Impossibility of Two-Round MPC with the Black-Box Use of Additive Homomorphic Encryption

by

Ali Ghadirli

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2024

© Ali Ghadirli 2024

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Minimizing the number of rounds in the context of the Multiparty Computation (MPC) realm with respect to an arbitrary number of semi-honest adversaries is considered one of the branches that has gotten attention from researchers recently. Garg et al. [2] proved that two-round semi-honest MPC is impossible from black-box use of two-round oblivious transfer (OT).

Before this work, Garg and Srinivasan [16] and Benhamouda and Lin [7] showed a construction of a two-round MPC with a non-black-box use of the underlying two-round OT. Constructions of cryptographic protocols with the black-box use of cryptographic primitives have the advantage of being more efficient compared to non-black-box constructions, since in these constructions treat the underlying primitives as oracles which simplifies protocol design and analysis, leading to potentially more efficient constructions.

Reducing the number of rounds has the advantage of making parties able to send their first messages and go offline until all the other parties send their message of the second round and compute the output.

Our main result in this paper is to prove an impossibility result: We show that a two-round MPC based on black-box use of additive homomorphic encryption is impossible. This result is stronger than the previous result by Garg et al. [2], mainly because OT can be constructed using additive homomorphic encryption.

Acknowledgements

I acknowledge my supervisor, Prof. Mohammad Hajiabadi, who helped me to learn more about Cryptography. I also acknowledge the help I received during my master's degree from: Shirley Chen, Soroush Mortazavi, Jesse Elliot, Sabrina Mokhtari, Vecna, Rasoul Akhavan Mahdavi, Vahidreza Asadi, Nils Lukas, Monireh Safari, and Simon Oya.

Dedication

To people who care about cryptography and liberty.

Table of Contents

Author's Declaration ii

Abstract iii

Acknowledgements iv

Dedication v

List of Figures viii

List of Tables ix

1 Introduction 1

1.1 Outsourced Computation 2

1.2 Multiparty Computation 3

2 Background 7

2.1 Yao's Protocol 7

2.1.1 Oblivious Transfer 10

2.1.2 Yao's Garbled Circuit Protocol 10

2.2 General MPC 12

2.3 Black-Box Reductions in Cryptography 13

2.4 Formal Definition of MPC . . . 15
2.5 Additive Homomorphic Encryption . . . 16
2.6 Related Works . . . 18
2.7 Gap Analysis: . . . 19

3 Main Result **23**
3.1 Setup . . . 23
3.2 Step 1: Publicly Decodable Transcript . . . 25
3.3 Step 2: Removing **d** oracle calls . . . 28
3.3.1 Proof of Lemma 3.3.1 . . . 28
3.4 Step 3: Removing Hom queries . . . 40

4 Conclusion **41**

5 Future Work **43**
5.1 Shoup's Generic Group Model . . . 43
5.2 Maurer's Generic Group Model . . . 45
5.3 Differences of GGMs . . . 45

References **47**

APPENDICES **52**

A Appendix **53**

List of Figures

2.1 Transformations of 2-round 3-party MPC with the black-box use of OT . . 20

3.1 Transformations of 2-round 3-party MPC with the black-box use of AHE . 26
3.2 Homomorphic Binary Tree constructed using Hom oracle calls 30
3.3 Binary Tree of Computation 34

List of Tables

2.1 Comparison of Multi-Party Computation (MPC) Protocols 22
2.2 Comparison of Multi-Party Computation (MPC) Protocols 22

Chapter 1

Introduction

Secure Multiparty Computation (MPC) protocols represent a fundamental subfield in the domain of cryptography, enabling a group of individuals to collaboratively compute a function f over their inputs while ensuring that none of the inputs are revealed to the other participants. This field was pioneered by Andrew Yao in 1982, through his introduction to the "Millionaires' Problem" which involves two millionaires, Alice and Bob, who want to determine who is wealthier without disclosing their exact wealth to each other or a trusted third party.

Yao's formulation of the Millionaires' Problem was the first to articulate the possibility of performing computations on private data without ever revealing the data itself. Since then, MPC has been an important topic within the cryptography community and over the years has grown to address different complex computational challenges in different fields such as finance, healthcare, and government, where privacy is important.

The growth of MPC has been accompanied by significant advancements in the design of the algorithms and having different cryptographic primitives, which allows the execution of more complex functions with more efficiency. The initial theoretical framework of Yao has been refined and expanded to include different computational models and different adversarial conditions. As an example, [18] extended Yao's protocol to allow more than two parties to compute a function f.

In addition, the practical applications of MPC have become more apparent recently. For example, in the healthcare sector, MPC can enable hospitals to collaborate on patient data for research purposes without compromising patient privacy. In the financial sector, institutions can use MPC for privacy-preserving risk analysis and fraud detection, enabling them to collaborate without exposing sensitive data.

1.1 Outsourced Computation

Outsourced computation protocols play an essential role in modern computing. In an outsourced computation protocol, one party, often referred to as a client, owns sensitive data and wants to compute a complex function on the data and get the result. To achieve this, the data are encrypted and sent to another party, known as the server, which has more computational power. The server performs the required computations on the encrypted data without gaining any knowledge of the underlying data, and after performing the function, the server will send back the result, which is in an encrypted format. The client then decrypts these results to obtain the desired output. This process ensures that data confidentiality is maintained throughout the process.

Homomorphic Encryption

Homomorphic Encryption (HE) is a cornerstone that enables outsourced computation. It allows the client to encrypt a message m and obtain a ciphertext c, send it to the server, then the server can homomorphically evaluate a function f on the ciphertext c to obtain a ciphertext c^* and send it back to the client. At this stage, the client can decrypt c^* using its secret key to obtain the result of applying the function f on the plaintext m. This powerful property of HE has revolutionized the potential of privacy-preserving computations.

Variants of Homomorphic Encryption

1. **Partially Homomorphic Encryption (PHE):** This form of HE supports operations of a specific type, which could be addition or multiplication, but not both. This type of HE is suitable for applications that require repetitive operations of a single type. For example, Paillier cryptosystem [30] and Boneh cryptosystem [8], which enables PHE for additive operations.

2. **Leveled Homomorphic Encryption (LHE):** LHE supports operations up to a certain depth, which means that it can handle more complex computations than PHE but still has limitations on the depth of the computational circuit it can evaluate. This limitation is due to the fact that on every operation, the ciphertext has certain amounts of error and if one keeps doing this operation enough, the error will pass a certain threshold which results in incorrect decryption. This variant is useful in scenarios where the complexity of the computation is moderate and well-defined. The FV cryptosystem [12] is one example of LHE.

3. **Fully Homomorphic Encryption (FHE):** Introduced by Craig Gentry in 2009, FHE is the most general form of HE. It enables both addition and multiplication operations on the ciphertext, which allows to have general computation on encrypted data. Gentry's breakthrough was built on the visions laid out by Rivest, Adleman, and Dertouzos [34], which opened new frontiers for secure computation across a vast number of applications.

In this thesis, we focus on a specific subtype of PHE known as Additive Homomorphic Encryption (AHE). AHE enables one to outsource performing an arbitrary additive function on the ciphertexts. AHE offers fewer functionalities compared to FHE, but what makes it valuable in certain applications is its simplicity and lower computational overhead.

By focusing on AHE, this thesis aims to explore the capabilities and limitations within the context of two-round multiparty computation protocols. The choice of AHE is driven by the fact that one can construct an MPC protocol with the use of AHE, thus it is a natural question to ask:

Problem. *is it possible to construct two-round MPC with the black-box use of Additive Homomorphic Encryption?*

By focusing on AHE, this thesis aims to explore the possibility of constructing a two-round MPC with black-box use of AHE. Since AHE is a stronger assumption than OT, this thesis provides an improvement over the recent results of Garg et al. [2].

1.2 Multiparty Computation

Secure multiparty computation (MPC) is a powerful cryptographic paradigm that enables mutually distrusting parties to input variables into the protocol and collaborate on computation without compromising the privacy of their inputs. This concept goes beyond the scope of outsourced computation by involving all the parties in the computation, rather than having a single party perform computation on behalf of others.

The theoretical foundations of MPC were laid by Andrew Yao [36] in the early 1980s, when he introduced the *Garbled Circuits* technique as a method for two parties to jointly compute a function without revealing their individual inputs. The use of *Garbled Circuits* was later generalized to multiple parties by Wigderson et al. [18] but with the difference that the number of rounds for this protocol was linear in the depth of the circuit it was computing. Over the years, MPC has evolved from a theoretical curiosity to a practical tool used in various applications such as secure voting systems, privacy-preserving data aggregation, and collaborative machine learning.

Adversarial Models in MPC

In the domain of MPC, security is typically analyzed under two main adversarial models.

- **Semi-honest Model:** All parties follow the protocol honestly, but a certain threshold of them might share their data with each other to learn information about the other parties' inputs.
- **Malicious Model:** In comparison to the semi-honest model, corrupted parties in the malicious model can deviate from the protocol by not sending the messages as defined in the protocol. These adversaries can collaborate with other corrupted parties to learn additional information about the input of honest parties.

In this thesis, we assume that the adversaries are semi-honest.

Adversary Adaptiveness Models

The adaptiveness of adversaries in MPC protocols is a critical factor in protocol design:

- **Non-Adaptive Model:** In this model, the set of corrupted parties is fixed before the protocol execution begins and does not change throughout the computation.
- **Adaptive Model:** In this model, adversaries can choose which parties to corrupt during the execution of the protocol, potentially based on the information gathered since starting the protocol, which introduces more complexity to ensure the security of the protocol.

In this paper, we consider the adversaries as *Non-Adaptive*.

Communication Models

Another aspect of MPC protocols is the communication model among participants.

- **Broadcast Channel:** In this model of communication, all messages sent in the protocol by all the parties are public to all parties involved in the computation. This model simplifies the communication model but may increase the risk of information leakage as all the parties see all the messages.

- **Private Channel:** In this model, for every pair of participants, there is an encrypted and secure communication channel that ensures that no outside observer can access the contents of the communications.

In this thesis we assume that the communication model is over private channels, since the result of this thesis is an impossibility result, this assumption makes the result of this thesis stronger.

Building cryptographic primitives relies on secure reduction, the primary works in MPC [36, 18] proved that the problem of realizing an n-party arbitrary function f reduces to realizing a basic 2-party functionality called Oblivious Transfer [11, 32].

In Oblivious Transfer (OT) functionality, there is a receiver and a sender in which, the receiver has $x \in \{0, 1\}$ and sends it to the sender, the sender sends back m_x where (m_0, m_1) are his inputs.

When the number of parties is equal to 2, the reduction of [36] is round-preserving and the number of rounds is equal to the number of the rounds for the underlying OT and also, in the original protocol of Yao, it makes black-box use of a PRG. A long-standing question has been whether n-party functionality f, where $n > 2$, can be reduced in a round-preserving manner to 2-round OT. Combining the result of [6] and [18], it shows that every n-party functionality f could be realized in constant rounds with parallel calls to OT oracle and black-box use of PRG. A better result of constructing general MPC with black-box use of 2-round OT is the work of [1] where they showed that an n-party functionality f could be constructed in 4 rounds by making parallel calls to OT. Since then, the question of whether it's possible to construct a general n-party MPC with black-box use of OT has been open until recently.

Srinivasan et al. [2] showed that if $n > 2$, there is no black-box reduction of n-party functionality f to 2-round Oblivious Transfer. On the other hand, Garg and Srinivasan [16, 7] constructed an MPC protocol with non-black-box use of the underlying OT.

A natural question in this line of work to ask is what primitives are enough to construct a general 2-round MPC. A good question to ask could be what are other primitives that one can construct Oblivious Transfer with in a black-box way but could not be used to construct 2-round MPC in a black-box manner?

Considering a general functionality MPC for 2 rounds is a good idea since it is the minimum number of rounds that can be considered; Also, the construction of reusable two-round MPC was studied assuming indistinguishability obfuscation [13, 15], witness encryption [19, 14], multi-key FHE [29, 31, 9] and also from DDH [4] and LPN [5] before in another line of work.

In this thesis, we prove that it is impossible to construct a two-round MPC based on the black-box use of additively homomorphic encryption.

Chapter 2

Background

In this chapter, we lay the groundwork for understanding secure multi-party computation (MPC) by delving into fundamental concepts and introducing one of the seminal protocols in the field: Yao's Protocol.

2.1 Yao's Protocol

Secure multi-party computation (MPC) protocols make a group of n people able to jointly compute a function of their inputs without revealing any of their inputs to other people. Yao [36] in his seminal work introduced the notion of MPC by asking a simple question for two parties:

How can two millionaires, Alice and Bob, determine who is richer without disclosing their actual wealth to each other?

Before diving into Yao's MPC protocol, we will first establish the essential background knowledge required for a comprehensive understanding of MPC protocols. Throughout this thesis, unless otherwise specified, we denote security parameters by λ.

Definition 2.1.1 (Negligible Function)**.** We call a function $g : \mathbb{N} \to \mathbb{R}^+$ a negligible function if for any positive integer c, there exists η_0, such that for all $\eta \geq \eta_0$, we have that $g(\eta) < \frac{1}{\eta^c}$.

In cryptographic protocols, the security parameter, denoted as λ, plays a crucial role in quantifying the level of security provided. It is often used to parameterize the size of

keys, the number of rounds, or other aspects of the protocol that influence its security. The concept of computational indistinguishability relies heavily on the security parameter.

Definition 2.1.2. (Computational Indistinguishability) Let $\lambda \in \mathbb{N}$ be a security parameter, let $D_1 = \{D_{1,\lambda}\}_{\lambda\in\mathbb{N}}$ and $D_2 = \{D_{2,\lambda}\}$ be two collection of distributions indexed by λ. Then we call D_1 and D_2 *computationally indistinguishable* if for every polynomial time adversary $\mathcal{A}$:

$$|\mathsf{Pr}[x_1 \leftarrow D_{1,\lambda} : \mathcal{A}(1^\lambda, x_1) = 1] - \mathsf{Pr}[x_2 \leftarrow D_{2,\lambda} : \mathcal{A}(1^\lambda, x_2) = 1]| \leq negl(\lambda)$$

We use $D_1 \approx_c D_2$ to show that D_1 and D_2 are computationally indistinguishable.

At its core, computational indistinguishability means that no polynomial-time bounded and non-uniform algorithm can distinguish between two samples from D_1 and D_2.

Now, we define the notion of round that we use in this thesis.

Definition 2.1.3 (Round definition)**.** In an MPC protocol, a round is a synchronized communication step where each party:

- **Sends**: Transmits messages to a specified subset of other parties. This subset may be all other parties, a predetermined group, or even just a single party, depending on the protocol's design.
- **Receives**: Collects messages sent to them by other parties during that same round.

A protocol is said to have m rounds if it completes the computation within m such synchronized communication steps.

To formally define the notion of MPC and its security and correctness, we need to define two more definitions to finally define the notion of multiparty computation. In what follows, we define the messages of the MPC protocols and also the view of each party in the protocol in a concrete way.

