Efficient Secure Two-Party Computation against Malicious Players

A Dissertation

Presented to

the Faculty of the School of Engineering and Applied Science

University of Virginia

In Partial Fulfillment

of the requirements for the Degree

Doctor of Philosophy (Computer Science)

by

Chih-hao Shen

August 2013

Contents

List of Figures

List of Tables

List of Protocols

Chapter 1

Introduction

In this thesis, we explore the research topic that is inspired by the seminal problem—the *Millionaire Problem*—asked by Andrew Yao [Yao82]:

> Two millionaires wish to know who is richer; however, they do not want to find out inadvertently [unduly give away] any additional information about each other's wealth. How can they carry out such a conversation?

This original problem soon inspired a general scenario, in which two mutually distrustful parties want to securely compute over their private inputs so that in the end, they both get their desired outputs but learn nothing more about each other's input.

To solve the problem of *secure 2PC* (two-party computation), we need to construct a protocol so that given the description of any function (called the *objective function*), this protocol satisfies two security properties—*correctness* and *privacy*. Loosely speaking, the correctness property guarantees that the protocol output is indeed the output of the objective function, and the privacy property ensures that in the course of the protocol execution, both parties cannot learn more than necessary, that is, no more than that derivable from their own input and output. A trivial solution is have both parties hand their private inputs to a trusted third party who performs the work and later distributes the computation result. However, the solution that we desire needs to achieve the effect of a trusted third party without one.

In particular, what we are seeking in this thesis is not just a solution but one that is efficient enough for real-world applications. Andew Yao proposed an efficient protocol that allows two parties—one (called the *generator*) with secret input x and the other (called the *evaluator*) with secret input y—to jointly compute objective function f so that both parties receive $f(x,y)$ in the end [Yao86]. His idea is to express f as a boolean circuit (called the *objective circuit*) and devise a way for the generator to garble the circuit and then for the evaluator to obliviously compute the value of each wire. The Yao protocol is arguably one of the most efficient solutions in the literature.

Unfortunately, the Yao protocol assumes both parties to be *honest-but-curious*, that is, the correctness and privacy properties hold only when both parties follow the protocol honestly. This assumption is too strong to be applicable to most real-world applications. Later, Goldreich, Micali, and Wigderson formulated a stronger notion, the *malicious* model, in which either (but not both) participant is allowed to cheat arbitrarily [GMW87]. An elegant solution was also suggested. Note that a solution in the presence of malicious adversaries ensures that if both participants are honest, they receive their outputs with two security properties satisfied; otherwise, a cheater is caught with high probability. However, Goldreich, Micali, and Wigderson's solution uses zero-knowledge proofs that are prohibitive in practice.

In summary, this thesis aims to provide efficient solutions to secure 2PC problem in the presence of malicious adversaries. We next outline each chapter as follows.

Chapter 3: Secure 2PC Framework.

The biggest vulnerability of the Yao protocol when dealing with malicious adversaries is that a malicious generator could cheat in the garbling process. In particular, a malicious generator could construct a faulty circuit that reveals the evaluator's input purposely. Since the circuit is "garbled," the evaluator cannot determine whether or not it is an honestly garbled version of the objective circuit and should not proceed without a guarantee of its correctness.

The *cut-and-choose technique* is one of the most efficient methods that prevent a malicious generator from cheating with faulty circuits. Loosely speaking, this technique instructs the generator to prepare multiple copies of the garbled circuit, each with independent randomness. The evaluator is then instructed to choose a random fraction of the garbled circuits whose randomness is then revealed. If any of the chosen garbled circuits is not consistent with the revealed randomness, the evaluator detects the generator cheating and aborts; otherwise, the evaluator starts to evaluate the remaining garbled circuits. After the evaluation is finished, the evaluator takes the majority output as the final output.

However, the cut-and-choose technique is not a cure-all. There are issues that this technique does not cover such as the evaluator's Input Polluting Attack and the evaluator's Input Key Inconsistency Attack (also known as the Selective Failure Attack), not to mention that it brings its own issues such as the generator's Input Inconsistency Attack. These issues naturally lead to an important question:

Question: Under the cut-and-choose paradigm, what are the fundamental cryptographic primitives needed in order to transform the Yao protocol into one that is secure in the presence of malicious adversaries?

To answer this question, we propose a framework. This framework contains a two-party interactive protocol and a few cryptographic primitives. In particular, the former comes from applying the cut-and-choose technique to the Yao protocol with the support of the latter, which are formally defined to capture the security properties needed in order to address the known attacks. Most importantly, we prove that the main protocol securely computes any single-output objective function even in the presence of malicious adversaries.

This framework brings benefits to both theoretical research and practical systems. In theory, this framework basically reduces the solving of secure 2PC problem to instantiating the underlying cryptographic primitives. We understand that the development of a full-fledged secure 2PC protocol could sometimes be a tedious process, the writing of a full hybrid argument for security in particular. To the best of our knowledge, a full description of the hybrid argument is rarely presented [LP07, SS11]. In fact, a few prior works have tackled the known security issues individually and taken the rest as a straightforward integration only to find out later that the full protocol is not clear to enjoy the simulation security [MF06, KS08a]. This framework allows future researchers to conquer sub-problems one at a time. Once those building blocks are instantiated, the simulation security follows automatically. In practice, this framework provides a blueprint for implementations. Its well-modularized nature allows easy plug-and-play for the building blocks. Moreover, the framework is designed to introduce as little overhead as possible, by building the main protocol on top of a fast 2PC protocol, the Yao protocol, with an efficient technique, the cut-and-choose. A system developer can focus on optimizing primitive instantiations, which are smaller and easier to manage.

Chapter 4: Secure 2PC Instantiations

In order to demonstrate the power of our framework, we instantiate the building blocks by following two directions—using number-theoretic assumptions and auxiliary circuits.

Secure 2PC via Number-Theoretic Assumptions

Number-theoretic assumptions have been a powerful tool for cryptography since the very first public key encryption scheme was proposed [DH76]. Tractability of factoring large numbers, discrete logarithm, and quadratic residuosity has endured decades of intense study without reaching a decisive answer. Most cryptographers are comfortable with building their work on top of these unproven but very likely hard assumptions. Moreover, these mathematical objects have well-understood structures and are often useful to manipulate with. We therefore use them as a stepping stone. In particular, we instantiate the needed building blocks with the *malleable claw-free collections* and *committing oblivious transfers*, the existence of which can be implied by the Decisional Diffie-Hellman assumption.

Informal Theorem (Instantiation using DDH Assumptions). Assume the existence of oblivious transfers secure in the presence of malicious adversaries and the Decisional Diffie-Hellman assumption. There exists a Yao-based protocol that securely computes any single-output function even in the presence of malicious adversaries.[1]

This theorem is the partial result in our paper published in EUROCRYPT'11 [SS11]. Prior to that paper, the state-of-the-art approach utilized $O(k^2)$ symmetric cryptographic operations to defeat all the known attacks [LP07], where k is the security parameter. The communication overhead was soon discovered to be the bottleneck [LPS08, PSSW09]. We were able to reduce the overhead to $O(k)$ elliptic-curve operations, which incurs a comparable computation cost but much less communication cost with reasonable security parameter k.

The use of the number-theoretic objects, however, becomes less appealing as the security level increases. For the purpose of practicality, it would be the best to use only $O(k)$ symmetric cryptographic operations. We therefore consider the second direction.

[1]We understand that in the above statement, the existence of maliciously secure OTs is redundant since it is implied by the DDH assumption already. However, we choose to specify both in order to emphasize the fact that the transformation itself relies on the DDS assumption too.

Secure 2PC via Auxiliary Circuits

To detect a certain attack, the standard technique requires a participant to provide a proof that it has behaved honestly during the collaboration. We observe that the Yao protocol is itself a powerful tool to provide such a proof. So we propose that in addition to the objective circuit, both participants jointly work on a few auxiliary circuits, whose output is exactly the proof of honest behavior that the other party expects. In particular, we instantiate the primitives needed in the framework with a *2-universal hash circuit* and a *probe-resistant matrix circuit*.

It is worth-mentioning that the use of auxiliary circuits is not new [Gol04, LP07], but it was not taken into serious consideration previously. The main reason is that many auxiliary circuits were considered too costly. Indeed, back in 2009, Pinkas et al. were able to report a system that barely handles a circuit of 33,880 gates at a rate of 40 gates per second [PSSW09]. It was not until recently that our work reported a system that is capable of handling a circuit of billions of gates at a rate of hundreds of thousands of gates per second [KShS12]. So the burden of those auxiliary circuits is no longer unbearable.

Informal Theorem (Instantiation using Minimal Assumptions). Assume the existence of oblivious transfers secure in the presence of malicious adversaries. There exists a Yao-based protocol that securely computes any single-output function even in the presence of malicious adversaries.

The result of this chapter shows that secure 2PC is possible in the malicious model as long as a maliciously secure OT exists, which is the minimal assumption needed.

Chapter 5. Secure 2PC over Two-Output Functions

For many real-world applications, both parties want to learn an output from the secure computation. In the Yao protocol, it is easier for the evaluator to learn its output. The challenge is for the generator to learn its ouput securely. In particular, a solution needs to achieve the generator's *output privacy* and *output authenticity*. The former requires that the generator's output is kept only to itself, and the latter requires that the generator gets either an authentic output or no output at all, in which case the evaluator is caught cheating. We stress that the two-output protocols we consider are not *fair*, that is, the evaluator may learn its own output but refuse to send the generator's back—but if so, the evaluator is caught cheating.

Similarly, we start with using number-theoretic assumptions to extend our objective functions from single-output to two-output. In particular, we use a *zero-knowledge proof* for the evaluator to prove the generator's output authenticity. Next, we suggest a light-weight *witness-indistinguiable proof* to achieve the same goal while using only symmetric cryptographic operations.

Chapter 6: Secure 2PC in Practice.

The goal of this chapter is to present a high performance secure 2PC system that integrates state-of-the-art techniques for dealing with *malicious* adversaries efficiently. Although some of these techniques have been reported individually, we are not aware of any attempt to incorporate them all into one system, while ensuring that a security proof can still be written for that system. Even though some of the techniques are claimed to be compatible, it is not until everything is put together and someone has gone through all the details can a system as a whole be said to be provably secure.

Our system is an implementation of our framework plus two proposed building block instantiations. We use Ishai et al.'s oblivious transfer extension [IKNP03] that has efficient amortized computation time for oblivious transfers secure in the presence of malicious adversaries. We implement our witness-indistinguishable proof for the generator's output authenticity.

For circuit garbling and evaluating, we incorporate Kolesnikov and Schneider's free-XOR technique that minimizes the computation and communication cost for XOR gates in a circuit [KS08b]. We also adopt Pinkas et al.'s garbled-row-reduction technique that reduces the communication cost for k-fan-in non-XOR gates by $1/2^k$ [PSSW09], which means at least 25% communication saving in our system since we only have 2-fan-in gates. Finally, we implement Goyal et al.'s technique for reducing communication as follows: during the cut-and-choose step, the check circuits are given to the evaluator by revealing the randomness used to produce them rather than the check circuits themselves [GMS08]. This technique will provide at least 50% saving in communication. Although these techniques exist individually, our work is the first system to incorporate all of these mutually compatible state-of-the-art techniques.

Besides the improvements by the algorithmic means, we also incorporate the latest hardware instruction sets. In particular, we use the SSE2 (Streaming SIMD 2) to speed up bit-wise operations such as 128-bit XOR, 128-bit AND, and we utilize AES-NI (Advanced Encryption Standard New Instructions) to improve the speed of circuit garbling and evaluation.

The most important new technique that we use is to exploit the embarrassingly parallelizable nature of our secure 2PC framework. Exploiting this, however, requires careful engineering in order to achieve good performance while maintaining security. We parallelize all computation-intensive operations such as oblivious transfers or circuit construction by splitting the generator-evaluator pair into hundreds of slave pairs. Each of the pairs works on an independently generated copy of the circuit in a parallel but synchronized manner as synchronization is required for our framework to be secure.

Experimental Results (Secure 2PC in Parallel) In the environment where there are poly(k) processors and poly(k) bandwidth, our system is able to achieve the computation ratio $1+\varepsilon$, in terms of wall clock time, between the malicious model and the honest-but-curious model, where ε approaches 0 as the circuit size increases.

Chapter 2

Preliminaries

2.1 Notations

General Notations

Integer and String Representation: We denote by $\mathbb{N}$ the set of natural numbers $\{1, 2, \ldots\}$, by $\mathbb{R}$ the set of real numbers, and by $\mathbb{R}^+$ the set of non-negative real numbers. For any $n \in \mathbb{N}$, we denote by $[n]$ the set of $\{1, 2, \ldots, n\}$, by 1^n the concatenation of n 1s, by $\{0, 1\}^n$ the set of n-bit strings, and by $\{0, 1\}^*$ the set of binary strings. If x and y are binary strings, we denote by $x||y$ (or simply xy) their concatenation and by $|x|$ the bit length of x.

Bit Operations: If $x, y \in \{0, 1\}$, we denote by $x \wedge y$ the output of AND(x, y), by $x \vee y$ the output of OR(x, y), and by $x \oplus y$ the output of XOR(x, y).

Cryptographic Notations

For convenience, we present the notations of cryptographic primitives needed here without much detail. These notations will be mentioned again later when their formal definitions are introduced.

For some proper $m, r \in \{0,1\}^*$, we denote by $\mathsf{com}(m; r)$ a commitment to m with randomness r (cf. Definition 3.1). For simplicity, the randomness may sometimes be omitted.

For any $x, y \in \{0,1\}^*$, we denote by $f(x, y) \mapsto (f_1(x, y), f_2(x, y))$ a two-output function, where the first party on input x gets output $f_1(x, y)$, and the second party on input y gets output $f_2(x, y)$. For simplicity, $f_1(x, y)$ and $f_2(x, y)$ are often simplified as f_1 and f_2, respectively. For some function, we use $f_b = \perp$ to indicate that the b-th party gets no output, where $b \in \{1,2\}$.

For some proper $m, e, c, d \in \{0,1\}^*$, we denote by $\mathsf{enc}_e(m)$ the encryption (or ciphertext) of message (or plaintext) m with encryption key e and by $\mathsf{dec}_d(c)$ the decryption of ciphertext c with decryption key d (cf. Definition 2.6).

Probabilistic Notations

We focus on probability distributions $D : S \to \mathbb{R}^+$ over some finite set S.

Probabilistic assignments: If $D : S \to \mathbb{R}^+$ is a probability distribution, we denote by $x \overset{R}{\leftarrow} D$ the elementary procedure that x is chosen from S according to distribution D. We denote by $x \overset{R}{\leftarrow} S$ the procedure that x is chosen from S according to a uniform distribution.

Probabilistic Experiments: Let p be a predicate and $D_1, D_2, \ldots$ be probability distributions. We denote by $\Pr[x_1 \overset{R}{\leftarrow} D_1; x_2 \overset{R}{\leftarrow} D_2; \cdots : p(x_1, x_2, \ldots)]$ the probability that $p(x_1, x_2, \ldots)$ is true after the ordered execution of probability assignments $x_1 \overset{R}{\leftarrow} D_1$, $x_2 \overset{R}{\leftarrow} D_2$, and so on.

New Probability Distributions: If $D_1, D_2, \ldots$ are probability distributions, we denote by $\{x_1 \overset{R}{\leftarrow} D_1; x_2 \overset{R}{\leftarrow} D2; \cdots : (x_1, x_2, \cdots)\}$ the new probability distribution over $(x_1, x_2, \cdots)$ generated by the ordered execution of probabilistic assignments $x_1 \overset{R}{\leftarrow} D_1$, $x_2 \overset{R}{\leftarrow} X_2$, and so forth.

Probability Ensemble Let I be a countable index set. We denote by $X = \{X_i\}_{i \in I}$ a *probability ensemble indexed by I*, which is is a sequence of random variables indexed by I.

2.2 Basic Notions

Negligible Functions: Negligible functions are those asymptotically smaller than the inverse of any fixed polynomial. The formal definition is as follows.

Definition 2.1. *A function $\mu(\cdot)$ from $\mathbb{N}$ to $\mathbb{R}^+$ is called* negligible *if for every constant $c > 0$ and all sufficiently large $n \in \mathbb{N}$, it holds that $\mu(n) < n^{-c}$.*

Polynomial-Time Algorithms By an algorithm we mean a Turing machine, hence will be used interchangeably. We only consider finite algorithms, that is, the machines that have some fixed upper bound on their running time. In particular, let M be an algorithm. We denote by STEPS(M, x) the number of computational steps taken by M on input x.

Definition 2.2. *Algorithm M is* polynomial-time *if there exists some polynomial $p(\cdot)$ such that for all $x \in \{0,1\}^*$ it holds that STEPS$(M, x) = O(p(|x|))$.*

Security Parameter It is a convention in modern cryptography that security is defined with respect to computationally bounded machines. Since computing power advances with the time, it is well-accepted that cryptographic protocols are designed with a security parameter as an auxiliary input so that their computational strength can change easily as technology improves. At the time this writing, it is generally considered sufficient to have security parameter 80, or equivalently, security level 2^{-80}, which means that an adversary either needs to perform at least 2^{80} operations or has probability at most 2^{-80} to break the security.

Indistinguishability

Definition 2.3 (Indistinguishability). *Two ensembles $X = \{X_n\}_{n\in\mathbb{N}}$ and $Y = \{Y_n\}_{n\in\mathbb{N}}$ are* computationally indistinguishable*, denoted by $X \overset{c}{\approx} Y$, if for every probabilistic polynomial-time algorithm D, every auxiliary input $z \in \{0,1\}^{poly(n)}$, there exists a negligible function $\mu(\cdot)$ such that*

$$|\Pr[x \leftarrow X_n : D(x,z) = 1] - \Pr[y \leftarrow Y_n : D(y,z) = 1]| < \mu(n).$$

X and Y are said to be statistically indistinguishable*, denoted by $X \overset{s}{\approx} Y$, if the above condition holds for all (possibly unbounded) algorithms D.*

Remark 2.1. *The auxiliary input z given to the distinguishing algorithm D implicitly implies that the two ensembles are indistinguishable to any non-uniform machine.*

2.3 Secure 2PC Notions

A *secure 2PC* (secure two-party computation) problem is specified by first identifying an objective function $f : \{0,1\}^{n_1} \times \{0,1\}^{n_2} \mapsto \{0,1\}^{m_1} \times \{0,1\}^{m_2}$ which maps a pair of inputs (x, y) to a pair of outputs $(f_1(x, y), f_2(x, y))$. The first party supplies input x and obtains output $f_1(x, y)$, and the second supplies y and obtains $f_2(x, y)$.

Next, a security notion for protocols needs to be formally defined. A standard approach is to follow the "ideal/real" paradigm in which we compare what an adversary can accomplish in the proposed protocol versus what an adversary can accomplish in an idealized setting in which a trusted third party aids in the computation. Here we summarize a well-accepted definition for secure two-party computation suggested by Goldreich [Gol04].

2.3.1 The Honest-but-Curious Model

In this model, both parties behave almost like an honest party, who always follows protocol instructions faithfully. The only difference is that an honest-but-curious adversary would attempt to extract useful information out of its own transcript of a proper execution. Loosely speaking, the security notion defined here captures the idea that *for either party, whatever can be obtained from an execution of the protocol could be essentially obtained from the input and output available to that party*. We now formally introduce this security notion.

Ideal Model: Let $\bar{S} = (S_1, S_2)$ be a pair of non-uniform expected polynomial-time machines. S_1 on input x and S_2 on input y engage in an ideal execution of f as follows. Both $S_1(x)$ and $S_2(y)$ send their inputs to an oracle O honestly. O computes $(f_1, f_2) \leftarrow f(x, y)$, and sends f_2 to S_2 and then f_1 to S_1. An honest party always outputs the message received from O, whereas an honest-but-curious adversary determines its output based on its own input and the message from O. Let variable $\mathbf{Ideal}_{f,\bar{S}}(x, y, 1^k)$ represent the joint output of $S_1(x)$ and $S_2(y)$ from the above experiment with security parameter k as a common input. If at least one of S_1, S_2 is honest, then $\bar{S}$ is *admissible*.

Real Model: We now consider the real model in which a real two-party protocol $\Pi = (\Pi_1, \Pi_2)$ consisting of a pair of interactive non-uniform probabilistic polynomial-time machines is executed without any oracle. In particular, if $\bar{P} = (P_1, P_2)$ is a pair of non-uniform probabilistic polynomial-time machines, in the real experiment, P_1 on input x and P_2 on input y begin exchanging messages until one party halts. Both parties follow protocol instructions honestly. An honest party outputs what the protocol instructs, whereas an honest-but-curious adversary computes its output based on its entire view of the execution. Let variable $\mathbf{Real}_{\Pi,\bar{P}}(x, y, 1^k)$ represent the final joint output of $P_1(x)$ and $P_2(y)$ with security parameter k as a common input. Again, if at least one P_i is the same as the honest protocol Π_i, then $\bar{P}$ is called *admissible.*

Definition 2.4 (Security in the Honest-but-Curious Model)**.** *A protocol* Π *is said to* securely compute *function* $f : \{0,1\}^{n_1} \times \{0,1\}^{n_2} \mapsto \{0,1\}^{m_1} \times \{0,1\}^{m_2}$ *in the presence of honest-but-curious adversaries if for every pair of admissible non-uniform probabilistic polynomial-time machines* $\bar{P} = (P_1, P_2)$ *for the real model, there exists a pair of admissible non-uniform probabilistic expected polynomial-time machines* $\bar{S} = (S_1, S_2)$ *for the ideal model such that*

$$\left\{\mathbf{Real}_{\Pi,\bar{P}}(x, y, 1^k)\right\}_{x\in\{0,1\}^{n_1}, y\in\{0,1\}^{n_2}, k\in\mathbb{N}} \overset{c}{\approx} \left\{\mathbf{Ideal}_{f,\bar{S}}(x, y, 1^k)\right\}_{x\in\{0,1\}^{n_1}, y\in\{0,1\}^{n_2}, k\in\mathbb{N}}.$$

2.3.2 The Malicious Model

In this model, a malicious party may refuse to participate in the protocol, may use a different input than it is given, or may perform extra instructions or instructions different from those specified in the protocol. In our case, we allow "unfair" protocols in which the malicious party may be able to learn its output and prevent the honest party from doing so. The malicious party, however, is fixed before the protocol begins, that is, the static corruption model.

Ideal Model: Let $\bar{S} = (S_1, S_2)$ be a pair of non-uniform expected polynomial-time machines. S_1 on input x and S_2 on input y engage in an ideal execution of f as follows. Both $S_1(x)$ and $S_2(y)$ send their inputs to an oracle O. An honest party always sends x (or y) to O, and a malicious party may send any n_1-bit (or n_2-bit) string or the special message $\perp$ (abort). If O does not receive two valid inputs, then O responds to both parties with the special message $\perp$. If O receives a valid input pair (x', y'), it computes $(f_1, f_2) \leftarrow f(x', y')$. It then sends f_2 to S_2 and waits for a one-bit response. If S_2 responds with 1, then O sends f_1 to S_1; otherwise, S_1 receives $\perp$ as output. If S_2 is honest, it always sends 1 to O if queried after receiving f_2. At the end, an honest party always outputs the message received from O. Let variable $\mathbf{Ideal}_{f,\bar{S}}(x, y, 1^k)$ represent the joint output of $S_1(x)$ and $S_2(y)$ from the above experiment with security parameter k as a common input. If at least one of S_1, S_2 is honest, then $\bar{S}$ is *admissible*.

