Multiparty Computation Combiners

Mingrui Zou

Master of Science

Cyber Security, Privacy and Trust

School of Informatics

University of Edinburgh

2024

Abstract

Combiners are a fundamental cryptographic tool that integrate multiple protocols to maintain security, even when some components may fail. This approach is particularly valuable in scenarios where the exact security of individual protocols cannot be guaranteed. This dissertation extends the concept of combiners to the realm of Secure Multi-Party Computation (MPC), an area crucial for scenarios where multiple parties jointly compute a function over their inputs while preserving the privacy of those inputs. MPC combiners offer a way to ensure that the overall system remains secure even if certain underlying protocols are compromised, thus providing an additional layer of robustness in cryptographic applications.

In this work, we provide the first formal definition of MPC combiners and focus on their construction within the two-party semi-honest model. Specifically, we propose and analyze a 1-out-of-2 MPC combiner, demonstrating its capability to ensure security for at least one party. Additionally, we extend our approach to a 2-out-of-3 MPC combiner, identifying potential flaws and subsequently proposing a less efficient but secure alternative construction, demonstrating the feasibility of achieving security even in more complex settings. Our findings suggest that innovative approaches may be required to overcome the inherent difficulties in achieving comprehensive security in MPC combiners.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Mingrui Zou*)

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Dr. Michele Ciampi, for his invaluable guidance, support, and encouragement throughout the course of this research. His expertise and insights have been instrumental in the completion of this dissertation.

I also wish to extend my heartfelt thanks to my parents and grandparents for their unwavering support and belief in my academic journey. Their love and encouragement have been my constant source of motivation.

Table of Contents

Chapter 1

Introduction

In recent years, Secure Multi-Party Computation (MPC) [2, 3, 10] has emerged as a critical area of cryptography, allowing multiple mutually suspicious parties to jointly compute a function of their local inputs without revealing any additional information beyond the output of the function to any subset of corrupted parties. This capability is indispensable in various fields, including cloud computing, artificial intelligence, and blockchain technologies, where secure data handling is paramount. MPC allows for private data analysis in cloud computing, facilitates collaborative machine learning in AI without compromising sensitive training data, and enhances privacy-preserving mechanisms in blockchain transactions. However, the security of these protocols often hinges on underlying cryptographic assumptions, such as the hardness of specific mathematical problems like factoring large integers or computing discrete logarithms. Additionally, assumptions like the availability of a Common Reference String (CRS), generated by a trusted party, are critical. If these assumptions fail, whether due to advances in computational power, such as the advent of quantum computing [1], or the dishonesty of the trusted entity, existing MPC protocols could become vulnerable.

To mitigate these risks, the concept of **Robust Combiner** has been introduced in cryptographic research [6, 8]. Combiner is a method that combines multiple cryptographic protocols in such a way that the overall system remains secure as long as at least a certain number of the underlying protocols is secure. This approach follows the principle of ”not putting all your eggs in one basket” and is particularly appealing in scenarios where the exact security of individual protocols cannot be guaranteed. For instance, combiners are particularly useful in scenarios where individual cryptographic primitives may become vulnerable over time due to advances in computational methods, yet the system remains secure by relying on the strength of the combined protocol. By

integrating multiple cryptographic protocols into a single framework through combiners, the security of the computation can be preserved even if some of the candidate protocols fail.

Despite the significance of combiners in other areas of cryptography, the concept of combiners specifically designed for MPC protocols has not been a focus of study in the existing academic research. This dissertation addresses this gap by providing the first formal definition of MPC combiners, proposing new black-box constructions that ensure a certain level of security, and offering rigorous analysis through theorems and proofs. Additionally, we discuss these results in the context of established cryptographic principles, linking our findings to broader theoretical frameworks.

1.1 Contributions

This dissertation makes several key contributions:

1. **Formal Definition of MPC Combiners:** We introduce the concept of MPC combiners, providing a rigorous definition that captures the security guarantees required in the presence of potentially faulty protocols. This is the first such definition in the cryptographic literature.

2. **1-out-of-2 MPC Combiner:** We propose and analyze a 1-out-of-2 MPC combiner designed to securely compute a function for two parties using two candidate MPC protocols, under the assumption that at least one of these protocols is secure. The security of the proposed combiner is rigorously examined through a series of theorems and proofs. While our findings confirm the combiner's ability to ensure security for one party, we also explore the inherent limitations in achieving full two-party security. These results align with existing cryptographic principles, highlighting the challenges and constraints in designing MPC combiners.

3. **2-out-of-3 MPC Combiner:** Building on the 1-out-of-2 combiner, we explore the construction of a 2-out-of-3 MPC combiner. This combiner aims to provide stronger security guarantees by requiring that at least two out of three candidate protocols are secure. We discuss the challenges and potential vulnerabilities of this approach and propose an alternative approach construction that, although less efficient, successfully ensures security.

1.2 Related Work

Robust combiners have been widely studied in cryptography, with significant contributions from Herzberg [8], who focused on developing efficient robust combiners for various cryptographic primitives. His research highlights the use of parallel and cascade constructions to create combiners that not only enhance efficiency but also ensure robustness. Herzberg's work includes the development of combiners for one-way functions (OWFs), digital signatures, message authentication codes (MACs), and other key cryptographic primitives.

While robust combiners have been successfully applied in several cryptographic areas, their application to secure multiparty computation (MPC) protocols remains less explored. Notably, the literature highlights the inherent challenges in constructing 1-out-of-2 Oblivious Transfer (OT) combiners, demonstrating the impossibility of such constructions under transparent black-box conditions [6]. This impossibility result, which will be further discussed in the context of the 1-out-of-2 MPC combiner presented in Section 4.3, underscores the complexity of extending robust combiners to MPC protocols and motivates the need for novel approaches in this domain.

1.3 Paper Organization

The dissertation is structured as follows:

- **Chapter 2** provides the necessary background on Secure Multi-Party Computation, including an overview of the semi-honest adversarial model and simulation-based security.
- **Chapter 3** introduces the formal security definitions and notations used throughout the dissertation, as well as the definition of black-box MPC combiners.
- **Chapter 4** details the construction and analysis of the 1-out-of-2 MPC combiner, including theorems and proofs, as well as insecurity cases and a comprehensive evaluation and discussion of the results.
- **Chapter 5** extends the discussion to the 2-out-of-3 MPC combiner, offering a detailed analysis of its design, security challenges, and an alternative secure construction.

- **Chapter 6** concludes the dissertation with a summary of the findings and a discussion of potential future research directions.

Chapter 2

Background

2.1 Secure Multi-Party Computation (MPC)

Secure Multi-Party Computation (MPC) [7] is a cryptographic framework that enables multiple parties to jointly compute a function over their inputs while maintaining the privacy of those inputs. The primary objective of MPC is to ensure that even in the presence of dishonest parties, the computation is executed correctly and no additional information about the inputs is revealed beyond what can be inferred from the output.

MPC protocols are typically designed to be resilient against various types of adversaries, ranging from semi-honest to fully malicious. In a common scenario, two parties, P_1 and P_2, wish to compute a function $f(x_1, x_2)$, where x_1 and x_2 are the private inputs of P_1 and P_2, respectively. The security of the protocol ensures that the parties learn nothing more than the output of the function, and the computation is performed as if a trusted third party were responsible for computing the function and then sharing the result with the parties.

In this paper, we will mainly focus on the semi-honest model. In the semi-honest model, also known as the honest-but-curious model, parties are assumed to follow the protocol correctly but may try to learn additional information by analyzing the messages they receive during the protocol's execution. Despite this, a secure MPC protocol guarantees that no information other than the intended output can be inferred from the execution of the protocol.

2.2 Semi-Honest Adversaries

A semi-honest adversary [7] is a type of adversary in the context of secure computation protocols. This model assumes that while the adversary adheres to the prescribed protocol correctly, it may attempt to gain as much information as possible by analyzing the data it receives during the execution of the protocol.

The semi-honest model is less stringent than the malicious model, where adversaries can deviate from the protocol in any arbitrary way. However, protocols that are secure against semi-honest adversaries are often more efficient and simpler to construct.

In the context of MPC, security against semi-honest adversaries is typically defined using a simulation-based approach. The idea is to demonstrate that whatever a semi-honest adversary can learn during the real execution of the protocol, it could also learn by participating in an idealized version of the protocol, where the computation is performed by a trusted party. If the adversary cannot distinguish between these two scenarios, the protocol is considered secure against semi-honest adversaries.

This model is particularly relevant when designing and analyzing protocols, as it allows for the construction of efficient solutions while providing a reasonable level of security. The results from such protocols can often be extended or adapted to handle more complex settings or stronger adversarial models.

2.3 Simulation-based Security

Simulation-based security [9] is a rigorous and widely used framework in cryptography for defining and proving the security of cryptographic protocols, including Secure Multi-Party Computation (MPC). The core idea behind simulation-based security is to compare the real execution of a protocol with an ideal execution, where the latter is assumed to be completely secure. The protocol is considered secure if an adversary cannot distinguish between these two executions.

