}[3]})$ from P_1. It then forwards the second message $\mathsf{m}_{1,2}^{\Pi_{\mathsf{flip}}^{\Leftrightarrow}[2]}$ to $\mathcal{S}^{\Leftrightarrow}$, which then returns the third simulated message $\mathsf{m}_{2,1}^{\Pi_{\mathsf{flip}}^{\Leftrightarrow}[3]}$. Finally it forwards the third message $\mathsf{m}_{1,2}^{\Pi_{\mathsf{flip}}^{\Leftrightarrow}[3]}$ to $\mathcal{S}^{\Leftrightarrow}$.

4. $\mathcal{S}$ outputs whatever transcript $\mathcal{S}^{\Leftrightarrow}$ outputs in the end.

5. Note that, whenever the inner simulator $\mathcal{S}^{\Leftrightarrow}$ asks to rewind the emulated $P_1^{\Leftrightarrow}$, $\mathcal{S}$ rewinds P_1.

It is not hard to see that the simulator $\mathcal{S}$ emulates correctly the party $P_1^{\Leftrightarrow}$ and hence by the security of $\Pi_{\mathsf{flip}}^{\Leftrightarrow}$, the inner simulator $\mathcal{S}^{\Leftrightarrow}$ returns an indistinguishable (with the real world) view. The key-point is that the re-scheduling of the messages from protocol $\Pi_{\mathsf{flip}}^{\Leftrightarrow}$ does not affect the dependency (hence the corresponding next message functions) and hence the correctness and security remains intact in $\Pi_{\mathsf{flip}}^{\overset{\leftarrow}{\rightarrow}}$.

We stress that the proof for the case where P_2 is corrupted is straightforward given the above. However, in that case, since P_2's first message depends on the first message of honest P_1, it is mandatory for the inner simulator $\mathcal{S}^{\Leftrightarrow}$ to output the first message before seeing anything even in order to run the corrupted P_2 which is not necessary in the above case. As we stated earlier this is possible as the inner simulator $\mathcal{S}^{\Leftrightarrow}$ should be able to handle rushing adversaries.

Hence we prove that if the underlying protocol $\Pi_{\mathsf{flip}}^{\Leftrightarrow}$ securely realizes simulatable coin-flipping in 3 simultaneous rounds then $\Pi_{\mathsf{flip}}^{\overset{\leftarrow}{\rightarrow}}$ securely realizes coin-flipping in 4 non-simultaneous rounds which contradicts the KO lower bound (Lemma 4.1). This concludes the proof.

[6]In particular, for so-called "rushing" adversaries, who can wait until receiving the first message and then send its own, the inner simulator must simulate the first message to get the first message from the adversary.

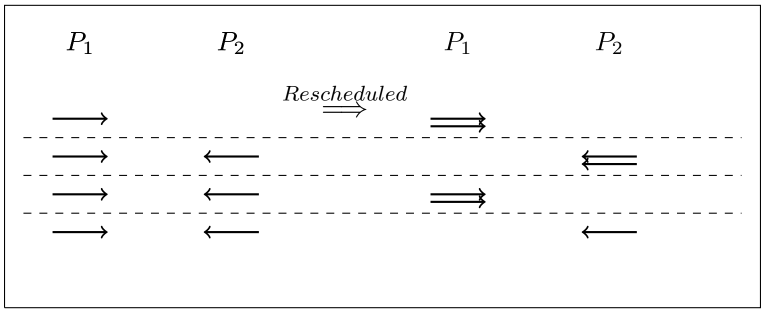


Figure 4.2: [GMPP16] Rescheduling when P_2 does not send the first message.

■

Going a step further we show that any four-round simultaneous message exchange protocol realizing simulatable coin-flipping must satisfy a necessary property, that is each round must be a strictly simultaneous message exchange round, in other words, both parties must send some "non-redundant" message in each round. By "non-redundant" we mean that the next message from the other party must depend on the current message. Below we show the above, otherwise the messages can be again subject to a "rescheduling" mechanism similar to the one in Lemma 4.2, to yield a four-round non-simultaneous protocol; thus contradicting Lemma 4.1. More specifically,

Lemma 4.3. Let $p(\kappa) = \omega(log\kappa)$, where κ is the security parameter. Then there does not exist a 4-round protocol with at least one unidirectional round (i.e. a round without simultaneous message exchange) for tossing $p(\kappa)$ coins which can be proven secure via black-box simulation.

Proof (Sketch). We provide a sketch for any protocol with exactly one unidirectional round where only one party, say P_1 sends a message to P_2. Clearly, there can be four such cases where P_2's message is omitted in one of the four rounds. In Fig. 4.2 we show the case where P_2 does not send the message in the first round, and any such protocol can be re-scheduled (similar to the proof of Lemma 4.2) to a non-simultaneous 4-round protocol without altering any possible message dependency. This observation can be formalized in a straightforward manner following the proof of Lemma 4.2 and hence we omit the details. Therefore, again combining with the impossibility from Lemma 4.1 by [KO04] such simultaneous protocol can not realize simulatable coin-flipping. The other cases can be easily observed by similar rescheduling trick and therefore we omit the details for those cases. ■

4.4 Two-Party Computation in the Simultaneous Message Exchange Model

In this section, we present our two party protocol for computing any functionality in the presence of a static, malicious and rushing adversary. As discussed earlier, we are in the simultaneous message exchange channel setting where both parties can simultaneously exchange messages in each round. The structure of this protocol will provide a basis for our later protocols as well.

An overview of the protocol appears in Section 4.1. In a high level, the protocol consists of two simultaneous executions of a one-sided (single-output) protocol to guarantee that both parties learn the output. The overall skeleton of the one-sided protocol resembles the KO protocol [KO04] which uses a clever combination of OT, coin-tossing, and Π_{WIPOK} to ensure that the protocol is executed with a fixed input (allowing at the same time simulation extractability of the input), and relies on the zero-knowledge property of Π_{FS} to "force the output". A sketch of the KO protocol is given in Section 4.2.2. In order to ensure "independence of inputs" our protocol relies heavily on non-malleable commitments. To this end, we change the one-sided protocol to further incorporate non-malleable commitments so that similar guarantees can be obtained even in the presence of the "opposite side" protocol, and we further rely on zero-knowledge proofs to ensure that parties use the same input in both executions.

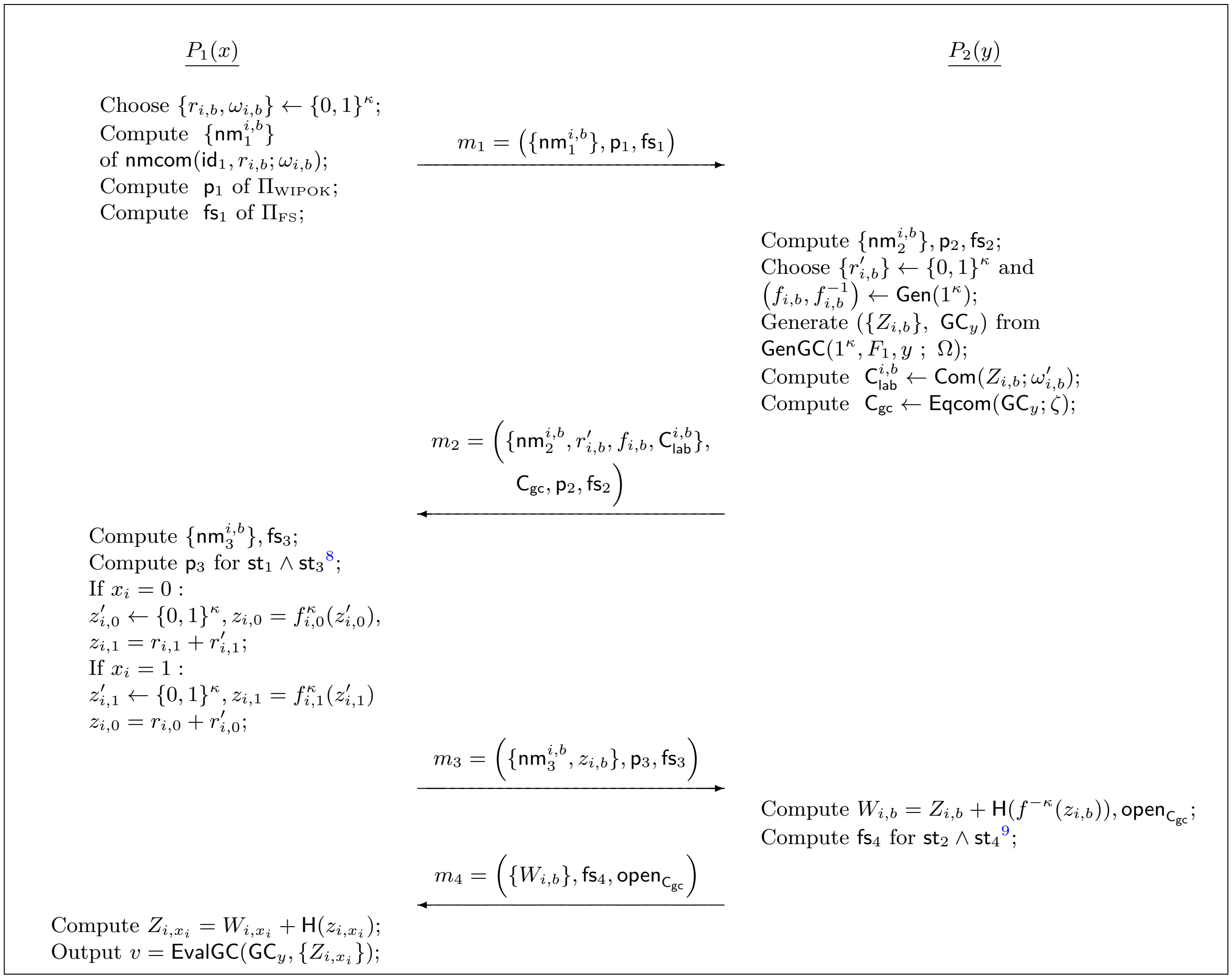
4.4.1 Our Protocol

To formally define our protocol, let:

- $(\mathsf{GenGC}, \mathsf{EvalGC})$ be the garbled-circuit mechanism with simulator SimGC; $\mathcal{F} = (\mathsf{Gen}, \text{Eval}, \text{Invert})$ be a family of TDPs with domain $\{0,1\}^\kappa$; H be the hardcore bit function for κ bits; com be a perfectly binding non-interactive commitment scheme; Eqcom be the equivocal scheme based on com, as described in Section 6.1.3;
- nmcom be a tag based, *parallel*[7] and *3-robust* non-malleable commitment scheme for strings, supporting tags/identities of length κ;
- Π_{WIPOK} be the witness-indistinguishable proof-of-knowledge for NP as described in Section 6.1.3;
- Π_{FS} be the proof system for NP, based on nmcom and Π_{WIPOK}, as described in Section 6.1.3;
- **Simplifying assumption:** *for notational convenience only*, we assume for now that nmcom consists of exactly three rounds, denoted by $(\mathsf{nm}_1, \mathsf{nm}_2, \mathsf{nm}_3)$. This assumption is removed later (see Remark 4.1).

  We also assume that the first round, nm_1, is from the committer and statistically determines the message to be committed. We use the notation $\mathsf{nm}_1 = \mathsf{nmcom}_1(\mathsf{id}, r; \omega)$ to denote the committer's first message when executing nmcom with identity id to commit to string r with randomness ω.

[7] We actually need security against an a-priori bounded number of polynomial executions. Almost all known protocols for nmcom have this additional property.



$\underline{P_1(x)}$

Choose $\{r_{i,b}, \omega_{i,b}\} \leftarrow \{0,1\}^\kappa$;
Compute $\{\mathsf{nm}_1^{i,b}\}$
of $\mathsf{nmcom}(\mathsf{id}_1, r_{i,b}; \omega_{i,b})$;
Compute p_1 of Π_{WIPOK};
Compute fs_1 of Π_{FS};

$$m_1 = \left(\{\mathsf{nm}_1^{i,b}\}, \mathsf{p}_1, \mathsf{fs}_1\right) \longrightarrow$$

$\underline{P_2(y)}$

Compute $\{\mathsf{nm}_2^{i,b}\}, \mathsf{p}_2, \mathsf{fs}_2$;
Choose $\{r'_{i,b}\} \leftarrow \{0,1\}^\kappa$ and
$\left(f_{i,b}, f_{i,b}^{-1}\right) \leftarrow \mathsf{Gen}(1^\kappa)$;
Generate $(\{Z_{i,b}\},\ \mathsf{GC}_y)$ from
$\mathsf{GenGC}(1^\kappa, F_1, y\ ;\ \Omega)$;
Compute $\mathsf{C}_{\mathsf{lab}}^{i,b} \leftarrow \mathsf{Com}(Z_{i,b}; \omega'_{i,b})$;
Compute $\mathsf{C}_{\mathsf{gc}} \leftarrow \mathsf{Eqcom}(\mathsf{GC}_y; \zeta)$;

$$m_2 = \left(\{\mathsf{nm}_2^{i,b}, r'_{i,b}, f_{i,b}, \mathsf{C}_{\mathsf{lab}}^{i,b}\}, \mathsf{C}_{\mathsf{gc}}, \mathsf{p}_2, \mathsf{fs}_2\right) \longleftarrow$$

Compute $\{\mathsf{nm}_3^{i,b}\}, \mathsf{fs}_3$;
Compute p_3 for $\mathsf{st}_1 \wedge \mathsf{st}_3$[8];
If $x_i = 0$:
$z'_{i,0} \leftarrow \{0,1\}^\kappa, z_{i,0} = f_{i,0}^\kappa(z'_{i,0})$,
$z_{i,1} = r_{i,1} + r'_{i,1}$;
If $x_i = 1$:
$z'_{i,1} \leftarrow \{0,1\}^\kappa, z_{i,1} = f_{i,1}^\kappa(z'_{i,1})$
$z_{i,0} = r_{i,0} + r'_{i,0}$;

$$m_3 = \left(\{\mathsf{nm}_3^{i,b}, z_{i,b}\}, \mathsf{p}_3, \mathsf{fs}_3\right) \longrightarrow$$

Compute $W_{i,b} = Z_{i,b} + \mathsf{H}(f^{-\kappa}(z_{i,b})), \mathsf{open}_{\mathsf{C}_{\mathsf{gc}}}$;
Compute fs_4 for $\mathsf{st}_2 \wedge \mathsf{st}_4$[9];

$$m_4 = \left(\{W_{i,b}\}, \mathsf{fs}_4, \mathsf{open}_{\mathsf{C}_{\mathsf{gc}}}\right) \longleftarrow$$

Compute $Z_{i,x_i} = W_{i,x_i} + \mathsf{H}(z_{i,x_i})$;
Output $v = \mathsf{EvalGC}(\mathsf{GC}_y, \{Z_{i,x_i}\})$;

Figure 4.3: [GMPP16] High-level description of the left execution of Π_{2PC}.

We are now ready to describe our protocol. A high level sketch of the left execution of our protocol where P_1 receives the output is given in Figure 4.3.

Protocol Π_{2PC}. We denote the two parties by P_1 and P_2; P_1 holds input $x \in \{0,1\}^\kappa$ and P_2 holds input $y \in \{0,1\}^\kappa$. Furthermore, the identities of P_1, P_2 are $\mathsf{id}_1, \mathsf{id}_2$ respectively where

[8]Informally, st_1 represents that P_1 "knows" one of the decommitment values of the first round for every i and st_3 says that P_1 correctly constructed $\{z_{i,b}\}$.

[9]Informally, st_2 is the statement that P_2 performed his first step correctly and st_4 is the statement that P_2 performed *both* oblivious transfers correctly.

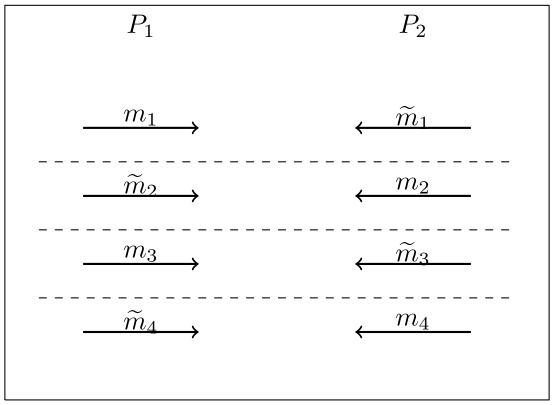


Figure 4.4: [GMPP16] 2-PC in the simultaneous message exchange model.

$\mathsf{id}_1 \neq \mathsf{id}_2$. Let $F := (F_1, F_2) : \{0,1\}^\kappa \times \{0,1\}^\kappa \to \{0,1\}^\kappa \times \{0,1\}^\kappa$ be the functions to be computed.

The protocol consists of four (strictly) simultaneous message exchange rounds, i.e., both parties send messages in each round. The protocol essentially consists of two simultaneous executions of a protocol in which only one party learns the output. In the first protocol, P_1 learns the output and the messages of this protocol are denoted by (m_1, m_2, m_3, m_4) where (m_1, m_3) are sent by P_1 and (m_2, m_4) are sent by P_2. Likewise, in the second protocol P_2 learns the output and the messages of this protocol are denoted by $(\widetilde{m}_1, \widetilde{m}_2, \widetilde{m}_3, \widetilde{m}_4)$ where $(\widetilde{m}_1, \widetilde{m}_3)$ are sent by P_2 and $(\widetilde{m}_2, \widetilde{m}_4)$ are sent by P_1. Therefore, messages $(m_j, \widetilde{m}_j)$ are exchanged simultaneously in the j-th round, $j \in \{1, \ldots, 4\}$ (see figure 4.4).

We now describe how these messages are constructed in each round below. In the following i always ranges from 1 to κ and b from 0 to 1.

Round 1. In this round P_1 sends a message m_1 and P_2 sends a symmetrically constructed message $\widetilde{m}_1$. We first describe how P_1 constructs m_1.

Actions of P_1:

1. P_1 starts by committing to 2κ random strings $\{(r_{1,0}, r_{1,1}), \ldots, (r_{\kappa,0}, r_{\kappa,1})\}$ using 2κ parallel and independent executions of nmcom with identity id_1. I.e., it uniformly chooses strings $r_{i,b}$, randomness $\omega_{i,b}$, and generates $\mathsf{nm}_1^{i,b}$ which is the first message corresponding to the execution of $\mathsf{nmcom}(\mathsf{id}_1, r_{i,b}; \omega_{i,b})$.
2. P_1 prepares the first message p_1 of Π_{WIPOK}, as well as the first message fs_1 of Π_{FS}. For later reference, define st_1 to be the following: $\exists\{(r_i, \omega_i)\}_{i\in[\kappa]}$ s.t.:

$$\forall i : \left(\mathsf{nm}_1^{i,0} = \mathsf{nmcom}_1(\mathsf{id}_1, r_i; \omega_i) \vee \mathsf{nm}_1^{i,1} = \mathsf{nmcom}_1(\mathsf{id}_1, r_i; \omega_i)\right)$$

Informally, st_1 represents that P_1 "knows" one of the decommitment values for every i.

3. Message m_1 is defined to be the tuple $\left(\{\mathsf{nm}_1^{i,b}\}, \mathsf{p}_1, \mathsf{fs}_1\right)$.

Actions of P_2:

Performs the same actions as P_1 to sample the values $\{(\widetilde{r}_{i,b}, \widetilde{\omega}_{i,b})\}$ and constructs $\widetilde{m}_1 := \left(\{\widetilde{\mathsf{nm}}_1^{i,b}\}, \widetilde{\mathsf{p}}_1, \widetilde{\mathsf{fs}}_1\right)$ where all $\widetilde{\mathsf{nm}}_1^{i,b}$ are generated with id_2. Define the statement $\widetilde{\mathsf{st}}_1$ analogously for these values.

Round 2. In this round P_2 sends a message m_2 and P_1 sends a symmetrically constructed message $\widetilde{m}_2$. We first describe how P_2 constructs m_2.

Actions of P_2:

1. P_2 generates the second messages $\{\mathsf{nm}_2^{i,b}\}$ corresponding to all executions of nmcom initiated by P_1 (with id_1).
2. P_2 prepares the second message p_2 of the Π_{WIPOK} protocol initiated by P_1.
3. P_2 samples random strings $\{r'_{i,b}\}$ and $(f_{i,b}, f_{i,b}^{-1}) \leftarrow \mathsf{Gen}(1^\kappa)$ for the oblivious transfer executions.
4. P_2 obtains the garbled labels and the circuit for F_1: $(\{Z_{i,b}\},\ \mathsf{GC}_y) = \mathsf{GenGC}(1^\kappa, F_1, y\ ;\ \Omega)$.
5. P_2 generates standard commitments to the labels, and an equivocal commitment to the garbled circuit: i.e., $\mathsf{C}_{\mathsf{lab}}^{i,b} \leftarrow \mathsf{Com}(Z_{i,b}; \omega'_{i,b})$ and $\mathsf{C}_{\mathsf{gc}} \leftarrow \mathsf{Eqcom}(\mathsf{GC}_y; \zeta)$.
6. P_2 prepares the second message fs_2 of the Π_{FS} protocol initiated by P_1.

   For later reference, define st_2 to be the following: $\exists\ (y, \Omega, \mathsf{GC}_y, \{Z_{i,b}, \omega'_{i,b}\}, \zeta)$ s.t.:

   a) $(\{Z_{i,b}\},\ \mathsf{GC}_y) = \mathsf{GenGC}(1^\kappa, F_1, y\ ;\ \Omega)$
   b) $\forall (i,b) : \mathsf{C}_{\mathsf{lab}}^{i,b} = \mathsf{Com}(Z_{i,b}; \omega'_{i,b})$
   c) $\mathsf{C}_{\mathsf{gc}} = \mathsf{Eqcom}(\mathsf{GC}_y; \zeta)$

   (Informally, st_2 is the statement that P_2 performed this step correctly.)
7. Define message $m_2 := \left(\{\mathsf{nm}_2^{i,b}, r'_{i,b}, f_{i,b}, \mathsf{C}_{\mathsf{lab}}^{i,b}\}, \mathsf{C}_{\mathsf{gc}}, \mathsf{p}_2, \mathsf{fs}_2\right)$.

Actions of P_1:

Performs the same actions as P_2 in the previous step to construct the message $\widetilde{m}_2 := \left(\{\widetilde{\mathsf{nm}}_2^{i,b}, \widetilde{r}'_{i,b}, \widetilde{f}_{i,b}, \widetilde{\mathsf{C}}_{\mathsf{lab}}^{i,b}\}, \widetilde{\mathsf{C}}_{\mathsf{gc}}, \widetilde{\mathsf{p}}_2, \widetilde{\mathsf{fs}}_2\right)$ w.r.t. identity id_2, function F_2, and input x. Define the (remaining) values $\widetilde{f}'^{-1}_{i,b}, \widetilde{Z}_{i,b}, \widetilde{\omega}'_{i,b}, \mathsf{GC}_x, \widetilde{\Omega}, \widetilde{\zeta}$ and statement $\widetilde{\mathsf{st}}_2$ analogously.

Round 3. In this round P_1 sends a message m_3 and P_2 sends a symmetrically constructed message $\widetilde{m}_3$. We first describe how P_1 constructs m_3.

Actions of P_1:

1. P_1 prepares the third message $\{\mathsf{nm}_3^{i,b}\}$ of nmcom (with id_1).

2. If any of $\{f_{i,b}\}$ are invalid, P_1 aborts. Otherwise, it invokes κ parallel executions of oblivious transfer to obtain the input-wire labels corresponding to its input x. More specifically, P_1 proceeds as follows:

   – If $x_i = 0$, sample $z'_{i,0} \leftarrow \{0,1\}^\kappa$, set $z_{i,0} = f^\kappa_{i,0}(z'_{i,0})$, and $z_{i,1} = r_{i,1} + r'_{i,1}$.

   – If $x_i = 1$, sample $z'_{i,1} \leftarrow \{0,1\}^\kappa$, set $z_{i,1} = f^\kappa_{i,1}(z'_{i,1})$, and $z_{i,0} = r_{i,0} + r'_{i,0}$.

3. Define st_3 to be the following: $\exists\{(r_i, \omega_i)\}_{i\in[\kappa]}$ s.t. $\forall i$:

   a) $(\mathsf{nm}_1^{i,0} = \mathsf{nmcom}_1(\mathsf{id}_1, r_i; \omega_i) \wedge z_{i,0} = r_i + r'_{i,0})$, **or**

   b) $(\mathsf{nm}_1^{i,1} = \mathsf{nmcom}_1(\mathsf{id}_1, r_i; \omega_i) \wedge z_{i,1} = r_i + r'_{i,1})$

   Informally, st_3 says that P_1 correctly constructed $\{z_{i,b}\}$.

4. P_1 prepares the final message p_3 of Π_{WIPOK} proving the statement: $\mathsf{st}_1 \wedge \mathsf{st}_3$.[10] P_1 also prepares the third message fs_3 of Π_{FS}.

5. Define $m_3 := \Big(\{\mathsf{nm}_3^{i,b}, z_{i,b}\}, \mathsf{p}_3, \mathsf{fs}_3\Big)$ to P_2.

<u>Actions of P_2:</u>

Performs the same actions as P_1 in the previous step to construct the message $\widetilde{m}_3 := \Big(\{\widetilde{\mathsf{nm}}_3^{i,b}, \widetilde{z}_{i,b}\}, \widetilde{\mathsf{p}}_3, \widetilde{\mathsf{fs}}_3\Big)$ w.r.t. identity id_2 and input y. The (remaining) values $\{\widetilde{z}_{i,b}, \widetilde{z}'_{i,b}\}$ and statement $\widetilde{\mathsf{st}}_3$ are defined analogously.

Round 4. In this round P_2 sends a message m_4 and P_1 sends a symmetrically constructed message $\widetilde{m}_4$. We first describe how P_2 constructs m_4.

<u>Actions of P_2:</u>

1. If $\mathsf{p}_3, \mathsf{fs}_3$ are not accepting, P_2 aborts. Otherwise, P_2 completes the execution of the oblivious transfers for every (i,b). I.e., it computes $W_{i,b} = Z_{i,b} + \mathsf{H}(f^{-\kappa}(z_{i,b}))$. Moreover, P_2 decommits $\mathsf{C_{gc}}$ as GC_y, denoted by $\mathsf{open}_{\mathsf{C_{gc}}}$ to P_i.

2. Define st_4 to be the following: $\exists\ (y, \Omega, \mathsf{GC}_y, \{Z_{i,b}\}, \omega'_{i,b}, z'_{i,b}, \widetilde{z}'_i\}_{i\in[\kappa], b\in\{0,1\}})$ s.t.

   a) $\forall(i,b)$: $\Big(\mathsf{C}^{i,b}_{\mathsf{lab}} = \mathsf{Com}(Z_{i,b}; \omega'_{i,b})\Big) \bigwedge \Big(f^\kappa_{i,b}(z'_{i,b}) = z_{i,b}\Big) \bigwedge \Big(W_{i,b} = Z_{i,b} + \mathsf{H}((z'_{i,b}))\Big)$

   b) $((\{Z_{i,b}\},\ \mathsf{GC}_y) = \mathsf{GenGC}(1^\kappa, F_1, y\ ;\ \Omega)) \bigwedge (\mathsf{C_{gc}} = \mathsf{Eqcom}(\mathsf{GC}_y; \zeta))$

   c) $\forall i$: $\widetilde{z}_{i,y_i} = \widetilde{f}^\kappa_{i,y_i}(\widetilde{z}'_i)$

   Informally, this means that P_2 performed *both* oblivious transfers correctly.

3. P_2 prepares the final message fs_4 of Π_{FS} proving the statement $\mathsf{st}_2 \wedge \mathsf{st}_4$.[11]

4. Define $m_4 := \Big(\{W_{i,b}\}, \mathsf{fs}_4, \mathsf{open}_{\mathsf{C_{gc}}}\Big)$.

[10]Honest P_1 knows multiple witnesses for st_1. For concreteness, we have to use one of them randomly in the proof.

[11]Recall that Π_{FS} is a modified version of FS protocol: it uses two executions of nmcom to construct its first message, namely, the first message consists of $(\mathsf{nm}_1^1, \mathsf{nm}_1^2))$ corresponding to two executions of nmcom committing to strings σ_1, σ_2 (see Section 6.1.3).

Actions of P_1:

Performs the same actions as P_2 in the previous step to construct the message $\widetilde{m}_4 := \left(\{\widetilde{W}_{i,b}\}, \widetilde{\mathsf{fs}}_4, \mathsf{topen}_{\mathsf{C_{gc}}}\right)$ and analogously defined statement $\widetilde{\mathsf{st}}_4$.

Output compuation.

P_1's output: If any of $(\mathsf{fs}_4, \mathsf{GC}_y, \mathsf{open}_{\mathsf{C_{gc}}})$ or the openings of $\{W_{i,b}\}$ are invalid, P_1 aborts. Otherwise, P_1 recovers the garbled labels $\{Z_i := Z_{i,x_i}\}$ from the completion of the oblivious transfer, and computes $F_1(x, y) = \mathsf{EvalGC}(\mathsf{GC}_y, \{Z_i\})$.

P_2's output: If any of $(\widetilde{\mathsf{fs}}_4, \mathsf{GC}_x, \mathsf{topen}_{\mathsf{C_{gc}}})$ or the openings of $\{\widetilde{W}_{i,b}\}$ are invalid, P_2 aborts. Otherwise, P_2 recovers the garbled labels $\{\widetilde{Z}_i := \widetilde{Z}_{i,y_i}\}$ from the completion of the oblivious transfer, and computes $F_2(x, y) = \mathsf{EvalGC}(\mathsf{GC}_x, \{\widetilde{Z}_i\})$.

Remark 4.1. If nmcom has $k > 3$ rounds, the first $k - 3$ rounds can be performed before the 4 rounds of Π_{2PC} start; this results in a protocol with $k + 1$ rounds. If $k < 3$, then the protocol has only 4 rounds. Also, for large k, it suffices if the first $k - 2$ rounds of nmcom statistically determine the message to be committed; the notation is adjusted to simply use the transcript up to $k - 2$ rounds to define the statements for the proof systems.

Finally, the construction is described for a deterministic F. Known transformations (see [Gol04, Section 7.3]) yield a protocol for randomized functionalities, without increasing the rounds.

4.4.2 Proof of Security

We prove the security of our protocol according to the ideal/real paradigm. We design a sequence of hybrids where we start with the real world execution and gradually modify it until the input of the honest party is not needed. The resulting final hybrid represents the simulator for the ideal world.

Theorem 4.4.1. Assuming the existence of a trapdoor permutation family and a k-round parallel and 3-robust non-malleable commitment scheme, protocol Π_{2PC} securely computes every two-party functionality $F = (F_1, F_2)$ with black-box simulation in the presence of a malicious adversary. The round complexity of Π_{2PC} is $k' = \max(4, k + 1)$.

Proof. Due to the symmetric nature of our protocol, it is *sufficient* to prove security against the malicious behavior of any party, say P_1. We show that for every adversary $\mathcal{A}$ who participates as P_1 in the "real" world execution of Π_{2PC}, there exists an "ideal" world adversary (simulator) $\mathcal{S}$ such that for all inputs x, y of equal length and security parameter $\kappa \in \mathbb{N}$:

$$\{\text{IDEAL}_{F,\mathcal{S}}(\kappa, x, y)\}_{\kappa,x,y} \overset{\text{c}}{\approx} \{\text{REAL}_{\Pi,\mathcal{A}}(\kappa, x, y)\}_{\kappa,x,y}$$

We prove this claim by considering hybrid experiments $H_0, H_1, \ldots$ as described below. We denote by $\mathbf{Hybrid}^i_{F,\mathcal{S}^i}(\kappa, x, y)$ the random variable that corresponds to the simulator's output in hybrid execution $\mathrm{H_i}$ when running against party $\mathcal{S}^i$ that plays the role of P_2 according to the specifications in this hybrid (where $\mathcal{S}^0$ refers to the honest real sender). For simplicity of exposition, we abbreviate it to Hybrid_i.

We start with H_0 which has access to both inputs x and y, and gradually get rid of the honest party's input y to reach the final hybrid.

Hybrid H_0: Identical to the real execution. More specifically, H_0 starts the execution of $\mathcal{A}$ providing it fresh randomness and input x, and interacts with it honestly by performing all actions of P_2 with uniform randomness and input y. The output consists of $\mathcal{A}$'s view.

By construction, H_0 and the output of $\mathcal{A}$ in the real execution are identically distributed.

Hybrid H_1: Identical to H_0 except that this hybrid also performs extraction of $\mathcal{A}$'s implicit input x^* from Π_{WIPOK}; in addition, it also extracts the "simulation trapdoor" σ from the first three rounds $(\mathsf{fs}_1, \mathsf{fs}_2, \mathsf{fs}_3)$ of Π_{FS}.[12] More specifically, H_1 proceeds as follows:

1. It completes the first three broadcast rounds exactly as in H_0, and waits until $\mathcal{A}$ either aborts or successfully completes the third round.
2. At this point, H_1 proceeds to extract the witness corresponding to each proof-of-knowledge completed in the first three rounds.

   Specifically, H_1 defines a cheating prover P^* which acts identically to H_0, simulating all messages for $\mathcal{A}$, except those corresponding to (each execution of) Π_{WIPOK} which are forwarded outside. It then applies the extractor of Π_{WIPOK} to obtain the "witnesses" which consists of the following: values $\{(r_i, \omega_i)\}_{i\in[\kappa]}$ which is the witness for $\mathsf{st}_1 \wedge \mathsf{st}_3$, and a value (σ, ω_σ) which is the simulation trapdoor for Π_{FS}.

   If extraction fails, H_1 outputs fail. Otherwise, let $b_i \in \{0,1\}$ be such that $\mathsf{nm}_1^{i,b_i} = \mathsf{nmcom}_1(\mathsf{id}_1, r_i; \omega_i)$. H_1 defines a string $x^* = (x_1^*, \ldots, x_\kappa^*)$ as follows:

$$\text{If } z_{i,b_i} = r_i + r'_{i,b_i} \text{ then } x_i^* = 1 - b_i; \text{ otherwise } x_i^* = b_i$$

3. H_1 completes the final round and prepares the output exactly as H_0.

Claim 1. $\text{Hybrid}_0 \approx_{\text{s}} \text{Hybrid}_1$.

Proof sketch: This is a (completely) standard proof which we sketch here. Let p be the probability with which $\mathcal{A}$ completes Π_{WIPOK} in the third round, and let trans be the transcript. The extractor for Π_{WIPOK} takes expected time $\mathsf{poly}(\kappa)/p$ and succeeds with probability $1 - \mu(\kappa)$. It follows that the expected running time of H_1 is $\mathsf{poly}(\kappa) + p \cdot \frac{\mathsf{poly}(\kappa)}{p} = \mathsf{poly}(\kappa)$, and its output is statistically close to that of H_0.[13] $\diamond$

[12]Recall that $(\mathsf{fs}_1, \mathsf{fs}_2, \mathsf{fs}_3)$ contains two non-malleable commitments (to values σ_1, σ_2) along with proof-of-knowledge of one of the committed values (see Appendix) using Π_{WIPOK}; this execution of Π_{WIPOK} runs in parallel and therefore, it is possible to extract from it at the same time as x^*.

[13]See "witness extended emulation" in [Lin01] for full exposition.

Hybrid H_2: Identical to H_1 except that this hybrid uses the simulation trapdoor (σ, ω_σ) as the witness to compute fs_4 in the last round. (Recall that fs_4 is the last round of an execution of Π_{WIPOK}.)

It is easy to see that H_2 and H_1 are computationally indistinguishable due the WI property of Π_{WIPOK}.

Hybrid H_3: In this hybrid, we change the witness used in Π_{WIPOK} on behalf of the honest party. In particular, protocol Π_{WIPOK} is emulated using a fake witness.

Claim 2. $\text{Hybrid}_2 \approx_{\text{c}} \text{Hybrid}_3$.

Proof. We rely on the non-malleability with respect to 3-round protocols of the nmcom and the witness indistinguishability property of Π_{WIPOK} to prove this claim. Assume, for contradiction, that there exists a distinguisher D that distinguishes H_2 and H_3. We show how this violates the robustness of nmcom. Consider the machine $B(b^\kappa)$, which upon receiving a statement x, and two witnesses for x, w_0 and w_1, provides a Π_{WIPOK} proof of the statement x using witness w_b. Since Π_{WIPOK} is a 3-round protocol where the second message is sent from the verifier, (x, w_0, w_1) can be sent with the second message from the Π_{WIPOK} verifier. It follows directly from the WI property of the Π_{WIPOK} proof that no (non-uniform) PPT adversary can distinguish interactions with $B(0^\kappa)$ and $B(1^\kappa)$.

Next, we construct an adversary $\tilde{M}$ such that the view and values that $\tilde{M}$ committed to while interacting with $B(0^\kappa)$ and $B(1^\kappa)$ can be distinguished by a distinguisher $\tilde{D}$ that appropriately incorporates D.

Description of $\tilde{M}$: Machine $\tilde{M}$ proceeds in the following two phases.

- **Main execution phase:** $\tilde{M}$ incorporates a man-in-the-middle adversary $\mathcal{A}$ internally, and proceeds exactly as H_2 by sampling all messages internally except for the messages of Π_{WIPOK}. In particular, $\tilde{M}$ honestly emulates the left committer and right receivers for $\mathcal{A}$ with the following exceptions:
  1. To emulate the proof of the left interaction it externally sends B the statement x and the witnesses w_0, w_1 where w_1 is the "fake witness". It next forwards the Π_{WIPOK} proof from B to $\mathcal{A}$.
  2. In the right interactions, it externally forwards messages from $\mathcal{A}$ to an honest receiver of nmcom.
- **Output phase:** After the first three rounds of the protocol are finished, $\tilde{M}$ halts by outputting its view, denoted by $\mathcal{V}_{\tilde{M}}$. In particular, $\tilde{M}$ does not continue further like H_2, it does not extract any values, and does not complete the fourth round. (In fact, $\tilde{M}$ cannot complete the fourth round, since it does not have the witness.) Formally, for $i \in [\kappa], b \in \{0, 1\}$, $\tilde{M}$ outputs $\mathcal{V}_{\tilde{M}}$ that includes the view of $\tilde{M}$ from the *main execution phase* and the values $\{\tilde{u}_{i,b}\}$ committed by $\tilde{M}$ in the κ sessions on the right with tags $\{\widetilde{\mathsf{id}}_i\}$ while receiving parallel commitments to $\{u_{i,b}\}$ on left with tags $\{\mathsf{id}_i\}$.

Description of $\tilde{D}$: Distinguisher $\tilde{D}$ proceeds in the following two phases.

- **Main execution phase:** $\tilde{D}$ incorporates both $\tilde{M}$ and D internally, it receives as input the view $\mathcal{V}_{\tilde{M}}$ and proceeds as follows:
  1. $\tilde{D}$ parses $\mathcal{V}_{\tilde{M}}$ to obtain the view $\mathcal{V}_{\mathcal{A}}$ of $\mathcal{A}$, the committed values $\{\widetilde{u}_{i,b}\}$ and consequently a string σ corresponding to the "trapdoor witness." [14]
  2. $\tilde{D}$ starts $\tilde{M}$ and feeds him the view $\mathcal{V}_{\mathcal{A}}$ together with the committed values and continues the execution just like H_2. It, however, does not rewind $\mathcal{A}$ (internal to $\tilde{M}$), instead it uses σ and values in $\mathcal{V}_{\mathcal{A}}$ to complete the last round of the protocol.
- **Output phase:** When $\mathcal{A}$ halts, $\tilde{D}$ feeds the view $\mathcal{V}_{\mathcal{A}}$ to D and outputs whatever D outputs.

Since $\tilde{M}$, in interaction with $B(0^\kappa)$ (resp., $B(1^\kappa)$), perfectly emulates the view of $\mathcal{A}$ in the hybrid experiment H_2 (resp., H_3), the extracted view $\mathcal{V}_{\mathcal{A}}$ and the committed values are identically distributed to the view of H_2 (resp., H_3). It follows that $\tilde{D}$ distinguishes the view and the values committed to by $\tilde{M}$ using nmcom, which contradicts the robustness property of nmcom. The claim follows. $\diamond$

Hybrid H_4: In this hybrid, we get rid of P_2's input y that is implicitly present in values $\{\widetilde{z}_{i,b}\}$ and $\{\widetilde{r}_{i,b}\}$ in nmcom (but keep it everywhere else for the time being). Formally, H_4 is identical to H_3 except that in round 3 it sets $\widetilde{z}_{i,b} = \widetilde{r}_{i,b} + \widetilde{r}'_{i,b}$ for all (i, b).

Claim 3. $\mathrm{Hybrid}_3 \approx_{\mathrm{c}} \mathrm{Hybrid}_4$.

Proof. We rely on the non-malleability of nmcom to prove this claim. Assume, for contradiction, that there exists a distinguisher D that distinguishes H_3 and H_4. We show how this violates the non-malleability property of nmcom.

The high level idea is as follows: first we define two string sequences $\{u^0_{i,b}\}$ and $\{u^1_{i,b}\}$ and a man-in-the-middle M which incorporates $\mathcal{A}$ and receives non-malleable commitments to one of these sequences in parallel. Then we define a distinguisher D_{nm} which incorporates both M and D, takes as input the value committed by M and its view, and can distinguish which sequence was committed to M. This violates non-malleability of nmcom.

Formally, we define a man-in-middle adversary M who receives 2κ nmcom commitments on the left and makes 2κ nmcom commitments on the right.
Description of M: Machine M proceeds in the following two phases.

- **Main execution phase:** M incorporates $\mathcal{A}$ internally, and proceeds exactly as H_3 by sampling all messages internally except for the messages of nmcom corresponding to P_2. These messages are received from an outside committer as follows.
  1. M samples uniformly random values $\{\widetilde{z}_{i,b}\}$ and $\{\widetilde{r}'_{i,b}\}$ and defines $\{u^0_{i,b}\}$ and $\{u^1_{i,b}\}$ as:
$$u^0_{i,\overline{y_i}} = \widetilde{z}_{i,\overline{y_i}} + \widetilde{r}'_{i,\overline{y_i}}, \quad u^0_{i,y_i} \leftarrow \{0,1\}^\kappa, \quad u^1_{i,b} = \widetilde{z}_{i,b} + \widetilde{r}'_{i,b} \quad \forall (i,b)$$

[14] Note that, by construction, such a value is guaranteed in both sequences and w.l.o.g. can be the value in the first nmcom.

2. M forwards $\{u^0_{i,b}\}$ and $\{u^1_{i,b}\}$ to the outside committer who commits to one of these sequences in parallel. M forwards these messages to $\mathcal{A}$, and forwards the message given by $\mathcal{A}$ corresponding to nmcom to the outside receiver.

- **Output phase:** After the first three rounds of the protocol are finished, $\tilde{M}$ halts by outputting its view, denoted by $\mathcal{V}_M$. In particular, M does not continue further like H_3, it does not extract any values, and does not complete the fourth round. (Note that M cannot complete the fourth round, since it does not have the witness.) Formally, for $i \in [\kappa], b \in \{0,1\}$, M outputs $\mathcal{V}_M$ that includes the view of M from the *main execution phase* and the values $\{\widetilde{u}_{i,b}\}$ committed by M in the κ sessions on the right with tags $\{\widetilde{\mathsf{id}}_i\}$ while receiving parallel commitments to $\{u_{i,b}\}$ on left with tags $\{\mathsf{id}_i\}$.

Let $\{\widetilde{u}^0_{i,b}\}$ (resp., $\{\widetilde{u}^1_{i,b}\}$) be the sequence of values committed by M with $\{\widetilde{\mathsf{id}}_i\}$ when it receives a commitment to $\{u^0_{i,b}\}$ (resp., $\{u^1_{i,b}\}$) with $\{\mathsf{id}_i\}$.

Description of D_{nm}: Distinguisher D_{nm} proceeds in the following two phases.

- **Main execution phase:** D_{nm} incorporates both M and D internally, it receives as input the view $\mathcal{V}_M$ and proceeds as follows:
  1. D_{nm} parses $\mathcal{V}_M$ to obtain the view of $\mathcal{V}_{\mathcal{A}}$, the committed values $\{\widetilde{u}_{i,b}\}$ and consequently a string σ corresponding to the "trapdoor witness."
  2. D_{nm} starts M and feeds him the view $\mathcal{V}_{\mathcal{A}}$ together with the committed values and continues the execution just like H_3. It, however, does not rewind $\mathcal{A}$ (internal to M), instead it uses σ and values in $\mathcal{V}_{\mathcal{A}}$ to complete the last round of the protocol.
- **Output phase:** When $\mathcal{A}$ halts, D_{nm} feeds the view $\mathcal{V}_{\mathcal{A}}$ to D and outputs whatever D outputs.

It is straightforward to verify that if M receives commitments corresponding to $\{u^0_{i,b}\}$ (resp., $\{u^1_{i,b}\}$) then the output of D_{nm} is identical to that of H_3 (resp., H_4). The claim follows. $\diamond$

Hybrid H$_5$: Identical to H_4 except that H_5 changes the "inputs of the oblivious transfer" from $(Z_{i,0}, Z_{i,1})$ to $(Z_{i,x^*_i}, Z_{i,x^*_i})$. Formally, in the last round, H_5 sets $W_{i,b} = Z_{i,x^*_i} + \mathsf{H}((z'_{i,b}))$ for every (i,b), but does everything else as H_4.

H_4 and H_5 are computationally indistinguishable due to the (indistinguishable) security of oblivious transfer w.r.t. a malicious receiver. This part is identical to the proof in [KO04], and relies on the fact that one of the two strings for oblivious transfer are obtained by "coin tossing;" and therefore its inverse is hidden, which implies that the hardcore bits look pseudorandom.

Hybrid H$_6$: Identical to H_5 except that now we simulate the garbled circuit and its labels for values x^* and $F_1(x^*, y)$. Formally, H_6 starts by proceeding exactly as H_5 up to round 3 except that instead of committing to correct garbled circuit and labels in round 2, it simply commits to random values. After completing round 3, H_6 extracts x^* exactly as in H_5.

If extraction succeeds, it sends x^* to the trusted party, receives back $v_1 = F_1(x^*, y)$, and computes $(\{Z_{i,b}\}, \mathsf{GC}_*) \leftarrow \mathsf{SimGC}(1^\kappa, F_1, x^*, v_1)$. It uses labels $\{Z_{i,x_i^*}\}$ to define the values $\{W_{i,b}\}$ as in H_5, and equivocates C_{gc} to obtain openings corresponding to the simulated circuit GC^*. It then computes fs_4 as before (by using the trapdoor witness (σ, ω_σ)), and constructs $m_4 := (\{W_{i,b}\}, \mathsf{fs}_4, \mathsf{GC}^*, \zeta)$. It feeds m_4 to $\mathcal{A}$ and finally outputs $\mathcal{A}$'s view and halts.

Claim 4. $\text{Hybrid}_5 \approx_c \text{Hybrid}_6$.

Proof. We claim that H_5 and H_6 are computationally indistinguishable. First observe that the joint distribution of values $(\{\mathsf{C}_{\mathsf{lab}}^{i,b}\}, \mathsf{C}_{\mathsf{gc}})$ and GC_y (along with real openings) in H_5 is indistinguishable from the joint distribution of the values $(\{\mathsf{C}_{\mathsf{lab}}^{i,b}\}, \mathsf{C}_{\mathsf{gc}})$ and GC^* (along with equivocal openings) in H_6. The two hybrids are identical except for sampling of these values, and can be simulated perfectly given these values from outside. The claim follows.[15] $\diamond$

Observe that H_6 is now independent of the input y. Our simulator $\mathcal{S}$ is H_6. This completes the proof. ■

Remark 4.2. Note that robustness is sufficient but it is not necessary since what is actually required is a non-malleable commitment robust against a specific class of protocols, i.e., 3-round WIPOK protocols. The commitment scheme of [PPV08] is proven to be robust only against a 3-round WIPOK based on adaptive OWFs. In particular, the authors of [PPV08] prove the following lemma.

Lemma 4.4.1 ([PPV08])**.** There exists a 3-round WIPOK protocol Π_{WIPOK} for NP-statements and a 2-round non-malleable commitment scheme that is robust w.r.t. to Π_{WIPOK} assuming the existence of adaptive OWFs.

[15] Let us note that changing the commitment in second round (from correct garbled labels/circuit to random strings) is performed from the beginning—i.e., in the “main thread” of simulation—therefore the running time stays expected polynomial time as in claim 4.

Chapter 5

Secure Computation in the Tamper-Proof Hardware Model

In this chapter we propose secure computation protocols in tamper-proof hardware model under minimal complexities. More specifically, we present our results in [HPV16]. See Section 2.1.3.1 for a detailed overview of our contributions.

Overview We put forth a new formulation of tamper-proof hardware in the Global Universal Composable (GUC) framework [CDPW07]. Almost all of the previous works rely on the formulation by Katz [Kat07] and this formulation does not fully capture tokens in a concurrent setting. We address these shortcomings by relying on the GUC framework where we make the following contributions:

- We construct secure 2PC protocols for general functionalities with optimal round complexity and computational assumptions using stateless tokens. More precisely, we show how to realize arbitrary functionalities with GUC security in two rounds under the minimal assumption of OWFs. Moreover, our construction relies on the underlying function in a black-box way. As a corollary, we obtain feasibility of MPC with GUC-security under the minimal assumption of OWFs.

- We then construct a 3-round MPC protocol to securely realize arbitrary functionalities with GUC-security starting from any semi-honest secure MPC protocol. For this construction, we require the so-called one-many commit-and-prove primitive introduced in the original work of [CLOS02] that is round-efficient and black-box in the underlying commitment. Using specially designed "input-delayed" protocols we realize this primitive (with a 3-round protocol in our framework) using stateless tokens and OWFs (where the underlying OWF is used in a black-box way).

Due to space constraints we refer to Section 2.1.3.3 for an overview of our results and techniques in the adaptive setting. More details can be found in [HPV17].

5.1 Techniques

Our starting point for our round optimal secure two-party computation is the following technique from [GIS+10] for an extractable commitment scheme.

Roughly speaking, in order to extract the receiver's input, the sender chooses a function F from a pseudorandom function family that maps $\{0,1\}^m$ to $\{0,1\}^n$ bits where $m >> n$, and incorporates it into a token that it sends to the receiver. Next, the receiver commits to its input b by first sampling a random string $u \in \{0,1\}^m$ and querying the PRF token on u to receive the value v. It sends as its commitment the string $\mathsf{com}_b = (\mathsf{Ext}(u;r) + b, r, v)$

where $\mathsf{Ext}(\cdot,\cdot)$ is a strong randomness extractor. Now, since the PRF is highly compressing, it holds with high probability that conditioned on v, u has very high min-entropy and therefore $\mathsf{Ext}(u;r)+b,r$ statistically hides b. Furthermore, it allows for extraction as the simulator can observe the queries made by the sender to the token and observe that queries that yields v to retrieve u. This commitment scheme is based on one-way functions but is only extractable. To obtain a full-fledged UC-commitment from an extractable commitment we can rely on standard techniques (See [PW09, HV15] for a few examples). Instead, in order to obtain round-optimal constructions for secure two-party computation, we extend this protocol directly to realize the UC oblivious transfer functionality. A first incorrect approach is the following protocol. The parties exchange two sets of PRF tokens. Next, the receiver commits to its bit com_b using the approach described above, followed by the sender committing to its input $(\mathsf{com}_{s_0},\mathsf{com}_{s_1})$ along with an OT token that implements the one-out-of-two string OT functionality. More specifically, it stores two strings s_0 and s_1, and given a single bit b outputs s_b. Specifically, the code of that token behaves as follows:

- On input b^*,u^*, the token outputs $(s_b,\mathsf{decom}_{s_b})$ only if $\mathsf{com}_b=(\mathsf{Ext}(u^*;r)+b^*,r,v)$ and $\mathsf{PRF}(u^*)=v$. Otherwise, the token aborts.

The receiver then runs the token to obtain s_b and verifies if decom_{s_b} correctly decommits com_{s_b} to s_b. This simple idea is vulnerable to an input-dependent abort attack, where the token aborts depending on the value b^*. The work of [GIS+10] provides a combiner to handle this particular attack which we demonstrate is problematic. We describe the attack in Section 5.6. We instead will rely on a combiner from the recent work of Ostrovsky, Richelson and Scafuro [ORS15] to obtain a two-round GUC-OT protocol. A further overview of our techniques can be found in each section.

GUC-secure multi-party computation protocols. In order to demonstrate feasibility, we simply rely on the work of [IPS08] who show how to achieve GUC-secure MPC protocols in the OT-hybrid. By instantiating the OT with our GUC-OT protocol, we obtain MPC protocols in the tamper proof model assuming only one-way functions. While this protocol minimizes the complexity assumptions, the round complexity would be high. In this work, we show how to construct a 3-round MPC protocol. Our starting point is to take any semi-honest MPC protocol in the stand-alone model and compile it into a malicious one using tokens following the paradigm in the original work of Canetti et al. [CLOS02] and subsequent works [Pas03, Lin03b]. Roughly, the approach is to define a commit-and-prove GUC-functionality $\mathcal{F}_{\mathrm{CP}}$ and compile the semi-honest protocol using this functionality following a GMW-style compilation.

We will follow an analogous approach where we directly construct a full-fledged $\mathcal{F}_{\mathrm{CP}}^{1:\mathrm{M}}$-functionality that allows a single prover to commit to a string and then prove multiple statements on the commitment simultaneously to several parties. In the token model, realizing this primitive turns out to be non-trivial. This is because we need the commitment in this protocol to be straight-line extractable and the proof to be about the value committed. Recall that, the extractable commitment is based on a PRF token supplied by the receiver of the commitment (and the verifier in the zero-knowledge proof). The prover cannot attest the validity of its commitment (via an NP-statement) since it does not know the code (i.e. key) of the PRF. Therefore, any commit and prove scheme in the token model necessarily must rely on a zero-knowledge proof that is black-box in the underlying commitment scheme. In fact, in the seminal work of Ishai et al. [IKOS09] they showed how to construct such protocols that have been extensively used in several works where the goal is to obtain constructions that are black-box in the underlying primitives. Following this approach and solving its difficulties that appear in the tamper-proof hardwire model, we can compile a T-round semi-honest secure MPC protocol to a $O(T)$-round protocol. Next, to reduce the rounds of the computation we consider the approach of Garg et al. [GGHR14] who show how to compress the round complexity of any MPC protocol to a two-round GUC-secure MPC protocol in the CRS model using obfuscation primitives.

In more detail, in the first round of the protocol in [GGHR14], every party commits to its input along with its randomness. The key idea is the following compiler used in the second round: it takes any (interactive) underlying MPC protocol, and has each party obfuscate their "next-message" function in that protocol, providing one obfuscation for each round. To ensure correctness, zero-knowledge proofs are used to validate the actions of each party w.r.t the commitments made in the first step. Such a mechanism is also referred to as a commit-and-prove strategy. This enables each party to independently evaluate the obfuscation one by one, generating messages of the underlying MPC protocol and finally obtain the output. The observation here is that party P_i's next-message circuit for round j in the underlying MPC protocol depends on its private input x_i and randomness r_i (which are hard-coded in the obfuscation) and on input the transcript of the communication in the first $j-1$ rounds outputs its message for the next round.

To incorporate this approach in the token model, we can simply replace the obfuscation primitives with tokens. Next, to employ zero-knowledge proofs via a black-box construction, we require a zero-knowledge protocol that allows commitment of a witness via tokens at the beginning of the protocol and then in a later step prove a statement about this witness where the commitment scheme is used in a "black-box" way. A first idea here would be to compile using the zero-knowledge protocol of [IKOS09] that facilitate such a commit-and-prove paradigm. However, as we explain later this would cost us in round-complexity. Instead we will rely on so-called input-delayed proofs [LS90] that have recently received much attention [CPS$^+$16a, CPS$^+$16b, HV16]. In particular, we will rely on the recent work of [HV16] who shows how to construct the so-called "input-delay" commit-and-prove protocols which allow a prover to commit a string in an initial commit phase and then prove a statement regarding this string at a later stage where the input statement is determined later. However, their construction only allows for proving one statement regarding the commitment. One of our technical contributions is to extend this idea to allow multiple theorems and further extend it so that a single prover can prove several theorems to multiple parties simultaneously. This protocol will be 4-round and we show how to use this protocol in conjunction with the Garg et al.'s round collapsing technique.

5.2 Modeling Tamper-Proof Hardware in the GUC Framework

In this section we describe our model and give our rationale for our approach. We provide a brief discussion on the Universal Composability (UC) framework [Can01], UC with *joint state*

[CR03] (JUC) and Generalized UC [CDPW07] (GUC). For more details, we refer the reader to the original works and the discussion in [CJS14].

Basic UC. Introduced by Canetti in [Can01], the Universal Composability (UC) framework provides a framework to analyse security of protocols in complex network environments in a modular way. One of the fundamental contributions of this work was to give a definition that will allow to design protocols and demonstrate security by "locally" analyzing a protocol but guaranteeing security in a *concurrent* setting where security of the protocol needs to be intact even when it is run concurrently with many instances of arbitrary protocols. Slightly more technically, in the UC-framework, to demonstrate that a protocol Π securely realizes an ideal functionality $\mathcal{F}$, we need to show that for any adversary $\mathcal{A}$ in the real world interacting with protocol Π in the presence of arbitrary environments $\mathcal{Z}$, there exists an ideal adversary $\mathcal{S}$ such that for any environment $\mathcal{Z}$ the view of an interaction with $\mathcal{A}$ is indistinguishable from the view of an interaction with the ideal functionality $\mathcal{F}$ and $\mathcal{S}$.