Real Model: We now consider the real model in which a real two-party protocol $\Pi = (\Pi_1, \Pi_2)$ consisting of a pair of interactive non-uniform probabilistic polynomial-time machines is executed without any oracle. In particular, if $\bar{P} = (P_1, P_2)$ is a pair of non-uniform probabilistic polynomial-time machines, in the real experiment, P_1 on input x and P_2 on input y begin exchanging messages until one party halts. An honest party P_i follows protocol instructions Π_i and outputs what the protocol instructs. A dishonest party may output an arbitrary string. Let variable $\mathbf{Real}_{\Pi,\bar{P}}(x, y, 1^k)$ represent the final joint output of $P_1(x)$ and $P_2(y)$ with security parameter k as a common input. Again, if at least one P_i is the same as the honest protocol Π_i, then $\bar{P}$ is called *admissible.*

Definition 2.5 (Secure two-party computation). *A protocol* Π *is said to* securely compute *function* $f : \{0,1\}^{n_1} \times \{0,1\}^{n_2} \mapsto \{0,1\}^{m_1} \times \{0,1\}^{m_2}$ *in the presence of malicious adversaries if for every pair of admissible non-uniform probabilistic polynomial-time machines* $\bar{P} = (P_1, P_2)$ *for the real model, there exists a pair of admissible non-uniform probabilistic expected polynomial-time machines* $\bar{S} = (S_1, S_2)$ *for the ideal model such that*

$$\left\{\mathbf{Real}_{\Pi,\bar{P}}(x, y, 1^k)\right\}_{x\in\{0,1\}^{n_1}, y\in\{0,1\}^{n_2}, k\in\mathbb{N}} \overset{c}{\approx} \left\{\mathbf{Ideal}_{f,\bar{S}}(x, y, 1^k)\right\}_{x\in\{0,1\}^{n_1}, y\in\{0,1\}^{n_2}, k\in\mathbb{N}}.$$

Remark 2.2. *We assume that we are given the objective circuit that is indeed functionally equivalent to the objective function. Among all the boolean circuits functionally equivalent to the objective function, the smaller the size of the circuit, the better. Conceivably, optimizations will be performed* *over the objective circuit for real-world applications. Note that deciding whether an optimized circuit is functionally equivalent to the objective function is NP-hard, which is out of the scope of this thesis. As a result, we preclude the possibility of corrupted objective circuits.*

2.4 Cryptographic Primitives

Definition 2.6 (Symmetric Cipher with Semantic Security). (enc, dec).

$$\Pr[(m_0, m_1, s) \leftarrow A_1(z); b \leftarrow \{0,1\}; b' \leftarrow A_2(enc_k(m_b), s) : b = b'] < \mu(k).$$

Chapter 3

Secure 2PC Framework

3.1 The Yao Protocol

3.1.1 The High-Level Construction

We first describe our starting point—the Yao protocol. This protocol begins with a clever way of evaluating a boolean gate in an oblivious manner. Let us take a NAND gate with two input wires w_1 and w_2 and one output wire w_3 as an example. Each wire w_i is first assigned a pair of random keys $(K_{i,0}, K_{i,1})$. The idea is to associate random keys $K_{i,0}$ and $K_{i,1}$ with the semantics that wire w_i is of bit value 0 and 1, respectively, and to *garble* the truth table accordingly, as shown in Table 3.1.

w_1	w_2	w_3
0	0	1
0	1	1
1	0	1
1	1	0

(a) $\text{NAND}(w_1, w_2) = w_3$

$$\begin{bmatrix} \mathsf{enc}_{K_{1,0}}(\mathsf{enc}_{K_{2,0}}(K_{3,1})) \\ \mathsf{enc}_{K_{1,0}}(\mathsf{enc}_{K_{2,1}}(K_{3,1})) \\ \mathsf{enc}_{K_{1,1}}(\mathsf{enc}_{K_{2,0}}(K_{3,1})) \\ \mathsf{enc}_{K_{1,1}}(\mathsf{enc}_{K_{2,1}}(K_{3,0})) \end{bmatrix}$$

(b)

Table 3.1: (a) The truth table of a NAND gate and (b) its garbled version.

Under this design, anyone given the garbled truth table and *exactly one* key for each input wire is able to decrypt one of the entries and retrieve *exactly one* key for the output wire. For instance, anyone given Table 3.1(b) and keys $K_{1,0}$ and $K_{2,1}$ is able to retrieve key $K_{3,1}$. This is equivalent to looking up in Table 3.1(a) with wire w_1 of value 0 and wire w_2 of value 1 and deciding that wire w_3 is of value 1. Observe that this evaluation does not require the semantics of input keys. In other words, it is possible for the evaluator to complete a correct computation but remain oblivious to the input and output. To develop this single gate oblivious evaluation into a full-fledged solution, researchers have made the following observations:

- **Two Parties:** It is natural that this oblivious evaluation is jointly performed by two independent parties. Conceivably, the semantics of random keys $K_{i,b}$s is necessary for garbling a truth table, whereas the same information is not needed for evaluating a garbled gate. In fact, the semantics has to be kept secret from the evaluator for the evaluation to be oblivious. As a consequence, one of the two parties (denoted by GEN hereafter) will be garbling the boolean gate, whereas the other party (denoted by EVAL) will be evaluating it.
- **Garbling the Whole Circuit:** The garbling technique is in fact applicable to all boolean gates, not just NAND gates, since the garbling is independent of the content of a truth table. Moreover, this technique applies to the entire circuit, not just a single gate. The output key from a gate can be the input key for a next-level gate. The same oblivious evaluation can propagate from circuit-input wires all the way to circuit-output wires.
- **Exactly One Key Learned per Wire:** It is of paramount importance to maintain the invariant that EVAL *learns exactly one random key for each wire during the course of the evaluation*. If this invariant is violated, GEN's privacy could be compromised. For example, if EVAL learns both random keys assigned to its input wires, it gets to evaluate $f(x, y')$ for an arbitrary y'. This obviously reveals more information about x than what EVAL could learn when using a trusted third party.

The single gate oblivious evaluation described above shows that if the invariant holds for all gate-input wires, it holds for the gate-output wire, like the inductive step of an induction argument. Therefore, it remains to ensure that the invariant holds for circuit-input wires, just like the base case of the induction.

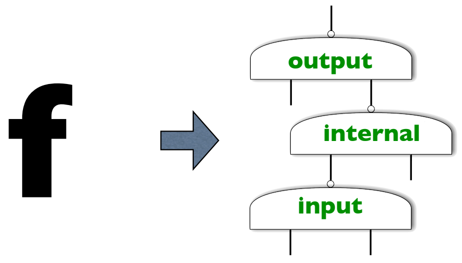
Without loss of generality, let us assume $x, y \in \{0, 1\}$. For the wire corresponding to GEN's input, GEN could simply reveal the key corresponding to x since the key looks random to EVAL. For the wire corresponding to EVAL's input, both parties could run a sub-protocol of a secure 1-out-of-2 oblivious transfer, denoted by $\binom{2}{1}$-OT. In this sub-protocol, GEN provides as input both random keys (K_0, K_1) assigned to EVAL's input wire, and EVAL provides as input its private input y. Upon completion, two conditions need to hold: (1) GEN learns nothing about y and (2) EVAL learns K_y but nothing about K_{1-y}. We can see that the former security property ensures EVAL's input privacy, and the latter ensures that the invariant holds for wires corresponding to EVAL's input.

In summary, the Yao protocol works as follows: GEN and EVAL first agree upon an objective circuit. GEN then garbles the circuit and sends it to EVAL along with the random keys corresponding to GEN's input. Next, GEN and EVAL jointly run a $\binom{2}{1}$-OT so that EVAL receives the random keys corresponding to its input. Finally, EVAL obliviously evaluates the garbled circuit and learns the output. In Figure 3.1, we visualize the Yao protocol.

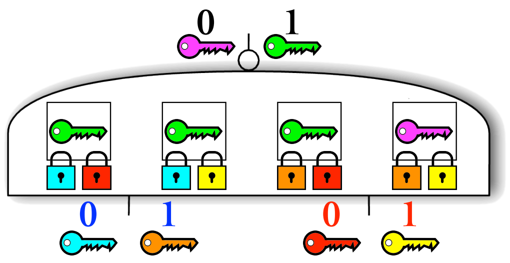
3.1.2 A Fast Construction

We next elaborate on some security issues of the Yao protocol and give a complete description of a fast construction at the end.

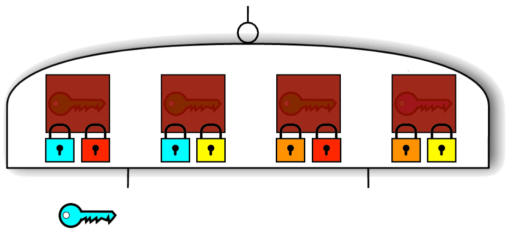
It is necessary to randomly permute the entries in the garbled truth table. If the entries are in a fixed order, EVAL would learn the semantics of the output key just by the location of the correctly decrypted entry. Even more importantly, there needs to be a mechanism for EVAL to



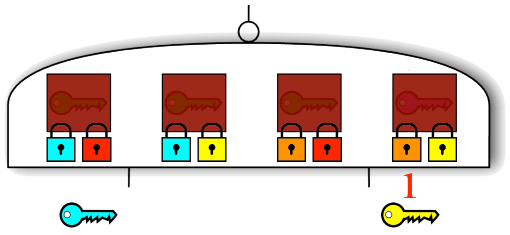
(a) Agree upon the circuit computing f



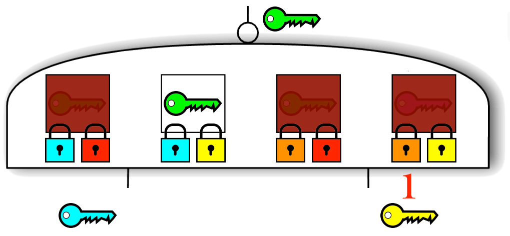
(b) Garble the circuit (e.g. a NAND gate)



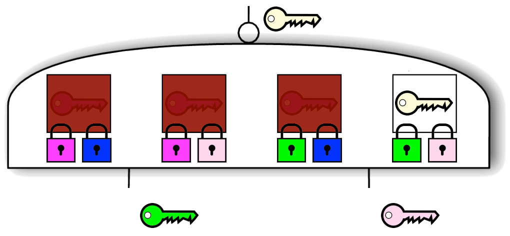
(c) The garbled circuit and GEN's input keys



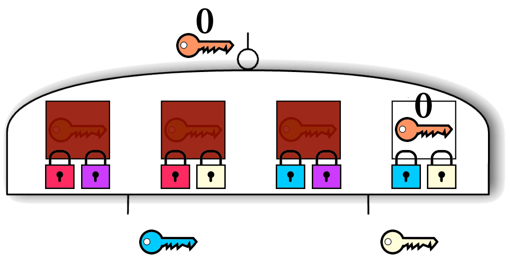
(d) Retrieve EVAL's input keys via $\binom{2}{1}$-OTs



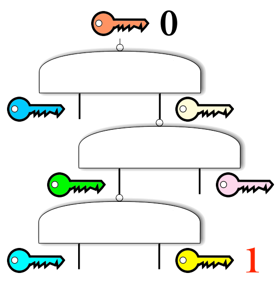
(e) Evaluate an input gate obliviously



(f) Evaluate an internal gate obliviously



(g) Evaluate an output gate obliviously



(h) Information learned by EVAL

Figure 3.1: Visualization of the Yao protocol: (a) Both parties agree upon an objective boolean circuit that computes f. (b) GEN assigns a pair of random keys to each wire and garbles each boolean gate according to its truth table. This example shows a garbled NAND gate. (c) EVAL is oblivious to the received garbled circuit and GEN's input keys. (d) EVAL receives its input keys via $\binom{2}{1}$-OT. (e)-(g) EVAL obliviously evaluates each gate while maintaining the invariant that exactly one out of the two random keys for each wire is retrieved. (h) What EVAL learns during the course of the evaluation includes the output and exactly one random key for each wire, which is close to what it would learn when delegating the computation to a trusted third party.

distinguish a wrong entry from a right one. Indeed, a wrong entry results in a mismatched decryption, in which EVAL decrypts $\mathsf{enc}_K(m)$ with some key $K' \neq K$, whereas a correct entry gives EVAL an output key. A mismatched decryption can look just as random as an output key.

There are simple ways to get around this problem. For example, the output key is always appended with 0^k for some $k \in \mathbb{N}$ before it is doubly encrypted. Later, EVAL could identify a mismatched decryption if the decrypted message does not end with 0^k. However, this solution could require EVAL to examine up to all entries in a garbled truth table, which is wasteful for real-world applications.

Here, we adopt a trick that requires examining only one entry in a garbled truth table. This trick, also known as the *point and permute* technique in the literature [MNPS04], satisfies the above properties that it randomly permutes the entries but in a way that the right entry can be easily identified by EVAL. The idea is to assign each wire an extra random bit. This bit, on one hand, determines a random order for the garbled entries, and on the other hand, helps EVAL locate the right entry deterministically.

In particular, this random bit indicates whether the corresponding pair of random keys need to be "swapped" while used as encryption keys. Let us consider the NAND gate example again. Suppose now each wire w_i is assigned $(K_{i,0}, K_{i,1}, \pi_i)$, where $\pi_i \in \{0,1\}$. The base case for the garbled version of this gate is $[\langle 0,0\rangle, \langle 0,1\rangle, \langle 1,0\rangle, \langle 1,1\rangle]$, where $\langle d_1, d_2\rangle$ denotes $\mathsf{enc}_{K_{1,d_1}}(\mathsf{enc}_{K_{2,d_2}}(K_{3,d_1 \oplus d_2}))$. When $\pi_1 = 1$ and $\pi_2 = 0$, GEN constructs the garbled truth table the same as the base case except that $K_{1,0}$ is replaced with $K_{1,1}$, and vice versa, but $K_{2,0}$ and $K_{2,1}$ stay the same because π_1 indicates to "swap" $K_{1,0}$ and $K_{1,1}$ but π_2 does not. The garbled truth table in this case therefore becomes $[\langle 1,0\rangle, \langle 1,1\rangle, \langle 0,0\rangle, \langle 0,1\rangle]$. In Table 3.2, we list the four possible swapping patterns.

Observe that in Table 3.2, regardless of what (π_1, π_2) is,

1. the ordered entries are always $[\langle \pi_1, \pi_2\rangle, \langle \pi_1, 1-\pi_2\rangle, \langle 1-\pi_1, \pi_2\rangle, \langle 1-\pi_1, 1-\pi_2\rangle]$;

(π_1, π_2)	Ordered Entries
$(0,0)$	$[\langle 0,0\rangle, \langle 0,1\rangle, \langle 1,0\rangle, \langle 1,1\rangle]$
$(0,1)$	$[\langle 0,1\rangle, \langle 0,0\rangle, \langle 1,1\rangle, \langle 1,0\rangle]$
$(1,0)$	$[\langle 1,0\rangle, \langle 1,1\rangle, \langle 0,0\rangle, \langle 0,1\rangle]$
$(1,1)$	$[\langle 1,1\rangle, \langle 1,0\rangle, \langle 0,1\rangle, \langle 0,0\rangle]$

Table 3.2: All possible swapping patterns for a four-entry garbled truth table, where $\langle d_1, d_2\rangle$ denotes $\mathsf{enc}_{K_{1,d_1}}(\mathsf{enc}_{K_{2,d_2}}(K_{3,d_1 \oplus d_2}))$.

2. $\langle d_1, d_2\rangle$ is always the $(2\cdot(d_1 \oplus \pi_1) + (d_2 \oplus \pi_2))$-th entry in the table (assuming entry index starts with 0).

The first observation shows that if permutation bit π_i is independently and randomly picked for each wire, the order of the garbled entries becomes independent even for gates of the same type. So this trick solves the entry permuting problem. Note that this observation also suggests an algorithm for GEN to construct garbled gates. The second observation shows that if EVAL learns $K_{i,b}$ along with locator $b \oplus \pi_i$, it could locate the right entry regardless of what (π_1, π_2) is. In other words, if EVAL learns (K_1, δ_1) and (K_2, δ_2), where $K_i \in \{K_{i,0}, K_{i,1}\}$ is the random key and $\delta_i \in \{0, 1\}$ is the corresponding locator for wire w_i, it could simply use K_1 and K_2 to decrypt the $(2 \cdot \delta_1 + \delta_2)$-th entry in the table. Hence, this trick ensures that EVAL deterministically locates the right entry.

With this trick, the invariant now becomes that EVAL *learns exactly one key-locator pair (or called the label) for each wire during the course of the evaluation.* To maintain this invariant, it remains for GEN to (1) always append $K_{3,b}$ with locator $b \oplus \pi_3$ before it is doubly encrypted when computing an entry in a garbled truth table, and (2) provide keys for circuit-input wires with the corresponding locators, too.

The last missing link for EVAL to complete the oblivious evaluation is to let it retrieve output $f(x, y)$. Recall that the invariant ensures that EVAL retrieves a key-locator pair for each wire, and each retrieved key-locator pair is of format $(K_{i,b}, \delta_i) = (K_{i,b}, b \oplus \pi_i)$, where $K_{i,b}$ indicates that wire w_i is of value b and π_i is the random bit picked independently for w_i. Locator δ_i is in fact an encryption of wire value b with one-time pad π_i. The fact that Eval does not get a second encryption with the same π_i suggests that on one hand, this key-locator trick is not leaking extra information about b to Eval; on the other hand, the random bit (the one-time pad of the retrieved value) assigned to each circuit-output wire can be disclosed for Eval to retrieve the circuit output.

The complete description of the Yao protocol is presented in Protocol 3.1. Note that we assume both parties agree in advance upon symmetric encryption scheme $(\mathsf{enc}, \mathsf{dec})$ that enjoys semantic security.

Protocol 3.1. The Yao Protocol

Common Input: security parameter 1^k and boolean circuit C that computes $f(x, y)$.

Private Input: GEN has $x = x_1x_2 \cdots x_{n_1}$ and EVAL has $y = y_1y_2 \cdots y_{n_2}$, where x_i and y_i denote the i-th bit of x and y, respectively.

Output: Both GEN and EVAL receive $f(x, y)$ at the end.

1. **(Garble Circuit)** GEN garbles C as follows:

   (a) For each wire w_i, GEN randomly picks $(K_{i,0}, K_{i,1}, \pi_i) \overset{R}{\leftarrow} \{0,1\}^k \times \{0,1\}^k \times \{0,1\}$. Let $W_{i,b}$ denote the key-locator pair $(K_{i,b}, b \oplus \pi_i)$. $W_{i,b}$ is called the *label* corresponding to wire w_i of value b from now on.

   (b) For each gate $g : \{0,1\} \times \{0,1\} \mapsto \{0,1\}$ with input wires w_a and w_b and output wire w_c, GEN garbles g by computing

   $$G(g) \leftarrow (\langle \pi_a, \pi_b \rangle, \langle \pi_a, 1 - \pi_b \rangle, \langle 1 - \pi_a, \pi_b \rangle, \langle 1 - \pi_a, 1 - \pi_b \rangle),$$

   where $\langle d_1, d_2 \rangle = \mathsf{enc}_{K_{a,d_1}}(\mathsf{enc}_{K_{b,d_2}}(W_{c,g(d_1,d_2)}))$.

   Let $\{w_i\}_{i \in O}$ be the circuit-output wires. GEN sends garbled circuit $G(C)$ to EVAL, where $G(C) = (\{G(g)\}_{g \in C}, \{\pi_i\}_{i \in O})$.

2. **(Retrieve Input Labels)** Let $\{w_i\}_{i \in [n_1]}$ be the wires corresponding to GEN's input, and let $\{w_{n_1+i}\}_{i \in [n_2]}$ be the wires corresponding to EVAL's input.

   (a) **(GEN's Input Labels)** GEN sends its input labels $\{W_{i,x_i}\}_{i \in [n_1]}$ to EVAL.

   (b) **(EVAL's Input Labels)** For each $i \in [n_2]$, GEN and EVAL jointly execute

   $$\binom{2}{1}\text{-OT} : ((W_{n_1+i,0}, W_{n_1+i,1}), y_i) \mapsto (\perp, W_{n_1+i,y_i}).$$

Remark 3.1. *The above $\binom{2}{1}$-OTs could all be run in parallel if they provide security for parallel execution.*

3. **(Evaluate Circuit)** Eval has now obtained garbled circuit $G(C)$ and $(n_1 + n_2)$ labels corresponding to the $(n_1 + n_2)$ circuit-input wires. Eval then evaluates the circuit as follows:
   (a) For each gate g with retrieved input labels $W_a = (K_a, \delta_a)$ and $W_b = (K_b, \delta_b)$, Eval picks the $(2 \cdot \delta_a + \delta_b)$-th entry T in garbled gate $G(g)$ and computes output label $W_c \leftarrow \mathsf{dec}_{K_b}(\mathsf{dec}_{K_a}(T))$.
   (b) For each circuit-output wire w_i with corresponding label $W_i = (K_i, \delta_i)$, Eval computes its value $b_i \leftarrow \delta_i \oplus \pi_i$. Recall that π_i for circuit-output wire w_i comes with $G(C)$.
4. Eval interprets $\{b_i\}_{i \in O}$ as $f(x, y)$ and outputs $f(x, y)$, whereas Gen outputs $\perp$.

Proposition 3.1 (Security of the Yao Protocol [LP09]). *Assume that the $\binom{2}{1}$-OT protocol is secure in the presence of honest-but-curious adversaries and there exists a symmetric encryption scheme that enjoys the semantic security. Protocol 3.1 securely computes function $f : \{0,1\}^{n_1} \times \{0,1\}^{n_2} \mapsto \{\perp\} \times \{0,1\}^*$ in the presence of honest-but-curious adversaries.*

3.2 Attacks in the Presence of Malicious Adversaries

Next, we proceed to discuss potential vulnerabilities of the Yao protocol in the malicious model, in which either participant is allowed to cheat in an arbitrary way. We stress that we do not consider the case in which *both* parties deviate the protocol arbitrary since we only care about honest people's privacy not getting compromised.

3.2.1 Attacks Regarding OTs

When dealing with malicious adversaries, it is natural that a protocol that securely computes $\binom{2}{1}$-OT in the presence of malicious adversaries is needed. Nonetheless, it is known that such a protocol is insufficient. Suppose for now that both parties have access to an oracle that does $\binom{2}{1}$-OTs perfectly, that is, when invoked, this oracle gets (K_0, K_1) from Gen and gets $\sigma \in \{0, 1\}$ from Eval, and then this oracle returns K_σ to Eval as its output. We show that even with access to such an oracle, a malicious Gen is still capable of the following two attacks:

- **Eval's Input Key Inconsistency Attack:** This attack, also known as the *Selective Failure Attack* in the literature [KS06], is that a malicious Gen uses inconsistent random keys for Eval's input wire and the corresponding $\binom{2}{1}$-OT so that Eval's input can be inferred according to Eval's reaction. Indeed, a malicious Gen could assign (K_0, K_1) to Eval's input wire during circuit garbling while using (K_0, K_1^*) instead in the corresponding $\binom{2}{1}$-OT such that $K_1 \neq K_1^*$. As a consequence, if Eval's input is 0, it learns K_0 and is able to complete the evaluation without complaints; otherwise, it learns K_1^* and is doomed to an evaluation failure, and as a result, Gen learns that Eval's input must have been 1.