In the ideal model, a trusted third party is assumed to exist. This third party receives the inputs from all participants, computes the function on those inputs, and then returns the correct outputs to the participants without revealing any additional information. Furthermore, it is assumed that the communication between the participants and the trusted third party is perfectly secure, meaning that all messages are confidential and protected from interception or tampering by adversaries. In contrast, the real model involves the actual protocol where no such trusted party exists, and the computation

is carried out by the participants themselves, without any assumptions about secure communication beyond what the protocol itself ensures.

To establish security, a ”simulator” is constructed in the simulation-based framework. The simulator aims to produce a view of the protocol's execution that is indistinguishable from what the adversary would see in the real execution. The key point is that the simulator operates without access to the honest parties' inputs, relying only on the output and any public information. If the simulator can convincingly replicate the adversary's view of the protocol, then the protocol is deemed secure.

Simulation-based security is particularly powerful because it offers a clear and strong guarantee: anything that an adversary can learn from interacting with the real protocol could have been learned in the ideal setting as well. This ensures that the protocol does not leak any unintended information, beyond what is inherent in the function being computed.

This approach to security is especially important in the context of MPC, where multiple parties must collaborate to compute a function securely despite potential adversaries. By proving that a protocol satisfies simulation-based security, one can ensure that even under adversarial conditions, the protocol behaves as though it were running in an ideal, perfectly secure environment.

In summary, simulation-based security provides a formal and robust framework for analyzing and proving the security of cryptographic protocols. By comparing the real execution of a protocol with an idealized version, it ensures that no additional information is leaked to adversaries, thus maintaining the confidentiality and integrity of the computation.

Chapter 3

Security Definitions and Notations

3.1 Notation

In this paper, we mainly focus on round-based secure MPC protocols. Without loss of generality, we assume that any protocol Π_i runs for K rounds. Rather than viewing a protocol Π_i as an n-tuple of interactive Turing machines, it is convenient to view each Turing machine as a sequence of multiple algorithms: `frst-msg`, `nxt-msg`, and `output` [3]. Specifically, each protocol Π_i can be defined as a collection of algorithms $\{\texttt{frst-msg}_j^{\Pi_i}, \texttt{nxt-msg}_j^{k,\Pi_i}, \texttt{output}_j^{\Pi_i}\}_{j\in[n],k\in[K]}$, where:

- `frst-msg` computes the first messages that each party sends to its peers.
- `nxt-msg` computes the messages for subsequent rounds.
- `output` computes the final output of each party.
- $[n]$: This denotes the set of integers $\{1,2,\ldots,n\}$, where n represents the total number of parties in the protocol.

Each protocol Π_i is capable of computing any valid function. The syntax and the specific functionalities of these algorithms are defined as follows:

- $\texttt{frst-msg}_j^{\Pi_i}(x_j;r_j)$: This algorithm computes P_j's first message to its peers, where Π_i denotes the protocol used for the MPC computation.
  - **Inputs:**
    - x_j: The private input of party P_j.

  - $*$ r_j: The internal randomness generated by party P_j at the start of the protocol.
- **– Outputs:**
  - $*$ $\texttt{msg}_{j\to l}^{1,\Pi_i}$: The message to be sent from P_j to each peer P_l in the first round.

- $\texttt{nxt-msg}_j^{k,\Pi_i}(x_j, \{\texttt{msg}_{l\to j}^{m,\Pi_i}\}_{l\in[n],m\in[k]}; r_j)$: This algorithm computes P_j's $(k+1)$-th round messages, where Π_i denotes the protocol used for the MPC computation.
  - **– Inputs:**
    - $*$ x_j: The private input of party P_j.
    - $*$ $\{\texttt{msg}_{l\to j}^{m,\Pi_i}\}_{l\in[n],m\in[k]}$: A set of messages received by P_j from its peers up to round k.
    - $*$ r_j: The internal randomness of party P_j is refreshed for each round, meaning that a new random value r_i is generated and used as input for each algorithm in every round.
  - **– Outputs:**
    - $*$ $\texttt{msg}_{j\to l}^{k+1,\Pi_i}$: The message to be sent from P_j to each peer P_l in the $(k+1)$-th round.

- $\texttt{output}_j^{\Pi_i}(x_j, \{\texttt{msg}_{l\to j}^{m,\Pi_i}\}_{l\in[n],m\in[K]}; r_j)$: This algorithm computes P_j's final output after K rounds, where Π_i denotes the protocol used for the MPC computation.
  - **– Inputs:**
    - $*$ x_j: The private input of party P_j.
    - $*$ $\{\texttt{msg}_{l\to j}^{m,\Pi_i}\}_{l\in[n],m\in[K]}$: A set of messages received by P_j from its peers over K rounds.
    - $*$ r_j: The internal randomness generated by party P_j.
  - **– Outputs:**
    - $*$ y_j: The final computed output of party P_j after K rounds of communication.

3.2 Definitions for Security

Computational Indistinguishability. A probability ensemble $X = \{X(a,n)\}_{a\in\{0,1\}^*,n\in\mathbb{N}}$ is an infinite sequence of random variables indexed by $a \in \{0,1\}^*$ and $n \in \mathbb{N}$. In the

context of secure computation, the value a will represent the parties' inputs, and n will represent the security parameter. Two probability ensembles $X = \{X(a,n)\}_{a\in\{0,1\}^*,n\in\mathbb{N}}$ and $Y = \{Y(a,n)\}_{a\in\{0,1\}^*,n\in\mathbb{N}}$ are said to be computationally indistinguishable [9], denoted by $X \stackrel{c}{\equiv} Y$, if for every non-uniform polynomial-time algorithm D, there exists a negligible function $\mu(\cdot)$ such that for every $a \in \{0,1\}^*$ and every $n \in \mathbb{N}$,

$$|\Pr[D(X(a,n)) = 1] - \Pr[D(Y(a,n)) = 1]| \leq \mu(n).$$

Two-Party Computation. A two-party protocol problem is defined by specifying a possibly random process that maps pairs of inputs to pairs of outputs (one for each party). This process is referred to as a functionality and is denoted by $f : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^* \times \{0,1\}^*$, where $f = (f_1, f_2)$. For every pair of inputs $x, y \in \{0,1\}^n$, the output-pair is a random variable $(f_1(x,y), f_2(x,y))$ ranging over pairs of strings. The first party (with input x) wishes to obtain $f_1(x,y)$, and the second party (with input y) wishes to obtain $f_2(x,y)$ [7].

Definition of Security. We begin with the following notation [5]:

- Let $f = (f_1, f_2)$ be a probabilistic polynomial-time functionality and let Π be a two-party protocol for computing f.
- The view of the i-th party ($i \in \{1,2\}$) during an execution of Π on (x_1,x_2) and security parameter n is denoted by $\text{view}_i^{\Pi}(x_1,x_2,n)$ and equals $(w, r_i; \{\mathtt{msg}_{l\to i}^{m,\Pi}\}_{l\in[2],m\in[K]})$, where $w \in \{x_1,x_2\}$ (its input depending on the value of i), r_i equals the contents of the i-th party's internal random tape, and $\{\mathtt{msg}_{l\to i}^{m,\Pi}\}_{l\in[2],m\in[K]}$ represents all the messages that it received (K is the total round number of the protocol).
- The output of the i-th party during an execution of Π on (x_1,x_2) and security parameter n is denoted by $\text{output}_i^{\Pi}(x_1,x_2,n)$ and can be computed from its own view of the execution. We denote the joint output of both parties by $\text{output}^{\Pi}(x_1,x_2,n) = (\text{output}_1^{\Pi}(x_1,x_2,n), \text{output}_2^{\Pi}(x_1,x_2,n))$.

Definition 1. *Let $f = (f_1, f_2)$ be a functionality. We say that Π securely computes f in the presence of static semi-honest adversaries if there exist probabilistic polynomial-time algorithms S_1 and S_2 such that [7]:*

$$\{(S_1(1^n, x_1, f_1(x_1,x_2)), f(x_1,x_2))\}_{x_1,x_2,n} \stackrel{c}{\equiv} \{(view_1^{\Pi}(x_1,x_2,n), output_1^{\Pi}(x_1,x_2,n))\}_{x_1,x_2,n}$$

$$\{(S_2(1^n, x_2, f_2(x_1,x_2)), f(x_1,x_2))\}_{x_1,x_2,n} \stackrel{c}{\equiv} \{(view_2^{\Pi}(x_1,x_2,n), output_2^{\Pi}(x_1,x_2,n))\}_{x_1,x_2,n}$$

where $x_1, x_2 \in \{0,1\}^*$ such that $|x_1| = |x_2|$, and $n \in \mathbb{N}$.

Definition 2. *Let $f = (f_1, f_2)$ be a functionality. We say that Π is faulty in the presence of static semi-honest adversaries if there exist probabilistic polynomial-time algorithms S_1 and/or S_2 such that:*

$$\{(S_1(1^n, x_1, f_1(x_1,x_2)), f(x_1,x_2))\}_{x_1,x_2,n} \stackrel{c}{\not\equiv} \{(view_1^{\Pi}(x_1,x_2,n), output_1^{\Pi}(x_1,x_2,n))\}_{x_1,x_2,n}$$

$$\{(S_2(1^n, x_2, f_2(x_1,x_2)), f(x_1,x_2))\}_{x_1,x_2,n} \stackrel{c}{\not\equiv} \{(view_2^{\Pi}(x_1,x_2,n), output_2^{\Pi}(x_1,x_2,n))\}_{x_1,x_2,n}$$

where $x_1, x_2 \in \{0,1\}^*$ such that $|x_1| = |x_2|$, and $n \in \mathbb{N}$.