Unfortunately, soon after its inception, a series of impossibility results [CF01, CKL06, Lin03b] demonstrated that most non-trivial functionalities cannot be realized in the UC-framework. Most feasibility results in the UC-framework relied on some sort of trusted setup such as the common reference string (CRS) model [CF01], tamper-proof model [Kat07] or relaxed security requirements such as super-polynomial simulation [Pas03, PS04, BS05a]. When modeling trusted setup such as the CRS model, an extension of the UC-framework considers the $\mathcal{G}$-hybrid model where "all" real-world parties are given access to an ideal setup functionality $\mathcal{G}$. In order for the basic composition theorem to hold in such a $\mathcal{G}$-hybrid model, two restrictions have to be made. First, the environment $\mathcal{Z}$ cannot access the ideal setup functionality *directly*; it can only do so indirectly via the adversary. In some sense, the setup $\mathcal{G}$ is treated as "local" to a protocol instance. Second, two protocol instances of the same or different protocol cannot share "state" for the UC-composition theorem to hold. Therefore, a setup model such as the CRS in the UC-framework necessitates that each protocol uses its own local setup. In other words, an independently sampled reference string for every protocol instance. An alternative approach that was pursued in a later work was to realize a multi-version of a functionality and proved security of the multi-version using a single setup. For example, the original feasibility result of Canetti, Lindell, Ostrovsky and Sahai [CLOS02] realized the $\mathcal{F}_{\text{MCOM}}$-functionality which is the multi-version of the basic commitment functionality$\mathcal{F}_{\text{COM}}$ in the CRS model.

JUC. Towards accommodating a global setup such as the CRS for multiple protocol instances, Canetti and Rabin [CR03] introduced the Universal Composition with Joint State (JUC) framework. Suppose we want to analyze several instances of protocol Π with an instance $\mathcal{G}$ as common setup, then at the least, each instance of the protocol must share some state information regarding $\mathcal{G}$ (e.g., the reference string in the CRS model). The JUC-framework precisely accommodates such a scenario, where a new composition theorem is proven, that allows for composition of protocols that share some state. However, the JUC-model for the CRS setup would only allow the CRS to be accessible to a pre-determined set of protocols and in particular still does not allow the environment to directly access the CRS.

GUC. For most feasibility results in the (plain) CRS model both in the UC and JUC framework, the simulator $\mathcal{S}$ in the ideal world needed the ability to "program" the CRS. In particular, it is infeasible to allow the environment to access the setup reference string. As a consequence, we can prove security only if the reference string is privately transmitted to the protocols that we demand security of and cannot be made *publicly* accessible. The work of Canetti, Pass, Dodis and Walfish [CDPW07] introduced the Generalized UC-framework to overcome this shortcoming in order to model the CRS as a global setup that is publicly available. More formally, in the GUC-framework, a global setup $\mathcal{G}$ is accessible by any protocol running in the system and in particular allows direct access by the environment. This, in effect, renders all previous protocols constructed in the CRS model not secure in the GUC framework as the simulator loses the programmability of the CRS. In fact, it was shown in [CDPW07] that the CRS setup is insufficient to securely realize the ideal commitment functionality in the GUC-framework. More generally, they show that any setup that simply provides only "public" information is not sufficient to realize GUC-security for most non-trivial functionalities. They further demonstrated a feasibility in the Augmented CRS model, where the CRS contains signature keys, one for each party and a secret signing key that is not revealed to the parties, except if it is corrupt, in which case the secret signing key for that party is revealed.

As mentioned before, the popular framework to capture the tamper-proof hardware is the one due to [Kat07] who defined the $\mathcal{F}_{\text{WRAP}}$-functionality in the UC-framework. In general, in the token model, the two basic advantages that the simulator has over the adversary is "observability" and "programmability". Observability refers to the ability of the simulator to monitor all queries made by an adversary to the token and programmability refers to the ability to program responses to the queries in an online manner. In the context of tokens, both these assumptions are realistic as tamper-proof tokens do provide both these abilities in a real-world. However, when modeling tamper proof hardware tokens in the UC-setting, both these properties can raise issues as we discuss next.

Apriori, it is not clear why one should model the tamper proof hardware as a global functionality. In fact, the tokens are local to the parties and it makes the case for it *not* to be globally accessible. Let us begin with the formulation by Katz [Kat07] who introduced the $\mathcal{F}_{\text{WRAP}}$-functionality (see Figure 5.1 for the stateless variant). In the real world the creator or sender of a token specifies the code to be incorporated in a token by sending the description of a Turing machine M to the ideal functionality. The ideal functionality then emulates the code of M to the receiver of the token, only allowing black-box access to the input and output tapes of M. In the case of stateful tokens, M is modeled as an interactive Turing machine while for stateless tokens, standard Turing machines would suffice. Slightly more technically, in the UC-model, parties are assigned unique identifiers PID and sessions are assigned identifiers sid. In the tamper proof model, to distinguish tokens, the functionality accepts an identifier mid when a token is created. More formally, when one party PID_i creates a token with program M with token identifier mid and sends it to another party PID_j in session sid, then the $\mathcal{F}_{\text{WRAP}}$ records the tuple $(\mathsf{PID}_i, \mathsf{PID}_j, \mathsf{mid}, M)$. Then whenever a party with identifier PID_j sends a query $(\mathsf{Run}, \mathsf{sid}, \mathsf{PID}_i, \mathsf{mid}, x)$ to the $\mathcal{F}_{\text{WRAP}}$-functionality, it first checks whether there is a tuple of the form $(\cdot, \mathsf{PID}_j, \mathsf{mid}, \cdot)$ and then runs the machine M in this tuple if one exists.

In the UC-setting (or JUC), to achieve any composability guarantees, we need to realize

Functionality $\mathcal{F}_{\text{WRAP}}^{\text{Stateless}}$

Functionality $\mathcal{F}_{\text{WRAP}}^{\text{Stateless}}$ is parameterized by a polynomial $p(\cdot)$ and an implicit security parameter κ.

Create. Upon receiving $(\mathsf{Create}, \mathsf{sid}, \mathsf{PID}_i, \mathsf{PID}_j, \mathsf{mid}, M)$ from P_1, where M is a Turing machine, do:

1. Send $(\mathsf{Create}, \mathsf{PID}_i, \mathsf{PID}_j, \mathsf{mid})$ to P_2.
2. Store $(\mathsf{PID}_i, \mathsf{PID}_j, \mathsf{mid}, M)$.

Execute. Upon receiving $(\mathsf{Run}, \mathsf{sid}, \mathsf{PID}_i, \mathsf{mid}, x)$ from P_2, find the unique stored tuple $(\mathsf{PID}_i, \mathsf{PID}_j, \mathsf{mid}, M)$. If no such tuple exists, do nothing. Run $M(x)$ for at most $p(\kappa)$ steps, and let out be the response ($\mathsf{out} = \perp$ if M does not halt in $p(k)$ steps). Send $(\mathsf{PID}_i, \mathsf{PID}_j, \mathsf{mid}, \mathsf{out})$ to P_2.

Figure 5.1: The ideal functionality for stateless tokens [Kat07].

the multi-use variants of the specified functionality and then analyze the designed protocol in a concurrent man-in-the-middle setting. In such a multi-instance setting, it is reasonable to assume that an adversary that receives a token from one honest party in a left interaction can forward the token to another party in a right interaction. Unfortunately, the $\mathcal{F}_{\text{WRAP}}$-functionality does not facilitate such a transfer.

Let us modify $\mathcal{F}_{\text{WRAP}}$ to accommodate transfer of tokens by adding a special "transfer" query that allows a token in the possession of one party to be transferred to another party. Since protocols designed in most works do not explicitly prove security in a concurrent man-in-the-middle setting, such a modification renders the previous protocols designed in $\mathcal{F}_{\text{WRAP}}$ insecure. For instance, consider the commitment scheme discussed in the introduction based on PRF tokens. Such a scheme would be insecure as an adversary can simply forward the token from the receiver in a right interaction to the sender in a left interaction leading to a malleable commitment.

In order to achieve security while allowing transferability we need to modify the tokens themselves in such a way to be not useful in an execution different from where it is supposed to be used. If every honestly generated token admits only queries that are prefixed with the correct session identifier then transferring the tokens created by one honest party to another honest party will be useless as honest parties will prefix their queries with the right session and the honestly generated tokens will fail to answer on incorrect session prefixes. This is inspired by an idea in [CJS14], where they design GUC-secure protocols in the Global Random Oracle model [CJS14]. As such, introducing transferrability naturally requires protocols to address the issue of non-malleability.

While this modification allows us to model transferrability, it still requires us to analyze protocols in a concurrent man-in-the-middle setting. In order to obtain a more modular definition, where each protocol instance can be analyzed in isolation we need to allow the token to be transferred from the adversary to the environment. In essence, we require the token to be somewhat "globally" accessible and this is the approach we take.

5.2.1 The Global Tamper-Proof Model

A natural first approach would be to consider the same functionality in the GUC-framework and let the environment to access the $\mathcal{F}_{\text{WRAP}}$-functionality. This is reasonable as an environment can have access to the tokens via auxiliary parties to whom the tokens were transferred to. However, naively incorporating this idea would deny "observability" and "programmability" to the simulator as all adversaries can simply transfer away their tokens the moment they receive them and let other parties make queries on their behalf. Indeed, one can show that the impossibility result of [CKS$^+$14] extends to this formulation of the tokens (at least if the code of the token is treated in a black-box manner).[1] A second approach would be to reveal to the simulator all queries made to the token received by the adversary even if transferred out to any party. However, such a formulation would be vulnerable to the following transferring attack. If an adversary received a token from one session, it can send it as its token to an honest party in another session and now observe all queries made by the honest party to the token. Therefore such a formulation of tokens is incorrect.

Our formulation will accommodate transferrability while still guaranteeing observability to the simulator. In more detail, we will modify the definition of $\mathcal{F}_{\text{WRAP}}$ so that it will reveal to the simulator all "illegitimate" queries made to the token by any other party. This approach is analogous to the one taken by Canetti, Jain and Scafuro [CJS14] where they model the Global Random Oracle Model and are confronted by a similar issue; here queries made to a globally accessible random oracle via auxiliary parties by the environment must be made available to the simulator while protecting the queries made by the honest party. In order to define "legitimate" queries we will require that all tokens created by an honest party, by default, will accept an input of the form (sid, x) and will respond with the evaluation of the embedded program M on input x, only if $\mathsf{sid} = \overline{\mathsf{sid}}$, where $\overline{\mathsf{sid}}$ corresponds to the session where the token is supposed to be used, i.e. the session where the honest party created the token. Furthermore, whenever an honest party in session $\overline{\mathsf{sid}}$ queries a token it received on input x, it will prefix the query with the correct session identifier, namely issue the query $(\overline{\mathsf{sid}}, x)$. An *illegitimate query* is one where the sid prefix in a query differs from the session identifier from which the party is querying from. Every illegitimate query will be recorded by our functionality and will be disclosed to the party whose session identifier is actually sid.

More formally, the $\mathcal{F}_{\text{gWRAP}}$-functionality is parameterized by a polynomial $p(\cdot)$ which is the time bound that the functionality will exercise whenever it runs any program. The functionality admits the following queries:

Creation Query: This query allows one party P_1 to create and send a token to another party P_2 by sending the query $(\mathsf{Create}, \mathsf{sid}, P_1, P_2, \mathsf{mid}, M)$ where M is the description of the

[1] Informally, the only advantage that remains for the simulator is to see the code of the tokens created by the adversary. This essentially reduces to the case where tokens are sent only in one direction and is impossible due to a result of [CKS$^+$14] when the code is treated as a black-box.

machine to be embedded in the token, mid is a unique identifier for the token and sid is the session identifier. The functionality records $(P_2, \mathsf{sid}, \mathsf{mid}, M)$.[2]

Transfer Query: We explicitly provide the ability for parties to transfer tokens to other parties that were not created by them (eg, received from another session). Such a query will only be used by the adversary in our protocols as honest parties will always create their own tokens. When a transfer query of the form $(\mathsf{transfer}, \mathsf{sid}, P_1, P_2, \mathsf{mid})$ is issued, the tuple $(P_1, \overline{\mathsf{sid}}, \mathsf{mid}, M)$ is erased and a new tuple $(P_2, \mathsf{sid}, \mathsf{mid}, M)$ is created where $\overline{\mathsf{sid}}$ is the identifier of the session where it was previously used.

Execute Query: To run a token the party needs to provide an input in a particular format. All honest parties will provide the input as $x = (\mathsf{sid}, x')$ and the functionality will run M on input x and supply the answer. In order to achieve non-malleability, we will make sure in all our constructions that tokens generated by honest parties will respond to a query only if it contains the correct sid.

Retrieve Query: This is the important addition to our functionality following the approach taken by [CJS14]. $\mathcal{F}_{\text{gWRAP}}$-functionality will record all illegitimate queries made to a token. Namely for a token recorded as the tuple $(P_2, \overline{\mathsf{sid}}, \mathsf{mid}, M)$ an illegitimate query is of the form (sid, x) where $\mathsf{sid} \neq \overline{\mathsf{sid}}$ and such a query will be recorded in a set $\mathcal{Q}_{\mathsf{sid}}$ that will be made accessible to the receiving party corresponding to sid.

A formal description of the ideal functionality $\mathcal{F}_{\text{gWRAP}}$ is presented in Figure 5.2. We emphasize that our formulation of the tamper-proof model will now have the following benefits:

1. It overcomes the shortcomings of the $\mathcal{F}_{\text{WRAP}}$-functionality as defined in [Kat07] and used in subsequent works. In particular, it allows for transferring tokens from one session to another while retaining "observability".

2. Our model allows for designing protocols in the UC-framework and enjoys the composition theorem as it allows the environment to access the token either directly or via other parties.

3. Our model explicitly rules out "programmability" of tokens. We remark that it is (potentially) possible to explicitly provide a mechanism for programmability in the $\mathcal{F}_{\text{gWRAP}}$-functionality. We chose to not provide such a mechanism so as to provide stronger composability guarantees.

4. In our framework, we can analyze the security of a protocol in isolation and guarantee concurrent multi-instance security directly using the GUC-composition theorem. Moreover, it suffices to consider a "dummy" adversary that simply forwards the environment everything (including the token).

[2]We remark here that the functionality does not explicitly store the PID of the creator of the token. We made this choice since the simulator in the ideal world will create tokens for itself which will serve as a token created on behalf of an honest party.

An immediate consequence of our formulation is that it renders prior works such as [Kat07, CGS08, DKM11, DKMN15a] that rely on the programmability of the token insecure in our model. The works of [GIS+10, CKS+14] on the other hand can be modified and proven secure in the $\mathcal{F}_{\text{gWRAP}}$-hybrid as they do not require the tokens to be programmed.

Functionality $\mathcal{F}_{\text{gWRAP}}$

Parameters: Polynomial $p(\cdot)$.

Create. Upon receiving $(\mathsf{Create}, \mathsf{sid}, P_1, P_2, \mathsf{mid}, M)$ from P_1, where M is a Turing machine, do:

1. Send $(\mathsf{Receipt}, \mathsf{sid}, P_1, P_2, \mathsf{mid})$ to P_2.
2. Store $(P_2, \mathsf{sid}, \mathsf{mid}, M)$.

Execute. Upon receiving $(\mathsf{Run}, \mathsf{sid}, \mathsf{mid}, x)$ from P_2, find the unique stored tuple $(P_2, \overline{\mathsf{sid}}, \mathsf{mid}, M)$. If such a tuple does not exist, do nothing. Otherwise, interpret $x = (\overline{\mathsf{sid}}, x')$ and run $M(x)$ for at most $p(\kappa)$ steps, and let out be the response ($\mathsf{out} = \bot$ if M does not halt in $p(k)$ steps). Send $(\mathsf{sid}, P_2, \mathsf{mid}, \mathsf{out})$ to P_2.
Handling Illegal Queries: If $\mathsf{sid} \neq \overline{\mathsf{sid}}$, then add $(x', \mathsf{out}, \mathsf{mid})$ to the list $\mathcal{Q}_{\overline{\mathsf{sid}}}$ that is initialized to be empty.

Transfer. Upon receiving $(\mathsf{transfer}, \mathsf{sid}, P_1, P_2, \mathsf{mid})$ from P_1, find the unique stored tuple $(P_1, \overline{\mathsf{sid}}, \mathsf{mid}, M)$. If no such tuple exists, do nothing. Otherwise,

1. Send $(\mathsf{Receipt}, \mathsf{sid}, P_1, P_2, \mathsf{mid})$ to P_2.
2. Store $(P_2, \mathsf{sid}, \mathsf{mid}, M)$. Erase $(P_1, \overline{\mathsf{sid}}, \mathsf{mid}, M)$.

Retrieve Queries: Upon receiving a request $(\mathsf{retreive}, \mathsf{sid})$ from a party P_2, return the list $\mathcal{Q}_{\mathsf{sid}}$ of illegitimate queries.

Figure 5.2: [HPV16] The global stateless token functionality.

We now provide the formal definition of UC-security in the Global Tamper-Proof model.

Definition 5.1. (GUC security in the global tamper-proof model) Let $\mathcal{F}$ be an ideal functionality and let π be a multi-party protocol. Then protocol π GUC realizes $\mathcal{F}$ in $\mathcal{F}_{\text{gWRAP}}$-hybrid model, if for every uniform PPT hybrid-model adversary $\mathcal{A}$, there exists a uniform PPT simulator $\mathcal{S}$, such that for every non-uniform PPT environment $\mathcal{Z}$, the following two ensembles are computationally indistinguishable,

$$\left\{\mathsf{view}^{\mathcal{F}_{\text{gWRAP}}}_{\pi,\mathcal{A},\mathcal{Z}}(\kappa)\right\}_{\kappa\in\mathbb{N}} \overset{c}{\approx} \left\{\mathsf{view}^{\mathcal{F}_{\text{gWRAP}}}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(\kappa)\right\}_{\kappa\in\mathbb{N}}.$$

5.3 Preliminaries in the Tamper-Proof Hardware Model

Basic notations. We denote the security parameter by κ. We say that a function $\mu : \mathbb{N} \to \mathbb{N}$ is *negligible* if for every positive polynomial $p(\cdot)$ and all sufficiently large κ's it holds that $\mu(\kappa) <$

$\frac{1}{p(\kappa)}$. We use the abbreviation PPT to denote probabilistic polynomial-time. We specify next the definition of computationally indistinguishable and statistical distance.

Definition 5.2. Let $X = \{X(a,\kappa)\}_{a\in\{0,1\}^*,\kappa\in\mathbb{N}}$ and $Y = \{Y(a,\kappa)\}_{a\in\{0,1\}^*,\kappa\in\mathbb{N}}$ be two distribution ensembles. We say that X and Y are *computationally indistinguishable*, denoted $X \overset{c}{\approx} Y$, if for every PPT machine D, every $a \in \{0,1\}^*$, every positive polynomial $p(\cdot)$ and all sufficiently large κ's,

$$\left|\Pr\left[D(X(a,\kappa),1^\kappa)=1\right]-\Pr\left[D(Y(a,\kappa),1^\kappa)=1\right]\right| < \frac{1}{p(\kappa)}.$$

Definition 5.3. Let X_κ and Y_κ be random variables accepting values taken from a finite domain $\Omega \subseteq \{0,1\}^\kappa$. The *statistical distance* between X_κ and Y_κ is

$$SD(X_\kappa,Y_\kappa) = \frac{1}{2}\sum_{\omega\in\Omega}\left|\Pr[X_\kappa=\omega]-\Pr[Y_\kappa=\omega]\right|.$$

We say that X_κ and Y_κ are *ε-close* if their statistical distance is at most $SD(X_\kappa,Y_\kappa) \le \varepsilon(\kappa)$. We say that X_κ and Y_κ are *statistically close*, denoted $X_\kappa \approx_s Y_\kappa$, if $\varepsilon(\kappa)$ is negligible in κ.

5.3.1 Pseudorandom Functions

Informally speaking, a pseudorandom function (PRF) is an efficiently computable function that looks like a truly random function to any PPT observer. Namely,

Definition 5.4 (Pseudorandom function ensemble)**.** Let $F = \{\mathsf{PRF}_\kappa\}_{\kappa\in\mathbb{N}}$ where for every κ, $\mathsf{PRF}_\kappa : \{0,1\}^\kappa \times \{0,1\}^m \to \{0,1\}^l$ is an efficiently computable ensemble of keyed functions. We say that $F = \{\mathsf{PRF}_\kappa\}_{\kappa\in\mathbb{N}}$ is *a pseudorandom function ensemble* if for every PPT machine D, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all sufficiently large κ's,

$$|\Pr[D^{\mathsf{PRF}_\kappa(k,\cdot)}(1^\kappa)] = 1 - \Pr[D^{f_\kappa}(1^\kappa)=1]| \le \mathsf{negl}(\kappa),$$

where k is picked uniformly from $\{0,1\}^\kappa$ and f_κ is chosen uniformly at random from the set of functions mapping m-bit strings into l-bit strings. We sometimes omit κ from our notation when it is clear from the context.

5.3.2 Commitment Schemes

Commitment schemes are used to enable a party, known as the *sender* P_1, to commit itself to a value while keeping it secret from the *receiver* P_2 (this property is called *hiding*). Furthermore, in a later stage when the commitment is opened, it is guaranteed that the "opening" can yield only a single value determined in the committing phase (this property is called *binding*). In this work, we consider commitment schemes that are *statistically binding*, namely while the hiding property only holds against computationally bounded (non-uniform) adversaries, the binding property is required to hold against unbounded adversaries. Formally,

Definition 5.5 (Commitment schemes)**.** A PPT machine $\mathsf{Com} = \langle S, R\rangle$ is said to be a non-interactive commitment scheme if the following two properties hold.

Computational hiding: For every (expected) PPT machine ${P_2}^*$, it holds that the following ensembles are computationally indistinguishable.

- $\{\mathsf{view}_{\mathsf{Com}}^{{P_2}^*}(m_1, z)\}_{\kappa \in N, m_1, m_2 \in \{0,1\}^\kappa, z \in \{0,1\}^*}$
- $\{\mathsf{view}_{\mathsf{Com}}^{{P_2}^*}(m_2, z)\}_{\kappa \in N, m_1, m_2 \in \{0,1\}^\kappa, z \in \{0,1\}^*}$

where $\mathsf{view}_{\mathsf{Com}}^{R^*}(m, z)$ denotes the random variable describing the output of ${P_2}^*$ after receiving a commitment to m using Com.

Statistical binding: For any (computationally unbounded) malicious sender ${P_1}^*$ and auxiliary input z, it holds that the probability that there exist valid decommitments to two different values for a view v, generated with an honest receiver while interacting with ${P_1}^*(z)$ using Com, is negligible.

We refer the reader to [Gol01] for more details. We recall that non-interactive perfectly binding commitment schemes can be constructed based on one-way permutation, whereas two-round statistically binding commitment schemes can be constructed based on one-way functions [Nao91]. To set up some notations, we let $\mathsf{com}_m \leftarrow \mathsf{Com}(m; r_m)$ denote a commitment to a message m, where the sender uses uniform random coins r_m. The decommitment phase consists of the sender sending the decommitment information $\mathsf{decom}_m = (m, r_m)$ which contains the message m together with the randomness r_m. This enables the receiver to verify whether decom_m is consistent with the transcript com_m. If so, it outputs m; otherwise it outputs $\perp$. For simplicity of exposition, in the sequel, we will assume that random coins are an implicit input to the commitment functions, unless specified explicitly.

Definition 5.6 (Trapdoor commitment schemes)**.** Let $\mathsf{Com} = (P_1, P_2)$ be a statistically binding commitment scheme. We say that Com is a trapdoor commitment scheme is there exists an expected PPT oracle machine $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ such that for any PPT ${P_2}^*$ and all $m \in \{0,1\}^\kappa$, the output (τ, w) of the following experiments is computationally indistinguishable:

- an honest sender P_1 interacts with ${P_2}^*$ to commit to m, and then opens the commitment: τ is the view of ${P_2}^*$ in the commit phase, and w is the message P_1 sends in the open phase.
- the simulator $\mathcal{S}$ generates a simulated view τ for the commit phase, and then opens the commitment to m in the open phase: formally $(\tau, state) \leftarrow \mathcal{S}_1^{{P_2}^*}(1^\kappa)$, $w \leftarrow \mathcal{S}_2(state, m)$.

5.3.3 Randomness Extractors

The min-entropy of a random variable X is $H_\infty(X) = -\log(\max_x \Pr[X = x])$.

Definition 5.7 (Extractors)**.** A function $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ is a (k, ε)-strong extractor if for all pairs of random variables (X, I) such that $X \in \{0,1\}^n$ and $H_\infty(X|I) \geq k$ it holds that

$$SD((\mathsf{Ext}(X, S), S, I), (U_m, S, I)) \leq \varepsilon,$$

where S is uniform over $\{0,1\}^t$ and U_m is the uniform distribution over $\{0,1\}^m$.

The Leftover Hash Lemma shows how to explicitly construct an extractor from a family of pairwise independent functions $\mathcal{H}$. The extractor uses a random hash function $h \leftarrow \mathcal{H}$ as its seed and keeps this seed in the output of the extractor.

Theorem 5.3.1 (Leftover Hash Lemma)**.** If $\mathcal{H} = \{h : \{0,1\}^n \to \{0,1\}^m\}$ is a pairwise independent family where $m = n - 2\log\frac{1}{\varepsilon}$, then $\mathsf{Ext}(x, h) = (h, h(x))$ is a strong (n, ε)-extractor.

In this work we will consider the case where $m = 1$ and $n \geq 2\kappa + 1$ where κ is the security parameter. This yields $\varepsilon = 2^{-\frac{2\kappa+1-1}{2}} = 2^{-\kappa}$.

5.3.4 Hardcore Predicates

Definition 5.8 (Hardcore predicate)**.** Let $f : \{0,1\}^\kappa \to \{0,1\}^*$ and $\mathsf{H} : \{0,1\}^\kappa \to \{0,1\}$ be a polynomial-time computable functions. We say H is a hardcore predicate of f, if for every PPT machine A, there exists a negligible function $\mathsf{negl}(\cdot)$ such that

$$\Pr[x \leftarrow \{0,1\}^\kappa; y = f(x) : A(1^\kappa, y) = \mathsf{H}(x)] \leq \frac{1}{2} + \mathsf{negl}(\kappa).$$

An important theorem by Goldreich and Levin [GL89] states that if f is a one-way function over $\{0,1\}^\kappa$ then the one-way function f' over $\{0,1\}^{2\kappa}$, defined by $f'(x, r) = (f(x), r)$, admits the following hardcore predicate $b(x, r) = \langle x, r\rangle = \Sigma x_i r_i \bmod 2$, where x_i, r_i is the ith bit of x, r respectively. In the following, we refer to this predicate as the GL bit of f. We will use the following theorem that establishes the list-decoding property of the GL bit.

Theorem 5.3.2 ([GL89])**.** There exists a PPT oracle machine Inv that on input (κ, ε) and oracle access to a predictor PPT B, runs in time $poly(\kappa, \frac{1}{\varepsilon})$, makes at most $O(\frac{\kappa^2}{\varepsilon^2})$ queries to B and outputs a list L with $|L| \leq \frac{4\kappa}{\varepsilon^2}$ such that if

$$\Pr[r \leftarrow \{0,1\}^\kappa : B(r) = \langle x, r\rangle] \geq \frac{1}{2} + \frac{\varepsilon}{2}$$

then

$$\Pr[L \leftarrow \mathsf{Inv}^B(\kappa, \varepsilon) : x \in L] \geq \frac{1}{2}.$$

5.3.5 Secret-Sharing

A secret-sharing scheme allows distribution of a secret among a group of n players, each of whom in a *sharing phase* receive a share (or piece) of the secret. In its simplest form, the goal of secret-sharing is to allow only subsets of players of size at least $t+1$ to reconstruct the secret. More formally a $t+1$-out-of-n secret sharing scheme comes with a sharing algorithm that on input a secret s outputs n shares $s_1, \ldots, s_n$ and a reconstruction algorithm that takes as input $((s_i)_{i \in S}, S)$ where $|S| > t$ and outputs either a secret s' or $\perp$. In this work, we will use the Shamir's secret sharing scheme [Sha79] with secrets in $\mathbb{F} = GF(2^\kappa)$. We present the sharing and reconstruction algorithms below:

Sharing algorithm: For any input $s \in \mathbb{F}$, pick a random polynomial $f(\cdot)$ of degree t in the polynomial-field $\mathbb{F}[x]$ with the condition that $f(0) = s$ and output $f(1), \ldots, f(n)$.

Reconstruction algorithm: For any input $(s'_i)_{i \in S}$ where none of the s'_i are $\perp$ and $|S| > t$, compute a polynomial $g(x)$ such that $g(i) = s'_i$ for every $i \in S$. This is possible using Lagrange interpolation where g is given by

$$g(x) = \sum_{i \in S} s'_i \prod_{j \in S/\{i\}} \frac{x-j}{i-j} \ .$$

Finally the reconstruction algorithm outputs $g(0)$.

We will additionally rely on the following property of secret-sharing schemes. To this end, we view the Shamir secret-sharing scheme as a linear code generated by the following $n \times (t+1)$ Vandermonde matrix

$$A = \begin{pmatrix} 1 & 1^2 & \cdots & 1^t \\ 1 & 2^2 & \cdots & 2^t \\ \vdots & \vdots & \vdots & \vdots \\ 1 & n^2 & \cdots & n^t \end{pmatrix}$$

More formally, the shares of a secret s that are obtained via a polynomial f in the Shamir scheme, can be obtained by computing $A\mathbf{c}$ where $\mathbf{c}$ is the vector containing the coefficients of f. Next, we recall that for any linear code A, there exists a parity check matrix H of dimension $(n-t-1) \times n$ which satisfies the equation $HA = \mathbf{0}_{(n-t-1)\times(t+1)}$, i.e. the all 0's matrix. We thus define the linear operator $\phi(v) = Hv$ for any vector v. Then it holds that any set of shares $\mathbf{s}$ is valid if and only if it satisfies the equation $\phi(\mathbf{s}) = \mathbf{0}_{n-t-1}$.

The authors in [DZ13] were the first to propose an algorithm for verifying membership in (binary) codes, i.e., verifying the product of Boolean matrices in quadratic time with exponentially small error probability, while previous methods only achieved constant error.

5.4 Two-Round Oblivious Transfer in the Stand-Alone Model

5.4.1 Building Blocks

Trapdoor commitment schemes. A core building block of our protocol is a trapdoor commitment scheme TCom (cf. Definition 5.6) introduced by Pass and Wee in [PW09]. In Figure 5.3 we describe their 4-round trapdoor commitment scheme that is based on one-way permutations. In particular, the protocol comprises a 4-round challenge-response protocol where the receiver commits to its challenge in the first message (using a non-interactive perfectly binding commitment scheme). The knowledge of the receiver's challenge enables the simulator to cheat in the commit phase and equivocate the committed message into any bit (this notion of "look ahead" trapdoor commitment is borrowed from the area of zero-knowledge proofs).

More specifically, the trapdoor commitment scheme TCom, described in Figure 5.3, proceeds as follows. In order to commit to a bit m the sender commits to a matrix M of size 2×2, so that m is split into two shares which are committed within the two rows of M. Next, the receiver sends a challenge bit e where the sender must open the two commitments that lie in the eth column (and must correspond to the same share of m, thus it is easy to verify correctness). Later, in the decommit phase the sender opens the values to a row of his choice enabling the receiver to reconstruct m. Note that if the sender knows the challenge bit in advance it can commit to two distinct bits by making sure that one of the columns has different bits. In order to decrease the soundness error this protocol is repeated multiple times in parallel. In this paper we implement the internal commitment of Pass and Wee using a statistical hiding commitment scheme that is based on pseudorandom functions; see details below.

Non-interactive commitment schemes. Our construction further relies on a non-interactive perfectly binding commitment scheme that is incorporated inside the sender's token $\mathsf{TK}_{P_1}^{\mathsf{com}}$. Such commitments can be build based on the existence of one-way permutations. Importantly, it is possible to relax our assumptions to one-way functions by relying on a two-round statistically binding commitment scheme [Nao91], and allowing the token $\mathsf{TK}_{P_1}^{\mathsf{com}}$ to take an additional input that will serve as the first message of the commitment scheme. Overall, that implies that we only need to assume one-way functions. For clarity of presentation, we use a non-interactive commitment scheme that is based on one-way permutations; see Section 5.4.2.1 for more details.

5.4.2 Our Protocol

We are now ready to introduce our first protocol that securely computes the functionality $\mathcal{F}_{\mathrm{OT}} : ((s_0, s_1), b) \mapsto (\perp, s_b)$ in the plain model, using only two rounds and a one-way tokens transfer phase that involves sending a set of tokens from the sender to the receiver in *one direction*. We begin with a protocol that comprises of three rounds where the first round only transfers tokens from one party and then later modify it to obtain a two-round protocol where the tokens are reusable and need to be transferred once at the beginning of the protocol. For simplicity of exposition, in the sequel we will assume that the random coins are an implicit input to the commitments and the extractor, unless specified explicitly. Informally, in the one-way tokens transfer phase the sender sends two types of tokens. The PRF tokens $\{\mathsf{TK}_{P_1}^{\mathsf{PRF},l}\}_{l\in[4\kappa^2]}$ are used by the receiver to commit to its input b using the shares $\{b_i\}_{i\in[\kappa]}$. Namely, the number of tokens equals 4κ (which denote the number of tokens per Pass-Wee commitment), times κ which is the number of the receiver's input shares. Whereas, the commitment token $\mathsf{TK}_{P_1}^{\mathsf{Com}}$ is used by the receiver to obtain the commitments of the sender in order to mask the values $\{(s_0^i, s_1^i)\}_{i\in[\kappa]}$ which are later used to conceal the sender's real inputs to the oblivious transfer. Next, the receiver shares its bit b into $b_1, \ldots, b_\kappa$ such that $b = \bigoplus_{i=1}^{\kappa} b_i$ and commits to these shares using the Pass-Wee trapdoor commitment scheme. Importantly, we consider a slightly variant of the Pass-Wee commitment scheme where we combine the last two steps of the commit phase with the decommit phase. In particular, the final verification in the commit phase is included as part of the decommitment phase and incorporated into the sender's tokens $\{\mathsf{TK}_i\}$ that are forwarded in the second round. The sender further sends the commitments to its inputs s_0, s_1 computed based on hardcore predicates for the (s_i^0, s_i^1) values and a combiner specified as follows. The sender chooses $z_1, \ldots, z_\kappa$ and Δ at random, where $\bigoplus_{i=1}^{\kappa} z_i$ masks s_0 and $\bigoplus_{i=1}^{\kappa} z_i + \Delta$ masks s_1.

Trapdoor Commitment Scheme TCom [PW09]

The commitment scheme TCom uses a statistically binding commitment scheme Com and runs between sender P_1 and receiver P_2.

Input: P_1 holds a message $m \in \{0,1\}$.

Commit Phase:

$P_2 \to P_1$: P_2 chooses a challenge $e = e_1, \ldots, e_\kappa \leftarrow \{0,1\}$ and sends the commitment $\mathsf{com}_e \leftarrow \mathsf{Com}(e)$ to P_1.

$P_1 \to P_2$: P_1 proceeds as follows:

1. P_1 chooses $\eta_1, \ldots, \eta_\kappa \leftarrow \{0,1\}^\kappa$.
2. For all $i \in [\kappa]$, P_1 commits to the following matrix:

$$\begin{pmatrix} \mathsf{com}^{00}_{\eta_i} & \mathsf{com}^{01}_{m+\eta_i} \\ \\ \mathsf{com}^{10}_{\eta_i} & \mathsf{com}^{11}_{m+\eta_i} \end{pmatrix} = \begin{pmatrix} \mathsf{Com}(\eta_i) & \mathsf{Com}(m+\eta_i) \\ \\ \mathsf{Com}(\eta_i) & \mathsf{Com}(m+\eta_i) \end{pmatrix}$$

$P_2 \to P_1$: P_2 sends decom_e of the challenge $e = e_1, \ldots, e_\kappa \leftarrow \{0,1\}$ to P_1.

$P_1 \to P_2$: P_1 proceeds as follows:

1. For all $i \in [\kappa]$, P_1 sends the decommitments of the column $(\mathsf{decom}^{0e_i}_{(e_i \cdot m)+\eta_i}, \mathsf{decom}^{1e_i}_{(e_i \cdot m)+\eta_i})$.
2. For all $i \in [\kappa]$, P_2 checks that the decommitments are valid and that $\mathsf{decom}^{0e_i}_{(e_i \cdot m)+\eta_i} = \mathsf{decom}^{1e_i}_{(e_i \cdot m)+\eta_i}$.

Decommit Phase:

1. For all $i \in [\kappa]$, P_1 chooses $r = r_1, \ldots, r_\kappa \leftarrow \{0,1\}$ and sends the bit m and the decommitments of the row $(\mathsf{decom}^{r_i 0}_{\eta_i}, \mathsf{decom}^{r_i 1}_{r_i+\eta_i})$.
2. For $i \in [\kappa]$, P_2 checks that the decommitments are valid and that $m = \mathsf{decom}^{r_i 0}_{\eta_i} + \mathsf{decom}^{r_i 1}_{r_i+\eta_i}$.

Figure 5.3: Trapdoor commitment scheme

Finally, the sender respectively commits to each z_i and $z_i + \Delta$ using the hardcore bits computed on the (s_i^0, s_i^1) values. More precisely, it sends

$$s_0' = w + s_0 \text{ and } s_1' = w + \Delta + s_1$$
$$\forall\ i \in [\kappa]\ w_i^0 = z_i + \mathsf{H}(s_i^0) \text{ and } w_i^1 = z_i + \Delta + \mathsf{H}(s_i^1)$$

where $w = \bigoplus_{i=1}^{\kappa} z_i$. If none of the tokens abort, the receiver obtains $s_i^{b_i}$ for all $i \in [\kappa]$ and computes $s_b = s_b' + (w_1^{b_1} + H(s_1^{b_1})) \cdots + (w_\kappa^{b_\kappa} + H(s_\kappa^{b_\kappa}))$. If any of the OT tokens, i.e. TK_i, aborts then the receiver assumes a default value for s_b.

Remark 5.1. In [GIS$^+$10], it is pointed out by Goyal et al. in Footnote 12 that assuming a default value in case the token aborts might cause an input-dependent abort. However, this problem arises only in their protocol as a result of the faulty simulation. In particular, our protocol is not vulnerable to this since the simulator for a corrupted sender follows the honest receiver's strategy to extract both the inputs via (statistical) equivocation. In contrast, the simulation in [GIS$^+$10] runs the honest receiver's strategy for a randomly chosen input in a main execution to obtain the (adversarially corrupted) sender's view and uses a "receiver-independent" strategy to extract the sender's inputs. For more details, see Section 5.6.

Remark 5.2. In Footnote 10 of [GIS$^+$10], Goyal et al. explain why it is necessary that the receiver run the token implementing the one-time memory functionality (OTM) in the prescribed round. More precisely, they provide a scenario where the receiver can violate the security of a larger protocol in the OT-hybrid by delaying when the token implementing the OTM is executed. Crucial to this attack is the ability of the receiver to run the OTM token on different inputs. In order to prevent such an attack, the same work incorporates a mechanism where the receiver is forced to run the token in the prescribed round. We remark here that our protocol is not vulnerable to such an attack. We ensure that there is only one input on which the receiver can query the OTM token and this invalidates the attack presented in [GIS$^+$10].

Next, we describe our OT protocol Π_{OT} in the $\mathcal{F}_{\text{gWRAP}}$-hybrid with sender P_1 and receiver P_2. Let (1) Com be a non-interactive perfectly binding commitment scheme, (2) $\mathsf{TCom} = \{\mathsf{TCmsg}_1, \mathsf{TCmsg}_2, \mathsf{TCmsg}_3\}$ denote the three messages exchanged in the commit phase of the trapdoor commitment scheme, (3) F, F' be two PRF families that map $\{0,1\}^{5\kappa} \to \{0,1\}^{\kappa}$ and $\{0,1\}^{\kappa} \to \{0,1\}^{p(\kappa)}$, respectively (4) H denote a hardcore bit function and (5) $\mathsf{Ext} : \{0,1\}^{5\kappa} \times \{0,1\}^{d} \to \{0,1\}$ denote a randomness extractor where the source has length 5κ and the seed has length d (for simpler exposition we drop the randomness in the description below).

Protocol 1. Protocol Π_{OT} - OT with stateless tokens in the plain model.

- **Input:** P_1 holds two strings $s_0, s_1 \in \{0,1\}^{\kappa}$ and P_2 holds a bit b. The common input is sid.
- **The Protocol:**

$P_1 \to P_2$: The sender creates two sets of tokens as follows and sends them to the receiver.

1. $\{\mathsf{TK}_{P_1}^{\mathsf{PRF},l}\}_{l\in[4\kappa^2]}$: P_1 chooses $4\kappa^2$ random PRF keys $\{\gamma_l\}_{l\in[4\kappa^2]}$ for family F. Let $\mathsf{PRF}_{\gamma_l} : \{0,1\}^{5\kappa} \to \{0,1\}^{\kappa}$ denote the pseudorandom function. For all $l \in [4\kappa^2]$, P_1 creates a token $\mathsf{TK}_{P_1}^{\mathsf{PRF},l}$ by sending $(\mathsf{Create}, \mathsf{sid}, P_1, P_2, \mathsf{mid}_l, M_1)$ to $\mathcal{F}_{\text{gWRAP}}$, that on input $(\overline{\mathsf{sid}}, x)$ outputs $\mathsf{PRF}_{\gamma_l}(x)$, where M_1 is the functionality; if $\overline{\mathsf{sid}} \neq \mathsf{sid}$ the token aborts.
2. $\mathsf{TK}_{P_1}^{\mathsf{Com}}$: P_1 chooses a random PRF' key γ' for family F'. Let $\mathsf{PRF}'_{\gamma'} : \{0,1\}^{\kappa} \to \{0,1\}^{p(\kappa)}$ denote the pseudorandom function. P_1 creates token $\mathsf{TK}_{P_1}^{\mathsf{Com}}$ by sending $(\mathsf{Create}, \mathsf{sid}, P_1, P_2, \mathsf{mid}_{l+1}, M_2)$ to $\mathcal{F}_{\text{gWRAP}}$ where M_2 is the functionality that on input $(\overline{\mathsf{sid}}, \mathsf{tcom}_{b_i}, i)$ proceeds as follows:
   - For the case where $\overline{\mathsf{sid}} \neq \mathsf{sid}$ the token aborts;
   - If $i = 0$: compute $V = \mathsf{PRF}'_{\gamma'}(0^{\kappa})$, parse V as $e\|r$ and output $\mathsf{com}_e \leftarrow \mathsf{Com}(e; r)$.
   - Otherwise: compute $V = \mathsf{PRF}'_{\gamma'}(\mathsf{tcom}_{b_i}\|i)$, parse V as $s_0^i\|s_1^i\|r_0\|r_1$, compute $\mathsf{com}_{s_b^i} \leftarrow \mathsf{Com}(s_b^i; r_b)$ for $b = \{0,1\}$, and output $\mathsf{com}_{s_0^i}, \mathsf{com}_{s_1^i}$.

We remark that if V is longer than what is required in either case, we simply truncate it to the appropriate length.

$P_2 \rightarrow P_1$:

1. P_2 sends $(\mathsf{Run}, \mathsf{sid}, \mathsf{mid}_{l+1}, (0^\kappa, 0))$ and receives com_e and interprets it as TCmsg_1.
2. For all $i \in [\kappa]$ and $j \in [4\kappa]$, P_2 sends $(\mathsf{Run}, \mathsf{sid}, \mathsf{mid}_{1_l}, u_i^j)$ where $u_i^j \leftarrow \{0,1\}^{5\kappa}$ and receives $v_i^j = \mathsf{TK}_{P_1}^{\mathsf{PRF},l}(u_i^j)$ (where $l \in [4\kappa^2]$ is an encoding of the pair (i,j)). If the token aborts the receiver aborts.
3. P_2 chooses $\kappa - 1$ random bits $b_1, \ldots, b_{\kappa-1}$ and sets b_κ such that $b = \bigoplus_{i=1}^{\kappa} b_i$. For all $i \in [\kappa]$, it commits to b_i be setting $\mathsf{tcom}_{b_i} = (M_1^i, \ldots, M_\kappa^i)$. In particular, $\forall j \in [\kappa]$ the receiver picks $\eta_{i,j} \leftarrow \{0,1\}^\kappa$ as per Figure 5.3 and computes:
$$M_j^i = \begin{pmatrix} (\mathsf{Ext}(u_i^{4j-3}) + \eta_{i,j}, v_i^{4j-3}) & (\mathsf{Ext}(u_i^{4j-1}) + b_i + \eta_{i,j}, v_i^{4j-1}) \\ (\mathsf{Ext}(u_i^{4j-2}) + \eta_{i,j}, v_i^{4j-2}) & (\mathsf{Ext}(u_i^{4j}) + b_i + \eta_{i,j}, v_i^{4j}) \end{pmatrix}.$$
4. For all $i \in [\kappa]$, P_2 sends tcom_{b_i}.

$P_1 \rightarrow P_2$:

1. P_1 chooses $z_1, \ldots, z_\kappa, \Delta \leftarrow \{0,1\}$, computes $w = \bigoplus_{i=1}^{\kappa} z_i$ and sends $s_0' = w + s_0$, $s_1' = w + \Delta + s_1$ and $\{w_i^0 = z_i + \mathsf{H}(s_i^0), w_i^1 = z_i + \Delta + \mathsf{H}(s_i^1)\}_{i \in [\kappa]}$ where (s_i^0, s_i^1) are computed by running the code of the token $\mathsf{TK}_S^{\mathsf{Com}}$ on input $\mathsf{tcom}_{b_i} \| i$.
2. P_1 sends $\mathsf{TCmsg}_3 = (e, \mathsf{decom}_e)$.
3. For all $i \in [\kappa]$, P_1 creates a token TK_i by sending $(\mathsf{Create}, \mathsf{sid}, P_1, P_2, \mathsf{mid}_{l+1+i}, M_3)$ to $\mathcal{F}_{\text{gWRAP}}$ where M_3 implements the following functionality:
   On input $(\overline{\mathsf{sid}}, b_i, \mathsf{TCdecom}_{b_i})$:
   – For the case where $\overline{\mathsf{sid}} \neq \mathsf{sid}$ the token aborts;
   If $\mathsf{TCdecom}_{b_i}$ is verified correctly then output $(s_b^i, \mathsf{decom}_{s_b^i})$, else output $(\perp, \perp)$

- **Output Phase:**

1. For all $i \in [\kappa]$, P_2 sends $(\mathsf{Run}, \mathsf{sid}, \mathsf{mid}_{l+1}, (\mathsf{tcom}_{b_i}, i))$ and receives $\mathsf{com}_{s_0^i}, \mathsf{com}_{s_0^i}$.
2. For all $i \in [\kappa]$, P_2 sends $(\mathsf{Run}, \mathsf{sid}, \mathsf{mid}_{l+1+i}, (b_i, \mathsf{TCdecom}_{b_i}))$ and receives $(s_b^i, \mathsf{decom}_{s_b^i})$. If the decommitments $\mathsf{decom}_{s_b^i}$ and decom_e are valid, P_2 computes $\tilde{z}_i = \mathsf{H}(s_b^i) + w_i^{b_i}$ and $s_b = \bigoplus_{i=1}^{\kappa} \tilde{z}_i + s_b'$. If any of the tokens abort, the receiver sets $s_b = \perp$, where $\perp$ is a default value.

Next, we prove the following theorem,

Theorem 5.4.1. Assume the existence of one-way permutations, then Protocol Π_{OT} securely realizes $\mathcal{F}_{\text{OT}}$ in the $\mathcal{F}_{\text{gWRAP}}$-hybrid.

Proof overview. On a high-level, when the sender is corrupted the simulator rewinds the adversary in order to extract both P_1's inputs to the OT. Namely, in the first execution simulator $\mathcal{S}$ plays the role of the honest receiver with input 0 and learns the challenge e. It then rewinds the adversary and changes the receiver's commitments b_i's in a way that allows equivocating these commitments into both $b = 0$ and $b = 1$. Finally, $\mathcal{S}$ queries the tokens $\{\mathsf{TK}_i\}_{i \in [\kappa]}$ twice by communicating with $\mathcal{F}_{\text{gWRAP}}$ and decommiting into two different sets of bit-vectors, which allows $\mathcal{S}$ to extract both inputs s_0 and s_1. The security proof follows by exploiting the trapdoor commitment property, which allows in the simulation to open the commitments of the receiver's input shares $\{b_i\}_{i\in[\kappa]}$ into two distinct bit-vectors that correspond to distinct bits. The indistinguishability argument asserts that the simulated and real views are statistically close, due to the statistical hiding property of the commitment scheme that we use within the Pass-Wee trapdoor commitment scheme.

On the other hand, when the receiver is corrupted the simulator extracts its input b based on the first message and the queries to the tokens. We note that extraction must be carried out carefully, as the receiver commits to each bit b_i using κ matrices and may commit to different bits within each set of matrices (specifically, there may be commitment for which the committed bit is not even well defined). Upon extracting b, the proof continues by considering a sequence of hybrids where we replace the hardcore bits for the positions $\{b_i + 1\}_{i\in[\kappa]}$. Specifically, these are the positions in which the receiver cannot ask for decommitments and hence does not learn $\{s^i_{b_i+1}\}_{i\in[\kappa]}$. Our proof of indistinguishability relies on the list-decoding ability of the Goldreich-Levin hardcore predicate (cf. Theorem 5.3.2), that allows extraction of the input from an adversary that can guess the hardcore predicate on the input with probability significantly better than a half.

Proof. We consider each corruption case separately. In case the adversary $\mathcal{A}$ issues a transfer query $(\mathsf{transfer}, \cdot)$, $\mathcal{S}$ transfers the query to the $\mathcal{F}_{\text{gWRAP}}$.

Note that in this protocol there is no need to allow transfer queries to the $\mathcal{F}_{\text{gWRAP}}$ functionality.

Simulating the corrupted P_1. Let $\mathcal{A}$ be a PPT adversary that corrupts P_1 then we construct a simulator $\mathcal{S}$ as follows,

1. $\mathcal{S}$ invokes $\mathcal{A}$ on its input and a random string of the appropriate length.

2. Adversary $\mathcal{A}$ communicates with functionality $\mathcal{F}_{\text{gWRAP}}$ on behalf of the corrupted party by sending create messages $\{(\mathsf{Create}, \mathsf{sid}, P_1, P_2, \mathsf{mid}_l, M_1)\}_{l\in[4\kappa^2]}$ and $(\mathsf{Create}, \mathsf{sid}, P_1, P_2, \mathsf{mid}_{l+1}, M_2)$. Then $\mathcal{F}_{\text{gWRAP}}$ forwards these tokens to the honest party by sending receipt messages $\{(\mathsf{Receipt}, \mathsf{sid}, P_1, P_2, \mathsf{mid}_l, M_1)\}_{l\in[4\kappa^2]}$ and $(\mathsf{Receipt}, \mathsf{sid}, P_1, P_2, \mathsf{mid}_{l+1}, M_2)$.

3. Upon receiving acknowledgement messages $\{(\mathsf{Receipt}, \mathsf{sid}, P_1, P_2, \mathsf{mid}_l, \cdot)\}_{l\in[4\kappa^2+1]}$ that all $[4\kappa^2] + 1$ tokens have been created by $\mathcal{A}$, $\mathcal{S}$ emulates the role of the honest receiver using an input bit $b = 0$. If com_e is decommitted correctly, $\mathcal{S}$ stores this value and rewinds the adversary to the first message. Otherwise, $\mathcal{S}$ halts and outputs $\mathcal{A}$'s view thus far, sending $(\perp, \perp)$ to the ideal functionality.

4. $\mathcal{S}$ picks two random bit-vectors $(b_1, \ldots, b_\kappa)$ and $(b'_1, \ldots, b'_\kappa)$ such that $\bigoplus_{i=1}^{\kappa} b_i = 0$ and $\bigoplus_{i=1}^{\kappa} b'_i = 1$. Let $e = e_1, \ldots, e_\kappa$ denote the decommitment of com_e obained from the previous step. Then, for all $i, j \in [\kappa]$, $\mathcal{S}$ sends matrix M_i^j where the e_ith column is defined by
$$\begin{pmatrix} (\mathsf{Ext}(u_i^{4j-3}) + \eta_{i,j}, & v_i^{4j-3}) \\ (\mathsf{Ext}(u_i^{4j-2}) + \eta_{i,j}, & v_i^{4j-2}) \end{pmatrix}$$

whereas the $(1-e_i)$th column is set

$$\text{w.p. } \frac{1}{2} \text{ to } \begin{pmatrix} (\mathsf{Ext}(u_i^{4j-1}) + \eta_{i,j}, & v_i^{4j-1}) \\ (\mathsf{Ext}(u_i^{4j}) + 1 + \eta_{i,j}, & v_i^{4j}) \end{pmatrix}, \text{ and}$$
$$\text{w.p. } \frac{1}{2} \text{ to } \begin{pmatrix} (\mathsf{Ext}(u_i^{4j-1}) + 1 + \eta_{i,j}, & v_i^{4j-1}) \\ (\mathsf{Ext}(u_i^{4j}) + \eta_{i,j}, & v_i^{4j}) \end{pmatrix}.$$

5. Upon receiving the sender's message the simulator checks if com_e is decommitted correctly. Otherwise, $\mathcal{S}$ rewinds the adversary to before the first message was sent and returns to Step 4. In each rewinding $\mathcal{S}$ uses fresh randomness to generate the receiver's message. It repeatedly rewinds until the malicious sender successfully decommits e. If it tries to make more than $2^{\kappa/2}$ attempts, it simply halts outputting fail.

   Next, to extract s_0, it decommits to $b_1, \ldots, b_n$ (and to extract s_1, it decommits to $(b'_1, \ldots, b'_n)$. Recall that to reveal a commitment to a value b_i the simulator decommits that row of the matrix that adds up to b_i. Notice that by out construction, such a row always exists and is either the first row or the second row with the probability $1/2$. We remark here that the simulator $\mathcal{S}$ creates the code of the actual Turing Machine incorporated in the token as opposed to running the token itself. Furthermore, each of the two extractions start with the Turing Machine in the same start (as opposed to running the machine in sequence). This is because the code in the malicious token can be stateful and rewinding it back to the start state prevents stateful behavior. More precisely, the simulator needs to proceed exactly as the honest receiver would in either case. If for any $b \in \{0,1\}$ extraction fails for s_b, then following the honest receiver's strategy the simulator sets s_b to the default value $\perp$.

6. Finally, $\mathcal{S}$ sends (s_0, s_1) to the trusted party that computes $\mathcal{F}_{\mathrm{OT}}$ and halts, outputting whatever $\mathcal{A}$ does.

We now prove that the sender's view in both the simulated and real executions is computationally indistinguishable via a sequence of hybrid executions. More formally,

Lemma 5.4.1. The following two ensembles are computationally indistinguishable,

$$\left\{\mathrm{IDEAL}_{\mathcal{F}_{\mathrm{OT}},\mathcal{S}(z),I}(\kappa,(s_0,s_1),b)\right\}_{\kappa\in\mathbb{N},s_0,s_1,b,z\in\{0,1\}^*} \overset{\mathrm{c}}{\approx} \left\{\mathrm{REAL}^{\mathcal{F}_{\mathrm{gWRAP}}}_{\Pi_{\mathrm{OT}},\mathcal{A}(z),I}(\kappa,(s_0,s_1),b)\right\}_{\kappa\in\mathbb{N},s_0,s_1,b,z\in\{0,1\}^*}$$

Proof. Roughly speaking, we prove that the joint output distribution of both the receiver and the sender is computationally indistinguishable. Our proof follows by a sequence of hybrid executions defined below. We denote by $\mathbf{Hybrid}^i_{\mathcal{F}_{\mathrm{OT}},\mathcal{S}_i(z),I}(\kappa,(s_0,s_1),b)$ the random variable that corresponds to the simulator's output in hybrid execution $\mathrm{H_i}$ when running against party $\mathcal{S}_i$ that plays the role of the receiver according to the specifications in this hybrid (where $\mathcal{S}_0$ refers to the honest real receiver). For simplicity of exposition, we abbreviate it to Hybrid_i.

Hybrid H_0: In the first hybrid, we consider a simulator S_0 that receives the real input b of the receiver and simply follows the protocol as the honest receiver would. Finally, it outputs the view of the adversary and the receiver's output as computed in the emulation. It follows from construction that the distribution of the output of the first hybrid is identical to the real execution.

Hybrid H_1: In this hybrid, the simulator $\mathcal{S}_1$ receives the real input of the receiver and proceeds as follows. It first interacts with the adversary with the actual receiver's input and checks if it successfully decommits e. If it does not, then the simulator simply outputs the view of the adversary and $\perp$ as the receiver's output. Otherwise, it proceeds to a rewinding phase. In this phase, it repeatedly rewinds the adversary to the first message and then samples a new first message by committing to b using fresh randomness. Specifically, $\mathcal{S}_1$ invokes token $\mathsf{TK}_{P_1}^{\mathsf{PRF},l}$ each time on new random inputs u_i^j (for all $i \in [\kappa]$, $j \in [4\kappa]$ where l encodes (i,j)), and continues rewinding $\mathcal{A}$ until it obtains an interaction in which the adversary successfully decommits to e again. If the simulation makes more than $2^{\kappa/2}$ rewinding attempts, then it aborts.

We now argue that the view produced in this hybrid is statistically close to the view produced within the previous hybrid. Observe that if the simulation does not cut off after $2^{\kappa/2}$ attempts, then the view is identically distributed to the view in H_0. Therefore to show that the views are statistically close, it suffices to prove that the simulation aborts with negligible probability. Let p be the probability with which the adversary decommits e correctly when the receiver honestly generates a commitment to b. We consider two cases:

- If $p > \frac{\kappa}{2^{\kappa/2}}$, then the probability that the simulation takes more than $2^{\kappa/2}$ steps can be computed as $\left(1 - \frac{\kappa}{2^{\kappa/2}}\right)^{2^{\kappa/2}} = e^{-\kappa}$ and which is negligible in κ.
- If $p < \frac{\kappa}{2^{\kappa/2}}$, then the probability that the simulation aborts is bounded by the probability that it proceeds to the rewinding phase which is at most p and hence negligible in κ.

It only remains to argue that the expected running time of the simulation is polynomial. We remark that this follows from Lemma 5.4.2 proven in the next hybrid by setting p_0 and p_b to p.