- **Eval's Input Polluting Attack:** Instead of entering (K_0, K_1) as input to the oracle, a malicious Gen could enter a swapped random key pair (K_1, K_0) or repeated key pairs (K_0, K_0) or (K_1, K_1). Recall that K_b has the semantics that Eval's input bit is b. Such a deviation implies that Eval's input is modified without its awareness, and as a result, the final output becomes $f(x, y')$, where y' is a polluted version of Eval's input. This attack clearly allows Gen to achieve something that it could not when delegating the computation of f to a trusted third party. In particular, the correctness property is compromised.

We stress that refusing to share the computation outcome (either unreasonable output or evaluation failure) to Gen could indeed minimize the potential damages caused by the above two attacks. Nevertheless, this quick remedy fails to provide any guarantee of correctness over Eval's output. What is even worse is that there are cases in which Gen is supposed to learn the output too. In those cases, not only is refusing to share not helping, an honest Eval could be accused of cheating for not cooperating while being cheated.

3.2.2 Attacks Regarding Circuit Garbling

The biggest vulnerability of the Yao protocol against malicious adversaries is that a malicious Gen could cheat in the garbling process. In particular, a malicious Gen could construct a *faulty circuit* that reveals Eval's input purposely. Since a garbled circuit is "garbled," Eval cannot determine whether or not it is a correct garbled version of the objective circuit and should not proceed without a guarantee of its correctness.

The cut-and-choose technique is one of the most efficient methods that prevent a malicious Gen from cheating with faulty circuits. Briefly, this technique instructs Gen to prepare multiple copies of the garbled circuit, each with independent randomness. Eval is then instructed to choose a random fraction of the garbled circuits whose randomness is then revealed. If any of the chosen garbled circuits (called the *check circuits*) is not consistent with the revealed randomness, Eval detects Gen cheating and aborts; otherwise, Eval starts to evaluate the remaining garbled circuits (called the *evaluation circuits*). Upon completion, Eval takes the *majority* of the outputs as its final output.

At a high level, the cut-and-choose technique ensures that the final output actually comes from the pre-agreed objective function. Indeed, if a malicious Gen wants to manipulate the final output, it needs to create a faulty majority among the evaluation circuits, but the chance that none of the faulty circuits is chosen and checked will be negligible. Therefore, in order to pass the check, a malicious Gen is allowed to sneak in only a few faulty circuits, and these (supposedly minority) circuits will have no influence on the final output at the end. In other words, if a malicious Gen attempts to cheat with faulty circuits, it either constructs many of them and gets caught with high probability, or constructs only a few and has no influence on the final output at all.

Unfortunately, although the cut-and-choose technique effectively deters the threat of faulty circuits, it brings a few new security issues that need to be addressed:

- **Gen's Input Inconsistency Attack:** It is conceivable that a malicious Gen could provide inconsistent inputs to different evaluation circuits. Lindell and Pinkas showed that for some objective function, it is not difficult for a malicious Gen to extract extra information about Eval's input [LP07]. For instance, suppose both parties agree upon objective function $f(x_1x_2x_3, y_1y_2y_3) \mapsto (\sum_{i=1}^{3} x_i \cdot y_i, \perp)$, where x_i and y_i is Gen's and Eval's i-th input bit, respectively. Instead of providing $x_1x_2x_3$ consistently to all evaluation circuits, a malicious Gen could provide 100, 010, and 001 to different evaluation circuits. In the end, Gen can learn the majority bit of Eval's input, which is some information about y that Gen is not supposed to learn when using a trusted third party.

- **Gen's Output Attack:** In most real-world applications, Gen also needs to receive an output from the computation, and this could be quite involved in the presence of malicious adversaries. Recall that Gen's output security needs both the privacy and correctness properties to be satisfied. The privacy property requires that Eval does not learn Gen's output, and the correctness property here requires that Eval *cannot trick* Gen *into accepting an arbitrary output*.

  The privacy property is the easy part. Both parties could agree in advance to convert objective function $f(x, y) \mapsto (f_1, f_2)$ into new objective function $g((x, r), y) \mapsto (\perp, (f_1 \oplus r, f_2))$ (and the objective circuit changes accordingly). After the completion of evaluating the new objective circuit, Eval sends $f_1 \oplus r$ back for Gen to recover f_1 with one-time pad

r that it enters as input in the first place. The one-time pad encryption's perfect privacy guarantees that Eval does not learn f_1 from $f_1 \oplus r$.

However, the non-trivial part is to guarantee the correctness property, which is equivalent to asking Eval for a proof of Gen's output authenticity. Note that when in the presence of honest-but-curious adversaries, random key K_b corresponding to a circuit-output wire of bit value b suffices to be the proof. This is because if the circuit-output wire is of value b, the invariant ensures that Eval does not learn K_{1-b}, and thus Eval cannot provide a proof of output value $1-b$, except with negligible probability. Nevertheless, this idea does not automatically apply to the malicious model.

3.3 The Cut-and-Choose Methodology

In this section, we will go through the cryptographic primitives used in our main protocol and explain at a high level their purposes. The formal definitions are also presented.

We stress that from now on, we denote by σ the statistical security parameter, namely, the repetition factor in the cut-and-choose technique. More importantly, we assume that statistical security parameter σ is some polynomial in security parameter k. This is a reasonable assumption since polynomial repetition is what polynomial-time parties are capable of handling.

3.3.1 Cut-and-Choose Circuit Garbling

Unlike dealing with honest-but-curious adversaries, trust is no longer free when the other party could be cheating in an arbitrary way. In the presence of malicious adversaries, there always has to be some proof on an action so that a potentially malicious party could not have done anything different. The cut-and-choose technique is a great method of providing such proofs as we briefly described earlier. In particular, there is a very important property needed for this technique to work. We will explain the reason of this property and show that *commitments* are a good supplementary tool. Next, we will show how this technique and commitments work in our main protocol. At the end of this section, we will give a formal definition of commitment schemes.

Let us first recap the cut-and-choose technique. The cut-and-choose technique is a generic proof methodology and is particularly useful in the following scenario:

> Suppose there is a probabilistic algorithm F (e.g., circuit garbling). What can we do to force Alice to run an honest execution of F with randomness ρ of her choice to produce output object O (e.g., a garbled circuit) while we are not allowed to learn randomness ρ?

As we mentioned earlier, the cut-and-choose technique suggests the following:

1. Ask Alice to run the algorithm multiple times using independent randomness.
2. Pick a random fraction of the objects and ask for their randomness.
3. Check the consistency of the input randomness and output object. If any of the revealed randomness does not match the corresponding object, Alice must have cheated; otherwise, with high probability, the majority of the unchecked objects must have been produced honestly.
4. Use this honest majority to simulate the one originally requested.

However, there is an exploit of which a malicious party could take advantage. In particular, F *may not be binding*. There could be easy ways for Alice to compute multiple pre-images of object O. Consider now that F is the garbling function of the Yao protocol, whose input has two parts—objective circuit C and randomness ρ. It is possible that a malicious GEN could efficiently come up with malicious circuit C' and efficient conversion function c such that for every randomness ρ, it holds that $F(C,\rho) = F(C', c(\rho))$. Therefore, after GEN generates multiple garbled circuits $F(C,\rho^{(1)}), F(C,\rho^{(2)}), \ldots$, GEN could still cheat by providing innocent input $(C,\rho^{(i)})$ for the checked circuits and using malicious input $(C', c(\rho^{(j)}))$ to work with the unchecked circuits. This attack clearly refutes the claim that the cut-and-choose technique ensures an honest majority among the unchecked objects.

Fortunately, this vulnerability can be fixed by using *commitments*. A commitment is best explained as a locked box in the physical world. A commitment scheme involves two parties—the committer and the receiver. Let us first consider the following scenario:

Commit Stage: When the committer commits to a value, it writes the value on a piece of paper, locks the paper in a box, and sends the box to the receiver.

Reveal Stage: Upon request, the committer sends over a key, and this key allows the receiver to unlock the box and learn the value inside.

In this scenario, the locked box captures two important properties of a commitment—the *hiding* and *binding* properties. The hiding property requires that the receiver cannot learn the committed value prior to the reveal stage (since the value is inside the box), and the binding property requires that the committer cannot magically change the committed value while the commitment is revealed (since the box was already delivered in the commit stage).

With the help of commitment, let us now give an improved cut-and-choose algorithm that forces Alice to provide an honest execution of probabilistic algorithm F:

1. Ask Alice to run F σ times using independent randomness $\rho^{(1)}, \rho^{(2)}, \ldots, \rho^{(\sigma)}$.
2. Ask Alice to commit to all randomness and provide all computation results, that is, ask Alice to provide commitment-object pairs $(c^{(1)}, O^{(1)}), (c^{(2)}, O^{(2)}), \ldots, (c^{(\sigma)}, O^{(\sigma)})$.
3. Pick a random fraction $E \subset [\sigma]$, and for all $j \in [\sigma] \backslash E$, do the following checks:

- Ask for the decommitments for $c^{(j)}$. If $c^{(j)}$ fails to decommit, Alice must have cheated. Let $\rho'^{(j)}$ denote the decommitted value from $c^{(j)}$.
- Check whether $O^{(j)}$ is indeed from an honest execution of F. If any of the objects does not match the revealed randomness $\rho'^{(j)}$, that is, $O^{(j)} \neq F(\rho'^{(j)})$, Alice must have cheated.

If none of the checks fails, with high probability, set E of unchecked objects has an honest majority, which can be used to simulate the one originally requested.

At a high level, this improved version endows F with the binding property without leaking extra information about ρ. It basically converts the probabilistic algorithm from F to F' such that $F'(\rho) = (\mathsf{com}(\rho), F(\rho))$ for some commitment scheme com. The binding property of commitment scheme com ensures that F' is also binding. More specifically, an adversary capable of providing distinct randomness ρ, ρ' such that $F'(\rho) = (c, O) = F'(\rho')$ has in fact broken the binding property of the commitment scheme since $\mathsf{com}(\rho) = c = \mathsf{com}(\rho')$ but $\rho \neq \rho'$. Furthermore, the hiding property ensures that c does not reveal information about ρ, or equivalently, $F'(\rho) = (c, F(\rho))$ does not reveal more information about ρ than $F(\rho)$ does.

Now let us have a closer look at how the improved cut-and-choose technique works in our main protocol:

- **Ensuring Correct Garbling:** Naturally, the most important task of our protocol is ask Gen to honestly garble the objective circuit. In other words, the probabilistic algorithm in this case is *the Yao protocol's garbling function*. Note that the garbling function is not binding. Thus, we will have to do something to ensure this, which will be covered later.
- **Ensuring Correct Input Keys:** In order to successfully evaluate a garbled circuit, the correctness of input keys is just as important as the correctness of the garbled circuit. The input key correctness has two aspects:

1. for each of the wires corresponding to GEN's input, the input key EVAL retrieves has to be one out of the two random keys that GEN assigned, and its semantics has to *be oblivious to* EVAL;
2. for each of the wires corresponding to EVAL's input, the input key EVAL retrieves has to be one out of the two random keys that GEN assigned, and its semantics has to *match* EVAL*'s input*.

To ensure the "one out of the two" property, the idea is to have GEN commit to both random keys assigned to a circuit-input wire. After GEN passes the cut-and-choose check, GEN will reveal one of the two commitments corresponding to a circuit-input wire. If the commitment is successfully decommitted, EVAL then knows that the retrieved key must be valid and could proceed to the circuit evaluation. In other words, the probabilistic algorithm that GEN needs to do in this case is to *commit to all pairs of random keys assigned to circuit-input wires*. Note that since this probabilistic algorithm (committing to pairs of random keys) is itself binding, the commitments to randomness are unnecessary.

Next, we need to handle the semantics issue. Previously, we let GEN commit to both random keys assigned to a circuit-input wire. We did not specify the order of the two commitments, but the truth is: this commitment order matters! Recall that the cut-and-choose technique requires GEN to commit to multiple pairs of random keys assigned to the same circuit-input wire, and some of the commitment pairs will be checked.

GEN's Input Key Obliviousness: Let us consider one of the wires corresponding to GEN's input. Apparently, if all the commitment pairs follow the same commitment order, EVAL would learn this order from the checked pairs, and then compromise GEN's input privacy when GEN decommits to its input keys for the unchecked pairs. To fix such a vulnerability, independent randomization for all the pairs of commitments is necessary. A simple method is to randomly swap the two commitments in each pair, and thus randomize the commitment order. However, this may require EVAL to try both commitments until the right one is found, which is wasteful for real-world applications. Fortunately, there are random bits that we can reuse—the permutation bits. Recall that in Protocol 3.1, GEN assigns two key-locator pairs $(K_{i,0}, 0 \oplus \pi_i)$ and $(K_{i,1}, 1 \oplus \pi_i)$ to each wire w_i, where π_i is a random permutation bit assigned to w_i. Since EVAL is going to learn one of the two key-locator pairs anyway, denoted by (K, δ), we could reuse δ to help EVAL locate the right commitment. In particular, GEN commits to the pair of key-locator pairs assigned to its input wire w_i in the format of $\mathsf{com}((K_{i,0\oplus\pi_i}, 0)), \mathsf{com}((K_{i,1\oplus\pi_i}, 1))$, that is, *the commitments are sorted according to the locators*. Since permutation bit π_i varies from circuit to circuit, EVAL will not be able to compromise GEN's input privacy like before, and only one out of each pair of commitments needs to be tried and revealed.

EVAL's Input Key Correctness: For each of the wires corresponding to EVAL's input, EVAL needs to be sure that the retrieved input keys indeed match its input. Similarly, we could arrange the commitment order to achieve this. In particular, we let GEN commit to the pair of key-locator pairs assigned to EVAL's input wire w_i in the format of $\mathsf{com}((K_{i,0}, 0 \oplus \pi_i)), \mathsf{com}((K_{i,1}, 1 \oplus \pi_i))$, that is, *the commitments are sorted according to the semantics of the random keys*. Under this design, EVAL can easily verify the semantics of its input keys retrieved from GEN (after this construction's correctness is proved via the cut-and-choose technique).

In summary, to ensure the input key correctness, the probabilistic algorithm for GEN to run is *to commit to both random keys assigned to a circuit-input wire. In particular,* GEN *provides* $com((K_{a,0\oplus\pi_a}, 0)), com((K_{a,1\oplus\pi_a}, 1))$ *for wire* w_a *that corresponds to* GEN*'s input and* $com((K_{b,0}, 0 \oplus \pi_b)), com((K_{b,1}, 1 \oplus \pi_b))$ *for wire* w_b *that corresponds to* EVAL*'s input.*

- **More on Ensuring Correct Garbling:** We mentioned that the garbling function itself is not binding. However, we observe that when given a garbled truth table and the labels (key-locator pairs) assigned to the input wires, the labels assigned to the output wire can be uniquely determined. In other words, when given a garbled truth table, as long as both labels assigned to the gate-input wires are binding, those assigned to the gate-output wires are binding too. More generally, when given a garbled circuit, as long as both labels assigned to all circuit-input wires are binding, those assigned to the intermediate (or circuit-output) wires are binding too. The garble circuit is automatically binding as long as Gen commits to both labels assigned to all circuit-input wires, which it does as we previously discussed.

Finally, we provide the formal definition of a commitment scheme:

Definition 3.1 (Commitment Schemes). *A deterministic polynomial-time algorithm com is called a* commitment scheme *if the following two properties hold:*

1. ***(Hiding)*** *For all non-uniform probabilistic polynomial-time distinguisher D, there exists a negligible function $\mu(\cdot)$ such that for all $k \in \mathbb{N}$ and all $v_0, v_1 \in \{0,1\}^k$, D distinguishes the following distributions with probability at most $\mu(k)$*

   - $\{r \overset{R}{\leftarrow} \{0,1\}^{poly(k)} : com(v_0; r)\}_k$
   - $\{r \overset{R}{\leftarrow} \{0,1\}^{poly(k)} : com(v_1; r)\}_k$

2. ***(Binding)*** *For all non-uniform probabilistic polynomial-time machine M, there exists a negligible function $\mu(\cdot)$ such that for all $k \in \mathbb{N}$, all $v_0 \in \{0,1\}^k$, and all $r_0 \in \{0,1\}^{poly(k)}$, it holds that*

$$\Pr\left[(v_1, r_1) \overset{R}{\leftarrow} M(com(v_0; r_0)) : v_0 \neq v_1 \wedge com(v_0; r_0) = com(v_1; r_1)\right] < \mu(k).$$

A commitment scheme is perfectly-hiding *if the hiding property holds for all (possibly unbounded) algorithms.*

A commitment scheme is perfectly-binding *if instead of the above binding property, for all $k \in \mathbb{N}$, all distinct $v_0, v_1 \in \{0,1\}^k$ and all $r_0, r_1 \in \{0,1\}^{poly(k)}$, it holds that $com(v_0; r_0) \neq com(v_1; r_1)$.*

3.3.2 Cut-and-Choose Oblivious Transfer

We have discussed that even an ideal OT will not provide enough security for secure 2PC in the presence of malicious adversaries due to EVAL's Input Key Inconsistency Attack and Input Polluting Attack. Recall that the Input Key Inconsistency Attack occurs when GEN provides invalid key pair (K_0^*, K_1) or (K_0, K_1^*) to the OT, and the Input Polluting Attack occurs when GEN provides valid but flipped key pair (K_1, K_0) or repeated keys (K_0, K_0) or (K_1, K_1) to the OT. The former can result in an evaluation failure or, even worse, EVAL's input leak, and the latter can result in a valid secure 2PC over an incorrect EVAL's input. As a result, the consistency here is twofold: (1) the consistency of the keys used in the OT and those used in the circuit garbling and (2) the semantics consistency of the keys that GEN provides and those that EVAL requests.

Here, we capture the security properties needed, in addition to those guaranteed by OTs, to defend these two attacks from a malicious GEN. Loosely speaking, after the OT, GEN needs to provide a proof for EVAL to check. This proof needs to satisfy the following two security properties.

1. No matter how EVAL reacts to the proof, GEN cannot deduce any extra information about EVAL's input. To put it another way, EVAL's response to the proof, either acceptance or rejection, needs to be independent of its input. This is also equivalent to saying that the probability that EVAL rejects the proof is independent of its input. This property is meant to prevent the Input Key Inconsistency Attack.

2. If EVAL accepts the proof, *most* of the random keys received in the OT are valid and correct. A random key is *valid* if it is indeed one of the two assigned to EVAL's corresponding

input wire by Gen in the circuit garbling, and a valid key is *correct* if its semantics indeed coincides with Eval's actual input. By "most" we mean that this property holds for the majority of the evaluation circuits. This property will defeat the Input Polluting Attack.

We stress that in addition to these two properties, the simulation security of the OT in the presence of a malicious receiver (or the sender security) is also indispensable. Loosely speaking, OT's sender security requires that the view of any malicious receiver in the real model can be generated by a simulator in the ideal model.

A technical difficulty arises when we attempt to modularize this check. Since the validity and correctness of a random key is defined externally via the circuit garbling, it is not well defined within the scope of this check. Fortunately, recall that Gen is asked to commit to both random keys assigned to Eval's input wire in the first place. These commitments can serve as the common ground for both the circuit garbling and this module. More specifically, let commitments $\{(\Omega_{i,0}^{(j)}, \Omega_{i,1}^{(j)})\}_{j\in[\sigma],i\in[n]}$ be a common input. Key $K_i^{(j)}$ is valid if it is the decommitted value from either commitment $\Omega_{i,0}^{(j)}$ or $\Omega_{i,1}^{(j)}$, and valid key $K_i^{(j)}$ is correct if it is the decommitted value from commitment $\Omega_{i,y_i}^{(j)}$, where y_i is the i-th bit of Eval's input.

Next, we provide the formal definition of the needed building block called the *Cut-and-Choose OT*. We will suggest a couple of instantiations in a later chapter.

Definition 3.2. *A CCOT (Cut-and-Choose OT) is an interactive protocol between two probabilistic polynomial-time algorithms—sender S and receiver R. In this protocol, the common input includes security parameter 1^k, statistical security parameter 1^σ, and commitments $\{\Omega_{i,0}^{(j)}, \Omega_{i,1}^{(j)}\}_{j\in[\sigma],i\in[n]}$. Moreover, the functionality of this protocol is as follows:*

$$CCOT : (\{W_{i,0}^{(j)}, W_{i,1}^{(j)}\}_{j\in[\sigma],i\in[n]}, (E, y)) \mapsto (\perp, (\{W_i^{(j)}\}_{j\in E,i\in[n]}, out)),$$

where $W_{i,b}^{(j)}, W_i^{(j)} \in \{0,1\}^{k+1}$, $E \subset [\sigma]$, $y \in \{0,1\}^n$ for some $n \in \mathbb{N}$, and $out \in \{0,1\}$. Here, E denotes the set of unchecked instances.

We say that the receiver accepts if $out = 1$*, or rejects otherwise. A CCOT is secure if for any* $E \subset [\sigma]$ *such that* $|E| = c \cdot \sigma$ *for some* $0 < c < 1$*, it satisfies the following conditions:*

1. ***(Completeness)*** *If both parties are honest, the receiver accepts with probability 1.*
2. ***(OT Sender Security)*** *When ignoring private input* E *and private output* out*, this protocol is an OT secure in the presence of a malicious receiver.*
3. ***(Indistinguishability)*** *Let* $out(y)$ *denote the receiver's output* out *from running this protocol honestly while using input* y*. For all non-uniform probabilistic polynomial-time algorithm* S^* *(malicious sender), there exists negligible function* $\mu(\cdot)$ *such that for all* $k \in \mathbb{N}$*, and any* $y_0, y_1 \in \{0,1\}^n$*,* S^* *distinguishes the following distributions with probability at most* $\mu(k)$*:*
   - $\{out(y_0)\}_k$
   - $\{out(y_1)\}_k$
4. ***(Soundness)*** *We say that the* j*-th instance is* good *if the following conditions hold:*
   - *the sender's private input* $(W_{i,0}^{(j)}, W_{i,1}^{(j)})$ *is in the proper format, that is, for all* $i \in [n]$*,* $W_{i,0}^{(j)} = (K_{i,0}^{(j)}, \pi_i^{(j)})$ *and* $W_{i,1}^{(j)} = (K_{i,1}^{(j)}, 1 - \pi_i^{(j)})$ *for some* $(K_{i,0}^{(j)}, K_{i,1}^{(j)}, \pi_i^{(j)}) \in \{0,1\}^k \times \{0,1\}^k \times \{0,1\}$.
   - *the sender's private input* $(W_{i,0}^{(j)}, W_{i,1}^{(j)})$ *matches common input* $(\Omega_{i,0}^{(j)}, \Omega_{i,1}^{(j)})$*, that is, for all* $i \in [n]$*,* $b \in \{0,1\}$*, commitment* $\Omega_{i,b}^{(j)}$ *decommits to label* $W_{i,b}^{(j)}$*;*

   For all non-uniform probabilistic polynomial-time algorithm S^* *(malicious sender), there exists negligible function* $\mu(\cdot)$ *such that for all* $k \in \mathbb{N}$*, if the honest receiver accepts, the following conditions hold with probability at least* $1 - \mu(k)$*:*

   - ***(Soundness of the checked instances)*** *For all* $j \in [\sigma] \backslash E$*, the* j*-th instance is good.*
   - ***(Soundness of the unchecked instances)*** *There exists* $F \subset E$ *such that*

     (a) $|F| > |E|/2$*;*

(b) for all $j \in F$, the j-th instance is good; and

(c) for all $j \in F$, all $i \in [n]$, commitment $\Omega_{i,y_i}^{(j)}$ decommits to label $W_i^{(j)}$.