Definition 2.1.4 (Message Definition)**.** With the definition of round in Definition 2.1.3, We denote the message that party P_i sends to party P_j in round t of the protocol by $\mathsf{msg}^t_{i\to j}$, meaning that party P_j doesn't have access to $\mathsf{msg}^t_{i\to j}$ while computing $\mathsf{msg}^t_{j\to *}$ in round t.

Definition 2.1.5 (View of Parties)**.** For an n party m round MPC protocol Π with inputs $\vec{x} = (x_1, x_2, \ldots, x_n)$ and randomness $\vec{r} = (r_1, r_2, \ldots, r_n)$, we define $\mathsf{view}_{P_i}^{\Pi}$ as a function that takes as input the security parameter of the protocol Π, inputs and randomness of the parties and outputs the view of the party P_i which consists of its input, randomness and the messages he has sent and received, in other words, for $1 \leq t \leq m$:

$$\mathsf{view}_{P_i}^{\Pi}(1^{\lambda}, \vec{x}, \vec{r}) = (x_i, r_i, \{\mathsf{msg}_{i \to *}^{t}\}_{t \leq m}, \{\mathsf{msg}_{* \to i}^{t}\}_{t \leq m})$$

Definition 2.1.6. A two-party protocol is defined with respect to a joint function $f : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ called functionality. The interactive protocol $\langle A, B \rangle$ for this functionality should satisfy the following properties:

- **Correctness:** For all inputs $x, y \in \{0,1\}^*$:

$$\mathsf{Pr}[\mathsf{output}(\langle A(x), B(y) \rangle = f(x, y)] = 1$$

- **Security:** There exists PPT algorithms Sim_A and Sim_B such that for all inputs $x, y \in \{x, y\}^*$:

$$\mathsf{Sim}_A(x, f(x, y)) \approx_c \mathsf{view}_A(\langle A(x), B(y) \rangle)$$
$$\mathsf{Sim}_B(y, f(x, y)) \approx_c \mathsf{view}_B(\langle A(x), B(y) \rangle)$$

The security property defined above is known as the simulation-based security or ideal/real-world paradigm. In this paradigm, we assume that there is an Ideal world in which there is a trusted third party that takes inputs $(x_1, x_2, \ldots, x_n)$ from the parties, computes and returns $(y_1, y_2, \ldots, y_n) = f(x_1, x_2, \ldots, x_n)$ to the parties.

Since all the communication in the ideal world is happening over secure channels, the protocol in the Ideal world, the above protocol in Ideal world satisfies the intuition of security that no party can gain more information other than their input and output of the computation. On the other hand, we can define a Real world in which the protocol performs for parties by sending messages back and forth and not having a trusted third party.

We argue that in a secure protocol, Π, Real world view of the adversary contains no more information than its Ideal world view. If the view of party P_i on a given input in the Ideal world is denoted by $\mathsf{view}_i^{\mathsf{Ideal}}(x_1, x_2, \ldots, x_n)$ and in the Real world as $\mathsf{view}_i^{\mathsf{Real}}(x_1, x_2, \ldots, x_n)$, then for a set of adversaries C in the protocol, if there exists a PPT algorithm Sim that can take input the view of parties in the Ideal world and produces a distribution that is

indistinguishable from the views of adversaries in the Real world, then intuitively it means this protocol is secure, or in another words:

$$\mathsf{Sim}(\mathsf{view}_i^{\mathsf{Ideal}}(x_1, x_2, \ldots, x_n) \approx_c \mathsf{view}_i^{\mathsf{Real}}(x_1, x_2, \ldots, x_n)$$

The interaction of the adversary with the real protocol is compared to its interaction with an ideal functionality. If these two interactions are computationally indistinguishable, the protocol is considered secure.

With this background, we can start to define the components of Yao's protocol.

2.1.1 Oblivious Transfer

Oblivious transfer is a protocol between a receiver and a sender in which the receiver has a bit b and the sender has two strings m_0 and m_1 and the goal of the protocol is that the sender sends m_b to the receiver without receiver revealing b to the sender.

Definition 2.1.7 (**Oblivious Transfer**)**.** A two-party protocol $\langle \mathcal{S}, \mathcal{R} \rangle$ between a sender $\mathcal{S}$ and a receiver $\mathcal{R}$ is called *Oblivious Transfer*(OT) if it satisfies the security and the correctness.

- **Correctness:** For any input $m_0, m_1 \in \{0,1\}^*$:

$$\mathsf{Pr}[\mathsf{output}_{\mathcal{R}}(\langle \mathcal{S}(m_0, m_1), \mathcal{R}(b) \rangle = m_b] = 1$$

- **Security:** There exists efficient simulators $\mathsf{Sim}_{\mathcal{R}}$ and $\mathsf{Sim}_{\mathcal{R}}$ such that for all $m_0, m_1 \in \{0,1\}^*$ and $b \in \{0,1\}$ we have:

$$\mathsf{view}_{\mathcal{R}}((m_0, m_1), b) \approx_c \mathsf{Sim}_{\mathcal{R}}(b, m_b)$$
$$\mathsf{view}_{\mathcal{S}}((m_0, m_1), b) \approx_c \mathsf{Sim}_{\mathcal{S}}(m_0, m_1)$$

2.1.2 Yao's Garbled Circuit Protocol

Let $f : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^k$ be a two-party functionality between Alice and Bob, which they wish to jointly compute. Alice holds input x and Bob has input y. The goal is to compute $f(x, y)$ without revealing x to Bob or y to Alice. The function f is represented as a boolean circuit consisting of gates $G_1, G_2, \ldots, G_t$. Here is the description of the protocol.

1. Alice Garbling the Circuit

   (a) **Wire Labeling**

   - For each wire w in the circuit, Alice generates two random keys k_w^0 and k_w^1, where k_w^b represents the label for wire w carrying the bit b.

   (b) **Garbling the Gates:**

   - For each gate G_i in the circuit with input wires u and v and output wire w, Alice makes a garbled table T_i. Each entry in the garbled table is an encryption of the output wire's label using the input wires' labels as keys.
   - For all combinations of input labels (k_u^a, k_v^b) for input bits (a, b), Alice computes $E_{k_u^a, k_v^b}(k_w^{G_i(a,b)})$.
   - The garbled table T_i for gate G_i for all $i \leq t$ consists of these encrypted values.

   (c) **Input Encoding:**

   - For all bits x_i of Alice's input x, Alice selects the corresponding label $k_{w_i}^{x_i}$.
   - Alice sends the collection of all garbled tables and the encoded inputs $\{k_{w_i}^{x_i}\}$ to Bob.

2. Oblivious Transfer for Bob's Input

   - For all bits y_i of Bob's input y, Alice and Bob engage in an OT protocol. In each OT, Alice acts as the sender, and Bob acts as the receiver.
   - For each bit input y_j of Bob's input, Alice inputs $(k_{w_j}^0, k_{w_j}^1)$ for each input wire w_j.
   - Bob receives the label $k_{w_j}^{y_j}$ through OT without revealing information to each other.

3. Evaluation of the Garbled Circuit by Bob

   (a) **Evaluation:**

   - Bob labels corresponding to his input bits y obtained through OT and the labels corresponding to Alice's input bits x received from Alice.
   - For each gate G_i with input wires labeled (k_u, k_v) and output wire w, Bob tries to decrypt the entries of the garbled table T_i and one of them will be decrypt to label k_w, where $k_w = D_{k_u, k_v}(T_i)$

- Bob repeats this process for every gate and propagates the labels through the circuit.

(b) **Decoding the output:**

- After propagation, Bob obtains the labels for the output bits.
- Alice provides the mapping from output wire labels to actual bit values, enabling Bob to decode the final output $f(x, y)$.

For rigorous proof of the security and correctness of Yao's protocol, look at [25].

2.2 General MPC

Goldreich, Micali, and Wigderson [18] developed a generalization of the Yao protocol to address the problem of secure computation between multiple parties. GMW protocol achieves this goal by using *secret sharing* and the security of the protocol is against semi-honest adversaries, but could be extended to security against malicious adversaries. In contrast to Yao's protocol which uses 2-round OT to construct a 2-round 2-party computation, the number of the rounds in GMW is linear in the depth of the circuit. Thus, we can conclude that 2-round OT functionality is a complete primitive for MPC in semi-honest adversaries since one can construct the GMW protocol. Also, other results in this field showed that 2-round OT is complete for MPC in the presence of malicious adversaries [22, 23]. Table 2.1 gives an overview of important MPC protocols and Table 2.2 compares them with respect to the number of rounds, communication complexity, computation complexity, and security model.

But the gap that remains here is the fact that Yao's protocol uses 2-round OT to achieve 2-round two-party computation but GMW for n parties realizes the protocol in more than two rounds. Therefore, the following natural question arises:

Problem 2.2.1. Is it possible to construct a generic 2-round MPC with the black-box use of 2-round OT?

Srinivasan et al. [2] answered this question by showing that 2-round MPC from black-box use of 2-round OT is impossible.

2.3 Black-Box Reductions in Cryptography

Black-box reductions are a fundamental concept in cryptographic research introduced by Trevisan et al. [33]. In cryptography, a cryptographic primitive is known to have two characteristics called *correctness* and *security*. The correctness property is defined as the probability that we expect the primitive to act as described, and the security property is defined as a computational task that an adversary aims to solve. If the adversary is successful, we call that the adversary *breaks* the security of the primitive. If there is no adversary of a certain class that can break the security of the primitive, we call this primitive secure.

A black-box reduction in cryptography refers to a method of proving the security of a cryptographic protocol or construction by relying solely on the assumption that an underlying cryptographic primitive is secure, without relying on the internal structure of these primitives.

This means that the primitive is used as a *black box*, where the reduction does not access any internal details of the primitive and interacts with it only via its standard interface. Black-box reductions have some characteristics:

- **Simplicity and Generality:** Black-box reductions are favored for their simplicity and the generality that they have. They allow for the broad application of results across different implementations of the primitive, as long as the interface remains consistent.
- **Modularity:** It enables a modular approach to protocol design, where different components of a system can be analyzed and designed independently and later combined without compromising security.

Definition 2.3.1. (from [2]) A cryptographic primitive P is called fully black-box reducible to Q if for every construction q of Q with polynomial time oracle R, B, construction $p = R^q$ of Q has a property that (1) if q is correct, then p is correct and (2) if an adversary A breaks the security of p, then there is an adversary $B^{A,p}$ that breaks the security of q.

Definition 2.3.2. Let P be a cryptographic primitive and $\mathcal{O}$ be a set of oracles and let M be an unbounded oracle machine. We call M is $\mathcal{O}$-query-bounded if M can make at most polynomial number of queries to $\mathcal{O}$. We say that $p = B^{\mathcal{O}}$ is a construction of the primitive P in the presence of oracles $\mathcal{O}$ if no $\mathcal{O}$-query-bounded adversary can break p.

Proposition 2.3.1. Let P and Q be two cryptographic primitives that there is a fully black-box reduction from P to Q and let $\mathcal{O}$ be a set of oracles. If there is a construction p of P in the presence of oracles $\mathcal{O}$, there is also a construction of Q in the presence of $\mathcal{O}$.

Proof. Let R be the fully black-box reduction from P to Q and $p = C^{\mathcal{O}}$ be the construction of P, then $R^{C^{\mathcal{O}}}$ is a construction of Q which has the correctness property. If there is an adversary $A^{\mathcal{O}}$ with bounded number of queries to $\mathcal{O}$, then $B^{A^{\mathcal{O}},C^{\mathcal{O}}}$ with that can break the security of p with bounded number of queries. □

2.4 Formal Definition of MPC

In this section, we delve into a formal definition of MPC that specifically addresses the presence of oracles entities that can provide computational assistance to the parties. This definition will focus on the semi-honest security model, where parties are assumed to follow the protocol honestly but may try to gain additional information from the messages they receive.

Definition 2.4.1 (MPC)**.** We say that an MPC protocol is (δ, ϵ, S)-secure in the presence of oracles and semi-honest adversaries if with probability at least δ the protocol is correct, the variable ϵ upper bounds the privacy error and also a distinguisher can at most make S queries to the oracle.

Definition 2.4.2. Let $\mathcal{O}$ be a set of oracles, X_λ and Y_λ be finite sets, and $f_\lambda : X_\lambda^n \to Y_\lambda^n$ be functionality from inputs of n-parties to outputs of n-parties. Let $\Pi^{\mathcal{O}}$ be a n-party protocol that computes f_λ for n parties. We say that $\Pi^{\mathcal{O}}$ effectively computes f_λ if there is a Turing machine with access to oracles $\mathcal{O}$ that can compute the desired functionality in $\mathsf{Poly}(\lambda)$. Assume that the inputs to the protocol is $x = (x_1, x_2, \ldots, x_n)$, where x_i is the input of party P_i. We show the view of party P_i by $\mathsf{view}_{P_i}^{\Pi}(1^\lambda, x_1, x_2, \ldots, x_n)$, which consists of the randomness and input of P_i, transcript and answers of the oracle. We denote the transcript of the protocol as $\mathbb{T}(\Pi^{\mathcal{O}}(1^\lambda, x_1, x_2, \ldots, x_n))$. We say that $\Pi^{\mathcal{O}}$ is a $(\delta(\lambda), \epsilon(\lambda), S(\lambda))$-secure protocol in presence of $\mathcal{O}$, for $\delta(\lambda), \epsilon(\lambda) : \mathbb{N} \to [0, 1]$ and $S(\lambda) : \mathbb{N} \to \mathbb{N}$, if:

- **Correctness:** for every input $(x_1, x_2, \ldots, x_n) \in X_\lambda$ and for $i \in [n]$, party P_i will get the output from $\Pi^{\mathcal{O}}(x_1, x_2, \ldots, x_n)$ which with probability δ is going to be equal to the i-th bit of $f_\lambda(x_1, x_2, \ldots, x_n)$.
- **Security:** for every subset $T \in [n]$ and vector of inputs $x = (x_1, x_2, \ldots, x_n)$ and $x' = (x'_1, x'_2, \ldots, x'_n)$ where $f(x) = f(x')$ and $x_T = x'_T$, and a non-uniform distinguisher D which makes at most S queries to oracle $\mathcal{O}$:

$$|Pr[D^{\mathcal{O}}(\{\mathsf{view}_{P_i}^{\Pi}(1^\lambda, x)\}_{i \in T}) = 1] - Pr[D^{\mathcal{O}}(\{\mathsf{view}_{P_i}^{\Pi}(1^\lambda, x')\}_{i \in T})]| \leq \epsilon(\lambda)$$

One can consider a relaxation of the above definition when the inputs are chosen uniformly random from X_λ^n. With the same definition for correctness, we say that the protocol securely computes f over *random inputs*. Also, for a distinguisher D that makes at most $S(\lambda)$ queries to $\mathcal{O}$, and for every $T \subseteq [n]$, we have:

$$|Pr[D^{\mathcal{O}}(\vec{x}, \vec{x}', \{\mathsf{view}_{P_i}^{\Pi}(1^{\lambda}, \vec{x})\}_{i\in T}) = 1] - Pr[D^{\mathcal{O}}(\vec{x}, \vec{x}', \{\mathsf{view}_{P_i}^{\Pi}(1^{\lambda}, \vec{x}')\}_{i\in T})]| \leq \epsilon(\lambda)$$

Where $\vec{x}$ is uniformly chosen randomly from X_{λ}^{n} and $\vec{x}'$ is chosen randomly subject to $\vec{x}_T = \vec{x}'_T$ and $f(\vec{x}) = f(\vec{x}')$.