Hybrid H_2: In this hybrid, the simulator $\mathcal{S}_2$ proceeds identically to $\mathcal{S}_1$ with the exception that in the first run where the simulator looks for the decommitment of e, it follows the honest receiver's strategy with input 0 instead of its real input. Now, since each commitment is generated honestly, it follows from Lemma 5.5.1 using an union bound that the first message generated by the receiver with input b and input 0 are $4\kappa^2 2^{-\kappa-1}$-close. Moreover, since the only difference in the two hybrids is within the first message sent by the receiver in the first execution, the distributions Hybrid_1 and Hybrid_2 are statistically close.

It only remains to argue that the running time of the simulation is still polynomial.

Lemma 5.4.2. The expected running time of $\mathcal{S}_2$ is polynomial-time and the probability that $\mathcal{S}_2$ aborts is negligible.

Proof: Let p_0 be the probability that adversary successfully decommits to e in the main execution of hybrid H_2 and p_b be the probability that the adversary successfully decommits when the receiver's commitments are made to the real input b. Now, since the first message of the receiver when the commitment is made to 0 or b is $2^{-O(\kappa)}$-close, we have that $|p_0 - p_b| < 2^{-O(\kappa)}$.

Next, we prove that the expected number of times the simulator runs the execution is

$$(1-p_0) \times 1 + p_0 \times \min\left\{2^{\kappa/2}, \frac{1}{p_b}\right\}.$$

We consider two cases and argue both regarding the running time and abort probability in each case.

- $p_0 > 2\kappa 2^{-\kappa/2}$: Since $|p_b - p_0| < 2^{-O(\kappa)}$, it follows that,

  $$p_b > p_0 - 2^{-O(\kappa)} = 2\kappa 2^{-\kappa/2} - 2^{-O(\kappa)} > \kappa 2^{-\kappa/2} = \frac{p_0}{2}$$

  Therefore, $p_0/p_b < 2$. Now, since $\min\left\{2^{\kappa/2}, \frac{1}{p_b}\right\} = \frac{1}{p_b}$, the expected number of rewinding attempts is

  $$(1-p_0) + p_0 \times \frac{1}{p_b} < 3$$

  which is polynomial.

  Next, we argue regarding the abort probability. Specifically, the probability that the number of attempts exceeds $2^{\kappa/2}$ is given by

  $$(1-p_b)^{2^{\kappa/2}} < \left(1 - \frac{\kappa}{2^{\kappa/2}}\right)^{2^{\kappa/2}} = O(e^{-\kappa}).$$

  Therefore, the probability that the simulator aborts is negligible.

- $p_0 < 2\kappa 2^{-\kappa/2}$: Since $\min\left\{2^{\kappa/2}, \frac{1}{p_b}\right\} = 2^{\kappa/2}$, the expected number of rewinding attempts is

  $$(1-p_0) + p_0 \times 2^{\kappa/2} < 1 + 2\kappa^2$$

  which is polynomial.

  The abort probability in this case is bounded by p_0 which is negligible.

$\square$

Hybrids $\mathrm{H}_{3,0} \ldots, \mathrm{H}_{3,\kappa}$: We define a collection of hybrid executions such that for every $i \in [\kappa]$ hybrid $\mathrm{H}_{3,\mathrm{i}}$ is defined as follows. Assume that $(b_1, \ldots, b_\kappa)$ correspond to the bit-vector for the real input of the receiver b. Then in $\mathrm{H}_{3,\mathrm{i}}$, the first i commitments are computed as in the simulation (i.e. equivocated using the trapdoor e), whereas the remaining $\kappa - i$ commitments are set as commitments of $b_{i+1}, \ldots, b_\kappa$ as in the real execution. Note that hybrid $\mathrm{H}_{3,0}$ is identical to hybrid H_2 and that the difference between every two consecutive hybrids $\mathrm{H}_{3,\mathrm{i}-1}$ and $\mathrm{H}_{3,\mathrm{i}}$ is regarding the way the ith commitment is computed, which is either a commitment to b_i computed honestly in the former hybrid, or equivocated using the trapdoor in the latter hybrid. Indistinguishability of $\mathrm{H}_{3,\mathrm{i}}$ and $\mathrm{H}_{3,\mathrm{i}-1}$ follows similarly to the indistinguishability argument of H_1 and H_2, as the only difference is in how the unopened commitments are generated. Therefore, we have the following lemma.

Claim 5. For every $i \in [\kappa]$, $\mathrm{Hybrid}_{1,i-1} \overset{\mathrm{s}}{\approx} \mathrm{Hybrid}_{1,i}$

Note that the proof regarding the expected running time of the simulator is identical to the proof of Lemma 5.4.2.

Hybrid H_4: In this hybrid, we consider the actual simulator. First, we observe that the view of the adversary output by $\mathcal{S}_{3,\kappa}$ in $\mathrm{H}_{3,\kappa}$ is independent of the receiver's real input b. This is because in $\mathrm{H}_{3,\kappa}$, all commitments are computed in an equivocation mode, where the real input b of the receiver is used only after the view of the adversary is generated. More precisely, only after $\mathcal{S}_{3,\kappa}$ obtains a second view on which the adversary successfully decommits to e, does it use the tokens to extract s_b by decommitting the equivocal commitments to $b_1, \ldots, b_n$ such that $\bigoplus_i b_i = b$. In fact, since in the rewinding phase all the commitments are equivocated, the b_i's themselves can also be sampled after the view of the adversary is generated.

Next, we observe that the actual simulator proceeds exactly as $\mathcal{S}_{3,\kappa}$ with the exception that it communicates with $\mathcal{F}_{\mathrm{gWRAP}}$ in order to run the tokens twice after the adversary's view is obtained and the rewinding phase is completed. Namely, it asks $\mathcal{F}_{\mathrm{gWRAP}}$ to run the token once with a vector of b_i's that add up to 0 in order to obtain s_0, then rewinds the tokens back to the original state and runs them another time with a vector of b_i''s that add up to 1 in order to extract s_1. (s_0, s_1) are then fed to the ideal functionality. Recall that $\mathcal{S}_{3,\kappa}$ on the other hand, runs the tokens only once for the actual receiver's input b. Now, since the view of the adversary in $\mathrm{H}_{3,\kappa}$ and **IDEAL** are identically distributed, it follows that the value extracted for s_b in $\mathrm{H}_{3,\kappa}$ is identically distributed to s_b in the ideal execution for both $b = 0$ and $b = 1$. Therefore, we can conclude that the output of the simulator in $\mathrm{H}_{3,\kappa}$ and the joint output of the simulator and honest receiver the ideal execution, are identically distributed.

Claim 6. $\mathrm{Hybrid}_{3,\kappa} \approx_{\mathrm{c}} \mathrm{Hybrid}_4$

■

Simulating the corrupted P_2. Let $\mathcal{A}$ be a PPT adversary that corrupts P_2 then we construct a simulator $\mathcal{S}$ as follows,

1. $\mathcal{S}$ invokes $\mathcal{A}$ on its input and a random string of the appropriate length.
2. $\mathcal{S}$ communicates with $\mathcal{F}_{\mathrm{gWRAP}}$ on behalf of the honest party by sending create messages $\{(\mathsf{Create}, \mathsf{sid}, P_1, P_2, \mathsf{mid}_l, M_1)\}_{l \in [4\kappa^2]}$ and $(\mathsf{Create}, \mathsf{sid}, P_1, P_2, \mathsf{mid}_{l+1}, M_2)$, where the code M_1 implements truly random functions (that is, M_1 is encoded with a lookup table that includes some polynomial number of queries bounded by the running time of the adversary).

Then $\mathcal{F}_{\text{gWRAP}}$ forwards these tokens by sending receipt messages $\{(\mathsf{Receipt}, \mathsf{sid}, P_1, P_2, \mathsf{mid}_l, M_1)\}_{l\in[4\kappa^2]}$ and $(\mathsf{Receipt}, \mathsf{sid}, P_1, P_2, \mathsf{mid}_{l+1}, M_2)$ to $\mathcal{A}$. For each query $u \in \{0,1\}^{5\kappa}$ made by $\mathcal{A}$ to token $\mathsf{TK}_{P_1}^{\mathsf{PRF},l}$, functionality $\mathcal{F}_{\text{gWRAP}}$ runs M_1 on that query and returns a random v from $\{0,1\}^\kappa$. Similarly, M_2 implements a random function that maps elements from $\{0,1\}^\kappa \to \{0,1\}^{p(\kappa)}$.

3. Next, $\mathcal{S}$ retrieves $\mathcal{A}$ queries for session sid from $\mathcal{F}_{\text{gWRAP}}$ by sending a $(\mathsf{retreive}, \mathsf{sid})$ message receiving the list $\mathcal{Q}_{\mathsf{sid}}$. $\mathcal{S}$ splits the set of receiver's queries (tcom_b, i^*) to the token $\mathsf{TK}_{P_1}^{\mathsf{Com}}$ (that were further part of the adversary's message), and adds them either to the "valid" set $\mathcal{I}_{\mathsf{Com}}$ or "invalid" set $\mathcal{J}_{\mathsf{Com}}$. More formally, let $T = q(\kappa)$ denote the number of times the token $\mathsf{TK}_{P_1}^{\mathsf{Com}}$ is queried by P_2 for some polynomial q. For each query (tcom_b, i^*), we say that the query is valid if and only if there exist values $\{(\beta_i^t, u_i^t, v_i^t)\}_{i\in[\kappa],t\in[4\kappa]}$ such that $\mathsf{tcom}_{b_i} = (M_1^i, \ldots, M_\kappa^i)$, $\forall i, j \in [\kappa]$,

$$M_j^i = \begin{pmatrix} \beta_i^{4j-3}, v_i^{4j-3} & \beta_i^{4j-1}, v_i^{4j-1} \\ \\ \beta_i^{4j-2}, v_i^{4j-2} & \beta_i^{4j}, v_i^{4j} \end{pmatrix}$$

and, for every $i \in [\kappa], t \in [4\kappa]$, the query/answer pair (u_i^t, v_i^t) has already been recorded as a query to the corresponding PRF token. Next, for every valid query, the simulator tries to extract the committed value. This it done by first computing

$$\gamma_{00}^j = \beta_i^{4j-3} + \mathsf{Ext}(u_i^{4j-3}) \qquad \gamma_{01}^j = \beta_i^{4j-1} + \mathsf{Ext}(u_i^{4j-1})$$
$$\gamma_{10}^j = \beta_i^{4j-2} + \mathsf{Ext}(u_i^{4j-2}) \qquad \gamma_{11}^j = \beta_i^{4j} + \mathsf{Ext}(u_i^{4j}).$$

Next it marks the indices j for which $\gamma_{00}^j = \gamma_{10}^j$ and $\gamma_{01}^j = \gamma_{11}^j$. Moreover, for the marked indices it computes $\gamma^j = \gamma_{00}^j + \gamma_{01}^j$. If there are at least more than half the indices that are marked and are commitments to the same value, say γ then $(\mathsf{tcom}_b, i^*, \gamma)$ is added to $\mathcal{I}_{\mathsf{Com}}$. Otherwise $(\mathsf{tcom}_b, i^*, \perp)$ is added to $\mathcal{J}_{\mathsf{Com}}$.

Next, $\mathcal{S}$ computes $b = \bigoplus_{i=1}^\kappa b_i$ and sends b to the trusted party that computes $\mathcal{F}_{\text{OT}}$. Upon receiving s_b, $\mathcal{S}$ picks a random s_{b+1} from the appropriate domain and completes the execution by playing the role of the honest sender on these two inputs.

We now prove that the receiver's view in both the simulated and real executions is computationally indistinguishable via a sequence of hybrid executions. More formally,

Lemma 5.4.3. The following two ensembles are computationally indistinguishable,

$$\left\{\text{IDEAL}_{\mathcal{F}_{\text{OT}},\mathcal{S}(z),I}(\kappa,(s_0,s_1),b)\right\}_{\kappa\in\mathbb{N},s_0,s_1,b,z\in\{0,1\}^*} \overset{c}{\approx} \left\{\text{REAL}_{\Pi,\mathcal{A}(z),I}^{\mathcal{F}_{\text{gWRAP}}}(\kappa,(s_0,s_1),b)\right\}_{\kappa\in\mathbb{N},s_0,s_1,,bz\in\{0,1\}^*}$$

Proof. Roughly speaking, we prove that the join output distribution of both the receiver and the sender is computationally indistinguishable. Now, since only the receiver (which is the corrupted party) has an input, the proof boils down to proving that the receiver's view is indistinguishable in both executions. Our proof follows by a sequence of hybrid executions defined below. We denote by $\mathbf{Hybrid}^i_{\mathcal{F}_{\text{OT}},\mathcal{S}_i(z),I}(\kappa,(s_0,s_1),b)$ the random variable that corresponds to the adversary's view in hybrid execution $\mathrm{H_i}$ when running against party $\mathcal{S}_i$ that plays the role of the sender according to the specifications in this hybrid (where $\mathcal{S}_0$ refers to the honest real sender).

Hybrid H_0: The first hybrid execution is the real execution. For simplicity of exposition, we abbreviate it to Hybrid_i.

Hybrids $\mathrm{H}_{1,0}\ldots,\mathrm{H}_{1,4\kappa^2}$: We define a collection of hybrid executions such that for every $l \in [4\kappa^2]$ hybrid $\mathrm{H_{1,l}}$ is defined as follows. We modify the code of token $\mathsf{TK}_{P_1}^{\mathsf{PRF},l}$ by replacing the function PRF_{γ_l} with a truly random function f_l. In particular, given a query u the token responds with a randomly chosen κ bit string v, rather than running the original code of M_1. We maintain a list of $\mathcal{A}$'s queries and responses so that repeated queries will be consistently answered. In addition, the code of token TK_i is modified so that it verifies the decommitment against the random functions f_l as opposed to the PRF functions previously embedded in $\mathsf{TK}_{P_1}^{\mathsf{PRF},l}$. It is simple to verify that the adversary's view in every two consecutive hybrid executions is computationally indistinguishable due to the security of the pseudorandom function PRF_{γ_l}. Moreover, since the PRF key is hidden from the receiver, it follows from the pseudorandomness property that the views in every two consecutive hybrid executions are computationally indistinguishable. More formally, we have the following lemma.

Claim 7. For every $l \in [4\kappa^2]$, $\mathrm{Hybrid}_{1,l-1} \approx_{\mathrm{c}} \mathrm{Hybrid}_{1,l}$

Hybrid H_2: Similarly, we consider a hybrid execution for which the code of token $\mathsf{TK}_{P_1}^{\mathsf{Com}}$ is modified so that it makes use of a truly random function f' rather than a pseudorandom function $\mathsf{PRF}_{\gamma'}$. Just as in the previous hybrid, we have the following Lemma.

Claim 8. $\mathrm{Hybrid}_1 \approx_{\mathrm{c}} \mathrm{Hybrid}_2$

Hybrids $\mathrm{H}_{3,0}\ldots,\mathrm{H}_{3,4\kappa^2}$: This sequence of hybrids executions is identical to hybrid H_2 except that here we ensure that no two queries made by $\mathcal{A}$ to the token $\mathsf{TK}_{P_1}^{\mathsf{PRF},l}$ have the same response. Specifically, in case of a collision simulator $\mathcal{S}_{3,l}$ aborts.

Claim 9. For every $l \in [4\kappa^2]$, $\mathrm{Hybrid}_{3,l-1} \approx_{\mathrm{s}} \mathrm{Hybrid}_{3,l}$

Proof: As we replaced PRF functions to truly random functions, we have that the probability the simulation aborts in $\mathrm{H_{3,l}}$ is at most the probability of finding a collision for a random function. To prove statistical indistinguishability it suffices to show that this probability is negligible. More formally, if the adversary makes a total of Q queries to both tokens, then the probability that any pair of queries yields a collision can be bounded by $\binom{Q}{2}2^{-\ell}$ where ℓ is the minimum length of the outputs of all random functions. In our case this is κ and hence the probability that the simulator aborts in every hybrid is negligible. $\square$

Hybrid H_4: In this hybrid execution, simulator $\mathcal{S}_4$ plays the role of the sender as in hybrid H_3 except that it extracts the adversary's input bit b as carried out in the simulation by $\mathcal{S}$.

First, we observe that for any $i \in [\kappa]$ and $t \in [4\kappa]$, the probability that the receiver reveals a valid pre-image u_i^t for v_i^t for which there does not exists a query/answer pair (u_i^t, v_i^t) collected by the simulator is exponentially small since we rely on truly random functions in this hybrid. Therefore, except with negligible probability, the receiver will be able to decommit only to $\gamma_{00}^j, \gamma_{01}^j, \gamma_{10}^j, \gamma_{11}^j$ as extracted by the simulator. Consequently, using the soundness of the Pass-Wee trapdoor commitment scheme, it follows that the receiver can only decommit to b_i and b as extracted by the simulator. Therefore, we can conclude that the probability that a malicious receiver can equivocate the commitment tcom_{b_i} is negligible. The above does not make any difference to the receiver's view which implies that,

Claim 10. $\mathrm{Hybrid}_3 \approx_{\mathrm{c}} \mathrm{Hybrid}_4$

Moreover, recall that extraction is straight-line, thus the simulator still runs in strict polynomial-time.

Hybrids $\mathrm{H}_{5,0}, \widetilde{\mathrm{H}}_{5,0}, \ldots, \mathrm{H}_{5,\kappa}, \widetilde{\mathrm{H}}_{5,\kappa}$: Let tcom_{b_i} be the ith commitment sent to P_1 in the first message. Then $\mathrm{H}_{5,\mathrm{i}}$ proceeds identically to $\widetilde{\mathrm{H}}_{5,i-1}$, whereas $\widetilde{\mathrm{H}}_{5,i}$ proceeds identically to $\mathrm{H}_{5,\mathrm{i}}$, with the following exceptions:

- If there exists a tuple $(\mathsf{tcom}_{b_i}, i, \gamma)$ in $\mathcal{I}_{\mathsf{Com}}$, then in experiment $\mathrm{H}_{5,\mathrm{i}}$, $\mathsf{H}(s_{\gamma+1}^i)$ is replaced by a random bit in the second message fed to the adversary.
- If there exists a tuple $(\mathsf{tcom}_{b_i}, i, \perp)$ in $\mathcal{J}_{\mathsf{Com}}$, then in experiment $\mathrm{H}_{5,\mathrm{i}}$, $\mathsf{H}(s_0^i)$ is replaced by a random bit in the second message fed to the adversary.
- If there exists a tuple $(\mathsf{tcom}_{b_i}, i, \perp)$ in $\mathcal{J}_{\mathsf{Com}}$, then in experiment $\widetilde{\mathrm{H}}_{5,i}$, $\mathsf{H}(s_1^i)$ is replaced by a random bit in the second message fed to the adversary.

Note that hybrid $\mathrm{H}_{5,0}$ is identical to hybrid H_4 and that the difference between every pair of consecutive hybrids $\mathrm{H}_{5,\mathrm{i}-1}$ and $\mathrm{H}_{5,\mathrm{i}}$ is with respect to $\mathsf{H}(s_{b_i+1}^i)$ in case $i \in [|\mathcal{I}_{\mathsf{Com}}|]$ or $(\mathsf{H}(s_0^i), \mathsf{H}(s_1^i))$ in case $i \in [|\mathcal{J}_{\mathsf{Com}}|]$, that are replaced with a random bit in $\mathrm{H}_{5,\mathrm{i}}$. We now prove the following.

Claim 11. For every $i \in [\kappa]$, $\widetilde{\mathrm{Hybrid}}_{5,i-1} \approx_{\mathrm{c}} \mathrm{Hybrid}_{5,i}$ and $\mathrm{Hybrid}_{5,i} \approx_{\mathrm{c}} \widetilde{\mathrm{Hybrid}}_{5,i}$

Proof: Intuitively, the indistinguishability of any pair of hybrids follows from the computational hiding property of the commitment scheme Com and the binding property of tcom_{b_i}. Assume for contradiction, that there exists $i \in [\kappa]$ for which $\widetilde{\mathrm{Hybrid}}_{5,i-1}$ and $\mathrm{Hybrid}_{5,i}$ are distinguishable by a PPT distinguisher D with probability ε.

If there exists a tuple $(\mathsf{tcom}_{b_i}, i, \gamma) \in \mathcal{I}_{\mathsf{Com}}$ then define $b^* = 1 + \gamma$, otherwise define $b^* = 0$. Then, it follows that the only difference between hybrids $\widetilde{\mathrm{H}}_{5,i-1}$ and $\mathrm{H}_{5,\mathrm{i}}$ is that $\mathsf{H}(s_{b^*}^i)$ is computed correctly in $\widetilde{\mathrm{H}}_{5,i-1}$ while replaced with a random bit in $\mathrm{H}_{5,\mathrm{i}}$. Next, we show how to build an adversary $\mathcal{A}_{\mathsf{Com}}$ that on input a commitment $\mathsf{Com}(s)$ identifies $\mathsf{H}(s)$ with probability non-negligibly better than $\frac{1}{2} + \frac{\varepsilon}{2}$. Then using the Goldreich-Levin Theorem

(Theorem 5.3.2), it follows that we can extract value s^i_{b*} and this violates the hiding property of the commitment scheme Com.

More formally, consider $\mathcal{A}_{\mathsf{Com}}$ that receives as input a commitment to a randomly chosen string s, namely $\mathsf{Com}(s)$. $\mathcal{A}_{\mathsf{Com}}$ internally incorporates the adversary $\mathcal{A}_{\mathsf{Com}}$ and emulates the experiment $\widetilde{\mathrm{H}}_{5,i}$ with the exception that in place of $\mathsf{Com}(s^i_{b*})$, $\mathcal{A}_{\mathsf{Com}}$ instead feeds $\mathsf{Com}(s)$ and replaces $\mathsf{H}(s^i_{b*})$ with a uniformly chosen bit, say $\tilde{b}$. Finally, it feeds the output of the hybrid experiment conducted internally, namely, the view of the adversary to D, and computes an output based on D's output g as follows:

- If $g = 1$, then $\mathcal{A}_{\mathsf{Com}}$ outputs the value for $\tilde{b}$ as the prediction for $\mathsf{H}(s)$, and outputs $1 - \tilde{b}$ otherwise.

Denote by $\overline{\mathrm{H}}_{5,i}$ the experiment that proceeds identically to $\widetilde{\mathrm{H}}_{5,i}$ with the exception that, in place of $\mathsf{H}(s^i_{b*})$ we feed $1 + \mathsf{H}(s^i_{b*})$, namely the complement of the value of the hardcore predicate. Let $\overline{\mathrm{Hybrid}}_{5,i}$ denote the distribution of the view of the adversary in this hybrid. It now follows that

$$\begin{aligned}
\varepsilon &< \Big| \Pr[(v, s_b) \leftarrow \widetilde{\mathrm{Hybrid}}_{5,i} : D(v) = 1] - \Pr[(v, s_b) \leftarrow \mathrm{Hybrid}_{5,i} : D(v) = 1] \Big| \\
&= \Big| \Pr[(v, s_b) \leftarrow \widetilde{\mathrm{Hybrid}}_{5,i} : D(v) = 1] \\
&\qquad - \frac{1}{2}\Big(\Pr[(v, s_b) \leftarrow \widetilde{\mathrm{Hybrid}}_{5,i} : D(v) = 1] + \Pr[(v, s_b) \leftarrow \overline{\mathrm{Hybrid}}_{5,i} : D(v) = 1] \Big) \Big| \\
&= \frac{1}{2}\Big| \Pr[(v, s_b) \leftarrow \widetilde{\mathrm{Hybrid}}_{5,i} : D(v) = 1] - \Pr[(v, s_b) \leftarrow \overline{\mathrm{Hybrid}}_{5,i} : D(v) = 1] \Big|.
\end{aligned}$$

Without loss of generality we can assume that,[3]

$$\frac{1}{2}\Big(\Pr[(v, s_b) \leftarrow \widetilde{\mathrm{Hybrid}}_{5,i} : D(v) = 1] - \Pr[(v, s_b) \leftarrow \overline{\mathrm{Hybrid}}_{5,i} : D(v) = 1] \Big) > \varepsilon$$

Therefore,

$$\begin{aligned}
&\frac{1}{2}\Big(\Pr[(v, s_b) \leftarrow \widetilde{\mathrm{Hybrid}}_{5,i} : D(v) = 1] - (1 - \Pr[(v, s_b) \leftarrow \overline{\mathrm{Hybrid}}_{5,i} : D(v) = 0)] \Big) > \varepsilon \\
&\textit{i.e.}, \frac{1}{2}\Pr[(v, s_b) \leftarrow \widetilde{\mathrm{Hybrid}}_{5,i} : D(v) = 1] + \frac{1}{2}\Pr[(v, s_b) \leftarrow \overline{\mathrm{Hybrid}}_{5,i} : D(v) = 0)] > \frac{1}{2} + \varepsilon \\
&\qquad\qquad \textit{i.e.}, \Pr[\beta \leftarrow \{0,1\} : (v, s) \leftarrow H^b : D(v) = b] > \frac{1}{2} + \varepsilon
\end{aligned}$$

where $H^0 = \overline{\mathrm{Hybrid}}_{5,i}$ and $H^1 = \widetilde{\mathrm{Hybrid}}_{5,i}$. We now observe that sampling from H^b where b is uniformly chosen is equivalent to sampling from $\mathrm{H}_{5,\mathrm{i}}$. Therefore, since $\mathcal{A}_{\mathsf{Com}}$ internally emulates $\mathrm{H}_{5,\mathrm{i}}$ by selecting $\tilde{b}$ at random and the distinguisher identifies precisely if this bit $\tilde{b}$ came from H^0 or H^1 correctly, we can conclude that $\tilde{b}$ is the value of the hardcore bit when it comes from H^0 and the complement of $\tilde{b}$ when it comes from H^1. Therefore, $\mathcal{A}_{\mathsf{Com}}$ guesses $\mathsf{H}(s)$ correctly with probability $\frac{1}{2} + \varepsilon$. Using the list-decoding algorithm of

[3]Otherwise, we can replace D with another distinguisher that flips D's output.

Goldreich-Leving hardcore-predicate (cf. Theorem 5.3.2), it follows the such an adversary can be used to extract s thereby contradicting the computational hiding property of the Com scheme.

We remark that proving indistinguishability of $\widetilde{\text{Hybrid}}_{5,i}, \text{Hybrid}_{5,i}$ follows analogously and this concludes the proof of the Lemma. $\square$

Hybrids H_6: In this hybrid execution simulator $\mathcal{S}_5$ does not know the sender's inputs (s_0, s_1), but rather communicates with a trusted party that computes $\mathcal{F}_{\text{OT}}$. $\mathcal{S}_6$ behaves exactly as $\mathcal{S}_{5,\kappa}$ except that when extracting the bit b it sends it to the trusted party, which returns s_b. Moreover, $\mathcal{S}_6$ uses a random value for s_{b+1}. We argue that hybrids $\mathcal{S}_{5,\kappa}$ and $\mathcal{S}_6$ are identically distributed as the set $\{w^i_{b+1}\}_{i\in[\kappa]}$ is independent of $\{s^i_{b_i+1}\}_{i\in[\kappa]}$.

Claim 12. $\text{Hybrid}_{5,\kappa} \approx_{\text{s}} \text{Hybrid}_5$

Proof: Following from the fact that $\mathsf{H}(s^i_{b_i+1})$ is replaced with a random bit for all $i \in [\kappa]$, it must hold that the values $w_i^{1+b_i}$ are random as well as these values are masked using random independent bits instead of the set $\{\mathsf{H}(s^i_{b_i+1})\}_{i\in[\kappa]}$. As a result, these values contribute to a random value s_{b+1}. In addition, we claim that the adversary can only learn s_b where b is the bit extracted by $\mathcal{S}_6$. This is because $\mathcal{A}$ can only invoke token TK_i on the commitment com_{b_i}, for which is can only open in a single specific way. $\square$

Finally, note that hybrid H_6 is identical to the simulation described above, which concludes the proof.

■

■

5.4.2.1 Relaxing to One-Way Functions

In our construction we rely on one-way permutations for a non-interactive perfectly binding commitment scheme. Recall that, the $\mathsf{TK}^{\mathsf{Com}}$ on input $(\cdot, 0)$ is required to output a commitment to the challenge e and else commitments to the s^i_0, s^i_1's values. To relax this assumption to one-way functions, we instead need to rely on the two-message Naor's statistically binding commitment scheme [Nao91] where the receiver sends the first message. Instead of communicating this message to the sender, the receiver directly feeds it to the token as input. More precisely, let $\widehat{\mathsf{Com}}(m; r, R)$ denote the honest committer's strategy function that responds according to Naor's commitment with input message m and random tape r, where the receiver's first message is R. We make the following modification and incorporate the following functionality: On input $(\mathsf{tcom}_{b_i}, i, R_0, R_1)$ proceed as follows:

- If $i = 0$: compute $V = \mathsf{PRF}'_{\gamma'}(0^\kappa \| R_0 \| R_1)$, parse V as $e\|r$ and output $\mathsf{com}_e \leftarrow \widehat{\mathsf{Com}}(e; r, R_0)$.
- Otherwise: compute $V = \mathsf{PRF}'_{\gamma'}(\mathsf{tcom}_{b_i} \| i \| R_0 \| R_1)$, parse V as $s^i_0 \| s^i_1 \| r_0 \| r_1$, compute $\mathsf{com}_{s^i_b} \leftarrow \widehat{\mathsf{Com}}(s^i_b; r_b, R_b)$ for $b = \{0, 1\}$, and output $\mathsf{com}_{s^i_0}, \mathsf{com}_{s^i_1}$.

Finally, along with the first message sent by the receiver to the sender, it produces R_0, R_1, the first messages corresponding to the commitments made so that the sender can reconstruct the values being committed to (using the same PRF function). We note that two issues arise when proving security using the modified token's functionality.

1. The first messages for the Naor commitment used when querying the token might not be the same as the one produced in the first message by the receiver. In this case, by the pseudorandomness property of the PRF it follows that the values for these commitments computed by the sender will be independent of the commitments received from the token by the receiver. Hence, the statistically-hiding property of the values used by the sender will not be violated.

2. The binding property of the commitment scheme is only statistical (as opposed to perfect). This will affect the failure probability of the simulator when extracting the sender's input only by a negligible amount and can be bounded overall by incorporating a union bound argument.

We refer to the full version for the reusability of tokens.

5.5 Two-Round Token-Based GUC Oblivious Transfer

In this section we present our main protocol that implements GUC OT in two rounds. We first construct a three-round protocol and then show, how to obtain a two-round protocol by exchanging tokens just once in a setup phase. Recall that the counter example to the [GIS+10] protocol shows that directly extracting the sender's inputs does not necessarily allow us to extract the sender's inputs correctly, as the tokens can behave maliciously. Inspired by the recently developed protocol from [ORS15] we consider a new approach here for which the sender's inputs are extracted directly by monitoring the queries it makes to the PRF tokens and using additional checks to ensure that the sender's inputs can be verified.

Protocol intuition. As a warmup consider the following sender's algorithm that first chooses two random strings x_0 and x_1 and computes their shares $[x_b] = (x_b^1, \ldots, x_b^{2\kappa})$ for $b \in \{0,1\}$ using the $\kappa + 1$-out-of-2κ Shamir secret-sharing scheme. Next, for each $b \in \{0,1\}$, the sender commits to $[x_b]$ by first generating two vectors α_b and β_b such that $\alpha_b + \beta_b = [x_b]$, and then committing to these vectors. Finally, the parties engage in 2κ parallel OT executions where the sender's input to the jth instance are the decommitments to $(\alpha_0[j], \beta_0[j])$ and $(\alpha_1[j], \beta_1[j])$. The sender further sends $(s_0 + x_0, s_1 + x_1)$. Thus, to learn s_b, the receiver needs to learn x_b. For this, it enters the bit b for $\kappa + 1$ or more OT executions and then reconstructs the shares for x_b, followed by reconstructing s_b using these shares. Nevertheless, this reconstruction procedure works only if there is a mechanism that verifies whether the shares are consistent.

To resolve this issue, Ostrovsky et al. made the observation that the Shamir secret-sharing scheme has the property for which there exists a linear function ϕ such that any vector of shares $[x_b]$ is valid if and only if $\phi(x_b) = 0$. Moreover, since the function ϕ is linear, it suffices to check whether $\phi(\alpha_b) + \phi(\beta_b) = 0$. Nevertheless, this check requires from the receiver to know the entire vectors α_b and β_b for its input b. This means it would have to use b as the input to all the 2κ OT executions, which may lead to an input-dependent abort attack. Instead, Ostrovsky et al. introduced a mechanism for checking consistency indirectly via a *cut-and-choose* mechanism. More formally, the sender chooses κ pairs of vectors that add up to $[x_b]$. It is instructive to view them as matrices $A_0, B_0, A_1, B_1 \in \mathbb{Z}_p^{\kappa\times 2\kappa}$ where for every row $i \in [\kappa]$ and $b \in \{0,1\}$, it holds that $A_b[i,\cdot] + B_b[i,\cdot] = [x_b]$. Next, the sender commits to each entry of each matrix separately and sets as input to the jth OT the decommitment information of the entire column $((A_0[\cdot,j], B_0[\cdot,j]), (A_1[\cdot,j], B_1[\cdot,j]))$. Upon receiving the information for a particular column j, the receiver checks if for all i, $A_b[i,j] + B_b[i,j]$ agree on the same value. We refer to this as the *shares consistency check*.

Next, to check the validity of the shares, the sender additionally sends vectors $[z_1^b], \ldots, [z_\kappa^b]$ in the clear along with the sender's message where it commits to the entries of A_0, A_1, B_0 and B_1 such that $[z_i^b]$ is set to $\phi(A_0[i,\cdot])$. Depending on the challenge message, the sender decommits to $A_0[i,\cdot]$ and $A_1[i,\cdot]$ if $c_i = 0$ and $B_0[i,\cdot]$ and $B_1[i,\cdot]$ if $c_i = 1$. If $c_i = 0$, then the receiver checks whether $\phi(A_b[i,\cdot]) = [z_i^b]$, and if $c_i = 1$ it checks whether $\phi(B_b[i,\cdot]) + z_i^b = 0$. This check ensures that except for at most $s \in \omega(\log\kappa)$ of the rows $(A_b[i,\cdot], B_b[i,\cdot])$ satisfy the condition that $\phi(A_b[i,\cdot]) + \phi(B_b[i,\cdot]) = 0$ and for each such row i, $A_b[i,\cdot] + B_b[i,\cdot]$ represents a valid set of shares for both $b = 0$ and $b = 1$. This check is denoted by the *shares validity check*. In the final protocol, the sender sets as input in the jth parallel OT, the decommitment to the entire jth columns of A_0 and B_0 corresponding to the receiver's input 0 and A_1 and B_1 for input 1. Upon receiving the decommitment information on input b_j, the receiver considers a column "good" only if $A_{b_j}[i,j] + B_{b_j}[i,j]$ add up to the same value for every i. Using another cut-and-choose mechanism, the receiver ensures that there are sufficiently many good columns which consequently prevents any input-independent behavior. We refer this to the shares-validity check.

Our oblivious transfer protocol. We obtain a two-round oblivious transfer protocol as follows. The receiver commits to its input bits $b_1, \ldots, b_{2\kappa}$ and the challenge bits for the share consistency check $c_1, \ldots, c_\kappa$ using the PRF tokens. Then, the sender sends all the commitments *a la* [ORS15] and $2\kappa + \kappa$ tokens, where the first 2κ tokens provide the decommitments to the columns, and the second set of κ tokens give the decommitments of the rows for the shares consistency check. The simulator now extracts the sender's inputs by retrieving its queries and we are able to show that there cannot be any input dependent behavior of the token if it passes both the shares consistency check and the shares validity check. See Figure 5.4 for the protocol overview. In Section 5.5.1 we discuss how to obtain a two-round two-party computation using our OT protocol.

We now describe our protocol $\Pi_{\mathrm{OT}}^{\mathrm{UC}}$ with sender P_1 and receiver P_2 using the following building blocks: let (1) Com be a non-interactive perfectly binding commitment scheme, (2) let $\mathscr{S} = (\mathsf{Share}, \mathsf{Recon})$ be a $(\kappa+1)$-out-of-2κ Shamir secret-sharing scheme over $\mathbb{Z}_p$, together with a linear map $\phi : \mathbb{Z}_p^{2\kappa} \to \mathbb{Z}_p^{\kappa-1}$ such that $\phi(v) = 0$ iff v is a valid sharing of some secret, (3) F, F' be two families of pseudorandom functions that map $\{0,1\}^{5\kappa} \to \{0,1\}^\kappa$ and $\{0,1\}^\kappa \to \{0,1\}^{p(\kappa)}$, respectively (4) H denote a hardcore bit function and (5) $\mathsf{Ext} : \{0,1\}^{5\kappa} \times \{0,1\}^d \to \{0,1\}$ denote a randomness extractor where the source has length 5κ and the seed has length d. See Protocol 2

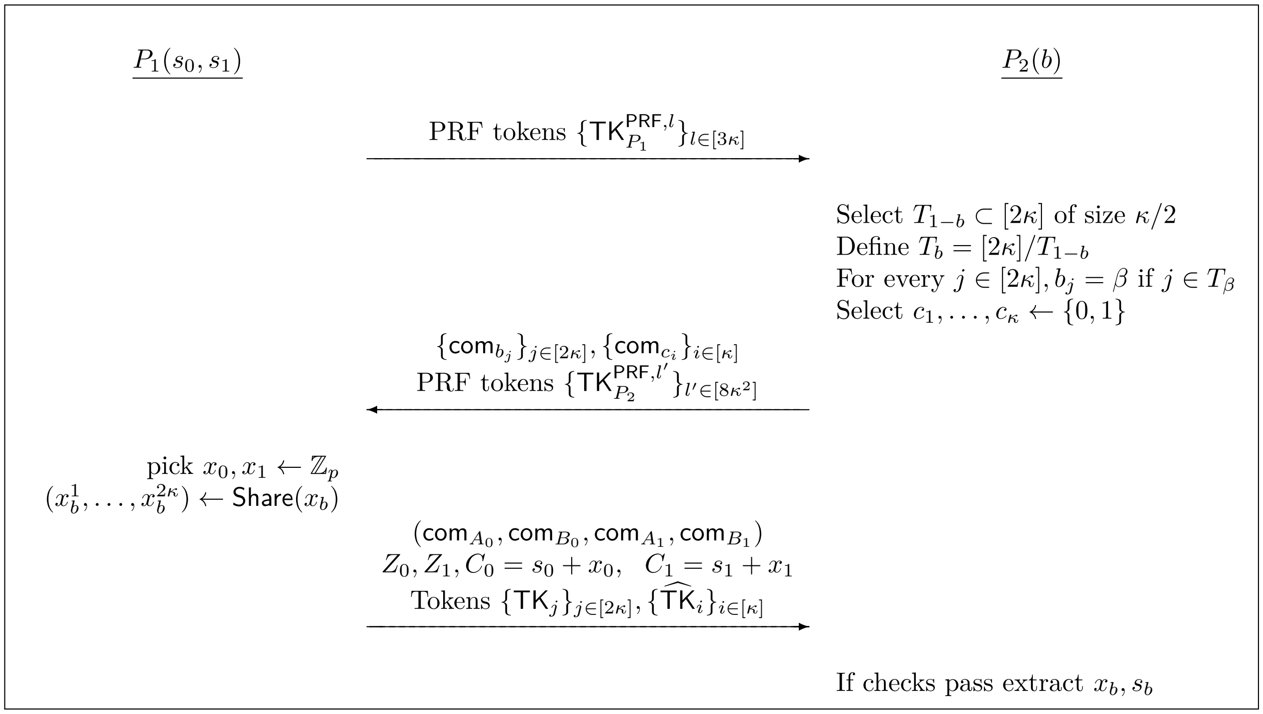

$P_1(s_0, s_1)$
$P_2(b)$
PRF tokens $\{\mathsf{TK}_{P_1}^{\mathsf{PRF},l}\}_{l\in[3\kappa]}$
Select $T_{1-b} \subset [2\kappa]$ of size $\kappa/2$
Define $T_b = [2\kappa]/T_{1-b}$
For every $j \in [2\kappa], b_j = \beta$ if $j \in T_\beta$
Select $c_1, \ldots, c_\kappa \leftarrow \{0,1\}$
$\{\mathsf{com}_{b_j}\}_{j\in[2\kappa]}, \{\mathsf{com}_{c_i}\}_{i\in[\kappa]}$
PRF tokens $\{\mathsf{TK}_{P_2}^{\mathsf{PRF},l'}\}_{l'\in[8\kappa^2]}$
pick $x_0, x_1 \leftarrow \mathbb{Z}_p$
$(x_b^1, \ldots, x_b^{2\kappa}) \leftarrow \mathsf{Share}(x_b)$
$(\mathsf{com}_{A_0}, \mathsf{com}_{B_0}, \mathsf{com}_{A_1}, \mathsf{com}_{B_1})$
$Z_0, Z_1, C_0 = s_0 + x_0, \quad C_1 = s_1 + x_1$
Tokens $\{\mathsf{TK}_j\}_{j\in[2\kappa]}, \{\widehat{\mathsf{TK}}_i\}_{i\in[\kappa]}$
If checks pass extract x_b, s_b

Figure 5.4: [HPV16] A high-level diagram of $\Pi_{\mathrm{UC}}^{\mathrm{OT}}$.

for the complete description.

Protocol 2. Protocol $\Pi_{\mathrm{UC}}^{\mathrm{OT}}$ - GUC OT with stateless tokens.

- **Inputs:** P_1 holds two strings $s_0, s_1 \in \{0,1\}^\kappa$ and P_2 holds a bit b. The common input is sid.
- **The protocol:**

1. $\boldsymbol{P_1 \to P_2}$: P_1 chooses 3κ random PRF keys $\{\gamma_l\}_{[l\in 3\kappa]}$ for family F. Let $\mathsf{PRF}_{\gamma_l} : \{0,1\}^{5\kappa} \to \{0,1\}^\kappa$ denote the pseudorandom function. P_1 creates token $\mathsf{TK}_{P_1}^{\mathsf{PRF},l}$ sending $(\mathsf{Create}, \mathsf{sid}, P_1, P_2, \mathsf{mid}_l, M_1)$ to $\mathcal{F}_{\text{gWRAP}}$ where M_1 is the functionality of the token that on input $(\overline{\mathsf{sid}}, x)$ outputs $\mathsf{PRF}_{\gamma_l}(x)$ for all $l \in [3\kappa]$; For the case where $\overline{\mathsf{sid}} \neq \mathsf{sid}$ the token aborts;
2. $\boldsymbol{P_2 \to P_1}$: P_2 selects a random subset $T_{1-b} \subset [2\kappa]$ of size $\kappa/2$ and defines $T_b = [2\kappa]/T_{1-b}$. For every $j \in [2\kappa]$, P_2 sets $b_j = \beta$ if $j \in T_\beta$. P_2 samples uniformly at random $c_1, \ldots, c_\kappa \leftarrow \{0,1\}$. Finally, P_2 sends
   a) $(\{\mathsf{com}_{b_j}\}_{j\in[2\kappa]}, \{\mathsf{com}_{c_i}\}_{i\in[\kappa]})$ to P_1 where
   $$\forall\ j \in [2\kappa], i \in [\kappa] \quad \mathsf{com}_{b_j} = (\mathsf{Ext}(u_j) + b_j, v_j) \quad \text{and} \quad \mathsf{com}_{c_i} = (\mathsf{Ext}(u'_i) + c_i, v'_i)$$
   $u_j, u'_i \leftarrow \{0,1\}^{5\kappa}$ and v_j, v'_i are obtained by sending respectively $(\mathsf{Run}, \mathsf{sid}, \mathsf{mid}_j, u_j)$ and $(\mathsf{Run}, \mathsf{sid}, \mathsf{mid}_{2\kappa+i}, u'_i)$.
   b) P_2 generates the tokens $\{\mathsf{TK}_{P_2}^{\mathsf{PRF},l'}\}_{l'\in[8\kappa^2]}$ which are analogous to the PRF tokens $\{\mathsf{TK}_{P_1}^{\mathsf{PRF},l}\}_{l\in[3\kappa]}$ by sending $(\mathsf{Create}, \mathsf{sid}, P_2, P_1, \mathsf{mid}_{l'}, M_2)$ to $\mathcal{F}_{\text{gWRAP}}$ for all $l' \in [8\kappa^2]$.
3. $\boldsymbol{P_1 \to P_2}$: P_1 picks two random strings $x_0, x_1 \leftarrow \mathbb{Z}_p$ and secret shares them using $\mathscr{S}$. In particular, P_1 computes $[x_b] = (x_b^1, \ldots, x_b^{2\kappa}) \leftarrow \mathsf{Share}(x_b)$ for $b \in \{0,1\}$. P_1 commits to the

shares $[x_0], [x_1]$ as follows. It picks random matrices $A_0, B_0 \leftarrow \mathbb{Z}_p^{\kappa\times 2\kappa}$ and $A_1, B_1 \leftarrow \mathbb{Z}_p^{\kappa\times 2\kappa}$ such that $\forall i \in [\kappa]$:

$$A_0[i,\cdot] + B_0[i,\cdot] = [x_0], \quad A_1[i,\cdot] + B_1[i,\cdot] = [x_1].$$

P_1 computes two matrices $Z_0, Z_1 \in \mathbb{Z}_p^{\kappa\times\kappa-1}$ and sends them in the clear such that:

$$Z_0[i,\cdot] = \phi(A_0[i,\cdot]), Z_1[i,\cdot] = \phi(A_1[i,\cdot]).$$

P_1 sends:

a) Matrices $(\mathsf{com}_{A_0}, \mathsf{com}_{B_0}, \mathsf{com}_{A_1}, \mathsf{com}_{B_1})$ to P_2, where,

$$\forall\ i \in [\kappa],\ j \in [2\kappa], \beta \in \{0,1\} \qquad \begin{aligned} \mathsf{com}_{A_\beta[i,j]} &= (\mathsf{Ext}(u^{A_\beta[i,j]} + A_\beta[i,j], v^{A_\beta[i,j]}) \\ \mathsf{com}_{B_\beta[i,j]} &= (\mathsf{Ext}(u^{B_\beta[i,j]} + B_\beta[i,j], v^{B_\beta[i,j]}) \end{aligned}$$

where $(u^{A_\beta[i,j]}, u^{B_\beta[i,j]}) \leftarrow \{0,1\}^{5\kappa}$ and $(v^{A_\beta[i,j]}, v^{B_\beta[i,j]})$ are obtained by sending $(\mathsf{Run}, \mathsf{sid}, \mathsf{mid}_{[i,j,\beta]}, u^{A_\beta[i,j]})$ and $(\mathsf{Run}, \mathsf{sid}, \mathsf{mid}_{2\kappa^2+[i,j,\beta]}, u^{B_\beta[i,j]})$, respectively, to the token $\mathsf{TK}_{P_2}^{\mathsf{PRF},[i,j,\beta]}$ where $[i,j,\beta]$ is an encoding of the indices i, j, β into an integer in $[2\kappa^2]$.

b) $C_0 = s_0 + x_0$ and $C_1 = s_1 + x_1$ to P_2.

c) For all $j \in [2\kappa]$, P_1 creates a token TK_j sending $(\mathsf{Create}, \mathsf{sid}, P_1, P_2, \mathsf{mid}_{3\kappa+j}, M_3)$ to $\mathcal{F}_{\text{gWRAP}}$ where M_3 is the functionality that on input $(\overline{\mathsf{sid}}, b_j, \mathsf{decom}_{b_j})$, aborts if $\overline{\mathsf{sid}} \neq \mathsf{sid}$ or if decom_{b_j} is not verified correctly. Otherwise it outputs $(A_{b_j}[\cdot,j], \mathsf{decom}_{A_{b_j}[\cdot,j]}, B_{b_j}[\cdot,j], \mathsf{decom}_{B_{b_j}[\cdot,j]})$.

d) For all $i \in [\kappa]$, P_1 creates a token $\widehat{\mathsf{TK}}_i$ sending $(\mathsf{Create}, \mathsf{sid}, P_1, P_2, \mathsf{mid}_{5\kappa+i}, M_4)$ to $\mathcal{F}_{\text{gWRAP}}$ where M_4 is the functionality that on input $(\overline{\mathsf{sid}}, c_i, \mathsf{decom}_{c_i})$ aborts if $\overline{\mathsf{sid}} \neq \mathsf{sid}$ or if decom_{c_i} is not verified correctly. Otherwise it outputs,

$$\begin{aligned} &(A_0[i,\cdot], \mathsf{decom}_{A_0[i,\cdot]}, A_1[i,\cdot], \mathsf{decom}_{A_1[i,\cdot]}), \text{ if } c = 0 \\ &(B_0[i,\cdot], \mathsf{decom}_{B_0[i,\cdot]}, B_1[i,\cdot], \mathsf{decom}_{B_1[i,\cdot]}), \text{ if } c = 1 \end{aligned}$$

4. **Output Phase:**

For all $j \in [2\kappa]$, P_2 sends $(\mathsf{Run}, \mathsf{sid}, \mathsf{mid}_{3\kappa+j}, (b_j, \mathsf{decom}_{b_j}))$ and receives

$$(A_{b_j}[\cdot,j], \mathsf{decom}_{A_{b_j}[\cdot,j]}, B_{b_j}[\cdot,j], \mathsf{decom}_{B_{b_j}[\cdot,j]}).$$

For all $i \in [\kappa]$, P_2 sends $(\mathsf{Run}, \mathsf{sid}, \mathsf{mid}_{5\kappa+i}, (c_i, \mathsf{decom}_{c_i}))$ and receives

$$(A_0[\cdot,i], A_1[\cdot,i]) \text{ or } (B_0[\cdot,i], B_1[\cdot,i]).$$

a) **Shares Validity Check Phase:** For all $i \in [\kappa]$, if $c_i = 0$ check that $Z_0[i,\cdot] = \phi(A_0[i,\cdot])$ and $Z_1[i,\cdot] = \phi(A_1[i,\cdot])$. Otherwise, if $c_i = 1$ check that $\phi(B_0[i,\cdot]) + Z_0[i,\cdot] = 0$ and $\phi(B_1[i,\cdot]) + Z_1[i,\cdot] = 0$. If the tokens do not abort and all the checks pass, the receiver proceeds to the next phase.

b) **Shares Consistency Check Phase:** For each $b \in \{0,1\}$, P_2 randomly chooses a set T_b for which $b_j = b$ of $\kappa/2$ coordinates. For each $j \in T_b$, P_2 checks that there exists a unique x_b^j such that $A_b[i,j] + B_b[i,j] = x_b^j$ for all $i \in [\kappa]$. If so, x_b^j is marked as consistent. If the tokens do not abort and all the shares obtained in this phase are consistent, P_2 proceeds to the reconstruction phase. Else it abort.

c) **Output Reconstruction:** For $j \in [2\kappa]/T_{1-b}$, if there exists a unique x_b^j such that $A_b[i,j] + B_b[i,j] = x_b^j$, mark share j as a good column. If R obtains less than $\kappa+1$ good shares, it aborts. Otherwise, let $x_b^{j_1}, \ldots, x_b^{j_{\kappa+1}}$ be any set of $\kappa+1$ consistent shares. P_2 computes $x_b \leftarrow \mathsf{Recon}(x_b^{j_1}, \ldots, x_b^{j_{\kappa+1}})$ and outputs $s_b = C_b + x_b$.

Next, we prove the following theorem,

Theorem 5.5.1. Assume the existence of one-way functions, then protocol $\Pi_{\mathrm{UC}}^{\mathrm{OT}}$ GUC realizes $\mathcal{F}_{\mathrm{OT}}$ in the $\mathcal{F}_{\mathrm{gWRAP}}$-hybrid.

Proof overview. On a high-level, when the sender is corrupted our simulation proceeds analogously to the simulation from [ORS15] where the simulator generates the view of the malicious sender by honestly generating the receiver's messages and then extracting all the values committed to by the sender. Nevertheless, while in [ORS15] the authors rely on extractable commitments and extract the sender's inputs via rewinding, we directly extract its inputs by retrieving the queries made by the malicious sender to the $\{\mathsf{TK}_{P_2}^{\mathsf{PRF},i}\}_i$ tokens. The proof of correctness follows analogously. More explicitly, the share consistency check ensures that for any particular column that the receiver obtains, if the sum of the values agree on the same bit, then the receiver extracts the correct share of $[x_b]$ with high probability. Note that it suffices for the receiver to obtain $\kappa+1$ good columns for its input b to extract enough shares to reconstruct x_b since the shares can be checked for validity. Namely, the receiver chooses $\kappa/2$ indices T_b and sets its input for these OT executions as b. For the rest of the OT executions, the receiver sets its input as $1-b$. Denote this set of indices by T_{1-b}. Then, upon receiving the sender's response to its challenge and the OT responses, the receiver first performs the shares consistency check. If this check passes, it performs the shares validity check for all columns, both with indices in T_{1-b} and for the indices in a random subset of size $\kappa/2$ within T_b. If one of these checks do not pass, the receiver aborts. If both checks pass, it holds with high probability that the decommitment information for $b=0$ and $b=1$ are correct in all but $s \in \omega(\log n)$ indices. Therefore, the receiver will extract $[x_b]$ successfully both when its input $b=0$ and $b=1$. Furthermore, it is ensured that if the two checks performed by the receiver pass, then a simulator can extract both x_0 and x_1 correctly by simply extracting the sender's input to the OT protocol and following the receiver's strategy to extract.

On the other hand, when the receiver is corrupted, our simulation proceeds analogous to the simulation in [ORS15] where the simulator generates the view of the malicious receiver by first extracting the receiver's input b and then obtaining s_b from the ideal functionality. It then completes the execution following the honest sender's code with (s_0, s_1), where s_{1-b} is set to random. Moreover, while in [ORS15] the authors rely on a special type of interactive commitment that allows the extraction of the receiver's input via rewinding, we instead extract this input directly by retrieving the queries made by the malicious receiver to the $\{\mathsf{TK}_{P_1}^{\mathsf{PRF},l}\}_{l\in[3\kappa]}$ tokens. The proof of correctness follows analogously. Informally, the idea is to show that the receiver can learn $\kappa+1$ or more shares for either x_0 or x_1 but not both. In other words there exists a bit b for which a corrupted receiver can learn at most κ shares relative to s_{1-b}. Thus, by replacing s_{1-b} with a random string, it follows from the secret-sharing property that obtaining at most κ shares keeps s_{1-b} information theoretically hidden.

The next claim establishes that the commitments made by the parties are statistically hiding. We remark that this claim is analogous to Claim 20 from [GIS+10]. For completeness, we present it below.

Lemma 5.5.1. For any $i \in [\kappa]$, let D_b denote the distribution obtained by sampling a random com_{b_i} with $b_i = b$. Then D_0 and D_1 are $2^{-\kappa+1}$-close.

Proof: Informally, the proof follows from the fact that u_i has high min-entropy conditioned on v_i and therefore $(\mathsf{Ext}(u_i, h), h)$ hides u_i information theoretically as it is statistically close to the uniform distribution. More formally, consider a possibly maliciously generated token M_1 that incorporates an *arbitrary functionality* from 5κ bits to κ. It is possible to think of M_1 as a function even if the token is stateful since we only consider the min-entropy of the input with respect to the output when M_1 is invoked from the same state.

Let S_v denote the subset of $\{0,1\}^{5\kappa}$ that contains all $x \in \{0,1\}^{5k}$ such that $M_1(x) = v$. First, we claim that for a randomly chosen $x \leftarrow \{0,1\}^{5\kappa}$, $S_{M_1(x)}$ is of size at least $2^{3\kappa}$ with probability at least $1 - 2^{-\kappa}$. Towards proving this we calculate the number of x's for which $|S_{M_1(x)}| < 2^{3\kappa}$ and denote such an x by *bad*. Now, since there are at most 2^k possible values that M_1 may output, then the number of bad x's is:

$$\sum_{v:|S_v|<2^{3\kappa}} |S_v| < 2^{\kappa} \times 2^{3\kappa} = 2^{4\kappa}.$$

Therefore, the probability that a uniformly chosen x is bad is at most $2^{4k}/2^{5k} = 2^{-k}$. Let U and V denote random variables such that V is the response of M_1 on U. It now holds that

$$\Pr[u \leftarrow \{0,1\}^{5\kappa} : H_\infty(U|V = M_1(u)) \geq 3\kappa] > 1 - 2^{-\kappa}.$$

In other words, the min-entropy of U is at least 3κ with very high probability. Now, whenever this is the case, using the Leftover Hash Lemma (cf. Definition 5.3.1) with $\epsilon = 2^{-\kappa}$, $m = 1$ and $k = 3\kappa$ implies that $(\mathsf{Ext}(U, h), h)$ is $2^{-\kappa}$-close to the uniform distribution. Combining the facts that $\mathsf{com}_b = (\mathsf{Ext}(U, h) + b, h, V)$ and that U has high min-entropy at least with probability $1 - 2^{-\kappa}$, we obtain that D_0 and D_1 are $2^{-\kappa} + 2^{-\kappa}$-close. □

We continue with the complete proof.

Proof. Let $\mathcal{A}$ be a malicious PPT real adversary attacking protocol $\Pi^{\mathrm{UC}}_{\mathrm{OT}}$ in the $\mathcal{F}_{\mathrm{gWRAP}}$-hybrid model. We construct an ideal adversary $\mathcal{S}$ with access to $\mathcal{F}_{\mathrm{OT}}$ which simulates a real execution of $\Pi^{\mathrm{UC}}_{\mathrm{OT}}$ with $\mathcal{A}$ such that no environment $\mathcal{Z}$ can distinguish the ideal process with $\mathcal{S}$ and $\mathcal{F}_{\mathrm{OT}}$ from a real execution of $\Pi^{\mathrm{UC}}_{\mathrm{OT}}$ with $\mathcal{A}$. $\mathcal{S}$ starts by invoking a copy of $\mathcal{A}$ and running a simulated interaction of $\mathcal{A}$ with environment $\mathcal{Z}$, emulating the honest party. We describe the actions of $\mathcal{S}$ for every corruption case.