Definition 3 (Output Round Correctness for ℓ-Round MPC Protocols)**.** *Let Π be an ℓ-round secure multiparty computation (MPC) protocol with parties $P_1, P_2, \ldots, P_n$. For any $\lambda, m \in \mathbb{N}$, for any inputs $(x_1, \ldots, x_n) \in (\{0,1\}^{\lambda})^n$, and for any set of functions $\{f_\gamma\}_{\gamma \in [n]}$ with $|f_\gamma| = m$ for all $\gamma \in [n]$, it must hold for all $i \in [n]$ that [4]:*

- *if $f_1 = \cdots = f_n$ then*

  $$\Pr\left[output_i^{\Pi}(x_i, \{msg_{1\to i}^1, \ldots, msg_{n\to i}^1, \ldots, msg_{1\to i}^j, \ldots, msg_{n\to i}^j\}, r_i) \neq f(x_1, \ldots, x_n)\right] = 0$$

  where $output_i^{\Pi}(x_i, \{msg_{1\to i}^1, \ldots, msg_{n\to i}^1, \ldots, msg_{1\to i}^j, \ldots, msg_{n\to i}^j\}, r_i)$ denotes the final output algorithm of party P_i at the end of the protocol.

- *if there exists $\alpha, \beta \in [n]$ such that $f_\alpha \neq f_\beta$, then*

  $$\Pr\left[output_i^{\Pi}(x_i, \{msg_{1\to i}^1, \ldots, msg_{n\to i}^1, \ldots, msg_{1\to i}^j, \ldots, msg_{n\to i}^j\}, r_i) \neq \perp\right] = 0$$

  where $\perp$ denotes an error output.

3.3 Black-box MPC Combiners

In this paper, we introduce the concept of a black-box MPC combiner. A black-box combiner is a method where the internal workings of the protocols being combined are not analyzed or modified; instead, they are treated as black boxes, and their outputs are combined to achieve security. In our setting, we can directly call the algorithms of the candidate protocols as subroutines without needing to understand or alter their internal processes.

Definition 4 (T-out-of-N MPC Combiner)**.** *A T-out-of-N MPC combiner is a construction that takes as input N candidate MPC protocols and produces a new protocol Π with the following properties [6]:*

1. *If at least T out of the N candidate protocols are secure, then the resulting protocol Π is secure.*

2. *The combiner operates without knowledge of which candidate protocols are faulty.*

Definition 5 (Black-Box MPC Combiner)**.** *A T-out-of-N MPC Combiner is said to be a black-box combiner if the following conditions hold [6]:*

Black-box implementation*: The combiner constructs the new MPC protocol by accessing the N candidate protocols via oracle calls to their implementation algorithms.*

The concept of black-box combiners is significant in cryptographic constructions because it allows for the secure combination of multiple protocols without the need to understand or modify the internal details of each protocol. This black-box approach is particularly useful when dealing with protocols of varying security assumptions or when the internal workings of the protocols are complex or unknown.

In summary, this chapter has established the foundational notations and definitions essential for understanding and analyzing secure MPC protocols, as well as formally defining black-box MPC combiners. Building on this foundation, the next chapter will propose a 1-out-of-2 MPC combiner and analyze its security properties.

Chapter 4

1-out-of-2 MPC Combiner

We consider the problem of securely computing a function $f(x_1, x_2)$ for two parties P_1 and P_2, using two MPC protocols, Π_1 and Π_2, at least one of which is secure. Our goal is to ensure security by combining these protocols using a black-box combiner.

In this chapter, we will explore the construction and analysis of a 1-out-of-2 MPC combiner. We will present the proposed protocol, analyze its security through formal theorems, and discuss its limitations within the context of established cryptographic principles.

4.1 Proposed Protocol

Consider two MPC protocols Π_1 and Π_2, where at least one of them is secure. Both protocols are assumed to have output round correctness. Our objective is to securely compute the function $f(x_1, x_2)$ by combining these two protocols. Assume that protocol Π_1 computes the function $f(x_1, x_2)$, while protocol Π_2 computes the function $g(x_2, \{\mathtt{msg}_{j\to 2}^{m,\Pi_1}\}_{j\in[1],m\in[k]}, r_2, k)$, where the function g returns the following function:

$$\mathtt{nxt\text{-}msg}_2^{k,\Pi_1}(x_2, \{\mathtt{msg}_{j\to 2}^{m,\Pi_1}\}_{j\in[1],m\in[k]}; r_2)$$

We propose a combined protocol Π to compute the function f as follows, assumes that P_1 speaks first:

- In the first round of Π, P_1 calls the $\mathtt{frst\text{-}msg}_1^{\Pi_1}(x_1; r_1)$ function using the protocol Π_1 with input x_1 and internal randomness r_1:

$$\mathtt{msg}_{1\to 2}^{1,\Pi_1} \leftarrow \mathtt{frst\text{-}msg}_1^{\Pi_1}(x_1; r_1)$$

Instead of sending it to P_2, P_1 initiates another MPC with P_2, using protocol Π_2 to compute the function

$$g(x_2, \{\mathtt{msg}_{j\to 2}^{1,\Pi_1}\}_{j\in[1]}, r_2, 1).$$

The input of P_1 is $\mathtt{msg}_{1\to 2}^{1,\Pi_1}$, and the inputs for P_2 are x_2 and its internal randomness r_2. P_1 and P_2 executes all the round of Π_2 and only P_1 gets the output message

$$\mathtt{msg}_{2\to 1}^{2,\Pi_1}.$$

- In the subsequent rounds ($3 \leq k < K$), P_1 calls the function $\mathtt{nxt\text{-}msg}_1^{k-1,\Pi_1}(x_1, \{\mathtt{msg}_{j\to 1}^{m,\Pi_1}\}_{j\in[2],m\in[k-1]}; r_1)$ using protocol Π_1:

$$\mathtt{msg}_{1\to 2}^{k,\Pi_1} \leftarrow \mathtt{nxt\text{-}msg}_1^{k-1,\Pi_1}(x_1, \{\mathtt{msg}_{j\to 1}^{m,\Pi_1}\}_{j\in[2],m\in[k-1]}; r_1)$$

  Then, P_1 initiates another MPC with P_2 using protocol Π_2 to compute the function

$$g(x_2, \{\mathtt{msg}_{j\to 2}^{m,\Pi_1}\}_{j\in[1],m\in[k]}, r_2, k).$$

  The input for P_1 is $\{\mathtt{msg}_{j\to 2}^{m,\Pi_1}\}_{j\in[1],m\in[k]}$, and the inputs for P_2 are x_2 and its internal randomness r_2. P_1 and P_2 executes all the round of Π_2 and only P_1 gets the output message

$$\mathtt{msg}_{2\to 1}^{k+1,\Pi_1}.$$

- In the final round, P_1 computes its output using the `output` function:

$$y_1 \leftarrow \mathtt{output}_1^{\Pi_1}(x_1, \{\mathtt{msg}_{j\to 1}^{m,\Pi_1}\}_{j\in[2],m\in[K]}; r_1)$$

  Then, P_1 initiates another MPC with P_2 using protocol Π_2 to compute the function

$$\mathtt{output}_2^{\Pi_1}(x_2, \{\mathtt{msg}_{j\to 2}^{m,\Pi_1}\}_{j\in[2],m\in[K]}; r_2).$$

  The input for P_1 is $\{\mathtt{msg}_{j\to 2}^{m,\Pi_1}\}_{j\in[2],m\in[k]}$, and the inputs for P_2 are x_2 and its internal randomness r_2. P_1 and P_2 executes all the round of Π_2 and P_2 gets the message

$$y_2.$$

4.2 Theorems and Proofs

4.2.1 Theorem 1

Theorem 1. *Given two MPC protocols Π_1 and Π_2, where Π_1 is secure and Π_2 is faulty, if P_1 is honest and P_2 is corrupted, then the combined protocol Π securely computes*

the function $f(x_1, x_2)$ for P_1. Specifically, the view of P_2 in the real execution of Π is computationally indistinguishable from the view generated by the simulator S:

$$\{(S(1^n, x_2, f_2(x_1, x_2)), f(x_1, x_2))\}_{x_1, x_2, n} \stackrel{c}{\equiv} \{(\text{view}_2^{\Pi}(x_1, x_2, n), \text{output}_2^{\Pi}(x_1, x_2, n))\}_{x_1, x_2, n}$$

Proof. We prove the security of the combined protocol Π under the assumption that Π_1 is secure and Π_2 is faulty. Suppose P_1 is honest and P_2 is corrupted. We will define simulators for Π_1 and Π and show that any adversary A that can distinguish the real and simulated views of P_2 in Π can be used to construct an adversary A' that can distinguish the real and simulated views of P_2 in Π_1, thus contradicting the security of Π_1.