Theorem 2.4.1 (Random security to standard security)**.** If $(1-\eta, \epsilon, S)$-secure protocol $\Pi^{\mathcal{O}}$ securely computes a function f over *random inputs*, then $\Pi^{\mathcal{O}}$ securely computes f over *worst case* inputs, where $\Pi^{\mathcal{O}}$ is $(1 - N\eta, N^2\epsilon, S)$-secure, where N is the domain size of f.

Proof. Assume that the correctness of the protocol $\Pi^{\mathcal{O}}$ fails for some worst-case input $\vec{x} \in X_{\lambda}^{n}$ with probability at least $N\eta$, then if we consider the protocol over random inputs, the probability of failure will at least be η. Hence, the probability of correctness of the protocol $\Pi^{\mathcal{O}}$ is at least $1 - N\eta$. Also, if there exists a distinguisher D that can have an advantage at least $N^2\epsilon$ over worst case inputs $\vec{x}$ and $\vec{x}'$ where $f(\vec{x}) = f(\vec{x}')$ and $\vec{x}_T = \vec{x}'_T$ then one can build a distinguisher D' which on input $\vec{x}$ and $\vec{x}'$ outputs the result of D and 1 otherwise. Hence, D' advantage is at least ϵ, which is a contradiction. Thus the result follows. □

This theorem establishes a powerful connection between the security of MPC protocols over random inputs and their security over worst-case inputs. Since we have this connection, later on in this thesis we use this theorem to prove our results over random inputs, which using this theorem one can prove the same result with normal inputs.

2.5 Additive Homomorphic Encryption

Additive Homomorphic Encryption (AHE) is a type of encryption that enables computation on encrypted data. In this section, we formally define AHE and its properties.

Definition 2.5.1. An **Additive Homomorphic Encryption** $\mathcal{E}$ consists of a tuple of algorithms $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Evaluate})$ as below:

- $\mathsf{Gen}(1^{\lambda})$: an algorithm that is taking the security parameter λ as input and outputs a pair $(\mathsf{sk}, \mathsf{pk})$ as secret and public key.
- $\mathsf{Enc}(\mathsf{pk}, m; r)$**:** a randomized algorithm that is taking a public key pk, plaintext m, and randomness r and outputs a ciphertext c.

- $\mathsf{Dec}(\mathsf{sk}, c)$**:** a deterministic algorithm that is taking a secret key sk and a ciphertext c as input and deterministically outputs the plaintext m.

- $\mathsf{Evaluate}(\mathsf{pk}, c^*, \mathcal{C})$**:** a deterministic algorithm that is taking a public key pk and a tuple of ciphertext $c^* = (c_1, c_2, \ldots, c_n)$ alongside an additive circuit $\mathcal{C}$, which is a circuit only consists of addition gates, as input and outputs c'.

An additive homomorphic encryption scheme should satisfy the correctness and security as below:

Correctness: we call an AHE $\mathcal{E}$ *correct* if for every $(\mathsf{pk}, \mathsf{sk})$ outputs by $\mathsf{Gen}(1^\lambda)$, any additive circuit $\mathcal{C}$, any plaintexts $(m_1, m_2, \ldots, m_n)$ and ciphertexts $c' = (c_1, c_2, \ldots, c_n)$ where $c_i = \mathsf{Enc}(\mathsf{pk}, m_i)$ we have that:

$$c^* \leftarrow \mathsf{Evaluate}(\mathsf{pk}, \mathcal{C}, c') \implies \mathcal{C}(m_1, m_2, \ldots, m_n) = \mathsf{Dec}(\mathsf{sk}, c^*)$$

Security: A challenger runs $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$. For any PPT algorithm $\mathcal{A}$ with the input (m_0, m_1) to the challenger and the bit $b \leftarrow \{0, 1\}$, the challenger returns $c_b \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)$ to $\mathcal{A}$ and $b' \leftarrow \mathcal{A}(\mathsf{pk}, c_b)$. We say $\mathcal{E}$ is secure, if

$$|\mathsf{Pr}[b' = b] - \frac{1}{2}| \leq negl(\lambda)$$

The adversary $\mathcal{A}$ can use $\mathsf{Evaluate}(\mathsf{pk}, c_b, \mathcal{C})$ in the process of producing the bit b'.

Fact 2.5.1. Additive Homomorphic Encryption (AHE) allows for efficient multiplication of a ciphertext by a constant. Given a ciphertext $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$ and a constant η, it is computationally feasible to obtain a new ciphertext $\mathsf{Enc}(\mathsf{pk}, \eta \cdot m)$ that represents the encryption of the plaintext $\eta \cdot m$ by adding $\mathsf{Enc}(\mathsf{pk}, m)$ with itself, store it in a new variable and do the same for $\mathcal{O}(\log(\eta))$ times.

Lemma 2.5.1. Using Additive Homomorphic Encryption, one can construct an oblivious transfer protocol.

Proof. An OT protocol $\mathsf{OT} = (\mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_3)$ is defined as follows:

- $\mathsf{OT}_1(b; r)$: it takes receiver's choice bit $b \in \{0, 1\}$ and its random tape r and outputs the receiver's message otm_1.

- $\mathsf{OT}_2(\mathsf{otm}_1, (m_0, m_1); s)$: it takes receiver's message otm_1, the sender's inputs $m_0, m_1 \in \{0,1\}$ and its random tape s, and outputs sender's message otm_2.
- $\mathsf{OT}_3(\mathsf{otm}_1, \mathsf{otm}_2, b, r)$: it takes as input receiver's and sender's message as input alongside the sender's input and random tape, and outputs m_b if there exists a unique (m_0, m_1, s) for which $\mathsf{OT}_1(b, r) = \mathsf{otm}_1$ and $\mathsf{OT}_2(\mathsf{otm}_1, m_0, m_1, s) = \mathsf{otm}_2$

One can define $\mathsf{OT} = (\mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_3)$ using additive homomorphic encryption functions as follows:

- $\mathsf{OT}_1(b, r) = \mathsf{Enc}(\mathsf{pk}, b; r)$
- $\mathsf{OT}_2(\mathsf{otm}_1, (m_0, m_1); s) = \mathsf{Enc}(\mathsf{pk}, z) = \mathsf{Enc}(\mathsf{Enc}(\mathsf{pk}, b).(m_0 \oplus m_1) \oplus m_0;\ \mathrm{s})$
- $\mathsf{OT}_3(\mathsf{otm}_1, \mathsf{otm}_2, b, r) = \mathsf{Dec}(\mathsf{sk}, \mathsf{otm}_2)$

The correctness and the security of this OT protocol follow from the correctness and the security of AHE. □

Thus, exploring the problem of the impossibility of two-round MPC with the black-box use of AHE is a contribution to the recent work of Srinivasan et al. [2].

2.6 Related Works

The work of Srinivasan et al. [2] showed that two-round MPC with black-box use of two-round OT is impossible. We will briefly summarize their work.

Definition 2.6.1 (External Functionality)**.** Let $g(x, y)$ denote a two-party functionality for an MPC protocol that takes x and y from two players and returns $g(x, y)$ to both of them at the end of the protocol. We denote by f_g as the external version of g when f_g extends to a three-party functionality. In this extended setup, the third party does not provide an input but it will receive the output of the two-party protocol $g(x, y)$ at the end of the protocol.

Next, we define when the transcript of an MPC protocol is called *publicly decodable*. Later in this thesis, we use this definition multiple times.

Definition 2.6.2 (Publicly Decodable Transcript)**.** In an MPC protocol Π in the presence of oracle $\mathcal{O}$ for the functionality f with inputs $\vec{x} = (x_1, x_2, \ldots, x_n)$, we call this protocol *publicly decodable transcript* if there exists a deterministic function Dec that takes as input the transcript of the protocol and outputs the output of the protocol. In other words, $\exists\mathsf{Dec}$ such that:

$$f(1^\lambda, x_1, x_2, \ldots, x_n) = \mathsf{Dec}(\mathbb{T}(\Pi^{\mathcal{O}}(1^\lambda, x_1, x_2, \ldots, x_n)))$$

If the input of Dec is an invalid ciphertext, Dec will return $\perp$.

Fact 2.6.1. Every MPC protocol Π can be transformed into a protocol Π_1 where Π_1 has publicly decodable transcript. If after the execution of the protocol Π, a party P sends the output of the protocol to other parties, then this new protocol has a publicly decodable transcript.

To prove the impossibility result, they assumed the existence of a protocol for external 3-party functionality for AND in the presence of oracles $\mathcal{O} = \{\mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_3\}$, as in Lemma 2.5.1. Then, through a couple of compilations, one can transform the first 3-party protocol for external-AND functionality to a multi-round 2 party protocol for AND functionality in a random oracle model which are known to be impossible [10, 24, 20, 26]. Therefore, the first assumption of having a generic 2 round MPC with the black-box use of 2 round OT is wrong. The process of transforming the former protocol into the latter consists of a sequence of steps as follows in Figure 2.1.

The approach in [2] was to rule out the usage of the oracle OT_3 by emulating its functionality without querying this oracle, with the cost of privacy and correctness of the original protocol. At the end of transformations, the protocol is a multi-round protocol for the AND functionality $g(x, y) = x.y$ with PDT (Publicly Decodable Transcript 2.6.2) in a random oracle model, which is known to be impossible [20].

2.7 Gap Analysis:

Since the recent work of [2] proved the impossibility of 2-round MPC with black-box use of 2-round OT and since AHE is a stronger assumption than OT, showing the impossibility of 2-round MPC with black-box use of AHE is a new contribution.

In this thesis, we will closely follow the scheme provided by [2] to show the impossibility result. We will define oracles for which there exists a AHE protocol and we will try to

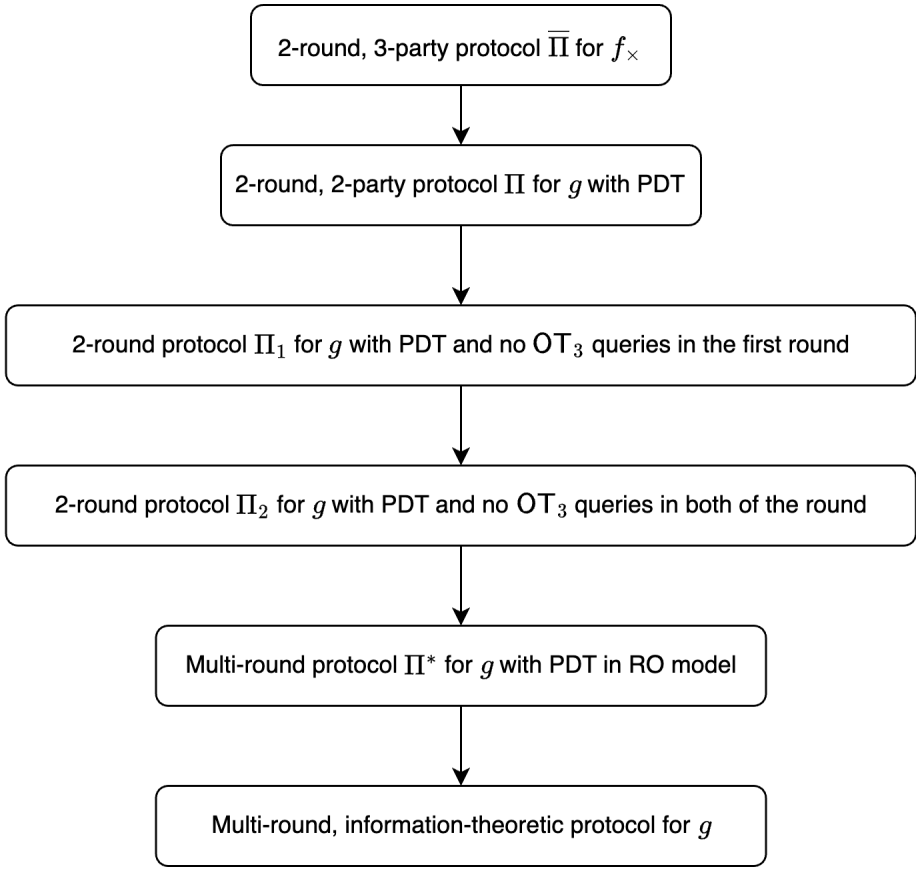


Figure 2.1: Process of the proof. Here, $g(x, y) = x.y$, $f_\times(x, y, \perp) = g(x, y)$, PDT denotes publicly decodable transcript

simulate the queries to some of the oracles without querying them which results in a multi-round protocol for $g(x, y) = x.y$ with PDT in random oracle model, which is known to be impossible.

Table 2.1: Comparison of Multi-Party Computation (MPC) Protocols

MPC Protocol	**Description**	**Strengths**	**Weaknesses**	**Use Cases**
Yao's Protocol	Two-party protocol where one party garbles a circuit, and the other evaluates it.	Secure against semi-honest adversaries, conceptually simple.	High communication complexity, not scalable for many parties.	Secure function evaluation, auctions
GMW Protocol	Secret sharing based works for both Boolean and arithmetic circuits.	Secure against active adversaries, efficient for small circuits.	High round complexity, communication overhead grows with circuit size.	Secure voting, private set intersection
BGW Protocol	Uses secret sharing and error correction codes.	Secure against active adversaries, efficient for arithmetic circuits.	High round complexity, communication overhead grows with circuit size.	Secure machine learning, secure auctions
SPDZ Family (SPDZ, MASCOT, etc.)	Preprocessing-based protocols using various techniques for efficient online phase.	Efficient for large-scale computations, actively secure.	Requires a trusted setup for preprocessing and complex implementation.	Large-scale secure computations, secure machine learning
BMR Protocol	Based on oblivious transfer and garbled circuits.	Secure against malicious adversaries, communication efficient.	Requires a large number of oblivious transfers, computationally expensive.	Secure two-party computations
Other Protocols (TinyOT, SPDZ2k, etc.)	Various protocols optimized for specific settings or types of computations.	Tailored to specific needs, can be very efficient.	Limited to specific scenarios, may require specialized hardware or assumptions.	Secure key management, verifiable secret sharing, private information retrieval

Table 2.2: Comparison of Multi-Party Computation (MPC) Protocols

MPC Protocol	**Rounds**	**Communication**	**Computation**	**Security Model**
Yao's Protocol	2	$O(\|C\|\lambda)$	$O(\|C\|\lambda)$	Semi-honest
GMW Protocol	$O(d)$	$O(\|C\|\lambda n^2)$	$O(\|C\|\lambda n^2)$	Active/Semi-honest
BGW Protocol	$O(d)$	$O(\|C\|\lambda n^2)$	$O(\|C\|\lambda n^2)$	Active/Semi-honest
SPDZ Family	$O(1)$ (online)	$O(\lambda n^2)$ (online)	$O(\lambda n^2)$ (online)	Active
BMR Protocol	2	$O(\|C\|\lambda + \lambda^2 \log \|C\|)$	$O(\|C\|\lambda^2)$	Malicious

Chapter 3

Main Result

In this section of the thesis, we describe the transformation of a 2-round 3-party MPC for external-AND with black-box use of AHE oracles into a multi-round, information-theoretic protocol for AND functionality in random oracle model, which we know that is impossible [20]. Hence, it proves the claim of this thesis. We begin by describing the protocol and the description of the oracles, and then we describe the transformations to rule out some of the oracles.