Simulating the communication with $\mathcal{Z}$: Every message that $\mathcal{S}$ receives from $\mathcal{Z}$ it internally feeds to $\mathcal{A}$ and every output written by $\mathcal{A}$ is relayed back to $\mathcal{Z}$.

In case the adversary $\mathcal{A}$ issues a transfer query $(\mathsf{transfer}, \cdot)$, $\mathcal{S}$ relays the query to the $\mathcal{F}_{\mathrm{gWRAP}}$.

Simulating the corrupted P_1. We begin by describing our simulation:

1. $\mathcal{A}$ communicates with the functionality $\mathcal{F}_{\text{gWRAP}}$ on behalf of the corrupted parties by sending create messages $\{(\mathsf{Create}, \mathsf{sid}, P_1, P_2, \mathsf{mid}_l, M_1)\}_{l\in[3\kappa]}$. Then $\mathcal{F}_{\text{gWRAP}}$ forwards these tokens to the honest parties by sending receipt messages $\{(\mathsf{Receipt}, \mathsf{sid}, P_1, P_2, \mathsf{mid}_l, M_1)\}_{l\in[3\kappa]}$.

2. Upon receiving acknowledgement messages $\{(\mathsf{Receipt}, \mathsf{sid}, P_1, P_2, \mathsf{mid}_l, M_1)\}_{l\in[3\kappa]}$ that all $[3\kappa]$ tokens have been created by $\mathcal{A}$, $\mathcal{S}$ communicates with the functionality $\mathcal{F}_{\text{gWRAP}}$ on behalf of the honest parties by sending create messages $\{(\mathsf{Create}, \mathsf{sid}, P_2, P_1, \mathsf{mid}_{l'}, M_2)\}_{l'\in[8\kappa^2]}$, where the code M_2 implements truly random functions (that is, M_2 is encoded with a lookup table that includes some polynomial number of queries bounded by the running time of the adversary). Then, the functionality $\mathcal{F}_{\text{gWRAP}}$ forwards receipt messages $\{(\mathsf{Receipt}, \mathsf{sid}, P_2, P_1, \mathsf{mid}_{l'}, M_2)\}_{l'\in[8\kappa^2]}$ to $\mathcal{A}$. For each query $u \in \{0,1\}^{5\kappa}$ made by $\mathcal{A}$ to the tokens $\mathsf{TK}_{P_2}^{\mathsf{PRF},l'}$, functionality $\mathcal{F}_{\text{gWRAP}}$ runs M_2 on that query and returns a random v from $\{0,1\}^{\kappa}$.

3. $\mathcal{S}$ generates the first message by following the code of the honest receiver with input $b = 0$.

4. Upon receiving the second message from $\mathcal{A}$, i.e. commitments $(\mathsf{com}_{A_0}, \mathsf{com}_{B_0}, \mathsf{com}_{A_1}, \mathsf{com}_{B_1})$ and (C_0, C_1), it completes the execution by following the honest receiver's code.

5. Next, $\mathcal{S}$ tries to extract s_0 and s_1. For this, it first extracts matrices A_0, B_0, A_1, B_1 from the respective commitments as described in the simulation for the proof of Π_{OT}. More precisely, given any commitment β, v, it first checks if there exists a query/answer pair (u, v) that has already been recorded by $\mathcal{F}_{\text{gWRAP}}$ with respect to that token by sending a retrieve message $(\mathsf{retreive}, \mathsf{sid})$ to $\mathcal{F}_{\text{gWRAP}}$ which returns the list $\mathcal{Q}_{\mathsf{sid}}$ of illegitimate queries. If there exists such a query then the simulator sets the decommitted value to be $\beta + \mathsf{Ext}(u)$, and $\perp$ otherwise. Next, to extract s_b, $\mathcal{S}$ proceeds as follows: For every $i \in [\kappa]$, it computes $A_b[i,j] + B_b[i,j]$ for all $j \in [2\kappa]$ and marks that column j good if they all agree to the same value, say, γ_j. If it finds more than $\kappa + 1$ good columns, it reconstructs the secret x_b by using share reconstruction algorithm on $\{\gamma_j\}_{j\in\mathsf{good}}$. Otherwise, it sets x_b to $\perp$.

6. $\mathcal{S}$ computes $s_0 = C_0 + x_0$ and $s_1 = C_1 + x_1$ and sends (s_0, s_1) to the trusted party that computes $\mathcal{F}_{\text{OT}}$ and halts, outputting whatever $\mathcal{A}$ does.

Next, we prove the correctness of our simulation.

Hybrid H_0: In this hybrid game there is no trusted party that computes functionality $\mathcal{F}_{\text{OT}}$. Instead, we define a simulator $\mathcal{S}_0$ that receives the real input of the receiver and internally emulates the protocol $\Pi_{\text{OT}}^{\text{UC}}$ with the adversary $\mathcal{A}$ by simply following the honest receiver's strategy. Finally, the output of the receiver in the internal emulation is just sent to the external honest receiver (as part of the protocol Π_{H_0}) that outputs it as its output. The execution in this hybrid proceeds identically to the real execution.

Hybrids $\mathrm{H}_{1,0}\ldots,\mathrm{H}_{1,8\kappa^2}$: We define a collection of hybrid executions such that for every $l' \in [8\kappa^2]$ hybrid $\mathrm{H}_{1,\mathrm{l}'}$ is defined as follows. We modify the code of token $\mathsf{TK}_{P_2}^{\mathsf{PRF},l'}$ by replacing the function $\mathsf{PRF}_{\gamma_{l'}}$ with a truly random function $f_{l'}$. In particular, given a query u the token responds with a randomly chosen κ bit string v, rather than running the original code of M_2. We maintain a list of $\mathcal{A}$'s queries and responses so that repeated queries will be consistently answered. It is simple to verify that the adversary's view in every two consecutive hybrid executions is computationally indistinguishable due to the security of the pseudorandom function embedded within $\mathsf{TK}_{P_2}^{\mathsf{PRF},l'}$. Moreover, since the PRF key is hidden from the sender, it follows from the pseudorandomness property that the views in every two consecutive hybrid are computationally indistinguishable. As in the previous hybrid, the simulator hands the output of the receiver in the internal emulation to the external receiver as part of the protocol $\Pi_{\mathrm{H}_{1,\mathrm{l}'}}$. More formally, we have the following claim,

Claim 13. For every $l' \in [8\kappa^2]$, $\mathrm{Hybrid}_{1,l'-1} \overset{\mathrm{c}}{\approx} \mathrm{Hybrid}_{1,l'}$.

Hybrids $\mathrm{H}_{2,0}\ldots,\mathrm{H}_{2,8\kappa^2}$: This sequence of hybrids executions is identical to hybrid $\mathrm{H}_{1,8\kappa^2}$ except that here $\mathcal{S}_2$ aborts if two queries made by $\mathcal{A}$ to the token $\mathsf{TK}_{P_2}^{\mathsf{PRF},l'}$ results in the same response. Using a proof analogous to Lemma 9, we obtain the following claim.

Claim 14. For every $l' \in [8\kappa^2]$, $\mathrm{Hybrid}_{2,l'-1} \overset{\mathrm{s}}{\approx} \mathrm{Hybrid}_{2,l'}$.

Hybrid H_3: In this hybrid, $\mathcal{S}_3$ proceeds identically to $\mathcal{S}_{2,8\kappa^2}$ using the honest receiver's input b with the exception that it does not report the output of the receiver as what is computed in the emulation by the simulator. Instead, $\mathcal{S}_3$ follows the code of the actual simulator to extract (s_0, s_1) and sets the receiver's output as s_b. Note that the view of the adversary is identical in both hybrids $\mathrm{H}_{2,8\kappa^2}$ and H_3. Therefore, to prove the indistinguishability of the joint output distribution, it suffices to show that the output of the honest receiver is the same. On a high-level, this will follow from the fact that if the honest receiver does not abort then the two checks performed by the receiver, namely, the shares validity check and the shares consistency check were successful, which would imply that there are at least $\kappa+1$ good columns from which the simulator can extract the shares. Finally, we conclude that the reconstruction performed by the honest receiver and the simulator will yield the same value for s_b.

More formally, we argue indistinguishability conditioned on when the two consistency checks pass in the execution emulated by the simulator (in the event at least one of them do not pass, the receiver aborts and indistinguishability directly holds). Then, the following hold for any $s \in \omega(\log n)$:

Step 1: Since the shares validity check passed, following a standard cut-and-choose argument, it holds except with probability $2^{-O(s)}$ that there are at least $\kappa - s$ rows for which $\phi(A_b[i,\cdot]) + \phi(B_b[i,\cdot]) = 0$. In fact, it suffices if this holds at least for one row, say i^*. For $b \in \{0,1\}$, let the secret corresponding to $A_b[i^*,\cdot] + B_b[i^*,\cdot]$ be $\tilde{s}_b$.

Step 2: If for any column $j \in [2\kappa]$ and $b \in \{0,1\}$ there exists a value γ_j such that for all $i \in [\kappa]$

$$\gamma_b[j] = A_b[i,j] + B_b[i,j],$$

then, combining with Step 1, we can conclude that $\gamma_b[j] = A_b[i^*,j]+B_b[i^*,j]$. Furthermore, if either the receiver or the simulator tries to extract the share corresponding to that column it will extract $\gamma_b[j]$ since the commitments made by the sender are binding. Therefore, we can conclude that if either the receiver or the simulator tries to reconstruct the secret for any $b \in \{0,1\}$, it will reconstruct only with shares in $\{\gamma_b[j]\}_{j \in J}$ which implies that they reconstruct only $\tilde{s}_b$.

Step 3: Now, since the shares consistency check passed, following another cut-and-choose argument, it holds except with probability $2^{-O(s)}$ that there is a set J of at least $2\kappa - s$ columns such that for any $j \in J$ the tokens do not abort on a valid input from the receiver and yield consistent values for both $b_j = 0$ and $b_j = 1$. This means that if the honest receiver selects $3\kappa/4$ columns with input as its real input b, the receiver is guaranteed to find at least $\kappa + 1$ indices in J. Furthermore, there will be $\kappa + 1$ columns in J for both inputs for the simulator to extract and when either of them extract they can only extract $\tilde{s}_b$.

Then to prove indistinguishability in this hybrid, it suffices to prove that the simulator reconstructs s_b if and only if the receiver extracts s_b and this follows directly from Step 3 in the proceeding argument, since there is a unique value $\tilde{s}_b$ that either of them can reconstruct and they will reconstruct that value with probability $1-2^{-O(s)}$ if the two checks pass. As the checks are independent of the real input of the receiver, indistinguishability of the hybrids follow.

Claim 15. $\text{Hybrid}_{2,8\kappa^2} \overset{\text{s}}{\approx} \text{Hybrid}_3$

Hybrid H_4: In this hybrid, $\mathcal{S}_4$ proceeds identically to $\mathcal{S}_3$ with the exception that the simulator sets the receiver's input in the main execution as 0 instead of the real input b. Finally, it reconstructs s_b and sets that as the honest receiver's output. It follows from Lemma 5.5.1 that the output of H_3 and H_4 are statistically-close. Therefore, we have the following claim,

Claim 16. $\text{Hybrid}_3 \overset{\text{s}}{\approx} \text{Hybrid}_4$

Hybrid H_5: In this hybrid, we consider the simulation. Observe that our simulator proceeds identically to the simulation with $\mathcal{S}_4$ with the exception that it communicates with $\mathcal{F}_{\text{gWRAP}}$ instead of creating/sending the tokens by itself and further it feeds the extracted values s_0 and s_1 to the ideal functionality while $\mathcal{S}_4$ instead just outputs s_b. Furthermore, the ideal simulator sends (s_0, s_1) to the $\mathcal{F}_{\text{OT}}$ functionality. It follows from our simulation that the view of the adversary in H_5 and the ideal execution are identically distributed. Furthermore, for both $b = 0$ and $b = 1$ we know that the value s_b extracted by the simulator and the value output by the honest receiver in the ideal execution are equal. Therefore, we can conclude that the output of H_4 and the ideal execution are identically distributed.

Claim 17. $\text{Hybrid}_4 \approx \text{Hybrid}_5$

Simulating the corrupted P_2. We begin by describing our simulation:

1. $\mathcal{S}$ communicates with $\mathcal{F}_{\text{gWRAP}}$ on behalf of the honest parties by sending create messages $\{(\mathsf{Create}, \mathsf{sid}, P_1, P_2, \mathsf{mid}_l, M_1)\}_{l\in[3\kappa]}$, where the code M_1 implements truly random functions (that is, M_1 is encoded with a lookup table that includes some polynomial number of queries bounded by the running time of the adversary). Then $\mathcal{F}_{\text{gWRAP}}$ forwards these tokens by sending receipt messages $\{(\mathsf{Receipt}, \mathsf{sid}, P_1, P_2, \mathsf{mid}_l, M_1)\}_{l\in[3\kappa]}$ to $\mathcal{A}$. For each query $u \in \{0,1\}^{5\kappa}$ made by $\mathcal{A}$ to the tokens $\mathsf{TK}_{P_1}^{\mathsf{PRF},l}$, functionality $\mathcal{F}_{\text{gWRAP}}$ runs M_1 on that query and returns a random v from $\{0,1\}^{\kappa}$.

2. $\mathcal{A}$ communicates with $\mathcal{F}_{\text{gWRAP}}$ on behalf of the corrupted parties by sending create messages to the functionality $\{(\mathsf{Create}, \mathsf{sid}, P_2, P_1, \mathsf{mid}_{l'}, M_2)\}_{l'\in[8\kappa^2]}$. Then, the functionality $\mathcal{F}_{\text{gWRAP}}$ forwards these tokens to the honest parties by sending receipt messages $\{(\mathsf{Receipt}, \mathsf{sid}, P_2, P_1, \mathsf{mid}_{l'}, M_2)\}_{l'\in[8\kappa^2]}$.

3. Upon receiving acknowledgement messages $\{(\mathsf{Receipt}, \mathsf{sid}, P_2, P_1, \mathsf{mid}_{l'}, M_2)\}_{l'\in[8\kappa^2]}$ that all $[8\kappa^2]$ tokens have been created by $\mathcal{A}$, and upon receiving the first message from $\mathcal{A}$, i.e. the commitments com_{b_j} and com_{c_i} where $i \in [\kappa]$ and $j \in [2\kappa]$, $\mathcal{S}$ tries to extract b by sending a retrieve message $(\mathsf{retreive}, \mathsf{sid})$ to $\mathcal{F}_{\text{gWRAP}}$ which returns the list $\mathcal{Q}_{\mathsf{sid}}$ of illegitimate queries. For this, just as in previous simulations, it first extracts all the b_j values and then sets the receiver's input as that bit that occurs at least $\kappa+1$ times among the b_j's. If no such bit exists, it sets b to be random. Next it sends b to the $\mathcal{F}_{\text{OT}}$ functionality to obtain s_b, and completes the protocol following the honest sender's code with inputs (s_0, s_1) where s_{1-b} is set to random. In particular, it computes $C_b = x_b + s_b$ and sets C_{1-b} to a random string.

Proof. Our proof follows by a sequence of hybrid executions defined below.

Hybrid H_0: In this hybrid game there is no trusted party that computes functionality $\mathcal{F}_{\text{OT}}$. Instead, we define a simulator $\mathcal{S}_0$ that receives the real input of the sender and internally emulates the protocol $\Pi_{\text{OT}}^{\text{UC}}$ with the adversary $\mathcal{A}$ by simply following the honest sender's strategy. Finally, the output of the sender in the internal emulation is just sent to the external honest sender (as part of the protocol Π_{H_0}) that outputs it as its output. The execution in this hybrid proceeds identically to the real execution.

Hybrids $\text{H}_{1,0} \ldots, \text{H}_{1,3\kappa}$: We define a collection of hybrid executions such that for every $l \in [3\kappa]$ hybrid $\text{H}_{1,l}$ is defined as follows. We modify the code of token $\mathsf{TK}_{P_1}^{\mathsf{PRF},l}$ by replacing the function PRF_{γ_l} with a truly random function f_l. In particular, given a query u the token responds with a randomly chosen κ bit string v, rather than running the original code of M_1. We maintain a list of $\mathcal{A}$'s queries and responses so that repeated queries will be consistently answered. In addition, the code of token TK_l (for $l \le 2\kappa$) or $\widehat{\mathsf{TK}}_{l-2\kappa}$ (for $2\kappa+1 \le l \le 3\kappa$) is modified, as now this token does not run a check with respect to the PRF that is embedded within token $\mathsf{TK}_{P_1}^{\mathsf{PRF},l}$ but with respect to the random function f_l. It is simple to verify that the adversary's view in every two consecutive hybrid executions is computationally indistinguishable due to the security of the pseudorandom function PRF_{γ_l}. Moreover, since the PRF key is hidden from the receiver, it follows from the pseudorandomness property that the views in every two consecutive hybrid are computationally indistinguishable. As in the previous hybrid, the simulator hands the output of the sender in the internal emulation to the external receiver as part of the protocol $\Pi_{\mathrm{H}_{1,l}}$. More formally, we have the following claim,

Claim 18. For every $l \in [3\kappa]$, $\mathrm{Hybrid}_{1,l-1} \overset{\mathrm{c}}{\approx} \mathrm{Hybrid}_{1,l}$.

Hybrids $\mathrm{H}_{2,0} \ldots, \mathrm{H}_{2,3\kappa}$: This sequence of hybrids executions is identical to hybrid $\mathrm{H}_{1,3\kappa}$ except that here $\mathcal{S}_2$ aborts if two queries made by $\mathcal{A}$ to the token $\mathsf{TK}_{P_1}^{\mathsf{PRF},l}$ results in the same response. Using a proof analogous to Lemma 9, we obtain the following claim.

Claim 19. For every $l \in [3\kappa]$, $\mathrm{Hybrid}_{2,l-1} \overset{\mathrm{s}}{\approx} \mathrm{Hybrid}_{2,l}$.

Hybrid H_3: In this hybrid execution, simulator $\mathcal{S}_3$ plays the role of the sender as in hybrid $\mathrm{H}_{2,3\kappa}$ except that it extracts the adversary's input bit b as carried out in the simulation by $\mathcal{S}$ and the challenge string c. Clearly, this does not make any difference to the receiver's view which implies that,

Claim 20. $\mathrm{Hybrid}_{2,3\kappa} \overset{\mathrm{s}}{\approx} \mathrm{Hybrid}_3$.

Hybrid H_4: In this hybrid execution, the simulator instead of creating the original tokens $\{\mathsf{TK}_j\}_{j\in[2\kappa]}$, simulator $\mathcal{S}_4$ emulates functionalities $\{\widetilde{\mathsf{TK}}_j\}_{j\in[2\kappa]}$ in the following way. For all $j \in [2\kappa]$, if $\widetilde{\mathsf{TK}}_j$ is queried on $(b_j, \mathsf{decom}_{b_j})$ and decom_{b_j} is verified correctly, $\mathcal{S}_4$ outputs the column

$$(A_{b_j}[\cdot, j], \mathsf{decom}_{A_{b_j}[\cdot,j]}, B_{b_j}[\cdot, j], \mathsf{decom}_{B_{b_j}[\cdot,j]})$$

where b_j is the bit extracted by $\mathcal{S}_4$ as in the prior hybrid. Otherwise, if $\widetilde{\mathsf{TK}}_j$ is queried on $(1 - b_j, \mathsf{decom}_{1-b_j})$ then $\mathcal{S}_4$ outputs $\perp$. Following the same argument as in Claim 10 it follows that the commitments made by the receiver are binding and thus a receiver will not be able to produce decommitments to obtain the value corresponding to $1-b_j$. Therefore, we have the following claim.

Claim 21. $\mathrm{Hybrid}_3 \overset{\mathrm{s}}{\approx} \mathrm{Hybrid}_4$.

Hybrid H_5: In this hybrid execution, instead of creating the original tokens $\{\widehat{\mathsf{TK}}_i\}_{i\in[\kappa]}$, simulator $\mathcal{S}_5$ emulates functionalities $\{\overline{\mathsf{TK}}_i\}_{i\in[\kappa]}$ in the following way. For all $i \in [\kappa]$, if $\overline{\mathsf{TK}}_i$ is queried on $(c_i, \mathsf{decom}_{c_i})$ and decom_{c_i} is verified correctly, $\mathcal{S}_5$ outputs the row

$$\begin{array}{ll} (A_0[i, \cdot], \mathsf{decom}_{A_0[i,\cdot]}, A_1[i, \cdot], \mathsf{decom}_{A_i[i,\cdot]}), & \text{if } c_i = 0 \\ (B_0[i, \cdot], \mathsf{decom}_{B_0[i,\cdot]}, B_1[i, \cdot], \mathsf{decom}_{B_i[i,\cdot]}), & \text{if } c_i = 1 \end{array}$$

where c_i is the bit extracted by $\mathcal{S}_5$ as in the prior hybrid. Otherwise, if $\overline{\mathsf{TK}}_i$ is queried on $(1-c_i, \mathsf{decom}_{1-c_i})$ then $\mathcal{S}_5$ outputs $\perp$. Indistinguishability follows using the same argument as in the previous hybrid. Therefore, we have the following claim.

Claim 22. $\text{Hybrid}_4 \overset{s}{\approx} \text{Hybrid}_5$.

Hybrid H_6: In this hybrid, the simulator $\mathcal{S}_6$ chooses an independent random string $x^* \leftarrow \mathbb{Z}_p$ instead of generating the matrices A_{1-b} and B_{1-b} according to the shares of x_{1-b}. We remark that $C_{1-b} = s_{1-b} + x_{1-b}$ is still computed as in H_5 with x_{1-b}.

Claim 23. $\text{Hybrid}_5 \overset{s}{\approx} \text{Hybrid}_6$.

Proof: Let $\tilde{A}_{1-b}, \tilde{B}_{1-b}$ contain the same entries as A_{1-b}, B_{1-b} in H_5 with the exception that the entries whose decommitments have been removed both in $\overline{\mathsf{TK}}$ and $\widetilde{\mathsf{TK}}$ as described in hybrids H_4 and H_5 are set to $\perp$. More precisely, given the extracted values for b_j's and c_i's, for every $j \in [2\kappa]$ such that $b_j = b$, $\tilde{A}_{1-b}(i,j) = \perp$ if $c_i = 1$ and $\tilde{B}_{1-b}(i,j) = \perp$ if $c_i = 0$ for all $i \in [\kappa]$.

Observe that, for every i, j, either $\tilde{A}_{1-b}[i,j] = A_{1-b}[i,j]$ or $\tilde{A}_{1-b}[i,j] = \perp$. The same holds for the $\tilde{B}_{1-b}$. We claim that the information of at most κ shares of x_{1-b} is present in matrices $\tilde{A}_{1-b}, \tilde{B}_{1-b}$. To this end, for every column j such that $b_j \neq 1-b$ and for every row i, depending on c_i, either $\tilde{A}_{1-b}[i,j] = \perp$, or $\tilde{B}_{1-b}[i,j] = \perp$. For every pair i, j, since $A_{1-b}[i,j]$ and $B_{1-b}[i,j]$ are both uniformly distributed, obtaining the value for at most one of them keeps $A_{1-b}[i,j] + B_{1-b}[i,j]$ statistically hidden. Now, since $b_j \neq 1-b$ for at least $\kappa+1$ shares, it follows that at least $\kappa+1$ shares of x_{1-b} are hidden. In other words, at most κ shares of x_{1-b} can be obtained by the receiver in H_5. Analogously at most κ shares of x^* are obtained in H_6. From our secret-sharing scheme, it follows that κ shares information theoretically hides the value. Therefore, the decommitments obtained by the receiver in H_5 and H_6 and identically distributed. The claim now follows from the fact that the commitments to the matrices $(\mathsf{com}_{A_0}, \mathsf{com}_{B_0}, \mathsf{com}_{A_1}, \mathsf{com}_{B_1})$ are statistically-hiding. □

Hybrid H_7: In this hybrid execution simulator $\mathcal{S}_7$ does not know the sender's inputs (s_0, s_1), but rather communicates with a trusted party that computes $\mathcal{F}_{\text{OT}}$. $\mathcal{S}_7$ behaves exactly as $\mathcal{S}_6$, except that when extracting the bit b, it sends it to the trusted party which sends back s_b. Moreover, $\mathcal{S}_7$ uses random values for s_{1-b} and C_{1-b}. Note that since the value committed to in the matrices corresponding to $1-b$ is independent of x_{1-b}, this hybrid is identically distributed to the previous hybrid. We conclude with the following claim.

Claim 24. $\text{Hybrid}_6 \equiv \text{Hybrid}_7$.

Hybrid H_8: In this hybrid execution, tokens $\{\mathsf{TK}_j\}_{j\in[2\kappa]}$ are created instead of tokens $\{\widetilde{\mathsf{TK}}_j\}_{j\in[2\kappa]}$. In addition, tokens $\{\widehat{\mathsf{TK}}_i\}_{i\in[\kappa]}$ are created instead of tokens $\{\widetilde{\mathsf{TK}}_i\}_{i\in[\kappa]}$. Due to similar claims as above, it holds that

Claim 25. $\text{Hybrid}_7 \overset{c}{\approx} \text{Hybrid}_8$.

Finally, we note that hybrid H_8 is identical to the simulated execution which concludes the proof. ■

■

On relying on one-way functions. In this protocol the only place where one-way permutations are used is in the commitments made by the sender in the second round of the protocol via a non-interactive perfectly-binding commitment. This protocol can be easily modified to rely on statistically-binding commitments which have two-round constructions based on one-way functions [Nao91]. Specifically, since the sender commits to its messages only in the second-round, the receiver can provide the first message of the two-round commitment scheme along with the first message of the protocol.

5.5.1 Two-Round 2PC Using Stateless Tokens

In [IKO$^+$11], the authors provide a two-round UC secure protocol in the OT-hybrid between a sender and a receiver where the receiver obtains the output of the computation at the end of the second round. First, we observe that we can repeat our OT protocol in parallel. Then, obtaining UC secure two-party computation using tokens is carried out by running the two-round protocol of [IKO$^+$11] in parallel with our OT protocol. Namely, upon receiving the second message for the [IKO$^+$11] and OT protocols, the receiver computes the OT outcome and uses these to compute the outcome of the [IKO$^+$11] protocol.

In more details, in order to achieve simulation when the sender is corrupted, we rely on the receiver simulation for both our OT protocol and the [IKO$^+$11] protocol. Next, we observe that, after the simulation submits the receiver's first message, it can extract the sender's input by extracting the sender's input to the OT tokens. To achieve simulation when the receiver is corrupted, the simulator first extracts the receiver's input by extracting the receiver's input to the OT tokens. Then the simulation queries the ideal functionality to obtain the output of the function evaluation on their private inputs. Using the output, the simulator next sets up the OT part of the sender's message using the OT simulation and submits the [IKO$^+$11] sender's message using the [IKO$^+$11] simulation. Thus, we obtain the following theorem:

Theorem 5.5.2. Assuming one-way functions, there exists a two-round two-party protocol for any well-formed functionality that is GUC secure in the presence of static malicious adversaries.

5.5.2 GUC-Secure MPC using Stateless Tokens from One-Way Functions

From the work of [IPS08], we know that assuming one-way functions, there exists a multiparty protocol in the OT-hybrid to securely realize any well-formed functionality with UC-security. Since, we realize the GUC-OT functionality combining with the works of [IPS08] we obtain the following corollary:

Theorem 5.5.3. Assuming one-way functions, there exists a $O(d_f)$ multi-party protocol for any well-formed functionality f that is GUC secure in the presence of static malicious adversaries where d_f is the depth of the circuit implementing the function f.

We refer the reader to the full version for our three round multi-party computation protocol.

5.6 Issue with *Over Extraction* in Oblivious Transfer Combiners [GIS+10]

In the following we identify an issue that affects one of the feasibility results in [GIS+10, Section 5]. More precisely, this result establishes that UC security for general functionalities is feasible in the tamper-proof hardware model in $O(\kappa)$-round assuming only OWFs (or $O(1)$-round based on CRHFs) based on stateless tokens. The issue arises as a result of *over extraction* where a fully-secure OT protocol is constructed from a weaker variant and the simulation extracts values for sender's inputs even on certain executions where the receiver aborts. The term over extraction has been studied before in the context of commitment schemes where a scheme with over extraction is constructed as an intermediate step towards achieving full security [PW09, GLOV12].

On a high-level, in the work of [GIS+10], they first construct an OT protocol with milder security guarantees. More precisely, a QuasiOT protocol achieves UC-security against a malicious receiver and straight-line extraction against malicious sender. However, the scheme is not fully secure as a malicious sender can cause an input-dependent abort for an honest receiver. Towards amplifying the security, [GIS+10] consider the following protocol:

1. The sender with input (s_0, s_1) and receiver with input b interact in n executions of QuasiOTs. The sender picks $z_1, \ldots, z_n$ and Δ at random and sets the inputs to the i^{th} QuasiOT instance as $(z_i, z_i + \Delta)$. The receiver on the other hand chooses bits $b_1, \ldots, b_n$ at random subject to the sum being its input b.

2. If the first step completes, the sender sends $(s'_0 = s_0 + \sum_i z_i, s'_1 = s_1 + \sum_i z_i + \Delta)$ to the receiver. The receiver computes its output as $s'_b + \sum_i w_i$ where w_i is the output of the receiver in the i^{th} QuasiOT.

This protocol remains secure against a malicious receiver. However, an issue arises with a malicious sender. To simulate a malicious sender in this protocol, [GIS+10] rely on the straight-line extractor of the n QuasiOTs by sampling two sets of random $(b_1, \ldots, b_n)$, one set summing up to 0 and another set summing up to 1 and computing what the receiver outputs in the two cases. As we demonstrate below such a strategy leads to failure in the simulation. More precisely, consider the following malicious sender strategy.

- Pick $z_1, z_2, ..., z_{n-1}$ and Δ at random.
- The inputs of the first $n-1$ tokens are set to $z_1, z_1 + \Delta, \ldots, z_{n-1}, z_{n-1} + \Delta$.
- Let $z_1 + \ldots + z_{n-1} = a$ and $z_1 + \ldots + z_{n-1} + \Delta = b$.
- The inputs to the n-th token are some fixed values c (when $b_n = 0$) and d (when $b_n = 1$), where $c + d \neq \Delta$.

Next, the sender modifies the code of the tokens used in the QuasiOT protocol so that the first $n-1$ QuasiOTs never abort. The n-th instantiations however is made to abort whenever the input b_n, the receiver's input is 1. Let $s_0 = 0$ and $s_1 = 1$ (we remark that we are not concerned about the actual inputs of the sender, but focus on what the receiver learns). We next examine the honest receiver's output in both the real and ideal worlds. First, in the real world the honest receiver learns an output only if $b_n = 0$ (since the n-th token aborts whenever $b_n = 1$). We consider two cases:

Case 1: The receiver's input is $b = 0$. Then $b_n = 0$ with probability 1/2, and $b_n = 1$ with probability 1/2. Moreover, when $b_n = 0$, the sum of the outputs obtained by the receiver is $a + c$. This is because when $b_n = 0$, then, $b_1 + \ldots + b_{n-1} = 0$, and the receiver learns a as the sum of the outputs in the the first $n - 1$ QuasiOTs and c from the n-th QuasiOT. On the other hand, if $b_n = 1$ then the receiver aborts in the n-th QuasiOT and therefore aborts.

Case 2: The receiver's input is $b = 1$. Similarly, in this case the receiver will learn $b + c$ with probability 1/2 and aborts with probability 1/2.

In the ideal world, the simulator runs first with a random bit-vector and extracts its inputs in the QuasiOTs by monitoring the queries to the corresponding PRF tokens. Next, it generates two bit-vectors b_i's and b'_i's that add up to 0 and 1, respectively, and computes the sums of the sender's input that correspond to these bits. Then the distribution of these sums can be computed as follows:

Case 1: In case that $\sum b_i = 0$, then $b_n = 0$ with probability 1/2, and $b_n = 1$ with probability 1/2. In the former case the receiver learns $a + c$, whereas in the latter case it learns $b + d$.

Case 2: In case that $\sum b'_i = 1$, then with probability 1/2, $b'_n = 0$ and with probability 1/2, $b'_n = 1$. In the former case the receiver learns $b + c$, whereas in the latter it learns $a + d$.

Note that this distribution is different from the real distribution, where the receiver never learns $b + d$ or $a + d$ since the token will always abort and not reveal d. We remark that in our example the abort probability of the receiver is independent of its input as proven in Claim 17 in [GIS+10], yet the distribution of what it learns is different.

On a more general note, our attack presents the subtleties that need to be addressed with the "selective" abort strategy. Recent works by Ciampi et al. [COSV16b, COSV16c] have identified subtleties in recent construction of non-malleable commitments [GRRV14] where selective aborts were not completely addressed.

Chapter 6

Secure Computation in the CRS model

In this chapter we present adaptively secure MPC protocols in the CRS model based on our works in [GP15, DPR16]. See Section 2.1.2 for a detailed overview of our contributions.

6.1 Two-Round MPC for Arbitrary Adaptive Corruptions

Overview. Adaptively secure MPC first studied by Canetti, Feige, Goldreich, and Naor in 1996, is a fundamental notion in cryptography. Adaptive security is particularly hard to achieve in settings where arbitrary number of parties can be corrupted and honest parties are not trusted to properly erase their internal state. We did not know how to realize constant round protocols for this task even if we were to restrict ourselves to semi-honest adversaries and to the simpler two-party setting. Specifically the round complexity of known protocols grows with the depth of the circuit the parties are trying to compute.

In this section, using indistinguishability obfuscation, we construct a UC two-round MPC protocol secure against any active, adaptive adversary corrupting an arbitrary number of parties.

6.1.1 Techniques

The key technical tool that we use in our construction is obfuscation so let us start by recalling it briefly.

Obfuscation. Obfuscation was first rigorously defined and studied by Barak et al. [BGI$^+$12]. Most famously, they defined the notion of *virtual black box (VBB)* obfuscation, and proved that this notion is impossible to realize in general — i.e., there exist functions, though a bit unnatural, that are VBB unobfuscatable.

Barak et al. also defined a weaker notion of *indistinguishability obfuscation ($i\mathcal{O}$)*, which avoids their impossibility results. Indistinguishability obfuscation requires that for any two circuits C_0, C_1 of similar size that *compute the same function*, it is hard to distinguish an obfuscation of C_0 from an obfuscation of C_1. In a recent result, Garg et al. [GGH$^+$13b] proposed a construction of $i\mathcal{O}$ for all circuits, basing security on assumptions related to multilinear maps [GGH13a].

Starting point — Garg et al. [GGHR14] construction. In a recent work, Garg et al. [GGHR14] constructed a two-round multiparty computation protocol secure against static adversaries. Though our goal is to realize a protocol secure in the adaptive setting it would be illustrative to see how Garg et al.'s construction works.

With the goal of explaining intuition [GGHR14] better we will describe the ideas assuming we have access to VBB obfuscation, rather than just indistinguishability obfuscation. We start by noting that two rounds of interaction are essential for realizing multiparty secure computation.

This is because a 1-round protocol is inherently susceptible to the “residual function" attack in which a corrupted party could repeatedly evaluate the “residual function" with the inputs of the honest parties fixed on many different inputs of its own (e.g., see [HLP11]). This attack can be circumvented by having two rounds of interaction — where in the first round the parties commit to their inputs and the output is generated only in the second round. The first round commitments help guarantee that the “residual function" attack can not be performed in this setting.

The key idea of the Garg et al. construction is to have every party commit to its input along with its randomness in the first round. The second round of the Garg et al. protocol is actually a simple compiler: it takes any (possibly highly interactive) underlying MPC protocol, and has each party obfuscate their “next-message" function in that protocol, providing one obfuscation for each round. This enables each party to independently evaluate the obfuscations one by one, generating messages of the underlying MPC protocol and finally obtain the output. Party i’s next-message circuit for round j in the underlying MPC protocol depends on its input x_i and randomness r_i (which are hard-coded in the obfuscation). This circuit takes as input the transcript through round $j-1$, and it produces as output the next broadcast message.

However, there is another complication. Unlike the initial interactive protocol being compiled, the obfuscations are susceptible to a “reset" attack – i.e., they can be evaluated on multiple inputs. To prevent such an attack, we need to limit the obfuscations to be used for evaluation only on a unique set of values – namely, values consistent with the inputs and randomness that the parties committed to in the first round, and the current transcript of the underlying MPC protocol. Note that this would implicitly fix the transcript to a unique value. To ensure this consistency, Garg et al. [GGHR14] use non-interactive zero-knowledge (NIZK) proofs. Since the NIZKs apply not only to the committed values of the first round, but also to the transcript as it develops in the second round, the obfuscations themselves must also generate these NIZKs “on the fly". In other words, the obfuscations are now augmented to perform not only the next-message function, but also to prove that their output is consistent with their input, randomness and transcript so far. Also, obfuscations in round j of the underlying MPC protocol verify NIZKs associated to obfuscations in previous rounds before providing any output.

Garg et al. show that this construction can be adapted so that security can be based on indistinguishability obfuscation alone but we will not delve into that. Instead we will argue that this approach is fundamentally problematic for achieving the task at hand.

Our approach – starting afresh. Note that the above intuitive description uses multiple obfuscations that are generated by honest parties. This however only works in the static setting and our goal is adaptive security. The challenge in proving adaptive security is that, a simulator would have a hard time explaining these obfuscations as being honestly generated. Towards solving this problem we first would like to limit the use of obfuscation in our construction; specifically not requiring honest parties to generate any obfuscations.

Still assuming we have access to VBB obfuscation, we need a fresh direction to solve the above problem. Here is our first stab at the problem: assume the parties had access to a trusted third party. In this case each party could encrypt its input and deliver it to the trusted party. The trusted party could then decrypts these values to obtain the inputs of all the parties, compute the function on the inputs and then deliver the output back to the parties. Our idea is to have an obfuscated program given out as part of the CRS implement this trusted party. Just like the Garg et al. construction, in order to make this construction secure against "residual function" attack we will need to consider a setting with two rounds. In the first round, we will have all parties commit to their inputs and then in the second round we will have them provide encryptions of the openings previously committed.

Making this construction adaptively secure seems more amenable — specifically, by using adaptive commitments for the first round and a deniable encryption scheme for the second. We actually need the first round commitments to be simulation extractable. This allows our simulator to extract the values committed to by the adversary on behalf of corrupted parties, even as it equivocates on its own commitments. Once the simulator has access to the inputs of the corrupted outputs it can force the output by including it in its own second round encryption.

Basing it on Indistinguishability Obfuscation. The protocol described so far relies on VBB and we would like to instantiate our construction based on $i\mathcal{O}$. The CRS of the scheme includes an obfuscation that takes as input encryptions of inputs of all the parties and computes the desired functionality on their decryptions. A reader might have observed that this bears resemblance with functional encryption or even multi-input functional encryption [GGG+14]. One might wonder if the use of "two key trick" can help us realize this construction using just indistinguishability obfuscation — in a way similar to the functional encryption construction of Garg et al. [GGH+13b]. In particular the idea would be that each party encrypts its input along with the opening twice under two different keys and attach along with them a NIZK proof proving that they indeed encrypt the same value.

Unfortunately, this solution is fundamentally problematic as we are in the adaptive setting. Even if we were to use an adaptively secure NIZK the problem is that NIZKs given on deniable encryptions are useless. This is because the encryption scheme is deniable. The deniability of the encryption scheme allows the adversary to encrypt two different plaintexts under the two public keys but still succeed in explaining them as encrypting the same message. This also allows the attacker to successfully prove that the two ciphertexts indeed encrypt the same message.

In summary, what we really need is a system with two ciphertexts and a proof proving that the two ciphertexts encrypt the same message with the property that only valid proofs exists. Additionally we need the property that both the ciphertexts and the proof can be denied upon in the proof of security. These requirements indeed seem to be in conflict with each other. For example, simultaneously achieving perfect soundness for NIZK and the ability to explain the simulated proofs as though they were honestly generated seems like a bottleneck.

Our solution to this seemingly paradoxical problem is to first construct a two key encryption scheme which comes attached with a NIZK and then build deniability on top of it. In particular, the underlying core encryption scheme consists of two copies of a perfectly correct encryption scheme along with a NIZK proving that the two ciphertexts encrypt the same message and it is combined with a language which also binds it with the commitments of the first round. The NIZK we use will have statistical soundness. This underlying encryption scheme is then made *deniable* using the Sahai and Waters [SW14] Universal Deniable Encryption (UDE) transformation. Briefly, UDE takes *any* encryption scheme and converts it to *deniable* so that ciphertexts are still indistinguishable from the usual ciphertexts of the underlying core encryption. Our resulting encryption is deniable in a very strong sense — specifically, it allows the encryptor to deny not just on the two ciphertexts but also on the NIZK directly. However, interestingly proofs for invalid statements do not exist.

Finally various other technical challenges arise in the security proof. For example, in the proof of security the simulator needs to hardcode the output that the adversary gets as part of its ciphertext in a way that remains indistinguishable from an honest execution. In order to force the output, the core encryption scheme which is plugged into the UDE transformation is combined with a 'trapdoor' language. In its trapdoor mode, the simulator can in particular plant the output of the function inside the encryptions it generates on behalf of honest parties. Then the obfuscation checks for such a trapdoor and acts accordingly. We refer the reader to the full construction and proof for details on how we resolve this and other issues.

6.1.2 Application to leakage tolerant protocols

As another application of our techniques, we observe that our adaptively secure protocol is also leakage tolerant in a way that previous protocols failed to be. The study of leakage tolerant protocols was initiated by Bitansky et al. [BCH12] and Garg et al. [GJS11]. Very roughly, leakage tolerant protocols preserve security even when the adversary can obtain arbitrary leakage on the entire internal state of honest parties, however only up to the leaked information.

One limitation of known leakage tolerant secure computation protocols [BGJ$^+$13] (see also [DHP11]) from the literature is that the leakage in the ideal world queries needs to depend on the inputs of all honest parties rather than just on the input of the party being leaked upon. Our adaptively secure protocol also turns out to be leakage resilient further avoiding this limitation. Another advantage of our protocol is that it is much simpler than the Boyle et al. [BGJ$^+$13] construction.

In a recent result, Garg et al. [GGKS14] show an alternative way of avoiding this limitation, without using indistinguishability obfuscation. However their result is restricted to a setting where at least one of the parties is never leaked on. We do not make such an assumption.

6.1.3 Preliminaries in the CRS Model

In this section we recall preliminary notions needed in this work. We will start by recalling notions of indistinguishability obfuscation and non-interactive zero-knowledge. Next we recall the notion of publicly deniable encryption scheme that we adapt from [SW14].

6.1.3.1 Notation

Throuhgout this section $\lambda \in \mathbb{N}$ will denote the security parameter. We say that a function $f : \mathbb{N} \to \mathbb{R}$ is negligible if $\forall c \; \exists \; n_c$ such that if $n > n_c$ then $f(n) < n^{-c}$. We will use $\mathsf{negl}(\cdot)$ to denote an unspecified negligible function. We often use $[n]$ to denote the set $\{1, ..., n\}$. The concatenation of a with b is denoted by $a||b$. Moreover, we use $d \leftarrow \mathcal{D}$ to denote the process of sampling d from the distribution $\mathcal{D}$ or, if $\mathcal{D}$ is a set, a uniform choice from it. If $\mathcal{D}_1$ and $\mathcal{D}_2$ are two distributions, then we denote that they are statistically close by $\mathcal{D}_1 \approx_s \mathcal{D}_2$; we denote that they are computationally indistinguishable by $\mathcal{D}_1 \approx_c \mathcal{D}_2$; and we denote that they are identical by $\mathcal{D}_1 \equiv \mathcal{D}_2$.

6.1.3.2 Indistinguishability Obfuscators

We will start by recalling the notion of indistinguishability obfuscation ($i\mathcal{O}$) recently realized in [GGH+13b] using candidate multilinear maps [GGH13a].

Definition 6.1 (Indistinguishability Obfuscator ($i\mathcal{O}$))**.** A uniform PPT machine $i\mathcal{O}$ is called an *indistinguishability obfuscator* for a circuit class $\{\mathcal{C}_\lambda\}$ if the following conditions are satisfied:

- For all security parameters $\lambda \in \mathbb{N}$, for all $C \in \mathcal{C}_\lambda$, for all inputs x, we have that
$$\Pr[C'(x) = C(x) : C' \leftarrow i\mathcal{O}(\lambda, C)] = 1$$
- For any (not necessarily uniform) PPT distinguisher D, there exists a negligible function α such that the following holds: For all security parameters $\lambda \in \mathbb{N}$, for all pairs of circuits $C_0, C_1 \in \mathcal{C}_\lambda$, we have that if $C_0(x) = C_1(x)$ for all inputs x, then
$$\Big| \Pr\left[D(i\mathcal{O}(\lambda, C_0)) = 1\right] - \Pr\left[D(i\mathcal{O}(\lambda, C_1)) = 1\right] \Big| \leq negl(\lambda)$$

6.1.3.3 Non-Interactive Zero-Knowledge Proofs

Let $\mathcal{R}$ be an NP-relation. For pairs $(x, w) \in \mathcal{R}$ we call x the statement and w the witness. Let $\mathcal{L}$ be the language consisting of statements in $\mathcal{R}$. A Non-Interactive Zero Knowledge (NIZK) Proof system [BFM90, FLS90] consists of three PPT algorithms (K, P, V), a common reference string generation algorithm K, a prover P and a verifier V.

- $K(1^\lambda)$ expects as input the unary representation of the security parameter λ and outputs a common reference string σ of length $\Omega(\lambda)$.
- $P(\sigma, x, w)$ takes as input a common reference string σ, a statement x together with a witness w such that $\mathcal{R}(x, w)$ and produces a proof π.
- $V(\sigma, x, \pi)$ takes as input a common reference string σ, a statement x, a proof π and outputs 1 if the proof is accepting and 0 otherwise.

We call (K, P, V) a non-interactive proof system for $\mathcal{R}$ if it satisfies the following properties.

Perfect completeness. A proof system is complete if an honest prover with a valid witness can convince an honest verifier. Formally, $\forall x \in \mathcal{L}$, $\forall w$ witness of x
$$\Pr\Big[\sigma \leftarrow K(1^\lambda); \pi \leftarrow P(\sigma, x, w) : V(\sigma, x, \pi) = 1\Big] = 1.$$

Statistical soundness. A proof system is sound if it is infeasible to convince an honest verifier when the statement is false. Formally, we have
$$\Pr\Big[\sigma \leftarrow K(1^\lambda); \exists (x, \pi) : x \notin \mathcal{L} \wedge V(\sigma, x, \pi) = 1\Big] < \mathsf{negl}(\lambda).$$

Computational zero-knowledge. We say a non-interactive proof (K, P, V) is computational zero-knowledge if there exists a PPT simulator $S = (S_1, S_2)$, where S_1 returns a simulated common reference string $\tilde{\sigma}$ together with a simulation trapdoor τ that enables S_2 to simulate proofs without having access to the witness. For all non-uniform PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ we have:

$$\Big| \Pr\Big[\sigma \leftarrow K(1^\lambda); (x, state) \leftarrow \mathcal{A}_1(\sigma); \pi \leftarrow P(\sigma, x, w) : \mathcal{A}_2(x, \sigma, \pi, state) = 1\Big] - \Pr\Big[(\sigma, \tau) \leftarrow S_1(1^\lambda); (x, state) \leftarrow \mathcal{A}_1(\sigma); \pi \leftarrow S_2(\sigma, \tau, x) : \mathcal{A}_2(x, \sigma, \pi, state) = 1\Big] \Big| < \mathsf{negl}(\lambda).$$

6.1.3.4 Double Key Encryption and its Deniable Variant

Our protocol will use a special publicly deniable encryption scheme that we construct by first describing a special public-key encryption scheme that we then transform it to its deniable variant using the Universal Deniable Encryption (UDE) transformation of [SW14].

Let $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ be a perfectly correct IND-CPA secure public-key encryption scheme and let (K, P, V) be a NIZK proof system with statistical soundness and computational zero-knowledge. The special encryption scheme we consider is very similar to the Naor-Yung CCA [NY90] secure encryption scheme. Recall that in the Naor-Yung construction a ciphertext consists of encryption of a message under two different public keys and a NIZK proof certifying that the two ciphertexts indeed encrypt the same message. In our encryption scheme a ciphertext will also consist of two ciphertexts under the two public keys but the NIZK proof will be used to certify a more sophisticated requirement. More formally:

Definition 6.2 (Double Key Encryption Scheme)**.** Let $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ be a IND-CPA secure encryption scheme with perfect correctness. Let (K, P, V) be a NIZK proof system for an NP-Language $\mathcal{L}$. A *Double Key encryption scheme*, parametrised by a language $\mathcal{L}$, consists of three algorithms $\mathsf{DK}_\mathcal{L} = (\mathsf{Setup}_\mathsf{DK}, \mathsf{Enc}_\mathsf{DK}, \mathsf{Dec}_\mathsf{DK})$.

- $\mathsf{Setup}_\mathsf{DK}(1^\lambda, 1^\ell)$ is a polynomial time procedure that takes as input the unary representation of the security parameter λ, the description of length of messages encrypted 1^ℓ. It computes $(pk_0, sk_0), (pk_1, sk_1) \leftarrow \mathsf{Setup}(1^\lambda)$ and the common reference string $\sigma \leftarrow K(1^\lambda)$ for the NIZK proof. It outputs the public key $\mathsf{pk} = (pk_0, pk_1, \sigma)$ and the secret key $\mathsf{sk} = (sk_0, sk_1)$.

- $\mathsf{Enc}_\mathsf{DK}(\mathsf{pk}, m_0, m_1, \mathsf{aux}, w; r)$: This polynomial time procedure takes as input public key $\mathsf{pk} = (pk_0, pk_1, \sigma)$, messages $m_0, m_1 \in \{0,1\}^\ell$, auxiliary information aux and some w which will be used as part of the witness for the language $\mathcal{L}$. It generates $c = \mathsf{Enc}(pk_0, m_0; s_0)$ and $c' = \mathsf{Enc}(pk_1, m_1; s_1)$ and outputs (c, c', π), where $\pi \leftarrow P(\sigma, (c, c', \mathsf{aux}), (m_0, m_1, s_0, s_1, w))$ for the language $\mathcal{L}$.

- $\mathsf{Dec}_\mathsf{DK}(\mathsf{pk}, \mathsf{sk}, (c, c', \pi))$: is a polynomial time procedure that takes as input $\mathsf{pk} = (pk_0, pk_1, \sigma)$, $\mathsf{sk} = (sk_0, sk_1)$ and ciphertext (c, c', π). Outputs $\perp$, in case that $V(\sigma, (c, c', \mathsf{aux}), \pi) = 0$ else output $(\mathsf{Dec}(sk_0, c), \mathsf{Dec}(sk_1, c'))$.

Double Key Deniable Encryption Scheme. Next we want to transform the above public key encryption into its deniable variant using the UDE transformation of Sahai and Waters [SW14, Section 4.2]. In particular, once we plug the above $\mathsf{DK}_{\mathcal{L}}$ double key encryption scheme in the UDE transformation, we obtain a double key *deniable* encryption scheme $\mathsf{DDK}_{\mathcal{L}} = (\mathsf{Setup}_{\mathsf{DDK}}, \mathsf{Enc}_{\mathsf{DDK}},$
$\mathsf{Dec}_{\mathsf{DDK}}, \mathsf{DenEnc}_{\mathsf{DDK}}, \mathsf{Explain}_{\mathsf{DDK}})$ parametrized by the language $\mathcal{L}$ with associate relation $\mathcal{R}_{\mathcal{L}}$ where the procedures $\mathsf{Enc}_{\mathsf{DDK}}$ and $\mathsf{Dec}_{\mathsf{DDK}}$ are same as $\mathsf{Enc}_{\mathsf{DK}}$ and $\mathsf{Dec}_{\mathsf{DK}}$. Here $\mathsf{Setup}_{\mathsf{DDK}}$ is obtained by augmenting the procedure $\mathsf{Setup}_{\mathsf{DK}}$ to additionally output a public denying key DK generated using $\mathsf{UniversalSetup}(\mathsf{pk})$ as defined in [SW14, Section 4.2] which is going to be included in pk. Further the scheme is augmented with the following two procedures where $\mathsf{pk} = (\sigma, pk_0, pk_1, DK)$.

- $\mathsf{DenEnc}_{\mathsf{DDK}}(\mathsf{pk}, m_0, m_1, \mathsf{aux}, w; r)$ is a polynomial time procedure that takes as input pk which includes the public denying key DK, $m_0, m_1 \in \{0,1\}^\ell$, auxiliary information aux and witness w and uses random coins r. It then outputs (c, c', π).

- $\mathsf{Explain}_{\mathsf{DDK}}(\mathsf{pk}, (c, c', \pi), (m_0, m_1, \mathsf{aux}, w))$: This polynomial time procedure takes as input public key pk which includes the public denying key DK, messages $m_0, m_1 \in \{0,1\}^\ell$, auxiliary information aux and witness w. It also takes as input a value (c, c', π) and outputs a string e, that is the same size as the randomness r taken by $\mathsf{DenEnc}_{\mathsf{DDK}}$ above.

This new scheme has the following two additional properties.

Indistinguishability of source of ciphertext. We say that the scheme has indistinguishability of source of ciphertext if for any λ and any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ the following quantity can be upper bounded by a negligible function:

$$\left| \Pr\begin{bmatrix} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}_{\mathsf{DDK}}(1^\lambda, 1^\ell), \\ (m_0, m_1, \mathsf{aux}, w) \leftarrow \mathcal{A}_1(\mathsf{pk}), \\ ct = \mathsf{Enc}_{\mathsf{DDK}}(\mathsf{pk}, m_0, m_1, \mathsf{aux}, w; r) \\ \mathcal{A}_2(\mathsf{pk}, ct) = 1 \end{bmatrix} - \Pr\begin{bmatrix} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}_{\mathsf{DDK}}(1^\lambda, 1^\ell), \\ (m_0, m_1, \mathsf{aux}, w) \leftarrow \mathcal{A}_1(\mathsf{pk}), \\ ct = \mathsf{DenEnc}_{\mathsf{DDK}}(\mathsf{pk}, m_0, m_1, \mathsf{aux}, w; r) \\ \mathcal{A}_2(\mathsf{pk}, ct) = 1 \end{bmatrix} \right|$$

Indistinguishability of explanation. We say that the scheme has indistinguishability of explanation if for any λ and any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ the following quantity can be upper bounded by a negligible function:

$$\left| \Pr\begin{bmatrix} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}_{\mathsf{DDK}}(1^\lambda, 1^\ell), \\ (m_0, m_1, \mathsf{aux}, w) \leftarrow \mathcal{A}_1(\mathsf{pk}), \\ ct = \mathsf{DenEnc}_{\mathsf{DDK}}(\mathsf{pk}, m_0, m_1, \mathsf{aux}, w; r) \\ \\ \mathcal{A}_2(\mathsf{pk}, ct, r) = 1 \end{bmatrix} - \Pr\begin{bmatrix} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}_{\mathsf{DDK}}(1^\lambda, 1^\ell), \\ (m_0, m_1, \mathsf{aux}, w) \leftarrow \mathcal{A}_1(\mathsf{pk}), \\ ct = \mathsf{DenEnc}_{\mathsf{DDK}}(\mathsf{pk}, m_0, m_1, \mathsf{aux}, w; r) \\ e = \mathsf{Explain}_{\mathsf{DDK}}(\mathsf{pk}, ct, (m_0, m_1, \mathsf{aux}, w)) \\ \mathcal{A}_2(\mathsf{pk}, ct, e) = 1 \end{bmatrix} \right|$$

6.1.3.5 Equivocal and Extractable Commitments

An Equivocal and Extractable Commitment scheme COM consists of a tuple of PPT algorithms $(\mathsf{Setup}^{\mathsf{bind}}_{\mathsf{Com}}, \mathsf{Setup}^{\mathsf{equiv}}_{\mathsf{Com}}, \mathsf{Com}, \mathsf{Extr}, \mathsf{Equiv})$. We will describe our definitions for the setting of bit commitment and note that they extend to the setting of strings in a natural way.

- $\mathsf{Setup}^{\mathsf{bind}}_{\mathsf{Com}}(1^\lambda)$ expects as input the unary representation of the security parameter λ and outputs a public parameter CK together with a trapdoor μ (used for extraction).
- $\mathsf{Setup}^{\mathsf{equiv}}_{\mathsf{Com}}(1^\lambda)$ expects as input the unary representation of the security parameter λ and outputs a public parameter CK together with trapdoors μ and ν (used for extraction and equivocation).
- $\mathsf{Com}(CK, b; r)$ takes as input CK, a bit $b \in \{0,1\}$ and randomness $r \in \{0,1\}^\lambda$ and outputs a commitment β.

Let us define the following language (the extraction procedure Extr is defined below):

$$\mathcal{L}_{\mathsf{Com}} = \{(\beta, b) \mid \exists t \ : \ \beta = \mathsf{Com}(CK, b; t) \vee b = \mathsf{Extr}(CK, t, \beta)\}.$$

We note that the language naturally extends to a setting where commitments are defined over strings instead of just bits. Also we defined associated relation $\mathcal{R}_{\mathsf{Com}}$. The above commitment scheme should satisfy the following properties.

Indistinguishability of Public Parameters. We require that:

$$\Big| \Pr\Big[(CK, \mu) \leftarrow \mathsf{Setup}^{\mathsf{bind}}_{\mathsf{Com}}(1^\lambda) : \mathcal{A}(CK, \mu) = 1\Big] - \Pr\Big[(CK, \mu, \nu) \leftarrow \mathsf{Setup}^{\mathsf{equiv}}_{\mathsf{Com}}(1^\lambda) : \mathcal{A}(CK, \mu) = 1\Big]\Big| < \mathsf{negl}(\lambda).$$

Computational Hiding. Hiding means that no computationally bounded adversary can distinguish as to which bit is locked in the commitment. Let $\mathcal{A}$ be any non-uniform adversary running in time $poly(\lambda)$. We say that the commitment scheme is computationally hiding if:

$$\Pr\left[b = b' \,\middle|\, \begin{array}{l} b \leftarrow \{0,1\}; (CK, \mu) \leftarrow \mathsf{Setup}^{\mathsf{bind}}_{\mathsf{Com}}(1^\lambda); \\ \beta = \mathsf{Com}(CK, b; r); b' \leftarrow \mathcal{A}(\beta) \end{array}\right] = \frac{1}{2} + \mathsf{negl}(\lambda) \ .$$

The same applies to the setup algorithm $\mathsf{Setup}^{\mathsf{equiv}}_{\mathsf{Com}}$.