Simulator for Π_1

Since Π_1 is secure, there exists a simulator S_{Π_1} that can simulate the view of P_2 such that the simulated view is indistinguishable from the real view. The simulator S_{Π_1} works by inputting an arbitrary value of length n (1^n) instead of x_1 while keeping x_2 unchanged:

$$\{(S_{\Pi_1}(1^n, x_2, f_2(x_1, x_2)), f(x_1, x_2))\}_{x_1, x_2, n} \stackrel{c}{\equiv} \left\{(\text{view}_2^{\Pi_1}(x_1, x_2, n), \text{output}_2^{\Pi_1}(x_1, x_2, n))\right\}_{x_1, x_2, n}$$

Simulator S for Π

The simulator S for the combined protocol Π invokes the simulator S_{Π_1} to generate the entire view that the corrupted party P_2 would observe in the simulation of Π_1. The simulated messages and outputs generated by S_{Π_1} include:

- $\texttt{msg}_{1\to 2}^{1,\Pi_1}$ for the first round,
- $\texttt{msg}_{1\to 2}^{k,\Pi_1}$ for subsequent rounds, where $1 < k < K$,
- The final output y_1.

Here, K represents the total number of rounds in protocol Π_1. These messages, collectively produced by the simulator S_{Π_1}, are then used in the subsequent simulation steps for protocol Π. The simulation for Π proceeds as follows:

- In the first round of Π, P_1 uses the message generated by S_{Π_1}:

$$\texttt{msg}_{1\to 2}^{1,\Pi_1}$$

Instead of sending it to P_2, P_1 initiates another MPC with P_2, using protocol Π_2 to compute the function

$$g(x_2, \{\mathtt{msg}_{1\to 2}^{1,\Pi_1}\}, r_2, 1).$$

The input of P_1 is $\mathtt{msg}_{1\to 2}^{1,\Pi_1}$, and the inputs for P_2 are x_2 and its internal randomness r_2. P_1 and P_2 execute all the rounds of Π_2 and P_1 gets the message

$$\mathtt{msg}_{2\to 1}^{2,\Pi_1}.$$

- In the subsequent rounds ($3 \leq k < K$), P_1 uses the message generated by S_{Π_1}:

$$\mathtt{msg}_{1\to 2}^{k,\Pi_1}$$

Instead of sending it to P_2, P_1 initiates another MPC with P_2, using protocol Π_2 to compute the function

$$g(x_2, \{\mathtt{msg}_{1\to 2}^{m,\Pi_1}\}_{m\in[k]}, r_2, k).$$

The input of P_1 is $\mathtt{msg}_{1\to 2}^{k,\Pi_1}$, and the inputs for P_2 are x_2 and its internal randomness r_2. P_1 and P_2 execute all the rounds of Π_2 and P_1 gets the message

$$\mathtt{msg}_{2\to 1}^{k+1,\Pi_1}.$$

- In the final round, P_1 uses the final output y_1 generated by S_{Π_1}. Then, P_1 initiates another MPC with P_2 using protocol Π_2 to compute the function

$$\mathtt{output}_2^{\Pi_1}(x_2, \{\mathtt{msg}_{j\to 2}^{m,\Pi_1}\}_{j\in[2],m\in[K]}; r_2).$$

The input for P_1 is $\{\mathtt{msg}_{j\to 2}^{m,\Pi_1}\}_{j\in[2],m\in[k]}$, and the inputs for P_2 are x_2 and its internal randomness r_2. P_1 and P_2 execute all the rounds of Π_2 and P_2 gets the output y_2 for P_2.

Reduction Proof

Assume there exists an adversary A that can distinguish the real and simulated views of P_2 in the combined protocol Π. We construct an adversary A' that can distinguish the real and simulated views of P_2 in Π_1, thereby contradicting the security of Π_1. We will demonstrate this by comparing the distributions generated by the real protocol and the simulator, and how these relate to the distributions in Π_1.

1. **Distributions for A in Protocol Π:**

   - **Real View**: The real view of P_2 in Π:

     $$\text{Real}_2^{\Pi}(x_1,x_2,n) = \left(x_2, r_2, \{\texttt{msg}_{1\to 2}^{k_1,\Pi_1}\}_{k_1\in[K]}, \{\texttt{msg}_{1\to 2}^{k_1,k_2,\Pi_2}, \texttt{msg}_{2\to 1}^{k_1,k_2,\Pi_2}\}_{k_1,k_2\in[K]}\right)$$

   - **Simulated View**: The simulated view of P_2 in Π generated by S:

     $$\text{Sim}_2^{\Pi}(1^n,x_2,n) = \left(x_2, r_2, \{\texttt{msg}_{1\to 2}^{k_1,\Pi_1}\}_{k_1\in[K]}, \{\texttt{msg}_{1\to 2}^{k_1,k_2,\Pi_2}, \texttt{msg}_{2\to 1}^{k_1,k_2,\Pi_2}\}_{k_1,k_2\in[K]}\right)$$

2. **Distributions for A' in Protocol Π_1:**

   - **Real View**: The real view of P_2 in Π_1:

     $$\text{Real}_2^{\Pi_1}(x_1,x_2,n) = \left(x_2, r_2, \{\texttt{msg}_{1\to 2}^{k,\Pi_1}\}_{k\in[K]}, \{\texttt{msg}_{2\to 1}^{k,\Pi_1}\}_{k\in[K]}\right)$$

   - **Simulated View**: The simulated view of P_2 in Π_1 generated by S_{Π_1}:

     $$\text{Sim}_2^{\Pi_1}(1^n,x_2,n) = \left(x_2, r_2, \{\texttt{msg}_{1\to 2}^{k,\Pi_1}\}_{k\in[K]}, \{\texttt{msg}_{2\to 1}^{k,\Pi_1}\}_{k\in[K]}\right)$$

Given that S calls S_{Π_1} as a subroutine, the simulated messages in S depend on the simulated messages in S_{Π_1}. We assume A can distinguish $\text{Real}_2^{\Pi}(x_1,x_2,n)$ from $\text{Sim}_2^{\Pi}(1^n,x_2,n)$. This implies:

$$\left\{\left(\text{Real}_2^{\Pi}(x_1,x_2,n), \text{output}_2^{\Pi}(x_1,x_2,n)\right)\right\}_{x_1,x_2,n} \stackrel{c}{\not\equiv} \left\{\left(\text{Sim}_2^{\Pi}(1^n,x_2,n), f(x_1,x_2)\right)\right\}_{x_1,x_2,n}$$

We can argue that the distribution of $\text{Real}_2^{\Pi_1}(x_1,x_2,n)$ and its output is equal to the distribution of $\text{Real}_2^{\Pi}(x_1,x_2,n)$ and its output, because the internal messages from Π_2 do not affect the distribution:

$$\left\{\left(\text{Real}_2^{\Pi_1}(x_1,x_2,n), \text{output}_2^{\Pi_1}(x_1,x_2,n)\right)\right\}_{x_1,x_2,n} = \left\{\left(\text{Real}_2^{\Pi}(x_1,x_2,n), \text{output}_2^{\Pi}(x_1,x_2,n)\right)\right\}_{x_1,x_2,n}$$

Similarly, the distribution of $\text{Sim}_2^{\Pi_1}(1^n,x_2,n)$ and its output is equal to the distribution of $\text{Sim}_2^{\Pi}(1^n,x_2,n)$ and its output:

$$\left\{\left(\text{Sim}_2^{\Pi_1}(1^n,x_2,n), f(x_1,x_2)\right)\right\}_{x_1,x_2,n} = \left\{\left(\text{Sim}_2^{\Pi}(1^n,x_2,n), f(x_1,x_2)\right)\right\}_{x_1,x_2,n}$$

Therefore, we have

$$\left\{\left(\text{Real}_2^{\Pi_1}(x_1,x_2,n), \text{output}_2^{\Pi_1}(x_1,x_2,n)\right)\right\}_{x_1,x_2,n} \stackrel{c}{\not\equiv} \left\{\left(\text{Sim}_2^{\Pi_1}(1^n,x_2,n), f(x_1,x_2)\right)\right\}_{x_1,x_2,n}$$

If A can distinguish the real and simulated views of P_2 in Π, then A' can use A to distinguish the views in Π_1, leading to a contradiction because Π_1 is assumed to be secure. Consequently, A cannot exist, and the simulated view for P_2 in Π must be indistinguishable from the real view. Thus, the combined protocol Π securely computes the function $f(x_1,x_2)$ for P_1 when P_1 is honest and P_2 is corrupted.

□

4.2.2 Theorem 2

Theorem 2. *Given two MPC protocols Π_1 and Π_2, where Π_1 is faulty and Π_2 is secure, if P_1 is honest and P_2 is corrupted, then the combined protocol Π securely computes the function $f(x_1,x_2)$ for P_1. Specifically, the view of P_2 in the real execution of Π is computationally indistinguishable from the view generated by the simulator S:*

$$\{(S(1^n,x_2,f_2(x_1,x_2)),f(x_1,x_2))\}_{x_1,x_2,n} \stackrel{c}{\equiv} \left\{(\text{view}_2^{\Pi}(x_1,x_2,n),\text{output}_2^{\Pi}(x_1,x_2,n))\right\}_{x_1,x_2,n}$$

Proof. We prove the security of the combined protocol Π under the assumption that Π_1 is faulty and Π_2 is secure. Suppose P_1 is honest and P_2 is corrupted. We will define simulators for Π_2 and Π and propose a sequence of hybrid experiments H_n to analyze the security of the combined protocol Π.