Remark 3.2. *Note that $W_{i,b}^{(j)}$ denotes the sender's private input, whereas $W_i^{(j)}$ denotes the receiver's private output. They are denoted similarly, but they are not necessarily related unless the soundness property holds.*

Remark 3.3. *We acknowledge that the name* Cut-and-Choose OT *is not new. Lindell and Pinkas have proposed a similar primitive [LP11]. However, their primitive is a stronger notion than ours. In particular, their primitive requires that the receiver learns $K_{i,y_i}^{(j)}$ for* all *$j \in E$, whereas ours requires that the receiver learns $K_{i,y_i}^{(j)}$ only for the* majority *of $j \in E$. We stress that this relaxation allows instantiations that do not make hardness assumptions in addition to OTs, whereas Lindell and Pinkas' instantiation relies on the Decisional Diffie-Hellman assumption [LP11].*

Remark 3.4. *Officially, this module should be referred to as* weak CCOT. *However, due to naming consistency, it will be referred to as CCOT in this thesis.*

3.3.3 Cut-and-Choose Consistent Transfer

Recall that carefully chosen inconsistent input keys from a malicious Gen could reveal more information about Eval's input than Gen is allowed to learn, which compromises Eval's input privacy. We therefore need to ensure that the semantics of Gen's input keys is consistent among different evaluation circuits.

At a high level, Gen and Eval run an interactive protocol such that in the end, Eval receives Gen's input keys along with a proof for Eval to check. This proof needs to satisfy the following two security properties. First and the most important, Eval should not learn any extra information about Gen's input from interacting with Gen. In other words, Eval's transcript from running the protocol should be independent of Gen's input. Second, if the proof is accepted, Gen's inputs keys to *most* of the evaluations circuits must have the same semantics. Similarly, by "most" we mean that this property holds for the majority of the evaluation circuits.

A technical difficulty similar to, but more complicated than, the one in Eval's input consistency check arises when dealing with Gen's input consistency. In order to construct a well-defined module, we again suggest that the commitments to both random keys assigned to Gen's input wire are made public in the beginning. Gen will later help Eval decommit one out of each pair of the commitments. However, because of the condition that Eval must not learn the semantics of the decommitted keys, which is Gen's input, the pursuit of the semantics consistency among different evaluation circuits is not as straightforward. Recall that in addition to a pair of random keys, a random bit is also assigned to each of the wires corresponding to Gen's input. This random bit is used to indicate whether the commitments to the random keys are swapped in order to prevent Eval from learning Gen's input by the location of the successfully decommitted key. However, those random bits in an evaluation circuit should be hidden from Eval. We therefore suggest that the commitments to these random bits become a common input too so that the semantics of Gen's input keys are well defined but in an oblivious manner. More specifically, let commitments $\{(\Theta^{(j)}_{i,0}, \Theta^{(j)}_{i,1}, \Pi^{(j)}_i)\}_{j\in[\sigma],i\in[n]}$ be common input. Label $W^{(j)}_i = (K^{(j)}_i, \delta^{(j)}_i)$ is *valid* if it is the decommitted value from $\Theta^{(j)}_{i,\delta^{(j)}_i}$, and a pair of valid labels $W^{(a)}_i = (K^{(a)}_i, \delta^{(a)}_i)$ and $W^{(b)}_i = (K^{(b)}_i, \delta^{(b)}_i)$ are said to *have consistent semantics* if $\delta^{(a)}_i \oplus \pi^{(a)}_i = \delta^{(b)}_i \oplus \pi^{(b)}_i$, where $\pi^{(a)}_i$ and $\pi^{(b)}_i$ are the decommitted bit from $\Pi^{(a)}_i$ and $\Pi^{(b)}_i$, respectively.

Next, we provide the formal definition of a primitive called *Cut-and-Choose Consistent Transfer*. We will suggest a couple of instantiations in a later chapter.

Definition 3.3. *A CCCT (Cut-and-Choose Consistent Transfer) is an interactive protocol between two probabilistic polynomial-time algorithms—sender S and receiver R. In this protocol, the common input includes security parameter 1^k, statistical security parameter 1^σ, and commitments*

$\{\Theta_{i,0}^{(j)}, \Theta_{i,1}^{(j)}, \Pi_i^{(j)}\}_{j\in[\sigma],i\in[n]}$. *Moreover, this protocol has the following functionality:*

$$CCCT : ((\{W_{i,0}^{(j)}, W_{i,1}^{(j)}\}_{j\in[\sigma],i\in[n]}, x), E) \mapsto (\bot, (\{W_i^{(j)}\}_{j\in E,i\in[n]}, out)),$$

where $W_{i,b}^{(j)}, W_i^{(j)} \in \{0,1\}^{k+1}$, $x \in \{0,1\}^n$ *for some* $n \in \mathbb{N}$, $E \subset [\sigma]$, *and* $out \in \{0,1\}$. *Here,* E *denotes the set of the uncheck instances. We say that the receiver accepts if* $out = 1$, *or rejects otherwise. A CCCT is secure if for any* $E \subset [\sigma]$ *such that* $|E| = c \cdot \sigma$ *for some* $0 < c < 1$, *it satisfies the following conditions:*

1. ***(Completeness)*** *If both parties are honest, the receiver accepts with probability 1.*

2. ***(Indistinguishability)*** *Let* $tran(x)$ *denote the receiver's transcript from running this protocol with the honest sender using input* x. *For all non-uniform probabilistic polynomial-time algorithm* R^* *(malicious receiver), there exists negligible function* $\mu(\cdot)$ *such that for all* $k \in \mathbb{N}$ *and any* $x_0, x_1 \in \{0,1\}^n$, R^* *distinguishes the following distributions with probability at most* $\mu(k)$:

   - $\{tran(x_0)\}_k$
   - $\{tran(x_1)\}_k$

3. ***(Soundness)*** *We say that the* j*-th instance is* good *if the following conditions hold:*

   - *the sender's private input* $(W_{i,0}^{(j)}, W_{i,1}^{(j)})$ *is in the proper format, that is, for all* $i \in [n]$, $W_{i,0}^{(j)} = (K_{i,0}^{(j)}, \pi_i^{(j)})$ *and* $W_{i,1}^{(j)} = (K_{i,1}^{(j)}, 1 - \pi_i^{(j)})$ *for some* $(K_{i,0}^{(j)}, K_{i,1}^{(j)}, \pi_i^{(j)}) \in \{0,1\}^k \times \{0,1\}^k \times \{0,1\}$.
   - *the sender's private input* $(W_{i,0}^{(j)}, W_{i,1}^{(j)})$ *matches common input* $(\Omega_{i,0}^{(j)}, \Omega_{i,1}^{(j)}, \Pi_i^{(j)})$, *that is, for all* $i \in [n]$, *commitment* $\Pi^{(j)}$ *decommits to permutation bit* $\pi_i^{(j)}$, *and for all* $i \in [n]$ *and* $b \in \{0,1\}$, *commitment* $\Omega_{i,b\oplus\pi_i^{(j)}}^{(j)}$ *decommits to label* $W_{i,b}^{(j)}$.

For all non-uniform probabilistic polynomial-time algorithm S^* (malicious sender), there exists negligible function $\mu(\cdot)$ such that for all $k \in \mathbb{N}$, if the honest receiver accepts, with probability at least $1 - \mu(k)$ the following conditions hold:

- ***(Soundness of the checked instances)*** *For all $j \in [\sigma] \backslash E$, the j-th instance is good.*
- ***(Soundness of the unchecked instances)*** *There exists $F \subset E$ such that*
  - *(a) $|F| > |E|/2$;*
  - *(b) for all $j \in F$, the j-th instance is good;*
  - *(c) for all $j \in F$ and $i \in [n]$, commitment $\Theta^{(j)}_{i,\delta^{(j)}_i}$ decommits to label $W^{(j)}_i$, where $W^{(j)}_i = (K^{(j)}_i, \delta^{(j)}_i)$ for some $K^{(j)}_i \in \{0,1\}^k$ and $\delta^{(j)}_i \in \{0,1\}$; and*
  - *(d) for all $j \in F$ and $i \in [n]$, $x_i = \delta^{(j)}_i \oplus \pi^{(j)}_i$.*

3.3.4 Cut-and-Choose Communication Channels

The cut-and-choose technique has two steps: *cut* and *reveal*. GEN first garbles multiple copies of the circuit, and then a random fraction is chosen. This fraction is called a cut. Next, GEN reveals the randomness of the cut for EVAL to check their correctness. The principle of this technique is that GEN should be oblivious of the cut while garbling the circuits and must not have any control over the choosing of the cut, either. Clearly, if this principle is violated, a malicious GEN could sneak in as many faulty circuits as it wants.

A popular instantiation of the cut-and-choose technique is based on a secure coin toss protocol [LP07, PSSW09, SS11, LP11]. After GEN constructs and delivers the garbled circuits, both parties jointly run a coin toss protocol, and the generated random bits are used to determine the cut. This approach works as long as the coin toss produces unbiased random bits even in the presence of malicious adversaries.

However, the coin toss has a couple of disadvantages. First, it adds several communication rounds to the original Yao protocol, which is less desirable. More importantly, this instantiation implicitly requires that EVAL stores all of the garbled circuits temporarily while both parties are running the coin toss. This assumption becomes impractical when the objective circuit is big, which is not uncommon for real-world applications. For example, a circuit that computes 1,024-bit RSA consists of tens of billions of gates. To run a secure computation of this circuit at the security level of 2^{-80} would require a machine that has memory at the terabyte scale in order to store all the garbled circuits temporarily. A machine at this scale can be too much to ask for general users.

Additionally, our framework has a security issue that needs to be handled carefully. Recall that our framework consists of three independent cut-and-choose modules: CCCG (Cut-and-Choose Circuit Garbling), CCOT (Cut-and-Choose Oblivious Transfer), and CCCT (Cut-and-Choose Consistent Transfer). We need to ensure that the cuts are consistent among these modules. Indeed, if a malicious EVAL has the j-th instance checked in CCCG but evaluated in CCCT, it will learn GEN's input. More specifically, recall that for each wire corresponding to GEN's input, it is assigned a random permutation bit, which serves as the one-time-pad to GEN's input in an evaluation circuit. For a checked instance in CCCG, EVAL will learn this bit; and for an evaluated instance in CCCT, EVAL will learn GEN's input one-time-padded with this bit. Therefore, a malicious EVAL learns GEN's input as long as it is capable of providing inconsistent cuts to CCCG and CCCT.

We propose the construction of the Cut-and-Choose Communication Channels in order to solve the above concerns. At a high level, GEN and EVAL jointly build two one-way (from GEN to EVAL) communication channels for each circuit: the *check channel* and the *evaluation channel*. These communication channels have the following two properties:

1. EVAL has access to at most one of the two channels per circuit.

2. Gen is oblivious of the channel to which Eval has access.

We next argue at a high level that these channels tackle the above issues.

- The security properties needed by the cut-and-choose technique can be provided by the cut-and-choose channels seamlessly. Recall that a garbled circuit can be either checked or evaluated. For a check circuit, Gen provides the randomness, and for an evaluation circuit, Gen provides its input keys. It is important that Eval does not learn both the randomness and Gen's input keys. It is also important that Gen does not know what Eval learns about the circuit. Given the cut-and-choose channels, Gen simply sends the randomness over the check channel, and provides its input labels via the evaluation channel. These channels ensure that Eval learns either the randomness or input labels, but never both. They also ensure that Gen is clueless about what Eval learns.
- With the help of the cut-and-choose communication channels, Gen and Eval can jointly garble and evaluate a circuit, even a huge one, with a moderate use of memory. The insight is that the cut can now be determined *before* the circuit garbling starts. At a high level, if the cut is determined before the circuit garbling, both parties can work together in a pipeline manner. Gen sends a garbled gate as soon as it is generated. While Eval is checking or evaluating a garbled gate, Gen could start garbling the next. As a result, rather than a whole garbled circuit, only a small amount of the garbled gates need to reside in memory at any moment. We will elaborate on this optimization technique in a later chapter when considering secure 2PC in practice.
- These cut-and-choose channels help ensure the cut consistency among CCCG, CCOT, and CCCT. Instead of allowing Eval choose cuts for these modules separately, Gen and Eval first jointly build the cut-and-choose channels. Later, Gen and Eval jointly run the protocols over these channels. Therefore, Eval does not have to choose a cut separately for each module, and as a result, has no capability of introducing inconsistent cuts.

Let CCOT(E) and CCCT(E) denote the CCOT and CCCT protocol, respectively, over the cut-and-choose channels for some $E \subset [\sigma]$ such that E is the set of unchecked instances.

It remains to show how the cut-and-choose communication channels are built. Naturally, $\binom{2}{1}$-OT comes into the picture. For the j-th circuit, both parties jointly run a $\binom{2}{1}$-OT such that Gen has key pair $(L_0^{(j)}, L_1^{(j)})$ as input and Eval has choice bit e_j as input. After the completion of the OT, whatever messages regarding the j-th circuit that Gen sends over the check channel is encrypted with $L_0^{(j)}$, and whatever sent over the evaluation channel is encrypted with $L_1^{(j)}$. The $\binom{2}{1}$-OT's sender security ensures that Eval learns only $L_{e_j}^{(j)}$, hence, has access to at most one of the two channels. Furthermore, the $\binom{2}{1}$-OT's receiver security ensures that Gen is oblivious of the channel to which Eval has access.

3.3.5 The Suggested Cut Size

In this section, we want to explore the best check circuit versus evaluation circuit ratio.

Suppose that Gen constructs σ copies of the garbled circuit that compute the objective function and sends them to Eval. According to the cut-and-choose strategy, Eval asks for Gen's input keys for e evaluation circuits and the randomness for $(\sigma - e)$ check circuits. After the correctness of the check circuits are verified, Eval proceeds to evaluate the remaining e copies of the circuits and takes the majority output as the final output. The following Lemma suggests that the best check circuit versus evaluation circuit ratio is 3:2 instead of the popular 1:1.

Lemma 3.2 (Best Cut-and-Choose Strategy). *Suppose that the cut-and-choose technique is adopted and* Gen *garbles σ coplies of the circuit. The probability that* Gen *sneaks in a faulty majority into the evaluation circuits without getting caught is minimized when* Eval *picks a cut of size $(\sigma - e)$, where either $e = \frac{1}{5}\left(\sigma - 7 + \sqrt{(\sigma-7)^2 - 40}\right)$ or $e = \frac{1}{5}(2\sigma - 9)$.*

Proof. Let b be the number of bad circuits created by GEN. A circuit is *bad* if either the circuit does not compute the objective function or EVAL's inputs are selectively failed via OT. The goal is to minimize the probability that a malicious GEN sneaks in faulty majority to the evaluation circuits with out getting caught. In other words, we want to find e and b that minimize $\binom{\sigma-b}{\sigma-e}/\binom{\sigma}{\sigma-e}$, that is, the probability that all the $(\sigma-e)$ check circuits are good assuming that there are $(\sigma-b)$ good circuits.

We first claim that GEN's best cheating strategy is to produce $b=\lfloor e/2\rfloor+1$ bad circuits. Indeed, if $b\le\lfloor e/2\rfloor$, the final output will not get affected since the faulty outputs will be overwhelmed by majority good ones. On the other hand, the more bad circuits, the more likely that GEN will get caught since $\binom{\sigma-(b-1)}{\sigma-e}>\binom{\sigma-b}{\sigma-e}$. So the best strategy for GEN to succeed in cheating is to construct as few bad circuits as possible while the majority of evaluation circuits are bad, which justifies the choice of b.

Our next goal is to find the e that minimizes $F(e)=\binom{\sigma-\lfloor\frac{e}{2}\rfloor-1}{\sigma-e}/\binom{\sigma}{\sigma-e}$. To get rid of the troublesome floor function, we will consider the case when e is even and odd separately.

- If $e=2n$ for some $n\in\mathbb{N}$ such that $n\le\frac{\sigma}{2}$, let $F_{\text{even}}(n)=\binom{\sigma-n-1}{\sigma-2n}/\binom{\sigma}{\sigma-2n}$. Observe that

$$\frac{F_{\text{even}}(n+1)}{F_{\text{even}}(n)}=\frac{(2n+1)(2n+2)}{n\cdot(\sigma-n-1)}.$$

It is not hard to solve a quadratic inequality and come to the conclusion that

$$\frac{F_{\text{even}}(n+1)}{F_{\text{even}}(n)}<1 \text{ when } 0<n<\tfrac{1}{10}\left(\sigma-7+\sqrt{(\sigma-7)^2-40}\right)\overset{\text{def}}{=}\alpha.$$

In other words, $F_{\text{even}}(\cdot)$ is monotonically decreasing when $0<n<\alpha$ and monotonically increasing when $\alpha\le n\le\frac{\sigma}{2}$. Therefore, $F_{\text{even}}(\cdot)$ is minimal when $n=\lceil\alpha\rceil$.

- If $e = 2n+1$, let $F_{\text{odd}}(n) = \binom{\sigma-n-1}{\sigma-2n-1} / \binom{\sigma}{\sigma-2n-1}$. By a similar computation, we know that

$$\frac{F_{\text{odd}}(n+1)}{F_{\text{odd}}(n)} = \frac{(2n+2)(2n+3)}{(n+1)(\sigma-n-1)} < 1 \text{ when } n < \frac{\sigma-7}{5} \stackrel{\text{def}}{=} \beta.$$

  Hence, $F_{\text{odd}}(\cdot)$ is minimal when $n = \lceil \beta \rceil$.

In summary, $F(\cdot)$ has the minimum when either $e = 2\lceil \alpha \rceil$ or $e = 2\lceil \beta \rceil + 1$. □

Remark 3.5. *If σ is sufficiently large, $F(\cdot)$ will have the minimum when $e \approx \frac{2}{5}\sigma$, which implies that the check circuit versus evaluation circuit ratio is roughly $(\sigma - \frac{2}{5}\sigma) : \frac{2}{5}\sigma = 3:2$.*

3.4 Main Protocol

Before we give the full description of our main framework, we would like to point out a few assumptions that we made:

- Our framework considers only those objective functions that allow Eval to receive the output. However, we are aware of the importance and need of two-output objective functions to real-world applications. We will present several solutions to secure 2PC over two-output functions in a later chapter.
- In the Main protocol, we often say that "[someone] uses randomness ρ to compute [something]." We implicitly assume that this algorithm that computes the random bits needed from randomness ρ is agreed upon in advance by both parties.
- Both parties have agreed in advance upon a symmetric encryption scheme (enc, dec) with semantic security, a perfectly-hiding commitment scheme com, a protocol for $\binom{2}{1}$-OT secure in the presence of malicious adversaries, a secure CCOT, and a secure CCCT.

Next, we put together all the modules to form the main protocol of our framework.

Protocol 3.2. The Main Protocol

Common Input: security parameter 1^k, statistical security parameter 1^σ, and boolean circuit C that computes $f(x, y)$.

Private Input: GEN has input $x = x_1 x_2 \ldots x_{n_1}$ and EVAL has input $y = y_1 y_2 \ldots y_{n_2}$, where x_i and y_i denote the i-th bit of x and y, respectively.

Private Output: EVAL receives output $f(x, y)$.

1. **(Circuit OTs)** Both parties jointly execute σ instances of the $\binom{2}{1}$-OT. More specifically, EVAL first randomly picks $E \subset [\sigma]$ such that $|E| = 2\sigma/5$. EVAL then computes $e_j \leftarrow 1$ if $j \in E$, or $e_j \leftarrow 0$ otherwise, while GEN randomly picks $L_0^{(j)}, L_1^{(j)} \xleftarrow{R} \{0,1\}^k$ so that in the j-th instance of the $\binom{2}{1}$-OTs, both parties jointly run

   $$\binom{2}{1}\text{-OT} : ((L_0^{(j)}, L_1^{(j)}), e_j) \mapsto (\perp, L^{(j)}).$$

   Either party aborts if any of the $\binom{2}{1}$-OTs fails. Both parties will use $\{L_0^{(j)}, L_1^{(j)}\}_{j \in [\sigma]}$ and $\{L^{(j)}\}_{j \in [\sigma]}$ to construct the cut-and-choose channels.

   Remark 3.6. *The above $\binom{2}{1}$-OTs could run in parallel if they provide security for parallel execution. If no one aborts, the security properties of the $\binom{2}{1}$-OT ensure that $L^{(j)} = L_{e_j}^{(j)}$.*

2. **(Pick the randomness)** For all $j \in [\sigma]$, GEN picks randomness $\rho^{(j)}$ for the j-th garbled circuit, and then sends $\rho^{(j)}$ over the check channel.

   Remark 3.7. *Randomness $\rho^{(j)}$ can be considered as either a pool of truly random bits or a truly random seed to a pseudo-random number generator. It has internal states that keep track of old and fresh random bits, and it always returns fresh ones when used.*

3. **(Pick random label pairs)** For all $j \in [\sigma]$ and every wire w_i, Gen uses randomness $\rho^{(j)}$ to compute $(K_{i,0}^{(j)}, K_{i,1}^{(j)}, \pi_i^{(j)}) \in \{0,1\}^k \times \{0,1\}^k \times \{0,1\}$. Let $W_{i,b}^{(j)}$ denote the key-locator pair $(K_{i,b}^{(j)}, b \oplus \pi_i^{(j)})$. $W_{i,b}^{(j)}$ is called the *label* corresponding to wire w_i of value b in the j-th garbled circuit hereafter.

4. **(Commit to input label pairs)** Let $\{w_i\}_{i\in[n_1]}$ be the wires corresponding to Gen's input and $\{w_{n_1+i}\}_{i\in[n_2]}$ be the wires corresponding to Eval's input. Gen uses randomness $\rho^{(j)}$ to commit to the label pairs assigned to $\{w_i\}_{i\in[n_1+n_2]}$ and random permutation bits assigned to $\{w_i\}_{i\in[n_1]}$ by sending $\{\Theta^{(j)}, \Omega^{(j)}, \Pi^{(j)}\}_{j\in[\sigma]}$ to Eval, where

$$\Theta^{(j)} = \{\Theta_{i,0}^{(j)}, \Theta_{i,1}^{(j)}\}_{i\in[n_1]} = \{\mathsf{com}(W^{(j)}_{i,0\oplus\pi_i^{(j)}}), \mathsf{com}(W^{(j)}_{i,1\oplus\pi_i^{(j)}})\}_{i\in[n_1]};$$

$$\Omega^{(j)} = \{\Omega_{n_1+i,0}^{(j)}, \Omega_{n_1+i,1}^{(j)}\}_{i\in[n_2]} = \{\mathsf{com}(W^{(j)}_{n_1+i,0}), \mathsf{com}(W^{(j)}_{n_1+i,1})\}_{i\in[n_2]};$$

$$\Pi^{(j)} = \{\Pi_i^{(j)}\}_{i\in[n_1]} = \{\mathsf{com}(\pi_i^{(j)})\}_{i\in[n_1]}.$$

Remark 3.8. Gen *commits to the label pairs assigned to* Gen*'s and* Eval*'s input wires so that when* Eval *later receives proper decommitments, it will know that the decommitted labels are valid. These commitments plus the commitments to permutaiton bits serve as the common ground for CCCG, CCOT, and CCCT.*

Moreover, the commitment pairs corresponding to Gen*'s input wires need to be randomly swapped so that each label's semantics is independent of its location. In other words, the location of a successfully decommitted label will not disclose* Gen*'s input to* Eval*. This random swap is done by reusing the permutation bit $\pi_i^{(j)}$ that is assigned to each wire and used to permute entries in garbled truth tables. In contrast, the commitment pairs corresponding to* Eval*'s input wires need to follow a known order so that* Eval *can verify that the semantics of the successfully decommitted labels actually matches its input. As a result, the commitment pairs in $\Theta^{(j)}$ are randomly swapped so that the commitment to b-label of the i-th wire is the $(2\cdot i + b \oplus \pi_i^{(j)})$-th entry, whereas in $\Omega^{(j)}$, the commitment to b-label of the i-th wire is the $(2\cdot i + b)$-th.*

5. **(Retrieve Eval's Input Labels)** With common input security parameter 1^k, statistical security parameter 1^σ, and commitments $\{\Omega_{i,0}^{(j)}, \Omega_{i,1}^{(j)}\}_{j\in[\sigma],i\in[n_1]}$, both parties jointly run CCOT over the cut-and-choose channels, that is,

$$\mathsf{CCOT}(E) : (\{W_{n_1+i,0}^{(j)}, W_{n_1+i,1}^{(j)}\}_{j\in[\sigma],i\in[n_2]}, y) \mapsto (\perp, (\{W_{n_1+i}^{(j)}\}_{j\in E,i\in[n_2]}, \mathsf{out})).$$

Eval aborts if $\mathsf{out} = 0$.