Perfectly Binding. Intuitively speaking, binding requires that no (even unbounded) adversary can open the commitment in two different ways. Here, we define the strongest variant known as *perfectly binding*. Formally we require that for all $(CK, \mu) \leftarrow \mathsf{Setup}^{\mathsf{bind}}_{\mathsf{Com}}(1^\lambda)$ there exists no values (r_0, r_1) such that $\mathsf{Com}(CK, 0; r_0) = \mathsf{Com}(CK, 1; r_1)$. For perfectly binding we require that either $(c, 0) \in \mathcal{L}_{\mathsf{Com}}$ or $(c, 1) \in \mathcal{L}_{\mathsf{Com}}$, but not both.

Polynomial equivocality. The setup algorithm $\mathsf{Setup}^{\mathsf{equiv}}_{\mathsf{Com}}$ generates public parameters together with trapdoors μ and ν such that Equiv using ν is able to produce polynomially many fake commitments using the same CK which can then be explained to either 0 and 1. More formally, Equiv can be viewed as a pair of PPT algorithms $(\mathsf{Equiv}_1, \mathsf{Equiv}_2)$ such that the following holds. Let $(CK, \mu, \nu) \leftarrow \mathsf{Setup}^{\mathsf{equiv}}_{\mathsf{Com}}(1^\lambda)$ then $(\beta, state) \leftarrow \mathsf{Equiv}_1(CK, \nu)$ and $r_b \leftarrow \mathsf{Equiv}_2(CK, \nu, \beta, state, b)$ such that $\mathsf{Com}(CK, b; r_b) = \beta$. Furthermore we require that for $b \in \{0, 1\}$ the distribution of $\{(CK, \beta, r_b)\}$ generated in this way is computationally indistinguishable from the distribution $\{(CK, \beta, r_b)\}$ where $\beta = \mathsf{Com}(CK, b; r_b)$.

Simulation extractability. We require that the commitment remains binding for any adversary $\mathcal{A}$, even after $\mathcal{A}$ obtains polynomially many equivocal commitments generated by Equiv along with their openings. More formally, it should hold that

$$\Pr\left[b \neq b' \;\middle|\; \begin{array}{l} (CK, \mu, \nu) \leftarrow \mathsf{Setup}^{\mathsf{equiv}}_{\mathsf{Com}}(1^\lambda); (\beta, b, r) \leftarrow \mathcal{A}^{\mathsf{Equiv}^*(CK,\nu)}(CK); \\ \mathsf{Com}((CK, b, r) = \beta \wedge \mathsf{Extr}(CK, \mu, \beta) = b' \end{array} \right] = \mathsf{negl}(\lambda) \; .$$

where Equiv^* is either invoked as Equiv_1 without revealing the *state*, or as Equiv_2 which only expects as input fake commitments generated by previous invocations of Equiv_1.

In this paper, we use the non-interactive equivocal and extractable commitment scheme of [CLOS02] (CLOS commitment) which assumes the existence of enhanced trapdoor permutations. At the heart of their commitment scheme is the Feige-Shamir trapdoor commitment scheme [FS90b], which they transform to obtain a UC Commitment scheme secure against adaptive adversaries.

6.1.4 Our Protocol

In this section we will present our adaptively secure two-round MPC protocol, described in Figure 1. For simplicity, we assume that the delivered messages are authenticated. Also for simplicity of exposition, in the sequel, we will assume that random coins are an implicit input to the commitment and encryption functions, unless specified explicitly.

Theorem 6.1. Let f be any deterministic poly-time function with n inputs and single output. Assume the existence of a sub-exponentially secure Indistinguishability Obfuscator $i\mathcal{O}$, a Double Key Deniable encryption scheme $\mathsf{DDK}_{\mathcal{L}} = (\mathsf{Setup}_{\mathsf{DDK}}, \mathsf{Enc}_{\mathsf{DDK}}, \mathsf{Dec}_{\mathsf{DDK}}, \mathsf{DenEnc}_{\mathsf{DDK}}, \mathsf{Explain}_{\mathsf{DDK}})$ and an adaptively secure Commitment scheme $\mathsf{COM} = (\mathsf{Setup}^{\mathsf{bind}}_{\mathsf{Com}}, \mathsf{Setup}^{\mathsf{equiv}}_{\mathsf{Com}}, \mathsf{Com}, \mathsf{Extr}, \mathsf{Equiv})$. Then the protocol Π presented in Figure 1 UC-securely realizes the ideal functionality $\mathcal{F}_f$ in the $\mathcal{F}_{CRS}$-hybrid model with computational security against any adaptive, active adversary corrupting an arbitrary number of parties in two rounds of broadcast.

Corollary 6.2. Assume the existence of a sub-exponentially secure indistinguishability obfuscation and doubly enhanced trapdoor permutation then any ideal functionality $\mathcal{F}_f$ can be UC-securely realized in the $\mathcal{F}_{CRS}$- model against any adaptive, active adversary corrupting an arbitrary number of parties. Furthermore this protocol involves only two rounds of broadcast.

We start by noting that the protocol is correct. Observe that if all the parties behave honestly then the protocol ends us executing the circuit f on the inputs of all parties, leading to the correct

Protocol Π

Protocol Π uses an Indistinguishability Obfuscator $i\mathcal{O}$, a Double Key Deniable encryption scheme $\mathsf{DDK}_{\mathcal{L}} = (\mathsf{Setup}_{\mathsf{DDK}}, \mathsf{Enc}_{\mathsf{DDK}}, \mathsf{Dec}_{\mathsf{DDK}}, \mathsf{DenEnc}_{\mathsf{DDK}}, \mathsf{Explain}_{\mathsf{DDK}})$ based on the scheme $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ with perfect correctness, where the relation $\mathcal{L}$ is defined below, and an adaptively secure Commitment scheme $\mathsf{COM} = (\mathsf{Setup}^{\mathsf{bind}}_{\mathsf{Com}}, \mathsf{Com})$.[a] Let $f : (\{0,1\}^{\ell_{in}})^n \to \{0,1\}^{\ell_{out}}$ be the circuit parties want to evaluate on their private inputs.

Private Inputs: Party P_i for $i \in [n]$, receives its input x_i.
CRS: Output $(\mathsf{pk}, CK, \mathsf{o}P)$ as the common reference string generated as follows:

- Generate $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}_{\mathsf{DDK}}(1^\lambda, 1^{\ell_{in}+\ell_{out}})$ where $\mathsf{pk} = (\sigma, pk_0, pk_1, DK)$ and $\mathsf{sk} = (sk_0, sk_1)$
- Generate $(CK, \mu) \leftarrow \mathsf{Setup}^{\mathsf{bind}}_{\mathsf{Com}}(1^\lambda)$.
- Let $\mathsf{o}P = i\mathcal{O}_{\mathsf{Prog}_{sk_0,\mathsf{pk},CK,f}}$ be the obfuscation of the program $\mathsf{Prog}_{sk_0,\mathsf{pk},CK,f}$, described in Figure 2.

Round 1: Each party P_i generates $\beta_i = \mathsf{Com}(CK, x_i; \omega_i)$ and broadcasts it to all parties.

Round 2: Each party P_i generates $(c_i, c'_i, \pi_i) = \mathsf{DenEnc}_{\mathsf{DDK}}(\mathsf{pk}, x_i||\phi^{\ell_{out}}, x_i||\phi^{\ell_{out}}, (i, \{\beta_j\}_{j\in[n]}), (0^{n\cdot\ell_{in}}, 0^{\ell_{out}}, \{t_j\}_{j\in[n]}); r_i)$ where ϕ is a special fixed symbol and $t_i = \omega_i$ and $t_j = 0^*$ for all $j \in [n]$ such that $j \neq i$. It then broadcasts (c_i, c'_i, π_i) to all parties.

Output phase: Each party P_i outputs $\mathsf{o}P(\{\beta_j\}_{j\in[n]}, \{c_j, c'_j, \pi_j\}_{j\in[n]})$.

Language $\mathcal{L}$ for the Double Key deniable encryption scheme $\mathsf{DDK}_{\mathcal{L}}$: Recall $\mathcal{L}_{\mathsf{Com}}$ as the language defined in Section 6.1.3.5, and let $\mathcal{R}_{\mathsf{Com}}$ be the associated relation. We have that $(c, c', (i, \{\beta_j\}_{j\in[n]})) \in \mathcal{L}$ if $(c, c', (i, \{\beta_j\}_{j\in[n]})) \in \mathcal{L}_1 \vee \mathcal{L}_2$ defined as follows:[b]

$$\mathcal{L}_1 = \left\{ (c, c', (i, \{\beta_j\}_{j\in[n]})) \;\middle|\; \begin{array}{l} \exists\ (m_0, m_1, s_0, s_1, (\{x_j\}_{j\in[n]}, out, \{t_j\}_{j\in[n]})) \text{ such that} \\ \quad c = \mathsf{Enc}(pk_0, m_0; s_0) \wedge c' = \mathsf{Enc}(pk_1, m_1; s_1) \\ \wedge\ m_0 = m_1 = x_i||\phi^{\ell_{out}} \\ \wedge\ \mathcal{R}_{\mathsf{Com}}((\beta_i, x_i), t_i) \end{array} \right\} \tag{6.1}$$

$$\mathcal{L}_2 = \left\{ (c, c', (i, \{\beta_j\}_{j\in[n]})) \;\middle|\; \begin{array}{l} \exists\ (m_0, m_1, s_0, s_1, (\{x_j\}_{j\in[n]}, out, \{t_j\}_{j\in[n]})) \text{ such that} \\ \quad c = \mathsf{Enc}(pk_0, m_0; s_0) \wedge c' = \mathsf{Enc}(pk_1, m_1; s_1) \\ \wedge\ m_0 = x_i||\phi^{\ell_{out}} \wedge m_1 = \phi^{\ell_{in}}||out \\ \wedge\ \forall j \in [n], \mathcal{R}_{\mathsf{Com}}((\beta_j, x_j), t_j) \wedge out = f(\{x_j\}_{j\in[n]}) \end{array} \right\} \tag{6.2}$$

[a] We note that COM provides more procedures but we note that they only affect the proof. Hence for simplicity of exposition we skip mentioning them here.
[b] Changes in $\mathcal{L}_2$ from $\mathcal{L}_1$ are highlighted in red.

Figure 6.1: [GP15] Protocol Π.

Program $\mathsf{Prog}_{\mathsf{sk}_b,\mathsf{pk},CK,f}$

Input: $(\{\beta_j\}_{j\in[n]}, \{c_j, c'_j, \pi_j\}_{j\in[n]})$.

Description:

1. If there exists $j \in [n]$ such that $\mathsf{Dec}_{\mathsf{DDK}}((c_j, c'_j, \pi_j)) = \perp$ then output $\perp$.
2. Parse c_j as $d_{j,0}||e_{j,0}$ where $d_{j,0}$ is the encryption of the first ℓ_{in} bits and $e_{j,0}$ is the encryption of the rest of the bits. Similarly parse c'_j as $d_{j,1}||e_{j,1}$.

   If $\exists j \in [n]$ such that $\mathsf{Dec}(sk_b, e_{j,b}) \neq \phi^{\ell_{out}}$, then let i be the first such j. If this is the case then output $\mathsf{Dec}(sk_b, e_{i,b})$.
3. Otherwise for each $j \in [n]$, let $x_j = \mathsf{Dec}(sk_b, d_{j,b})$ and output $f(\{x_j\}_{j\in[n]})$.

Figure 6.2: [GP15] Program $\mathsf{Prog}_{\mathsf{sk}_b,\mathsf{pk},CK,f}$.

output. Security is proved via a simulator provided in Section 6.1.5 and indistinguishability is argued in Section 6.1.6.

6.1.4.1 Extensions

Now we give some natural extensions of our protocol and remove assumptions that were made to simplify exposition.

General Functionality. Our basic MPC protocol as described in Figure 1, only considers deterministic functionalities where all the parties receive the same output. We would like to generalize it to handle randomized functionalities and individual outputs (just as in [AJW11]). First, the standard transformation from a randomized functionality to a deterministic one (See [Gol04, Section 7.3]) works for this case as well. In this transformation, instead of computing some randomized function $g(x_1, \ldots x_n; r)$, the parties compute the deterministic function $f((r_1, x_1), \ldots, (r_n, x_n)) \overset{def}{=} g(x_1, \ldots, x_n; +_{i=1}^{n} r_i)$. We note that this computation does not add any additional rounds. We note that since we are in the setting of adaptive security we can only realize *adaptively well-formed* [CLOS02] functionalities, which reveals its randomness if all the parties are corrupted.

Next, we move to individual outputs. Again, we use a standard transformation (See [LP09b], for example). Given a function $g(x_1, \ldots, x_n) \to (y_1, \ldots, y_n)$, the parties can evaluate the following function which has a single output:

$$f((k_1, x_1), \ldots, (k_n; x_n)) = (g_1(x_1, \ldots, x_n) + k_1|| \ldots ||g_n(x_1, \ldots, x_n) + k_n)$$

where g_i indicates the i^{th} output of g, and k_i is randomly chosen by the i^{th} party. Then, the parties can evaluate f, which is a single output functionality, instead of g. Subsequently every party P_i uses its secret input k_i to recover its own output. The only difference is that f has one additional exclusive-or gate for every circuit-output wire. Again, this transformation does not add any additional rounds of interaction.

Making CRS independent of the circuit being computed. Note that in our construction the obfuscation $\mathsf{o}P$ that is given as part of the CRS depends on the circuit f parties are trying to compute on their joint inputs. We can remove this dependence by using a universal circuit. Then the parties can feed in the universal circuit the actual circuit that they want along with their private inputs. However, the CRS will still depend on the size of the circuit. We can avoid this by setting a priori bound on the size of the circuit being computed. Once we have an $i\mathcal{O}$ candidate for Turing machines the above can be avoided yielding a CRS independent of the size of the circuit.

6.1.5 Description of our Simulator

Let $\mathcal{A}$ be an active, adaptive adversary that interacts with parties running the protocol Π from Figure 1 in the $\mathcal{F}_{CRS}$-hybrid model. We construct a simulator $\mathcal{S}$ (the ideal world adversary) with access to the ideal functionality $\mathcal{F}_f$, which simulates a real execution of Π with $\mathcal{A}$ such that no environment $\mathcal{Z}$ can distinguish the ideal world experiment with $\mathcal{S}$ and $\mathcal{F}_f$ from a real execution of Π with $\mathcal{A}$.

Recall that $\mathcal{S}$ interacts with the ideal functionality $\mathcal{F}_f$ and with the environment $\mathcal{Z}$. The ideal adversary $\mathcal{S}$ starts by invoking a copy of $\mathcal{A}$ and running a simulated interaction of $\mathcal{A}$ with the environment $\mathcal{Z}$ and the parties running the protocol. Our simulator $\mathcal{S}$ proceeds as follows:

Simulated CRS: The common reference string is chosen by $\mathcal{S}$ in the following manner (recall that $\mathcal{S}$ chooses the CRS for the simulated $\mathcal{A}$ as we are in the $\mathcal{F}_{CRS}$-hybrid model):

1. $\mathcal{S}$ runs the setup algorithm $\mathsf{Setup}_{\mathsf{DDK}}(1^\lambda, 1^{\ell_{in}+\ell_{out}})$ of the Double Key deniable encryption scheme, but replaces its internal call to the algorithm K with $S = (S_1, S_2)$ of the NIZK proof system. More specifically, $\mathcal{S}$ generates $(pk_0, sk_0), (pk_1, sk_1) \leftarrow \mathsf{Setup}(1^\lambda)$, $(\sigma, \tau) \leftarrow S_1(1^\lambda)$, along with the public denying key DK. It sets the public key $\mathsf{pk} = (pk_0, pk_1, \sigma, DK)$.

2. $\mathcal{S}$ runs the algorithm $\mathsf{Setup}^{\mathsf{equiv}}_{\mathsf{Com}}(1^\lambda)$ of the adaptively secure commitment scheme COM and obtains (CK, μ, ν).

3. $\mathcal{S}$ computes $\mathsf{o}P = i\mathcal{O}_{\mathsf{Prog}_{sk_1,\mathsf{pk},CK,f}}$ where the latter is the obfuscation of the program Prog, as described in Figure 2, parameterized with the key sk_1.

$\mathcal{S}$ sets the common reference string equal to $(\mathsf{pk}, CK, \mathsf{o}P)$ and locally stores $(\mathsf{sk}, \tau, \mu, \nu)$.

Looking ahead, the trapdoor μ will be used to extract the inputs of the corrupted parties and ν to equivocate on the commitment $\mathcal{S}$ provides on behalf of honest parties. The trapdoor τ for the simulated σ will be used to generate simulated proofs.

Simulating the communication with $\mathcal{Z}$: Every input value that $\mathcal{S}$ receives from $\mathcal{Z}$ is written on $\mathcal{A}$'s input tape. Similarly, every output value written by $\mathcal{A}$ on its own output tape is directly copied to the output tape of $\mathcal{S}$.

Simulating actual protocol messages in Π: Note that there might be multiple sessions executing concurrently. Let sid be the session identifier for one specific session. We will specify the simulation strategy corresponding to this specific session. The simulator strategy for all other sessions will be the same. Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be the set of parties participating in the execution of Π corresponding to the session identified by the session identifier sid. Also let $\mathcal{P}^{\mathcal{A}} \subseteq \mathcal{P}$ be the set of parties corrupted by the adversary $\mathcal{A}$ at any time. Recall that we are in the setting of adaptive corruption so more parties could be added to this set as the protocol proceeds. At any point $\mathcal{S}$ only generates messages on behalf of parties $\mathcal{P}\backslash\mathcal{P}^{\mathcal{A}}$. In the following, if at the end of some round all parties are corrupted then $\mathcal{S}$ does not need to go to do anything else.

Round 1 Messages $\mathcal{S} \to \mathcal{A}$: In the first round $\mathcal{S}$ must generate messages on behalf of the honest parties, i.e. parties in the set $\mathcal{P}\backslash\mathcal{P}^{\mathcal{A}}$. For each party $P_i \in \mathcal{P}\backslash\mathcal{P}^{\mathcal{A}}$ our simulator proceeds as:

1. Generate a fake commitment $(\beta_i, state_i) \leftarrow \mathsf{Equiv}_1(CK, \nu)$.

It then sends β_i to $\mathcal{A}$ on behalf of party P_i and it internally saves $state_i$.

Round 1 Messages $\mathcal{A} \to \mathcal{S}$: Also in the first round the adversary $\mathcal{A}$ generates the messages on behalf of corrupted parties in $\mathcal{P}^{\mathcal{A}}$. For each party $P_i \in \mathcal{P}^{\mathcal{A}}$ our simulator proceeds as follows:

1. Extracting inputs of corrupted parties: Let β_i be the commitment that $\mathcal{A}$ sends on behalf of P_i. Our simulator $\mathcal{S}$ runs the extraction algorithm $\mathsf{Extr}(CK, \mu, \beta_i)$ in order to obtain x_i.

   Note that it is possible that $\mathcal{A}$ sends a commitment β_i on behalf of P_i such that it is not well-formed, or in other words extraction using the function Extr fails. In this case $\mathcal{S}$ sets $x_i = \perp$ and proceeds to the next step. (Looking ahead, we note that in this case the adversary will not be able to generate a valid second round message.)

2. Next $\mathcal{S}$ sends $(\mathsf{input}, \mathsf{sid}, \mathcal{P}, P_i, x_i)$ to $\mathcal{F}_f$ on behalf of the corrupted party P_i.

Simulating corruption of parties in Round 1: When $\mathcal{A}$ corrupts a real world party P_i, then $\mathcal{S}$ first corrupts the corresponding ideal world party P_i and obtains its input x_i. Next $\mathcal{S}$ prepares the internal state on behalf of P_i such that it will be consistent with the commitment value β_i that it had provided to $\mathcal{A}$ earlier. Specifically $\mathcal{S}$ computes $\mathsf{Equiv}_2(CK, \nu, \beta_i, state_i, x_i)$ in order to obtain randomness ω_i such that $\beta_i = \mathsf{Com}(CK, \beta_i; \omega_i)$. $\mathcal{S}$ provides ω_i as the randomness of party P_i to $\mathcal{A}$. Note that $\mathcal{S}$ can do this at any point during 1st round.

Completion of Round 1: After $\mathcal{S}$ has submitted the inputs of all the corrupted parties to $\mathcal{F}_f$ then it responds by sending back the message $(\mathsf{output}, \mathsf{sid}, \mathcal{P}, out)$ where $out = f(\{x_j\}_{j\in[n]})$. Note that in case $\mathcal{S}$ had failed to extract an input for some player P_i then it would have sent $\perp$ to $\mathcal{F}_f$ and would have received $\perp$ as the output from the ideal functionality.

Round 2 Messages $\mathcal{S} \rightarrow \mathcal{A}$: In the second round $\mathcal{S}$ generates messages on behalf of the honest parties, i.e. parties in the set $\mathcal{P} \backslash \mathcal{P}^{\mathcal{A}}$ as follows:

1. For each party $P_i \in \mathcal{P} \backslash \mathcal{P}^{\mathcal{A}}$, $\mathcal{S}$ generates $c_i = \mathsf{Enc}(pk_0, \phi^{\ell_{in}} || out), c'_i = \mathsf{Enc}(pk_1, \phi^{\ell_{in}} || out)$ and generates π_i as a simulated proof for the statement $(c_i, c'_i, (i, \{\beta_j\}_{j \in [n]}))$. More specifically it generates $\pi_i \leftarrow S_2(\sigma, \tau, (c_i, c'_i, (i, \{\beta_j\}_{j \in [n]})))$.

$\mathcal{S}$ sends (c_i, c'_i, π_i) to $\mathcal{A}$ on behalf of P_i.

Round 2 Messages $\mathcal{A} \rightarrow \mathcal{S}$: In the second round the adversary $\mathcal{A}$ generates the messages on behalf of corrupted parties in $\mathcal{P}^{\mathcal{A}}$. For each party $P_i \in \mathcal{P}^{\mathcal{A}}$ our simulator proceeds as:

1. Let (c_i, c'_i, π_i) be the message that $\mathcal{A}$ sends on behalf of party P_i. $\mathcal{S}$ checks to see if $V(\sigma, (c_i, c'_i, (i, \{\beta_j\}_{j \in [n]})), \pi_i) = 1$ for each $P_i \in \mathcal{P}^{\mathcal{A}}$.

If all the proofs verify then $\mathcal{S}$ sends the message $(\mathsf{generateOutput}, \mathsf{sid}, \mathcal{P})$ to the ideal functionality $\mathcal{F}_f$.

Simulating corruption of parties during/at the end of Round 2: When $\mathcal{A}$ corrupts a party P_i in the real word, then $\mathcal{S}$ corrupts the corresponding party P_i in the ideal world and obtains its input x_i. Next $\mathcal{S}$ prepares the internal state on behalf of P_i such that it will be consistent with messages it had sent on behalf of P_i. As explained before, $\mathcal{S}$ generates randomness ω_i that explains the commitment β_i to the value x_i running the algorithm $\omega_i = \mathsf{Equiv}_2(CK, \nu, \beta_i, state_i, x_i)$. Next $\mathcal{S}$ needs to explain the second round message (c_i, c'_i, π_i). $\mathcal{S}$ has to explain the message (c_i, c'_i, π_i) by computing the randomness as $\psi_i = \mathsf{Explain}_{\mathsf{DDK}}(\mathsf{pk}, (c_i, c'_i, \pi_i), (x_i || \phi^{\ell_{out}}, x_i || \phi^{\ell_{out}}, (i, \{\beta_j\}_{j \in [n]}),$
$(0^{n \cdot \ell_{in}}, 0^{\ell_{out}}, \{t_j\}_{j \in [n]}))$ where $t_i = \omega_i$ and $t_j = 0^*$ for all $j \in [n]$ such that $j \neq i$. $\mathcal{S}$ provides $\omega_i || \psi_i$ as the randomness of party P_i to $\mathcal{A}$. Note that $\mathcal{S}$ can do this at any point during or after the round 2 of the protocol.

This completes the description of the simulator.

6.1.6 Proof of Security

In this section, via a sequence of hybrids, we will prove that no environment $\mathcal{Z}$ can distinguish the ideal world experiment with $\mathcal{S}$ and $\mathcal{F}_f$ (as defined in the Appendix) from a real execution of Π with Adv. We will start with the real world execution in which the adversary Adv interacts directly with the honest parties holding their inputs and step-by-step make changes till we finally reach the simulator as described in Section 6.1.5. At each step we will argue that the environment cannot distinguish the change except with negligible probability.

We prove security in a model where the inputs are selective in the sense that the environment determines the inputs to the computation before it sees the CRS. The proof for the setting where the environment chooses the inputs adaptively only follows using sub-exponential security of the indistingushabiity obfuscator. We have sub-exponential loss in the security since we use the universal deniable encryption transformation of [SW14] which supports one bit messages at a time. Our construction needs to support multi-bit messages. However, we remark that the

[SW14] deniable encryption scheme immediately implies a deniable encryption scheme for multi-bit messages of any polynomial length k bits by creating a ciphertext for k bit message as a sequence of k single bit encryptions. Our construction cannot support the above bit-by-bit encryption since every single encryption apart from the messages which can be bits it also takes as input the witness for the NIZK proof which length depends on all the commitment messages of the first round.

Hybrid H_0: This hybrid corresponds to the $\mathcal{Z}$ interacting with the real world adversary Adv and honest parties that hold their private inputs.

We can restate the above experiment with the simulator as follows. We replace the real world adversary Adv with the ideal world adversary $\mathcal{S}$. The ideal adversary $\mathcal{S}$ starts by invoking a copy of Adv and running a simulated interaction of Adv with the environment $\mathcal{Z}$ and the honest parties. $\mathcal{S}$ forwards the messages that Adv generates for it environment directly to $\mathcal{Z}$ and vice versa (as explained in the description of the simulator $\mathcal{S}$). In this hybrid the simulator $\mathcal{S}$ holds the private inputs of the honest parties and generates messages on their behalf using the honest party strategies as specified by Π.

Hybrid H_1: In this hybrid, we change how the internal randomness of the corrupted party is explained to $\mathcal{A}$ on being adaptively corrupted. Specifically we change the randomness that is used to explain the ciphertext $\mathcal{S}$ generates on behalf of parties in round 2 of protocol Π.

Recall that in the second round $\mathcal{S}$ on behalf of an honest party P_i generates the second message as $(c_i, c'_i, \pi_i) = \mathsf{DenEnc}_{\mathsf{DDK}}(\mathsf{pk}, x_i||\phi^{\ell_{out}}, x_i||\phi^{\ell_{out}}, (i, \{\beta_j\}_{j\in[n]}), (0^{n\cdot\ell_{in}}, 0^{\ell_{out}}, \{t_j\}_{j\in[n]}); r_i)$ where t_i is the randomness used in generating commitment β_i and $t_j = 0^*$ for all $j \in [n]$ such that $j \neq i$. So if $\mathcal{A}$ corrupts P_i then the randomness r_i would be reveal to $\mathcal{A}$. In Hybrid 1, instead we provide $\psi_i = \mathsf{Explain}_{\mathsf{DDK}}(\mathsf{pk}, (c_i, c'_i, \pi_i), (x_i||\phi^{\ell_{out}}, x_i||\phi^{\ell_{out}}, (i, \{\beta_j\}_{j\in[n]}),$
$(0^{n\cdot\ell_{in}}, 0^{\ell_{out}}, \{t_j\}_{j\in[n]}))$ where t_j values are as before.

Lemma 6.3. $\mathrm{Hybrid}_0 \approx_c \mathrm{Hybrid}_1$.

Proof. The indistinguishability of Hybrid_1 from Hybrid_0 follows from the indistinguishability of explanation property of the Double Key deniable encryption scheme. ■

Hybrid H_2: In this hybrid we change the way $\mathcal{S}$ generates the message (c_i, c'_i, π) on behalf of the honest parties.

Recall that in the second round in Hybrid 1, $\mathcal{S}$ on behalf of an honest party P_i generates the second message as $(c_i, c'_i, \pi_i) = \mathsf{DenEnc}_{\mathsf{DDK}}(\mathsf{pk}, x_i||\phi^{\ell_{out}}, x_i||\phi^{\ell_{out}}, (i, \{\beta_j\}_{j\in[n]}), (0^{n\cdot\ell_{in}}, 0^{\ell_{out}}, \{t_j\}_{j\in[n]}); r_i)$ where t_i is the randomness used in generating commitment β_i and $t_j = 0^*$ for all $j \in [n]$ such that $j \neq i$. We will change this by generating the ciphertexts directly using procedures Enc and the prover P.

Specifically, $c_i = \mathsf{Enc}(pk_0, x_i||\phi^{\ell_{out}}; s_0^i)$ and $c'_i = \mathsf{Enc}(pk_1, x_i||\phi^{\ell_{out}}; s_1^i)$ and outputs (c_i, c'_i, π_i), where $\pi_i \leftarrow P(\sigma, (c_i, c'_i, \{i, \{\beta\}_{j\in[n]}\}), (x_i||\phi^{\ell_{out}}, x_i||\phi^{\ell_{out}}, s_0^i, s_1^i, (0^{n\cdot\ell_{in}},$

$0^{\ell_{out}}, \{t_j\}_{j\in[n]})))$ where t_i is the randomness used in generating commitment β_i and $t_j = 0^*$ for all $j \in [n]$ such that $j \neq i$.

Lemma 6.4. $\text{Hybrid}_1 \approx_c \text{Hybrid}_2$.

Proof. The indistinguishability of Hybrid_2 from Hybrid_1 follows immediately from the indistinguishability of *source of ciphertext* property of the Double Key deniable encryption scheme.

■

Hybrid H_3: In this hybrid, we change how σ, which is a part of pk, and the proofs π_i for every $P_i \in \mathcal{P}\backslash\mathcal{P}^{\mathcal{A}}$ are generated.

More specifically, $\mathcal{S}$ runs the setup algorithm $\mathsf{Setup}_{\mathsf{DDK}}(1^\lambda, 1^{\ell_{in}+\ell_{out}})$ of the Double Key deniable encryption scheme, but replaces its internal call to the algorithm K with $S = (S_1, S_2)$ of the NIZK proof system. More specifically, $\mathcal{S}$ generates $(pk_0, sk_0), (pk_1, sk_1) \leftarrow \mathsf{Setup}(1^\lambda)$, $(\sigma, \tau) \leftarrow S_1(1^\lambda)$, along with the public denying key DK. It sets the public key $\mathsf{pk} = (\sigma, pk_0, pk_1, DK)$.

We also generate fake proofs π_i using trapdoor τ. Specifically we generate $\pi_i \leftarrow S_2(\sigma, \tau, (c_i, c'_i, (i, \{\beta_j\}_{j\in[n]})))$.

Lemma 6.5. $\text{Hybrid}_2 \approx_c \text{Hybrid}_3$.

Proof. The indistinguishability of Hybrid_3 from Hybrid_2 follows immediately from the computational zero-knowledge property of the NIZK proof system. ■

Hybrid H_4: We don't change anything in the output of the hybrid itself. We just use knowledge of μ to extract the inputs $\mathcal{A}$ commits to in the 1st round messages that it sends on behalf of the corrupted parties.

More specifically, $\mathcal{S}$ for every $P_i \in \mathcal{P}^{\mathcal{A}}$ obtains $x_i = \mathsf{Extr}(CK, \mu, \beta_i)$. If extraction fails then it sets $x_i = \perp$.

Hybrid H_5: In this hybrid, we change how the simulator $\mathcal{S}$ generates c'_i in the second round message (c_i, c'_i, π_i) on behalf of honest parties $P_i \in \mathcal{P}\backslash\mathcal{P}^{\mathcal{A}}$. In particular, $\mathcal{S}$ instead of computing the ciphertext $c'_i = \mathsf{Enc}(pk_1, x_i||\phi^{\ell_{out}}; s^i_1)$, generates $c'_i = \mathsf{Enc}(pk_1, \phi^{\ell_{in}}||out; s^i_1)$, where *out* is the output computed as $f(\{x_j\}_{j\in[n]})$ using the inputs x_i of the honest parties, that the simulator has access to, and extracted inputs of the malicious parties.

Lemma 6.6. $\text{Hybrid}_4 \approx_c \text{Hybrid}_5$.

Proof. We base the indistinguishability between hybrids Hybrid_4 and Hybrid_5 on the semantic security of the encryption scheme $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$.

■

Hybrid H_6: In this hybrid we essentially reverse the change that was made in going from Hybrid 2 to Hybrid 3. In particular we change the σ so that it is sampled from the honest distribution and generate the proof honestly. Note that since now we have changed the ciphertext c'_i the proof will have to be generated with respect to language $\mathcal{L}_2$.

More specifically, $\mathcal{S}$ uses K to generate σ instead of S_1. Also for every $P_i \in \mathcal{P} \backslash \mathcal{P}^{\mathcal{A}}$, $\mathcal{S}$ generates $\pi_i \leftarrow P(\sigma, (c_i, c'_i, (i, \{\beta_j\}_{j\in[n]})), (x_i || \phi^{\ell_{out}}, \phi^{\ell_{in}} || out, s_0^i, s_1^i, (\{x_i\}_{i\in[n]}, out, \{t_j\}_{j\in[n]})))$ where t_j is the witness that $\beta_j \in \mathcal{L}_{\mathsf{Com}}$.

Lemma 6.7. $\mathrm{Hybrid}_5 \approx_{\mathrm{c}} \mathrm{Hybrid}_6$.

Proof. The indistinguishability of Hybrid_5 from Hybrid_6 follows immediately from the computational zero-knowledge property of the NIZK proof system.

■

Hybrid H_7: In this hybrid we change how $\mathsf{o}P$, the obfuscated program in the CRS is generated. More specifically, $\mathsf{o}P$ is generated as an obfuscation of $\mathsf{Prog}_{sk_1,\mathsf{pk},CK,f}$ instead of $\mathsf{Prog}_{sk_0,\mathsf{pk},CK,f}$.

In the following we show that the program Prog is equivalent under sk_0 and sk_1 with overwhelming probability. This allows us to conclude that the Hybrid 6 and Hybrid 7 are indistinguishable based on indistinguishability obfuscation.

Lemma 6.8. $\mathsf{Prog}_{sk_0,\mathsf{pk},CK,f} \equiv \mathsf{Prog}_{sk_1,\mathsf{pk},CK,f}$.

Proof. Recall that the underlying language $\mathcal{L}$ of the Double Key deniable encryption scheme consists of two languages, namely $\mathcal{L}_1$ and $\mathcal{L}_2$. Note that since the NIZK has statistical soundness with overwhelming probability over the choices of σ we have that all ciphertexts with an accepting proof must be from one of the two languages. We refer to the two types of ciphertexts corresponding to the language $\mathcal{L}_1$ and $\mathcal{L}_2$, as Type-1 and Type-2 ciphertext, respectively.

Recall that the program Prog takes $\{\beta_j\}_{j\in[n]}$ and $\{c_j, c'_j, \pi_j\}_{i\in[n]}$ as input. Recall from Figure 2 that in Step 1, Prog checks to see that all the proofs π_i are accepting and otherwise it outputs $\perp$. This means that for the program to do anything interesting all the proofs must be valid. Next we will show that in such cases the output of the program is identical regardless of whether sk_0 or sk_1 is used.

All ciphertexts are of Type-1: In this case, c_j and c'_j for $j \in [n]$ encrypted under pk_0 and pk_1 respectively, encrypt the same value. Hence, regardless of whether sk_0 is used or sk_1 is used the program outputs the exact same value $f(\{x_j\}_{j\in[n]})$.

There is at least one Type-2 ciphertext: Note that, in case sk_0 is used then we have that Step 2 of Prog is never invoked. On the other hand in case sk_1 is used then we have that Step 2 of Prog is necessarily invoked.

In other words if sk_0 is used then the x_j values are decrypted and output is calculated. On the other hand if sk_1 is used then a hard-coded *out* value is generated. We will argue that in both cases the output generated by Prog is identical. We argue this by showing that the only acceptable value for the hard-coded value *out* is $f(\{x_j\}_{j\in[n]})$, where x_j are the inputs parties commit to in the first round. Recall that the commitment scheme is perfectly binding, meaning that for every commitment β_i there is exactly one x_i such that $(\beta_i, x_i) \in \mathcal{L}_{\mathsf{COM}}$. This proves our claim. ∎

∎

Hybrid H_8: In this hybrid we do the same change that was made in going from Hybrid 2 to Hybrid 3. In this hybrid, we change how σ, which is a part of pk, and the proofs π_i for every $P_i \in \mathcal{P}\backslash\mathcal{P}^{\mathcal{A}}$ are generated.

More specifically, $\mathcal{S}$ runs the setup algorithm $\mathsf{Setup}_{\mathsf{DDK}}(1^\lambda, 1^{\ell_{in}+\ell_{out}})$ of the Double Key deniable encryption scheme, but replaces its internal call to the algorithm K with $S = (S_1, S_2)$ of the NIZK proof system. More specifically, $\mathcal{S}$ generates $(pk_0, sk_0), (pk_1, sk_1) \leftarrow \mathsf{Setup}(1^\lambda)$, $(\sigma, \tau) \leftarrow S_1(1^\lambda)$, along with the public denying key DK. It sets the public key $\mathsf{pk} = (\sigma, pk_0, pk_1, DK)$.

We also generate fake proofs π_i using trapdoor τ. Specifically, it generates $\pi_i \leftarrow S_2(\sigma, \tau, (c_i, c_i', (i, \{\beta_j\}_{j\in[n]})))$.

Lemma 6.9. $\mathrm{Hybrid}_7 \approx_c \mathrm{Hybrid}_8$.

Proof. The indistinguishability of Hybrid_7 from Hybrid_8 follows immediately from the computational zero-knowledge of the NIZK proof system. ∎

Hybrid H_9: In this hybrid, we change how the simulator $\mathcal{S}$ generates c_j in the second round message (c_j, c_j', π_j) on behalf of honest parties $P_j \in \mathcal{P}\backslash\mathcal{P}^{\mathcal{A}}$. More specifically, $\mathcal{S}$ instead of computing $c_j = \mathsf{Enc}(pk_0, x_i||\phi^{\ell_{out}})$, it computes $c_j = \mathsf{Enc}(pk_0, \phi^{\ell_{in}}||out)$ where $out = f(\{x_j\}_{j\in[n]})$.

Lemma 6.10. $\mathrm{Hybrid}_8 \approx_c \mathrm{Hybrid}_9$.

Proof. We base the indistinguishability between hybrids Hybrid_8 and Hybrid_9 on the semantic security of the encryption scheme, $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$. ∎

Hybrid H_{10}: In this hybrid we change the way the public parameters of the commitment scheme COM are generated. In particular, $\mathcal{S}$ runs the setup algorithm $\mathsf{Setup}^{\mathsf{equiv}}_{\mathsf{Com}}(1^\lambda)$ (instead of $\mathsf{Setup}^{\mathsf{bind}}_{\mathsf{Com}}(1^\lambda)$) of the adaptively secure commitment scheme COM and obtains (CK, μ, ν) where the trapdoor μ is still being used for extraction of adversary's inputs.

Lemma 6.11. $\text{Hybrid}_9 \approx_c \text{Hybrid}_{10}$.

Proof. Indistinguishability between hybrids Hybrid_9 and Hybrid_{10} follows from the indistinguishability of the public parameters of the commitment scheme COM. ■

Hybrid H_{11}: In this hybrid we change the way $\mathcal{S}$ generates the commitments on behalf of the honest parties. In particular we will remove the inputs and make these commitments equivocal. More specifically, for every party $P_i \in \mathcal{P} \backslash \mathcal{P}^{\mathcal{A}}$ the first round message is computed by $\mathcal{S}$ running $(\beta_i, state_i) \leftarrow \mathsf{Equiv}_1(CK, \nu)$. If the party later gets corrupted then $\mathcal{S}$ will produce randomness ω_i to equivocate the commitment β_i to the prescribed input x_i. To this end, $\mathcal{S}$ will run $\omega_i = \mathsf{Equiv}_2(CK, \nu, \beta_i, state_i, x_i)$.

Lemma 6.12. $\text{Hybrid}_{10} \approx_c \text{Hybrid}_{11}$.

Proof. We base the indistinguishability between hybrids Hybrid_{10} and Hybrid_{11} on the polynomial equivocality of the adaptively secure commitment scheme COM. ■

Note that Hybrid_{11} is identical to the simulation strategy described in Section 6.1.5. This concludes the proof.

6.2 Constant-Round MPC for all-but-one Adaptive Corruptions

Overview. The problem of constructing adaptively secure constant-round MPC protocols against arbitrary corruptions is considered a notorious hard problem. A recent line of works based on indistinguishability obfuscation [GP15] (see Section 6.1) constructs such protocols with optimal number of rounds against arbitrary corruptions. However, based on standard assumptions, adaptively secure protocols secure against even just all-but-one corruptions with near-optimal number of rounds are not known. In this work we provide a three-round solution based only on LWE and NIZK secure against all-but-one corruptions. Asharov et al. [AJL$^+$12] and more recently Mukherjee and Wichs [MW16] presented two-round protocols based on LWE which are secure *only* in the presence of static adversaries.

Our protocols are constructed based on a special type of cryptosystem we call equivocal FHE from LWE which is the focus of this section. In the full version of [DPR16] we provide our full protocols and also build adaptively secure UC commitments and UC zero-knowledge proofs (of knowledge) from LWE. Moreover, in the decryption phase using an AMD code mechanism we avoid the use of ZK and achieve communication complexity that does not scale with the decryption circuit.

6.2.1 Techniques

To construct our adaptively secure protocol, we start from the well known blue-print for FHE-based MPC: players encrypt their inputs under a common public key, evaluate the desired function locally and then jointly decrypt the result. This is possible under an appropriate set-up assumption, which is always needed for UC security and dishonest majority. Namely, we assume that a public key has been distributed, and players have been given shares of the corresponding secret key.

This approach has been used before and usually leads to static security. One reason for this is that encryptions are usually committing, so we are in trouble if the sender of a ciphertext is corrupted later. This can be solved using a cryptosystem with equivocal properties and this would mean that the input and the evaluation phase of the protocol can be simulated, even for adaptive corruptions. Players need, of course, to prove that they know the inputs they contribute, but this can be done once we construct constant round adaptively secure UC commitment and ZK proofs from LWE.

An important tool we would like to get in order to achieve constant-round adaptively secure MPC protocols may be a Fully Homomorphic Encryption (FHE) scheme with equivocal properties.

Starting point – Fully Homomorphic NCE. It is tempting to consider a generic solution from FHE and Non-Commiting Encryption (NCE). In particular, in such a hypothetical construction, the secret key would be a secret key for an FHE scheme, the public key an FHE encryption of the NCE secret key and the NCE public key. Encryption would be performed using the NCE, and homomorphic evaluation and decryption would be performed as expected. However, there are fundamental caveats with this approach.

It does not seem to buy us any efficiency at all. In particular, NCE schemes are interactive, in that the receiver must send fresh (public-)key material for each new message to be encrypted. There is even a result by Nielsen saying that this is inherent for NCE [Nie02]. It will be hard for an interactive scheme to fit the above suggestion. Indeed, the public key material would run out after encrypting some number of inputs. Therefore, in generic NCE the public-key cannot be reused, and has to be updated for each new message. Moreover, one may go around this issue by having an NCE public-key for each party where the FHE encryption in the public key will include all the public keys. However, such a solution is highly inefficient since it is not the number of parties that matter but the amount of data to be encrypted. The amount of public-key material has to be proportional to size of the plaintext data. For instance, if only a constant number of parties had input, but a lot of, we would have a significant problem.

Another suggestion is to always regenerate this setup afresh using a constant round adaptive protocol prior to each new execution. This might work but unfortunately set-up data are considered reasonable if its size does not depend on the function to be computed (otherwise we are in the preprocessing model which is a completely different ball game). Hence, one would in fact always need this key regeneration step per execution.

It turns out that the motivation of considering NCE in this context is very weak.

Our approach – Starting afresh. Towards minimising the above caveat we propose a scheme we call Equivocal FHE. An equivocal FHE scheme is a fully homomorphic encryption scheme with additional properties. Most importantly, it is possible to generate "fake" public keys that look like normal keys but where encryption leads to ciphertexts that contain no information on the plaintext. This is similar to the known notion of meaningful/meaningless keys, but in addition we want that fake public keys come with a trapdoor that allows to "explain" (equivocate) a ciphertext as an encryption of any desired plaintext. This is similar to (but not the same as) what is required for NCE: for NCE one needs to equivocate a ciphertext even if the decryption key is also given (say, by corrupting the receiver), here we only need to give the adversary valid looking randomness for the encryption. In order to achieve such a cryptosystem the main properties we require from an FHE scheme is formula privacy, invertible sampling and homomorphishm over the randomness. Given this, we managed to obtain the required equivocation directly with much less overhead compared to a possible NCE solution.

We give a concrete instantiation of equivocal FHE based on LWE, starting from the FHE scheme by Brakerski et al. [BV11b].

Achieving a simulatable protocol. A harder problem is how to simulate the output phase in which ciphertexts containing the outputs are decrypted. In the simulation we cannot expect that these ciphertexts are correctly formed and hold the actual outputs, so the simulator needs to "cheat". However, each player holds a share of the secret key which we have to give to the adversary if he is corrupted. If this happens after some executions of the decryption protocol, we (the simulator) may already be committed to this share. It is therefore not clear how the simulator can achieve the desired decryption results by adjusting the shares of the secret key.

To get around this, we adapt an idea from Damgård and Nielsen [DN03], who proposed an adaptively secure protocol based on additively homomorphic threshold encryption but in the honest majority scenario. The idea is to add a step to the protocol where each ciphertext is re-randomised just before decryption. This gives the simulator a chance to cheat and turn the ciphertext into one that contains the correct result, and one can therefore simulate the decryption without having to modify the shares of the secret key. The re-randomisation from [DN03] only works for honest majority, we show a different method that works for dishonest majority and augment our Equivocal FHE scheme with the *ciphertext randomisation* property to achieve our goal.

General purpose Equivocal FHE. We mention for completeness that there is also a more generic approach which will give us adaptive security based only on our Equivocal FHE: namely, we follow the same blueprint as before, with input, evaluation and output phases. However, we implement the verification of ciphertexts in the input phase and the decryptions in the output phase using generic adaptively secure MPC a la [CLOS02, IPS08]. This way, the communication and the number of rounds do not depend on the size of circuit to be computed securely. However, it would not be genuinely constant round, and the communication complexity would depend on the circuits computing the encryption and decryption functions of the underlying cryptosystem. Hence, unlike our protocol, any such solution would have communication complexity proportional to the Boolean circuit complexity of the decryption function (which seems inherent since one needs Yao garbling underneath). We measure the round and communication complexity of such a possible solution based on the IPS compiler. The bottom line is that using IPS generically would yield a larger (constant) number of rounds (20-30 rounds) and worse dependence on the security parameter. A concise estimate can be found in the full version. Clearly the above estimate should be taken with large grains of salt. We have tried to be optimistic on the part of IPS, to not give our concrete protocol an unfair advantage. Thus, actual numbers could be larger. On the other hand, we propose a three-round solution.

AMD code solution to replace ZK. However, contrary to the above generic IPS solution, our approach allows for significant optimization of the decryption as follows. Instead of using ZK proofs to prove that the player's evaluation shares to the decryption phase are correct, we change the evaluation phase of the protocol. In particular, instead of having ciphertexts containing the desired output z, the evaluation phase computes encryptions containing a codeword $c = (z, \alpha)$ in an algebraic manipulation detection code, where z is the data and α is the key/randomness. In the decryption stage, players commit to their decryption shares (recall that we have UC commitment available), and then all shares are opened. If decryption fails, or decoding the codeword fails, we abort, else we output the decoded z. If z and α are thought of as elements in a (large) finite field, then the codeword can just be $(z, \alpha, \alpha z)$. According to our optimization, the communication complexity of our protocol is not only independent of the the size of the evaluated circuit but also independent of the circuit size of the decryption circuit.

UC adaptive commitments and ZKPoK from LWE. A tool we need for our MPC protocol is constant-round UC-secure commitments and zero-knowledge proofs. For the commitments we start from a basic construction appeared in [CLOS02], which was originally based on claw-free trapdoor permutations (CFTP). We show that it can be instantiated based on LWE (which is not known to imply CFTP).

Commitment schemes that satisfy both equivocality and extractability form useful tools in achieving adaptive security. We show how to use an equivocal scheme based on the LWE assumption to build equivocal and extractable commitments. Note that such commitments based on LWE can be of independent interest. We remark that any encryption scheme that satisfies the properties specified in Definition 6.6 suffice for our purposes – the multiplicative homomorphic property of our QFHE scheme will not be of use here; however, since we are using our commitment scheme as a tool in our adaptive MPC protocol based on LWE, we use the same QFHE scheme in our commitment scheme too.

Since we are interested in UC security against adaptive adversaries, our commitment scheme is in the CRS model. The scheme must satisfy the following two properties, polynomial equivocality and simulation extractability. The former guarantees that the simulator $\mathcal{S}$ needs to be able to produce polynomially many equivocal commitments using the same CRS. More specifically, $\mathcal{S}$ can open the equivocal commitments to any value of its choice and give consistent randomness to adversary $\mathcal{A}$. The latter property says that the simulator $\mathcal{S}$ needs to be able to extract the contents of any valid commitment generated by adversary $\mathcal{A}$, even after $\mathcal{A}$ obtains polynomially many equivocal commitments generated by $\mathcal{S}$. Note that there is only an apparent conflict between equivocality and the binding property and between the extractability and the hiding property, as the simulator is endowed with additional power (trapdoors) in comparison with the parties in the real world execution. In the following we overview our technique on how our commitment scheme satisfies the above properties.

Equivocation in our scheme is achieved via QFHE. In particular, the commitment algorithm is simply the encryption algorithm of a QFHE scheme. In order to add extractability we must enhance our scheme in such a way that we do not sacrifice equivocality. A failed attempt is to include a public key for an encryption scheme secure against CCA2 attacks in the CRS. In this case, the committer will send an encryption of the decommitment information along with the commitment itself. Then, as the simulator has the associated decryption key, it can decrypt the decommitment information and hence extract the committed value from any adversarially prepared commitment. However, notice that such an encryption is binding even to the simulator, so equivocality cannot be achieved.

The solution to the problem is to send the commitment along with two pseudorandom ciphertexts. One ciphertext is an encryption of the decommitment information and the other ciphertext is a uniformly random string. In this way, the simulator can encrypt both decommitment values and later show that it only knows the decryption to one and that the other was uniformly chosen.

For the security of our construction, the encryption scheme used to encrypt the decommitment information has to be a CCA2-secure encryption scheme with the property that any produced ciphertext is pseudorandom and has deterministic decryption. To this end, the CCA2 encryption scheme of Micciancio and Peikert [MP12] based on LWE satisfies the above properties. They obtain their result via relatively generic assumptions using either strongly unforgeable one-time signatures [DDN91a], or a message authentication code and a weak form of commitment [BCHK07]. The first assumption does not yield pseudorandom ciphertexts, thus another encryption producing pseudorandom ciphertexts on top of the scheme of [MP12] could have been used, resulting in a double encryption scheme. However, it turns out that their construction with the latter set of assumptions has pseudorandom ciphertexts.

The reader might have observed that this bears some resemblance with the trick used in the seminal work of [CLOS02], to achieve extractability. Their scheme is based on enhanced trapdoor permutations, also needed in order to get double encryption CCA2 security. Moreover, in order to build equivocal commitments they need an NP reduction to graph Hamiltonicity since the CRS of their commitment scheme consists of a graph G sampled from a distribution such that it is computationally hard to tell if G has a Hamiltonian cycle. Interestingly, the CLOS commitment scheme does not give an instantiation based on LWE and to begin with, there are no known trapdoor permutations based on LWE. On the other hand, assuming the hardness of LWE, we propose an extractable and equivocal commitment *with no need of an NP reduction*, leading to a huge improvement in efficiency.

Our UC commitment scheme serves towards the realization of a commit-and-prove functionality based on LWE. Such a functionality is generic and hence is quite useful – it allows a party to prove NP statements relative to its commitment value in the setting where parties commit to their inputs but they never decommit. Using the power of the UC commitment scheme, commit-and-prove follows quite easily from known techniques.

In section 6.2.2 we define our *Equivocal fully homomorphic encryption* QFHE scheme and its properties. A concrete example of equivocal FHE based on the scheme of [BV11b] is given in the full version. However, any FHE scheme that satisfies the properties in Definition 6.6 can be equivocal.

Notation. We write $\boxplus$ and $\boxdot$ to denote operations over encrypted data including multiplication of a ciphertext with a non encrypted string. For a randomized algorithm A, we use $a \leftarrow A(x;r)$ to denote running A on input x and uniformly random bits $r \in \{0,1\}^*$, producing output a.

Invertible Sampling [OPW11]: We recall the notion of invertible sampling, which is closely connected to adaptive security in simulation models where erasures are not allowed. We say that an algorithm A with input space X has invertible sampling if there exists a PPT inverting algorithm, denoted by Inv_A, such that for all input $x \in X$, the outputs of the following two experiments are either computationally, or statistically indistinguishable:

$$\begin{array}{l|l} y \leftarrow A(x,r) & y \leftarrow A(x,r) \\ & r' \leftarrow \mathsf{Inv}_A(y,x) \\ \text{Return } (x,y,r) & \text{Return } (x,y,r') \end{array}$$

6.2.2 Equivocal Fully Homomorphic Encryption Scheme

We start by recalling the notions of (fully) homomorphic encryption. Next we define the new notion of Equivocal FHE and we specify the properties needed for such an instantiation. We give a concrete instantiation of our Equicocal FHE scheme from the LWE assumption, based on Brakerski and Vaikutanathan [BV11b] FHE scheme, in the full version.

6.2.2.1 Homomorphic Encryption

A homomorphic encryption scheme $\mathsf{HE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ is a quadruple of PPT algorithms. In this work, the message space M of the encryption schemes will be some (modulo 2) ring, and the functions to be evaluated will be represented as arithmetic circuits over this ring, composed of addition and multiplication gates. The syntax of these algorithms is given as follows.

- *K*ey-Generation. The algorithm KeyGen, on input the security parameter 1^λ, outputs $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$, where pk is a public encryption key and sk is a secret decryption key.
- *E*ncryption. The algorithm Enc, on input pk and a message $m \in M$, outputs a ciphertext $\mathrm{ct} \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m)$.
- *D*ecryption. The algorithm Dec on input sk and a ciphertext ct, outputs a message $\tilde{m} \leftarrow \mathsf{Dec}_{\mathsf{sk}}(\mathrm{ct})$.
- *H*omomorphic-Evaluation. The algorithm Eval, on input pk, an arithmetic circuit ckt, and a tuple of ℓ ciphertexts $(\mathrm{ct}_1, \ldots, \mathrm{ct}_\ell)$, outputs a ciphertext $\mathrm{ct}' \leftarrow \mathsf{Eval}_{\mathsf{pk}}(\mathrm{ckt}(\mathrm{ct}_1, \ldots, \mathrm{ct}_\ell))$.

We note that we can treat the evaluation key as a part of the public key. The security notion needed in this work is security against chosen plaintext attacks (IND-CPA security), defined as follows.

Definition 6.3 (IND-CPA security)**.** A scheme HE is IND-CPA secure if for any PPT adversary Adv it holds that:

$$\mathsf{Adv}^{\mathsf{CPA}}_{\mathsf{HE}}[\lambda] := |Pr[\mathsf{Adv}(\mathsf{pk}, \mathsf{Enc}_{\mathsf{pk}}(0)) = 1] - Pr[\mathsf{Adv}(\mathsf{pk}, \mathsf{Enc}_{\mathsf{pk}}(1)) = 1]| = \mathsf{negl}(\lambda),$$

where, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$.

6.2.2.2 Fully Homomorphic Encryption

A scheme HE is fully homomorphic if it is both compact and homomorphic with respect to a class of circuits. More formally:

Definition 6.4 (Fully homomorphic encryption)**.** A homomorphic encryption scheme $\mathsf{FHE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ is *fully homomorphic* if it satisfies the following properties:

1. *Homomorphism:* Let $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ be the set of all polynomial sized arithmetic circuits. $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$, $\forall \mathrm{ckt} \in \mathcal{C}_\lambda$, $\forall (m_1, \ldots, m_\ell) \in M^\ell$ where $\ell = \ell(\lambda)$, $\forall (\mathrm{ct}_1, \ldots, \mathrm{ct}_\ell)$ where $\mathrm{ct}_i \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m_i)$, it holds that:

$$Pr[\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Eval}_{\mathsf{pk}}(\mathrm{ckt}, \mathrm{ct}_1, \ldots, \mathrm{ct}_\ell)) \neq \mathrm{ckt}(m_1, \ldots, m_\ell)] = \mathsf{negl}(\lambda)$$

2. *Compactness:* There exists a polynomial $\mu = \mu(\lambda)$ such that the output length of Eval is at most μ bits long regardless of the input circuit ckt and the number of its inputs.

6.2.2.3 Equivocal Fully Homomorphic Encryption Scheme

Our *Equivocal fully homomorphic encryption scheme* consists of a tuple ($\mathsf{KeyGen}, \mathsf{KeyGen}^*, \mathsf{QEnc}, \mathsf{Rand}, \mathsf{Eval}, \mathsf{Dec}, \mathsf{Equiv}$) of algorithms where the syntax of the procedures ($\mathsf{KeyGen}, \mathsf{QEnc}, \mathsf{Eval}, \mathsf{Dec}$) is defined as in the above FHE scheme. Our scheme is augmented with two algorithms ($\mathsf{KeyGen}^*, \mathsf{Equiv}$) used for equivocation. Jumping ahead, in this paper we are interested in building adaptively secure n-party protocols generically using an equivocal QFHE scheme and gain in terms of round and communication efficiency. Two extra properties needed for the MPC purpose, are distributed decryption and ciphertext randomisation where the latter one guarantees simulatable decryption [1]. If the purpose of our Equivocal scheme is not MPC then these properties are not required,.In the sequel, we will use blue color to stress whether a part is relevant to the ciphertext randomisation property.