Simulator for Π_2

Since Π_2 is secure, there exists a simulator S_{Π_2} that can simulate the view of the corrupted party P_2 such that the simulated view is indistinguishable from the real view. In the combined protocol, Π_2 is called multiple times to compute the following fuction:

$$g(x_2,\{\mathtt{msg}_{j\to 2}^{m,\Pi_1}\}_{j\in[1],m\in[k]},r_2,k)$$

The simulator S_{Π_2} works by simulating the inputs for both P_1 and P_2 in the following way:

- Input for P_1: The simulator inputs arbitrary values as placeholders for the messages $\{\mathtt{msg}_{1\to 2}^{m,\Pi_1}\}_{m\in[k]}$ that would have been generated by Π_1. These dummy inputs have the same length and structure as the real messages, ensuring consistency in the simulation.
- Input for P_2: The simulator uses the real input x_2 and internal randomness r_2 of P_2, as these values are known to the corrupted party.

The simulator S_{Π_2} generates the messages and outputs in such a way that they are computationally indistinguishable from those in the real protocol execution.

$$\left\{(S_{\Pi_2}(\{1^n\}_{m\in[k]},x_2),g(\{\mathtt{msg}_{1\to 2}^{m,\Pi_1}\}_{m\in[k]},x_2,r_2,k))\right\}_{\{\mathtt{msg}_{1\to 2}^{m,\Pi_1}\}_{m\in[k]},x_2,n} \stackrel{c}{\equiv}$$

$$\left\{(\text{view}_2^{\Pi_2}(\{\mathtt{msg}_{1\to 2}^{m,\Pi_1}\}_{m\in[k]},x_2,n),\text{output}_2^{\Pi_2}(\{\mathtt{msg}_{1\to 2}^{m,\Pi_1}\}_{m\in[k]},x_2,n))\right\}_{\{\mathtt{msg}_{1\to 2}^{m,\Pi_1}\}_{m\in[k]},x_2,n}$$

Simulator S for Π

The simulator S for the combined protocol Π operates as follows. It uses the real executions of Π_1 throughout, and replaces all invocations of Π_2 with the simulator S_{Π_2}. Note that $\bar{r}_1$ represents a random value chosen by the simulator to simulate P_1's internal randomness r_1:

- First Round:
  - Real Execution of Π_1: P_1 computes the first message using the real protocol Π_1 with arbitrary input 1^n:

$$\mathtt{msg}_{1\to 2}^{1,\Pi_1} \leftarrow \mathtt{frst\text{-}msg}_1^{\Pi_1}(1^n;\bar{r}_1)$$

  - Simulation of Π_2 with S_{Π_2}: The simulator S invokes S_{Π_2} with variable $k=1$ to generate the view and output message in this round. The output message $\mathtt{msg}_{2\to 1}^{2,\Pi_2}$ is provided to P_1.
- Subsequent Rounds ($3 \leq k < K$):
  - Real Execution of Π_1: For each subsequent round, P_1 computes the next message using the real protocol Π_1:

$$\mathtt{msg}_{1\to 2}^{k,\Pi_1} \leftarrow \mathtt{nxt\text{-}msg}_1^{k-1,\Pi_1}(1^n,\{\mathtt{msg}_{j\to 1}^{m,\Pi_1}\}_{j\in[2],m\in[k-1]};\bar{r}_1)$$

  - Simulation of Π_2 with S_{Π_2}: The simulator S invokes S_{Π_2} with variable k to generate the view and output message in each of these rounds. The output message $\mathtt{msg}_{2\to 1}^{k+1,\Pi_2}$ is provided to P_1.
- Final Round:
  - Real Execution of Π_1: P_1 computes its output using the real protocol Π_1:

$$y_1 \leftarrow \mathtt{output}_1^{\Pi_1}(1^n,\{\mathtt{msg}_{j\to 1}^{m,\Pi_1}\}_{j\in[2],m\in[K]};\bar{r}_1)$$

  - Simulation of Π_2 with S_{Π_2}: The simulator S invokes S_{Π_2} with variable $k=K$ to generate the final view and output message. The final output message y_2 is provided to P_2.

Proof

We propose a sequence of hybrid experiments H_n to analyze the security of the combined protocol Π. The experiments transition from the real execution of the protocol to the full simulation performed by the simulator S. The sequence is described as follows:

- H_0**:** The experiment H_0 represents the real execution of the protocol Π, where both Π_1 and Π_2 are executed according to their specifications without any simulation. Specifically:

  $$H_0 : \text{Real execution of } \Pi_1 \text{ and } \Pi_2 \text{ for all rounds.}$$

- H_1**:** The experiment H_1 modifies the real execution by simulating the first invocation of Π_2 using the simulator S_{Π_2}. The remaining invocations of Π_2 are executed according to the real protocol. Specifically:

  $H_1 :$ Simulate the first round of Π_2 using S_{Π_2}, real execution for subsequent rounds.

- H_n**:** The experiment H_n extends this approach by simulating the first n invocations of Π_2 using S_{Π_2}, while the remaining invocations of Π_2 are executed according to the real protocol. Specifically:

  $H_n :$ Simulate the first n rounds of Π_2 using S_{Π_2}, real execution for subsequent rounds.

- H_K**:** The final experiment H_K represents the scenario where all invocations of Π_2 are simulated using S_{Π_2}. This corresponds to the full simulation as performed by the simulator S for the protocol Π. Specifically:

  $H_K :$ Simulate all K rounds of Π_2 using S_{Π_2}, corresponding to the simulator S.

We want to prove that $H_i \overset{c}{\equiv} H_{i+1}$ for all $0 \leq i < K$, and therefore $H_0 \overset{c}{\equiv} H_K$. To show that H_i is computationally indistinguishable from H_{i+1}, we consider the difference between the two experiments:

- H_i**:** The first i rounds of Π_2 are simulated using S_{Π_2}, and the remaining $K-i$ rounds are executed according to the real protocol Π_2.

- H_{i+1}**:** The first $i+1$ rounds of Π_2 are simulated using S_{Π_2}, while the remaining $K-(i+1)$ rounds are executed according to the real protocol Π_2.

The only difference between H_i and H_{i+1} is in the simulation of the $(i+1)$-th round of Π_2:

- In H_i, the $(i+1)$-th round of Π_2 is executed according to the real protocol.
- In H_{i+1}, the $(i+1)$-th round of Π_2 is simulated using S_{Π_2}.

Since Π_2 is a secure protocol, by the definition of security in the presence of static semi-honest adversaries, the view of the corrupted party P_2 when the $(i+1)$-th round of Π_2 is simulated using S_{Π_2} is computationally indistinguishable from the view when it is executed according to the real protocol. Formally, we have:

$$\{(S_{\Pi_2}(1^n, x_2, f_2(x_1,x_2)), f(x_1,x_2))\}_{x_1,x_2,n} \stackrel{c}{\equiv} \left\{(\text{view}_2^{\Pi_2}(x_1,x_2,n), \text{output}_2^{\Pi_2}(x_1,x_2,n))\right\}_{x_1,x_2,n}$$

Therefore, H_i and H_{i+1} are computationally indistinguishable:

$$H_i \stackrel{c}{\equiv} H_{i+1}$$

Given that $H_i \stackrel{c}{\equiv} H_{i+1}$ for all $0 \leq i < K$, by the transitivity of computational indistinguishability, we have:

$$H_0 \stackrel{c}{\equiv} H_1 \stackrel{c}{\equiv} H_2 \stackrel{c}{\equiv} \ldots \stackrel{c}{\equiv} H_{K-1} \stackrel{c}{\equiv} H_K$$

Thus, we conclude that:

$$H_0 \stackrel{c}{\equiv} H_K$$

Since H_0 is the real execution of the protocol and H_K is the full simulation by the simulator S, we conclude that the view of the corrupted party in the real execution is computationally indistinguishable from the view generated by the simulator. Therefore, the combined protocol Π securely computes the function $f(x_1,x_2)$ for P_1 when Π_1 is honest and Π_2 are corrupted.

□

4.2.3 Theorem 3

Theorem 3. *Given two secure MPC protocols* Π_1 *and* Π_2*, if* P_1 *is honest and* P_2 *is corrupted, then the combined protocol* Π *securely computes the function* $f(x_1,x_2)$ *for* P_1.

Proof. Since Theorem 1 establishes security for P_1 when Π_1 is secure and Π_2 is faulty, and Theorem 2 establishes security for P_1 when Π_2 is secure and Π_1 is faulty, the current theorem, which considers the case where both Π_1 and Π_2 are secure, follows as a direct consequence. Thus, we conclude that the combined protocol Π securely computes the function $f(x_1, x_2)$ for P_1 when P_1 is honest and P_2 is corrupted, and both Π_1 and Π_2 are secure.