6. **(Retrieve Gen's Input Labels)** With common input security parameter 1^k, statistical security parameter 1^σ, and commitments $\{\Theta_{i,0}^{(j)}, \Theta_{i,1}^{(j)}, \Pi_i^{(j)}\}_{j\in[\sigma],i\in[n_1]}$, both parties jointly run CCCT over the cut-and-choose channels, that is,

$$\mathsf{CCCT}(E) : ((\{W_{i,0}^{(j)}, W_{i,1}^{(j)}\}_{j\in[\sigma],i\in[n_1]}, x), \perp) \mapsto (\perp, (\{W_i^{(j)}\}_{j\in E,i\in[n_1]}, \mathsf{out})).$$

Eval aborts if $\mathsf{out} = 0$.

7. **(Retrieve Garbled Circuits)** For each gate $g : \{0,1\} \times \{0,1\} \mapsto \{0,1\}$ with input wires w_a and w_b and output wire w_c, Gen garbles g by computing

$$G(g)^{(j)} \leftarrow (\langle \pi_a^{(j)}, \pi_b^{(j)} \rangle, \langle \pi_a^{(j)}, 1 \oplus \pi_b^{(j)} \rangle, \langle 1 \oplus \pi_a^{(j)}, \pi_b^{(j)} \rangle, \langle 1 \oplus \pi_a^{(j)}, 1 \oplus \pi_b^{(j)} \rangle),$$

where $\langle d_1, d_2 \rangle = \mathsf{enc}_{K_{a,d_1}^{(j)}}(\mathsf{enc}_{K_{b,d_2}^{(j)}}(W_{c,g(d_1,d_2)}^{(j)}))$. Let $\{w_i\}_{i\in O}$ be the circuit-output wires. Gen then sends $\{G(C)^{(j)}\}_{j\in[\sigma]}$ to Eval, where

$$G(C)^{(j)} = \left(\{G(g)^{(j)}\}_{g\in C}, \{\pi_i^{(j)}\}_{i\in O} \right).$$

Remark 3.9. *Note that once $(K_{i,0}^{(j)}, K_{i,1}^{(j)}, \pi_i^{(j)})$ are chosen for every wire w_i in Step 3, no more randomness is needed for $G(C)^{(j)}$. So randomness $\rho^{(j)}$ is not used here.*

1

8. **(Check Circuits)** For each $j \in [\sigma]\backslash E$, Eval checks if randomness $\rho^{(j)}$ received in Step 2 can regenerate commitments $\{\Theta^{(j)}, \Omega^{(j)}, \Pi^{(j)}\}$ received in Step 4 and reconstruct garbled circuit $G(C)^{(j)}$. Eval aborts if any of the checks fails.

9. **(Evaluate Circuits)** For each $j \in E$, Eval has now obtained garbled circuit $G(C)^{(j)}$ and $(n_1 + n_2)$ labels corresponding to the circuit-input wires of C. Eval evaluates these (evaluation) circuits as follows:

   (a) For each gate g with retrieved input labels $W_a^{(j)} = (K_a^{(j)}, \delta_a^{(j)})$ and $W_b^{(j)} = (K_b^{(j)}, \delta_b^{(j)})$, Eval picks the $(2 \cdot \delta_a^{(j)} + \delta_b^{(j)})$-th entry T in garbled truth table $G(g)^{(j)}$ and computes output label $W_c^{(j)} \leftarrow \mathsf{dec}_{K_b^{(j)}}(\mathsf{dec}_{K_a^{(j)}}(T))$.

   (b) For each circuit-output wire w_i with corresponding label $W_i^{(j)} = (K_i^{(j)}, \delta_i^{(j)})$, Eval computes wire value $b_i^{(j)} \leftarrow \delta_i^{(j)} \oplus \pi_i^{(j)}$.

   (c) Eval interprets $\{b_i^{(j)}\}_{i \in O}$ as $f^{(j)}(x, y)$.

10. **(Majority Operation)** Let $f(x,y)$ be the most common value in multiset $\{f^{(j)}(x,y)\}_{j \in E}$. Eval aborts if $f(x,y)$ is not the majority in Σ, that is, $f(x,y)$ does not appear more than $|E|/2$ times in Σ; otherwise, Eval outputs $f(x,y)$. On the other side, Gen always outputs $\perp$.

Theorem 3.3. *Assume that the $\binom{2}{1}$-OT protocol is secure in the presence of malicious adversaries, and there exist a perfectly-hiding commitment scheme, a semantically secure symmetric encryption scheme, a secure CCOT, and a secure CCCT. Protocol 3.2 securely computes polynomial-time function $f : \{0,1\}^{n_1} \times \{0,1\}^{n_2} \mapsto \{\perp\} \times \{0,1\}^*$ in the presence of malicious adversaries.*

Remark 3.10. *Due to the Selective Decommitment Attack, the commitment scheme* com *needs to be perfectly-hiding so that the unopened commitments remain hiding [Hof11].*

3.5 Security Proof for the Main Protocol

To simplify the notations and clearly identify malicious parties, we use P_1 to denote the honest Gen and P_1^* to denote an arbitrary cheating Gen. Similarly, we use P_2 to denote the honest Eval and P_2^* to denote an arbitrary cheating Eval. To formally prove Theorem 3.3, we need to construct two non-uniform probabilistic expected polynomial-time simulators S_1 and S_2 in the ideal model against P_1^* and P_2^* in the real model, respectively.

3.5.1 For Malicious Generator

Intuition: First of all, without loss of generality, we assume that P_1^* always provides valid keys in the circuit OTs. Conceivably, P_1^* could use $(L_0^{(j)}, L_1^{(j)})$ in a circuit OT while using $(L_0'^{(j)}, L_1'^{(j)})$ to build the corresponding cut-and-choose channels, where $L_b'^{(j)}$ may or may not be equal to $L_b^{(j)}$ for $b \in \{0, 1\}$. In the case that $L_0^{(j)} \neq L_0'^{(j)}$, all the messages transmitted over the check channel cannot be properly received. This is equivalent to that the j-th circuit fails to be a check circuit. Similarly, in the case of $L_1^{(j)} \neq L_1'^{(j)}$, it is equivalent to that the j-th circuit fails to be an evaluation circuit. As a result, even if P_1^* provides invalid keys in the j-th instance of the circuit OTs, we can always think of it as P_1^* provides valid keys in the OT but cheats in the corresponding garbled circuit.

Next, we observe that P_1^* can only cheat in the following three ways: (1) provide malicious labels for P_2's input; (2) provide malicious labels for its own input; and (3) provide faulty circuits. Let $F_1, F_2, F_3 \subset [\sigma]$ denote the set of the evaluation circuits in which P_1^* *does not* cheat in the above three ways, respectively. Note that CCOT, CCCT, and CCCG ensure that if P_2 does not abort before the step of circuit evaluation, then $|F_i| > |E|/2$ for $i = 1, 2, 3$. The insight is that $|F_1 \cap F_2 \cap F_3| > |E|/2$, too, with high probability. Indeed, since set E of evaluation instances are consistent among these three building blocks, $|F_1 \cap F_2 \cap F_3| \leq |E|/2$ implies that P_1^* cheats in at least one of the above three ways in more than $|E|/2$ garbled circuits, and none of them is checked. The probability of this event is negligible in σ as long as GEN is oblivious of E, which is guaranteed by the receiver's security in the circuit OTs. In other words, with high probability the majority of the evaluation circuits are honest execution of the Yao protocol. As a consequence, if P_2 does not abort before the step of circuit evaluation, the final majority operation will give P_2 output $f(x, y)$.

We now proceed to the formal proof. Protocol 3.3 shows how simulator S_1 works.

Protocol 3.3. The Simulator for an Arbitrary Malicious Generator P_1^*

Common Input: security parameter 1^k, statistical security parameter 1^σ, and boolean circuit C that computes $f(x, y)$.

Private Input: P_1^* has private input x but uses x' (which will be extracted) in the protocol instead. Note that x' may or may not equal x.

1. **(Circuit OTs)** S_1 randomly picks $E \subset [\sigma]$ such that $|E| = 2\sigma/5$ and computes $e_j \leftarrow 1$ if $j \in E$; or $e_j \leftarrow 0$ otherwise. In the j-th instance of the $\binom{2}{1}$-OTs, S_1 invokes externally the simulator that comes with the $\binom{2}{1}$-OT's receiver security and extracts both P_1^*'s input, that is, $L_0^{(j)}$ and $L_1^{(j)}$. Upon request, S_1 provides e_j as input to the external simulator. If the simulation does not fail, S_1 uses $L_{e_j}^{(j)}$ to construct the cut-and-choose channels for all $j \in [\sigma]$.

   Remark 3.11. *Since S_1 learns both $L_0^{(j)}$ and $L_1^{(j)}$, it gets to retrieve information sent over both the check channel and the evaluation channel for the j-th circuit.*

2. **(Pick the randomness)** For all $j \in [\sigma]$, S_1 use $L_0^{(j)}$ to retrieve $\rho^{(j)}$, even though $\rho^{(j)}$ is sent over the check channel meant only for a check circuit in a real-model execution.

3. **(Pick random label pairs)** For all $j \in [\sigma]$ and every wire w_i, S_1 uses randomness $\rho^{(j)}$ to compute $(K_{i,0}^{(j)}, K_{i,1}^{(j)}, \pi_i^{(j)}) \in \{0,1\}^k \times \{0,1\}^k \times \{0,1\}$. Let label $W_{i,b}^{(j)}$ denote key-locator pair $(K_{i,b}^{(j)}, b \oplus \pi_i^{(j)})$ hereafter.

4. **(Commit to input label pairs)** S_1 receives commitments $\{\Theta^{(j)}, \Omega^{(j)}, \Pi^{(j)}\}_{j \in [\sigma]}$.

5. **(Retrieve Eval's Input Labels)** S_1 uses $y = 0^{n_2}$ as its private input. With common input security parameter 1^k, statistical security parameter 1^σ, and commitments 1

$\{\Omega^{(j)}_{n_1+i,0}, \Omega^{(j)}_{n_1+i,1}\}_{j\in[\sigma],i\in[n_2]}$, both parties jointly run CCOT over the cut-and-choose channels, that is,

$$\mathsf{CCOT}(E) : (\{V^{(j)}_{n_1+i,0}, V^{(j)}_{n_1+i,1}\}_{j\in[\sigma],i\in[n_2]}, y) \mapsto (\perp, (\{V^{(j)}_{n_1+i}\}_{j\in E,i\in[n_2]}, \mathsf{out})).$$

If $\mathsf{out} = 0$, S_1 sends $\perp$ to external trusted party O. Besides, S_1 uses randomness $\{\rho^{(j)}\}_{j\in[\sigma]}$ to verify the soundness of the CCOT. More specifically, let N_1 denote the check instances that are good and F_1 denote the set of the unchecked instances with honest EVAL's input labels, that is,

$$N_1 = \left\{ j \in [\sigma]\backslash E \;\middle|\; \begin{array}{l} \forall i \in [n_2], b \in \{0,1\}, \\ \quad \Omega^{(j)}_{n_1+i,b} = \mathsf{com}(W^{(j)}_{n_1+i,b}) \end{array} \right\} \text{ and}$$

$$F_1 = \left\{ j \in E \;\middle|\; \begin{array}{l} \forall i \in [n_2], b \in \{0,1\}, \\ \quad \Omega^{(j)}_{n_1+i,b} = \mathsf{com}(W^{(j)}_{n_1+i,b}) \wedge V^{(j)}_{n_1+i} = W^{(j)}_{n_1+i,y_i} \end{array} \right\},$$

where label $W^{(j)}_{n_1+i,b} = (K^{(j)}_{n_1+i,b}, b \oplus \pi^{(j)}_{n_1+i})$ is computed from randomness $\rho^{(j)}$ in Step 3. S_1 sends $\perp$ to O if $N_1 \neq [\sigma]\backslash E$ or $|F_1| \le |E|/2$.

6. **(Retrieve GEN's Input Labels)** With common input security parameter 1^k, statistical security parameter 1^σ, and commitments $\{\Theta^{(j)}_{i,0}, \Theta^{(j)}_{i,1}, \Pi^{(j)}_i\}_{j\in[\sigma],i\in[n_1]}$, both parties jointly run CCCT over the cut-and-choose channels, that is,

$$\mathsf{CCCT}(E) : ((\{V^{(j)}_{i,0}, V^{(j)}_{i,1}\}_{j\in[\sigma],i\in[n_1]}, x), \perp) \mapsto (\perp, (\{V^{(j)}_i\}_{j\in E,i\in[n_1]}, \mathsf{out})).$$

If $\mathsf{out} = 0$, S_1 sends $\perp$ to O. Otherwise, S_1 uses $\{\rho^{(j)}\}_{j\in[\sigma]}$ to extract the semantics of $\{V^{(j)}_i\}_{j\in E,i\in[n_1]}$ and check the CCCT's soundness. Let $x^{(j)} \in \{0,1\}^{n_1}$ denote the extracted input from the j-th instance and let x' be the most common value in multiset $\{x^{(j)}\}_{j\in E}$. Moreover, let N_2 denote the set of the unchecked instances that are good, and F_2 denote the set of the unchecked instances with honest GEN's input labels, that is,

$$N_2 = \left\{ j \in [\sigma]\backslash E \;\middle|\; \begin{array}{r} \forall i \in [n_1], b \in \{0,1\}, \\ \Pi_i^{(j)} = \mathsf{com}(\pi_i^{(j)}) \\ \Theta_b^{(j)} = \mathsf{com}(W_{i,b\oplus\pi_i^{(j)}}^{(j)}) \end{array} \right\} \text{ and}$$

$$F_2 = \left\{ j \in E \;\middle|\; \begin{array}{r} \forall i \in [n_1], b \in \{0,1\}, \\ \Pi_i^{(j)} = \mathsf{com}(\pi_i^{(j)}) \\ \Theta_{i,b}^{(j)} = \mathsf{com}(W_{i,b\oplus\pi_i^{(j)}}^{(j)}) \wedge V_i^{(j)} = W_{i,x_i'}^{(j)} \end{array} \right\},$$

where label $W_{i,b}^{(j)} = (K_{i,b}^{(j)}, b \oplus \pi_i^{(j)})$ is computed from randomness $\rho^{(j)}$ in Step 3. S_1 sends $\perp$ to O if $N_2 \neq [\sigma]\backslash E$ or $|F_2| \leq |E|/2$.

7. **(Retrieve Garbled Circuits)** S_1 receives garbled circuits $\{G(C)^{(j)}\}_{j\in[\sigma]}$.

8. **(Check Circuits)** We say that the j-th circuit is honest if commitments $(\Theta^{(j)}, \Omega^{(j)}, \Pi^{(j)})$ received in Step 4 and garbled circuit $G(C)^{(j)})$ received in Step 7 can be properly regenerated from $\rho^{(j)}$. S_1 uses randomness $\{\rho^{(j)}\}_{j\in[\sigma]}$ to check the correctness of the received garbled circuits. More specifically, let $N_3 \subset [\sigma]\backslash E$ be the set of the checked circuit that are honest and $F_3 \subset E$ be the set of the evaluation circuits that are honest. S_1 sends $\perp$ to O if $N_3 \neq [\sigma]\backslash E$ or $F_3 \leq |E|/2$.

9. **(Evaluate Circuits)** If $|F_1 \cap F_2 \cap F_3| \leq |E|/2$, S_1 sends $\perp$ to O; otherwise, S_1 sends x'.

10. **(Majority Operation)** S_1 outputs whatever P_1^* outputs.

Lemma 3.4 (Security against a Malicious Generator). *Let* $\Pi = (\text{GEN}, \text{EVAL})$ *denote Protocol 3.2. Assume that the* $\binom{2}{1}$*-OT protocol is secure in the presence of malicious adversaries, and there exist a perfectly-hiding commitment scheme, a semantically secure symmetric encryption scheme, a secure CCOT, and a secure CCCT.*

For any single-output function $f : \{0,1\}^{n_1} \times \{0,1\}^{n_2} \mapsto \{\bot\} \times \{0,1\}^*$, and any non-uniform probabilistic strict polynomial-time machine P_1^*, simulator S_1 presented in Protocol 3.3 is a non-uniform probabilistic expected polynomial-time machine satisfying that

$$\left\{\mathbf{Real}_{\Pi,\bar{P}}(x,y,1^k)\right\}_{x,y,k} \approx_c \left\{\mathbf{Ideal}_{f,\bar{S}}(x,y,1^k)\right\}_{x,y,k},$$

where $\bar{P} = (P_1^*, P_2)$ and $\bar{S} = (S_1, S_2)$. Note that S_2 is honest as described in Definition 2.5.

Remark 3.12. *For brevity, subscript "x,y,k" denotes that both distributions range over all $x \in \{0,1\}^{n_1}$, all $y \in \{0,1\}^{n_2}$, and all $k \in \mathbb{N}$.*

Proof. Recall that in this case, P_2 follows the Main protocol faithfully, and S_2 always provides y to the trusted party and responds with 1 after getting its result. To prove this lemma, consider the following hybrid experiments and lemmas. Note that for the i-th hybrid experiment, we denote by H_i the algorithm that plays the role of Eval and by $\mathbf{Hybrid}_i(x,y,1^k)$ the joint output.

$\mathbf{Hybrid}_1(x,y,1^k)$: H_1 is the same as P_2 except that in all the σ instances of the circuit OT, instead of running as a real-model OT receiver, H_1 invokes ideal-model OT simulator S_1^{OT}, the existence of which is guaranteed by OT's receiver security. In particular, for all $j \in [\sigma]$, H_1 invokes externally simulator S_1^{OT} to extract both P_1^*'s inputs, $L_0^{(j)}$ and $L_1^{(j)}$, and provides e_j as input upon the request from simulator S_1^{OT}.

Lemma 3.5. $\left\{\mathbf{Real}_{\Pi,\bar{P}}(x,y,1^k)\right\}_{x,y,k} \overset{c}{\approx} \left\{\mathbf{Hybrid}_1(x,y,1^k)\right\}_{x,y,k}$

Proof. Intuitively, this lemma follows OT's receiver security. Formally, suppose that there exists non-uniform probabilistic polynomial-time distinguisher D such that for infinitely many k, D distinguishes **Real** from $\mathbf{Hybrid}_1$ with probability at least $1/p(k)$ for some polynomial $p(\cdot)$. Note that we can think of D as a potential P_1^* that attempts to compromise the security. In particular, D plays the role of Gen in both **Real** and $\mathbf{Hybrid}_1$ to interact with P_2 and H_1, respectively. Upon completion, D outputs a decision bit indicating in which experiment it was working.

We will show another non-uniform probabilistic polynomial-time machine A that breaks the OT's receiver security with black-box access to D. Again, we consider A as a malicious sender that interacts with either a real-model OT receiver or an ideal-model OT simulator such that upon completion, A outputs a decision bit indicating in which model it was working.

Let machine $H_{0,j}$ be the same as P_2 except that for the first j instances of the circuit OT, $H_{0,j}$ invokes simulator S_1^{OT} to simulate the sender's view. Note that $H_{0,0} = P_2$ and $H_{0,\sigma} = H_1$. Since D distinguishes **Real** from **Hybrid**$_1$ with probability at least $1/p(k)$, by averaging, there exists some j such that D distinguishes **Hybrid**$_{0,j-1}$ and **Hybrid**$_{0,j}$ with probability at least $1/(\sigma \cdot p(k))$. Although j is unknown, it can be guessed with probability $1/\sigma$. Recall that σ is also assumed to be a polynomial in k.

A can therefore be constructed as follows: A first randomly picks $j \overset{R}{\leftarrow} [\sigma]$. A then internally runs as $H_{0,j-1}$ with D except in the j-th instance of the circuit OTs, A forwards D's message to an external OT receiver, and vice versa. After this OT, A continues to run as $H_{0,j-1}$ with D. In the end, A outputs the decision bit that D outputs. We know that if the external receiver is a real-model OT receiver, D sees **Hybrid**$_{0,j-1}$; and if the external receiver is an ideal-model OT simulator, D sees **Hybrid**$_{0,j}$. Therefore, the statement that D distinguishes **Hybrid**$_{0,j-1}$ from **Hybrid**$_{0,j}$ with probability at least $1/(\sigma \cdot p(k))$ implies that A distinguishes the view in a real OT execution from that in the ideal OT simulation with probability at least $1/(\sigma^2 \cdot p(k))$, which contradicts the assumption of OT's receiver security. □

Remark 3.13. *We stress that the successful simulation of P_1^*'s view indicates that P_1^* is oblivious of e_j for all $j \in [\sigma]$. In other words, if the circuit OTs are successfully simulated, P_1^* does not know E, the set of the unchecked instances.*

Hybrid$_2(x, y, 1^k)$**:** H_2 is the same as H_1 except that H_2 sends $\perp$ to O if H_2 accepts the CCOT but the CCOT's soundness cannot be verified with randomness $\{\rho^{(j)}\}_{j\in[\sigma]}$. More specifically, recall that for *all* $j \in [\sigma]$, H_2 learned randomness $\rho^{(j)}$ since the OT simulator extracts both $L_0^{(j)}$ and $L_1^{(j)}$, the keys used to build the cut-and-choose communication channels. H_2 first uses randomness $\rho^{(j)}$ to compute $(K_{i,0}^{(j)}, K_{i,1}^{(j)}, \pi_i^{(j)}) \in \{0,1\}^k \times \{0,1\}^k \times \{0,1\}$ for wires $\{w_{n_1+i}\}_{i\in[n_2]}$ and all $j \in [\sigma]$. Next, H_2 verifies the CCOT's soundness as follows:

- H_2 computes N_1, the set of the checked instances that are good, that is,

$$N_1 = \left\{ j \in [\sigma]\backslash E \;\middle|\; \begin{array}{r} \forall i \in [n_2], b \in \{0,1\}, \\ \Omega_{n_1+i,b}^{(j)} = \mathsf{com}(W_{n_1+i,b}^{(j)}) \end{array} \right\},$$

  where $W_{n_1+i,b}^{(j)} = (K_{n_1+i,b}^{(j)}, b \oplus \pi_{n_1+i}^{(j)})$.

- H_2 computes F_1, the set of the unchecked instances that are good and have honest Eval's input labels, that is,

$$F_1 = \left\{ j \in E \;\middle|\; \begin{array}{r} \forall i \in [n_2], b \in \{0,1\}, \qquad\qquad\qquad\qquad \\ \Omega_{n_1+i,b}^{(j)} = \mathsf{com}(W_{n_1+i,b}^{(j)}) \wedge V_{n_1+i}^{(j)} = W_{n_1+i,y_i}^{(j)} \end{array} \right\},$$

  where $W_{n_1+i,b}^{(j)} = (K_{n_1+i,b}^{(j)}, b \oplus \pi_{n_1+i}^{(j)})$ is computed from randomness $\rho^{(j)}$ and $V_{n_1+i}^{(j)}$ is H_2's private output from the CCOT.