Definition 6.5 (Equivocal fully homomorphic encryption)**.** An Equivocal fully homomorphic encryption scheme $\mathsf{QFHE} = (\mathsf{KeyGen}, \mathsf{KeyGen}^*, \mathsf{QEnc}, \mathsf{Rand}, \mathsf{Eval}, \mathsf{Dec}, \mathsf{Equiv})$ with message space M is made up of the following PPT algorithms:

- ($\mathsf{KeyGen}, \mathsf{QEnc}, \mathsf{Eval}, \mathsf{Dec}$) is an FHE scheme with the same syntax as in section 6.2.2.1.
- The *Equivocal* key generation algorithm $\mathsf{KeyGen}^*(1^\lambda)$, outputs an *equivocal* public-key secret-key pair $(\widetilde{\mathsf{PK}}, \widetilde{\mathsf{SK}})$.
- The *Equivocation* algorithm $\mathsf{Equiv}(\widetilde{\mathsf{PK}}, \widetilde{\mathsf{SK}}, \mathrm{ct}, r_{\mathrm{ct}}, m)$, given $\widetilde{\mathsf{PK}}$, $\widetilde{\mathsf{SK}}$, a plaintext m, a ciphertext ct and random coins r_{ct}, outputs a value e in the randomness space.
- The *Ciphertext Randomisation* algorithm $\mathsf{Rand}(\mathrm{ct}, \mathrm{ct}'_1, \ldots, \mathrm{ct}'_n)$, given ciphertexts $\mathrm{ct}, \mathrm{ct}'_1, \ldots, \mathrm{ct}'_n$ generated by the procedure QEnc outputs a ciphertext CT.

  We require the following properties:

  1. *Indistinguishability of equivocal keys.* We say that the scheme has indistinguishability of equivocal keys if the distributions of PK and $\widetilde{\mathsf{PK}}$ are computationally indistinguishable, where $(\mathsf{PK}, \cdot) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and $(\widetilde{\mathsf{PK}}, \cdot) \leftarrow \mathsf{KeyGen}^*(1^\lambda)$.
  2. *Indistinguishability of equivocation.* Let $\mathcal{D}_{rand}(1^\lambda)$ denote the distribution of randomness used by QEnc. Let $\mathcal{O}(\widetilde{\mathsf{PK}}, m)$ and $\mathcal{O}'(\widetilde{\mathsf{PK}}, \widetilde{\mathsf{SK}}, m)$ be the following oracles:

$$
\begin{array}{l|l}
\textit{Let } \ \mathcal{O}(\widetilde{\mathsf{PK}}, m): & \textit{Let } \ \mathcal{O}'(\widetilde{\mathsf{PK}}, \widetilde{\mathsf{SK}}, m): \\
\quad r_{\mathrm{ct}} \leftarrow \mathcal{D}_{rand}(1^\lambda) & \quad r_{\mathrm{ct}} \leftarrow \mathcal{D}_{rand}(1^\lambda) \\
\quad \mathrm{ct} = \mathsf{QEnc}_{\widetilde{\mathsf{PK}}},(m; r_{\mathrm{ct}}) & \quad \mathrm{ct} = \mathsf{QEnc}_{\widetilde{\mathsf{PK}}}(\widetilde{m}; r_{\mathrm{ct}}) \\
 & \quad e = \mathsf{Equiv}(\widetilde{\mathsf{PK}}, \widetilde{\mathsf{SK}}, \mathrm{ct}, r_{\mathrm{ct}}, m) \\
\quad \textit{Output } (\widetilde{\mathsf{PK}}, \mathrm{ct}, r_{\mathrm{ct}}) & \quad \textit{Output } (\widetilde{\mathsf{PK}}, \mathrm{ct}, e)
\end{array}
$$

[1]Ciphertext randomisation is needed in order to force the output in the simulation.

There exists $\widetilde{m} \in M$ such that for any PPT adversary $\mathcal{A}$ with oracle access to $\mathcal{O}(\widetilde{\mathsf{PK}}, \cdot)$ and $\mathcal{O}'(\widetilde{\mathsf{PK}}, \widetilde{\mathsf{SK}}, \cdot)$ the following holds.

$$\left| \Pr\left[\begin{array}{l} (\widetilde{\mathsf{PK}}, \widetilde{\mathsf{SK}}) \leftarrow \mathsf{KeyGen}^*(1^\lambda) \\ 1 \leftarrow \mathcal{A}^{\mathcal{O}(\widetilde{\mathsf{PK}}, \cdot)} \end{array} \right] - \Pr\left[\begin{array}{l} (\widetilde{\mathsf{PK}}, \widetilde{\mathsf{SK}}) \leftarrow \mathsf{KeyGen}^*(1^\lambda) \\ 1 \leftarrow \mathcal{A}^{\mathcal{O}'(\widetilde{\mathsf{PK}}, \widetilde{\mathsf{SK}}, \cdot)} \end{array} \right] \right| \leq \mathsf{negl}(\lambda)$$

3. *Ciphertext Randomisation.* Let PK be the public key used in the procedure QEnc for generating ciphertexts $\mathrm{ct}, \mathrm{ct}'_1 \ldots \mathrm{ct}'_n$ from the plaintexts $m, m'_1, \ldots, m'_n \in M$, respectevely. If $Pr[\mathsf{Dec}_{\mathsf{sk}}(\mathrm{ct}) = m] = 1 - \mathsf{negl}(\lambda)$ and for all $i \in [n]$, $Pr[\mathsf{Dec}_{\mathsf{sk}}(\mathrm{ct}'_i) = m'_i] = 1 - \mathsf{negl}(\lambda)$ then it holds that

$$Pr[\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Rand}(\mathrm{ct}, \mathrm{ct}'_1 \ldots \mathrm{ct}'_n)) = m] = 1 - \mathsf{negl}(\lambda).$$

On the other hand, let $\widetilde{\mathsf{PK}}$ be the public key used in the procedure QEnc for generating ciphertexts $\mathrm{ct}, \mathrm{ct}'_1 \ldots \mathrm{ct}'_n$, respectevely. If $Pr[\mathsf{Dec}_{\mathsf{sk}}(\mathrm{ct}) = m] = 1 - \mathsf{negl}(\lambda)$ and for all $i \in [n]$, $Pr[\mathsf{Dec}_{\mathsf{sk}}(\mathrm{ct}'_i) = m'_i] = 1 - \mathsf{negl}(\lambda)$ then it holds that

$$Pr[\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Rand}(\mathrm{ct}, \mathrm{ct}'_1 \ldots \mathrm{ct}'_n)) = m'_1 + \ldots + m'_n] = 1 - \mathsf{negl}(\lambda).$$

In the sequel for simplicity of exposition, we call the ciphertexts $\mathrm{ct}'_1 \ldots \mathrm{ct}'_n$ `redundant` in case they are generated by $\mathsf{QEnc}_{\mathsf{PK}}$ and $\mathtt{non-redundant}$ if they are generated by $\mathsf{QEnc}_{\widetilde{\mathsf{PK}}}$. Analogously, we call the ciphetext ct $\mathtt{non-redundant}$ or `redundant` if it is generated by $\mathsf{QEnc}_{\mathsf{PK}}$ or $\mathsf{QEnc}_{\widetilde{\mathsf{PK}}}$, respectively [2].

In order to construct our equivocal QFHE scheme we use the following *special* FHE scheme with some additional properties.

Definition 6.6. [Special fully homomorphic encryption] We call a fully homomorphic encryption scheme $\mathsf{FHE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ a *special* FHE scheme, if it is IND-CPA secure and satisfies the following properties: Let $\mathcal{D}_{rand}(1^\lambda)$ denote the distribution of randomness used by Enc.

1. *Additive homomorphism over random coins*: $\forall r_1, r_2 \in \mathsf{Supp}(\mathcal{D}_{rand}(1^\lambda))$ and $\forall m \in M$, it holds that $(m \boxdot \mathsf{Enc}_{\mathsf{pk}}(0; r_1)) \boxplus \mathsf{Enc}_{\mathsf{pk}}(0; r_2) = \mathsf{Enc}_{\mathsf{pk}}(0; m \cdot r_1 + r_2)$.

2. *E-Hiding*: There exists $\mathcal{D}'_{rand}(1^\lambda)$ such that $\forall m \in M$, if $r^{blind} \leftarrow \mathcal{D}_{rand}(1^\lambda)$ and $r^K \leftarrow \mathcal{D}'_{rand}(1^\lambda)$ then the distribution of $(r^{blind} - m \cdot r^K)$ is statistically close to $\mathcal{D}_{rand}(1^\lambda)$. [3]

3. *Invertible Sampling*: The distribution $\mathcal{D}_{rand}(1^\lambda)$, has invertible sampling via the algorithm $\mathsf{Inv}_{\mathcal{D}_{rand}}$.

[2]By the ciphertext randomisation property, the reader can think of the `redundant` messages as encryptions of zeros.

[3]Intuitively, E-Hiding can be argued in the same way as formula privacy for some FHE schemes. This requires *dwarfing* in the sense that r^{blind} should be *large* enough to dwarf mr^K where $\mathcal{D}_{rand}(1^\lambda)$ and $\mathcal{D}'_{rand}(1^\lambda)$ are Gaussian distributions. Hence, $r^K \leftarrow \mathcal{D}'_{rand}(1^\lambda)$ and $r^{blind} \leftarrow \mathcal{D}_{rand}(1^\lambda)$ such that the noise of $\mathcal{D}_{rand}(1^\lambda)$ is super-polynomially larger than the noise of $\mathcal{D}'_{rand}(1^\lambda)$.

Recall that we defined an invertible sampler of an algorithm A in Section 6.2.1 as an algorithm Inv_A that takes as inputs the input x and output y with consistent random coins. In our case, $x = 1^\lambda$ and y is a sample from the range of $\mathcal{D}_{rand}$. Next, in Figure 6.3, we show how to build an equivocal FHE scheme using a special FHE scheme. The high level intuition is as follows. In order to achieve equivocality we modify an FHE scheme satisfying the properties of Definition 6.6 as follows: The public key contains an encryption of 1 and an encryption of 0. More specifically, $\mathsf{PK} = (\mathsf{pk}, K = \mathsf{Enc}_{\mathsf{pk}}(1), R = \mathsf{Enc}_{\mathsf{pk}}(0))$ where pk is the public key of an FHE scheme. An encryption of a message m in the real world is computed using K as $(m \boxdot K \boxplus \mathsf{Enc}_{\mathsf{pk}}(0))$ and encryption for re-randomisation is computed using R as $(z \boxdot R \boxplus \mathsf{Enc}_{\mathsf{pk}}(0))$ for a random value z. In the simulation, the values encrypted in K and R are switched, in particular, $K = \mathsf{Enc}_{\mathsf{pk}}(0)$ and $R = \mathsf{Enc}_{\mathsf{pk}}(1)$. Therefore, normal encryption leads to encryption of 0 with the guarantee of equivocation. However, encryption for re-randomisation actually encrypts non-zero values i.e., z, in order to force the output.

Theorem 6.13. Let FHE be a *special* fully homomorphic encryption scheme. Then $\mathsf{QFHE} = (\mathsf{KeyGen}, \mathsf{KeyGen}^*, \mathsf{QEnc}, \mathsf{Rand}, \mathsf{Eval}, \mathsf{Dec}, \mathsf{Equiv})$ in Figure 6.3 is an equivocal QFHE scheme.

Proof. Indistinguishability of equivocal keys. Let $(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and $(\widetilde{\mathsf{PK}}, \widetilde{\mathsf{SK}}) \leftarrow \mathsf{KeyGen}^*(1^\lambda)$, then the indistinguishability of the two pairs of public keys follows from the IND-CPA security of the FHE scheme.

Indistinguishability of equivocation. Without loss of generality, we will show that indistinguishability of equivocation holds for $\widetilde{m} = 0$. Let $\mathcal{A}$ be an adversary that breaks indistinguishability of equivocation; then we construct a PPT algorithm R such that $R^{\mathcal{A}}$ breaks *E-hiding*. R simulates the oracle for every query m_i as follows. R invokes $\mathcal{A}$ and receives some message m_i and forwards it to the *E-hiding* challenger. Next it receives the challenge r_{ct_i} and computes $\mathrm{ct}_i = \mathsf{QEnc}_{\widetilde{\mathsf{PK}}}(0, m_i; r_{\mathrm{ct}_i})$ and forwards $(r_{\mathrm{ct}_i}, \mathrm{ct}_i)$ to $\mathcal{A}$ and outputs whatever $\mathcal{A}$ does. Now, if $r_{\mathrm{ct}_i} \leftarrow \mathcal{D}_{rand}(1^\lambda)$ then $\mathrm{ct}_i \leftarrow \mathsf{QEnc}_{\widetilde{\mathsf{PK}}}(0, m_i; r_{\mathrm{ct}_i})$, namely, the view of $\mathcal{A}$ follows the distribution which corresponds to the left game in Definition 6.5 of indistinguishability of equivocation. On the other hand, if $r_{\mathrm{ct}_i} = (r_i^{blind} - m_i \cdot r^{\widetilde{K}})$; then $\mathrm{ct}_i = (m_i \boxdot \widetilde{K}) \boxplus \mathsf{Enc}_{\mathsf{pk}}(0; r_i^{blind} - m_i \cdot r^{\widetilde{K}}) = \mathsf{Enc}_{\mathsf{pk}}(0; r_i^{blind}) = \mathsf{QEnc}_{\widetilde{\mathsf{PK}}}(0, 0; r_i^{blind})$ which implies that in this case the view of $\mathcal{A}$ follows the distribution of the right game in Definition 6.5 of indistinguishability of equivocation. This means that the distinguishing advantage of R is the same as that of Adv which leads to a contradiction.

Ciphertext Randomisation. The algorithm Rand adds the ciphertexts $(\mathrm{ct}, \mathrm{ct}'_1, \ldots, \mathrm{ct}'_n)$. If ct is a ciphertext generated by $\mathsf{QEnc}_{\mathsf{PK}}$ for $b = 0$ and $(\mathrm{ct}'_1 \ldots \mathrm{ct}'_n)$ are ciphertexts generated by $\mathsf{QEnc}_{\mathsf{PK}}$ for $b = 1$ then

$$Pr[\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Rand}(\mathrm{ct}, \mathrm{ct}'_1 \ldots \mathrm{ct}'_n)) = m] = 1 - \mathsf{negl}(\lambda)$$

since it is easy to see that the ciphertexts $(\mathrm{ct}'_1 \ldots \mathrm{ct}'_n)$ contain encryptions of zeros due to the fact that $R = \mathsf{Enc}_{\mathsf{pk}}(0)$. An analogous argument holds for ct and $\mathrm{ct}'_1 \ldots \mathrm{ct}'_n$ generated by $\mathsf{QEnc}_{\widetilde{\mathsf{PK}}}$ for $b = 0$ and $b = 1$, respectively, since in this case the ciphertext ct contain

an encryption of a zero (because in this case $\widetilde{K} = \mathsf{Enc}_{\mathsf{pk}}(0)$) and ciphertexts $(\mathrm{ct}'_1 \ldots \mathrm{ct}'_n)$ contain encryptions of the corresponding m'_i since $\widetilde{R} = \mathsf{Enc}_{\mathsf{pk}}(1)$.
■

Distributed Decryption: As we mentioned above, we need distributed decryption to implement our MPC protocol. To this end, we assume that the common public key pk has been set up where the secret key sk has been secret-shared among the players in such a way that they can collaborate to decrypt. Notice that some setup assumption is always required to show UC security in the dishonest majority setting. Roughly, we assume that a functionality is available which generates a key pair and secret-shares the secret key among the players using a secret-sharing scheme that is assumed to be given as part of the specification of the cryptosystem. Since we allow corruption of all but one player, the maximal unqualified sets must be all sets of $n - 1$ players. We point out that we could make a weaker set-up assumption, such as a common reference string, and using a general UC secure multiparty computation protocol for the common reference string model to implement the above functionality. While this may not be very efficient, one only needs to run this protocol once in the life-time of the system. The properties needed for the distributed decryption and its protocol are specified later.

QFHE

Let $\mathsf{FHE} = (\mathsf{KeyGen}_{\mathsf{FHE}}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ be a *special* fully homomorphic encryption scheme. $\mathsf{QFHE} = (\mathsf{KeyGen}, \mathsf{KeyGen}^*, \mathsf{QEnc}, \mathsf{Eval}, \mathsf{Rand}, \mathsf{Dec}, \mathsf{Equiv})$ is defined as follows:

$\mathsf{KeyGen}(1^\lambda)$**:**

1. $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}_{\mathsf{FHE}}(1^\lambda)$.
2. $K = \mathsf{Enc}_{\mathsf{pk}}(1; r^K)$ where $r^K \leftarrow \mathcal{D}'_{rand}(1^\lambda)$ and $R = \mathsf{Enc}_{\mathsf{pk}}(0; r^R)$ where $r^R \leftarrow \mathcal{D}'_{rand}(1^\lambda)$
3. Return as public key $\mathsf{PK} = (\mathsf{pk}, K, R)$ and secret key $\mathsf{SK} = \mathsf{sk}$.[a]

$\mathsf{KeyGen}^*(1^\lambda)$**:**

1. $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}_{\mathsf{FHE}}(1^\lambda)$.
2. $\widetilde{K} = \mathsf{Enc}_{\mathsf{pk}}(0; r^{\widetilde{K}})$ where $r^{\widetilde{K}} \leftarrow \mathcal{D}'_{rand}(1^\lambda)$ and $\widetilde{R} = \mathsf{Enc}_{\mathsf{pk}}(1; r^{\widetilde{R}})$ where $r^{\widetilde{R}} \leftarrow \mathcal{D}'_{rand}(1^\lambda)$.
3. Return as public key $\widetilde{\mathsf{PK}} = (\mathsf{pk}, \widetilde{K}, \widetilde{R})$ and secret key $\widetilde{\mathsf{SK}} = (\mathsf{sk}, r^{\widetilde{K}}, r^{\widetilde{R}})$.

$\mathsf{QEnc}_{\mathsf{PK}}(b, m)$:

1. Compute $\mathrm{ct}^{\mathsf{blind}} = \mathsf{Enc}_{\mathsf{pk}}(0; r^{\mathsf{blind}})$ where $r^{\mathsf{blind}} \leftarrow \mathcal{D}_{rand}(1^\lambda)$.
2. If $b \notin \{0, 1\}$ then output $\perp$.
3. If $b = 0$ then output $\mathrm{ct} = (m \boxdot K) \boxplus \mathrm{ct}^{\mathsf{blind}}$ *otherwise* output $\mathrm{ct} = (m \boxdot R) \boxplus \mathrm{ct}^{\mathsf{blind}}$.

$\mathsf{QEnc}_{\widetilde{\mathsf{PK}}}(b, \widetilde{m})$:

1. Compute $\widetilde{\mathrm{ct}}^{\mathsf{blind}} = \mathsf{Enc}_{\mathsf{pk}}(0; \widetilde{r}^{\mathsf{blind}})$ where $\widetilde{r}^{\mathsf{blind}} \leftarrow \mathcal{D}_{rand}(1^\lambda)$.
2. If $b \notin \{0, 1\}$ then output $\perp$.
3. If $b = 0$ then output $\widetilde{\mathrm{ct}} = (\widetilde{m} \boxdot \widetilde{K}) \boxplus \widetilde{\mathrm{ct}}^{\mathsf{blind}}$ *otherwise* output $\widetilde{\mathrm{ct}} = (\widetilde{m} \boxdot \widetilde{R}) \boxplus \widetilde{\mathrm{ct}}^{\mathsf{blind}}$.

$\mathsf{Equiv}(b, \widetilde{\mathsf{PK}}, \widetilde{\mathsf{SK}}, \widetilde{\mathrm{ct}}, \widetilde{r}^{\mathsf{blind}}, m, \widetilde{m})$**:**

1. If $b = 0$ compute $r^{\mathsf{blind}} := \widetilde{r}^{\mathsf{blind}} + (\widetilde{m} - m) \cdot r^{\widetilde{K}}$ *otherwise* $r^{\mathsf{blind}} := \widetilde{r}^{\mathsf{blind}} + (\widetilde{m} - m) \cdot r^{\widetilde{R}}$
2. Run $r_{state} \leftarrow \mathsf{Inv}_{\mathcal{D}_{rand}}(r^{\mathsf{blind}})$ and output r_{state}.

$\mathsf{Rand}(\mathrm{ct}, \mathrm{ct}'_1 \ldots, \mathrm{ct}'_n)$: Output $\mathrm{CT} = \mathrm{ct} \boxplus \mathrm{ct}'_1 \boxplus \ldots \boxplus \mathrm{ct}'_n$.

Procedures $(\mathsf{Eval}, \mathsf{Dec})$ are as defined in normal FHE schemes.

[a] Note that procedure Dec, given sk, runs as in normal FHE schemes (see Section 6.2.2.1), so there is no need to provide r^K in SK. We also enhance the notation of QEnc to include a bit b which indicates whether the encryption is performed using the key K or R, respectively. In addition, the plaintext $\widetilde{m}$ is usually set to zero.

Figure 6.3: Description of QFHE scheme [DPR16]

Part III

Results in the Information Theoretic Setting

Chapter 7

On the Communication Complexity of Gate-by-Gate Protocols

In this chapter we present our lower bounds for IT MPC protocols. In particular, we present our results in [DNPR16]. See Section 2.2.1 for a detailed overview of our contributions.

Overview. A plethora of IT secure protocols are known for general secure multi-party computation, in the honest majority setting, and in the dishonest majority setting with preprocessing. All known protocols that are efficient in the circuit size of the evaluated function follow the same "gate-by-gate" design pattern: we work through an arithmetic (boolean) circuit on secret-shared inputs, such that after we process a gate, the output of the gate is represented as a random secret sharing among the players. This approach usually allows non-interactive processing of addition gates but requires communication for every multiplication gate. Thus, while information-theoretic secure protocols are very efficient in terms of computational work, they (seem to) require more communication and more rounds than computationally secure protocols. Whether this is inherent is an open and probably very hard problem. However, in this chapter we show that it is indeed inherent for protocols that follow the "gate-by-gate" design pattern. We present the following results:

- In the honest majority setting, as well as for dishonest majority with preprocessing, any gate-by-gate protocol must communicate $\Omega(n)$ bits for every multiplication gate, where n is the number of players.

- In the honest majority setting, we show that one cannot obtain a bound that also grows with the field size. Moreover, for a constant number of players, amortizing over several multiplication gates does not allow us to save on the computational work, and – in a restricted setting – we show that this also holds for communication.

All our lower bounds are met up to a constant factor by known protocols that follow the typical gate-by-gate paradigm. Our results imply that a fundamentally new approach must be found in order to improve the communication complexity of known protocols, such as BGW, GMW, SPDZ etc.

Due to space constraints we refer to Section 2.2.2 for our contributions and high level techniques for malicious IT MPC constructions with optimal (up to a constant factor) communication complexity. More details can be found in [GIP15].

7.1 Techniques

Our Model. To avoid misunderstandings, let us be more precise about the model we assume: we consider synchronous protocols that are semi-honest and statistically secure against static corruption of at most t of the n players. We assume that point-to-point secure channels are available, and protocols are allowed to have dynamic communication patterns (in a certain sense we make precise later), i.e., it is not fixed a priori whether a protocol sends a message in a given time slot. Moreover, there is no bound on the computational complexity of protocols, in particular arbitrary secret sharing schemes are allowed. A *gate-by-gate* protocol is a protocol that evaluates an arithmetic circuit and for every multiplication gate, it calls a certain type of subprotocol we call a *Multiplication Gate Protocol* (MGP). We define MGPs precisely later, but they basically take as input random *shares* of two values a, b from a field and output random *shares* of $c = ab$. Neither the MGP nor the involved secret sharing schemes have to be the same for all gates. We do not even assume that the same secret sharing scheme is used for the inputs and outputs of an MGP, we only require that the *reconstruction threshold* for the output sharing is at most $2t$ for honest majority and at most n for dishonest majority.

An *ordered* gate-by-gate protocol must call the MGP's in an order corresponding to the order in which one would visit the gates when evaluating the circuit, whereas this is not required in general. Thus the gate-by-gate notion is somewhat more general than what one might intuitively expect and certainly includes much more than, say the standard BGW protocol – which, of course, makes our negative results stronger.

Note that if multiplications did not require communication, it would immediately follow (for semi-honest security) that we would have an unconditionally secure two-round protocol for computing any function. But as mentioned above this is not *a priori* impossible: it follows, for instance, from [IK00, IKM+13], that if less than a third of the players are corrupted, there is indeed such a two-round protocol (which, however, requires super polynomial computational work in general).

Honest Majority Setting. For honest majority protocols it is relatively easy to show that multiplications do require communication: we argue in the paper that any MGP secure against t corruptions requires that at least $2t + 1$ players communicate. For protocols with dynamic communication pattern this bound holds in expectation. It turns out that a protocol beating this bound would imply an unconditionally secure two-party protocol computing a multiplication, which is well known to be impossible. This implies that the communication complexity of any gate-by-gate protocol for honest majority must be proportional to $n \cdot s$ where s is the circuit size and that the round complexity of an ordered gate-by-gate protocol must be at least proportional to the multiplicative depth of the circuit. This matches the best protocols we know for general Boolean circuits up to a constant factor. For arithmetic circuits over large fields one might wonder whether the communication must grow with the field size. However, this cannot be shown via a general bound on MGPs: we give an example secret sharing scheme allowing for an MGP with communication complexity independent of the field size.

A gate-by-gate protocol is not allowed to amortise over several multiplications that can be done in parallel. This is anyway not possible in general, for instance if we evaluate a "tall and skinny" circuit forcing us to do multiplications sequentially. But for more benign circuits, amortization is indeed an option. However, we show that in a restricted setting, MGPs doing k multiplication gates in parallel must have communication that grows linearly with k. We also show (in full generality) that amortization can save at most an $O(n)$ factor in the computational work, matching what we can get from known techniques based on packed secret-sharing. This proof technique for this bound is quite interesting: We base it on a lower bound by Winkler and Wullschleger [WW10] on the amount of preprocessed data one needs for (statistically) secure two-party computation of certain functions. We find it somewhat surprising that an information theoretic bound on the size of data translates to a bound on local computation.

Dishonest Majority Setting with Preproccesing. The argument used for the honest majority case breaks down if we consider protocols in the preprocessing model (where correlated randomness is considered): here it is indeed possible to compute multiplications with unconditional security, even if $t = n-1$ of the n players are corrupt. Nevertheless, we show similar results for this setting: here, any MGP secure against $t = n-1$ corruptions must have all n players communicate. This implies that, also in this setting, any gate-by-gate protocol has communication complexity $\Omega(n \cdot s)$. Note that existing constructions [DPSZ12] meet the resulting bound for gate-by-gate protocols up to a constant factor.

To obtain the result, we exploit again the lower bound by Winkler and Wullschleger, but in a different way. In a nutshell, we show that constructions beating our bound would imply a protocol that is too good to be true according to [WW10].

The result holds exactly as stated above assuming that the target secret-sharing scheme that the protocol outputs shares in is of a certain type that includes the simple additive secret-sharing scheme (which is also used in [DPSZ12], [NNOB12]). If we put no restrictions on the target scheme, the results get a bit more complicated. Essentially what we show is the following: suppose we replace the multiplication gate by a more general gate that does some computation on a fixed number of inputs, such as the inner product of two vectors. Then we show that once the computation done by the gate gets large enough (in a certain sense we define in the paper), again a protocol handling such a gate must communicate a lot. It is the target secret-sharing scheme that determines how "large" the gate needs to be, see more details within.

7.2 Preliminaries in the IT setting

Notation. We say that a function ε is negligible if $\forall c\ \exists\ \sigma_c \in \mathbb{N}$ such that if $\sigma \geq \sigma_c$ then $\varepsilon(\sigma) < \sigma^{-c}$. We write $[n]$ to denote the set $\{1, 2, ..., n\}$. Moreover, calligraphic letters denote sets. The complement of a set $\mathcal{A}$ is denoted by $\overline{\mathcal{A}}$. The distribution of a random variable X over $\mathcal{X}$ is denoted by P_X. Given the distribution P_{XY} over $\mathcal{X} \times \mathcal{Y}$, the marginal distribution is denoted by $P_X(x) := \sum_{y \in \mathcal{Y}} P_{XY}(x, y)$. A conditional distribution $P_{X|Y}(x, y)$ over $\mathcal{X} \times \mathcal{Y}$ defines for every $y \in \mathcal{Y}$ a distribution $P_{X|Y=y}$. The *statistical distance* between two distributions P_X and P'_X over the domain $\mathcal{X}$ is defined as the maximum, over all (inefficient) distinguishers $D : \mathcal{X} \to \{0, 1\}$, of the distinguishing advantage $\mathrm{SD}(P_X, P'_X) = |Pr[D(X) = 1] - Pr[D(X') = 1]|$. The *conditional Shannon entropy* of X given Y is defined as $H(X|Y) := -\sum_{x,y} P_{XY}(x, y) \log P_{X|Y}(x, y)$ where all logarithms are binary and the *mutual information* of X and Y as $I(X; Y) = H(X) - H(X|Y)$. We also use $h(p) = -p \log p - (1-p) \log(1-p)$ for the binary entropy function. Furthermore, we denote by Π_f an n-party protocol for a function f and by $\Pi_f^{A,B}$ a two-party protocol between parties A and B.

Protocols. We consider protocols involving n parties, denoted by the set $\mathcal{P} = \{\mathsf{P}_1, \ldots, \mathsf{P}_n\}$. The parties communicate over synchronous, point-to-point secure channels. We consider non-reactive secure computation tasks, defined by a deterministic or randomized functionality $f : \mathcal{X}_1 \times \ldots \times \mathcal{X}_n \to \mathcal{Z}_1 \times \ldots \times \mathcal{Z}_n$. The functionality specifies a mapping from n inputs to n outputs the parties want to compute. The functionality can be fully specified by a conditional probability distribution $P_{Z_1 \cdots Z_n | X_1 \cdots X_n}$, where X_i is a random variable over $\mathcal{X}_i$, Z_i is a random variable over $\mathcal{Z}_i$, and for all inputs $(x_1, \ldots, x_n)$ we have a probability function $P_{Z_1 \cdots Z_n | X_1 \cdots X_n = (x_1, \ldots, x_n)}$ and $P_{Z_1 \cdots Z_n | X_1 \cdots X_n = (x_1, \ldots, x_n)}(z_1, \ldots, z_n)$ is the probability that the output is $(z_1, \ldots, z_n)$ when the input is $(x_1, \ldots, x_n)$. Vice versa, we can consider any conditional probability distribution $P_{Z_1 \cdots Z_n | X_1 \cdots X_n}$ as a specification of a probabilistic functionality. In the following we will freely switch between the terminology of probabilistic functionalities and conditional probability distributions.

We consider stand-alone security as well as static and passive corruptions of t out of n parties for some $t \leq n$. This means that a set of t parties are announced to be corrupted before the protocol is executed, and the corrupted parties still follow the protocol but might pool their views of the protocol to learn more than they should. We consider statistical correctness and statistical security. We allow simulators to be inefficient. Except that we do not consider computational security, the above model choices are the possible weakest ones, which just makes our impossibility proofs stronger.

The Security Parameter. The security is measured in a security parameter σ and we require that the "insecurity" goes to 0 as σ grows. We do not allow n to grow with σ, i.e., we require that the protocol can be made arbitrarily secure when run among a fixed set of parties by just increasing σ. The literature sometimes consider protocol which only become secure when run among a sufficiently large number of parties. We do not cover such protocols.

Communication Model. We assume that each pair of parties are connected by a secure communication channel, which only leaks to the adversary the length of each message sent[1]. We consider protocols proceeding in synchronous rounds. Following [DPP14] we assume that in each round each pair of parties $(\mathsf{P}_i, \mathsf{P}_j)$ will specify a prefix free code $M_{i,j} \subset \{0,1\}^*$ and then P_i will send a message $m \in M_{i,j}$. The codes might be dynamically chosen, but we require that the parties agree on the codes. If the length of a sent message does not match the length specified by the receiver, the receiver will terminate with an error symbol $\perp$ as output, which will make it count as a violation of correctness.

Let ϵ denote the empty string and let $E = \{\epsilon\}$. If $M_{i,j} = E$, then we say that P_i sends no message to P_j in that round, i.e., we use the empty string to denote the lack of a message. Notice that if $M_{i,j} \neq E$, then $\epsilon \notin M_{i,j}$ as $M_{i,j}$ must be prefix free. Therefore, at the point where P_j specifies the code $M_{i,j}$ for a given round, P_j already knows whether or not P_i will send a message in that round. We in particular say that P_j anticipates a message from P_i when $M_{i,j} \neq E$. We will only be interested in counting the number of messages sent, not their size.

[1]This is a standard way to model secure communication by an ideal functionality since any implementation using crypto would leak the message length.

When the protocol is correct, the number of messages sent is obviously equal to the number of messages anticipated.

Definition 7.1 (Anticipated message complexity)**.** We say that the expected message complexity of a party is the expected number of times a non-empty message is sent or anticipated by the party. The expected message complexity of a protocol is simply the sum of the expected message complexity of the parties, divided by 2. We divide by 2 to avoid counting a transmitted message twice. The expectation is taken over the randomness of the players and maximised over all inputs.

The reason for insisting on a prefix free code for this slightly technical notion is to avoid a problem we would have if we allowed the communication pattern to vary arbitrarily: consider a setting where P_j wants to send a bit b to P_i. If $b = 0$ it sends no message to P_i or say the empty string. If $b = 1$ it sends 0 to P_i. If b is uniformly random, then in half the cases P_j sends a message of length 0 and in half the cases it sends a message of length 1. This means that a more liberal way of counting the communication complexity would say that the expected communication complexity is $\frac{1}{2}$. This would allow to exchange 1 bit of information with an expected $\frac{1}{2}$ bits of communication. This does not seem quite reasonable. The prefix-free model avoids this while still allowing the protocol to have a dynamic communication pattern. Note that since we want to prove impossibility it is stronger to allow protocols with dynamic rather than fixed communication patterns.

Protocols with Preprocessing. We will also consider protocols for the preprocessing model. In the preprocessing model, the specification of a protocol also includes a joint distribution $P_{R_1 \cdots R_n}$ over $\mathcal{R}_1 \times \ldots \times \mathcal{R}_n$, where the $\mathcal{R}_i$'s are finite randomness domains. This distribution is used for sampling correlated random inputs $(r_1, \ldots, r_n) \leftarrow P_{R_1 \cdots R_n}$ received by the parties before the execution of the protocol. Therefore, the preprocessing is independent of the inputs. The actions of a party P_i in a given round may in this case depend on the private random input r_i received by P_i from the distribution $P_{R_1 \cdots R_n}$ and on its input x_i and the messages received in previous rounds. In addition, the action might depend on the statistical security paramenter σ which is given as input to all parties along with x_i and r_i. Using the standard terminology of secure computation, the preprocessing model can be thought of as a hybrid model where the parties have one-time access to an ideal randomized functionality P (with no inputs) providing them with correlated, private random inputs r_i.

Security Definition. A protocol securely implements an ideal functionality with an error of ε, if the entire view of each corrupted player can be simulated with an error of at most ε in an ideal setting, where the players only have black-box access to the ideal functionality. Formally, consider Definition 7.2 below.

Definition 7.2. Let Π be a protocol for the $P_{R_1 \cdots R_n}$-preprocessing model. Let $P_{Z_1 \cdots Z_n | X_1 \cdots X_n}$ be an n-party functionality. Let Adv be a randomized algorithm, which chooses to corrupt a set $\mathcal{A} \subseteq \{1, \ldots, n\}$ of at most $t \in \mathbb{N}$ parties. Let $\mathbf{x} = (x_1, \ldots, x_n) \in \mathcal{X}_1 \times \ldots \times \mathcal{X}_n$ be an input. Let $\mathsf{Pattern}^{\Pi}(\sigma, \mathbf{x})$ denote the communication pattern in a random run of the protocol Π, i.e., the list of the length of the messages exchanged between all pairs of parties in all rounds, on input $\mathbf{x}$ and with security parameter σ. Define $\mathsf{View}^{\Pi}_{\mathsf{Adv}}(\sigma, \mathbf{x})$ to be the $\mathsf{Pattern}^{\Pi}(\sigma, \mathbf{x})$ concatenated with the view of the parties P_i for $i \in \mathcal{A}$ in the same random run of the protocol Π. Let $\mathsf{Output}^{\Pi}_{\overline{\mathcal{A}}}(\sigma, \mathbf{x})$ be just the inputs and outputs of the honest parties P_i for $i \notin \mathcal{A}$ in the same random run of the protocol Π. Let

$$\mathsf{Exec}^{\Pi}_{\mathsf{Adv}}(\sigma, \mathbf{x}) = (\mathsf{View}^{\Pi}_{\mathsf{Adv}}(\sigma, \mathbf{x}), \mathsf{Output}^{\Pi}_{\overline{\mathcal{A}}}(\sigma, \mathbf{x})) \ .$$

Let S be a randomized function called the *simulator*. Sample $\mathbf{z}$ according to $P_{Z_1 \cdots Z_n | X_1 \cdots X_n}(\mathbf{x})$. Give input $\{(x_i, z_i)\}_{i \in \mathcal{A}}$ to S. Let $\mathsf{S}(\{(x_i, z_i)\}_{i \in \mathcal{A}})$ denote the random variable describing the output of S. Let

$$\mathcal{S}_{\mathsf{S}}(\sigma, \mathbf{x}) = \Big(\mathsf{S}(\{(x_i, z_i)\}_{i \in \mathcal{A}}), \{(x_i, z_i)\}_{i \notin \mathcal{A}}\Big) \ .$$

The protocol is ε-semi-honest secure with threshold t if there exist S such that for all $\mathbf{x}$ and all $\mathcal{A}$ with $|\mathcal{A}| \leq t$ it holds that

$$\mathrm{SD}(\mathsf{Exec}^{\Pi}_{\mathsf{Adv}}(\sigma, \mathbf{x}), \mathcal{S}_{\mathsf{S}}(\sigma, \mathbf{x})) \leq \varepsilon(\sigma) \ .$$

The protocol is *statistically* semi-honest secure with threshold t if it is ε-semi-honest secure for a negligible ε.

Secret-Sharing. Unlike Definition 5.3.5, in this section we present a definition of secret sharing tailored to this section. A $(t+1)$-out-of-n secret-sharing scheme takes as input a secret s from some input domain and outputs n shares, with the property that it is possible to efficiently reconstruct s from every subset of $t+1$ shares, but every subset of at most t shares reveals nothing about the secret s. The value t is called the privacy *threshold* of the scheme.

A secret-sharing scheme consists of two algorithms: the first algorithm, called the *sharing algorithm* Share, takes as input the secret s and the parameters t and n, and outputs n shares. The second algorithm, called the *recovery algorithm* $\mathsf{Recover}$, takes as input $t+1$ shares and outputs a value s. It is required that the reconstruction of shares generated from a value s produces the same value s. Formally, consider the above definition.

Definition 7.3 (Secret-sharing)**.** Let $\mathbb{F}$ be a finite field and let $n, t \in \mathbb{N}$. A pair of algorithms $\mathscr{S}^n_t = (\mathsf{Share}, \mathsf{Recover})$ where Share is randomized and $\mathsf{Recover}$ is deterministic are said to be a secret-sharing scheme if for every $n, t \in \mathbb{N}$, the following conditions hold.

Reconstruction: For any set $\mathcal{T} \subseteq \{1, \ldots, n\}$ such that $|\mathcal{T}| > t$ and for any $s \in \mathbb{F}$ it holds that

$$\Pr[\mathsf{Recover}(\mathsf{Share}_{\mathcal{T}}(s, n, t)) = s] = 1$$

where $\mathsf{Share}_{\mathcal{T}}$ is the restriction of the outputs of Share to the elements in $\mathcal{T}$.

Privacy: For any set $\mathcal{T} \subseteq \{1, \ldots, n\}$ such that $|\mathcal{T}| \leq t$ and for any $s, s' \in \mathbb{F}$ it holds that

$$\mathsf{Share}_{\mathcal{T}}(s, n, t) \equiv \mathsf{Share}_{\mathcal{T}}(s', n, t)$$

where we use $\equiv$ to denote that two random variables have the same distribution.

Additive Secret-Sharing. In an additive secret-sharing scheme, n parties hold shares the sum of which yields the desired secret. By setting all but a single share to be a random field element, we ensure that any subset of $n-1$ parties cannot recover the initial secret.

Definition 7.4 (Additive secret-sharing)**.** Let $\mathbb{F}$ be a finite field and let $n \in \mathbb{N}$. Consider the secret-sharing scheme $\mathscr{A}^n = (\mathsf{Share}, \mathsf{Recover})$ defined below.

- The algorithm Share on input (s, n) performs the following:
  1. Generate $(s_1, \ldots, s_{n-1})$ uniformly at random from $\mathbb{F}$ and define $s_n = s - \sum_{i=1}^{n-1} s_i$.
  2. Output $(s_1, \ldots, s_n)$ where s_i is the share of the i-th party.
- The recovery algorithm $\mathsf{Recover}$ on input $(s_1, \cdots, s_n)$, outputs $\sum_{i=1}^{n} s_i$.

It is easy to show that the distribution of any $n-1$ of the shares is the uniform one on $\mathbb{F}^{n-1}$ and hence independent of s.

Secret-sharing Notation. In the sequel for a value $s \in \mathbb{F}$ we denote by $[s]^{\mathscr{S}_t^n}$ a random sharing of s for the secret-sharing scheme $\mathscr{S}_t^n$. That is, $[s]^{\mathscr{S}_t^n} \leftarrow \mathsf{Share}(s, n, t)$ where $[s]^{\mathscr{S}_t^n} = (s_1, \ldots, s_n)$. Similarly, we denote by $[s]^{\mathscr{A}^n}$ a random additive sharing of s secret shared among n parties.

Primitives. In the sequel we consider the following two-party functionalities which naturally extend to the multi-party setting.

Definition 7.5 (Multiplication MULT functionality)**.** Let $\mathbb{F}$ be a finite field. Consider two parties A and B. We define the two-party functionality $\mathrm{MULT}(a, b)$ which on input $a \in \mathbb{F}$ from party A and $b \in \mathbb{F}$ from party B outputs $\mathrm{MULT}(a, b) = a \cdot b$ to both parties.

Definition 7.6 (Inner Product IP_κ functionality)**.** Let $\mathbb{F}$ be a finite field and let $\kappa \geq 1$. Consider two parties A and B. We define the two-party functionality $\mathrm{IP}_\kappa(a, b)$ which on input $a \in \mathbb{F}^\kappa$ from party A and $b \in \mathbb{F}^\kappa$ from party B outputs $\mathrm{IP}_\kappa(a, b) = \sum_{i=1}^{\kappa} a_i b_i$ to both parties.

7.3 Secure Computation in the Plain Model

We first investigate the honest majority scenario. As explained in the introduction, we will consider protocols that compute arithmetic circuits over some field securely using secret-sharing. All known protocols of this type handle multiplication gates by running a subprotocol that takes as input shares in the two inputs a and b to the gate and output shares of the product ab, such that the output shares contain only information about ab (and no side information on a nor b). Accordingly, we define below a *multiplication gate protocol* (MGP) to be an interactive protocol for n players that does exactly this, and then show a lower bound on the communication required for such a protocol.

Definition 7.7 (MULTIPLICATION GATE PROTOCOL Π_{MULT})**.** Let $\mathbb{F}$ be a finite field and let $n \in \mathbb{N}$. Let $\mathscr{S}_t^n$ and $\hat{\mathscr{S}}_{t'}^n$ be two secret-sharing schemes as per Definition 7.3. A protocol Π_{MULT} is an n-party *Multiplication Gate Protocol* (MGP) with thresholds t, t', input sharing-scheme $\mathscr{S}_t^n$ and output sharing-scheme $\hat{\mathscr{S}}_{t'}^n$ if it satisfies the following properties:

Correctness: In the interactive protocol Π_{MULT}, players start from sets of shares $[a]^{\mathscr{S}_t^n} \leftarrow \mathsf{Share}(a, n, t)$ and $[b]^{\mathscr{S}_t^n} \leftarrow \mathsf{Share}(b, n, t)$. Each player outputs a share such that these together form a set of shares $[ab]^{\hat{\mathscr{S}}_{t'}^n}$. Moreover, $t' < 2t$.

***t*-privacy:** If the protocol is run on randomly sampled shares $[a]^{\mathscr{S}_t^n}$ and $[b]^{\mathscr{S}_t^n}$, then the only new information the output shares can reveal to the adversary is ab. We capture this by requiring that for any adversary corrupting a player subset $\mathcal{A}$ of size at most t, there exists a simulator $\mathsf{S}_{\mathcal{A}}$ which when given the input shares of the parties in $\mathcal{A}$ (denoted by $[a]_{\mathcal{A}}^{\mathscr{S}_t^n}$, $[b]_{\mathcal{A}}^{\mathscr{S}_t^n}$) and the product ab, will simulate the honest parties' output shares (denoted by $[ab]_{\overline{\mathcal{A}}}^{\hat{\mathscr{S}}_{t'}^n}$) and the view of the parties in $\mathcal{A}$ with statistically indistinguishable distribution.

Formally, for any adversary ADV corrupting a player set $\mathcal{A}$ with $|\mathcal{A}| \leq t$ there exist $\mathsf{S}_{\mathcal{A}}$ such that for randomly sampled shares $[a]^{\mathscr{S}_t^n} \leftarrow \mathsf{Share}(a, n, t)$ and $[b]^{\mathscr{S}_t^n} \leftarrow \mathsf{Share}(b, n, t)$, it holds that

$$\mathrm{SD}\left(\left(\mathsf{View}_{\mathsf{ADV}}^{\Pi_{\mathrm{MULT}}}(\sigma, [a]^{\mathscr{S}_t^n}, [b]^{\mathscr{S}_t^n})), [ab]_{\overline{\mathcal{A}}}^{\hat{\mathscr{S}}_{t'}^n}\right),\ \mathsf{S}_{\mathcal{A}}(\sigma, [a]_{\mathcal{A}}^{\mathscr{S}_t^n}, [b]_{\mathcal{A}}^{\mathscr{S}_t^n}, ab)\right) \leq \varepsilon(\sigma), \tag{7.1}$$

where σ is a security parameter and where, in the underlying random experiment, probabilities are taken over the choice of input shares as well as random coins of the protocol and simulator.

Note that we do not require the input and output sharing schemes to be the same, we only require that the output threshold is not too large ($t' < 2t$). Known MPG's actually have $t' = t$ to allow continued computation, we want to be more generous to make our lower bound stronger. Note also that we do not require the simulators to be efficient.

Recall that we use the term *gate-by-gate* protocol to refer to any protocol that computes an arithmetic circuit securely by invoking an MGP for each multiplication gate in the circuit such that the sets of shares that are input are randomly chosen. We leave unspecified what happens with addition gates as this is irrelevant for the bounds we show. An *ordered* gate-by-gate protocol invokes MGP's for multiplication gates in an order corresponding to the order in which one would visit the gates when evaluating the circuit.

In the following we show that any MGP in a gate-by-gate protocol must communicate for every multiplication gate in the honest majority setting even if only semi-honest security is required. The technique of our proof is as follows. We build an information-theoretic two-party computation protocol utilizing an n-party MGP by emulating multiple parties (in the head) and then use the impossibility result on the existence of an information-theoretic two-party computation protocol to show a contradiction.

Theorem 7.1. There exists *no* MGP Π_{MULT} as per Definition 7.7 with thresholds t, t', and with expected anticipated message complexity $\leq 2t$.

Proof. Suppose for contradiction that there exists an MGP Π_{MULT} with expected anticipated communication complexity at most $2t$. We first show a proof in the simpler case where the communication pattern is fixed. This means that at most $2t$ parties are communicating, i.e., they send or receive messages and the set of parties that communicate is known and fixed. For simplicity of exposition, suppose that these parties are $\mathsf{P}_1, \dots, \mathsf{P}_{2t}$. We are going to use Π_{MULT} to construct a two-party unconditionally secure protocol $\Pi^{A,B}_{\text{MULT}}$ which securely computes the MULT function between parties A, B as per Definition 7.5.

In particular, given two parties A and B, with inputs $a, b \in \mathbb{F}$, respectively, involved in the $\Pi^{A,B}_{\text{MULT}}$ protocol, we are going to let A emulate the first t parties that communicate and B emulate the other t parties, say $\mathsf{P}_{t+1}, \dots, \mathsf{P}_{2t}$. The protocol $\Pi^{A,B}_{\text{MULT}}$ proceeds as follows:

Protocol $\Pi^{A,B}_{\text{MULT}}(\sigma, a, b)$

Input Phase:

1. Parties A, B secret share their inputs a, b using the secret-sharing scheme $\mathscr{S}^n_t$. More specifically, A computes $[a]^{\mathscr{S}^n_t} \leftarrow \mathsf{Share}(a, n, t)$ and B computes $[b]^{\mathscr{S}^n_t} \leftarrow \mathsf{Share}(b, n, t)$.
2. Party A sends the input shares $(a_{t+1}, \dots, a_{2t})$ to party B and Party B sends the input shares $(b_1, \dots, b_t)$ to party A.

Evaluation Phase:

1. Parties A, B invoke the protocol $\Pi_{\text{MULT}}(\sigma, a_1, \dots a_n, b_1, \dots b_n)$. The emulation of Π_{MULT} yields a set of shares $[c]^{\hat{\mathscr{S}}^n_{t'}}$ and outputs $(c_1, \dots, c_t)$ to party A and $(c_{t+1}, \dots, c_{2t})$ to party B.

Output Phase:

2. Party A sends the output shares $(c_1, \dots, c_t)$ to party B and Party B sends the output shares $(c_{t+1}, \dots, c_{2t})$ to party A.
3. Each party given $2t > t'$ shares of c recovers the output $c = a \cdot b$

We now show that the above protocol is correct and secure. Correctness follows immediately from $t' < 2t$ - as then $2t$ shares are enough to reconstruct. The protocol is secure (private) due to the t-privacy property of Π_{MULT}. More precisely, if party A is corrupted, we need to simulate his view of the protocol given a and the product ab. We do this as follows: Let $\mathcal{A}$ be the set of parties A emulates in the MGP. We now compute $[a]^{\mathscr{S}^n_t} \leftarrow \mathsf{Share}(a, n, t)$ and sample $[b]^{\mathscr{S}^n_t}_{\mathcal{A}}$ which can be done by the privacy property of $\mathscr{S}^n_t$. We then run the simulator $\mathsf{S}_{\mathcal{A}}$ guaranteed by the t-privacy property to get $\mathsf{S}_{\mathcal{A}}(\sigma, [a]^{\mathscr{S}^n_t}_{\mathcal{A}}, [b]^{\mathscr{S}^n_t}_{\mathcal{A}}, ab)$. Note that this output includes $\mathcal{A}$'s view of the MGP as well as all output shares.

The simulator now outputs $[a]^{\mathscr{S}^n_t}$, $[b]^{\mathscr{S}^n_t}_{\mathcal{A}}$ and $\mathsf{S}_{\mathcal{A}}(\sigma, [a]^{\mathscr{S}^n_t}_{\mathcal{A}}, [b]^{\mathscr{S}^n_t}_{\mathcal{A}}, ab)$. This is statistically indistinguishable from A's view of $\Pi^{A,B}_{\text{MULT}}(\sigma, a, b)$ by the privacy property of $\mathscr{S}^n_t$ and equation (7.1). A similar simulator for B's view is easy to construct.

However, the above leads to a contradiction since it is well known [BOGW88, CCD88] that it is impossible to realize passively secure two-party multiplication (such as the $\Pi^{A,B}_{\text{MULT}}$ protocol) in the information theoretic setting (even if inefficient simulators are allowed). Therefore, the theorem follows.

We now address the case where the communication pattern might be dynamic. We say that a party communicated if it sent a non-empty message or if it anticipated a non-empty message. So by definition, the expected number of communicating parties is $\leq 2t$. Since the observed value is an integer, there is some non-zero, constant probability p such that the observed value of the number of communication parties is at most $2t$. We can therefore pick a subset $\mathcal{C}$ of the parties of size $2t$ such that it happens with probability at least $p/\binom{n}{2t}$ that only the parties in $\mathcal{C}$ communicate. Since we can increase the security parameter σ independently of n, the number $p/\binom{n}{2t}$ is a positive constant (in σ). We can then modify $\Pi^{A,B}_{\mathrm{MULT}}(a,b)$ such that B runs t parties in $\mathcal{C}$ and A runs the other t parties. The protocol runs as $\Pi^{A,B}_{\mathrm{MULT}}(a,b)$ except that if it A or B observe that a party in $\mathcal{C}$ anticipates a non-empty message from a party outside $\mathcal{C}$, then the execution is terminated. In case the protocol terminates, the two parties just try again. Since $p/\binom{n}{2t}$ is a positive constant this succeeds in an expected constant number of tries. Notice that when the protocol succeeds, all parties in $\mathcal{C}$ received all the messages they would have received in a run of $\Pi^{A,B}_{\mathrm{MULT}}(a,b)$ where all the parties were active, as parties only receive the messages they anticipate. Hence the parties in $\mathcal{C}$ have correct outputs (except with negligible probability). For the same reason the output of the parties simulated by A and B will be correct. Hence A and B can reconstruct the output from the $2t$ shares. We can also argue that the protocol is private: We will simulate A's (or B)'s view by running the simulator $\mathsf{S}_{\mathcal{A}}$ (where again $\mathcal{A}$ is the set of parties emulated by A) repeatedly until a view is produced where no party in $\mathcal{C}$ anticipates a message from outside of $\mathcal{C}$. Note that $\mathsf{S}_{\mathcal{A}}$ simulates the view of an adversary corrupting $\mathcal{A}$, and this view includes the communication pattern from which it is evident who anticipates messages. ■

The above theorem immediately implies the following.

Corollary 7.2. Any gate-by-gate protocol that is secure against $t = \Theta(n)$ corruptions must communicate $\Omega(n \cdot |C|)$ bits where $|C|$ is the size of the circuit C to compute, and moreover, an ordered gate-by-gate protocol must have a number of rounds that is proportional to the (multiplicative) depth of C.

Jumping ahead, we note that the arguments for this conclusion break down completely when we consider secure computation in the preprocessing model with dishonest majority since in such a model it is no longer true that two-party unconditionally secure multiplication is impossible: just a single preprocessed multiplication triple will be enough to compute a multiplication. We return to this issue in the next section.

A bound that grows with the field size? It is natural to ask if we can get a lower bound on the complexity of an MPG that grows with the field size? after all, existing MGPs do need to send more bits for larger fields. However, the answer is no, as the following example shows: for $a \in \mathbb{F}$, define z_a to be 0 if $a = 0$ and 1 otherwise. Then we represent an element $a \in \mathbb{F}$ as a pair (z_a, ℓ_a) where ℓ_a is randomly chosen if $a = 0$ and otherwise $\ell_a = \log_g(a)$, where g

is a fixed generator of the multiplicative group $\mathbb{F}^*$. Let $u = |\mathbb{F}^*|$. Observe that now we have $(z_{ab}, \ell_{ab}) = (z_a \cdot z_b, (\ell_a + \ell_b) \bmod u)$.

We now construct a secret sharing scheme: given a secret $a \in \mathbb{F}$, we first compute (z_a, ℓ_a) and then share z_a using, e.g., Shamir's scheme and share ℓ_a additively modulo u. An MGP for this scheme can use a standard protocol to compute shares in $z_a \cdot z_b$ and local addition to get shares in $(\ell_a + \ell_b) \bmod u$. Clearly, the communication complexity of this MGP does not depend on $|\mathbb{F}|$.

Of course, the secret sharing scheme we defined is not efficient (at least not in all fields) because one needs to take discrete logs. This is not formally a problem since we did not make any assumptions on the efficiency of secret sharing schemes. But we can in fact get a more satisfactory solution by replacing the additive sharing of the discrete log with black-box sharing directly over the group $\mathbb{F}^*$ [CF02]. This is doable in polynomial time, will cost a factor that is logarithmic in the number of players, but since black-box secret-sharing is homomorphic over the group operation, the resulting MGP still has communication independent of $|\mathbb{F}|$.

Amortized Multiplication Gate Protocols. There is one clear possibility for circumventing the bounds we just argued for gate-by-gate protocols, namely: what if the circuit structure allows us to do, say k multiplications in parallel? Perhaps this can be done more efficiently than k separate multiplications? Of course, this will not help for a worst case circuit whose depth is comparable to its size. But in fact, for "nicer" circuits, we know that such optimizations are possible, based on so-called packed secret-sharing. The catch, however, is that apart from loosing in resilience this only works if there is a gap of size $\Theta(k)$ between the privacy and reconstruction thresholds of the secret-sharing scheme used, so the number of players must grow with k.

One may ask if this is inherent, i.e., can we save on the *communication* needed for many multiplication gates in parallel, only by increasing the number of players? While we believe this is true, we were not able to show it in full generality. But we were able to do so for *computational* complexity, as detailed below. Furthermore, for a restricted setting we explain below and a fixed number of players, we could show that the communication must grow linearly with k.