□

4.2.4 Theorem 4

Theorem 4. *Given two secure MPC protocols Π_1 and Π_2, if P_1 is corrupted and P_2 is honest, then the combined protocol Π securely computes the function $f(x_1, x_2)$ for P_2. Specifically, the view of P_1 in the real execution of Π is computationally indistinguishable from the view generated by the simulator S:*

$$\{(S(1^n, x_1, f_1(x_1, x_2)), f(x_1, x_2))\}_{x_1, x_2, n} \stackrel{c}{\equiv} \left\{(\text{view}_1^{\Pi}(x_1, x_2, n), \text{output}_1^{\Pi}(x_1, x_2, n))\right\}_{x_1, x_2, n}$$

Proof. We prove the security of the combined protocol Π under the assumption that both Π_1 and Π_2 are secure. Suppose P_1 is corrupted and P_2 is honest. We will define simulators for Π_1, Π_2 and Π and prove that the view of P_1 in the real execution of the protocol Π is computationally indistinguishable from the view generated by the simulator S.

Simulator for Π_1

The simulator S_{Π_1} for the protocol Π_1 constructs the entire view that the corrupted party P_1 would observe during the execution of Π_1, ensuring that this simulated view is indistinguishable from the one P_1 would obtain in a real execution of Π_1. The view consists of:

- x_1: The input provided by P_1.
- r_1: The internal randomness used by P_1 during the protocol.
- $\{\text{msg}_{2\to1}^{k,\Pi_1}\}_{k\in[K]}$: The set of all messages received by P_1 from P_2 in every round of the protocol.
- $\{\text{msg}_{1\to2}^{k,\Pi_1}\}_{k\in[K]}$: The set of all messages sent by P_1 to P_2 in every round of the protocol.
- y_1: The final output computed by P_1 at the end of the protocol.

Simulator for Π_2

Since Π_2 is secure, there exists a simulator S_{Π_2} that can simulate the view of the corrupted party P_1 such that the simulated view is indistinguishable from the real view. In the combined protocol, Π_2 is called multiple times to compute the following fuction:

$$g(x_2, \{\mathtt{msg}_{j\to 2}^{m,\Pi_1}\}_{j\in[1],m\in[k]}, r_2, k)$$

The simulator S_{Π_2} works by simulating the inputs for both P_1 and P_2 in the following way:

- Input for P_1: The simulator uses the real input for messages $\{\mathtt{msg}_{1\to 2}^{m,\Pi_1}\}_{m\in[k]}$ that would have been generated by Π_1, as these values are known to the corrupted party P_1.
- Input for P_2: The simulator inputs arbitrary values as placeholders for x_2 and internal randomness r_2 of P_2. These dummy inputs have the same length and structure as the real messages, ensuring consistency in the simulation.

The simulator S_{Π_2} generates the messages and outputs in such a way that they are computationally indistinguishable from those in the real protocol execution.

$$\left\{(S_{\Pi_2}(\{\mathtt{msg}_{1\to 2}^{m,\Pi_1}\}_{m\in[k]}, 1^n), g(\{\mathtt{msg}_{1\to 2}^{m,\Pi_1}\}_{m\in[k]}, x_2, r_2, k))\right\}_{\{\mathtt{msg}_{1\to 2}^{m,\Pi_1}\}_{m\in[k]}, x_2, n} \stackrel{c}{\equiv}$$

$$\left\{(\mathrm{view}_1^{\Pi_2}(\{\mathtt{msg}_{1\to 2}^{m,\Pi_1}\}_{m\in[k]}, x_2, n), \mathrm{output}_1^{\Pi_2}(\{\mathtt{msg}_{1\to 2}^{m,\Pi_1}\}_{m\in[k]}, x_2, n))\right\}_{\{\mathtt{msg}_{1\to 2}^{m,\Pi_1}\}_{m\in[k]}, x_2, n}$$

Simulator S for Π

The simulator S for the combined protocol Π operates by sequentially invoking the simulators S_{Π_1} and S_{Π_2} to generate the necessary messages and outputs that simulate the view of the corrupted party P_1 as follows:

- **First Round:**
  - Simulation of Π_1 with S_{Π_1}: P_1 uses the message generated by S_{Π_1}:

    $$\mathtt{msg}_{1\to 2}^{1,\Pi_1}$$

  - Simulation of Π_2 with S_{Π_2}: The simulator S invokes S_{Π_2} with $k = 1$ to generate the view and output message for the first invocation of Π_2. The output message $\mathtt{msg}_{2\to 1}^{2,\Pi_2}$ is provided to P_1.

- **Subsequent Rounds** $(3 \leq k \leq K)$**:**
  - Simulation of Π_1 with S_{Π_1}: P_1 uses the message generated by S_{Π_1}:

$$\mathtt{msg}_{1\to 2}^{k,\Pi_1}$$

  - Simulation of Π_2 with S_{Π_2}: The simulator S invokes S_{Π_2} with variable k to generate the view and output message for each invocation of Π_2. The output message $\mathtt{msg}_{2\to 1}^{k+1,\Pi_2}$ is provided to P_1.
- **Final Round:**
  - Simulation of Π_1 with $S_{\Pi_1}0$: P_1 uses the output generated by S_{Π_1}:

$$y_1$$

  - Simulation of Π_2 with S_{Π_2}: The simulator S invokes S_{Π_2} with variable $k = K$ to generate the final view and output message. The final output message y_2 is provided to P_2.

Proof

The total view of P_1 in the protocol Π is the combination of the views from each invocation of Π_2 and the messages generated by S_{Π_1} during all rounds $k_1 \in [K]$. Specifically, the full view includes:

- The input x_1.
- The internal randomness r_1.
- The messages $\{\mathtt{msg}_{2\to 1}^{k_1,\Pi_1}\}_{k_1\in[K]}$, which are outputs from each invocation of Π_2, simulated by S_{Π_2}.
- The messages generated by S_{Π_1} in the protocol Π_1, represented as:

$$\{\mathtt{msg}_{1\to 2}^{k_1,\Pi_1}\}_{k_1\in[K]}$$

- The messages exchanged in Π_2, represented as:

$$\{\mathtt{msg}_{1\to 2}^{k_1,k_2,\Pi_2}, \mathtt{msg}_{2\to 1}^{k_1,k_2,\Pi_2}\}_{k_1,k_2\in[K]}$$

Since each k_1 invocation of Π_2 is simulated by S_{Π_2} and is indistinguishable from the real execution, and the combined view of P_1 in Π includes the simulated messages from S_{Π_1}, it follows that the overall view of P_1 in the protocol Π is indistinguishable from the view generated by the simulator S. The simulator S combines the views from S_{Π_1} and S_{Π_2}, ensuring that P_1 cannot distinguish whether it is observing a real execution or a simulated one. Therefore, the combined protocol Π securely computes the function $f(x_1, x_2)$ for P_2 when both Π_1 and Π_2 are honest.

□

4.2.5 Insecurity Cases

4.2.5.1 Case 1

Given two MPC protocols Π_1 and Π_2, where Π_1 is faulty and Π_2 is secure, if P_1 is corrupted and P_2 is honest, then the combined protocol Π **does not** securely compute the function $f(x_1, x_2)$ for P_2. The reason is that the corrupted party P_1 can exploit the weaknesses in the faulty protocol Π_1 to gain unauthorized access to P_2's private input x_2. Even though Π_2 is secure on its own, the vulnerabilities in Π_1 compromise the overall security of the combined protocol Π, leading to a breach in the security guarantees for P_2.

4.2.5.2 Case 2

Given two MPC protocols Π_1 and Π_2, where Π_1 is secure and Π_2 is faulty, if P_1 is corrupted and P_2 is honest, then the combined protocol Π **does not** securely compute the function $f(x_1, x_2)$ for P_2. The reason is that the corrupted party P_1 might exploit the weaknesses in Π_2 to compromise the security of the entire combined protocol Π. In particular, P_1 could manipulate the execution of Π_2 to gain unauthorized access to P_2's private inputs, thus violating the security guarantees for P_2.

4.3 Security Evaluation and Discussion

The analysis of the proposed protocol, as established by Theorems 1, 2, and 3, demonstrates that under all conditions, the security of the function $f(x_1, x_2)$ for the honest party P_1 is consistently preserved. Specifically, even when one of the protocols, Π_1 or Π_2, is faulty, the combined protocol Π still securely computes the function for P_1.

However, the security for the honest party P_2 is more fragile, as shown in Theorems 4 and the Insecurity Cases 4.2.5.1 and 4.2.5.2. These results indicate that P_2's security is only assured when both MPC protocols, Π_1 and Π_2, are secure. If either protocol is faulty, the corrupted party P_1 can exploit these vulnerabilities to compromise P_2's private inputs, thereby violating the security guarantees. Thus, while P_1's security is maintained in all scenarios, P_2 can only be secure if both protocols are secure. In summary, on the positive side, the proposed protocol provides one-sided security by protecting the interests of P_1. However, on the negative side, it leaves P_2 vulnerable if either protocol is compromised, meaning it does not offer general security for both parties.