Recall that for a secure CCOT, the soundness for the checked instances ensures that $N_1 = [\sigma]\backslash E$ and the soundness for the unchecked instances ensures that $|F_1| > |E|/2$. So H_2 checks these two properties and sends $\perp$ to O if either $N_2 \neq [\sigma]\backslash E$ or $|F_1| \leq |E|/2$.

Lemma 3.6. $\left\{\mathbf{Hybrid}_1(x, y, 1^k)\right\}_{x,y,k} \overset{c}{\approx} \left\{\mathbf{Hybrid}_2(x, y, 1^k)\right\}_{x,y,k}$

Proof. At a high level, this lemma holds due to the CCOT's soundness property. In particular, if H_1 rejects the CCOT, H_2 rejects too. These two experiments differ when H_1 accepts but H_2 rejects, that is, when P_1^* corrupts the labels assigned to Eval's input wires in more than $|E|/2$ instances while being able to prove the opposite in the CCOT. The soundness property ensures that this only happens with negligible probability. □

Hybrid$_3(x, y, 1^k)$**:** H_3 is the same as H_2 except that H_3 sends $\perp$ to O if H_3 accpets the CCCT but the CCCT's soundness cannot be verified with randomness $\{\rho^{(j)}\}_{j\in[\sigma]}$. More specifically, for all $j \in [\sigma]$ and all wires $\{w_i\}_{i\in[n_1]}$, H_3 first uses randomness $\rho^{(j)}$ to compute $(K_{i,0}^{(j)}, K_{i,1}^{(j)}, \pi_i^{(j)}) \in \{0,1\}^k \times \{0,1\}^k \times \{0,1\}$. H_3 then extracts P_1^*'s input to the j-th instance by comparing its private output $\{V_i^{(j)}\}_{i\in[n_1]}$ from the CCCT with label pairs $\{W_{i,0}^{(j)}, W_{i,1}^{(j)}\}_{i\in[n_1]}$ computed from randomness $\rho^{(j)}$. In particular, for each $i \in [n_1]$,

$$\begin{cases} x_i^{(j)} \leftarrow b & \text{if } V_i^{(j)} = W_{i,b}^{(j)} \text{ for } b \in \{0,1\} \\ x_i^{(j)} \overset{R}{\leftarrow} \{0,1\} & \text{otherwise.} \end{cases}$$

Let x' be the most common value from multiset $\{x^{(j)}\}_{j\in E}$.

At last, H_3 verifies the CCOT's soundness as follows:

- H_3 computes N_2, the set of the checked instances that are good, that is,

$$N_2 = \left\{ j \in [\sigma] \backslash E \;\middle|\; \begin{array}{r} \forall i \in [n_1], b \in \{0,1\}, \\ \Pi_i^{(j)} = \mathsf{com}(\pi_i^{(j)}) \wedge \\ \Theta_{i,b}^{(j)} = \mathsf{com}(W_{i,b\oplus\pi_i^{(j)}}^{(j)}) \end{array} \right\},$$

  where $W_{i,b}^{(j)} = (K_{i,b}^{(j)}, b \oplus \pi_i^{(j)})$.

- H_3 computes F_2, the set of the unchecked instances that are good and have honest GEN's input labels, that is,

$$F_2 = \left\{ j \in E \;\middle|\; \begin{array}{r} \forall i \in [n_1], b \in \{0,1\}, \\ \Pi_i^{(j)} = \mathsf{com}(\pi_i^{(j)}) \wedge \\ \Theta_{i,b}^{(j)} = \mathsf{com}(W_{i,b\oplus\pi_i^{(j)}}^{(j)}) \wedge V_i^{(j)} = W_{i,x_i'}^{(j)} \end{array} \right\},$$

  where $W_{i,b}^{(j)} = (K_{i,b}^{(j)}, b \oplus \pi_i^{(j)})$ is computed from randomness $\rho^{(j)}$ and $V_i^{(j)}$ is H_3's private output from the CCCT.

Recall that the CCCT's soundness for the checked instances ensures that $N_2 = [\sigma]\backslash E$ and the soundness for the unchecked instances ensures that $|F_2| > |E|/2$. Therefore, H_3 verifies these two properties and sends $\perp$ to O if either $N_2 \neq [\sigma]\backslash E$ or $|F_2| \leq |E|/2$.

Lemma 3.7. $\left\{\mathbf{Hybrid}_2(x, y, 1^k)\right\}_{x,y,k} \overset{c}{\approx} \left\{\mathbf{Hybrid}_3(x, y, 1^k)\right\}_{x,y,k}$

Proof. At a high level, this lemma holds due to the CCCT's soundness property. In particular, if H_2 rejects the CCCT, H_3 rejects too. These two experiments differ when H_2 accepts but H_3 rejects, that is, when P_1^* corrupts the labels assigned to GEN's input wires in more than $|E|/2$ instances while being able to prove the opposite in the CCCT. The soundness property ensures that this only happens with negligible probability. □

Hybrid$_4(x, y, 1^k)$**:** H_4 is the same as H_3 except that H_4 sends $\perp$ to O if all the checked circuits are honest but more than $|E|/2$ of the evaluation circuits received in Step 8 are faulty. Since for all $j \in [\sigma]$, H_4 learns $\rho^{(j)}$ even though $\rho^{(j)}$ is sent over the check channel. H_4 can then easily use $\rho^{(j)}$ to regenerate $\{\Theta^{(j)}, \Omega^{(j)}, \Pi^{(j)}\}$ and $G(C)^{(j)}$. If the regenerated one coincides with the one received from P_1^*, the j-th garbled circuit is honest. Let N_3 denote the checked circuits that are honest and F_3 denote the evaluation circuits that are honest. H_4 sends $\perp$ to O if $N_3 \neq [\sigma]\backslash E$ or $|F_3| \leq |E|/2$.

Lemma 3.8. $\left\{\mathbf{Hybrid}_3(x, y, 1^k)\right\}_{x,y,k} \overset{s}{\approx} \left\{\mathbf{Hybrid}_4(x, y, 1^k)\right\}_{x,y,k}$

Proof. Note that both H_3 and H_4 get to compute N_3. So if $N_3 \neq [\sigma]\backslash E$, that is, any of the check circuits is faulty, both H_3 and H_4 abort, in **Hybrid**$_3$ and **Hybrid**$_4$, respectively. The only case that **Hybrid**$_3$ and **Hybrid**$_4$ differ is when H_3 accepts H_4 rejects due to $|F_3| \leq |E|/2$, that is, more than half of the evaluation circuits are faulty. In other words, P_1^* generates $|E|/2 = \sigma/5$ faulty circuits, and H_3 randomly picks $3\sigma/5$ to check and all of the chosen ones are among those

$4\sigma/5$ honest circuits. The probability of this case is at most

$$\frac{\binom{4\sigma/5}{3\sigma/5}}{\binom{\sigma}{3\sigma/5}} = \frac{(2\sigma/5)!(4\sigma/5)!}{\sigma!(\sigma/5)!} \le \frac{\sqrt{2\pi\frac{2\sigma}{5}}\cdot\left(\frac{2\sigma/5}{e}\right)^{2\sigma/5}\sqrt{2\pi\frac{4\sigma}{5}}\cdot\left(\frac{4\sigma/5}{e}\right)^{4\sigma/5}}{\sqrt{2\pi\sigma}\cdot\left(\frac{\sigma}{e}\right)^{\sigma}\sqrt{2\pi\frac{\sigma}{5}}\cdot\left(\frac{\sigma/5}{e}\right)^{\sigma/5}} \le 2^{-.32\sigma}. \qquad (3.1)$$

Thus, the probability that **Hybrid**$_3$ and **Hybrid**$_4$ differ is negligible in σ, hence negligible in k. It is noteworthy that the first inequality comes from Stirling's approximation for factorials. □

Hybrid$_5(x, y, 1^k)$**:** H_5 is the same as H_4 except that after confirming that all the check circuits are honest, it sends $\perp$ to O if $|F| \le \sigma/5$, where $F = F_1 \cap F_2 \cap F_3$.

Lemma 3.9. $\left\{\textbf{Hybrid}_4(x, y, 1^k)\right\}_{x,y,k} \overset{s}{\approx} \left\{\textbf{Hybrid}_5(x, y, 1^k)\right\}_{x,y,k}$.

Proof. Our argument here will be conditioned on that neither H_4 nor H_5 abort before confirming that all the check circuits are honest, since both hybrids behave identically before that step.

Let M denote set $E \backslash F$. We know that $F_i \subset E$ and $|F_i| > |E|/2$ for $i \in \{1, 2, 3\}$ from previous hybrids. We also know that H_4 and H_5 differ when $|F| \le |E|/2$, or equivalently, $|M| > |E|/2$. We will argue that this happens with probability negligible in σ.

Observe that for all $j \in M$, P_1^* must have cheated in the j-th instance in at least one of the following three ways: (1) provide malicious labels for GEN's input; (2) provide malicious labels for EVAL's input; or (3) provide a faulty circuit. Recall that due to the circuit OT's receiver security, P_1^* is completely oblivious of E the whole time. In other words, the choosing of M is independent of the choosing of E. Recall that $E \subset [\sigma]$ is chosen uniformly such that $|E| = 2\sigma/5$. The probability that H_4 and H_5 differ is equal to the probability that all those in M are evaluation instances, in particular, the probability that $|M| > |E|/2$ and $M \subset E$, which is

$$\Pr((|M| > \sigma/5) \wedge (|E| = 2\sigma/5) \wedge (M \subset E)) \le \frac{\binom{\sigma/5}{\sigma/5}\cdot\binom{4\sigma/5}{\sigma/5}}{\binom{\sigma}{2\sigma/5}} = \frac{(2\sigma/5)!(4\sigma/5)!}{\sigma!(\sigma/5)!} < 2^{-.32\sigma},$$

which is negligible in σ. Note that the second inequality comes from Equation (3.1). □

Hybrid$_6(x, y, 1^k)$**:** H_6 is the same as H_5 except that H_6 sends x' to O and outputs whatever P_1^* outputs. Furthermore, while **Hybrid**$_5$ denotes the joint distribution of P_1^*'s and H_5's outputs, **Hybrid**$_6$ denotes the joint distribution of H_6's and S_2's outputs.

Lemma 3.10. $\left\{\textbf{Hybrid}_5(x, y, 1^k)\right\}_{x,y,k} = \left\{\textbf{Hybrid}_6(x, y, 1^k)\right\}_{x,y,k}$

Proof. Since H_6 outputs whatever P_1^* outputs, P_1^*'s output in **Hybrid**$_5$ is identical to H_6's in **Hybrid**$_6$. Moreover, $|F| > |E|/2$ and $F = F_1 \cap F_2 \cap F_3$ imply that the majority of those in E are honest executions of the Yao protocol that compute $f(x', y)$. Since H_5's output comes from the majority output of those from E, H_5's output in **Hybrid**$_5$ coincides with S_2's output in **Hybrid**$_6$. This lemma therefore follows. □

Hybrid$_7(x, y, 1^k)$**:** H_7 is the same as H_6 except that H_7 uses 0^{n_2} as input to the CCOT.

Lemma 3.11. $\left\{\textbf{Hybrid}_6(x, y, 1^k)\right\}_{x,y,k} \overset{c}{\approx} \left\{\textbf{Hybrid}_7(x, y, 1^k)\right\}_{x,y,k}$

Proof. (Due to the CCOT's indistinguishability property) □

Finally, since every step finishes in expected polynomial time, including σ invocations of simulator S_1^{OT}, H_7 runs in expected polynomial time. Observe that H_7 and S_1 are syntactically similar, hence **Hybrid**$_7$ = **Ideal**. In particular, P_2's private input y is no longer used (replaced by 0^{n_2} to be exact). By Lemma 3.5–3.11, we prove the simulation security of the Main protocol in the presence of arbitrary malicious P_1^*. □

3.5.2 For Malicious Evaluator

Protocol 3.4. The Simulator for an Arbitrary Malicious Evaluator P_2^*

Common Input: security parameter 1^k, statistical security parameter 1^σ, and boolean circuit C that computes $f(x, y)$.

Private Input: P_2^* has private input y but uses y' (which will be extracted) in the protocol instead.

Private Output: P_2^* receives output $f_2(x, y')$.

1. **(Circuit OTs)** For each $j \in [\sigma]$, S_2 randomly picks $L_0^{(j)}, L_1^{(j)} \xleftarrow{R} \{0,1\}^k$, and in the j-th instance of the $\binom{2}{1}$-OTs, S_1 invokes externally the simulator that comes with the $\binom{2}{1}$-OT's sender security and extracts P_2^*'s input e_j. Upon request, S_1 provides $(L_0^{(j)}, L_1^{(j)})$ as input to the external simulator. Moreover, S_2 uses $(L_0^{(j)}, L_1^{(j)})$ to construct the cut-and-choose channels for the j-th circuit.
2. **(Pick the randomness)** For all $j \in [\sigma]$, S_2 picks randomness $\rho^{(j)}$ for the j-th garbled circuit and sends $\rho^{(j)}$ over the check channel.
3. **(Pick random label pairs)** For all $j \in [\sigma]$ and every wire w_i, S_2 uses randomness $\rho^{(j)}$ to compute $(K_{i,0}^{(j)}, K_{i,1}^{(j)}, \pi_i^{(j)}) \in \{0,1\}^k \times \{0,1\}^k \times \{0,1\}$. Let $W_{i,b}^{(j)}$ denote the key-locator pair $(K_{i,b}^{(j)}, b \oplus \pi_i^{(j)})$. $W_{i,b}^{(j)}$ is called the *label* corresponding to wire w_i of value b in the j-th garbled circuit hereafter.
4. **(Commit to input label pairs)** Let $\{w_i\}_{i \in [n_1]}$ be the wires corresponding to GEN's input and $\{w_{n_1+i}\}_{i \in [n_2]}$ be those corresponding to EVAL's input. S_2 uses randomness $\rho^{(j)}$ to commit to the label pairs assigned to $\{w_i\}_{i \in [n_1+n_2]}$ and random permutation

bits assigned to $\{w_i\}_{i\in[n_1]}$ by sending $\{\Theta^{(j)}, \Omega^{(j)}, \Pi^{(j)}\}_{j\in[\sigma]}$ to Eval, where

$$\Theta^{(j)} = \{\Theta^{(j)}_{i,0}, \Theta^{(j)}_{i,1}\}_{i\in[n_1]} = \{\mathsf{com}(W^{(j)}_{i,0\oplus\pi^{(j)}_i}), \mathsf{com}(W^{(j)}_{i,1\oplus\pi^{(j)}_i})\}_{i\in[n_1]};$$
$$\Omega^{(j)} = \{\Omega^{(j)}_{n_1+i,0}, \Omega^{(j)}_{n_1+i,1}\}_{i\in[n_2]} = \{\mathsf{com}(W^{(j)}_{n_1+i,0}), \mathsf{com}(W^{(j)}_{n_1+i,1})\}_{i\in[n_2]};$$
$$\Pi^{(j)} = \{\Pi^{(j)}_i\}_{i\in[n_1]} = \{\mathsf{com}(\pi^{(j)}_i)\}_{i\in[n_1]}.$$

5. **(Retrieve Eval's Input Labels)** Instead of jointly running CCOT with P_2^*, simulator S_2 invokes externally the simulator that comes with CCOT's sender security to extract P_2^*'s input y'. Upon request, S_2 sends $\{W^{(j)}_{n_1+i,y'_i}\}_{j\in[\sigma],i\in[n_2]}$ over the check channels to the external simulator. If P_2^* aborts, S_2 sends $\perp$ to the external trusted party O.

6. **(Retrieve Gen's Input Labels)** S_2 uses 0^{n_1} as its private input in the CCCT. With common input security parameter 1^k, statistical security parameter 1^σ, and commitments $\{\Theta^{(j)}_{i,0}, \Theta^{(j)}_{i,1}, \Pi^{(j)}_i\}_{j\in[\sigma],i\in[n_1]}$, both parties jointly run CCCT over the cut-and-choose channels, that is,

$$\text{CCCT}(E) : ((\{W^{(j)}_{i,0}, W^{(j)}_{i,1}\}_{j\in[\sigma],i\in[n_1]}, 0^{n_1}), \perp) \mapsto (\perp, (\{W^{(j)}_i\}_{j\in E,i\in[n_1]}, \mathsf{out})).$$

If P_2^* aborts, S_2 sends $\perp$ to O.

7. **(Retrieve Garbled Circuits)** S_2 sends y' to O and receives $f(x, y')$ in return. Let $\{w_i\}_{i\in O}$ be the circuit-output wires. S_2 parses $f(x, y')$ into $\{b_i\}_{i\in O}$, where $b_i \in \{0, 1\}$. Next, S_2 generates garbled circuit $G(C)^{(j)}$ such that if $e_j = 1$, then $G(C)^{(j)}$ always outputs $f(x, y')$; or if $e_j = 0$, then $G(C)^{(j)}$ is an honest garbled version of objective circuit C. In particular, for each gate $g : \{0, 1\} \times \{0, 1\} \mapsto \{0, 1\}$ with input wires w_a and w_b and output wire w_c, S_2 garbles gate g by computing

$$G(g)^{(j)} \leftarrow (\langle\pi^{(j)}_a, \pi^{(j)}_b\rangle, \langle\pi^{(j)}_a, 1-\pi^{(j)}_b\rangle, \langle 1-\pi^{(j)}_a, \pi^{(j)}_b\rangle, \langle 1-\pi^{(j)}_a, 1-\pi^{(j)}_b\rangle),$$

where

$$\langle d_1, d_2 \rangle = \begin{cases} \mathsf{enc}_{K^{(j)}_{a,d_1}}(\mathsf{enc}_{K^{(j)}_{b,d_2}}(W^{(j)}_{c,b_c})) & \text{if } e_j = 1 \text{ and } c \in O \\ \mathsf{enc}_{K^{(j)}_{a,d_1}}(\mathsf{enc}_{K^{(j)}_{b,d_2}}(W^{(j)}_{c,g(d_1,d_2)})) & \text{otherwise.} \end{cases}$$

S_2 then sends $\{G(C)^{(j)}\}_{j\in[\sigma]}$ to P_2^*, where $G(C)^{(j)} = \left(\{G(g)^{(j)}\}_{g\in C}, \{\pi_i^{(j)}\}_{i\in O}\right)$.

Remark 3.14. *Recall that $e_j = 1$ means that the j-th circuit is an evaluation circuit. If $e_j = 1$ and w_c is a circuit-output wire, then $G(g)$ always outputs $W^{(j)}_{c,b_c}$, that is, gate g always outputs b_c. This implies that if $e_j = 1$, the j-th garbled circuit always outputs $\{b_i\}_{i\in O}$, which is equivalent to $f(x, y')$.*

8. **(Check Circuits)** S_2 outputs $\perp$ if P_2^* aborts.

9. **(Evaluate Circuits)** S_2 does nothing in this step.

10. **(Majority Operation)** S_2 outputs whatever P_2^* outputs. 1

Lemma 3.12 (Security against a Malicious Evaluator). *Let $\Pi = (\text{Gen}, \text{Eval})$ denote Protocol 3.2.* 2
Assume that the $\binom{2}{1}$-OT protocol is secure in the presence of malicious adversaries, and there exist a 3
perfectly-hiding commitment scheme, a semantically secure symmetric encryption scheme, a secure 4
CCOT, and a secure CCCT. 5

For any single-output function $f : \{0,1\}^{n_1} \times \{0,1\}^{n_2} \mapsto \{\perp\} \times \{0,1\}^*$, and any non-uniform probabilistic strict polynomial-time machine P_2^*, simulator S_2 presented in Protocol 3.4 is a non-uniform probabilistic expected polynomial-time machine satisfying that

$$\left\{\mathbf{Real}_{\Pi,\bar{P}}(x, y, 1^k)\right\}_{x,y,k} \approx_c \left\{\mathbf{Ideal}_{f,\bar{S}}(x, y, 1^k)\right\}_{x,y,k},$$

where $\bar{P} = (P_1, P_2^*)$ and $\bar{S} = (S_1, S_2)$. Note that S_1 is honest as described in Definition 2.5. 6

Remark 3.15. *For brevity, subscript "x, y, k" denotes that both distributions range over all $x \in \{0,1\}^{n_1}$, all $y \in \{0,1\}^{n_2}$, and all $k \in \mathbb{N}$.*

Proof. Recall that in this case, P_1 follows the Main protocol faithfully. To prove this lemma, consider the following hybrid experiments and lemmas. Note that for the i-th hybrid experiment, we denote by H_i the algorithm that plays the role of GEN and by $\mathbf{Hybrid}_i(x, y, 1^k)$ the joint output.

$\mathbf{Hybrid}_1(x, y, 1^k)$: H_1 is the same as P_1 except that in all the σ instances of the circuit OT, instead of running as a real-model OT sender, H_1 invokes ideal-model OT simulator S_2^{OT}, the existence of which is guaranteed by OTs' sender security.

Lemma 3.13. $\left\{\mathbf{Real}_{\Pi,\bar{P}}(x, y, 1^k)\right\}_{x,y,k} \approx_c \left\{\mathbf{Hybrid}_1(x, y, 1^k)\right\}_{x,y,k}$

Proof. The only difference between $\mathbf{Hybrid}_1$ and **Real** is that S_2^{OT} is invoked in $\mathbf{Hybrid}_1$. After this step, both experiments are identical since this experiment acts exactly the same as P_1 does in **Real**. So this lemma follows from the simulation property of the secure OT protocol. A little bit more formally, let $\mathbf{Hybrid}_{0,i}$ be the same as **Real** except that the first i executions of the OT protocol are replaced with invocations to S_2^{OT}. Observe that $\mathbf{Hybrid}_{0,0} = \mathbf{Real}$ and $\mathbf{Hybrid}_{0,n_1+\sigma} = \mathbf{Hybrid}_1$. If there exists some probabilistic polynomial-time distinguisher D such that for infinitely many ks, D distinguishes **Real** from **Hybrid** with probability $\Pr(k)$. Then by averaging, there must exist some i such that D distinguishes $\mathbf{Hybrid}_{0,i}$ from $\mathbf{Hybrid}_{0,i+1}$ with probability $\Pr(k)/(n_1 + \sigma)$. We can then construct an algorithm A with black-box access to D to break the (sender) security of the OT protocol, which results in a contradiction. □

$\mathbf{Hybrid}_2(x, y, 1^k)$: This experiment is the same as $\mathbf{Hybrid}_1$ except that this experiment invokes the external oracle with input y' and gets $f(x, y')$ in return. This experiment aborts if this result does not coincide with the result from the circuit evaluation.