First, we can trivially extend Definition 7.3 to cover schemes in which the secret is a vector $\mathbf{a} = (a_1, \ldots, a_k)$ of field elements instead of a single value. A further extension covers *ramp* schemes in which there are two thresholds: the privacy threshold t which is defined as in Definition 7.3 and a reconstruction threshold $r > t$, where any set of size at least r can reconstruct the secret. Such a scheme is denoted by $\mathscr{S}^n_{t,r}$. Note that the shares in this case may be shorter than the secret, perhaps even a single field element per player. We can now define a simple extension of the multiplication gate protocol concept:

Definition 7.8 (*k-Multiplication Gate Protocol* Π_{MULT^k})**.** Let $\mathbb{F}$ be a finite field and let $n \in \mathbb{N}$. Let $\mathscr{S}^n_{t,r}$ and $\hat{\mathscr{S}}^n_{t,r}$ be two ramp sharing schemes defined over $\mathbb{F}$, for sharing vectors in $\mathbb{F}^k$. Π_{MULT^k} is said to be a *k-Multiplication Gate Protocol* (k-MGP) with thresholds t, r, input sharing scheme $\mathscr{S}^n_{t,r}$ and output sharing scheme $\hat{\mathscr{S}}^n_{t,r}$ if it satisfies the following properties:

Correctness: In the interactive protocol Π_{MULT^k}, players start from sets of shares $[\mathbf{a}]^{\mathscr{S}^n_{t,r}}$ and $[\mathbf{b}]^{\mathscr{S}^n_{t,r}}$. Each player outputs a share such that these together form a set of shares $[\mathbf{a} * \mathbf{b}]^{\hat{\mathscr{S}}^n_{t,r}}$, where $\mathbf{a} * \mathbf{b}$ is the coordinatewise product of $\mathbf{a}$ and $\mathbf{b}$.

t-privacy: If the protocol is run on randomly sampled shares $[\mathbf{a}]^{\mathscr{S}_t^n}$ and $[\mathbf{b}]^{\mathscr{S}_t^n}$, then the only new information the output shares can reveal to the adversary is $\mathbf{a} * \mathbf{b}$. We capture this by requiring that for any adversary corrupting player subset $\mathcal{A}$ of size at most t, there exists a simulator $\mathsf{S}_{\mathcal{A}}$ which when given the input shares of the parties in $\mathcal{A}$ (denoted by $[\mathbf{a}]_{\mathcal{A}}^{\mathscr{S}_t^n}$, $[\mathbf{b}]_{\mathcal{A}}^{\mathscr{S}_t^n}$) and the product ab, will simulate the honest parties' output shares (denoted by $[\mathbf{a} * \mathbf{b}]_{\overline{\mathcal{A}}}^{\hat{\mathscr{S}}_{t'}^n}$) and the view of the parties in $\mathcal{A}$ with statistically indistinguishable distribution. Formally, for any adversary ADV corrupting player set $\mathcal{A}$ with $|\mathcal{A}| \leq t$ there exist $\mathsf{S}_{\mathcal{A}}$ such that for randomly sampled shares $[\mathbf{a}]^{\mathscr{S}_t^n} \leftarrow \mathsf{Share}(\mathbf{a}, n, t)$ and $[\mathbf{b}]^{\mathscr{S}_t^n} \leftarrow \mathsf{Share}(\mathbf{b}, n, t)$, it holds that

$$\mathrm{SD}\left(\left(\mathsf{View}_{\mathsf{ADV}}^{\Pi^k_{\mathrm{MULT}}}(\sigma, [\mathbf{a}]^{\mathscr{S}_t^n}, [\mathbf{b}]^{\mathscr{S}_t^n})), [\mathbf{a} * \mathbf{b}]_{\overline{\mathcal{A}}}^{\hat{\mathscr{S}}_{t'}^n}\right),\ \mathsf{S}_{\mathcal{A}}(\sigma, [\mathbf{a}]_{\mathcal{A}}^{\mathscr{S}_t^n}, [\mathbf{b}]_{\mathcal{A}}^{\mathscr{S}_t^n}, \mathbf{a} * \mathbf{b})\right) \leq \varepsilon(\sigma), \quad (7.2)$$

where σ is a security parameter and where, in the underlying random experiment, probabilities are taken over the choice of input shares as well as random coins of the protocol and simulator.

Before giving our result on k-MGPs we note that for any interactive protocol, it is always possible to represent the total computation done by the players as an arithmetic circuit over a finite field (arithmetic circuits can emulate Boolean circuit which can in turn emulate Turing machines). We can encode messages as field elements and represent sending of messages by wires between the parts of the circuit representing sender and receiver. For a protocol Π, we refer to an algorithm outputting such a circuit as *an arithmetic representation of* Π. Note that such a representation is not in general unique, but once we have chosen one, it makes sense to talk about, e.g., the number of multiplications done by a player in Π.

Theorem 7.3. Let $t < r \leq n \in \mathbb{N}$. Also let $\mathcal{P} = \{\mathsf{P}_1, \ldots, \mathsf{P}_n\}$ be a set of parties. Assume that the k-MGP Π_{MULT^k} defined over $\mathbb{F}$ has thresholds t, r. Then for any arithmetic representation of Π_{MULT^k} (over any finite field) and for each subset $\mathcal{S} \subset \mathcal{P}$ of size $n - 2t$, the total number of multiplications done by players in $\mathcal{S}$ is $\Omega(k)$

Proof. Suppose for contradiction that there exists a k-MGP Π_{MULT^k} in which the total number of multiplications done by players in $\mathcal{S}$ is $o(k)$. Assume for notational convenience that $\mathcal{S} = \{\mathsf{P}_{2t+1}, \ldots, \mathsf{P}_n\}$. We are going to use it to construct a two-party unconditionally secure protocol $\Pi_{\mathrm{MULT}}^{A,B}$ in the preprocessing model which securely computes k multiplications as follows. We let $u \leftarrow P_U$ denote the correlated randomness we will use in $\Pi_{\mathrm{MULT}}^{A,B}$. Given two parties A and B involved in the $\Pi_{\mathrm{MULT}}^{A,B}$ protocol, the idea is to use the assumed k-MGP where A emulates t players and B emulates another t players. In addition, parties A, B together emulate the rest of the parties in $\mathcal{S}$. This can be done using the preprocessed data u: we consider the parties in $\mathcal{S}$ as a reactive functionality $f_{\mathcal{S}}$ which can be implemented using an existing protocol in the preprocessing model. One example of such a protocol is the SPDZ protocol [DPSZ12] denoted by $\Pi_{f_{\mathcal{S}}}^{SPDZ}$ [2] which uses additive-secret sharing. Therefore, protocol $\Pi_{\mathrm{MULT}}^{A,B}$ proceeds as follows:

[2]We do passive security here, so a simpler variant of SPDZ will suffice, without authentication codes on the shared values.

Protocol $\Pi^{A,B}_{\text{MULT}}(\{a_i\}_{i\in[k]}, \{b_i\}_{i\in[k]}, u)$:

Input Phase:

1. $\forall i \in [k]$, parties A, B secret share their inputs a_i, b_i using the ramp sharing scheme $\mathscr{S}^n_{t,r}$. So A computes $[\mathbf{a}]^{\mathscr{S}^n_{t,r}} \leftarrow \mathsf{Share}((\mathbf{a}), n, t)$ and B computes $[\mathbf{b}]^{\mathscr{S}^n_{t,r}} \leftarrow \mathsf{Share}((\mathbf{b}), n, t)$. For simplicity of exposition, we denote by $(\bar{a}_1, \ldots, \bar{a}_n), (\bar{b}_1, \ldots, \bar{b}_n)$ the shares of $[\mathbf{a}]^{\mathscr{S}^n_{t,r}}$ and $[\mathbf{b}]^{\mathscr{S}^n_{t,r}}$, respectively.
2. Party A sends the input shares $(\bar{a}_1, \ldots, \bar{a}_t)$ to party B and Party B sends the input shares $(\bar{b}_t, \ldots, \bar{b}_{2t})$ to party A.
3. Additively secret share the inputs $(\bar{a}_{2t+1}, \ldots, \bar{a}_n, \bar{b}_{2t+1}, \ldots, \bar{b}_n)$ of the parties in $\mathcal{S}$ between A and B using the additive secret-sharing $\mathscr{A}^2$ and obtain the shares $([\bar{a}_{2t+1}]^{\mathscr{A}^2}, \ldots, [\bar{a}_n]^{\mathscr{A}^2}, [\bar{b}_{2t+1}]^{\mathscr{A}^2}, \ldots, [\bar{b}_n]^{\mathscr{A}^2})$. For the following phase, as we mentioned above, we will think of the computation done by the parties in $\mathcal{S}$ as a reactive functionality $f_{\mathcal{S}}$ which is implemented using the protocol $\Pi^{SPDZ}_{f_{\mathcal{S}}}$ in the preprocessing model.

Evaluation Phase:

Parties A, B invoke the protocol $\Pi_{\text{MULT}^k}([\mathbf{a}]^{\mathscr{S}^n_{t,r}}, [\mathbf{b}]^{\mathscr{S}^n_{t,r}})$ in which A, B emulate t parties each, and they together emulate the rest, $n-2t$ players, using the preprocessed data u invoking protocol $\Pi^{SPDZ}_{f_{\mathcal{S}}}$. To this end, note that $\Pi^{SPDZ}_{f_{\mathcal{S}}}$ represents data by additive secret-sharing. Values $(\bar{a}_{2t+1}, \ldots, \bar{a}_n, \bar{b}_{2t+1}, \ldots, \bar{b}_n)$ of the parties in $\mathcal{S}$ were already additively shared, so they can be used directly as input to $\Pi^{SPDZ}_{f_{\mathcal{S}}}$.

Now, the emulation of Π_{MULT^k} is augmented with the protocol $\Pi^{SPDZ}_{f_{\mathcal{S}}}$ as follows: when a party in $\mathcal{S}$ would do a local operation, we do the same operation in $\Pi^{SPDZ}_{f_{\mathcal{S}}}$. When a party outside $\mathcal{S}$ sends a message to a party in $\mathcal{S}$ an additive secret-sharing of that message is formed between A and B. When a party in $\mathcal{S}$ sends a message to a party outside $\mathcal{S}$ the corresponding additive secret-sharing is reconstructed towards A or B, depending on who emulates the receiver. In the end, we will obtain additive sharings between A and B of the outputs of parties in $\mathcal{S}$, namely $([\bar{c}_{2t+1}]^{\mathscr{A}^2}, \ldots, [\bar{c}_n]^{\mathscr{A}^2})$.

Output Phase:

1. A sends the output shares $(\bar{c}_1, \ldots, \bar{c}_t)$ to B, B sends the output shares $(\bar{c}_{t+1}, \ldots, \bar{c}_{2t})$ to A computed by Π_{MULT^k}, and A and B exchange their additive shares $([\bar{c}_{2t+1}]^{\mathscr{A}^2}, \ldots, [\bar{c}_n]^{\mathscr{A}^2})$ in order to recover $(\bar{c}_{2t+1}, \ldots, \bar{c}_n)$.
2. Now both A and B have $n \geq r$ shares of the output and can recover the result $\mathbf{a} * \mathbf{b}$.

We now show that the above protocol is correct and secure. Correctness follows immediately from the correctness of Π_{MULT^k} and $\Pi^{SPDZ}_{f_{\mathcal{S}}}$. We argue that the protocol is secure (private) due to the security of $\Pi^{SPDZ}_{f_{\mathcal{S}}}$ and the t-privacy property of the MGP Π_{MULT^k} (see equation (7.1)). For the case where A is corrupted, we first observe that by using the simulator for the $\Pi^{SPDZ}_{f_{\mathcal{S}}}$

protocol, we can argue that the view of A in the real protocol is statistically close to the one obtained by replacing players in $\mathcal{S}$ by the ideal functionality $f_{\mathcal{S}}$.

We can then make a simulator for corrupt A in the $f_{\mathcal{S}}$-hybrid model, as follows: The shares received by A in the input phase can be simulated by the privacy property of the input sharing scheme, and the rest of the view can be simulated by invoking the simulator $\mathsf{S}_{\mathcal{A}}$ of the protocol Π_{MULT^k} guaranteed by Definition 7.7, on input $[\mathbf{a}]_{\mathcal{A}}^{\mathscr{S}_t^n}, [\mathbf{b}]_{\mathcal{A}}^{\mathscr{S}_t^n}, \mathbf{a} * \mathbf{b}$. Note that $\mathsf{S}_{\mathcal{A}}$ is in charge of simulating $f_{\mathcal{S}}$. It can therefore define the responses of $f_{\mathcal{S}}$ such that they are consistent with the view generated by $\mathsf{S}_{\mathcal{A}}$.[3]

We therefore conclude from equation (7.2) that S_A generates a view that is statistically indistinguishable from the real view of an adversary corrupting A. A similar argument holds for B.

Now note that the preprocessed data required by the protocol $\Pi_{f_P}^{SPDZ}$ amount to a constant number of field elements for each multiplication done. This means that our 2-party protocol needs $o(k)$ preprocessed field elements by assumption on Π_{MULT^k}. However, this leads to a contradiction since by results in [WW10], it is impossible for two parties to compute k multiplications with statistical security using preprocessed data of size $o(k)$ field elements. ■

What this theorem shows is, for instance, that if we want each player to do only a constant number of local multiplications in a k-MGP, then n needs to be $\Omega(k)$. Since this is precisely what protocols based on packed sharing can achieve (see, e.g., [DIK$^+$08]), the bound in the theorem is in this sense tight. What the theorem also says is that *every* subset of size $n - 2t$ needs to work hard, so in the case where we tolerate a maximal number of corruptions, i.e., $n = 2t + 1$, we see that a gate by gate protocol in this case must have computational complexity $\Omega(n|C|)$, for *any* circuit of size $|C|$, not only for "tall and skinny" circuits as we had before.

7.4 Secure Computation in the Preprocessing Model

It is well known that all functions can be computed with unconditional security in the setting where $n - 1$ of the n players may be corrupted, and where the players are given correlated randomness, also known as preprocessed data, that does not have to depend on the function to be computed, nor on the inputs. Winkler and Wullschleger [WW10] proved lower bounds on the the amount of preprocessed data needed to compute certain functions with statistical security where the bound depends on certain combinatorial properties of the target function.

All existing protocols in the preprocessing model that are efficient in the circuit size of the function, work according to the gate-by-gate approach we encountered in the previous section. We can define (ordered) gate-by-gate protocols and MGPs exactly as for the honest majority setting, with two exceptions: MGPs are allowed to consume preprocessed data, and the output threshold t' must equal the input threshold t. This is because we typically have $t = n - 1$ in this setting, and then it does not make sense to consider $t' > t$, then even all players cannot reconstruct the output,

[3] Note that $\Pi_{f_{\mathcal{S}}}^{SPDZ}$ reveals the structure of the circuit for $f_{\mathcal{S}}$. This is secure as we assume that the parties in $\mathcal{S}$ are represented as known arithmetic circuits.

As before, we want to show that multiplication gate protocols require a certain amount of communication, but as mentioned before, we can no longer base ourselves on impossibility of unconditionally secure multiplication for two parties, since this is in fact possible in the preprocessing model. Instead, the contradiction will come from the known lower bounds on the size of the preprocessed data needed to compute certain functions.

7.4.1 Protocols based on Additive Secret-Sharing

We start by showing that any gate-by-gate protocol must communicate for every multiplication gate when the underlying secret sharing scheme is the additive one. We show that an MGP that does not communicate enough implied a protocol that contradicts the lower bound by Winkler and Wullschleger [WW10] on the the amount of preprocessed data needed to compute certain functions with statistical security.

Theorem 7.4. Consider the preprocessing model where $n-1$ of the n players may be passively corrupted. In this setting, there exists *no* MGP Π_{MULT} with expected anticipated communication complexity $\leq n-1$ and with additive secret-sharing $\mathscr{A}^n$ as output sharing scheme.

Proof. Suppose for contradiction that there exists an MGP Π_{MULT} (with preprocessed data $u \leftarrow P_U$) which contradicts the claim of the theorem. Similar to Theorem 7.1 we will first assume a fixed communication pattern. Assume for notational convenience that only the parties $\mathsf{P}_1, \ldots, \mathsf{P}_{n-1}$ communicate. Given two parties A and B, we are going to construct a two-party protocol $\Pi_{\mathrm{MULT}}^{A,B}$ which on input $a, b \in \mathbb{F}$ from A, B, respectively, securely computes ab. The idea is for A to emulate the $n-1$ players who communicate in Π_{MULT} while B emulates the last player. In particular, protocol $\Pi_{\mathrm{MULT}}^{A,B}$ proceeds as follows:

Protocol $\Pi_{\mathrm{MULT}}^{A,B}$

Input Phase:

1. Parties A, B secret share their inputs a, b using the input secret-sharing scheme $\mathscr{A}^n$ of Π_{MULT}. More specifically, A computes $[a]^{\mathscr{A}^n} \leftarrow \mathsf{Share}(a, n, n-1)$ and B computes $[b]^{\mathscr{A}^n} \leftarrow \mathsf{Share}(b, n, n-1)$.
2. Party A sends the input share a_n to party B and Party B sends the input shares $(b_1, .., b_{n-1})$ to party A.

Evaluation Phase:

1. Parties A, B invoke the MGP Π_{MULT} as per Definition 7.7 in the preprocessing model where A emulates the $n-1$ players who communicate, and we assume these are the first $n-1$ players. This means that this phase involves no communication between A and B, but it may consume some preprocessed data u. The execution of Π_{MULT} yields a sharing of $[c]^{\mathscr{A}^n}$ and outputs $(c_1, ..., c_{n-1})$ to party A and c_n to party B.

Output Phase:

1. A sends $\sum_{i=1}^{n-1} c_i$ to B and B sends c_n to A. The parties add the received values to recover the output $c = a \cdot b$.

Correctness of this protocol follows immediately. The protocol can be argued to be secure(private). In particular, the simulator S for $\Pi^{A,B}_{\mathrm{MULT}}$ proceeds as follows. The preprocessing data to be used by the corrupted party can be simulated with the correct distribution without any knowledge of the inputs. In the input phase, the corrupted party receives only an unqualified set of shares whose distribution can be simulated perfectly. There is no communication to be simulated in the evaluation phase. In the output phase, it is the case that whenever the protocol computes the correct result, then the share received from the honest party is trivial to simulate because it is determined from the corrupted party's own share and the result ab. Hence, the only source of error is the negligible probability that the output is wrong in the real execution, so it follows that

$$\mathsf{SD}(\mathsf{Exec}^{\Pi^{A,B}_{\mathrm{MULT}}}_{\mathsf{Adv}}(\sigma,(a,b)),\mathcal{S}_{\mathsf{S}}(\sigma,(a,b))) \leq \epsilon(\sigma).$$

However, we can say even more: Let $u \leftarrow P_U$ be the preprocessed data that is consumed during the protocol (Π_{MULT} uses preprocessed data). We now define a new protocol $\Pi^{A,B}_{\mathrm{MULT}^k}$ that will compute k *independent* multiplications (do not confuse this protocol with the amortized and honest majority protocol in Definition 7.8). It does this by running k instances of $\Pi^{A,B}_{\mathrm{MULT}}$, *using the same preprocessed data* u for all instances.

Normally, it is of course not secure to reuse preprocessed data, but in this particular case it works because the communication in $\Pi^{A,B}_{\mathrm{MULT}}$ is independent of u, and so is the simulation. More precisely, $\Pi^{A,B}_{\mathrm{MULT}^k}$ is clearly correct because each instance of $\Pi^{A,B}_{\mathrm{MULT}}$ runs with correctly distributed preprocessed data. It is also private: we can simulate by first simulating the corrupted party's part of u and then running k instances of the rest of S's code. Again, the only source of error is the case where the real protocol computes an incorrect result, but the probability of this happening for any of the k instances is at most a factor k larger than for a single instance, by a union bound, and so is still negligible.

However, this leads to a contradiction with the result of [WW10]: they showed that the amount of preprocessed data needed for a secure multiplication is at least some non-zero number of bits w. It also follows from [WW10] that if we want k multiplications on independently chosen inputs this requires kw bits. So if we consider a k large enough that kw is larger than the size of u, we have a contradiction and the theorem follows.

We now generalise to dynamic communication patterns. As in the proof of Theorem 7.1 we can find a party P_i such that with some constant positive probability p the party P_i does not send a message and no party anticipates a message from P_i. Assume without loss of generality that this is party P_n. Assume first that p is negligibly close to 1. In that case the parties can apply the above protocol unmodified. Consider then the case where p is not negligibly close to 1. We also have that p is not negligibly close to 0. Hence there is a non-negligible probability that P_n sends a message and a non-negligible probability that P_n does not send a message. The decision of P_n to communicate or not can depend only on four values:

- Its share a_n of a.
- Its share b_n of b.
- Its share u_n of the correlated randomness.

- Its private randomness, call it r_n.

This means that there exist a function $\varrho(a_n, b_n, u_n, r_n) \in \{0,1\}$ such that P_n communicates iff $\varrho(a_n, b_n, u_n, r_n) = 1$. Observe that the decision can in fact not depend more than negligibly on a_n and b_n. If it did, this would leak information on these shares to the parties $\mathsf{P}_1, \ldots, \mathsf{P}_{n-1}$ which already know all the other shares. This would in turn leak information on a or b to the parties $\mathsf{P}_1, \ldots, \mathsf{P}_{n-1}$, which would contradict the simulatability property of the protocol. We can therefore without loss of generality assume that there exist a function $\varrho(u_n, r_n) \in \{0,1\}$ such that P_n communicates iff $\varrho(u_n, r_n) = 1$.

Assume that with non-negligible probability over the choice of the u_n received by P_n it happens that the function $\varrho(u_n, r_n)$ depends non-negligibly on r_n, i.e., for a uniform r_n it happens with non-negligible probability that $\varrho(u_n, r_n) = 0$ and it also happens with non-negligible probability that $\varrho(u_n, r_n) = 1$. Since r_n is independent of the view of the parties $\mathsf{P}_1, \ldots, \mathsf{P}_{n-1}$, as it is the private randomness of P_n, it follows that the probability that one of the other parties anticipate a message from P_n is independent of whether $\varrho(u_n, r_n) = 0$ or $\varrho(u_n, r_n) = 1$. Hence it either happens with non-negligible probability that $\varrho(u_n, r_n) = 0$ and yet one of the other parties anticipate a message from P_n or it happens with non-negligible probability that $\varrho(u_n, r_n) = 1$ and yet none of the other parties anticipate a message from P_n. Both events contradict the correctness of the protocol. We can therefore without loss of generality assume that there exist a function $\varrho(u_n) \in \{0,1\}$ such that P_n communicates iff $\varrho(u_n) = 1$. By assumption we have that p is non-zero, so there exist some u_n such that $\varrho(u_n) = 0$. We can therefore condition the execution on the event $\varrho(u_n) = 0$. Let P_U be the distribution from which u is sampled. Consider then the random variable $P_{U'}$ which is distributed as P_U under the condition that $\varrho(u_n) = 0$. We claim that if we run Π_{MULT} with $P_{U'}$ instead of P_U then the protocol is still secure. Assuming that this claim is true, A and B can apply the above protocol, but simply use $(\Pi_{\mathrm{MULT}}, P_{U'})$ instead of $(\Pi_{\mathrm{MULT}}, P_U)$.

What remains is therefore only to argue that $(\Pi_{\mathrm{MULT}}, P_{U'})$ is secure. To simulate the protocol, run the simulator $\mathsf{S}'_{\mathcal{A}}$ for $(\Pi_{\mathrm{MULT}}, P_U)$ until it outputs a simulated execution where P_n did not communicate. Let E be the event that P_n does not communicate. Since it can be checked from just inspecting the view of the real execution of $(\Pi_{\mathrm{MULT}}, P_U)$ (or the simulation) whether E occurred, it follows that E occurs with the same probability in the real execution and the simulation (or at least probabilities which are negligible close) or we could use the occurrence of E to distinguish. Since E happens with a positive constant probability it then also follows that the real execution conditioned on E and the simulation condition on E are indistinguishable, or we could apply a distinguisher for the conditioned distributions when E occurs and otherwise make a random guess to distinguish the real execution of $(\Pi_{\mathrm{MULT}}, P_U)$ from its simulation. This shows that $\mathsf{S}'_{\mathcal{A}}$ simulates $(\Pi_{\mathrm{MULT}}, P_{U'})$. ■

A generalisation. We note that Theorem 3 easily extends to any output secret sharing scheme with the following property: Given shares $c_1, ..., c_n$ of c, there is a function ϕ such that one can reconstruct c from $c_1, ..., c_{n-1}, \phi(c_n)$ and given c and $c_1, ..., c_{n-1}$ one can simulate $\phi(c_n)$ with statistically close distribution. The proof is the same as above except that in the output phase, B sends $\phi(c_n)$ to A, who computes c and sends it to B.

Theorem 3 shows, for instance, that the SPDZ protocol [DPSZ12] has optimal communication for the class of gate-by-gate protocols using additive secret-sharing: it sends $O(n)$ messages for each multiplication gate, and of course one needs to send $\Omega(n)$ messages if all n players are to communicate, as mandated in the theorem. Note also that in the dishonest majority setting, the privacy threshold of the secret-sharing scheme used has to be $n-1$, so we cannot have a gap between the reconstruction and privacy thresholds, and so amortisation tricks based on packed secret-sharing cannot be applied. We therefore do not consider any lower bounds for amortised MGP's.

7.4.2 Protocols based on any Secret-Sharing Scheme

Note that if we consider an MGP whose output sharing scheme is not the additive scheme, the protocol $\Pi^{A,B}_{\text{MULT}}$ in the proof of Theorem 7.4 may not work. This is because it is no longer clear that given your own share of the product and the result, the other party's share is determined. In particular, the distribution of the other share may depend on the preprocessed data we consume and so if we just send that share in the clear, it is not obvious that we can reuse the preprocessing.

The solution is to not send shares in the clear, but have the parties securely compute the output from their shares. This can be done using an existing general protocol for secure computation in the preprocessing model. This will mean that we can indeed reuse preprocessed data consumed by the MGP protocol itself. However, we now consume new preprocessed data for every instance of the reconstruction protocol since this protocol requires communication. It turns out that if we use a variant of the MGP that computes, not just one product, but an inner product of long enough vectors, we can still obtain a contradiction. This works because we can show that computing the inner product of long vectors requires lots of preprocessed data. On the other hand, the inner product itself is just one field element, therefore the cost of reconstructing such a small result is not significant.

In order to obtain the above result and give more details, we proceed by proving some auxiliary results with lower bounds on the amount of preprocessed data needed for a secure evaluation of a function f.

7.4.2.1 Lower bounds for secure function evaluation in the preprocessing model.

In this section we will give lower bounds for secure implementations of functions $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ in the P_U, P_V-preprocessing model, which for simplicity of exposition we refer to as P_{U_f,V_f}, that outputs correlated randomness for the semi-honest setting. In particular, we are in the setting where the parties A, B have access to a functionality that gives a random variable U_f to A and V_f to B with some guaranteed joint distribution P_{U_f,V_f} of U_f, V_f. Given this, the parties compute securely a function $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ where A holds $x \in \mathcal{X}$, and B holds $y \in \mathcal{Y}$. This function should have no redundant inputs for party A [4] :

$$\forall x, x' \in \mathcal{X}(x \neq x' \to \exists y \in \mathcal{Y} : f(x,y) \neq f(x',y)) \tag{7.3}$$

[4] Party A must enter all the information about X into the protocol. An example of a function that satisfies this property is the inner product IP.

The authors of [WW10] obtained Theorem 7.5 that gives a lower bound on the conditional entropy of P_{U_f,V_f}. Their bound applies for input distributions X and Y which are independent and uniformly distributed. This implies worst case communication complexity. Our bound in Theorem 7.6 also applies to independent and uniform distributions.

Theorem 7.5. Let $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ be a function that satisfies property (7.3). Assume there exists a protocol having access to P_{U_f,V_f} which is an ε-secure implementation of f in the semi-honest model with $t = 1$ corruptions. Then

$$H(U_f|V_f) \geq \max_y H(X|f(X,y)) - (3|\mathcal{Y}| - 2)(\varepsilon \log |\mathcal{Z}| + h(\varepsilon)) - \varepsilon \log |\mathcal{X}| - h(\varepsilon).$$

Our general result will only apply to functions where the output lives in a ring $\mathcal{Z}$. As it will become apparent, for the next theorem we require the following property for a function $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$:

$$\forall x, x' \in \mathcal{X}(x \neq x' \to \exists y_1, y_2 \in \mathcal{Y} : f(x, y_1) - f(x, y_2) \neq f(x', y_1) - f(x', y_2)) \tag{7.4}$$

Note that the bound in Theorem 7.5 still applies for functions f that satisfy properties (7.3) and (7.4).

In the following we explore the lower bounds on the amount of preprocessed data with respect to composition of functions. In Theorem 7.6 we prove a lower bound on the conditional entropy of P_{U_h,V_h} for a function h which is a linear combination of two functions f and g. Our bound also applies to compositions of k functions where k is an arbitrary number. Basically, we show that the amount of preprocessed data you need to compute the sum of f and g is the sum of what you need to compute f and g separately, as long as f and g are applied to distinct and independent inputs. We clearly need this assumption, as otherwise the theorem is clearly false, just think of applying $f = g$ on the same inputs.

Theorem 7.6. Let $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}_f$, $g : \mathcal{Z} \times \mathcal{W} \to \mathcal{Z}_g$ be functions that satisfy properties (7.3) and (7.4). Assume that $\mathcal{Z}_f = \mathcal{Z}_g$. Let h be a linear combination of f and g, namely: $\forall x \in \mathcal{X}, y \in \mathcal{Y}, z \in \mathcal{Z}, w \in \mathcal{W}$, $h(x, z, y, w) := \alpha f(x, y) + \beta g(z, w)$ for some $\alpha, \beta \neq 0$. If there exists a protocol that securely implements the function h with access to P_{U_h,V_h}, then it holds that

$$H(U_h|V_h) \geq \max_y H(X|f(X,y)) + \max_w H(Z|g(Z,w)) \ .$$

Furthermore, the function h will have the following property:

$$\begin{aligned} &\forall x \neq x' \in \mathcal{X}, z \neq z' \in \mathcal{Z}\ \exists y_1, y_2 \in \mathcal{Y}, w_1, w_2 \in \mathcal{W} : \\ &\qquad h(x, z, y_1, w_1) - h(x, z, y_2, w_2) \neq h(x', z', y_1, w_1) - h(x', z', y_2, w_2) \end{aligned} \tag{7.5}$$

Proof. We start by proving that the function h has this property:

$$\begin{aligned} &\forall x, x' \in \mathcal{X}, z, z' \in \mathcal{Z}((x, z) \neq (x', z') \to \\ &\qquad \exists y \in \mathcal{Y}, w \in \mathcal{W} : h(x, z, y, w) \neq h(x', z', y, w) \end{aligned} \tag{7.6}$$

By assumption we consider the following two properties on the function g:

$$\forall z \neq z' \in \mathcal{Z}\ \exists w \in \mathcal{W} : g(z,w) \neq g(z',w) \tag{7.7}$$

$$\forall z \neq z' \in \mathcal{Z}\ \exists w_1, w_2 \in \mathcal{W} : g(z,w_1) - g(z,w_2) \neq g(z',w_1) - g(z',w_2) \tag{7.8}$$

and properties (7.3) and (7.4).

In order to prove properties (7.6) and (7.5) for the function h we proceed as follows:

Case 1. $x = x', z \neq z'$:
Suppose that $\exists y$ such that $f(x',y) = f(x',y)$. By assumption $\exists w \in \mathcal{W} : g(z,w) \neq g(z',w)$. Therefore, it follows that $f(x',y) - f(x,y) \neq g(z,w) - g(z',w)$ and property (7.6) holds.

Case 2. $x \neq x', z = z'$:
Suppose that $\exists w$ such that $g(z',w) = g(z',w)$. By assumption $\exists y \in \mathcal{Y} : f(x,y) \neq g(x',y)$. It follows that $f(x',y) - f(x,y) \neq g(z,w) - g(z',w)$ and property (7.6) holds.

Case 3. $x \neq x', z \neq z'$:
Let $c = f(x',y) - f(x,y)$ for some $y \in \mathcal{Y}$. By assumption $\exists w_1, w_2 \in \mathcal{W}$ such that $c_1 = g(z,w_1) - g(z',w_1)$ and $c_2 = g(z,w_2) - g(z',w_2)$ such that $c_1 \neq c_2$. Without loss of generality, assume that $c \neq c_1$ then $f(x',y) - f(x,y) \neq g(z,w_1) - g(z',w_1)$ and property (7.5) follows.

Since the function h satisfy property (7.6) it also has property (7.3) and hence we get from Theorem 7.5 that

$$H(U_h|V_h) \geq \max_{y,w} H(X,Z|h(X,Z,y,w)) \ .$$

We then get that:

$$\begin{aligned} H(U_h|V_h) &\geq \max_{y,w} H(X,Z|\alpha f(X,y) + \beta g(Z,w)) && (7.9) \\ &\geq \max_{y,w} H(X,Z|f(X,y), g(Z,w)) && (7.10) \\ &\geq \max_{y} H(X|f(X,y)) + \max_{w} H(Z|g(Z,w)) && (7.11) \end{aligned}$$

Inequality (7.11) follows from the independence of X, Z. This proves the theorem. ■

Remark 7.1. The above theorem also applies to multiplicative relations ruling out the cases where $g(z,w) = 0$ and $f(x,y) = 0$.

Exploiting Theorem 7.6 we prove a lower bound for the inner product function IP_k as per Definition 7.6.

Lemma 7.7. Let $\kappa \geq 1$ and let $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ be a multiplication function as per Definition 7.5. If there exist a protocol Π_{IP_k} which securely implements the inner product function IP_k with error probability ε in the semi-honest model and having access to $P_{U_{\mathrm{IP}_k} V_{\mathrm{IP}_k}}$ then

$$H(U_{\mathrm{IP}_k}|V_{\mathrm{IP}_k}) \geq k \cdot \max_{y} H(X|f(X,y)) \tag{7.12}$$

Proof. Since the function f satisfies properties (7.3) and (7.4), a straightforward application of Theorem 7.6 for $k = 2$ yields $H(U_{\mathrm{IP}_2}|V_{\mathrm{IP}_2}) \geq 2 \cdot \max_y H(X|f(X,y))$. However it is easy to see that the proof of Theorem 7.6 extends to addition of k functions for any k, so the lemma follows in the same way from this more general result. ∎

Utilising Theorem 7.6 in the following we prove that any function whose "preprocessing complexity" is large enough requires lots of communication. What "large enough" means here is determined by the output secret-sharing scheme used in the protocol, in a sense we make precise below. In the following, when f is a function with two inputs and one output, we will speak about *a protocol for computing shares of an f-output*, denoted by $\Pi_{f-output}$. This is essentially the same as an MGP except that we replace multiplication by f. So the protocol takes as input shares of x_1 and x_2 and computes shares of $f(x_1, x_2)$ as output. Note that the inputs x_1, x_2 may be vectors of field elements, whereas we will by default assume that the output is a single field element.

In the sequel, for simplicity of exposition let $\mathtt{L}_f$ denote a lower bound on the amount of preprocessed data needed for a secure implementation of f in the preprocessing model and let $\mathtt{U}_f$ denote an upper bound.

Reconstruction Protocol Π_{rec}. Let $\mathscr{S}_t^n$ be the secret-sharing scheme as per Definition 7.3 and let $f'_{\mathscr{S}_t^n}$ be the reconstruction function of $\mathscr{S}_t^n$. Then, we can securely implement the function $f'_{\mathscr{S}_t^n}$ in the preprocessing model via the protocol Π_{SPDZ} yielding the protocol Π_{rec}.[5]. It follows that Π_{rec} demands communication and that its complexity depends only on the underlying secret-sharing scheme $\mathscr{S}_t^n$. In this case we obtain an upper bound $\mathtt{U}_{rec}$ on the amount of preprocessed data consumed by Π_{rec}.

Theorem 7.8. Consider the preprocessing model where t of the n players may be passively corrupted. Let Π_{rec} be a secure output reconstruction protocol with access to $P_{U_{rec},V_{rec}}$ for the secret-sharing scheme $\hat{\mathscr{S}}_t^n$. Let f be a function with two inputs and one field element as output such that $\mathtt{U}_{rec} < \mathtt{L}_f$. There exists no passively secure n-player protocol $\Pi_{f-output}$ with expected anticipated communication complexity $\leq t$ for computing shares of an f-output with $\hat{\mathscr{S}}_t^n$ as output secret-sharing scheme.

Proof. We start by assuming a fixed communication pattern. Suppose for contradiction that there exists a protocol Π_f where at most t players communicate. Assume that it is the t first parties. Given two parties A and B, we are going to construct a two-party protocol $\Pi_f^{A,B}$ which on input a, b from A, B, respectively, securely computes $f(a, b)$. The idea is to execute the $\Pi_{f-output}$ protocol in which A emulates the t players who communicate while B emulates the rest of the parties but we are interest just for one additional party, say P_{t+1}. In particular, protocol $\Pi_f^{A,B}(a, b)$ proceeds as follows:

Protocol $\Pi_f^{A,B}(a, b)$:

Input Phase:

[5] Note that any protocol in the preprocessing model can be used.

1. Parties A, B secret share their inputs a, b using the secret-sharing scheme $\mathscr{S}_t^n$. More specifically, A computes $[a]^{\mathscr{S}_t^n} \leftarrow \mathsf{Share}(a, n, t)$ and B computes $[b]^{\mathscr{S}_t^n} \leftarrow \mathsf{Share}(b, n, t)$.
2. Party A sends the input share $(a_{t+1}, \ldots, a_n)$ to party B and Party B sends the input shares $(b_1, \ldots, b_t)$ to party A.

Evaluation Phase:

1. Parties A, B invoke the protocol $\Pi_{f-output}$ where A emulates the t players who communicate, and we assume these are the first t players. This means that this phase involves no communication between A and B, but it may consume some preprocessed data. The execution of $\Pi_{f-output}$ yields a sharing of $[c]^{\mathscr{S}_t^n}$ and outputs $(c_1, ..., c_t)$ to party A and $(c_{t+1}, \ldots, c_n)$ to party B.

Output Phase:

1. Both parties locally invoke protocol Π_{Rec} with access to $P_{U_{rec}, V_{rec}}$ which on input $[c]^{\hat{\mathscr{S}}_t^n}$ outputs the result $f(a, b)$.

Correctness of the protocol follows immediately from the correctness of $\Pi_{f-output}$ and Π_{Rec}. The protocol can be argued to be secure(private). More specifically, the simulator $\mathsf{S}_{\mathcal{A}}$ of $\Pi_f^{A,B}$ proceeds as follows. In the input phase, the parties receive only an unqualified set of shares whose distribution can be simulated perfectly. There is no communication to be simulated in the evaluation phase. In the output phase, simulation is guaranteed by the invocations of the sub-simulator of the secure protocol Π_{Rec}. Hence, it follows that

$$\mathrm{SD}(\mathsf{Exec}_{\mathsf{Adv}}^{\Pi_f^{A,B}}(\sigma, (a, b)), \mathcal{S}_{\mathsf{S}_{\mathcal{A}}}(\sigma, (a, b))) \leq \varepsilon(\sigma).$$

We can claim the following: Note that the communication in $\Pi_f^{A,B}$ is actually independent of the preprocessed data needed in order to securely compute f. Therefore, while reusing the same preprocessed data for each invocation of $\Pi_{f-output}$, we could have executed ℓ instances of $\Pi_f^{A,B}$ on independent inputs without affecting correctness since the simulation is independent of the preprocessed data. However, since protocol Π_{Rec} is interactive its preprocessed data must be refreshed for each of the ℓ executions of Π_{Rec}. This means that the amount of preprocessed data needed in order to compute ℓ instances of f is $\mathtt{U}_f + \ell \cdot \mathtt{U}_{rec}$. So if we consider an ℓ large enough such that $\ell \cdot \mathtt{L}_f > \mathtt{U}_f + \ell \cdot \mathtt{U}_{rec}$, we have a contradiction and the theorem follows.

The generalization to dynamic communication patterns follows along the lines of the proof of Theorem 7.4: there we split the players in a maximal unqualified set ($n-1$ players) and the rest (1 player). Here we do the same except that the maximal unqualified set has t players and $n-t$ remain. We then argue exactly as in the proof of Theorem 7.4 that decisions to send/receive cannot depend on private randomness or shares, and therefore we can build a new protocol that can be used in our construction of a 2-party protocol.

■

Given a function f with one output and a non-zero lower bound, we can add it to itself on distinct inputs a sufficient number of times in order to satisfy the condition in the above theorem.

An example of a function f is the inner product function IP_k which is the composition of k MULT functions. In Lemma 7.7 we obtained a lower bound $\mathtt{L}_{\mathrm{IP}^k}$ on the amount of preprocessed data consumed by a protocol that securely implements the function IP^k. Now, if k is large enough to satisfy the condition $\mathtt{U}_{rec} < \mathtt{L}_{\mathrm{IP}_k}$, then it holds that $\ell \cdot \mathtt{U}_{rec} + \mathtt{L}_{\mathrm{MULT}} < \ell \cdot \mathtt{L}_{\mathrm{IP}_k}$ for large enough ℓ leading to a contradiction with Theorem 7.8.

7.5 On the Necessity of Model Extension

In this section we provide a simple example to illustrate why deviations made by an active adversary from passively secure protocols based on packed secret sharing schemes, result in a *linear* attack on the underlaying SIMD circuit, as opposed to an additive attack. Indeed consider the n-party protocol where P_1 holds an input block $\mathbf{a} = (\mathtt{a}^1, \mathtt{a}^2) \in \mathbb{F}^2$, P_2 holds an input block $\mathbf{b} = (\mathtt{b}^1, \mathtt{b}^2) \in \mathbb{F}^2$ and after the protocol terminates P_2 should learn $\mathtt{c}^2 = \mathtt{a}^2 \otimes \mathtt{b}^2$. Consider the following simplified passively secure MPC protocol against an adversary corrupting P_1.

1. P_1 computes and deals among all parties a degree-d packed Shamir sharing $[\mathbf{a}]_d$.
2. P_2 computes and deals among all parties a degree-d packed Shamir sharing $[\mathbf{b}]_d$.
3. Every party P_i locally computes $c_i = a_i \cdot b_i$ and sends the result to P_2.
4. P_2 recovers the value c^2 in $\mathbf{c}$ by computing some fixed linear combinations on $(c_1, \cdots, c_n)$. That is, $c^2 = \sum_{i=1}^n \gamma_i c_i$ for some $\gamma_1, \cdots, \gamma_n \in \mathbb{F}$.

Next, notice that since $[\mathbf{b}]_d$ is a degree-d packed Shamir sharing, any set of $d+1$ parties can completely recover $\mathbf{b}$. Thus, there exists $\delta_1^1, \cdots, \delta_1^{d+1}$ such that $b^1 = \sum_{i=1}^{d+1} \delta_i^1 b_i$.

Next, consider an adversary corrupting P_1 instructing it to modify the shares $(a_1, \cdots, a_n)$ computed in Step 1 above by setting $a_i \leftarrow a_i + \delta_i/\gamma_i$, for all $1 \leq i \leq d+1$. Next we examine the effects of this attack on the outputs of P_2 in Step 4 above. Indeed, notice that

$$c^2 = \sum_{i=1}^{n} \gamma_i c_i = \sum_{i=1}^{d+1} \gamma_i (a_i + \delta_i/\gamma_i) b_i + \sum_{i=d+2}^{n} \gamma_i a_i b_i = \sum_{i=1}^{n} \gamma_i a_i b_i + \sum_{i=1}^{d+1} \delta_i b_i = a^2 b^2 + b^1.$$

Notice that the above attack changed the output value of P_2 from a^2b^2 to $a^2b^2 + b^1$. Such an effect cannot be achieved by any additive attack on the circuit computing $\mathbf{c} = \mathbf{a} \otimes \mathbf{b}$ since an additive attack cannot mix intermediate values between the two different copies being simultaneously evaluated. However, this effect on the output of P_2 can be easily simulated using a linear attack on the second wire in the output bundle corresponding to $\mathbf{c}$, adding to the it the first wire in the input bundle corresponding to $\mathbf{b}$.

Bibliography

[AIK05] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Computationally private randomizing polynomials and their applications. In *20th Annual IEEE Conference on Computational Complexity (CCC 2005), 11-15 June 2005, San Jose, CA, USA*, pages 260–274, 2005. 31, 32

[AJL$^+$12] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In *EUROCRYPT*, pages 483–501, 2012. 32, 33, 126

[AJW11] Gilad Asharov, Abhishek Jain, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. Cryptology ePrint Archive, Report 2011/613, 2011. http://eprint.iacr.org/2011/613. 117

[AMP10] Gagan Aggarwal, Nina Mishra, and Benny Pinkas. Secure computation of the median (and other elements of specified ranks). *J. Cryptology*, 23(3):373–401, 2010. 22

[App14] Benny Applebaum. *Cryptography in Constant Parallel Time.* Information Security and Cryptography. Springer, 2014. 36

[Bar01] Boaz Barak. How to go beyond the black-box simulation barrier. In *42nd Annual Symposium on Foundations of Computer Science*, pages 106–115. IEEE Computer Society Press, October 2001. 32

[Bar02] Boaz Barak. Constant-round coin-tossing with a man in the middle or realizing the shared random string model. In *43rd Annual Symposium on Foundations of Computer Science*, pages 345–355. IEEE Computer Society Press, November 2002. 6, 31

[BCH12] Nir Bitansky, Ran Canetti, and Shai Halevi. Leakage-tolerant interactive protocols. In Ronald Cramer, editor, *TCC 2012: 9th Theory of Cryptography Conference*, volume 7194 of *Lecture Notes in Computer Science*, pages 266–284. Springer, Heidelberg, March 2012. 110

[BCHK07] Dan Boneh, Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. *SIAM Journal on Computing*, 36(5):1301–1328, 2007. 130

[BCNP04] Boaz Barak, Ran Canetti, Jesper Buus Nielsen, and Rafael Pass. Universally composable protocols with relaxed set-up assumptions. In *FOCS*, pages 186–195, 2004. 12

[BD10] Rikke Bendlin and Ivan Damgård. Threshold decryption and zero-knowledge proofs for lattice-based cryptosystems. In Daniele Micciancio, editor, *TCC 2010: 7th*

Theory of Cryptography Conference, volume 5978 of *Lecture Notes in Computer Science*, pages 201–218. Springer, Heidelberg, February 2010. 32

[BDOZ11] Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. Semi-homomorphic encryption and multiparty computation. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 169–188. Springer, Heidelberg, May 2011. 33

[BELO14] Joshua Baron, Karim El Defrawy, Joshua Lampkins, and Rafail Ostrovsky. How to withstand mobile virus attacks, revisited. In Magnús M. Halldórsson and Shlomi Dolev, editors, *33rd ACM Symposium Annual on Principles of Distributed Computing*, pages 293–302. Association for Computing Machinery, July 2014. 23

[BFKR91] Donald Beaver, Joan Feigenbaum, Joe Kilian, and Phillip Rogaway. Security with low communication overhead. In Alfred J. Menezes and Scott A. Vanstone, editors, *Advances in Cryptology – CRYPTO'90*, volume 537 of *Lecture Notes in Computer Science*, pages 62–76. Springer, Heidelberg, August 1991. 20

[BFM90] Manuel Blum, Paul Feldman, and Silvio Micali. Proving security against chosen cyphertext attacks. In Shafi Goldwasser, editor, *Advances in Cryptology – CRYPTO'88*, volume 403 of *Lecture Notes in Computer Science*, pages 256–268. Springer, Heidelberg, August 1990. 111

[BFO12] Eli Ben-Sasson, Serge Fehr, and Rafail Ostrovsky. Near-linear unconditionally-secure multiparty computation with a dishonest minority. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 663–680. Springer, Heidelberg, August 2012. 20, 35, 38

[BGI$^+$12] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6, 2012. 107

[BGJ$^+$13] Elette Boyle, Sanjam Garg, Abhishek Jain, Yael Tauman Kalai, and Amit Sahai. Secure computation against adaptive auxiliary information. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 316–334. Springer, Heidelberg, August 2013. 110

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012: 3rd Innovations in Theoretical Computer Science*, pages 309–325. Association for Computing Machinery, January 2012. 32

[BHP17] Zvika Brakerski, Shai Halevi, and Antigoni Polychroniadou. Four round secure computation without setup. Cryptology ePrint Archive, Report 2017/386, 2017. 8

[BI05] Omer Barkol and Yuval Ishai. Secure computation of constant-depth circuits with applications to database search problems. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 395–411. Springer, Heidelberg, August 2005. 20

[BIP17] Elette Boyle, Yuval Ishai, and Antigoni Polychroniadou. Limits of practical sublinear secure computation. Manuscript, 2017. 22

[BL02] Boaz Barak and Yehuda Lindell. Strict polynomial-time in simulation and extraction. In *34th Annual ACM Symposium on Theory of Computing*, pages 484–493. ACM Press, May 2002. 49

[BMR90] Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In *22nd Annual ACM Symposium on Theory of Computing*, pages 503–513. ACM Press, May 1990. 5, 31

[BOGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *20th Annual ACM Symposium on Theory of Computing*, pages 1–10. ACM Press, May 1988. 3, 21, 22, 33, 35, 38, 147

[BP16] Zvika Brakerski and Renen Perlman. Lattice-based fully dynamic multi-key FHE with short ciphertexts. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 190–213. Springer, Heidelberg, August 2016. 32

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *CCS*, pages 62–73, 1993. 17

[Bra87] Gabriel Bracha. An O(log n) expected rounds randomized byzantine generals protocol. *J. ACM*, 34(4):910–920, 1987. 23

[BS05a] Boaz Barak and Amit Sahai. How to play almost any mental game over the net - concurrent composition via super-polynomial simulation. In *FOCS*, pages 543–552, 2005. 17, 68

[BS05b] Justin Brickell and Vitaly Shmatikov. Privacy-preserving graph algorithms in the semi-honest model. In *Advances in Cryptology - ASIACRYPT 2005, 11th International Conference on the Theory and Application of Cryptology and Information Security, Chennai, India, December 4-8, 2005, Proceedings*, pages 236–252, 2005. 22

[BSPV99] Carlo Blundo, Alfredo De Santis, Giuseppe Persiano, and Ugo Vaccaro. Randomness complexity of private computation. *Computational Complexity*, 8(2):145–168, 1999. 36

[BTH08] Zuzana Beerliová-Trubíniová and Martin Hirt. Perfectly-secure MPC with linear communication complexity. In Ran Canetti, editor, *TCC 2008: 5th Theory of Cryptography Conference*, volume 4948 of *Lecture Notes in Computer Science*, pages 213–230. Springer, Heidelberg, March 2008. 20, 35

[BV11a] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *52nd Annual Symposium on Foundations of Computer Science*, pages 97–106. IEEE Computer Society Press, October 2011. 27, 32

[BV11b] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 505–524. Springer, Heidelberg, August 2011. 27, 127, 130, 131

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science*, pages 136–145. IEEE Computer Society Press, October 2001. 5, 9, 12, 67, 68

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *20th Annual ACM Symposium on Theory of Computing*, pages 11–19. ACM Press, May 1988. 3, 21, 22, 33, 35, 147

[CDD$^+$04] Ran Canetti, Ivan Damgård, Stefan Dziembowski, Yuval Ishai, and Tal Malkin. Adaptive versus non-adaptive security of multi-party protocols. *Journal of Cryptology*, 17(3):153–207, June 2004. 9

[CDF$^+$08] Ronald Cramer, Yevgeniy Dodis, Serge Fehr, Carles Padró, and Daniel Wichs. Detection of algebraic manipulation with applications to robust secret sharing and fuzzy extractors. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 471–488. Springer, Heidelberg, April 2008. 11

[CDG$^+$17] Chongwon Cho, Nico Döttling, Sanjam Garg, Divya Gupta, Peihan Miao, and Antigoni Polychroniadou. Laconic oblivious transfer and its applications. *IACR Cryptology ePrint Archive*, 2017:491, 2017. 27

[CDN01] Ronald Cramer, Ivan Damgård, and Jesper Buus Nielsen. Multiparty computation from threshold homomorphic encryption. In Birgit Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 280–299. Springer, Heidelberg, May 2001. 33

[CDPW07] Ran Canetti, Yevgeniy Dodis, Rafael Pass, and Shabsi Walfish. Universally composable security with global setup. In Salil P. Vadhan, editor, *TCC 2007: 4th Theory of Cryptography Conference*, volume 4392 of *Lecture Notes in Computer Science*, pages 61–85. Springer, Heidelberg, February 2007. 12, 15, 17, 65, 68, 69

[CF01] Ran Canetti and Marc Fischlin. Universally composable commitments. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 19–40. Springer, Heidelberg, August 2001. 9, 12, 17, 68

[CF02] Ronald Cramer and Serge Fehr. Optimal black-box secret sharing over arbitrary Abelian groups. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 272–287. Springer, Heidelberg, August 2002. 149

[CFGN96] Ran Canetti, Uriel Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *28th Annual ACM Symposium on Theory of Computing*, pages 639–648. ACM Press, May 1996. 11, 33

[CGP15] Ran Canetti, Shafi Goldwasser, and Oxana Poburinnaya. Adaptively secure two-party computation from indistinguishability obfuscation. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 557–585. Springer, Heidelberg, March 2015. 17, 33

[CGS08] Nishanth Chandran, Vipul Goyal, and Amit Sahai. New constructions for UC secure computation using tamper-proof hardware. In *EUROCRYPT*, pages 545–562, 2008. 12, 13, 34, 35, 73

[CJS14] Ran Canetti, Abhishek Jain, and Alessandra Scafuro. Practical UC security with a global random oracle. In *CCS*, pages 597–608, 2014. 15, 17, 19, 68, 70, 71, 72

[CK93] Benny Chor and Eyal Kushilevitz. A communication-privacy tradeoff for modular addition. *Inf. Process. Lett.*, 45(4):205–210, 1993. 36

[CKL03] Ran Canetti, Eyal Kushilevitz, and Yehuda Lindell. On the limitations of universally composable two-party computation without set-up assumptions. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 68–86. Springer, Heidelberg, May 2003. 9, 12, 17

[CKL06] Ran Canetti, Eyal Kushilevitz, and Yehuda Lindell. On the limitations of universally composable two-party computation without set-up assumptions. *J. Cryptology*, 19(2):135–167, 2006. 68

[CKS$^+$13] Seung Geol Choi, Jonathan Katz, Dominique Schröder, Arkady Yerukhimovich, and Hong-Sheng Zhou. (efficient) universally composable oblivious transfer using a minimal number of stateless tokens. *IACR Cryptology ePrint Archive*, 2013:840, 2013. 14

[CKS$^+$14] Seung Geol Choi, Jonathan Katz, Dominique Schröder, Arkady Yerukhimovich, and Hong-Sheng Zhou. (efficient) universally composable oblivious transfer using a minimal number of stateless tokens. In *TCC*, pages 638–662, 2014. 12, 13, 14, 15, 34, 35, 71, 73

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *34th Annual ACM Symposium on Theory of Computing*, pages 494–503. ACM Press, May 2002. 9, 14, 22, 31, 33, 65, 66, 68, 115, 117, 128, 129, 130

[CLP10] Ran Canetti, Huijia Lin, and Rafael Pass. Adaptive hardness and composable security in the plain model from standard assumptions. In *FOCS*, pages 541–550, 2010. 12, 17

[CM15] Michael Clear and Ciaran McGoldrick. Multi-identity and multi-key leveled FHE from learning with errors. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 630–656. Springer, Heidelberg, August 2015. 32

[COSV16a] Michele Ciampi, Rafail Ostrovsky, Luisa Siniscalchi, and Ivan Visconti. 4-round concurrent non-malleable commitments from one-way functions. Cryptology ePrint Archive, Report 2016/621, 2016. http://eprint.iacr.org/2016/621. 8

[COSV16b] Michele Ciampi, Rafail Ostrovsky, Luisa Siniscalchi, and Ivan Visconti. Concurrent non-malleable commitments (and more) in 3 rounds. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part III*, volume 9816 of *Lecture Notes in Computer Science*, pages 270–299. Springer, Heidelberg, August 2016. 7, 8, 106

[COSV16c] Michele Ciampi, Rafail Ostrovsky, Luisa Siniscalchi, and Ivan Visconti. On round-efficient non-malleable protocols. *IACR Cryptology ePrint Archive*, 2016:621, 2016. 106

[CP16] Ran Canetti and Oxana Poburinnaya. Better two-round adaptive multiparty computation. Cryptology ePrint Archive, Report 2016/614, 2016. http://eprint.iacr.org/2016/614. 10, 33

[CPS07] Ran Canetti, Rafael Pass, and Abhi Shelat. Cryptography from sunspots: How to use an imperfect reference string. In *FOCS*, pages 249–259, 2007. 12

[CPS$^+$16a] Michele Ciampi, Giuseppe Persiano, Alessandra Scafuro, Luisa Siniscalchi, and Ivan Visconti. Improved or-composition of sigma-protocols. In *TCC*, pages 112–141, 2016. 67

[CPS$^+$16b] Michele Ciampi, Giuseppe Persiano, Alessandra Scafuro, Luisa Siniscalchi, and Ivan Visconti. Online/offline OR composition of sigma protocols. In *EUROCRYPT*, pages 63–92, 2016. 67

[CPV16] Ran Canetti, Oxana Poburinnaya, and Muthuramakrishnan Venkitasubramaniam. Equivocating yao: Constant-round adaptively secure multiparty computation in the plain model. Cryptology ePrint Archive, Report 2016/1190, 2016. http://eprint.iacr.org/2016/1190. 19, 34

[CR03] Ran Canetti and Tal Rabin. Universal composition with joint state. In *CRYPTO*, pages 265–281, 2003. 68

[DDN91a] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography. In *Proceedings of the Twenty-third Annual ACM Symposium on Theory of Computing*, STOC '91, pages 542–552, New York, NY, USA, 1991. ACM. 129

[DDN91b] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography (extended abstract). In *23rd Annual ACM Symposium on Theory of Computing*, pages 542–552. ACM Press, May 1991. 6, 31

[DDO$^+$01] Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 566–598. Springer, Heidelberg, August 2001. 33

[DHP11] Ivan Damgård, Carmit Hazay, and Arpita Patra. Leakage resilient secure two-party computation. *IACR Cryptology ePrint Archive*, 2011:256, 2011. 110

[DI05] Ivan Damgård and Yuval Ishai. Constant-round multiparty computation using a black-box pseudorandom generator. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 378–394. Springer, Heidelberg, August 2005. 5, 31, 33

[DI06] Ivan Damgård and Yuval Ishai. Scalable secure multiparty computation. In Cynthia Dwork, editor, *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 501–520. Springer, Heidelberg, August 2006. 5, 31, 33