This observation aligns with established results in cryptographic literature [6], particularly the impossibility of constructing black-box combiners for 1-out-of-2 Oblivious Transfer (OT). Specifically, it has been shown that there are no robust "transparent black-box" combiners for OT. The idea is that if we could construct a secure OT protocol by combining two potentially insecure ones, where at least one must be secure, we would contradict the established security assumptions of OT. This impossibility result is significant because OT is a foundational building block for secure Multi-Party Computation (MPC) [10]. If it were possible to construct secure 1-out-of-2 OT combiners, we could similarly build secure 1-out-of-2 MPC combiners using the same approach. Conversely, by appropriately designing the functionality of an MPC protocol, it can be made to function as an OT protocol. Specifically, in an OT protocol, one party (the sender) has two inputs, and the other party (the receiver) chooses one of these inputs to receive, without the sender learning which one was chosen. To mimic this in MPC, we can define the function to be computed by the MPC as one where the receiver's input determines which of the sender's inputs is revealed to them, while the other input remains hidden. The secure computation ensures that the sender does not learn the receiver's choice, and the receiver learns only the selected input.

Thus, if we were able to construct secure 1-out-of-2 MPC combiners, this approach could be used to construct secure 1-out-of-2 OT combiners as well. Such a result would directly contradict the established impossibility of constructing secure 1-out-of-2 OT combiners, thereby breaking the impossibility results in the literature.

Chapter 5

2-out-of-3 MPC Combiner

We consider the problem of securely computing a function $f(x_1, x_2)$ for two parties P_1 and P_2, using three MPC protocols Π_1, Π_2, and Π_3; at least two out of these three are secure. Our goal is to ensure security by combining these protocols using a black-box combiner.

Motivated by the observation that the proposed 1-out-of-2 combiner provides one-sided security, we explore the possibility of constructing a secure 2-out-of-3 combiner by cascading 1-out-of-2 combiners in both directions, protecting both parties' inputs. This approach initially appears promising, as it suggests a potential path to achieving full security by combining the strengths of the 1-out-of-2 combiners. However, upon closer examination, we find that this protocol is flawed and fails to meet the desired security guarantees. In the following sections, we analyze the shortcomings of this approach and propose an alternative 2-out-of-3 combiner that, while potentially less efficient, addresses the identified issues and ensures security.

5.1 Proposed Protocol

Consider three MPC protocols Π_1, Π_2, and Π_3, where at least two out of the three are secure. All protocols are assumed to have output round correctness. Our objective is to securely compute the function $f(x_1, x_2)$ by combining these three protocols. Assume that protocol Π_1 computes the function $f(x_1, x_2)$, while protocol Π_2 computes the function $g_1(x_2, \{\mathtt{msg}_{j\to 2}^{m,\Pi_1}\}_{j\in[1],m\in[k]}, r_2, k)$, where the function g_1 returns the following function in computing f using Π_1:

$$\mathtt{nxt\text{-}msg}_2^{k,\Pi_1}(x_2, \{\mathtt{msg}_{j\to 2}^{m,\Pi_1}\}_{j\in[1],m\in[k]}; r_2)$$

and protocol Π_3 computes the function

$$g_2(\{\mathtt{msg}_{j\to 2}^{m,\Pi_1}\}_{j\in[1],m\in[k_1]}, \{\mathtt{msg}_{j\to 1}^{k_1,m,\Pi_2}\}_{j\in[1],m\in[k_2]}, r_2, k_1, k_2)$$

for Π_2, where the function g_2 returns the following function in computing g_1 using Π_2:

$$\mathtt{nxt\text{-}msg}_1^{k_1,k_2,\Pi_2}(\{\mathtt{msg}_{j\to 2}^{m,\Pi_1}\}_{j\in[1],m\in[k_1]}, \{\mathtt{msg}_{j\to 1}^{k_1,m,\Pi_2}\}_{j\in[1],m\in[k_2]}; r_2)$$

We propose a combined protocol Π to compute the function f as follows:

- **First Round:**
  - P_1 calls the $\mathtt{frst\text{-}msg}_1^{\Pi_1}(x_1; r_1)$ function using protocol Π_1 with input x_1 and internal randomness r_1:

    $$\mathtt{msg}_{1\to 2}^{1,\Pi_1} \leftarrow \mathtt{frst\text{-}msg}_1^{\Pi_1}(x_1; r_1)$$

  - Instead of sending it to P_2, P_1 and P_2 jointly compute the function below using protocol Π_2:

    $$g_1(x_2, \{\mathtt{msg}_{1\to 2}^{1,\Pi_1}\}, r_2, 1)$$

    * In this computation, P_2 speaks first and computes the first $\mathtt{frst\text{-}msg}$ message:

      $$\mathtt{msg}_{2\to 1}^{1,1,\Pi_2} \leftarrow \mathtt{frst\text{-}msg}_2^{\Pi_2}(x_2; r_2)$$

      Instead of sending it to P_1, P_2 initiates another MPC with P_1 using protocol Π_3 to compute the function:

      $$g_2(\{\mathtt{msg}_{1\to 2}^{1,\Pi_1}\}, \{\mathtt{msg}_{2\to 1}^{1,1,\Pi_2}\}, r_2, 1, 1)$$

      and P_2 gets the message $\mathtt{msg}_{1\to 2}^{1,2,\Pi_2}$.

    * Subsequently ($3 \le k_2 < K$), P_2 first calls the function:

      $$\mathtt{msg}_{2\to 1}^{1,k_2,\Pi_2} \leftarrow \mathtt{nxt\text{-}msg}_2^{1,k_2-1,\Pi_2}(x_2, \{\mathtt{msg}_{j\to 2}^{1,m,\Pi_2}\}_{j\in[2],m\in[k_2-1]}; r_2)$$

      Then P_2 initiates another MPC with P_1 using protocol Π_3 to compute the function:

      $$g_2(\{\mathtt{msg}_{1\to 2}^{1,\Pi_1}\}, \{\mathtt{msg}_{2\to 1}^{1,m,\Pi_2}\}_{m\in[k_2]}, r_2, 1, k_2)$$

      and P_2 gets the message $\mathtt{msg}_{1\to 2}^{1,k_2+1,\Pi_2}$.

* Finally, P_2 computes its output using the $\texttt{output}$ function:

$$\texttt{msg}_{2\to1}^{2,\Pi_1} \leftarrow \texttt{output}_2^{\Pi_2}(x_2, \{\texttt{msg}_{1\to2}^{1,m,\Pi_2}\}_{m\in[K]}; r_2)$$

and share this message with P_1.

- **Subsequent Rounds** ($3 \le k_1 < K$)**:**
  - P_1 calls the function $\texttt{nxt-msg}_1^{k_1-1,\Pi_1}(x_1, \{\texttt{msg}_{j\to1}^{m,\Pi_1}\}_{j\in[2],m\in[k_1-1]}; r_1)$ using protocol Π_1:

$$\texttt{msg}_{1\to2}^{k_1,\Pi_1} \leftarrow \texttt{nxt-msg}_1^{k_1-1,\Pi_1}(x_1, \{\texttt{msg}_{j\to1}^{m,\Pi_1}\}_{j\in[2],m\in[k_1-1]}; r_1)$$

  - Instead of sending it to P_2, P_1 and P_2 jointly compute the function below using protocol Π_2:

$$g_1(x_2, \{\texttt{msg}_{j\to2}^{m,\Pi_1}\}_{j\in[2],m\in[k_1]}, r_2, k_1)$$

    * In this computation, P_2 speaks first and calls the function:

$$\texttt{msg}_{1\to2}^{k_1,1,\Pi_2} \leftarrow \texttt{frst-msg}_2^{\Pi_2}(x_2; r_2)$$

Then P_2 initiates another MPC with P_1 using protocol Π_3 to compute the function:

$$g_2(\{\texttt{msg}_{j\to2}^{m,\Pi_1}\}_{j\in[2],m\in[k_1]}, \{\texttt{msg}_{j\to1}^{k_1,1,\Pi_2}\}_{j\in[2]}; r_2, k_1, 1)$$

and P_2 gets the message $\texttt{msg}_{1\to2}^{k_1,2,\Pi_2}$.

    * Subsequently ($3 \le k_2 < K$), P_2 calls the function:

$$\texttt{msg}_{2\to1}^{k_1,k_2,\Pi_2} \leftarrow \texttt{nxt-msg}_2^{k_1,k_2-1,\Pi_2}(x_2, \{\texttt{msg}_{j\to2}^{k_1,m,\Pi_1}\}_{j\in[2],m\in[k_2-1]}; r_2)$$

Then P_2 initiates another MPC with P_1 using protocol Π_3 to compute the function:

$$g_2(\{\texttt{msg}_{j\to2}^{m,\Pi_1}\}_{j\in[1],m\in[k_1]}, \{\texttt{msg}_{j\to1}^{k_1,m,\Pi_2}\}_{j\in[1],m\in[k_2]}, r_2, k_1, k_2)$$

and P_2 gets the message $\texttt{msg}_{1\to2}^{k_1,k_2+1,\Pi_2}$.

    * Finally, P_2 computes its output using the $\texttt{output}$ function:

$$\texttt{msg}_{2\to1}^{k_1+1,\Pi_1} \leftarrow \texttt{output}_2^{\Pi_2}(x_2, \{\texttt{msg}_{j\to2}^{k_1,m,\Pi_2}\}_{j\in[2],m\in[K]}; r_2)$$

and share this message with P_1.