Lemma 3.14. $\left\{\textbf{Hybrid}_1(x,y,1^k)\right\}_{x,y,k} \overset{s}{\approx} \left\{\textbf{Hybrid}_2(x,y,1^k)\right\}_{x,y,k}$

Proof. Recall that P_1 is assumed to be honest. So the circuits are correctly garbled and P_1's input labels are honestly provided. Since P_2^* got the labels corresponding to its input $\bar{y}'$ via the simulator S_2^{OT}, both input labels and circuit are correct. So $\textbf{Hybrid}_1$ and $\textbf{Hybrid}_2$ are identical, and this lemma holds. □

$\textbf{Hybrid}_3(x,y,1^k)$**:** Recall that the objective circuit's output includes three parts: a 2-universal hash of Gen's input, encryption of Gen's output, and Eval's output. Let $O = O_1 \cup O_2 \cup O_3$ denote all the circuit-output wires, where O_i denotes the wires corresponding to the i-th part of Eval's output. This experiment is the same as $\textbf{Hybrid}_2$ except that the evaluation circuits are constructed to output a random number as the first part of its output. In particular, this experiment

1. picks a random number $\varepsilon \in \{0,1\}^k$,
2. computes garbled truth table $\{G(C)^{(j)}\}_{j\in[\sigma]}$ honestly just like an honest P_1 does, and
3. computes garbled circuit

$$G(C)^{(j)} = \begin{cases} \left(\{G(g)^{(j)}\}_{g\in C}, \{\pi_i^{(j)}\}_{i\in O}\right) & \text{if } s'_j = 0, \\ \left(\{G(g)^{(j)}\}_{g\in C}, \{\pi_i^{(j)} \oplus \varepsilon_i\}_{i\in O_1} \cup \{\pi_i^{(j)}\}_{i\in O_2\cup O_3}\right) & \text{if } s'_j = 1. \end{cases}$$

We slightly abuse the notation that ε_i denotes the corresponding bit to circuit-output wire $w_i \in O_1$.

Lemma 3.15. $\left\{\textbf{Hybrid}_2(x,y,1^k)\right\}_{x,y,k} \overset{s}{\approx} \left\{\textbf{Hybrid}_3(x,y,1^k)\right\}_{x,y,k}$

Proof. Note that the differences of the two hybrid experiments are the permutation bits of wires in O_1 and the evaluation output. In particular, P_2^* learns

$$(\{\pi_i^{(j)}\}_{i\in O_1}, \quad H\cdot\bar{x}, \quad f_1(x,y')\oplus e, \quad f_2(x,y')) \quad \text{in } \mathbf{Hybrid}_2\text{, but learns}$$

$$(\{\pi_i^{(j)}\oplus\varepsilon_i\}_{i\in O_1}, \quad H\cdot\bar{x}\oplus\varepsilon, \quad f_1(x,y')\oplus e, \quad f_2(x,y')) \quad \text{in } \mathbf{Hybrid}_3,$$

where $\bar{x} = x||e||r$. Note that since H comes from a coin tossing protocol, it is uniformly distributed as long as one of the parties is honest. Since $|r| = 2k + \lg(k)$, Lemma 4.10 shows that $H \cdot x$ is also uniformly distributed.

Lemma 3.16 (Claim 8 of [LP07]). *Given a boolean circuit C computing function f and an output value z, it is possible to efficiently construct a garbled circuit $G(C)'$ such that*

1. *For all inputs, the output of $G(C)'$ is always z,*
2. *If $z = f(x,y)$ for some x and y, then no non-uniform probabilistic polynomial-time adversary can distinguish between the distribution ensemble consisting of $\{G(C)', K_{x'}, K_{y'}\}$ where $K_{x'}, K_{y'}$ are the garbled keys for arbitrary x', y', and the ensemble consisting of $\{G(C), K_x, K_y\}$ where $G(C)$ is a real garbled version of C, and K_x, K_y are the garbled keys representing inputs x, y.*

Formally, let $\{G(C)^{(j)}\}_{j\in[\sigma]}$ be the garbled circuits in $\mathbf{Hybrid}_1$ and $\{G(C)'^{(j)}\}_{j\in[\sigma]}$ be the fake garbled circuits in $\mathbf{Hybrid}_2$.[1] Let $\mathbf{Hybrid}_{1,i}$ be the same as $\mathbf{Hybrid}_1$ except that the first i circuits $\{G(C)^{(j)}\}_{j\in[i]}$ are replaced with $\{G(C)'^{(j)}\}_{j\in[i]}$. Observe that $\mathbf{Hybrid}_{1,0} = \mathbf{Hybrid}_1$ and $\mathbf{Hybrid}_{1,\sigma} = \mathbf{Hybrid}_2$. Suppose that there exists some $\bar{x}, \bar{y}$ and some probabilistic polynomial-time distinguisher D such that for infinitely many k, D distinguishes $\mathbf{Hybrid}_1$ from $\mathbf{Hybrid}_2$ with probability $\Pr(k)$. Then by averaging, there must be some i such that D distinguishes $\mathbf{Hybrid}_{1,i}$ from $\mathbf{Hybrid}_{1,i+1}$ with probability $\Pr(k)/\sigma$. With such distinguisher D, an algorithm

[1]Technically, only 40% among $\{G(C)'^{(j)}\}_{j\in[\sigma]}$ are fake. In each fake circuit, only P_2's output gates are modified to output the constant $f_2(\bar{x}, \bar{y}')$.

A distinguishing $\{G(C)', K_x, K_{y'}\}$ from $\{G(C), K_x, K_{y'}\}$ can be constructed as follows: On input either $\{G(C)', K_x, K_{y'}\}$ or $\{G(C), K_x, K_{y'}\}$, A first simulates $\mathbf{Hybrid}_{1,i}$ except that A uses its input for the $(i+1)$-th garbled circuit, and completes the rest of experiment $\mathbf{Hybrid}_{1,i}$. Then A runs D on the resulting transcript and echo D's output. A does a perfect simulation of either $\mathbf{Hybrid}_{1,i}$ or $\mathbf{Hybrid}_{1,i+1}$, and therefore distinguishes garbled circuits with probability $\Pr(k)/\sigma$. By Lemma 3.16, $\Pr(k)$ must be negligible and we complete the proof. Note that both $\mathbf{Hybrid}_1$ and $\mathbf{Hybrid}_2$ still use the real input keys $\{K_x\}$. □

$\mathbf{Hybrid}_3(x, y, 1^k)$**:** This is the same as $\mathbf{Hybrid}_2$ except that input 0^{n_1} is used in stead of input $\bar{x}$.

Lemma 3.17. $\left\{\mathbf{Hybrid}_2(x, y, 1^k)\right\}_{x,y,k} \approx_s \left\{\mathbf{Hybrid}_3(x, y, 1^k)\right\}_{x,y,k}$

Proof. Intuitively, this follows the hiding and binding property of the commitments and the use of 2-universal hash function to ensure GEN's input consistency. Formally, ... □

$\mathbf{Hybrid}_4(x, y, 1^k)$**:** This is the same as experiment $\mathbf{Hybrid}_3$ with the exception that $\mathbf{Hybrid}_4$ sends abort to the external oracle if the last proof protocol does not verify. Afterwards, S_2 outputs whatever P_2^* outputs.

Lemma 3.18. $\left\{\mathbf{Hybrid}_3(x, y, 1^k)\right\}_{x,y,k} \approx_c \left\{\mathbf{Hybrid}_4(x, y, 1^k)\right\}_{x,y,k}$

Lemma 3.19. *The simulator $S_2^{P_2^*}$ runs in expected polynomial time.*

Proof. Since P_2^* is a strict polynomial-time adversary, most of the steps of the simulation are also strictly polynomial-time except the running of OT simulator S_2^{OT}. Thus, the running time of simulator S_2 can be expressed as $p(k) + r_{\mathrm{OT}}$ where r_{OT} is a random variable denoting the running time of S_2^{OT}. This lemma follows because S_2^{OT} occurs only once, that is, it is never rewound or part of a loop. Thus, the overall running time is expected polynomial time by linearity of expectations. □

Finally, note that S_2 and H_4 are syntactically similar. By Lemma 3.13–3.19, we prove the security of the Main protocol in the presence of an arbitrary malicious P_2^*. $\square$

Chapter 4

Secure 2PC Instantiations

4.1 Secure 2PC via Number-Theoretic Assumptions

CCCT via Malleable Claw-Free Collections In Section 4.1.1, we present an instantiation of CCCT, in which we make use of the *claw-free collections* that have a weak malleability property.

Loosely speaking, a pair of functions (f_0, f_1) are said to be *claw-free* if they are (1) easy to evaluate, (2) identically distributed over the same range, and (3) hard to find a *claw*. A claw is a pair of elements, one from f_0's domain and the other from f_1's domain, that are mapped to the same range element.

Intuitively, GEN's input is encoded using elements from the domain of the claw-free collections. Those elements can later be used to prove their consistency among circuits. In particular, GEN uses the elements to construct random keys corresponding to its input wires. The rest of the gates in the circuit use fast symmetric operations as per the Yao protocol. A concrete example is to instantiate the claw-free functions under the Discrete Logarithm assumption by letting $f_b(r) = g^b h^r$ for some primes p and q such that $p = 2q + 1$, and some distinct group elements g and h of $\mathbb{Z}_p^*$ such that $\langle g \rangle = \langle h \rangle = q$. A toy example of CCCT built on this pair of claw-free functions works as follows:

1. Gen samples $\{r_0^{(j)}\}_{j\in[\sigma]}$ and $\{r_1^{(j)}\}_{j\in[\sigma]}$ from f_0 and f_1's domain $\mathbb{Z}_q$. Range elements $\{h^{r_0^{(j)}}\}_{j\in[\sigma]}$ and $\{gh^{r_1^{(j)}}\}_{j\in[\sigma]}$ are then used to construct garbled circuits such that $g^b h^{r_b^{(j)}}$ is associated with Gen's input wire of value b in the j-th garbled circuit.

2. The cut-and-choose method enforces that the majority of the evaluation circuits are correctly constructed. Let E be the set of the evaluation circuits.

3. At the onset of the evaluation phase, Gen with input bit x sends $\vec{K} = \{g^x h^{r_x^{(j)}}\}_{j\in E}$ to Eval and then proves that these range elements are the encodings of the same x. Intuitively, by the identical range distribution property, Eval with $f_x(r_x^{(j)})$ at hand has no information about x. Furthermore, after Gen proves the knowledge of the pre-image of $\vec{K}$ under the same $f_x(\cdot)$, the claw-free property ensures that Gen's input to most of the evaluation circuits are consistent.

It is well-known that such a pair of claw-free functions have efficient zero-knowledge proofs. However, in the course of developing our proof, we noticed that witness indistinguishable proofs suffice in place of zero-knowledge proofs. Even more generally, when the claw-free collection has a very weak malleability property (which holds for all known concrete instantiations), sending a simple function of the witness itself suffices.

CCOT via 2-Out-of-1 Committing Oblivious Transfers In Section 4.1.2, we present a secure CCOT instantiated from $\binom{2}{1}$-COT proposed by Kiraz and Schoenmakers [KS06]. At a high level, this approach instructs the sender (Gen in the Yao protocol) of the $\binom{2}{1}$-OT to *post-facto* prove that it ran the $\binom{2}{1}$-OT correctly by revealing the randomness used during the protocol execution. Normally, this would break the $\binom{2}{1}$-OT's sender security. However, in the cut-and-choose framework, Gen is already opening many circuits, so the keys used as inputs for the $\binom{2}{1}$-OT are no longer secret. Therefore, the idea is that Gen can prove that it executed the $\binom{2}{1}$-OT correctly for all check circuits by simply sending the random coins used in those instances. We stress that not every $\binom{2}{1}$-OT can be used here. Loosely speaking, a $\binom{2}{1}$-COT is the $\binom{2}{1}$-OT with the binding property so that it is hard for a cheating Gen to produce random coins different from what it really used. An efficient $\binom{2}{1}$-COT based on Decisional Diffie-Hellman assumption is also presented.

4.1.1 CCCT via Malleable Claw-Free Collections

The main idea of our solution is inspired both by Mahassel and Franklin's Committed-Input scheme [MF06] and by the *claw-free collections*. We first give the formal definition of the claw-free collections as follows:

Definition 4.1 (Claw-Free Collections in [GK96]). *A three-tuple of algorithms (G, D, F) is called a* claw-free collection *if the following conditions hold*

1. ***Easy to evaluate:*** *Both index selecting algorithm G and the domain sampling algorithm D are probabilistic polynomial-time, while evaluating algorithm F is deterministic polynomial-time.*

2. ***Identical range distribution:*** *Let $f_b(r)$ denote the output of F on input (b, I, r). For any I in the range of G, the following two distributions are identical:*

   - $\{r \leftarrow D(0, I) : f_0(r)\}$
   - $\{r \leftarrow D(1, I) : f_1(r)\}$

3. ***Hard to form claws:*** *For every non-uniform probabilistic polynomial-time algorithm A, the probability that for a randomly generated index I, A could come up with r_0 and r_1 such that $f_0(r_0) = f_1(r_1)$ is negligible.*

Let E denote the set of the evaluation circuits and assume that GEN's input x is only one-bit. With the claw-free collections, our idea works as follows:

1. EVAL first generates I by invoking the index generating algorithm $G(1^k)$, where k is the security parameter.
2. GEN computes $r \leftarrow D(x, I)$ and $K \leftarrow f_x(r)$. Then GEN sends K to EVAL.
3. GEN invokes sampling algorithms $D(I, b)$ to pick $\{r_b^{(j)}\}_{j \in [\sigma]}$ for $b \in \{0, 1\}$.
4. For $j \in [\sigma]$ and $b \in \{0, 1\}$, GEN computes $K_b^{(j)} \leftarrow f_b(r_b^{(j)})$ and commits to $K_b^{(j)}$.
5. Both parties jointly do the following:
   - For $j \in [\sigma] \backslash E$, GEN decommits to $(K_0^{(j)}, K_1^{(j)})$ and reveals $(r_0^{(j)}, r_1^{(j)})$ for EVAL to check the correctness of $(K_0^{(j)}, K_1^{(j)})$.
   - For $j \in E$, GEN decommits to $K_x^{(j)}$ and proves that it computes K and $K_x^{(j)}$ via the same function $f_x(\cdot)$ without revealing x.

At a high level, this idea works for the following two reasons.

- The identical range distribution property implicitly implies that both $f_0(\cdot)$ and $f_1(\cdot)$ have the perfectly-hiding property. For any $r_0 \leftarrow D(0, I)$, there always exists r_1 in $f_1(\cdot)$'s domain such that $f_0(r_0) = f_1(r_1)$. In other words, K and $\{K_x^{(j)}\}_{j \in E}$ will not leak any information about x to EVAL.
- If a malicious GEN is able to introduce inconsistent inputs, that is, there exists $j \in E$ such that GEN can prove that both K and $K^{(j)}$ are computation results from $f_x(\cdot)$, while $K^{(j)}$ is actually computed via $f_{1-x}(\cdot)$. This implies that GEN knows r_0 and r_1 such that $f_x(r_0) = K^{(j)} = f_{1-x}(r_1)$, which contradicts the "hard to form claws" property of the claw-free functions.

It remains to show how GEN proves to EVAL that it computes K and $K_x^{(j)}$ via the same $f_x(\cdot)$ efficiently. Let us consider the claw-free collection instantiated from the Discrete Logarithm assumption again, that is, let $f_b(r) = g^b h^r$, where $I = (g, h, p, q)$ includes two primes p and q such that $p = 2q + 1$, and distinct elements g and h of $\mathbb{Z}_p^*$ such that $\langle g \rangle = \langle h \rangle = q$. Again, assume GEN only has one-bit input x. After revealing K and $\{K^{(j)}\}_{j \in E}$ to EVAL, where $K = g^x h^r$ and $K^{(j)} = g^x h^{r_x^{(j)}}$, a natural solution is for GEN to prove in zero-knowledge the knowledge of $d^{(j)} = r^{(j)} - r$ such that $K^{(j)} = K \cdot h^{d^{(j)}}$ for all $j \in E$. Nonetheless, we observe that it is unnecessary for GEN to hide $(r_x^{(j)} - r)$ from EVAL. Indeed, since r and $\{r_x^{(j)}\}_{j \in E}$ are new random variables introduced by GEN, giving $(r_x^{(j)} - r)$ in clear to EVAL will not compromise GEN's input privacy.

To generalize this idea, we introduce the following notion.

Definition 4.2 (Malleable Claw-Free Collections). *A four-tuple of algorithms (G, D, F, R) is a* malleable claw-free collection *if the following conditions hold.*

1. ***A subset of claw-free collections:** (G, D, F) is a claw-free collection, and the range of D and F are cyclic groups, denoted by $(\mathbb{G}_1, +)$ and $(\mathbb{G}_2, \cdot)$ respectively.*

2. ***Uniform domain sampling:** For any I in the range of G, random variable $D(0, I)$ and $D(1, I)$ are uniform over $\mathbb{G}_1$, and denoted by $D(I)$ for simplicity.*

3. ***Malleability:** $R : \mathbb{G}_1 \to \mathbb{G}_2$ runs in polynomial time, and for $b \in \{0, 1\}$, any I in the range of G, and any $r_1, r_2 \in \mathbb{G}_1$, $f_b(r_1 + r_2) = f_b(r_1) \cdot R_I(r_2)$.*

Remark 4.1. *It is noteworthy that group operations are only needed for the input gates, those of* GEN *in particular. All of the remaining gates in the circuit are generated as per the Yao protocol. So, unlike solutions with committed OT such as [JS07], number-theoretic operations are only used for the input gates rather than the entire circuit.*

A CCCT based on the malleable claw-free collections is shown in Protocol 4.1.

Protocol 4.1. Secure CCCT via Malleable Claw-Free Collections

Common Input: security parameter 1^k, statistical security parameter 1^σ, perfectly-hiding commitment scheme com, and malleable claw-free collection (G, D, F, R). Also, both parties have built the cut-and-choose communication channels. Let E denote the set of the evaluation instances.

Before CCCT starts: Both parties jointly set up the CCCT environment as follows:

1. EVAL computes $I \leftarrow G(1^k)$ and sends I to GEN.
2. For all $j \in [\sigma]$, GEN picks $\pi^{(j)} \overset{R}{\leftarrow} \{0,1\}$. Then GEN computes $r_b^{(j)} \leftarrow D(I)$ and $K_b^{(j)} \leftarrow f_b(r_b^{(j)})$ for all $j \in [\sigma]$ and $b \in \{0,1\}$. Recall that we use $f_b(r)$ to denote $F(b, I, r)$ for simplicity.

   Remark 4.2. *Recall that in the Yao protocol, for each w_i of the wires corresponding to* GEN*'s input,* GEN *picks $(K_0^{(j)}, K_1^{(j)}, \pi^{(j)}) \overset{R}{\leftarrow} \{0,1\}^{2k+1}$. Here, we encode the random keys differently.*
3. For all $j \in [\sigma]$, all $b \in \{0,1\}$, GEN computes $W_b^{(j)} \leftarrow (K_b^{(j)}, b \oplus \pi^{(j)})$, $\Theta_b^{(j)} \leftarrow \mathsf{com}(W_{b \oplus \pi^{(j)}}^{(j)})$, and $\Pi^{(j)} \leftarrow \mathsf{com}(\pi^{(j)})$.
4. GEN sends $\{\Theta_0^{(j)}, \Theta_1^{(j)}, \Pi^{(j)}\}_{j \in [\sigma]}$ to EVAL.

   Remark 4.3. *Now both parties have set up the common inputs for CCCT, including security parameter 1^k, statistical security parameter 1^σ, and commitments $\{\Theta_0^{(j)}, \Theta_1^{(j)}, \Pi^{(j)}\}_{j \in [\sigma]}$.*

Private Input: GEN has $x \in \{0,1\}$ and $\{K_0^{(j)}, K_1^{(j)}, \pi^{(j)}\}_{j \in [\sigma]}$.

Output: EVAL receives $\{W^{(j)}\}_{j \in E}$ and out $\in \{0,1\}$.

1. GEN picks $r \leftarrow D(I)$ and sends to EVAL $K \leftarrow f_x(r)$.

Remark 4.4. *Recall that the identical range distribution property ensures that K reveals no information about x, in particular, for all $x \in \{0,1\}$ and r in $f_x(\cdot)$'s domain, there always exists r' in $f_{1-x}(\cdot)$'s domain such that $f_{1-x}(r') = f_x(r)$.*

2. **(Check instances)**

   - For all $j \in [\sigma]$, Gen sends $(r_0^{(j)}, r_1^{(j)})$ and decommitments $(W_0^{(j)}, W_1^{(j)}, \pi^{(j)})$ over the check channel.
   - For all $j \in [\sigma]\backslash E$, Eval parses $W_0^{(j)}$ as $(K_0^{(j)}, \delta_0^{(j)})$ and $W_1^{(j)}$ as $(K_0^{(j)}, \delta_1^{(j)})$ such that $|\delta_0^{(j)}| = |\delta_1^{(j)}| = 1$. Eval checks the following:
     - if $W_0^{(j)}$ and $W_1^{(j)}$ successfully decommit $\Theta^{(j)}_{\delta_0^{(j)}}$ and $\Theta^{(j)}_{\delta_1^{(j)}}$, respectively;
     - if $\pi^{(j)}$ successfully decommits $\Pi^{(j)}$;
     - if $\pi^{(j)} = \delta_0^{(j)}$ and $\delta_0^{(j)} \oplus \delta_1^{(j)} = 1$; and
     - if $K_0^{(j)} = f_0(r_0^{(j)})$ and $K_1^{(j)} = f_1(r_1^{(j)})$.

     If any of the checks fails, Eval computes out $\leftarrow 0$.

3. **(Evaluation instances)**

   - For all $j \in [\sigma]$, Gen computes $d^{(j)} \leftarrow r^{(j)} - r$ and sends $d^{(j)}$ and decommitments $W^{(j)} \leftarrow W_x^{(j)}$ over the evaluation channel.
   - For $j \in E$, Eval parses $W^{(j)}$ as $(K^{(j)}, \delta^{(j)})$ such that $|\delta^{(j)}| = 1$. Eval then checks the following:
     - if $W^{(j)}$ successfully decommits $\Theta^{(j)}_{\delta^{(j)}}$; and
     - if $K^{(j)} = K \cdot R(d^{(j)})$.

     If any of the checks fails, Eval computes out $\leftarrow 0$.

Remark 4.5. *Recall that the cut-and-choose communication channels ensure that over the check channel,* Eval *receives only the communication for check instances; and likewise, over the evaluation channel,* Eval *receives only that for evaluation instances.*

4. If $\mathsf{out} \neq 0$, then EVAL computes $\mathsf{out} \leftarrow 1$.

5. Finally, EVAL outputs $(\{W^{(j)}\}_{j\in E}, \mathsf{out})$.

Lemma 4.1. *Protocol 4.1 is a secure CCCT instantiation.*

Proof. To show that this protocol is indeed a secure CCCT, we show its completeness, indistinguishability, and soundness as follows.

Completeness: If both parties are honest, all the decommitments reveal the corresponding commitments successfully. Moreover, $K \cdot R(d^{(j)}) = f_x(r) \cdot R(r^{(j)} - r) = f_x(r + r^{(j)} - r) = f_x(r^{(j)}) = K^{(j)}$, which means that all evaluation circuits will pass all the checks. Hence, EVAL always accepts.