We start by defining oracles that give rise to a 2-round protocol for AHE.

3.1 Setup

Oracles are abstract entities that provide specific functionalities to cryptographic protocols. The presence of oracles can significantly impact the security of MPC protocols, as they may provide adversaries with additional information or computational power. In this section, we present a formal definition of the oracles that give rise to an additive homomorphic encryption protocol.

Definition 3.1.1 (Oracles)**.** $\mathcal{O}$ is a tuple of oracles $(\mathsf{g}, \mathsf{e}, \mathsf{d}, \mathsf{Hom})$ with the following syntax:

- g is a function that takes a secret key sk as input and outputs a public key pk.
- e is a probabilistic length tripling function that takes a public key pk, receiver's bit $b \in \{0,1\}$ and its random tape $r \in \{0,1\}^n$ as input and outputs a ciphertext c. Thus $\mathsf{e}(b; r) : \{0,1\}^{n+1} \to \{0,1\}^{3n+3}$

- d is a deterministic function that takes a secret key sk and a ciphertext c as input and outputs $b' \in \{0,1\} \cup \{\perp\}$.
- $\mathsf{Hom}(\mathsf{pk}, c_1, c_2)$ is a deterministic function that takes a public key pk and ciphertexts c_1 and c_2 as input and outputs $c \in \{0,1\}^* \cup \{\perp\}$, where $c = \mathsf{e}(\mathsf{pk}, \mathsf{d}(\mathsf{sk}, c_1) + \mathsf{d}(\mathsf{sk}, c_2))$.

Definition 3.1.2 (Oracle Transcript)**.** For an arbitrary oracle $\mathcal{O}$ and party P in an arbitrary MPC protocol, we denote $\mathbb{T}(\mathcal{O}, P)$ to show the transcript of the communication between the party P and the oracle $\mathcal{O}$. For the oracle $\mathcal{O}$, we use the notation $(u, v) \in \mathbb{T}(\mathcal{O}, P)$ if P queries oracle $\mathcal{O}$ on input u and gets the answer v.

Definition 3.1.3 (Valid Ciphertext)**.** For an arbitrary ciphertext $c^* \in \{0,1\}^{3n+3}$ in the range of the oracle e, we call it a *valid* ciphertext, if $\exists(\mathsf{pk}, b', r')$ such that $c^* \leftarrow \mathsf{e}(\mathsf{pk}, b'; r')$. If there is no such tuple, we call c^* an *invalid* ciphertext.

Lemma 3.1.1. If a party asks for the decryption of a ciphertext c^* by calling the d oracle, with probability at most $1/2^{(2n+2)}$ it could be the case that c^* is not produced by calling e oracle but is a valid ciphertext. In other words, a party can find a valid ciphertext c^* without querying oracle e with probability $\frac{1}{2^{(2n+2)}}$.

Proof. We know that the oracle e takes a public key pk, a plaintext $b \in \{0,1\}$ and a random tape $r \in \{0,1\}^n$ as input and outputs a ciphertext $c^* \in \{0,1\}^{(3n+3)}$, we know that the size of the domain of oracle e is 2^{n+1} and size of the range of this oracle is $2^{(3n+3)}$, thus, if one randomly picks a string $c \in \{0,1\}^{3n+3}$, with probability $\frac{2^{(n+1)}}{2^{(3n+3)}} = \frac{1}{2^{(2n+2)}}$ could be a valid ciphertext. □

For the proof of the main result of this thesis, we need to use a lemma in probability theory.

Lemma 3.1.2. Let $X_1, X_2, ..., X_{t+1}$ be *i.i.d* random variables coming from a Bernoulli distribution, where $Pr[X_i = 1] = p$ for $i \leq t+1$, then:

$$\mathsf{Pr}[X_1 = 0 \wedge X_2 = 0 ... \wedge X_t = 0 \wedge X_{t+1} = 1] \leq \frac{1}{t}$$

The proof is discussed in Appendix A.0.1

Lemma 3.1.3. Let $\{X_\lambda\}$, $\{Y_\lambda\}$ and $\{Z_\lambda\}$ be a sequence of finite sets. For any function $g : X_\lambda \times Y_\lambda \rightarrow Z_\lambda$ if there exists a 2-round $(1 - \mathsf{negl}(\lambda), \mathsf{negl}(\lambda), \lambda^{w(1)})$-secure protocol to efficiently compute the external function f_g, as defined in 2.6.1, in the presence of

oracle $\mathcal{O}$ defined as 3.1.1, then for an arbitrary polynomial $\rho(\lambda)$, there exists a multi-round $(1-\frac{1}{\rho(\lambda)}, \frac{1}{\rho(\lambda)}, \lambda^{w(1)})$-secure protocol that computes f_g on random inputs and private channels in random oracle model.

If we prove Lemma 3.3.3, then one can conclude that the lemma is also true for a specific functionality $g(x, y) = x.y$. Thus, one can conclude that there exists a multi-round 2-party MPC protocol in the random oracle model for AND functionality. This results in a contradiction and shows the impossibility as we discussed in the last chapter.

Next, We try to prove this lemma through a series of steps in Figure 3.1. We assume the existence of a 2-round 3-party protocol $\overline{\Pi}$ for the functionality $f_\times(x_1, x_2, \perp) = x_1.x_2$. In the first transformation, we emulate the third player and construct a new 2-round 2-party protocol Π for AND functionality of the inputs with the publicly decodable transcript. Then, we rule out the usage of d oracle in the first and second rounds of the protocol to achieve a new 2-round 2-party protocol Π, with a loss in correctness probability and gain in privacy error of the protocol. After this transformations, we will have a Multi-round, information-theoretic protocol for the AND functionality, which we know from [20] that it is impossible, where it leads to the contradiction of the existence of $\overline{\Pi}$.

3.2 Step 1: Publicly Decodable Transcript

In this section, we define a 3-party 2-round MPC protocol $\overline{\Pi}$ and we give a transformation of this protocol to another protocol Π where the new protocol is 2-party 2-round with an extra feature that its transcript is publicly decodable.

Definition 3.2.1 (Protocol $\overline{\Pi}$)**.** The MPC protocol $\overline{\Pi}$ is a 2-round 3-party protocol for the external functionality $f_g(x_1, x_2, \perp) = x_1.x_2$ is a $(1 - \mathsf{negl}(\lambda), \mathsf{negl}(\lambda), \lambda^{w(1)})$-secure protocol in the presence of oracle $\mathcal{O}$ on random inputs over secure channels.

We now prove that we can transform the 2-round 3-party protocol $\overline{\Pi}$ to the new 2-party 2-round protocol Π with the cost of having more oracle calls. This proof follows the proof in [2].

Lemma 3.2.1. Suppose that the 2-round 3-party protocol $\overline{\Pi}$ is a (δ, ϵ, S)-secure protocol $\overline{\Pi}$ for f_g on random inputs over secure channels in the case that at most 2 of the parties are corrupted. Then there exists a 2-round 2-party $(\delta, \epsilon, \Omega(S))$-secure protocol Π for g on random input over secure channels with publicly decodable transcript.

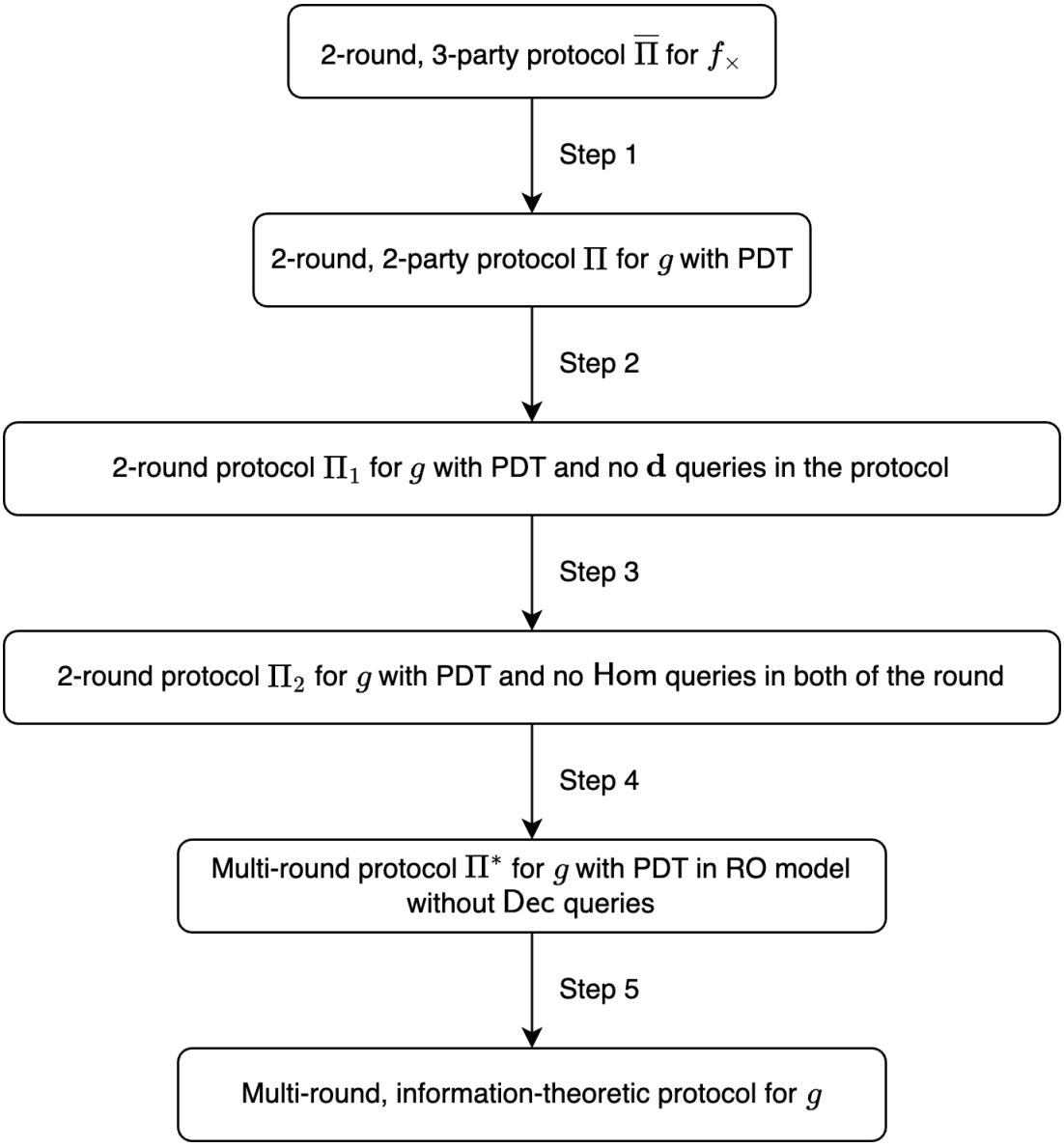


Figure 3.1: Different transformations of the proof. Here, $g(x, y) = x.y$, $f_\times(x, y, \perp) = g(x, y)$, PDT denotes publicly decodable transcript

Proof. We convert $\overline{\Pi}$ to Π in such a way that we emulate P_3 using a random tape provided by P_1 and performing P_3's functions by P_1. We emulate party P_3 in Π and remove it as below:

1. In the first round, party P_1 chooses a randomness r and emulates the actions of player P_3 using this randomness. In this way, we can consider P_1 and P_3 as a group that they are working together. In this round, party P_1 sends messages $\mathsf{msg}^1_{1\to 2}$ and $\mathsf{msg}^1_{1\to 3}$ to P_2 and P_1.

   Party P_1 can generate messages that P_3 would have sent in the original protocol, $\mathsf{msg}^1_{3\to 1}$ and $\mathsf{msg}^1_{3\to 2}$, and send it to P_1 and P_2 respectively. At the end, party P_2 will produce the messages $\mathsf{msg}^1_{2\to 1}$ and $\mathsf{msg}^1_{2\to 3}$ and will send both to P_1.

2. In the second round, P_1 can produce $\mathsf{msg}^2_{1\to 3}$, $\mathsf{msg}^2_{1\to 2}$, $\mathsf{msg}^2_{3\to 1}$ and $\mathsf{msg}^2_{1\to 3}$ based on its randomness and the messages P_1 received in the first round and send it to themselves and P_2 respectively. Then, P_1 uses the randomness of P_3 to generate $\mathsf{msg}^2_{3\to 1}$ and $\mathsf{msg}^2_{3\to 2}$ it would have sent in the original protocol and also P_1 will produce $\mathsf{msg}^2_{1\to 3}$ and $\mathsf{msg}^2_{1\to 2}$ and after that, she will send $(\mathsf{msg}^2_{3\to 2}, \mathsf{msg}^2_{1\to 2}, \mathsf{msg}^2_{1\to 3}, \mathsf{msg}^1_{1\to 3}, r)$ to P_2 and in this round, P_2 will send $(\mathsf{msg}^2_{2\to 1}, \mathsf{msg}^2_{2\to 3})$ to party P_1.

□

Lemma 3.2.2. The transcript $\mathsf{T} = \mathbb{T}(\Pi^{\mathcal{O}}(1^\lambda, x_1, x_2, \perp))$ of the 2-round 2-party MPC protocol Π is *publicly decodable*.

Proof. After the execution of the protocol $\overline{\Pi}$, P_3 is able to calculate output f_g based on its randomness r_3 and messages received from P_1 and P_2. In the protocol Π, messages to P_3 and its randomness r are embedded in the transcript T of the protocol. Let $h(\mathsf{T}; r)$ be the function of P_3 in $\overline{\Pi}$ in the second round that obtains the output. In Π, if we apply h to the transcript, it will obtain the output of the protocol and thus, this function acts as Dec, and Π has the publicly decodable transcript.

□

The security of the protocol Π follows the security of the protocol $\overline{\Pi}$. If there is a simulator for P_1 in the new case, we will consider it as a simulator for P_1 and P_3 in the primary protocol. In other words $\mathsf{Sim}^{\mathcal{O}}_1 = \overline{\mathsf{Sim}}^{\mathcal{O}}_{\{1,3\}}$ and $\overline{\mathsf{Sim}}^{\mathcal{O}}_2 = \mathsf{Sim}^{\mathcal{O}}_{\{2,3\}}$.

3.3 Step 2: Removing d oracle calls

In this section, we concretely define the protocol Π which we obtained in the last section. Later on in this chapter, we transform the 2-party 2-round protocol Π to another 2-party 2-round protocol $\Pi_1[\rho]$ where this protocol doesn't make d oracle calls and its security correctness probability depends on the polynomial $\rho(\lambda)$. For convenience, we call party P_1 and P_2, Alice and Bob respectively.