[DIK$^+$08] Ivan Damgård, Yuval Ishai, Mikkel Krøigaard, Jesper Buus Nielsen, and Adam Smith. Scalable multiparty computation with nearly optimal work and resilience. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 241–261. Springer, Heidelberg, August 2008. 21, 33, 152

[DIK10] Ivan Damgård, Yuval Ishai, and Mikkel Krøigaard. Perfectly secure multiparty computation and the computational overhead of cryptography. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 445–465. Springer, Heidelberg, May 2010. 21, 23, 25, 35, 38

[DKL$^+$13] Ivan Damgård, Marcel Keller, Enrique Larraia, Valerio Pastro, Peter Scholl, and Nigel P. Smart. Practical covertly secure MPC for dishonest majority - or: Breaking the SPDZ limits. In Jason Crampton, Sushil Jajodia, and Keith Mayes, editors, *ESORICS 2013: 18th European Symposium on Research in Computer Security*, volume 8134 of *Lecture Notes in Computer Science*, pages 1–18. Springer, Heidelberg, September 2013. 33

[DKM11] Nico Döttling, Daniel Kraschewski, and Jörn Müller-Quade. Unconditional and composable security using a single stateful tamper-proof hardware token. In *TCC*, pages 164–181, 2011. 13, 35, 73

[DKMN15a] Nico Döttling, Daniel Kraschewski, Jörn Müller-Quade, and Tobias Nilges. From stateful hardware to resettable hardware using symmetric assumptions. In *ProvSec*, pages 23–42, 2015. 15, 73

[DKMN15b] Nico Döttling, Daniel Kraschewski, Jörn Müller-Quade, and Tobias Nilges. General statistically secure computation with bounded-resettable hardware tokens. In *TCC*, pages 319–344, 2015. 12, 13, 35

[DKR15] Dana Dachman-Soled, Jonathan Katz, and Vanishree Rao. Adaptively secure, universally composable, multiparty computation in constant rounds. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 586–613. Springer, Heidelberg, March 2015. 17, 33

[DMMN13] Nico Döttling, Thilo Mie, Jörn Müller-Quade, and Tobias Nilges. Implementing resettable uc-functionalities with untrusted tamper-proof hardware-tokens. In *TCC*, pages 642–661, 2013. 13, 35

[DMRV13] Dana Dachman-Soled, Tal Malkin, Mariana Raykova, and Muthuramakrishnan Venkitasubramaniam. Adaptive and concurrent secure computation from new adaptive, non-malleable commitments. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology – ASIACRYPT 2013, Part I*, volume 8269 of *Lecture Notes in Computer Science*, pages 316–336. Springer, Heidelberg, December 2013. 9, 12, 17, 33, 35

[DN03] Ivan Damgård and Jesper Buus Nielsen. Universally composable efficient multiparty computation from threshold homomorphic encryption. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 247–264. Springer, Heidelberg, August 2003. 33, 128

[DN07] Ivan Damgård and Jesper Buus Nielsen. Scalable and unconditionally secure multiparty computation. In Alfred Menezes, editor, *Advances in Cryptology – CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 572–590. Springer, Heidelberg, August 2007. 20, 25, 35, 38

[DNO10] Ivan Damgård, Jesper Buus Nielsen, and Claudio Orlandi. On the necessary and sufficient assumptions for UC computation. In Daniele Micciancio, editor, *TCC 2010: 7th Theory of Cryptography Conference*, volume 5978 of *Lecture Notes in Computer Science*, pages 109–127. Springer, Heidelberg, February 2010. 12

[DNOR16] Ivan Damgård, Jesper Buus Nielsen, Rafail Ostrovsky, and Adi Rosén. Unconditionally secure computation with reduced interaction. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 420–447. Springer, Heidelberg, May 2016. 37

[DNPR16] Ivan Damgård, Jesper Buus Nielsen, Antigoni Polychroniadou, and Michael Raskin. On the communication required for unconditionally secure multiplication. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 459–488. Springer, Heidelberg, August 2016. 5, 21, 28, 31, 33, 35, 36, 37, 139

[DPP14] Deepesh Data, Manoj Prabhakaran, and Vinod M. Prabhakaran. On the communication complexity of secure computation. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part II*, volume 8617 of *Lecture Notes in Computer Science*, pages 199–216. Springer, Heidelberg, August 2014. 36, 37, 142

[DPR16] Ivan Damgård, Antigoni Polychroniadou, and Vanishree Rao. Adaptively secure multi-party computation from LWE (via equivocal FHE). In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *PKC 2016: 19th International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 9615 of *Lecture Notes in Computer Science*, pages 208–233. Springer, Heidelberg, March 2016. 5, 10, 11, 28, 31, 33, 107, 126, 136

[DPSZ12] Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 643–662. Springer, Heidelberg, August 2012. 21, 33, 35, 141, 150, 156

[DZ13] Ivan Damgård and Sarah Zakarias. Constant-overhead secure computation of Boolean circuits using preprocessing. In Amit Sahai, editor, *TCC 2013: 10th Theory of Cryptography Conference*, volume 7785 of *Lecture Notes in Computer Science*, pages 621–641. Springer, Heidelberg, March 2013. 33, 77

[FDB93] Yair Frankel, Yvo Desmedt, and Mike Burmester. Non-existence of homomorphic general sharing schemes for some key spaces (extended abstract). In Ernest F. Brickell, editor, *Advances in Cryptology – CRYPTO'92*, volume 740 of *Lecture Notes in Computer Science*, pages 549–557. Springer, Heidelberg, August 1993. 37

[Fei90] Uriel Feige. *Alternative Models for Zero Knowledge Interactive Proofs*. PhD thesis, 1990. 46

[FH96] Matthew Franklin and Stuart Haber. Joint encryption and message-efficient secure computation. *J. Cryptol.*, 1996. 33

[FJN$^+$13] Tore Kasper Frederiksen, Thomas Pelle Jakobsen, Jesper Buus Nielsen, Peter Sebastian Nordholt, and Claudio Orlandi. MiniLEGO: Efficient secure two-party computation from general assumptions. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 537–556. Springer, Heidelberg, May 2013. 23

[FJNT15] Tore Kasper Frederiksen, Thomas P. Jakobsen, Jesper Buus Nielsen, and Roberto Trifiletti. Tinylego: An interactive garbling scheme for maliciously secure two-party computation. Cryptology ePrint Archive, Report 2015/309, 2015. `http://eprint.iacr.org/2015/309`. 23

[FKN94] Uriel Feige, Joe Kilian, and Moni Naor. A minimal model for secure computation (extended abstract). In *26th Annual ACM Symposium on Theory of Computing*, pages 554–563. ACM Press, May 1994. 36

[FLS90] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In *31st Annual Symposium on Foundations of Computer Science*, pages 308–317. IEEE Computer Society Press, October 1990. 111

[FLS99] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple noninteractive zero knowledge proofs under general assumptions. *SIAM J. Comput.*, 29(1):1–28, 1999. 46

[FS90a] Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *22nd Annual ACM Symposium on Theory of Computing*, pages 416–426. ACM Press, May 1990. 46

[FS90b] Uriel Feige and Adi Shamir. Zero knowledge proofs of knowledge in two rounds. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 526–544. Springer, Heidelberg, August 1990. 115

[FY92] Matthew K. Franklin and Moti Yung. Communication complexity of secure computation (extended abstract). In *24th Annual ACM Symposium on Theory of Computing*, pages 699–710. ACM Press, May 1992. 23, 35, 36, 38

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 169–178. ACM Press, May / June 2009. 27, 33

[GGG$^+$14] Shafi Goldwasser, S. Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 578–602. Springer, Heidelberg, May 2014. 109

[GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 1–17. Springer, Heidelberg, May 2013. 107, 111

[GGH$^+$13b] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th Annual Symposium on Foundations of Computer Science*, pages 40–49. IEEE Computer Society Press, October 2013. 8, 33, 107, 109, 111

[GGHR14] Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana Raykova. Two-round secure MPC from indistinguishability obfuscation. In Yehuda Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, pages 74–94. Springer, Heidelberg, February 2014. 8, 32, 44, 67, 107, 108

[GGKS14] Sanjam Garg, Divya Gupta, Dakshita Khurana, and Amit Sahai. All-but-one leakage resilient multiparty computation and incoercible multiparty computation. Personal Communication, 2014. 110

[GGMP16] Sanjam Garg, Divya Gupta, Peihan Miao, and Omkant Pandey. Secure multiparty RAM computation in constant rounds. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B: 14th Theory of Cryptography Conference, Part I*, volume 9985 of *Lecture Notes in Computer Science*, pages 491–520. Springer, Heidelberg, October / November 2016. 27

[GHL$^+$14] Craig Gentry, Shai Halevi, Steve Lu, Rafail Ostrovsky, Mariana Raykova, and Daniel Wichs. Garbled RAM revisited. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 405–422. Springer, Heidelberg, May 2014. 27

[GHV10] Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. i-Hop homomorphic encryption and rerandomizable Yao circuits. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 155–172. Springer, Heidelberg, August 2010. 28

[GIKR02] Rosario Gennaro, Yuval Ishai, Eyal Kushilevitz, and Tal Rabin. On 2-round secure multiparty computation. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 178–193. Springer, Heidelberg, August 2002. 35

[GIP$^+$14] Daniel Genkin, Yuval Ishai, Manoj Prabhakaran, Amit Sahai, and Eran Tromer. Circuits resilient to additive attacks with applications to secure computation. In David B. Shmoys, editor, *46th Annual ACM Symposium on Theory of Computing*, pages 495–504. ACM Press, May / June 2014. 20, 23, 24, 35, 38

[GIP15] Daniel Genkin, Yuval Ishai, and Antigoni Polychroniadou. Efficient multi-party computation: From passive to active security via secure SIMD circuits. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 721–741. Springer, Heidelberg, August 2015. 5, 20, 21, 24, 26, 29, 35, 38, 139

[GIS$^+$10] Vipul Goyal, Yuval Ishai, Amit Sahai, Ramarathnam Venkatesan, and Akshay Wadia. Founding cryptography on tamper-proof hardware tokens. In *TCC*, pages 308–326, 2010. ix, 12, 13, 14, 16, 34, 35, 65, 66, 73, 80, 92, 97, 105, 106

[GIW16] Daniel Genkin, Yuval Ishai, and Mor Weiss. Binary AMD circuits from secure multiparty computation. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B: 14th Theory of Cryptography Conference, Part I*, volume 9985 of *Lecture Notes in Computer Science*, pages 336–366. Springer, Heidelberg, October / November 2016. 26

[GJS11] Sanjam Garg, Abhishek Jain, and Amit Sahai. Leakage-resilient zero knowledge. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 297–315. Springer, Heidelberg, August 2011. 110

[GKK$^+$12] S. Dov Gordon, Jonathan Katz, Vladimir Kolesnikov, Fernando Krell, Tal Malkin, Mariana Raykova, and Yevgeniy Vahlis. Secure two-party computation in sublinear (amortized) time. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *ACM CCS 12: 19th Conference on Computer and Communications Security*, pages 513–524. ACM Press, October 2012. 27

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. One-time programs. In *CRYPTO*, pages 39–56, 2008. 34

[GKS16] Vipul Goyal, Dakshita Khurana, and Amit Sahai. Breaking the three round barrier for non-malleable commitments. In Irit Dinur, editor, *57th Annual Symposium on Foundations of Computer Science*, pages 21–30. IEEE Computer Society Press, October 2016. 8

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *STOC*, pages 25–32, 1989. 76

[GLO15] Sanjam Garg, Steve Lu, and Rafail Ostrovsky. Black-box garbled RAM. In Venkatesan Guruswami, editor, *56th Annual Symposium on Foundations of Computer Science*, pages 210–229. IEEE Computer Society Press, October 2015. 27

[GLOS15] Sanjam Garg, Steve Lu, Rafail Ostrovsky, and Alessandra Scafuro. Garbled RAM from one-way functions. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th Annual ACM Symposium on Theory of Computing*, pages 449–458. ACM Press, June 2015. 27

[GLOV12] Vipul Goyal, Chen-Kuei Lee, Rafail Ostrovsky, and Ivan Visconti. Constructing non-malleable commitments: A black-box approach. In *53rd Annual Symposium on Foundations of Computer Science*, pages 51–60. IEEE Computer Society Press, October 2012. 5, 32, 43, 105

[GMPP16] Sanjam Garg, Pratyay Mukherjee, Omkant Pandey, and Antigoni Polychroniadou. The exact round complexity of secure computation. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 448–476. Springer, Heidelberg, May 2016. 5, 7, 8, 28, 31, 32, 41, 51, 53, 55, 56

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th Annual ACM Symposium on Theory of Computing*, pages 218–229. ACM Press, May 1987. 3, 21, 22, 25, 32, 35, 38

[GO96] Oded Goldreich and Rafail Ostrovsky. Software protection and simulation on oblivious rams. *J. ACM*, 43(3):431–473, 1996. 34

[Gol87] Oded Goldreich. Towards a theory of software protection and simulation by oblivious RAMs. In Alfred Aho, editor, *19th Annual ACM Symposium on Theory of Computing*, pages 182–194. ACM Press, May 1987. 27

[Gol01] Oded Goldreich. *Foundations of Cryptography: Basic Tools*, volume 1. Cambridge University Press, Cambridge, UK, 2001. 75

[Gol03] Oded Goldreich. Draft of a chapter on cryptographic protocols,. http://www.wisdom.weizmann.ac.il/ oded/foc-vol2.html, June 2003. 42

[Gol04] Oded Goldreich. *Foundations of Cryptography: Basic Applications*, volume 2. Cambridge University Press, Cambridge, UK, 2004. 59, 117, 184, 187

[GOS12] Jens Groth, Rafail Ostrovsky, and Amit Sahai. New techniques for noninteractive zero-knowledge. *J. ACM*, 59(3):11, 2012. 11

[Goy11] Vipul Goyal. Constant round non-malleable protocols using one way functions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd Annual ACM Symposium on Theory of Computing*, pages 695–704. ACM Press, June 2011. 5, 6, 32, 43

[GP15] Sanjam Garg and Antigoni Polychroniadou. Two-round adaptively secure MPC from indistinguishability obfuscation. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 614–637. Springer, Heidelberg, March 2015. 5, 9, 17, 19, 28, 31, 33, 107, 116, 117, 126

[GPR16] Vipul Goyal, Omkant Pandey, and Silas Richelson. Textbook non-malleable commitments. In Daniel Wichs and Yishay Mansour, editors, *48th Annual ACM Symposium on Theory of Computing*, pages 1128–1141. ACM Press, June 2016. 7

[GR03] Anna Gál and Adi Rosén. Lower bounds on the amount of randomness in private computation. In *35th Annual ACM Symposium on Theory of Computing*, pages 659–666. ACM Press, June 2003. 36

[GRRV14] Vipul Goyal, Silas Richelson, Alon Rosen, and Margarita Vald. An algebraic approach to non-malleability. In *55th Annual Symposium on Foundations of Computer Science*, pages 41–50. IEEE Computer Society Press, October 2014. 106

[GS12] Sanjam Garg and Amit Sahai. Adaptively secure multi-party computation with dishonest majority. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 105–123. Springer, Heidelberg, August 2012. 9, 33

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 75–92. Springer, Heidelberg, August 2013. 27

[GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Daniel Wichs. Leveled fully homomorphic signatures from standard lattices. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th Annual ACM Symposium on Theory of Computing*, pages 469–477. ACM Press, June 2015. 11

[HHRS15] Iftach Haitner, Jonathan J. Hoch, Omer Reingold, and Gil Segev. Finding collisions in interactive protocols - tight lower bounds on the round and communication complexities of statistically hiding commitments. *SIAM J. Comput.*, 44(1):193–242, 2015. 13

[HK07] Omer Horvitz and Jonathan Katz. Universally-composable two-party computation in two rounds. In *CRYPTO*, pages 111–129, 2007. 32

[HKE13] Yan Huang, Jonathan Katz, and David Evans. Efficient secure two-party computation using symmetric cut-and-choose. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 18–35. Springer, Heidelberg, August 2013. 31

[HLP11] Shai Halevi, Yehuda Lindell, and Benny Pinkas. Secure computation on the web: Computing without simultaneous interaction. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 132–150. Springer, Heidelberg, August 2011. 9, 108

[HPV16] Carmit Hazay, Antigoni Polychroniadou, and Muthuramakrishnan Venkitasubramaniam. Composable security in the tamper-proof hardware model under minimal complexity. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B: 14th Theory of Cryptography Conference, Part I*, volume 9985 of *Lecture Notes in Computer Science*, pages 367–399. Springer, Heidelberg, October / November 2016. 5, 15, 16, 17, 28, 31, 34, 35, 65, 73, 94

[HPV17] Carmit Hazay, Antigoni Polychroniadou, and Muthuramakrishnan Venkitasubramaniam. Constant round adaptively secure protocols in the tamper-proof hardware model. In *Public-Key Cryptography - PKC 2017 - 20th IACR International Conference on Practice and Theory in Public-Key Cryptography, Amsterdam, The Netherlands, March 28-31, 2017, Proceedings, Part II*, pages 428–460, 2017. 5, 18, 28, 34, 35, 65

[HV15] Carmit Hazay and Muthuramakrishnan Venkitasubramaniam. On black-box complexity of universally composable security in the CRS model. In *ASIACRYPT*, pages 183–209, 2015. 66

[HV16] Carmit Hazay and Muthuramakrishnan Venkitasubramaniam. On the power of secure two-party computation. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 397–429. Springer, Heidelberg, August 2016. 17, 67

[IK00] Yuval Ishai and Eyal Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *41st Annual Symposium on Foundations of Computer Science*, pages 294–304. IEEE Computer Society Press, November 2000. 20, 22, 31, 36, 140

[IKHC14] Dai Ikarashi, Ryo Kikuchi, Koki Hamada, and Koji Chida. Actively private and correct mpc scheme in t<n/2 from passively secure schemes with small overhead. *IACR Cryptology ePrint Archive*, 2014:304, 2014. 23

[IKKP15] Yuval Ishai, Ranjit Kumaresan, Eyal Kushilevitz, and Anat Paskin-Cherniavsky. Secure computation with minimal interaction, revisited. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 359–378. Springer, Heidelberg, August 2015. 36

[IKL$^+$13] Yuval Ishai, Eyal Kushilevitz, Xin Li, Rafail Ostrovsky, Manoj Prabhakaran, Amit Sahai, and David Zuckerman. Robust pseudorandom generators. In Fedor V. Fomin, Rusins Freivalds, Marta Z. Kwiatkowska, and David Peleg, editors, *ICALP 2013: 40th International Colloquium on Automata, Languages and Programming, Part I*, volume 7965 of *Lecture Notes in Computer Science*, pages 576–588. Springer, Heidelberg, July 2013. 26

[IKM$^+$13] Yuval Ishai, Eyal Kushilevitz, Sigurd Meldgaard, Claudio Orlandi, and Anat Paskin-Cherniavsky. On the power of correlated randomness in secure computation. In Amit Sahai, editor, *TCC 2013: 10th Theory of Cryptography Conference*, volume 7785 of *Lecture Notes in Computer Science*, pages 600–620. Springer, Heidelberg, March 2013. 20, 140

[IKO$^+$11] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, Manoj Prabhakaran, and Amit Sahai. Efficient non-interactive secure computation. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 406–425. Springer, Heidelberg, May 2011. 27, 104

[IKOS07] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In David S. Johnson and Uriel Feige, editors, *39th Annual ACM Symposium on Theory of Computing*, pages 21–30. ACM Press, June 2007. 23

[IKOS09] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge proofs from secure multiparty computation. *SIAM J. Comput.*, 39(3):1121–1152, 2009. 67

[IKP10] Yuval Ishai, Eyal Kushilevitz, and Anat Paskin. Secure multiparty computation with minimal interaction. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 577–594. Springer, Heidelberg, August 2010. 36

[IPS08] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 572–591. Springer, Heidelberg, August 2008. 10, 11, 14, 18, 23, 31, 33, 66, 104, 128

[IPS09] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Secure arithmetic computation with no honest majority. In Omer Reingold, editor, *TCC 2009: 6th Theory of Cryptography Conference*, volume 5444 of *Lecture Notes in Computer Science*, pages 294–314. Springer, Heidelberg, March 2009. 21, 23, 35, 38

[JJ00] Markus Jakobsson and Ari Juels. Mix and match: Secure function evaluation via ciphertexts. In Tatsuaki Okamoto, editor, *Advances in Cryptology – ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 162–177. Springer, Heidelberg, December 2000. 33

[JS07] Stanislaw Jarecki and Vitaly Shmatikov. Efficient two-party secure computation on committed inputs. In Moni Naor, editor, *Advances in Cryptology – EUROCRYPT 2007*, volume 4515 of *Lecture Notes in Computer Science*, pages 97–114. Springer, Heidelberg, May 2007. 32

[Kat07] Jonathan Katz. Universally composable multi-party computation using tamper-proof hardware. In *EUROCRYPT*, pages 115–128, 2007. 12, 13, 34, 65, 68, 69, 70, 72, 73

[Kil88] Joe Kilian. Founding cryptography on oblivious transfer. In *STOC*, pages 20–31, 1988. 14

[KLP07] Yael Tauman Kalai, Yehuda Lindell, and Manoj Prabhakaran. Concurrent composition of secure protocols in the timing model. *J. Cryptology*, 20(4):431–492, 2007. 12

[KM97] Eyal Kushilevitz and Yishay Mansour. Randomness in private computations. *SIAM J. Discrete Math.*, 10(4):647–661, 1997. 36

[KO04] Jonathan Katz and Rafail Ostrovsky. Round-optimal secure two-party computation. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 335–354. Springer, Heidelberg, August 2004. 5, 6, 31, 41, 44, 46, 47, 50, 51, 53, 54, 63

[KOS03] Jonathan Katz, Rafail Ostrovsky, and Adam Smith. Round efficiency of multi-party computation with a dishonest majority. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 578–595. Springer, Heidelberg, May 2003. 6, 31, 32, 50

[KR94] Eyal Kushilevitz and Adi Rosén. A randomnesss-rounds tradeoff in private computation. In Yvo Desmedt, editor, *Advances in Cryptology – CRYPTO'94*, volume 839 of *Lecture Notes in Computer Science*, pages 397–410. Springer, Heidelberg, August 1994. 36

[KTZ13] Jonathan Katz, Aishwarya Thiruvengadam, and Hong-Sheng Zhou. Feasibility and infeasibility of adaptively secure fully homomorphic encryption. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013: 16th International Conference on Theory and Practice of Public Key Cryptography*, volume 7778 of *Lecture Notes in Computer Science*, pages 14–31. Springer, Heidelberg, February / March 2013. 11, 33

[Kus92] Eyal Kushilevitz. Privacy and communication complexity. *SIAM J. Discrete Math.*, 5(2):273–284, 1992. 36

[Lin01] Yehuda Lindell. Parallel coin-tossing and constant-round secure two-party computation. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 171–189. Springer, Heidelberg, August 2001. 31, 60

[Lin03a] Yehuda Lindell. Bounded-concurrent secure two-party computation without setup assumptions. In *35th Annual ACM Symposium on Theory of Computing*, pages 683–692. ACM Press, June 2003. 9, 12, 17

[Lin03b] Yehuda Lindell. General composition and universal composability in secure multi-party computation. In *FOCS*, pages 394–403, 2003. 66, 68

[Lin13] Yehuda Lindell. Fast cut-and-choose based protocols for malicious and covert adversaries. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 1–17. Springer, Heidelberg, August 2013. 31

[LO13a] Steve Lu and Rafail Ostrovsky. Distributed oblivious RAM for secure two-party computation. In Amit Sahai, editor, *TCC 2013: 10th Theory of Cryptography Conference*, volume 7785 of *Lecture Notes in Computer Science*, pages 377–396. Springer, Heidelberg, March 2013. 27

[LO13b] Steve Lu and Rafail Ostrovsky. How to garble RAM programs. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 719–734. Springer, Heidelberg, May 2013. 27

[LP07] Yehuda Lindell and Benny Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. In Moni Naor, editor, *Advances in Cryptology – EUROCRYPT 2007*, volume 4515 of *Lecture Notes in Computer Science*, pages 52–78. Springer, Heidelberg, May 2007. 22, 31

[LP09a] Huijia Lin and Rafael Pass. Non-malleability amplification. In Michael Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 189–198. ACM Press, May / June 2009. 45, 46

[LP09b] Yehuda Lindell and Benny Pinkas. A proof of security of Yao's protocol for two-party computation. *Journal of Cryptology*, 22(2):161–188, April 2009. 18, 47, 117

[LP11a] Huijia Lin and Rafael Pass. Constant-round non-malleable commitments from any one-way function. In Lance Fortnow and Salil P. Vadhan, editors, *43rd Annual ACM Symposium on Theory of Computing*, pages 705–714. ACM Press, June 2011. 5, 32

[LP11b] Yehuda Lindell and Benny Pinkas. Secure two-party computation via cut-and-choose oblivious transfer. In Yuval Ishai, editor, *TCC 2011: 8th Theory of Cryptography Conference*, volume 6597 of *Lecture Notes in Computer Science*, pages 329–346. Springer, Heidelberg, March 2011. 22, 31, 43

[LPTV10] Huijia Lin, Rafael Pass, Wei-Lung Dustin Tseng, and Muthuramakrishnan Venkitasubramaniam. Concurrent non-malleable zero knowledge proofs. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 429–446. Springer, Heidelberg, August 2010. 43

[LPV09] Huijia Lin, Rafael Pass, and Muthuramakrishnan Venkitasubramaniam. A unified framework for concurrent security: universal composability from stand-alone non-malleability. In Michael Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 179–188. ACM Press, May / June 2009. 12, 17, 32, 34, 35

[LPV12] Huijia Lin, Rafael Pass, and Muthuramakrishnan Venkitasubramaniam. A unified framework for UC from only OT. In *ASIACRYPT*, pages 699–717, 2012. 14

[LR14] Yehuda Lindell and Ben Riva. Cut-and-choose Yao-based secure computation in the online/offline and batch settings. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part II*, volume 8617 of *Lecture Notes in Computer Science*, pages 476–494. Springer, Heidelberg, August 2014. 33

[LR15] Yehuda Lindell and Ben Riva. Blazing fast 2PC in the offline/online setting with security for malicious adversaries. In Indrajit Ray, Ninghui Li, and Christopher Kruegel:, editors, *ACM CCS 15: 22nd Conference on Computer and Communications Security*, pages 579–590. ACM Press, October 2015. 33

[LS90] Dror Lapidot and Adi Shamir. Publicly verifiable non-interactive zero-knowledge proofs. In *CRYPTO*, pages 353–365, 1990. 67

[LSS16] Yehuda Lindell, Nigel P. Smart, and Eduardo Soria-Vazquez. More efficient constant-round multi-party computation from BMR and SHE. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B: 14th Theory of Cryptography Conference, Part I*, volume 9985 of *Lecture Notes in Computer Science*, pages 554–581. Springer, Heidelberg, October / November 2016. 11

[LTV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In Howard J. Karloff and Toniann Pitassi, editors, *44th Annual ACM Symposium on Theory of Computing*, pages 1219–1234. ACM Press, May 2012. 32

[MMN16] Jeremias Mechler, Jörn Müller-Quade, and Tobias Nilges. Universally composable (non-interactive) two-party computation from untrusted reusable hardware tokens. *IACR Cryptology ePrint Archive*, 2016:615, 2016. 34

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 700–718. Springer, Heidelberg, April 2012. 129, 130

[MS08] Tal Moran and Gil Segev. David and goliath commitments: UC computation for asymmetric parties using tamper-proof hardware. In *EUROCRYPT*, pages 527–544, 2008. 13, 15, 35

[MSS11] Steven Myers, Mona Sergi, and Abhi Shelat. Threshold fully homomorphic encryption and secure computation. Cryptology ePrint Archive, Report 2011/454, 2011. http://eprint.iacr.org/2011/454. 32

[MW16] Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key FHE. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 735–763. Springer, Heidelberg, May 2016. 8, 32, 33, 44, 126

[Nao91] Moni Naor. Bit commitment using pseudorandomness. *J. Cryptology*, 4(2):151–158, 1991. 75, 78, 91, 104

[Nie02] Jesper Buus Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 111–126. Springer, Heidelberg, August 2002. 127

[Nil15] Tobias Nilges. *The Cryptographic Strength of Tamper-Proof Hardware*. PhD thesis, Karlsruhe Institute of Technology, 2015. 34

[NN01] Moni Naor and Kobbi Nissim. Communication preserving protocols for secure function evaluation. In *33rd Annual ACM Symposium on Theory of Computing*, pages 590–599. ACM Press, July 2001. 20

[NNOB12] Jesper Buus Nielsen, Peter Sebastian Nordholt, Claudio Orlandi, and Sai Sheshank Burra. A new approach to practical active-secure two-party computation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 681–700. Springer, Heidelberg, August 2012. 21, 33, 141

[NO09] Jesper Buus Nielsen and Claudio Orlandi. LEGO for two-party secure computation. In Omer Reingold, editor, *TCC 2009: 6th Theory of Cryptography Conference*, volume 5444 of *Lecture Notes in Computer Science*, pages 368–386. Springer, Heidelberg, March 2009. 23

[NO16] Jesper Buus Nielsen and Claudio Orlandi. Cross and clean: Amortized garbled circuits with constant overhead. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B: 14th Theory of Cryptography Conference, Part I*, volume 9985 of *Lecture Notes in Computer Science*, pages 582–603. Springer, Heidelberg, October / November 2016. 33

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd Annual ACM Symposium on Theory of Computing*, pages 427–437. ACM Press, May 1990. 112

[OPW11] Adam O'Neill, Chris Peikert, and Brent Waters. Bi-deniable public-key encryption. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 525–542. Springer, Heidelberg, August 2011. 130

[ORS15] Rafail Ostrovsky, Silas Richelson, and Alessandra Scafuro. Round-optimal black-box two-party computation. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 339–358. Springer, Heidelberg, August 2015. 31, 66, 92, 93, 96

[OS97] Rafail Ostrovsky and Victor Shoup. Private information storage (extended abstract). In *29th Annual ACM Symposium on Theory of Computing*, pages 294–303. ACM Press, May 1997. 27

[Ost90] Rafail Ostrovsky. Efficient computation on oblivious RAMs. In *22nd Annual ACM Symposium on Theory of Computing*, pages 514–523. ACM Press, May 1990. 27

[Pas03] Rafael Pass. Simulation in quasi-polynomial time, and its application to protocol composition. In *EUROCRYPT*, pages 160–176, 2003. 17, 66, 68

[Pas04] Rafael Pass. Bounded-concurrent secure multi-party computation with a dishonest majority. In László Babai, editor, *36th Annual ACM Symposium on Theory of Computing*, pages 232–241. ACM Press, June 2004. 5, 32

[PPV08] Omkant Pandey, Rafael Pass, and Vinod Vaikuntanathan. Adaptive one-way functions and applications. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 57–74. Springer, Heidelberg, August 2008. 5, 7, 8, 32, 64

[PS04] Manoj Prabhakaran and Amit Sahai. New notions of security: achieving universal composability without trusted setup. In *STOC*, pages 242–251, 2004. 17, 68

[PS16] Chris Peikert and Sina Shiehian. Multi-key FHE from LWE, revisited. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B: 14th Theory of Cryptography Conference, Part II*, volume 9986 of *Lecture Notes in Computer Science*, pages 217–238. Springer, Heidelberg, October / November 2016. 32

[PW09] Rafael Pass and Hoeteck Wee. Black-box constructions of two-party protocols from one-way functions. In *TCC*, pages 403–418, 2009. 66, 77, 79, 105

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 84–93. ACM Press, May 2005. 8

[Sha79] Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979. 76

[SV15] Abhi Shelat and Muthuramakrishnan Venkitasubramaniam. Secure computation from millionaire. In *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part I*, pages 736–757, 2015. 22

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *46th Annual ACM Symposium on Theory of Computing*, pages 475–484. ACM Press, May / June 2014. 109, 110, 112, 113, 120, 121

[Ven14] Muthuramakrishnan Venkitasubramaniam. On adaptively secure protocols. In Michel Abdalla and Roberto De Prisco, editors, *SCN 14: 9th International Conference on Security in Communication Networks*, volume 8642 of *Lecture Notes in Computer Science*, pages 455–475. Springer, Heidelberg, September 2014. 17, 33, 35

[Wee10] Hoeteck Wee. Black-box, round-efficient secure computation via non-malleability amplification. In *51st Annual Symposium on Foundations of Computer Science*, pages 531–540. IEEE Computer Society Press, October 2010. 5, 6, 32

[WHC$^+$14] Xiao Shaun Wang, Yan Huang, T.-H. Hubert Chan, Abhi Shelat, and Elaine Shi. SCORAM: Oblivious RAM for secure computation. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 14: 21st Conference on Computer and Communications Security*, pages 191–202. ACM Press, November 2014. 27

[WW10] Severin Winkler and Jürg Wullschleger. On the efficiency of classical and quantum oblivious transfer reductions. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 707–723. Springer, Heidelberg, August 2010. 141, 152, 153, 154, 157

[Yao82] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science*, pages 160–164. IEEE Computer Society Press, November 1982. 3, 5, 9, 31, 47

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science*, pages 162–167. IEEE Computer Society Press, October 1986. 5, 18, 31

Secure Computation Definitions

For completeness, we recall the definition of secure computation based on [Gol04, Chapter 7] here. We only recall the two party case as it is most relevant to our proofs. The description naturally extends to multi-party case as well (details can be found in [Gol04]).

Two-party computation. A two-party protocol problem is cast by specifying a random process that maps pairs of inputs to pairs of outputs (one for each party). We refer to such a process as a functionality and denote it $F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^* \times \{0,1\}^*$ where $F = (F_1, F_2)$. That is, for every pair of inputs (x, y), the output-pair is a random variable $(F_1(x, y), F_2(x, y))$ ranging over pairs of strings. The first party (with input x) wishes to obtain $F_1(x, y)$ and the second party (with input y) wishes to obtain $F_2(x, y)$.

Adversarial behavior. Loosely speaking, the aim of a secure two-party protocol is to protect an honest party against dishonest behavior by the other party. In this paper, we consider malicious adversaries who may arbitrarily deviate from the specified protocol. When considering malicious adversaries, there are certain undesirable actions that cannot be prevented. Specifically, a party may refuse to participate in the protocol, may substitute its local input (and use instead a different input) and may abort the protocol prematurely. One ramification of the adversary's ability to abort, is that it is impossible to achieve *fairness*. That is, the adversary may obtain its output while the honest party does not. In this work we consider a static corruption model, where one of the parties is adversarial and the other is honest, and this is fixed before the execution begins.

Communication channel. In our results we consider a secure simultaneous message exchange channel in which all parties can simultaneously send messages over the channel at the same communication round. Moreover, we assume an asynchronous network[6] where the communication is open (i.e. all the communication between the parties is seen by the adversary) and delivery of messages is not guaranteed. For simplicity, we assume that the delivered messages are authenticated. This can be achieved using standard methods.

Security of protocols (informal). The security of a protocol is analyzed by comparing what an adversary can do in the protocol to what it can do in an ideal scenario that is secure by definition. This is formalized by considering an ideal computation involving an incorruptible trusted third party to whom the parties send their inputs. The trusted party computes the functionality on the inputs and returns to each party its respective output. Loosely speaking, a protocol is secure if any adversary interacting in the real protocol (where no trusted third party exists) can do no more harm than if it was involved in the above-described ideal computation.

Execution in the ideal model. As we have mentioned, some malicious behavior cannot be prevented (for example, early aborting). This behavior is therefore incorporated into the ideal model. An ideal execution proceeds as follows:

Inputs: Each party obtains an input, denoted w ($w = x$ for P_1, and $w = y$ for P_2).

Send inputs to trusted party: An honest party always sends w to the trusted party. A malicious party may, depending on w, either abort or send some $w' \in \{0,1\}^{|w|}$ to the trusted party.

[6]The fact that the network is asynchronous means that the messages are not necessarily delivered in the order which they are sent.

Trusted party answers first party: In case it has obtained an input pair (x, y), the trusted party first replies to the first party with $F_1(x, y)$. Otherwise (i.e., in case it receives only one valid input), the trusted party replies to both parties with a special symbol $\perp$.

Trusted party answers second party: In case the first party is malicious it may, depending on its input and the trusted party's answer, decide to stop the trusted party by sending it $\perp$ after receiving its output. In this case the trusted party sends $\perp$ to the second party. Otherwise (i.e., if not stopped), the trusted party sends $F_2(x, y)$ to the second party.

Outputs: An honest party always outputs the message it has obtained from the trusted party. A malicious party may output an arbitrary (probabilistic polynomial-time computable) function of its initial input and the message obtained from the trusted party.

Let $F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^* \times \{0,1\}^*$ be a functionality where $F = (F_1, F_2)$ and let $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ be a pair of non-uniform probabilistic expected polynomial-time machines (representing parties in the ideal model). Such a pair is *admissible* if for at least one $i \in \{1, 2\}$ we have that $\mathcal{S}_i$ is honest (i.e., follows the honest party instructions in the above-described ideal execution). Then, the *joint execution of F under $\mathcal{S}$ in the ideal model* (on input pair (x, y) and security parameter κ), denoted $\mathrm{IDEAL}_{F,\mathcal{S}}(\kappa, x, y)$ is defined as the output pair of $\mathcal{S}_1$ and $\mathcal{S}_2$ from the above ideal execution.

Execution in the real model. We next consider the real model in which a real (two-party) protocol is executed (and there exists no trusted third party). In this case, a malicious party may follow an arbitrary feasible strategy; that is, any strategy implementable by non-uniform probabilistic polynomial-time machines. In particular, the malicious party may abort the execution at any point in time (and when this happens prematurely, the other party is left with no output). Let F be as above and let Π be a two-party protocol for computing F. Furthermore, let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be a pair of non-uniform probabilistic polynomial-time machines (representing parties in the real model). Such a pair is *admissible* if for at least one $i \in \{1, 2\}$ we have that $\mathcal{A}_i$ is honest (i.e., follows the strategy specified by Π). Then, the *joint execution of* Π *under $\mathcal{A}$ in the real model*, denoted $\mathrm{REAL}_{\Pi,\mathcal{A}}(\kappa, x, y)$, is defined as the output pair of $\mathcal{A}_1$ and $\mathcal{A}_2$ resulting from the protocol interaction.

Security as emulation of a real execution in the ideal model. Having defined the ideal and real models, we can now define security of protocols. Loosely speaking, the definition asserts that a secure two-party protocol (in the real model) emulates the ideal model (in which a trusted party exists). This is formulated by saying that admissible pairs in the ideal model are able to simulate admissible pairs in an execution of a secure real-model protocol.

Definition .9 (secure two-party computation)**.** Let F and Π be as above. Protocol Π is said to securely compute F (in the malicious model) if for every pair of admissible non-uniform probabilistic polynomial-time machines $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ for the real model, there exists a pair of admissible non-uniform probabilistic expected polynomial-time machines $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ for the ideal model, such that:

$$\{\text{IDEAL}_{F,\mathcal{S}}(\kappa, x, y)\}_{\kappa\in\mathbb{N},x,y \text{ s.t. } |x|=|y|} \overset{c}{\approx} \{\text{REAL}_{\Pi,\mathcal{A}}(\kappa, x, y)\}_{\kappa\in\mathbb{N},x,y \text{ s.t. } |x|=|y|}$$

We note that the above definition assumes that the parties know the input lengths (this can be seen from the requirement that $|x| = |y|$). Some restriction on the input lengths is unavoidable, see [Gol04, Section 7.1] for discussion. We also note that we allow the ideal adversary/simulator to run in expected (rather than strict) polynomial-time. This is essential for constant-round protocols.

Proof Systems

We provide a detailed description of the proof systems used in this work.

Protocol Π_{WIPOK}

This is essentially the Feige-Lapidot-Shamir protocol, slightly reworded in [KO04], mostly for notational convenience. We recall this protocol here. We denote the messages of this protocol by $(\mathsf{p}_1, \mathsf{p}_2, \mathsf{p}_3)$.

We will be working with the NP-complete language *HC* of graph Hamiltonicity, and thus assume statements to be proven take the form of graphs, while witnesses correspond to Hamilton cycles. If thm is a graph, we abuse notation and also let thm denote the statement "$\mathsf{thm} \in HC$". We show how the proof system can be used to prove the following statement: $\mathsf{thm} \wedge \mathsf{thm}'$, where thm will be included as part of the first message, while thm' is only decided in the last round. The proof system Π_{WIPOK} runs κ parallel executions of the following 3-round protocol:

1. The prover commits to two adjacency matrices for two randomly-chosen cycle graphs G, G'. The commitment is done bit-by-bit using a perfectly-binding commitment scheme.

2. The verifier responds with a single bit b, chosen at random.

3. If $b = 0$, the prover opens all commitments. If $b = 1$, the prover sends two permutations mapping the cycle in thm (resp., thm') to G (resp., G'). For each non-edge in thm (resp., thm'), the prover opens the commitment at the corresponding position in G (resp., G').

   The verifier checks that all commitments were opened correctly. If $b = 0$, the verifier additionally checks whether both decommitted graphs are indeed cycle graphs. If $b = 1$, the verifier checks whether each non-edge in thm (resp., thm') corresponds to a non-edge in G (resp., G').

Note that the prover does not need to know either thm or thm' (or the corresponding witnesses) until the beginning of the third round. In the above proof system, we assume that thm is fixed as part of the first-round message enabling us to claim stronger properties about the proof system. In particular, Π_{WIPOK} proof system is complete and sound. More specifically, the probability that an all-powerful prover can cause a verifier to accept when either thm or thm' are not true is at most $2^{-\kappa}$. We stress that this holds even if the prover can adaptively choose thm' after viewing the second-round message of the verifier. Moreover, Π_{WIPOK} is witness indistinguishable and it is a proof of knowledge for thm. (More formally, we can achieve a notion similar to that of "witness-extended emulation" [Lin01] for thm.) Note also that the first round of the above proof system (as well as the internal state of the prover immediately following this round) is independent of thm or the associated witness.

Protocol Π_{FS}

As noted in Section 6.1.3, this is essentially the four round zero-knowledge protocol of Feige-Shamir, except that we use *non-malleable commitments* in the first three round of the protocol. Following the discussion in Section 6.1.3, we let nmcom be a non-malleable commitment scheme, and make the simplifying assumption that nmcom has just three rounds and the first round is committing. Again, these are purely for notational convenience and can easily be removed (as discussed earlier).

We now simply list all the steps of this protocol following [KO04], but using nmcom. The messages of this protocol are denoted by $(\mathsf{fs}_1, \mathsf{fs}_2, \mathsf{fs}_3, \mathsf{fs}_4)$. It allows the prover to prove $\mathsf{thm} \wedge \mathsf{thm}'$ where thm is sent as part of the second round yet thm' is only sent as part of the last round. (Intuitively, statements $\mathsf{thm}, \mathsf{thm}'$ will correspond to statements $\mathsf{st}_1, \mathsf{st}_2$ of Π_{WIPOK} described above.)

The proof system Π_{FS} proceeds as follows:

1. The first round is as in the original Feige-Shamir protocol but augmented with an nmcom scheme. Explicitly, the verifier V selects randomly and independently two values σ_1 and σ_2 and computes the first message of two independent executions of nmcom for σ_1 and σ_2, with randomness ρ_1, ρ_2 respectively. Let $\mathsf{nm}_1^{\sigma_1}$ and $\mathsf{nm}_2^{\sigma_2}$ be these messages, which V sends to P.

   Moreover, V sends the first message p_1 of a WIPOK proof system.

2. The prover P chooses a random challenge $R \in \{0,1\}^{2\kappa}$ and computes $\mathsf{C_R} = \mathsf{Eqcom}(R; \zeta)$. Let eqthm denote the statement that Eqcom was formed correctly.

   Let $\widetilde{\mathsf{thm}}$ denote the statement: $(\mathsf{thm} \wedge \mathsf{eqthm}) \vee (\mathsf{nm}_1^{\sigma_1} = \mathsf{nmcom}_1(\sigma_1; \rho_1)) \vee (\mathsf{nm}_1^{\sigma_2} = \mathsf{nmcom}_1(\sigma_2; \rho_2))$ (this statement is reduced to a single graph $\widetilde{\mathsf{thm}}$). Then, P sends $\mathsf{C_R}$ and also the first message $\tilde{\mathsf{p}}_1$ of a separate WIPOK proof system and message p_2 of V's proof.

3. V sends the last message p_3 of his WIPOK proof system and completes the proof for the knowledge of the values in nmcom (which is also completed along with the first and second rounds [7]). V additionally sends a random $R' \in \{0,1\}^{2\kappa}$ and message $\tilde{\mathsf{p}}_2$ of P's proof

4. P decommits to R. Let prg be the statement that $r = R + R'$ is pseudorandom (i.e., $\exists s$ s.t. $\mathsf{PRG}(s) = r$, where PRG is a pseudorandom function). Let $\widetilde{\mathsf{thm}}'$ be the statement $\mathsf{thm}' \vee \mathsf{prg}$ (reduced to a single graph $\widetilde{\mathsf{thm}}'$). The prover send the last message $\tilde{\mathsf{p}}_3$ of the Π_{WIPOK} proof system and completes the proof for the statement $\widetilde{\mathsf{thm}} \wedge \widetilde{\mathsf{thm}}'$.

[7]If $k > 3$ then V completes its WIPOK after the completion of nmcom.

V checks the decommitment of R, and verifies the proof.

As claimed in [KO04] Π_{FS} proof system satisfies the following properties. It is complete and sound (for a poly-time prover) for thm and thm'. Rounds $2-4$ constitute a proof of knowledge for $\widetilde{\mathsf{thm}}$. If a poly-time prover can cause a verifier to accept with "high" probability, then a witness for $\mathsf{thm} \wedge \mathsf{eqthm}$ can be extracted with essentially the same probability. If eqthm is true, then with all but negligible probability prg will not be true. Soundness of the proof of knowledge sub-protocol then implies that $\widetilde{\mathsf{thm}}'$ is true. But this means that thm' is true. Π_{FS} is also zero-knowledge (in addition, to simulating for $\widetilde{\mathsf{thm}}$, the simulator also uses the equivocal commitment property to decommit to an R such that prg is true.). Furthermore, Π_{FS} is an argument of knowledge for thm.

Note that although we are using nmcom we are not making any claim here that uses non-malleability. All claims above simply rely on the hiding of nmcom. The non-malleability is used by the two-party protocol which uses Π_{FS}.

Also note that in order to handle a general nmcom of k rounds, simply execute the first $k-3$ rounds before the protocol above begins. The statements are then modified to work with the transcript, rather than the first message of the protocol.

UC Security

In this section we briefly review UC security. For full details see [Can01].

The basic model of execution. Following [Gol01], a protocol is represented as an interactive Turing machine (ITM), which represents the program to be run within each participant. Specifically, an ITM has three tapes that can be written to by other ITMs: the input and subroutine output tapes model the inputs from and the outputs to other programs running within the same "entity" (say, the same physical computer), and the incoming communication tapes and outgoing communication tapes model messages received from and to be sent to the network. It also has an identity tape that cannot be written to by the ITM itself. The identity tape contains the program of the ITM (in some standard encoding) plus additional identifying information specified below. Adversarial entities are also modeled as ITMs.

We distinguish between ITMs (which represent static objects, or programs) and *instances of ITMs*, or ITIs, that represent interacting processes in a running system. Specifically, an ITI is an ITM along with an identifer that distinguishes it from other ITIs in the same system. The identifier consists of two parts: A session-identifier (SID) which identifies which protocol instance the ITM belongs to, and a party identifier (PID) that distinguishes among the parties in a protocol instance. Typically the PID is also used to associate ITIs with "parties", or clusters, that represent some administrative domains or physical computers.

The model of computation consists of a number of ITIs that can write on each other's tapes in certain ways (specified in the model). The pair (SID,PID) is a unique identifier of the ITI in the system.

With one exception (discussed within) we assume that all ITMs are probabilistic polynomial time (PPT). An ITM is PPT if there exists a constant $c > 0$ such that, at any point during its run, the overall number of steps taken by M is at most n^c, where n is the overall number of bits written on the *input tape* of M in this run. (In fact, in order to guarantee that the overall protocol execution process is bounded by a polynomial, we define n as the total number of bits written to the input tape of M, *minus the overall number of bits written by M to input tapes of other ITMs.*; see [Can01].)

Security of protocols. Protocols that securely carry out a given task (or, protocol problem) are defined in three steps, as follows. First, the process of executing a protocol in an adversarial environment is formalized. Next, an "ideal process" for carrying out the task at hand is formalized. In the ideal process the parties do not communicate with each other. Instead they have access to an "ideal functionality," which is essentially an incorruptible "trusted party" that is programmed to capture the desired functionality of the task at hand. A protocol is said to securely realize an ideal functionality if the process of running the protocol amounts to "emulating" the ideal process for that ideal functionality. Below we overview the model of protocol execution (called the *real-life model*), the ideal process, and the notion of protocol emulation.

The model for protocol execution. The model of computation consists of the parties running an instance of a protocol Π, an adversary $\mathcal{A}$ that controls the communication among the parties, and an *environment* $\mathcal{Z}$ that controls the inputs to the parties and sees their outputs. We assume that all parties have a security parameter $n \in \mathbb{N}$. (We remark that this is done merely for convenience and is not essential for the model to make sense). The execution consists of a sequence of *activations*, where in each activation a single participant (either $\mathcal{Z}$, $\mathcal{A}$, or some other ITM) is activated, and may write on a tape of at most *one* other participant, subject to the rules below. Once the activation of a participant is complete (i.e., once it enters a special waiting state), the participant whose tape was written on is activated next. (If no such party exists then the environment is activated next.)

The environment is given an external input z and is the first to be activated. In its first activation, the environment invokes the adversary $\mathcal{A}$, providing it with some arbitrary input. In the context of UC security, the environment can from now on invoke (namely, provide input to) only ITMs that consist of a single instance of protocol Π. That is, all the ITMs invoked by the environment must have the same SID and the code of Π.

Once the adversary is activated, it may read its own tapes and the outgoing communication tapes of all parties. It may either deliver a message to some party by writing this message on the party's incoming communication tape or report information to $\mathcal{Z}$ by writing this information on the subroutine output tape of $\mathcal{Z}$. For simplicity of exposition, in the rest of this paper we assume authenticated communication; that is, the adversary may deliver only messages that were actually sent.

Once a protocol party (i.e., an ITI running Π) is activated, either due to an input given by the environment or due to a message delivered by the adversary, it follows its code and possibly writes a local output on the subroutine output tape of the environment, or an outgoing message on the adversary's incoming communication tape. Finally our adversary can decide to corrupt any honest party. In this case the input and the random coins used by this party are revealed to the adversary.

The protocol execution ends when the environment halts. The output of the protocol execution is the output of the environment. Without loss of generality we assume that this output consists of only a single bit.

Let $\text{REAL}_{\pi,\mathcal{A},\mathcal{Z}}(n, z, r)$ denote the output of the environment $\mathcal{Z}$ when interacting with parties running protocol Π on security parameter n, input z and random input $r = r_{\mathcal{Z}}, r_{\mathcal{A}}, r_1, r_2, \ldots$ as described above (z and $r_{\mathcal{Z}}$ for $\mathcal{Z}$; $r_{\mathcal{A}}$ for $\mathcal{A}$, r_i for party P_i). Let $\text{REAL}_{\pi,\mathcal{A},\mathcal{Z}}(n, z)$ random variable describing $\text{REAL}_{\pi,\mathcal{A},\mathcal{Z}}(n, z, r)$ where r is uniformly chosen. Let $\text{REAL}_{\pi,\mathcal{A},\mathcal{Z}}$ denote the ensemble $\{\text{REAL}_{\pi,\mathcal{A},\mathcal{Z}}(n, z)\}_{n\in\mathbb{N},z\in\{0,1\}^*}$.

Ideal functionalities and ideal protocols. Security of protocols is defined via comparing the protocol execution to an *ideal protocol* for carrying out the task at hand. A key ingredient in the ideal protocol is the *ideal functionality* that captures the desired functionality, or the specification, of that task. The ideal functionality is modeled as another ITM (representing a "trusted party") that interacts with the parties and the adversary. More specifically, in the ideal protocol for functionality $\mathcal{F}$ all parties simply hand their inputs to an ITI running $\mathcal{F}$. (We will simply call this ITI $\mathcal{F}$. The SID of $\mathcal{F}$ is the same as the SID of the ITIs running the ideal protocol. (the PID of $\mathcal{F}$ is null.)) In addition, $\mathcal{F}$ can interact with the adversary according to its code. Whenever $\mathcal{F}$ outputs a value to a party, the party immediately copies this value to its own output tape. We call the parties in the ideal protocol dummy parties. Let $\Pi(\mathcal{F})$ denote the ideal protocol for functionality $\mathcal{F}$.

Securely realizing an ideal functionality. We say that a protocol Π *emulates* protocol ϕ if for any adversary $\mathcal{A}$ there exists an adversary $\mathcal{S}$ such that no environment $\mathcal{Z}$, on any input, can tell with non-negligible probability whether it is interacting with $\mathcal{A}$ and parties running Π, or it is interacting with S and parties running ϕ. This means that, from the point of view of the environment, running protocol Π is 'just as good' as interacting with ϕ. We say that Π *securely realizes* an ideal functionality $\mathcal{F}$ if it emulates the ideal protocol $\Pi(\mathcal{F})$. More precise definitions follow. A distribution ensemble is called *binary* if it consists of distributions over $\{0, 1\}$.

Definition .10. Let Π and ϕ be protocols. We say that Π UC-emulates ϕ if for any adversary $\mathcal{A}$ there exists an adversary $\mathcal{S}$ such that for any environment $\mathcal{Z}$ that obeys the rules of interaction for UC security we have $\text{IDEAL}_{\phi,\mathcal{S},\mathcal{Z}} \approx \text{REAL}_{\pi,\mathcal{A},\mathcal{Z}}$.

Definition .11. Let $\mathcal{F}$ be an ideal functionality and let Π be a protocol. We say that Π UC-realizes $\mathcal{F}$ if Π UC-emulates the ideal process $\Pi(\mathcal{F})$.

The Common Reference String Model. In the common reference string (CRS) model [CF01, CLOS02], all parties in the system obtain from a trusted party a reference string, which is sampled according to a pre-specified distribution D. The reference string is referred to as the *CRS*. In the UC framework, this is modeled by an ideal functionality $\mathcal{F}_{CRS}^{D}$ that samples a string ρ from a pre-specified distribution D and sets ρ as the CRS. $\mathcal{F}_{CRS}^{D}$ is described in Figure 1.

General Functionality. We consider the general-UC functionality $\mathcal{F}$, which securely evaluates any polynomial-time (possibly randomize) function $f : (\{0, 1\}^{\ell_{in}})^n \to (\{0, 1\}^{\ell_{out}})^n$. The functionality $\mathcal{F}_f$ is parameterized with a function f and is described in Figure 2.

Our protocol in Figure 6.1.4 (also Theorem 6.1) is for UC-securely realizing general functionality $\mathcal{F}_f$ when the function f is restricted to be any deterministic poly-time function with n

Functionality $\mathcal{F}_{\mathbf{CRS}}^{\mathbf{D}}$

1. Upon activation with session id sid proceed as follows. Sample $\rho = D(r)$, where r denotes uniform random coins, and send $(\texttt{crs}, sid, \rho)$ to the adversary.

2. On receiving $(\texttt{crs}, sid)$ from some party send $(\texttt{crs}, sid, \rho)$ to that party.

Figure .1: The Common Reference String Functionality.

inputs and single output. This functionality has been formally defined in Figure 3. As explained in Section 6.1.4.1 the same protocol can be used to obtain a protocol that UC-securely realizes the general functionality $\mathcal{F}_f$ for any function f.

Functionality $\mathcal{F}_f$

$\mathcal{F}_f$ parameterized by an (possibly randomized) n-ary function f, running with parties $\mathcal{P} = \{P_1, \ldots P_n\}$ (of which some may be corrupted) and an adversary $\mathcal{S}$, proceeds as follows:

1. Each party P_i (and $\mathcal{S}$ on behalf of P_i if P_i is corrupted) sends $(\mathsf{input}, \mathsf{sid}, \mathcal{P}, P_i, x_i)$ to the functionality.

2. Upon receiving the inputs from all parties, evaluate $(y_1, \ldots y_n) \leftarrow f(x_1, \ldots, x_n)$. For every P_i that is corrupted send adversary $\mathcal{S}$ the message $(\mathsf{output}, \mathsf{sid}, \mathcal{P}, P_i, y_i)$.

3. On receiving $(\mathsf{generateOutput}, \mathsf{sid}, \mathcal{P}, P_i)$ from $\mathcal{S}$ the ideal functionality outputs $(\mathsf{output}, \mathsf{sid}, \mathcal{P}, P_i, y_i)$ to P_i. (And ignores the message if inputs from all parties in $\mathcal{P}$ have not been received.)

4. If all the parties in $\mathcal{P}$ are corrupted then the ideal functionality reveals the internals coins used in the computation of f.

Figure .2: General Functionality.

Functionality $\mathcal{F}_f$

$\mathcal{F}_f$ parameterized by an n-ary deterministic single output function f, running with parties $\mathcal{P} = \{P_1, \dots P_n\}$ (of which some may be corrupted) and an adversary $\mathcal{S}$, proceeds as follows:

1. Each party P_i (and $\mathcal{S}$ on behalf of P_i if P_i is corrupted) sends $(\mathsf{input}, \mathsf{sid}, \mathcal{P}, P_i, x_i)$ to the functionality.

2. Upon receiving the inputs from all parties, evaluate $y \leftarrow f(x_1, \dots, x_n)$. Send adversary $\mathcal{S}$ the message $(\mathsf{output}, \mathsf{sid}, \mathcal{P}, y)$.

3. On receiving $(\mathsf{generateOutput}, \mathsf{sid}, \mathcal{P}, P_i)$ from $\mathcal{S}$ the ideal functionality outputs $(\mathsf{output}, \mathsf{sid}, \mathcal{P}, y)$ to P_i. (And ignores the message if inputs from all parties in $\mathcal{P}$ have not been received.)

Figure .3: General Functionality for Deterministic Single Output Functionalities.