- **Final Round:**
  - P_1 computes its output using the $\texttt{output}$ function:

$$y_1 \leftarrow \texttt{output}_1^{\Pi_1}(x_1, \{\texttt{msg}_{j\to 1}^{m,\Pi_1}\}_{j\in[2],m\in[K]}; r_1)$$

  - P_1 and P_2 jointly compute the function below using protocol Π_2:

$$\texttt{output}_2^{\Pi_1}(x_2, \{\texttt{msg}_{j\to 2}^{m,\Pi_1}\}_{j\in[2],m\in[K]}; r_2)$$

    * In this computation, P_2 speaks first and calls the function:

$$\texttt{msg}_{1\to 2}^{K,1,\Pi_2} \leftarrow \texttt{frst-msg}_2^{\Pi_2}(x_2; r_2)$$

      Then P_2 initiates another MPC with P_1 using protocol Π_3 to compute the function:

$$g_2(\{\texttt{msg}_{j\to 2}^{m,\Pi_1}\}_{j\in[2],m\in[K]}, \{\texttt{msg}_{j\to 1}^{K,1,\Pi_2}\}_{j\in[2]}; r_2, K, 1)$$

      and P_2 gets the message $\texttt{msg}_{1\to 2}^{K,2,\Pi_2}$.
    * Subsequently ($3 \leq k_2 < K$), P_2 calls the function:

$$\texttt{msg}_{2\to 1}^{K,k_2,\Pi_2} \leftarrow \texttt{nxt-msg}_2^{K,k_2-1,\Pi_2}(x_2, \{\texttt{msg}_{j\to 2}^{K,m,\Pi_1}\}_{j\in[2],m\in[k_2-1]}; r_2)$$

      Then P_2 initiates another MPC with P_1 using protocol Π_3 to compute the function:

$$g_2(\{\texttt{msg}_{j\to 2}^{m,\Pi_1}\}_{j\in[1],m\in[K]}, \{\texttt{msg}_{j\to 1}^{K,m,\Pi_2}\}_{j\in[1],m\in[k_2]}, r_2, K, k_2)$$

      and P_2 gets the message $\texttt{msg}_{1\to 2}^{K,k_2+1,\Pi_2}$.
    * Finally, P_2 computes its output using the $\texttt{output}$ function:

$$y_2 \leftarrow \texttt{output}_2^{\Pi_2}(x_2, \{\texttt{msg}_{j\to 2}^{K,m,\Pi_2}\}_{j\in[2],m\in[K]}; r_2)$$

5.2 Security Analysis

The proposed 2-out-of-3 combiner protocol is an extension of the 1-out-of-2 combiner, designed to enhance security by treating each intermediate MPC call as another 1-out-of-2 protocol initiated by the other party. The intuition behind this design is grounded in the security guarantee provided by the 1-out-of-2 combiner, which ensures the security of the initiating party in the protocol. By cascading this structure in both

directions—ensuring that both parties, P_1 and P_2, initiate and participate in the 1-out-of-2 combiners—the protocol appears promising in providing full security coverage for both parties' inputs.

Specifically, at each intermediate step of the MPC protocol, when computing the function

$$g_1(x_2, \{\mathtt{msg}_{j\to 2}^{m,\Pi_1}\}_{j\in[2],m\in[k_1]}, r_2, k_1),$$

we structure the computation to function as an additional 1-out-of-2 combiner applied to the remaining two protocols Π_2 and Π_3. This 1-out-of-2 combiner is initiated by P_2. Given that the original 1-out-of-2 protocol already guarantees security for P_1, the goal of this design is to extend security to P_2 as well. If successful, this would result in a secure 2-out-of-3 combiner that ensures the security of both parties.

However, upon closer examination, the protocol fails to achieve the desired security guarantees. The flaw becomes evident when P_1 is corrupted and Π_1 is faulty. In this scenario, P_1 can exploit the weakness in Π_1 by using the return messages $\{\mathtt{msg}_{2\to 1}^{k_1,\Pi_1}\}_{k_1\in[K]}$ to extract P_2's private input, x_2. This vulnerability arises because P_1 requires these messages to compute the `nxt-msg` function in the upcoming round, which appears inevitable in the current construction.

In conclusion, the proposed protocol does not provide the robust security initially intended. While the idea of cascading 1-out-of-2 combiners is promising, it fails to address the inherent vulnerability. As a consequence, we propose an alternative 2-out-of-3 combiner that, while potentially less efficient, prioritizes security and mitigates the identified flaws.

5.3 A Secure 2-out-of-3 MPC Combiner

Given the limitations of the previously proposed 2-out-of-3 combiner, we turn to a more reliable construction based on established results that demonstrate the feasibility of constructing a secure 2-out-of-3 oblivious transfer (OT) combiner [6]. This result can be leveraged to build a secure MPC combiner.

The idea is as follows: Suppose we have three candidate MPC protocols, Π_1, Π_2, and Π_3. These MPC protocols can be designed to perform OT, as shown in section 4.3. The core of the construction is to treat the three MPC protocols as candidates in a 2-out-of-3 OT combiner. We can summarize the approach as follows:

1. Each of the three candidate MPC protocols is first configured to implement an

OT protocol. This transformation allows us to apply the robust OT combiner mechanism.

2. The 2-out-of-3 OT combiner is then applied to these three OT implementations. This combiner guarantees that the combined protocol securely performs the OT as long as two out of the three OT protocols are secure.

3. Finally, since secure MPC protocols can be constructed from OT [10], the existence of a 2-out-of-3 OT combiner allows us to construct an secure MPC protocol.

While this approach is theoretically sound, it introduces significant inefficiencies. Transforming MPC protocols to perform OT, and then applying the OT combiner, results in a protocol that is less efficient than the original 1-out-of-2 combiner. Moreover, the complexity of the construction increases due to the need to secure the OT operations at each step. Although this approach sacrifices efficiency, it provides a robust security guarantee, making it a preferable choice when security is paramount.

Chapter 6

Conclusions

This dissertation introduced the first formal definition of MPC combiner and explored its construction in both the 1-out-of-2 and 2-out-of-3 scenarios, primarily within the two-party semi-honest setting. We successfully proposed and analyzed a 1-out-of-2 MPC combiner, demonstrating its effectiveness in ensuring security for one party. We also explored a 2-out-of-3 combiner, which, while initially promising, ultimately led us to develop a secure alternative approach that involves some efficiency trade-offs. These results underscore the inherent challenges in constructing robust MPC combiners, aligning with established impossibility results of 1-out-of-2 OT combiners.

Future Work

The work presented in this dissertation opens several avenues for future research. One immediate direction is to extend the current combiners to more complex adversarial models, such as the malicious model, where adversaries may deviate arbitrarily from the protocol. Additionally, while the current study focused on two-party settings, extending combiners to multi-party scenarios is another promising area.

Another crucial area for future exploration is the development of new techniques and methodologies for constructing MPC combiners. The limitations encountered in this work suggest that achieving robust, general-purpose combiners for MPC may require fundamentally new approaches that break away from existing paradigms in cryptographic combiners. Such advancements could significantly enhance the applicability and security of MPC protocols across a broader range of cryptographic applications.

Bibliography

[1] Daniel J Bernstein and Tanja Lange. Post-quantum cryptography. *Nature*, 549(7671):188–194, 2017.

[2] David Chaum, Ivan B Damgård, and Jeroen Van de Graaf. Multiparty computations ensuring privacy of each party's input and correctness of the result. In *Advances in Cryptology—CRYPTO'87: Proceedings 7*, pages 87–119. Springer, 1988.

[3] Michele Ciampi, Ivan Damgård, Divya Ravi, Luisa Siniscalchi, Yu Xia, and Sophia Yakoubov. Broadcast-optimal four-round mpc in the plain model. In *Theory of Cryptography Conference*, pages 3–32. Springer, 2023.

[4] Michele Ciampi, Rafail Ostrovsky, Hendrik Waldner, and Vassilis Zikas. Round-optimal and communication-efficient multiparty computation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 65–95. Springer, 2022.

[5] Oded Goldreich. *Foundations of Cryptography, Volume 2*. Cambridge university press Cambridge, 2004.

[6] Danny Harnik, Joe Kilian, Moni Naor, Omer Reingold, and Alon Rosen. On robust combiners for oblivious transfer and other primitives. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 96–113. Springer, 2005.

[7] Carmit Hazay and Yehuda Lindell. *Efficient secure two-party protocols: Techniques and constructions*. Springer Science & Business Media, 2010.

[8] Amir Herzberg. On tolerant cryptographic constructions. In *Topics in Cryptology–CT-RSA 2005: The Cryptographers' Track at the RSA Conference 2005, San*

Francisco, CA, USA, February 14-18, 2005. Proceedings, pages 172–190. Springer, 2005.

[9] Yehuda Lindell. How to simulate it–a tutorial on the simulation proof technique. *Tutorials on the Foundations of Cryptography: Dedicated to Oded Goldreich*, pages 277–346, 2017.

[10] Andrew Chi-Chih Yao. How to generate and exchange secrets. In *27th annual symposium on foundations of computer science (Sfcs 1986)*, pages 162–167. IEEE, 1986.