Definition 3.3.1 (Protocol Π)**.** The MPC protocol Π is a 2-round 3-party protocol for the functionality $g(x_1, x_2) = x_1.x_2$ is a $(1 - \mathsf{negl}(\lambda), \mathsf{negl}(\lambda), \lambda^{w(1)})$-secure protocol in the presence of the oracle $\mathcal{O}$ on random inputs over secure channels with publicly decodable transcript.

Next, we state a lemma that one can transform the protocol Π to another protocol $\Pi_1[\rho]$ where the new protocol does not make d oracle calls and loses probability and security, which will be parameterized over an arbitrary polynomial $\rho(\lambda)$.

Lemma 3.3.1. Let ρ be an arbitrary polynomial. A 2-party 2-round $(\delta, \epsilon, \Omega(S))$-secure protocol Π on random inputs over the secure channel and with a publicly decodable transcript can be transformed into another 2-round 2-party $(\delta - 1/\rho(\lambda), \epsilon + 1/\rho(\lambda), \Omega(S))$-secure protocol $\Pi_1[\rho]$ on random inputs over secure channel with a publicly decodable transcript.

In the remainder of this section, we describe the compilation that can be performed on the protocol Π to obtain the protocol $\Pi_1[\rho]$ and the proofs of its correctness and security.

3.3.1 Proof of Lemma 3.3.1

We start to prove the lemma 3.3.1 by introducing a compilation that uses the functions of Π to form protocol $\Pi_1[\rho]$ over random inputs, in which $\mathbf{d}$ queries can be answered by parties without making a call to $\mathbf{d}$ oracle.

The approach to remove the calls to the $\mathbf{d}$ oracle for the first round of the protocol $\Pi_1[\rho]$ is to save the transcript of the calls to oracles $\mathbf{d}$, $\mathbf{e}$ and Hom and for a $\mathbf{d}$ query of a ciphertext c^* each party tries to find its corresponding plaintext using these transcripts, instead of calling $\mathbf{d}$ oracle. Also, we denote the set of oracles $\{\mathsf{g}, \mathsf{e}, \mathsf{Hom}\}$ by $\mathcal{O}'$.

The approach to remove the $\mathbf{d}$ oracle calls in the second round is to use a set of ciphertexts and their decryption produced by the other party in the first round to obtain the decryption of the ciphertexts in the second round with an overwhelming probability without calling $\mathbf{d}$ oracle.

Definition 3.3.2 (Functions of Π)**.** We denote parties P_1 and P_2 by Alice and Bob and the functions they are performing by Alice and Bob, *e.g.* the function that Alice performs in the second round of the protocol Π in the presence of the oracle $\mathcal{O}$ is denoted by $\mathsf{Alice}_2^{\mathcal{O}}$. Let $\mathsf{P} \in \{\mathsf{Alice}, \mathsf{Bob}\}$, we denote by $\mathsf{P}_k^{\mathcal{O}}$ as the function in protocol Π that party P operates on their input, randomness and the transcript of the protocol that is visible to this party up until round k in the presence of the oracle $\mathcal{O}$ over random inputs. We denote by $\overline{\mathsf{P}}_k^{\mathcal{O}}$ the same function, but in protocol $\Pi_1[\rho]$. For example, $\overline{\mathsf{Alice}}_1^{\mathcal{O}'}$ is the function that party P_1 (Alice) is operating in the first round of the protocol $\Pi_1[\rho]$ in the presence of the oracle $\mathcal{O}'$ over random inputs.

To compile out the usage of $\mathbf{d}$ oracle queries in the second round of $\Pi_1[\rho]$, Alice (Bob) constructs a hint in the first round, as described in Definition 3.3.6, and sends it to Bob (Alice). This hint helps the parties find the decryption of the ciphertexts that produced by the other party in the first round with high probability. We continue by defining the Tree data structure that we are using and also an auxiliary function that helps us understand the proof.

Definition 3.3.3 (Homomorphic Binary Tree)**.** In an n-party MPC protocol Π, for a set of ciphertexts $C = (c_1, c_2, \ldots, c_n)$ and an additive function f, we consider the computation $f(c_1, c_2, \ldots, c_n)$ that party P performs as a binary tree $T = (V, E, r)$ where:

1. V represents the vertices of the binary tree where $\forall v \in V$ this node includes a ciphertext c.
2. E contains the edges of the tree where $(u, v) \in E$ if $v, u \in V$ and there exists an edge between these nodes.
3. r represents the root of the tree.

In addition, $\forall v \in V$, $(v.\mathsf{left}, v.\mathsf{right})$ represents the left and right child node of of v in T. Also, $\forall v \in V$, if $\{v.\mathsf{left}, v.\mathsf{right}\} \neq \emptyset$, then we will have that:

$$v = \mathsf{Hom}(\mathsf{pk}, v.\mathsf{left}, v.\mathsf{right})$$

for $\forall v \in V$, we denote the ciphertext it contains by $v.\mathsf{value}$. Also, we denote the color of each node $v \in V$ by $v.\mathsf{color}$. If party P can find the leaves of the sub-tree of node v all being in $\mathbb{T}(\mathsf{e}, P)$, as defined in 3.1.2 , then $v.\mathsf{color}$ is 1, otherwise 0.

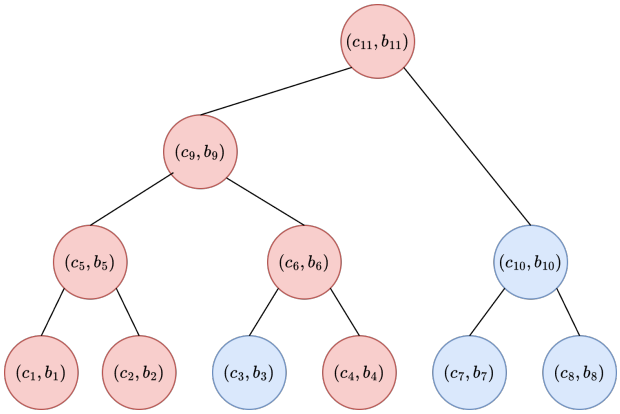


Figure 3.2: Homomorphic Binary Tree constructed using Hom oracle calls

We start the compilation by defining an auxiliary function. The function $\mathsf{LeafTraverse}$ takes a homomorphic binary tree T, a public key pk, and a pair of plaintext and ciphertext (b, c) with respect to the public key. The function $\mathsf{LeafTraverse}$ returns the values of those leaves of T with color 0.

Definition 3.3.4. Function $\mathsf{LeafTraverse}(T, c^*, \mathsf{pk})$ takes a tree T which represents a binary tree and a ciphertext c^* as an input and outputs a set of nodes v where $v.\mathsf{color} = 0$. let $L = \emptyset$ and do the following:

1. If $\exists(c_1, c_2, \mathsf{pk})$ such that $\{c^*.\mathsf{left.value}, c^*.\mathsf{right.value}\} = \{c_1, c_2\}$, then add the output of $\mathsf{LeafTraverse}(T, c_1, \mathsf{pk})$ and $\mathsf{LeafTraverse}(T, c_2, \mathsf{pk})$ to L.

2. else, add (c^*, pk) to L and return L.

Definition 3.3.5 (Decryptor)**.** For a list $L = \{(c_1, \mathsf{pk}), \ldots, (c_n, \mathsf{pk})\}$, the function $\mathsf{Decryptor}$ takes as input L and sk and outputs $L' = \{(c_1, b_1), \ldots, (c_n, b_n)\}$ where $\mathsf{pk} \leftarrow \mathsf{g}(\mathsf{sk})$ and $b_i = \mathsf{d}(\mathsf{sk}, c_i)$ for $i \in [1 : n]$.

Example 3.3.1. As an example of the Definition 3.3.3, Figure 3.2 shows a homomorphic binary tree T. $\mathsf{LeafTraverse}(T, c_{11}, \mathsf{pk})$, where T is the tree and c_{11} is the value of a node, this function returns the set $\{(c_1, \mathsf{pk}), (c_2, \mathsf{pk}), (c_4, \mathsf{pk})\}$ as the output of $\mathsf{LeafTraverse}$. Also, in Figure 3.2, a red leaf represents a node v where $v.\mathsf{color} = 0$, which means that the party who performed the computation, didn't make the corresponding encryption of $v.\mathsf{value}$.

Now, we are ready to introduce the protocol $\Pi_1[\rho]$. First, we introduce another free variable t in the definition below and later parameterize the polynomial ρ over t.

Definition 3.3.6 (Compilation of d oracle)**.** We define the compiled protocol $\Pi_1[\rho]$ as below, where it is parameterized over t, which we will specify later.

- $\overline{\mathsf{Alice}_1}^{\mathcal{O}'}$: Let $S_1, S_2 = \emptyset$. simulate the execution of $\mathsf{Alice}_1^{\mathcal{O}}(x_1; r_1)$, where we show the oracles as $\mathcal{O} = \{\mathsf{g}, \mathsf{e}, \mathsf{d}, \mathsf{Hom}\}$ and $\mathcal{O}' = \{\mathsf{g}, \mathsf{e}, \mathsf{Hom}\}$:

  1. For every g-query sk asked by Alice, call $\mathsf{pk} \leftarrow \mathsf{g}(\mathsf{sk})$, add $(\mathsf{pk}, \mathsf{sk})$ to S_1 and return pk.
  2. For every e-query (pk, b, r) asked by Alice, call $c \leftarrow \mathsf{e}(\mathsf{pk}, b; r)$, add (pk, b, c) to S_2 and return c.
  3. For every d-query (sk, c) asked by Alice, if $\exists \mathsf{pk}, b$ such that $(\mathsf{pk}, \mathsf{sk}) \in S_1$ and $(\mathsf{pk}, b, c) \in S_2$, return b; else $\perp$.
  4. For every Hom-query (pk, c_1, c_2) asked by Alice, if $\exists (b_1, b_2, \mathsf{sk})$ such that $(\mathsf{pk}, \mathsf{sk}) \in S_1$, $(\mathsf{pk}, b_1, c_1) \in S_2$ and $(\mathsf{pk}, b_2, c_2) \in S_2$:
     (a) call $c^* \leftarrow \mathsf{e}(\mathsf{pk}, b_1 + b_2)$
     (b) add $(\mathsf{pk}, b_1 + b_2, c^*)$ to S_2.
     (c) return c^*.

     else, return $\perp$.
  5. Let $\mathsf{msg}^1_{1\to 2}$ be the output of $\mathsf{Alice}_1^{\mathcal{O}}(x_1; r_1)$ simulated as above.

  Bob in Head: Let $H_{Alice} = \emptyset$, for $i = 1$ to t do:

  Let $\mathsf{PTree} = \emptyset$. Simulate the execution of $\mathsf{Bob}_2^{\mathcal{O}}(x_2', \mathsf{msg}^1_{1\to 2}; r_2')$ for $(x_2', r_2') \leftarrow \{0,1\}^*$:

  1. For every g-query sk asked by Bob's simulation, call $\mathsf{pk} \leftarrow \mathsf{g}(\mathsf{sk})$ and add $(\mathsf{pk}, \mathsf{sk})$ to S_1.
  2. For every e-query $(\mathsf{pk}, b; r)$ asked by Bob's simulation, call $c \leftarrow \mathsf{e}(\mathsf{pk}, b; r)$, add (pk, b, c) to S_2, return c.
  3. For every d-query (sk, c^*) asked by Bob's simulation:
     if $\exists \mathsf{pk}, b$ such that $(\mathsf{pk}, \mathsf{sk}) \in S_1$ and $(\mathsf{pk}, b, c^*) \in S_2$:
     (a) add $\mathsf{Decryptor}(\mathsf{LeafTraverse}(\mathsf{PTree}, c^*, \mathsf{pk}), \mathsf{sk})$ to H_{Alice}.
     (b) return b.

     else, return $\perp$.
  4. For every Hom-query (pk, c_1, c_2) asked by Bob's simulation:
     if $\exists (b_1, b_2, \mathsf{sk})$ such that $\{(\mathsf{pk}, b_1, c_1), (\mathsf{pk}, b_2, c_2)\} \subseteq S_2$ and $(\mathsf{pk}, \mathsf{sk}) \in S_1$

(a) call $c^* \leftarrow \mathsf{e}(\mathsf{pk}, b_1 + b_2)$, add $(\mathsf{pk}, b_1 + b_2, c^*)$ to S_2

(b) add $(c^*, c_1, c_2, \mathsf{pk})$ to PTree.

(c) return c^*.

else, return $\perp$.

Return $(H, \mathsf{msg}^1_{1\to 2})$.

- $\overline{\mathsf{Bob}_1}^{\mathcal{O}'}$: same as $\overline{\mathsf{Alice}_1}^{\mathcal{O}'}$.
- $\overline{\mathsf{Bob}_2}^{\mathcal{O}'}(x_2, \mathsf{msg}^1_{1\to 2}, H_{Alice}; r_2)$: Simulate the execution of $\mathsf{Bob}_2^{\mathcal{O}}(x_2, \mathsf{msg}^1_{1\to 2}; r_2)$:
  1. Let $S_3 = \emptyset, S_4 = H$ and $T = \emptyset$.
  2. For every g-query sk asked by Bob, call $\mathsf{pk} \leftarrow \mathsf{g}(\mathsf{sk})$ and add $(\mathsf{pk}, \mathsf{sk})$ to S_3.
  3. For every e-query (pk, b) asked by Bob, call $c \leftarrow \mathsf{e}(\mathsf{pk}, b; r_2)$ and add (pk, b, c) to S_4
  4. For every d-query (sk, c^*), call $\mathsf{pk} \leftarrow \mathbf{g}(\mathsf{sk})$, add $(\mathsf{pk}, \mathsf{sk})$ to S_3, if $\exists b$ such that $(\mathsf{pk}, b, c^*) \in S_4$, return b; else return $\perp$.
  5. For every Hom-query (pk, c_1, c_2) asked by Bob, if there exists (b_1, b_2) such that $(\mathsf{pk}, b_1, c_1) \subseteq S_4$ and $(\mathsf{pk}, b_2, c_2) \subseteq S_4$, call $c^* \leftarrow \mathsf{e}(\mathsf{pk}, b_1 + b_2; r_2)$,add the tuple $((c^*, b_1 + b_2), (c_1, b_1), (c_2, b_2)), \mathsf{pk})$ to T and $(\mathsf{pk}, b_1 + b_2, c^*)$ to S_4, otherwise call $c^* \leftarrow \mathsf{Hom}(c_1, c_2, \mathsf{pk})$ and add $(\mathsf{pk}, \perp, c^*)$ to S_4.
- $\overline{\mathsf{Alice}_2}^{\mathcal{O}'}$: Same as $\overline{\mathsf{Bob}_2}^{\mathcal{O}'}$
- $\overline{\mathsf{Dec}}^{\mathcal{O}'}$: Simulate the execution of $\mathsf{Dec}^{\mathcal{O}}(\mathbb{T}(1^\lambda, x_1, x_2))$, For every d-query by Alice (Bob) on c^*, if $\exists b$ such that $(c^*, b, \mathsf{pk}) \in \text{Alice}, \mathsf{e} \cup \text{Bob}, \mathsf{e}$, then return b, else return $\perp$.

Fact 3.3.1. Protocol $\Pi_1[\rho]$ does not call the $\mathbf{d}$ oracle in the first and second rounds and the decoding phase of the protocol.

Fact 3.3.2. In the protocol $\Pi_1[\rho]$, if party P in the first round asks for the decryption of a ciphertext c^* if it is a valid ciphertext, then with probability at least $1 - negl(\lambda)$ it is in the transcript of the oracle e.

Definition 3.3.7 (Ciphertext Types)**.** If Bob (Alice) asks for the decryption of a ciphertext c^* in the second round of the protocol $\Pi_1[\rho]$, then it could be either:

1. An invalid ciphertext.
2. A valid ciphertext that has been asked by one of the parties in the protocol $\Pi_1[\rho]$.
3. A valid ciphertext in which no parties produced this ciphertext.

Based on Lemma 3.1.1, the third case appears with a negligible probability. Thus, with an overwhelming probability ciphertext c^* is either invalid or valid with the definition above. Considering $P \in \{\text{Alice}, \text{Bob}\}$, where $\{P, \overline{P}\} = \{\text{Alice}, \text{Bob}\}$, and for $i \in \{1, 2\}$ we define different types of ciphertexts with respect to P:

1. Atomic_P^i ciphertext: produced by party P in Round i of the protocol $\Pi_1[\rho]$ by calling e oracle.
2. $\mathsf{Composed}_P^1$ ciphertext: produced by party P in the first Round of the protocol $\Pi_1[\rho]$ by operating an additive function f on a set of Atomic_P^1 ciphertexts.
3. $\mathsf{Composed}_P^2$ ciphertext: produced by party P in the second Round of the protocol $\Pi_1[\rho]$ by operating an additive function f on a set of Atomic_P^1, Atomic_P^2, and $\mathsf{Atomic}_{\overline{P}}^1$ ciphertexts.

Definition 3.3.8. For $P \in \{\text{Alice}, \text{Bob}\}$ and $i \in \{1, 2\}$, we use the word Atomic ciphertext to generally refer to Atomic_P^i ciphertexts and the word $\mathsf{Composed}$ ciphertext to refer to $\mathsf{Composed}_P^i$ ciphertexts.

Fact 3.3.3. Every $\mathsf{Composed}$ ciphertext c^* produced in protocol $\Pi_1[\rho]$ is produced by applying an additive function on a set of Atomic ciphertexts.

Example 3.3.2. As an example of fact 3.3.3 for the party $P \in \{\text{Alice}, \text{Bob}\}$ in the second round, Figure 3.3 shows a binary tree $T = (V, E, r)$ for $V = \{c_i\}_{i \in [1,11]}$, edges are as in Figure 3.3, and the root c_{11}.

The tree T represents the function $f = c_1 + c_2 + c_3 + c_4 + c_7 + c_8$ where the root c_{11} represents the result of computation of the function f.

Assuming that these ciphertexts are valid and considering the definition 3.3.7, set of ciphertexts $S_1 = \{c_1, c_2, c_4\}$ is of type $\mathsf{Atomic}_{\overline{P}}^1$ ciphertexts and thus we can state that $S_1 \in \mathsf{Atomic}_{\overline{P}}^1 \cup \mathsf{Composed}_{\overline{P}}^1$. In addition, we can state that the set of ciphertexts $S_2 =$

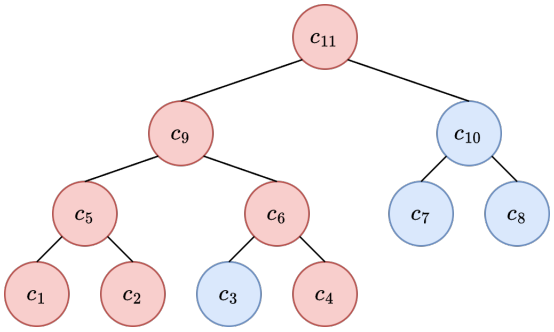


Figure 3.3: Binary tree of computation

$\{c_3, c_7, c_8\}$ is a subset of $\mathsf{Atomic}_P^1 \cup \mathsf{Atomic}_P^2$. This assumption is correct since, if for a $c_i \in S_2$ we have that $c_j \in \mathsf{Composed}_P^1 \cup \mathsf{Composed}_P^2$, then one can consider the tree $T' = (V', E', r')$ where $r'.\mathsf{value} = c_j$ to attach to the current tree T. Thus, we can assume that for every computation the tree is maximal, in the sense that the leaves are Atomic ciphertexts.

The binary tree represents an additive function performed on S_1 and S_2. A blue node in this binary tree represents an $\mathsf{Atomic}_P^1 \cup \mathsf{Atomic}_P^2$ ciphertext where party P can obtain the decryption of blue ciphertexts without calling d oracle by simply looking at the transcript of its calls to the e oracle.

A red node in the binary tree represent an $\mathsf{Atomic}_{\overline{P}}^1 \cup \mathsf{Composed}_{\overline{P}}^1$ ciphertext. Also, we can assume that since the tree T is maximal, every red leaf is an $\mathsf{Atomic}_{\overline{P}}^1$ because we can consider a $\mathsf{Composed}_{\overline{P}}^1$ and the Atomic ciphertexts that make them, as another tree T' contained in T.

In the second round of $\Pi_1[\rho]$, party P can not find the decryption of a red node simply by searching in $\mathbb{T}(P, \mathsf{e})$. As we will see later in this thesis, party P will use the hint that party $\overline{P}$ constructs in the first round and sends it over to P after the first round, to try to decrypt the $\mathsf{Atomic}_{\overline{P}}^1$ ciphertexts in the second round.

Fact 3.3.4. For every d query that party $P \in \{\text{Alice}, \text{Bob}\}$ wants to ask for a ciphertext c in the second round of the protocol $\Pi_1[\rho]$, we can assume that if it is a valid ciphertext, then with overwhelming probability $\exists T = (V, E, r)$ such that $r.\mathsf{value} = c$ and for every leaf $v_l \in V$ we have $v_l.\mathsf{value} \in \mathsf{Atomic}_{\overline{P}}^1 \cup \mathsf{Atomic}_P^1 \cup \mathsf{Atomic}_P^2$.

In what follows, we show how the hint that each party produces in the first round is going to be *helpful* for the other party in the second round. We will define what *helpful* means and then prove the claim.

Lemma 3.3.2. In the first round of the protocol $\Pi_1[\rho]$, if parties ask for q (where $q \in \mathbb{N}$) number of d queries, assuming that the correctness of the protocol Π is δ, then after the round one of $\Pi_1[\rho]$, this protocol has at least $\delta - \frac{q}{2^{(2n+2)}}$ correctness probability.

Proof. In the first round of the protocol $\Pi_1[\rho]$, if party $P \in \{\text{Alice}, \text{Bob}\}$ calls the d oracle for a ciphertext c^*, then c^* could be any of these cases:

1. An invalid ciphertext.
2. A ciphertext in $\mathsf{Atomic}_P^1 \cup \mathsf{Composed}_P^1$.
3. With probability $\frac{1}{2^{(2n+2)}}$ could be a valid ciphertext but it is not obtained by calling e oracle.

If party P asks for the decryption of a ciphertext c^* in the first round of $\Pi_1[\rho]$, then if the first case happens, it will output $\perp$ as the original protocol Π. If second case happens, then we know that $\exists(b, \mathsf{pk})$ such that $(\mathsf{pk}, b, c^*) \in \mathbb{T}(P, \mathsf{e})$, then party P can simply return b without actually calling d oracle.

On the other hand, if $\nexists(b, \mathsf{pk})$ such that $(\mathsf{pk}, b, c^*) \in \mathbb{T}(P, \mathsf{e})$, then P outputs $\perp$ which is consistent with the protocol Π. Thus, since the number of d queries that P and $\overline{P}$ ask in the first round of the protocol $\Pi_1[\rho]$ is q, the probability of third case happening is insignificant and bounded by $\frac{q}{2^{(2n+2)}}$. Thus, the correctness probability of the protocol $\Pi_1[\rho]$ after the first round is at least $\delta - \frac{q}{2^{(2n+2)}}$. □

Lemma 3.3.3. Assuming the existence of the 2 round 2 party $(\delta, \epsilon, \Omega(S))$-secure MPC protocol Π, the new protocol $\Pi_1[\rho]$ will have correctness of $\delta - 1/\rho(\lambda) - \frac{q}{2^{2n+2}}$, where the loss term of $\frac{q}{2^{2n+2}}$ comes from the first round of execution of $\Pi_1[\rho]$ and the correctness loss $1/\rho(\lambda)$ comes from the second round of execution of the protocol $\Pi_1[\rho]$.

Proof. Let $P \in \{\text{Alice}, \text{Bob}\}$ be one of the parties in the protocol $\Pi_1[\rho]$ where $\{P, \overline{P}\} = \{\text{Alice}, \text{Bob}\}$. Also, we denote the parties' functions of the protocol as $\mathsf{P} = \{\mathsf{Alice}, \mathsf{Bob}\}$. For example if $P = \text{Alice}$, by $\overline{\mathsf{P}}_1^{\mathcal{O}'}$ we mean the function that Alice performs in the first round of the protocol $\Pi_1[\rho]$ in the presence of the oracle $\mathcal{O}'$ or, in another word $\overline{\mathsf{Alice}}_1^{\mathcal{O}'}$ as defined in Definition 3.3.6.

Let r_P be the number of $\mathsf{Atomic}_{\overline{P}}^1$ ciphertexts that party P wants to query d oracle in the second round of protocol $\Pi_1[\rho]$. Also, we denoted the number of times P simulates $\mathsf{P}_2^{\mathcal{O}}$ in head in the first round of the protocol $\Pi_1[\rho]$ by t in 3.3.1. Later in this proof, we

establish the relation between this free variable t and the polynomial $\rho(\lambda)$ that protocol $\Pi_1[\rho]$ depends on.

We will prove that the party P can answer all of the d queries that party P asks in the second round of $\Pi_1[\rho]$ without calling the d oracle with the probability at least $1 - r_P/t$. We call the $H_{\overline{P}}$ *helpful* if party P can answer *all* the d queries using $H_{\overline{P}}$ and without calling d oracle.

After t times of simulation of $\mathsf{P}_2^{\mathcal{O}}$ in head 3.3.6 by party $\overline{P}$ in the first round, party $\overline{P}$ forms the hint $H_{\overline{P}}$ from the set S_2 in the first round and sends it to party P which s/he is going to use in the second round.

We denote the set of tuples $\{(\mathsf{pk}, b, c)\}$ were added to $H_{\overline{P}}$ in the j-th iteration of 1 by C_j. Thus, $H_{\overline{P}} = \bigcup_{j=1}^{t} C_j$. C_j contains a set of ciphertext, plaintext, and the public key of $\mathsf{Atomic}^1_{\overline{P}}$ ciphertexts that party $\overline{P}$ produced in the first round and the party P in the head asked for their decryption while simulating the function $P_2^{\mathcal{O}}$.

We know that $(t+1)$-th simulation of $P_2^{\mathcal{O}}$ over random inputs happens in the second round of the protocol when executing $\overline{\mathsf{P}}_2^{\mathcal{O}'}(x_2, \mathsf{msg}^1_{1\to 2}, H_{\overline{P}}; r_2)$ in $\Pi_1[\rho]$ over random inputs. When P wants to find out the decryption of a ciphertext c^* during the simulation of $\mathsf{P}_2^{\mathcal{O}}$ in the second round of $\Pi_1[\rho]$, if it is a valid ciphertext, then the ciphertext would be one of:

1. $\mathsf{Atomic}^1_P \cup \mathsf{Atomic}^2_P$ ciphertexts: In this case, since party P has access to $\mathbb{T}(P, \mathsf{e})$, then with overwhelming probability $1 - 1/2^{(2n+2)}$, party P can find the decryption of c^* by looking at $\mathbb{T}(P, \mathsf{e})$ and if $\exists(\mathsf{pk}, b, c^*) \in \mathbb{T}(P, \mathsf{e})$ such that $c^* \leftarrow \mathsf{e}(\mathsf{pk}, b)$, return b. Thus, in this case, the party P doesn't need to call d oracle.

2. $\mathsf{Atomic}^1_{\overline{P}}$ ciphertexts: In this case, party P tries to find the decryption of this $\mathsf{Atomic}^1_{\overline{P}}$ ciphertext by searching in hint $H_{\overline{P}}$. If $\exists(\mathsf{pk}, b)$ such that $(\mathsf{pk}, b, c^*) \in H_{\overline{P}}$, then party P answers d oracle queries of simulation of $\mathsf{P}_2^{\mathcal{O}}$ in the second round of $\Pi_1[\rho]$ without calling d oracle by simply returning b. We will find the probability of this event later in the proof.

3. $\mathsf{Composed}^2_P$ ciphertexts: In this case, we know that $\exists T = (V, E, r)$ where $r.\mathsf{value} = c^*$. The tree T is produced by party P by performing an additive function over Atomic ciphertexts. If party P is able to find the decryption of all the ciphertexts in the leaves of the tree T, then it can obtain the decryption of c^* by trying to find the decryption of Atomic ciphertexts as described above. If s/he is able to find the decryption of $\mathsf{Atomic}^1_{\overline{P}}$ ciphertexts in the leaves, then P can find the decryption of c^*. Thus, in the end, we need to find this the decryption of $\mathsf{Atomic}^1_{\overline{P}}$ ciphertexts

Call the event that party P can answer the $\mathsf{Atomic}^1_{\overline{P}}$ ciphertext $\mathbf{d}$ queries in 2 without calling $\mathbf{d}$ oracle by looking at the hint $H_{\overline{P}}$ a *hit*. On the other hand, if party P cannot find the decryption of a $\mathsf{Atomic}^1_{\overline{P}}$ ciphertext in $H_{\overline{P}}$ in 4, we call this event a *miss*, meaning that the hint provided by Alice is not useful to find the decryption of this particular ciphertext. We claim that for a large enough t, with overwhelming probability every decryption query in 4 could be answered.

For a valid ciphertext c, we define a set of random variables $S = \{X_{c,1}, X_{c,2}, \ldots, X_{c,t}, \ldots\}$ where $X_{c,i} = 1$ if in the i-th execution of $\mathsf{P}_2^{\mathcal{O}}$ with random inputs, party P asks for the decryption of this $\mathsf{Atomic}^1_{\overline{P}}$ ciphertext c which $\overline{P}$ produced in the first round.

We know that after 3.3.6, $\mathsf{P}_2^{\mathcal{O}}$ has been executed for t times and the next simulation of $\mathsf{P}_2^{\mathcal{O}}$ is in the execution of $\overline{\mathsf{P}}_2^{\mathcal{O}'}$ during the second round of the protocol. We know that in order for *miss* event to happen, party P needs to find the decryption of a ciphertext c^* in the second round of the protocol $\Pi_1[\rho]$ which did not appear in the hint $H_{\overline{P}}$. This means that in the executions of party P in the head in the first round of the protocol by $\overline{P}$, this valid $\mathsf{Atomic}^1_{\overline{P}}$ ciphertext wasn't asked by P in the head.

Since the correctness and the privacy of protocol $\Pi_1[\rho]$ is with respect to random input and during the execution of party P in the head, we use random inputs, thus the random variables in S are *i.i.d* random variables. Using the lemma A.0.1 we know that for a ciphertext c:

$$\mathsf{Pr}[hit] = 1 - \mathsf{Pr}[miss] = 1 - Pr[X_{c,1} = 0 \wedge X_{c,2} = 0 \wedge \ldots \wedge X_{c,t} = 0 \wedge X_{c,t+1} = 1] \geq 1 - \frac{1}{t} \quad (3.1)$$

We know that if party P asks for the decryption of r_P number of $\mathsf{Atomic}^1_{\overline{P}}$ ciphertexts $\mathsf{Ciph} = \{c_1, c_2, \ldots, c_{r_P}\}$ in the second round of the protocol, then $X_{c_i,t+1} = 1$ for $i \in [1 : r_P]$. We define a set of random variables Y_i for $i \in [1 : r_P]$ where $Y_i = 1$ if $\exists(\mathsf{pk}, b)$ such that $(\mathsf{pk}, b, c_i) \in H_{\overline{P}}$ or in another words, $Y_i := X_{c_i,1} \vee X_{c_i,2} \vee \ldots \vee X_{c_i,r_P}$. In order to see how useful is the hint $H_{\overline{P}}$ provided by party $\overline{P}$ in the first round, we define a random variable $\mathsf{success}$ for party P in the protocol $\Pi_1[\rho]$ where $\mathsf{success} = 1$ if $Y_i = 1$, $\forall i \in [1 : r_P]$. We know from 3.1 that $\mathsf{Pr}[Y_i = 1] \geq 1 - \frac{1}{t}$ and $\mathsf{Pr}[Y_i = 0] \leq \frac{1}{t}$. Using union bound we have that:

$$\begin{aligned}\mathsf{Pr}[\mathsf{success}] = \mathsf{Pr}[Y_1 = 1 \wedge Y_2 = 1 \wedge \ldots \wedge Y_r = 1] = 1 - \mathsf{Pr}[Y_1 = 0 \vee Y_2 = 0 \vee \ldots \vee Y_r = 0] \\ \geq 1 - (\sum_{i=1}^{r} \mathsf{Pr}[Y_i = 0]) \geq 1 - \frac{r_P}{t} \quad (3.2)\end{aligned}$$

Since r_P is the number of d queries of party P wants to ask for $\mathsf{Atomic}^1_{\overline{P}}$ ciphertexts, then we should have that $r_P + r_{\overline{P}}$ is bounded by the number of queries which is S_λ. Thus, if we choose t to be equal to $S_\lambda.\rho(\lambda)$ for an arbitrary polynomial $\rho(\lambda)$, then probability of success in 3.2 is at least $1 - 1/\rho(\lambda)$. Thus, protocol $\Pi_1[\rho]$ will lose $1/\rho(\lambda)$ in correctness probability in the second round and the proof of lemma 3.3.3 is complete.

□

Next, we discuss the security of the protocol $\Pi_1[\rho]$.

Lemma 3.3.4. Protocol $\Pi_1[\rho]$ is $(\epsilon + 1/\rho)$-secure, where ϵ is the security probability of the protocol Π.

Proof. We assume that there exists an adversary $\mathcal{A}$ that can compromise the security of the protocol $\Pi_1[\rho]$ and also we assume that the privacy probability of the protocol $\Pi_1[\rho]$ is $\epsilon'(\lambda)$. This means that for the adversary $\mathcal{A}$ and a party $P \in \{\text{Alice}, \text{Bob}\}$ and random set of inputs $\vec{x}, \vec{x}'$ we have:

$$|\mathsf{Pr}[\mathcal{A}^{\mathcal{O}'}(\vec{x}, \vec{x}', \mathsf{view}_P^{\Pi_1[\rho]}(1^\lambda, \vec{x})) = 1] - \mathsf{Pr}[\mathcal{A}^{\mathcal{O}'}(\vec{x}, \vec{x}', \mathsf{view}_P^{\Pi_1[\rho]}(1^\lambda, \vec{x}')) = 1]| > \epsilon'(\lambda) \tag{3.3}$$

We use this adversary $\mathcal{A}$ to build another adversary $\mathcal{A}'$ for the protocol Π that this adversary breaks the security of Π, this way, we prove that if the original protocol Π is secure, then the new protocol $\Pi_1[\rho]$ is also secure. Consider this adversary $\mathcal{A}'$, we know that for the party $P \in \{\text{Alice}, \text{Bob}\}$:

$$\mathsf{view}_P^{\Pi_1[\rho]}(1^\lambda, b) = (\mathsf{view}_P^{\Pi}(1^\lambda, b), H_{\overline{P}}) \tag{3.4}$$

In other words, the adversary $\mathcal{A}$ of the protocol $\Pi_1[\rho]$ takes as input two sets of inputs $\vec{x}$ and $\vec{x}'$ and view of one of the parties P and is able to distinguish between these two sets of inputs with a probability at least $\epsilon'(\lambda)$.

To prove the claim, we construct another adversary $\mathcal{A}'$ using the protocol $\Pi[\rho]$ and the adversary $\mathcal{A}$ that can break the security of the protocol Π. This way we can reach a conclusion that if Π is ϵ-secure, then $\Pi[\rho]$ is $(\epsilon + 1/\rho(\lambda))$-secure.

To construct the adversary $\mathcal{A}'$ for the protocol Π, we try to construct the view of party P in the protocol $\Pi_1[\rho]$ using the view of party P in the protocol Π and the functions of the parties in Π.

To show this, we provide an algorithm Alg such that:

$$\mathsf{view}_P^{\Pi_1[\rho]}(1^\lambda, b) \leftarrow \mathsf{Alg}^{\mathcal{O}}(\mathsf{view}_P^{\Pi}(1^\lambda, b)) \tag{3.5}$$

The algorithm Alg has access to the oracle $\mathcal{O}$,where $\mathcal{O} = \{\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathsf{Hom}\}$, which for $P = \{\text{Alice}, \text{Bob}\}$ simulates $\mathsf{view}_P^{\Pi_1[\rho]}$ using view_P^{Π} and $\mathbf{d}$ oracle calls. Next, we will give a description of the algorithm Alg.

For a party $P \in \{\text{Alice}, \text{Bob}\}$, let $\tilde{H}_{\overline{P}} = \emptyset$ and for $i = 1$ to t' do:

1. Party P simulates $\mathsf{P}_2^{\mathcal{O}}(x_1'; r_2')$ for $(x_1', r_1') \leftarrow \{0,1\}^*$.
2. For every d query (sk, c^*) for a $\mathsf{Atomic}_{\overline{P}}^1$ ciphertext that P in head is asking, P calls on d oracle on (sk, c^*) and adds (pk, b, c^*) to $\tilde{H}_{\overline{P}}$, where $b = \mathsf{d}(\mathsf{sk}, c^*)$.

Now we argue that for a large enough t', $\tilde{H}_{\overline{P}}$ is simulating the actual hint $H_{\overline{P}}$. For now, let's assume that the probability of this event happening is equal to Q which we will parameterize later. We will show that with at least probability Q, for every $(\mathsf{pk}, b, c) \in H_{\overline{P}}$ we have that $(\mathsf{pk}, b, c) \in \tilde{H}_{\overline{P}}$. If we show that this is possible, then the adversary $\mathcal{A}'$ can produce $\tilde{H}_{\overline{P}}$ themselves and use $\mathcal{A}$ as an adversary and distinguish between two different sets of inputs in Π. In other words since in Π party P can perform a function on its view, then:

$$\begin{aligned}
|\mathsf{Pr}[\mathcal{A}'^{\mathcal{O}}(\vec{x}, \vec{x}', \mathsf{view}_P^{\Pi}(1^\lambda, \vec{x})) = 1] - \mathsf{Pr}[\mathcal{A}'^{\mathcal{O}}(\vec{x}, \vec{x}', \mathsf{view}_P^{\Pi}(1^\lambda, \vec{x}')) = 1]| = \\
|\mathsf{Pr}[\mathcal{A}'^{\mathcal{O}}(\vec{x}, \vec{x}', \mathsf{Alg}(\mathsf{view}_P^{\Pi}(1^\lambda, \vec{x}))) = 1] - \mathsf{Pr}[\mathcal{A}'^{\mathcal{O}}(\vec{x}, \vec{x}', \mathsf{Alg}(\mathsf{view}_P^{\Pi}(1^\lambda, \vec{x}'))) = 1]| = \\
Q|\mathsf{Pr}[\mathcal{A}^{\mathcal{O}'}(\vec{x}, \vec{x}', \mathsf{view}_P^{\Pi_1[\rho]}(1^\lambda, \vec{x})) = 1] - \mathsf{Pr}[\mathcal{A}^{\mathcal{O}'}(\vec{x}, \vec{x}', \mathsf{view}_P^{\Pi_1[\rho]}(1^\lambda, \vec{x}')) = 1]| > Q.\epsilon'(\lambda)
\end{aligned} \tag{3.6}$$

Thus, if an adversary has $\epsilon'(\lambda)$ security advantage for the protocol $\Pi_1[\rho]$, then it will have advantage $Q.\epsilon'(\lambda)$ over the protocol Π.

Now we try to find the parameter Q. For every execution of P in the head in 3.3.1, if P asks for the decryption of a $\mathsf{Atomic}_{\overline{P}}^1$ ciphertext, P adds it to $\tilde{H}_{\overline{P}}$. As in 3.3.3, for a $m \in \tilde{H}_{\overline{P}}$ we define an event *hit* if $m \in H_{\overline{P}}$ and we define an event *miss* if $m \notin H_{\overline{H}}$. We use the same set of *i.i.d* random variables as in 3.3.3 by $S = \{X_{c,1}, X_{c,2}, \ldots, X_{c,t}, \ldots\}$ where $X_{c,i} = 1$ if in the i-th execution of $\mathsf{P}_2^{\mathcal{O}}$ with random inputs, party P asks for the decryption of this $\mathsf{Atomic}_{\overline{P}}^1$ We know that in order to have a *hit* in this case, for ciphertexts that c that

has been added to the hint in the j-th round of running $\overline{P}$ in the head, then the event of *miss* can be interpreted as:

$$\mathsf{Pr}[miss_c] = \mathsf{Pr}[X_{c,j} = 1 \wedge X_{c,t+1} = 0 \wedge \ldots \wedge X_{c,t+t'} = 0] \leq 1/t' \tag{3.7}$$

So, we call the event of missing a ciphertext c in the simulation of the hint $\tilde{H}_{\overline{P}}$, thus, the probability of generating every $(\mathsf{pk}, b, c) \in H_{\overline{P}}$ in $\tilde{H}_{\overline{P}}$ is as follows, where $Y_i = 1$ if $(\mathsf{pk}, b, c_i) \in H_{\overline{P}}$ and we will have a *miss* for c_i, thus:

$$\mathsf{Pr}[\mathsf{failure}] = \mathsf{Pr}[Y_1 = 1 \vee Y_2 = 1 \vee \ldots \vee Y_{|H_{\overline{P}}|}] \leq (\sum_{i=1}^{|H_{\overline{P}}|} \mathsf{Pr}[Y_i = 1]) \leq \frac{|H_{\overline{P}}|}{t'} \tag{3.8}$$

□

Hence, we can choose $t' = \rho(\lambda).|H_{\overline{P}}|$ where in this way $\mathsf{Pr}[\mathsf{failure}] \leq 1/\rho(\lambda)$ which shows the probability of not being able to produce $\mathsf{view}_P^{\Pi}(1^\lambda, b)$ from $\mathsf{view}_P^{\Pi_1[\rho]}(1^\lambda, b)$ is at most $1/\rho(\lambda)$, which proves the result. Thus, if there exists an adversary $\mathcal{A}$ for the protocol $\Pi_1[\rho]$, there exists an adversary $\mathcal{A}'$ for the protocol Π, which results in a contradiction.

Fact 3.3.5. Based on Lemma 3.3.3 and Lemma 3.3.4, we can conclude that protocol $\Pi_1[\rho]$ is a 2-round 2-party $(1 - 1/\rho(\lambda), 1/\rho(\lambda), \Omega(S)$ secure protocol with random inputs over secure channels in the presence of an oracle $\mathcal{O}'$.

Thus, this proves Lemma 3.3.1 which was the claim of this part of the thesis.

3.4 Step 3: Removing Hom queries

In the last section, we define the protocol $\Pi_1[\rho]$. As in definition 3.3.6, protocol $\Pi_1[\rho]$ does not use an oracle Hom. It can be done using the ideas from the impossibility of PKE in random oracle models. Those impossibility proofs can be extended to this work in the presence of a homomorphic oracle.

Chapter 4

Conclusion

In this thesis, we tried to answer the following question:

Is it possible to construct a generic 2-round n-party multiparty computation protocol with the black-box use of additive homomorphic encryption?

We started the thesis by defining different flavors of secure computation and defined multiparty computation and some applications of it later on in the thesis.

We started showing the impossibility of the problem by assuming that it is possible to have a generic 2-round n-party multiparty computation protocol $\overline{\Pi}$ for $n > 2$.

Since for every function f and $n > 2$ there exists such a 2-round MPC protocol, we assumed the existence of this protocol $\overline{\Pi}$ with a special functionality $f_\times$ for 3 parties P_1, P_2, and P_3 where party P_3 does not provide inputs. The functionality $f_\times(x_1, x_2, \perp) = x_1.x_2$ of the protocol $\overline{P}$ takes as input the inputs of the parties P_1 and P_2 and returns $x_1.x_2$ to all the parties after 2 rounds.

We used the existence of protocol Π to form the secondary protocol $\overline{\Pi}$ in which there exist only two parties P_1 and P_2 with inputs x_1 and x_2 and at the end of the protocol both parties get the output of $x_1.x_2$, with the difference that the transcript of this protocol $\overline{\Pi}$ is *publicly decodable*.

For the next step, we assumed the existence of a set of oracles $\mathcal{O} = \{\mathsf{g}, \mathsf{e}, \mathsf{d}, \mathsf{Hom}\}$ where it gives rise to additive homomorphic encryption primitive. We know that from [20] it's impossible to have a 2 party MPC protocol over random inputs in a random oracle model. Thus, we tried to rule out the usage of d and Hom oracles. We gave the algorithm to rule out the usage of d oracle with the cost of adding some privacy error and losing some correctness probability.

We didn't give the description of ruling out the usage of Hom oracle and we will complete in the paper version of this thesis later.

Chapter 5

Future Work

In this chapter, we introduce the original problem that this thesis was aiming to solve but appeared to be harder than we thought. We start by introducing some definitions that help the reader to understand the problem.

Problem 5.0.1. Is it possible to construct a generic n-party 2-round MPC protocol in Shoup's generic group model?

5.1 Shoup's Generic Group Model

The generic group model (GGM), as introduced by Shoup [35] is a theoretical framework for analyzing the computation hardness of cryptographic problems in a group-theoretic setting. It provides a way to abstract away the specific details of a group's representation and focus on the computational capabilities of generic algorithms that only interact with the group through a limited set of operations.

To understand Shoup's generic group model, we need to have some definitions.

Definition 5.1.1 (**Discrete Logarithm Problem**)**.** Let G be a finite cyclic group of order n with generator g. The discrete logarithm problem (DLP) is defined as follows:

Given an element h in G, find an integer x (if it exists) such that $g^x = h$.

The integer x is called the discrete logarithm of h to the base g and is denoted as $x = \log_g(h)$

Next, we need to define the DDH assumption:

Definition 5.1.2 (**DDH Assumption**)**.** Let G be a cyclic group of prime order q with generator g. The DDH assumption states that the following two distributions are computationally indistinguishable:

1. (g^a, g^b, g^{ab}) where a and b are randomly chosen from $\mathbb{Z}_q$.
2. (g^a, g^b, g^c) where a, b and c are randomly chosen from $\mathbb{Z}_q$

In other words, given g^a and g^b, it is computationally infeasible to distinguish between g^{ab} and a random element in G.

Definition 5.1.3 (Shoup's generic group model)**.** Let G be a finite cyclic group of order n. In the GGM, group elements are represented by randomly chosen unique labels (or encodings). These labels have no inherent algebraic meaning and can be thought of as abstract identifiers. The adversary is given access to two oracles:

1. **Group Operation Oracle**: This oracle takes two labels as input and returns the label corresponding to their product in the group.
2. **Equality Oracle**: This oracle takes two labels as input and determines whether they represent the same group element.

The adversary can only interact with the group through these oracles and is not allowed to access the underlying representation of the group elements.

Note 5.1.1 (Generic Algorithm)**.** A generic algorithm is an algorithm that only utilizes the group operation and equality oracles and does not exploit any specific properties of the group representation. The runtime of a generic algorithm is measured by the number of queries it makes to the oracle.

To better understand how Shoup's generic group model captures the computational limitations of generic algorithms, we present the following example focusing on the discrete logarithm problem:

Example 5.1.1. In the discrete logarithm problem, the adversary is given a generator g of the group and an element $h = g^x$. The goal is to find the integer x. In the GGM, the adversary only has access to the labels of g and h and can only interact with the group through the oracles. Shoup's work proved a lower bound on the number of oracle queries required by any generic algorithm to solve the discrete logarithm problem.

5.2 Maurer's Generic Group Model

To further explore the landscape of generic group models, we now present the definition of Maurer's model.

Maurer's Generic Group Model (GGM), introduced by Ueli Maurer [28], is another theoretical framework for analyzing the computational hardness of cryptographic problems in group-theoretic settings. It shares similarities with Shoup's GGM but differs in the way group elements are represented and how the adversary interacts with the group [37, 27]. In what follows, we define Shoup's and Maurer's generic group model.

Definition 5.2.1. Let G be a finite cyclic group of order n. In Maurer's GGM, group elements are represented by abstract handles (or pointers) that have no inherent algebraic meaning. The adversary cannot directly manipulate or examine the handles; they can only interact with them through two oracles:

1. **Group Operation Oracle**: This oracle takes two handles as input and returns a new handle representing their product in the group.
2. **Equality Oracle**: This oracle takes two handles as input and determines whether they represent the same group element.

The adversary can only access the group through these oracles and is not given any information about the underlying representation of the group elements.

Example 5.2.1. In the discrete logarithm problem, the adversary is given a generator g of the group and an element $h = g^x$. The goal is to find the integer x. In Maurer's GGM, the adversary only has access to the handles of g and h and can only interact with the group through the oracles. The same lower bounds on the number of oracle queries required by any generic algorithm to solve the discrete logarithm problem apply in Maurer's GGM as in Shoup's GGM.

5.3 Differences of GGMs

Key Differences between Shoup's and Maurer's Generic Group Models:

- **Representation of Group Elements**: A fundamental distinction lies in how group elements are represented within each model. In Shoup's GGM, group elements are assigned random labels, effectively providing the adversary with a means to directly identify and distinguish between individual elements. Conversely, Maurer's GGM employs abstract handles for group elements, denying the adversary any direct insight into their internal representation or relationships.

- **Adversary's Access to Group Elements**: This difference in representation leads to varying levels of access for the adversary. Within Shoup's GGM, the adversary is granted direct visibility of the group element labels, enabling them to compare and manipulate these labels during their analysis. In contrast, Maurer's GGM restricts the adversary's interaction with group elements solely through oracles, preventing them from directly accessing or manipulating the handles themselves.

In this thesis, we tried to prove the following impossibility result concerning the limitations of multiparty computation:

general n-party 2-round MPC is impossible with black box use of Additive Homomorphic Encryption

However, a potentially more impactful result and a promising direction for future research would be to delve deeper and investigate the following question:

Problem 5.3.1. Is it possible to construct a generic n-party 2-round MPC protocol in Shoup's generic group model?

Since additive homomorphic encryption can be constructed in Shoup's generic group model, proving an impossibility result for the problem 5.3.1 will be a stronger result.

References

[1] Prabhanjan Ananth, Arka Rai Choudhuri, and Abhishek Jain. A new approach to round-optimal secure multiparty computation. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 468–499, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Cham, Switzerland.

[2] Benny Applebaum, Zvika Brakerski, Sanjam Garg, Yuval Ishai, and Akshayaram Srinivasan. Separating two-round secure computation from oblivious transfer. In Thomas Vidick, editor, *ITCS 2020: 11th Innovations in Theoretical Computer Science Conference*, volume 151, pages 71:1–71:18, Seattle, WA, USA, January 12–14, 2020. LIPIcs.

[3] Judit Bar-Ilan and Donald Beaver. Non-cryptographic fault-tolerant computing in constant number of rounds of interaction. In Piotr Rudnicki, editor, *8th ACM Symposium Annual on Principles of Distributed Computing*, pages 201–209, Edmonton, Alberta, Canada, August 14–16, 1989. Association for Computing Machinery.

[4] James Bartusek, Sanjam Garg, Daniel Masny, and Pratyay Mukherjee. Reusable two-round MPC from DDH. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020: 18th Theory of Cryptography Conference, Part II*, volume 12551 of *Lecture Notes in Computer Science*, pages 320–348, Durham, NC, USA, November 16–19, 2020. Springer, Cham, Switzerland.

[5] James Bartusek, Sanjam Garg, Akshayaram Srinivasan, and Yinuo Zhang. Reusable two-round MPC from LPN. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *PKC 2022: 25th International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 13177 of *Lecture Notes in Computer Science*, pages 165–193, Virtual Event, March 8–11, 2022. Springer, Cham, Switzerland.

[6] 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, Baltimore, MD, USA, May 14–16, 1990. ACM Press.

[7] Fabrice Benhamouda and Huijia Lin. k-round multiparty computation from k-round oblivious transfer via garbled interactive circuits. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 500–532, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Cham, Switzerland.

[8] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In Joe Kilian, editor, *TCC 2005: 2nd Theory of Cryptography Conference*, volume 3378 of *Lecture Notes in Computer Science*, pages 325–341, Cambridge, MA, USA, February 10–12, 2005. Springer, Berlin, Heidelberg, Germany.

[9] 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, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Berlin, Heidelberg, Germany.

[10] Benny Chor and Eyal Kushilevitz. A zero-one law for Boolean privacy (extended abstract). In *21st Annual ACM Symposium on Theory of Computing*, pages 62–72, Seattle, WA, USA, May 15–17, 1989. ACM Press.

[11] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *Advances in Cryptology – CRYPTO'82*, pages 205–210, Santa Barbara, CA, USA, 1982. Plenum Press, New York, USA.

[12] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144, 2012.

[13] 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, San Diego, CA, USA, February 24–26, 2014. Springer, Berlin, Heidelberg, Germany.

[14] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors,

45th Annual ACM Symposium on Theory of Computing, pages 467–476, Palo Alto, CA, USA, June 1–4, 2013. ACM Press.

[15] Sanjam Garg and Akshayaram Srinivasan. Garbled protocols and two-round MPC from bilinear maps. In Chris Umans, editor, *58th Annual Symposium on Foundations of Computer Science*, pages 588–599, Berkeley, CA, USA, October 15–17, 2017. IEEE Computer Society Press.

[16] Sanjam Garg and Akshayaram Srinivasan. Two-round multiparty secure computation from minimal assumptions. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 468–499, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Cham, Switzerland.

[17] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 169–178, Bethesda, MD, USA, May 31 – June 2, 2009. ACM Press.

[18] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th Annual ACM Symposium on Theory of Computing*, pages 218–229, New York City, NY, USA, May 25–27, 1987. ACM Press.

[19] S. Dov Gordon, Feng-Hao Liu, and Elaine Shi. Constant-round MPC with fairness and guarantee of output delivery. 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 63–82, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Berlin, Heidelberg, Germany.

[20] Iftach Haitner, Eran Omri, and Hila Zarosim. Limits on the usefulness of random oracles. *Journal of Cryptology*, 29(2):283–335, April 2016.

[21] 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, Tallinn, Estonia, May 15–19, 2011. Springer, Berlin, Heidelberg, Germany.

[22] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In David Wagner, editor, *Advances in Cryptology –*

CRYPTO 2008, volume 5157 of *Lecture Notes in Computer Science*, pages 572–591, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Berlin, Heidelberg, Germany.

[23] Joe Kilian. Founding cryptography on oblivious transfer. In *20th Annual ACM Symposium on Theory of Computing*, pages 20–31, Chicago, IL, USA, May 2–4, 1988. ACM Press.

[24] Eyal Kushilevitz. Privacy and communication complexity. In *30th Annual Symposium on Foundations of Computer Science*, pages 416–421, Research Triangle Park, NC, USA, October 30 – November 1, 1989. IEEE Computer Society Press.

[25] 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.

[26] Mohammad Mahmoody, Hemanta K. Maji, and Manoj Prabhakaran. Limits of random oracles in secure computation. In Moni Naor, editor, *ITCS 2014: 5th Conference on Innovations in Theoretical Computer Science*, pages 23–34, Princeton, NJ, USA, January 12–14, 2014. Association for Computing Machinery.

[27] Ueli Maurer, Christopher Portmann, and Jiamin Zhu. Unifying generic group models. Cryptology ePrint Archive, Report 2020/996, 2020.

[28] Ueli M. Maurer and Stefan Wolf. Lower bounds on generic algorithms in groups. In Kaisa Nyberg, editor, *Advances in Cryptology – EUROCRYPT'98*, volume 1403 of *Lecture Notes in Computer Science*, pages 72–84, Espoo, Finland, May 31 – June 4, 1998. Springer, Berlin, Heidelberg, Germany.

[29] 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, Vienna, Austria, May 8–12, 2016. Springer, Berlin, Heidelberg, Germany.

[30] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238, Prague, Czech Republic, May 2–6, 1999. Springer, Berlin, Heidelberg, Germany.

[31] 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, Beijing, China, October 31 – November 3, 2016. Springer, Berlin, Heidelberg, Germany.

[32] Michael O. Rabin. How to exchange secrets with oblivious transfer. Cryptology ePrint Archive, Report 2005/187, 2005.

[33] Omer Reingold, Luca Trevisan, and Salil P. Vadhan. Notions of reducibility between cryptographic primitives. In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 1–20, Cambridge, MA, USA, February 19–21, 2004. Springer, Berlin, Heidelberg, Germany.

[34] Ronald L. Rivest and Michael L. Dertouzos. On data banks and privacy homomorphisms. 1978.

[35] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266, Konstanz, Germany, May 11–15, 1997. Springer, Berlin, Heidelberg, Germany.

[36] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science*, pages 162–167, Toronto, Ontario, Canada, October 27–29, 1986. IEEE Computer Society Press.

[37] Mark Zhandry. To label, or not to label (in generic groups). In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022, Part III*, volume 13509 of *Lecture Notes in Computer Science*, pages 66–96, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Cham, Switzerland.

APPENDICES

Appendix A

Proofs

Lemma A.0.1. Let $X_1, X_2, ..., X_{t+1}$ be independent random variables coming from Bernoulli distribution, where $Pr[X_i = 1] = p$ for $i \leq t + 1$, then:

$$\mathsf{Pr}[X_1 = 0 \wedge X_2 = 0... \wedge X_t = 0 \wedge X_{t+1} = 1] \leq \frac{1}{t}$$

Proof. Let $X_1, X_2, \ldots, X_{t+1}$ be independent random variables from a Bernoulli distribution where $\mathsf{Pr}[X_i = 1] = p$ for $i \leq t + 1$. We want to show that:

$$\mathsf{Pr}[X_1 = 0 \wedge X_2 = 0 \wedge \cdots \wedge X_t = 0 \wedge X_{t+1} = 1] \leq \frac{1}{t}.$$

Since the random variables are independent, the probability of the joint event can be written as the product of individual probabilities:

$$\mathsf{Pr}[X_1 = 0 \wedge \cdots \wedge X_t = 0 \wedge X_{t+1} = 1] = \mathsf{Pr}[X_1 = 0] \cdot \mathsf{Pr}[X_2 = 0] \cdot \ldots \cdot \mathsf{Pr}[X_t = 0] \cdot \mathsf{Pr}[X_{t+1} = 1].$$

Given $X_i \sim \text{Bernoulli}(p)$, we have $\mathsf{Pr}[X_i = 0] = 1 - p$ and $\mathsf{Pr}[X_{t+1} = 1] = p$. Therefore:

$$\mathsf{Pr}[X_1 = 0 \wedge X_2 = 0 \wedge \cdots \wedge X_t = 0 \wedge X_{t+1} = 1] = (1 - p)^t \cdot p.$$

To show that this probability is less than or equal to $\frac{1}{t}$, we need to demonstrate:

$$(1-p)^t \cdot p \leq \frac{1}{t}.$$

To maximize $(1-p)^t \cdot p$, we take the derivative with respect to p and set it to zero:

$$\frac{d}{dp}\left((1-p)^t \cdot p\right) = (1-p)^t - tp(1-p)^{t-1} = 0.$$

Solving this, we get:

$$(1-p)^t = tp(1-p)^{t-1} \implies 1-p = tp \implies p = \frac{1}{t+1}.$$

Evaluating the function at $p = \frac{1}{t+1}$:

$$(1-\frac{1}{t+1})^t \cdot \frac{1}{t+1} = \left(\frac{t}{t+1}\right)^t \cdot \frac{1}{t+1}.$$

Using the approximation $\left(1-\frac{1}{t+1}\right)^{t+1} \approx \frac{1}{e}$:

$$\left(\frac{t}{t+1}\right)^t \approx \frac{1}{e} \implies \left(\frac{t}{t+1}\right)^t \cdot \frac{1}{t+1} \approx \frac{1}{e} \cdot \frac{1}{t+1} \approx \frac{1}{t \cdot e}.$$

Since $\frac{1}{t \cdot e} \leq \frac{1}{t}$ for all $t \geq 1$, we conclude:

$$(1-p)^t \cdot p \leq \frac{1}{t}.$$

Thus, we have proven that:

$$\mathsf{Pr}[X_1 = 0 \wedge X_2 = 0 \wedge \cdots \wedge X_t = 0 \wedge X_{t+1} = 1] \leq \frac{1}{t}.$$

□