Indistinguishability: Note that in EVAL's transcript from running this protocol, only K and $\{K^{(j)}\}$ have information about GEN's input x. Since $K = f_x(r)$ and $K^{(j)} = f_x(r^{(j)})$, we first define $\mathsf{tran}(x) = \left(f_x(r), \{f_x(r^{(j)})\}_{j\in E}\right)$. It suffices to show that $\{\mathsf{tran}(0)\} = \{\mathsf{tran}(1)\}$. Note that due to the identical range distribution property, there must exist $z \in \mathbb{G}_1$ such that $f_0(0) = f_1(z)$. As a result,

$$\begin{aligned}
\mathsf{tran}(0) &= \left\{\left(f_1(r), \{f_1(r^{(j)})\}_{j\in E}\right)\right\}_{r, r^{(j)} \in \mathbb{G}_1} \\
&= \left\{\left(f_0(0+r), \{f_0(0+r^{(j)})\}_{j\in E}\right)\right\}_{r, r^{(j)} \in \mathbb{G}_1} \\
&= \left\{\left(f_0(0)\cdot R(r), \{f_0(0)\cdot R(r^{(j)})\}_{j\in E}\right)\right\}_{r, r^{(j)} \in \mathbb{G}_1} && \text{(Malleability Property)} \\
&= \left\{\left(f_1(z)\cdot R(r), \{f_1(z)\cdot R(r^{(j)})\}_{j\in E}\right)\right\}_{r, r^{(j)} \in \mathbb{G}_1} && (f_0(0) = f_1(z)) \\
&= \left\{\left(f_1(z+r), \{f_1(z+r^{(j)})\}_{j\in E}\right)\right\}_{r, r^{(j)} \in \mathbb{G}_1} \\
&= \left\{\left(f_1(s), \{f_1(s^{(j)})\}_{j\in E}\right)\right\}_{s, s^{(j)} \in \mathbb{G}_1} \\
&= \mathsf{tran}(1)
\end{aligned}$$

Soundness: Let $F \subset E$ be those instances that are honest constructed. We first claim that if out $= 1$, then the probability that $|F| < |E|/2$ is negligible in σ. We further claim that GEN's inputs to those in F are consistent with high probability; otherwise, GEN must be capable of finding a claw.

Claim 4.2. $\Pr(|F| < |E|/2)$ *is negligible in* σ.

Proof. Recall that $|E| = c \cdot \sigma$ for some $0 < c < 1$. The event that $|F| < |E|/2$ happens when there are more than $|E|/2$ bad instances and none of them is chosen for check. The probability of this event is

$$\begin{aligned}
&\Pr(|F| < |E|/2) \\
&< \binom{\sigma - |E|/2}{\sigma - |E|}\binom{\sigma}{\sigma - |E|}^{-1} \\
&= \left(\frac{(\sigma - |E|/2)!}{(\sigma - |E|)!(|E|/2)!}\right)\left(\frac{\sigma!}{(\sigma - |E|)!(|E|)!}\right)^{-1} = \frac{(\sigma - |E|/2)!(|E|)!}{\sigma!(|E|/2)!} \\
&= \left(\frac{|E|/2 + 1}{\sigma - |E|/2 + 1}\right)\cdot\left(\frac{|E|/2 + 2}{\sigma - |E|/2 + 2}\right)\cdots\left(\frac{|E|/2 + |E|2}{\sigma - |E|/2 + |E|/2}\right) \\
&< \left(\frac{|E|}{\sigma}\right)^{|E|/2} = c^{-c\cdot\sigma/2}. && (|E| = c \cdot \sigma)
\end{aligned}$$

This probability decreases exponentially as σ increases and is therefore negligible. □

Claim 4.3. *Recall that* EVAL*'s output* $W^{(j)}$ *can be parsed as* $(K^{(j)}, \delta^{(j)})$ *such that* $|\delta^{(j)}| = 1$. *If out* $= 1$, *then for all* $a, b \in F$, $\delta^{(a)} \oplus \pi^{(a)} = \delta^{(b)} \oplus \pi^{(b)}$ *except with probability negligible in* k.

Proof. Let $x^{(j)}$ denote GEN's input to the j-th circuit. Recall that $\delta^{(j)} = x^{(j)} \oplus \pi^{(j)}$, or equivelently, $\delta^{(j)} \oplus \pi^{(j)} = x^{(j)}$. Hence, this claim is equivalent to proving that $x^{(a)} = x^{(b)}$ for all $a, b \in F$. To prove in contradiction, suppose there exist $a, b \in F$ such that $x^{(a)} \neq x^{(b)}$. Without loss of generality, let $x^{(a)} = 0$ and $x^{(b)} = 1$. We therefore know that

$$K^{(a)} = f_0(r^{(a)}) \text{ and } K^{(b)} = f_1(r^{(b)}) \tag{4.1}$$

for some $r^{(a)}, r^{(b)} \in \mathbb{G}_1$. Since out $= 1$, we also know that

$$K^{(a)} = K \cdot R(d^{(a)}) \text{ and } K^{(b)} = K \cdot R(d^{(b)}). \tag{4.2}$$

From Eqn (4.2), we know that $K^{(a)} \cdot R(d^{(a)})^{-1} = K = K^{(b)} \cdot R(d^{(b)})^{-1}$. Along with Eqn (4.1), we conclude that

$$f_0(r^{(a)}) \cdot R(d^{(a)})^{-1} = f_1(r^{(b)}) \cdot R(d^{(b)})^{-1} \Rightarrow f_0(r^{(a)} - d^{(a)}) = f_1(r^{(b)} - d^{(b)}).$$

In other words, Gen has found a claw $(r^{(a)} - d^{(a)}, r^{(b)} - d^{(b)})$, which happens only with probability negligible in k. □

We therefore prove the soundness and complete the proof of this lemma. □

Consider the claw-free collection constructed above under the Discrete Logarithm assumption, we know that it can become a malleable claw-free collection simply by letting $\mathbb{G}_1 = \mathbb{Z}_q$, $\mathbb{G}_2 = \mathbb{Z}_p^*$, and $R_I(r) = h^r$ for any $r \in \mathbb{G}_1$. We therefore have the following corollary.

Corollary 4.4. *If there exists a group in which the Discrete Logarithm assumption holds, then there exists a secure CCCT instantiation.*

4.1.2 CCOT via Committing OTs

Kiraz and Schoenmakers [KS06] introduced another notion of OT called *COT (committing oblivious transfer)* in which the receiver also receives perfectly-hiding commitments to both of the sender's inputs, and the sender receives as output the corresponding decommitments. In fact, Kiraz and Schoenmakers introduced this notion specifically for use in a Yao circuit evaluation context. We adopt the idea behind their construction.

Formally, a $\binom{2}{1}$-COT is a pair of interactive probabilistic polynomial-time algorithms sender and receiver that jointly run the following protocol

$$\binom{2}{1}\text{-COT} : (((m_0, r_0), (m_1, r_1)), b) \mapsto ((\phi_0, \phi_1), (m, \Phi_0, \Phi_1)),$$

where $b \in \{0,1\}$ and decommitment ϕ_c can reveal Φ_c to m_c for $c \in \{0,1\}$. Correctness requires that $m = m_b$ for all messages m_0, m_1, for all choice $b \in \{0,1\}$, and for all coin tosses of the algorithms. Here, we use the standard notion of security given in Definition 2.5.

Protocol 4.2. Secure CCOT via $\binom{2}{1}$-COTs

Common Input: security parameter 1^k, statistical security parameter 1^σ, perfectly-hiding commitment scheme com, and commitments $\{\Omega_0^{(j)}, \Omega_1^{(j)}\}_{j\in[\sigma]}$. Also, both parties have built the cut-and-choose communication channels. Let E denote the set of the evaluation instances.

Private Input: Gen has $\{W_0^{(j)}, W_1^{(j)}\}_{j\in[\sigma]}$ such that $W_b^{(j)} = (K_b^{(j)}, b \oplus \pi^{(j)})$ for some $\pi^{(j)} \in \{0,1\}$, and Eval has $y \in \{0,1\}$.

Output: Eval receives $\{W^{(j)}\}_{j\in[\sigma]}$ and out $\in \{0,1\}$.

1. For $j \in [\sigma], b \in \{0,1\}$, Gen computes $U_b^{(j)} \leftarrow W_b^{(j)}$ and picks $r_b^{(j)} \overset{R}{\leftarrow} \{0,1\}^{\text{poly}(k)}$.

2. Both parties jointly run a committing protocol

$$\binom{2}{1}\text{-COT} : (((\vec{U}_0, \vec{r}_0), (\vec{U}_1, \vec{r}_1)), y) \mapsto ((\vec{\phi}_0, \vec{\phi}_1), (\vec{U}, \vec{\Phi}_0, \vec{\Phi}_1)),$$

where for $b \in \{0,1\}$, $\vec{U}_b$ denotes local data $\{U_b^{(j)}\}_{j\in[\sigma]}$, $\vec{r}_b$ denotes local randomness $\{r_b^{(j)}\}_{j\in[\sigma]}$, $\vec{\phi}_b$ denotes decommitments $\{\phi_b^{(j)}\}_{j\in[\sigma]}$, $\vec{U}$ denotes Eval's retrieved labels $\{U^{(j)}\}_{j\in[\sigma]}$, and $\vec{\Phi}_b$ denotes the commitment $\{\Phi_b^{(j)}\}_{j\in[\sigma]}$.

Remark 4.6. *Note that if* Gen *is honest, then $U_b^{(j)} = W_b^{(j)}$ for all $j \in [\sigma], b \in \{0,1\}$. The security properties of the committing $\binom{2}{1}$-COT then ensure that once the $\binom{2}{1}$-COT successfully completes,* Eval*'s output $U^{(j)} = U_y^{(j)} = W_y^{(j)}$ and $\phi_b^{(j)}$ reveals $\Phi_b^{(j)}$ to $U_b^{(j)} = W_b^{(j)}$ for all $j \in [\sigma], b \in \{0,1\}$.*

However, Gen *could be malicious. We therefore resort to the cut-and-choose technique that we check those in $[\sigma]\backslash E$. This guarantees the correctness of a majority in E, which achieves the goal of a secure CCOT.*

3. For all $j \in [\sigma]$ and $b \in \{0,1\}$, GEN sends decommitments $(\omega_b^{(j)}, \phi_b^{(j)})$ over the check channel.

4. **(Check the check instances)** For all $j \in [\sigma]\backslash E$, EVAL checks the following:

   - if decommitment $\omega_b^{(j)}$ successfully reveals commitment $\Omega_b^{(j)}$, a common input, to label $W_b^{(j)}$ for $b \in \{0,1\}$;
   - if decommitment $\phi_b^{(j)}$ successfully reveals commitment $\Phi_b^{(j)}$, an output from the $\binom{2}{1}$-COT, to label $U_b^{(j)}$ for $b \in \{0,1\}$;
   - if the revealed $W_b^{(j)}$ and $U_b^{(j)}$ coincide for $b \in \{0,1\}$; and
   - if labels $(W_0^{(j)}, W_1^{(j)})$ are in the format that $W_b^{(j)} = (K_b^{(j)}, b \oplus \pi^{(j)})$ for some $\pi^{(j)} \in \{0,1\}$ for $b \in \{0,1\}$.

   If any of the checks fails, EVAL computes out $\leftarrow 0$. Otherwise, out $\leftarrow 1$.

5. Finally, EVAL outputs $(\{W^{(j)}\}_{j\in[\sigma]}, \text{out})$.

Lemma 4.5. *Protocol 4.2 is a secure CCOT instantiation.*

Theorem 4.6 ([NP01])**.** *If there exists a group in which the Decisional Diffie-Hellman assumption holds, there exists a protocol securely computes the committing $\binom{2}{1}$-OT.*

Protocol 4.3 constructively demonstrates this theorem. This protocol is a simple modification of the OT protocols designed by Bellare and Micali [BM89] and later Naor and Pinkas [NP01]. We simply add a ZK proof of knowledge in intermediate steps. Intuitively, the receiver-security is achieved due to the Decisional Diffie-Hellman assumption and the fact that the ZK proof of knowledge is independent of the receiver's input. On the other hand, the sender security comes from the uniform distributions of $X_b^{(j)}$ and $Y_b^{(j)}$ over G given $r_b^{(j)}$ and $s_b^{(j)}$ are uniformly chosen and the fact that the ZK proof has an ideal-world simulator for the verifier (or the receiver in the OT). Note that the efficient proof of knowledge is due to Cramer, Damgård, and MacKenzie [CDM00].

Protocol 4.3. A $\binom{2}{1}$-COT based on the Decisional Diffie-Hellman Assumption

Common Input: Security parameter 1^k, statistical security parameter 1^σ, perfectly-hiding commitment scheme com, group $\mathbb{G}$ of prime order p, and generator g of group $\mathbb{G}$.

Private Input: Gen has input $((\vec{W}_0, \vec{r}_0), (\vec{W}_1, \vec{r}_1))$, where $\vec{W}_b$ denotes private data $\{W_b^{(j)}\}_{j\in[\sigma]}$ and $\vec{r}_b$ denotes randomness $\{r_b^{(j)}\}_{j\in[\sigma]}$. Eval has input $y \in \{0,1\}$.

Output: Gen receives $\{\vec{\phi}_0, \vec{\phi}_1\}$, where for $b \in \{0,1\}$, $\vec{\phi}_b$ denotes decommitments $\{\phi_b^{(j)}\}_{j\in[\sigma]}$. Eval receives $(\vec{W}, \vec{\Phi}_0, \vec{\Phi}_1)$, where $\vec{W}$ denotes received input $\{W^{(j)}\}_{j\in[\sigma]}$ and $\vec{\Phi}_b$ denotes commitments $\{\Phi_b^{(j)}\}_{j\in[\sigma]}$ for $b \in \{0,1\}$.

1. Eval picks $a, b \xleftarrow{R} \mathbb{Z}_p$ and computes $g_0 \leftarrow g$, $g_1 \leftarrow g^a$, and $(h_0, h_1) \leftarrow (g_0^b, g_1^{b+1})$. Then Eval sends (g_0, g_1, h_0, h_1) to Gen.

2. Eval proves in ZK the knowledge of b such that $(h_0 = g_0^b) \wedge (h_1 = g_1^{b+1})$.

3. Eval picks $r \xleftarrow{R} \mathbb{Z}_p$, computes $(g, h) \leftarrow (g_y^r, h_y^r)$, and then sends (g, h) to Gen.

4. For $j \in [\sigma], b \in \{0,1\}$, Gen

   (a) maps $W_b^{(j)}$ into some $K_b^{(j)} \in \mathbb{G}$;

   (b) uses $r_b^{(j)}$ as randomness to compute $s_b^{(j)}, t_b^{(j)} \in \mathbb{Z}_p$;

   (c) lets $\phi_b^{(j)} \leftarrow (W_b^{(j)}, r_b^{(j)})$;

   (d) computes $X_b^{(j)} \leftarrow g_b^{s_b^{(j)}} h_b^{t_b^{(j)}}$ and $Y_b^{(j)} \leftarrow g^{s_b^{(j)}} h^{t_b^{(j)}} \cdot K_b^{(j)}$; and

   (e) sends $(X_b^{(j)}, Y_b^{(j)})$ to Eval.

5. For $j \in [\sigma]$, Eval

1

(a) computes $K^{(j)} \leftarrow Y_y^{(j)} \cdot (X_y^{(j)})^{-1}$;

(b) unmaps $K^{(j)}$ to $W^{(j)}$; and

(c) lets $\Phi_0^{(j)} \leftarrow (X_0^{(j)}, Y_0^{(j)})$ and $\Phi_1^{(j)} \leftarrow (X_1^{(j)}, Y_1^{(j)})$.

6. Finally, Gen outputs $(\vec{\phi}_0, \vec{\phi}_1)$ and Eval outputs $(\vec{W}, \vec{\Phi}_0, \vec{\Phi}_1)$.

Remark 4.7. *Mapping a random input $W_b^{(j)}$ into group element $K_b^{(j)} \in \mathbb{G}$ and its reverse operation should be made public before this protocol starts. More importantly, this mapping has to be injective.*

Corollary 4.7. *If there exists a group in which the Decisional Diffie-Hellman assumption holds, then there exists a secure CCOT instantiation.*

4.2 Secure 2PC via Auxiliary Circuits

In this section, we study the use of auxiliary circuits in order to instantiate CCCT and CCOT modules. Generally speaking, to instantiate the modules, the standard technique requires Gen to provide a proof that it has behaved honestly during the collaboration. We observed that the Yao protocol is itself a powerful tool to provide such a proof. So we propose that besides the objective circuit, both participants jointly work on a few auxiliary circuits, whose output is exactly the proof of honest behavior that Eval expects.

Secure CCCT vis 2-Universal Hash Circuit In Section 4.2.1, we suggest to instantiate a secure CCCT by using a 2-universal hash circuit. The insight is that the cut-and-choose technique will ensure that this auxiliary circuit is correctly constructed and its output (the hash) will ensure that Gen's inputs are consistent among the evaluation circuits but nothing about their exact values. For this technique to work, we need to endow a 2-universal hash of Gen's input with the *collision-free* and *hiding* properties. The former ensures that the consistency of the hashes implies the consistency of GEN's inputs, and the latter ensures that the hashes do not reveal information about GEN's input.

Secure CCOT via Probe-Resistant Matrices Lindell and Pinkas suggested an auxiliary circuit to deter the Selective Failure Attack [LP07]. In their solution, EVAL picks $M \in \{0,1\}^{n\times m}$ for some $m \in \mathbb{N}$ and computes its new input $\bar{y} \in \{0,1\}^m$ such that $M \cdot \bar{y} = y$. The auxiliary circuit will later convert $\bar{y}$ back to y to ensure correctness. The insight is that now the fact that GEN gets to probe some partial information about $\bar{y}$, due to Input Key Inconsistency Attack, should not grant it non-negligible advantage to learn partial information about EVAL's real input y. In Section 4.2.2, we first capture the security property needed and formulate the *probe-resistant* matrices. Then we present a secure CCOT that is instantiated from a probe-resistant matrix circuit. Finally, we suggest parameters that achieve the asymptotically optimal performance.

It is worth-mentioning that these auxiliary circuits were not taken into serious consideration previously for they were considered too costly. Indeed, back in 2009, Pinkas et al. were able to report a system that only handles a circuit of 33,880 gates at a rate of 40 gates per second [PSSW09]. A system that is able to handle a circuit of billions of gates at a rate of hundreds of thousands of gates per second is only recently reported [KShS12]. So the burden of those auxiliary circuits is no longer unbearable.

4.2.1 CCCT via 2-Universal Hash Circuit

In this section, we first give the definition of universal hash functions, and then we discuss how we endow a universal hash circuit with both collision-free and hiding properties. Finally, we present a secure CCCT and sugguest proper parameters.

Definition 4.3 (Universal Hash). *A collection of hash functions $\mathbb{H} = \{h|h : A \to B\}$ is called* universal *if for any distinct $x, y \in A$, the probability that a uniformly chosen $h \in \mathbb{H}$ satisfies that $h(x) = h(y)$ is at most $1/|B|$.*

The collision-free property comes naturally with universal hash functions. Indeed, Definition 4.3 shows that for any distinct x, y, if they are *fixed* before h is *uniformly* chosen, their hashes are unlikely to collide. This suggests that the collision-free property can be achieved by letting Gen commit to (fix) its inputs before a hash function is jointly (uniformly) picked. Later, the consistency of Gen's inputs can be implied by the consistency of the outputs—the hashes. Our protocol is outlined as follows:

1. Gen commits to its inputs $x^{(1)}, x^{(2)}, \ldots, x^{(\sigma)}$, where $x^{(j)}$ denotes Gen's input to the j-th circuit.

2. Gen and Eval jointly and uniformly pick $h \in \mathbb{H}$.

3. Gen constructs σ copies of the garbled circuit that computes $(x, \perp) \mapsto (\perp, h(x))$.

4. Eval asks to check the correctness of a random fraction of the garbled circuits. If the check fails, Eval aborts; otherwise, Eval asks Gen to decommit to its inputs for the remaining (unchecked) circuits.

5. Eval evaluates the remaining circuits. If the evaluation outputs (the hashes) are not consistent, Eval aborts; otherwise, Eval accepts.

We next argue, at a high level, that with high probability, this protocol guarantees the desired collision-free property, that is, Gen's inputs to the *majority* of the remaining circuits are consistent. Indeed, if a malicious Gen is able to pass the hash consistency check and provide inconsistent inputs to the majority of the remaining circuits, there are only three possibilities:

1. The circuits are faulty: The cut-and-choose technique ensures that with high probability, this only happens to a minority of the remaining circuits.

2. Gen is really lucky to have found a collision: By Definition 4.3, this happens with probability at most $1/|B|$, which becomes negligible if B is properly chosen.

3. Gen is able to break the binding property of the commitments: Note that since universal hash functions do not even provide pre-image resistance, given h and $h(x^{(a)})$, it can be easy to find $x^{(b)}$ such that $h(x^{(a)}) = h(x^{(b)})$. Therefore, if Gen is able to open the commitment from Step 1 to some value computed after h is chosen in Step 2, it breaks the desired security property. However, this would imply that Gen is able to break the commitment scheme's binding property. This happens with negligible probability too.

A deterministic universal hash $h : A \rightarrow B$ provides the collision-free property we need, yet it is insufficient for the purposes here due to the lack of the hiding property. Indeed, if the size of A is small, Eval could exhaustively compute the hash of all possibilities in A and then deduce $x^{(j)} \in A$ from $h(x^{(j)})$. As a result, the hash function has to be randomized. We further observe that if the hashes are pseudo-random, they reveal little information about the input, which is the hiding property we desire. In particular, Leftover Hash Lemma (Lemma 4.8) states that the output of a uniformly picked universal hash function h is pseudo-random (even if h is made public) as long as the input has enough entropy. All we need to do is introduce entropy to the input. We hence suggest that objective function $(x, \perp) \mapsto (\perp, h(x))$ is converted to $(x||r, \perp) \mapsto (\perp, h(x||r))$, where r is a proper randomness picked by Gen at the beginning and h is a universal hash function uniformly picked *after* Gen commits to its new input $x||r$.

Lemma 4.8 (Leftover Hash Lemma [IZ89]). *Let X_n denote $\{0,1\}^n$ for some $n \in \mathbb{N}$, and let $\mathbb{H} = \{h | h : X_{k+2\varepsilon} \rightarrow X_k\}$ be a collection of universal hash functions for some $\varepsilon, k \in \mathbb{N}$. Distributions $\{(h, h(x))\}$ and $\{(h, y)\}$ are statistically indistinguishable with probability at least $1 - 2^{-\varepsilon}$, where h, x, and y are uniformly chosen from $\mathbb{H}$, $X_{k+2\varepsilon}$, and X_k, respectively.*

We argue that the introduction of random input r does provide the hiding property even when h is public. Indeed, given $h(x||r)$ and h, as long as r is long enough (has enough entropy), for any x', there must exist r' such that $h(x||r) = h(x'||r')$. This shows that fixing $h(x||r)$ does not rule out any possibilities of x.

Next, we suggest the use of the matrix universal hash family

$$\mathbb{M}_{k,m} = \{h_M | \; h_M(x) = M \cdot x \text{ for some } M \in \{0,1\}^{k \times m}\},$$

for some $m \in \mathbb{N}$. A nice property of this hash family is that it is $\oplus$-homomorphic, that is, for any $x, y \in \{0,1\}^m$ and $h_M \in \mathbb{M}_{k,m}$, it holds that $h_M(x \oplus y) = h_M(x) \oplus h_M(y)$. This homomorphism allows a very efficient instantiation of the above protocol outline with the following twists:

1. Gen commits not to $x^{(j)}$ directly but to $x^{(j)} \oplus \pi^{(j)}$ and $\pi^{(j)}$ instead, where $\pi_i^{(j)}$ is random bit assigned to the wire corresponding to Gen's i-th input bit.
2. Gen provides not a garbled circuit that lets Eval compute $h(x^{(j)})$ obliviously but rather actual hash $h(\pi^{(j)})$ that Gen computes locally.
3. Let E be the set of the evaluation circuits.
    - For $j \in [\sigma] \backslash E$, Eval checks not the correctness of the j-th garbled circuit but the correctness of $h(\pi^{(j)})$ (by asking Gen to decommit to $\pi^{(j)}$ first).
    - For $j \in E$, Eval learns $h(x^{(j)})$ not via evaluating the j-th garbled circuit but via asking Gen to decommit to $z^{(j)} = x^{(j)} \oplus \pi^{(j)}$ and then computing $h(z^{(j)}) \oplus h(\pi^{(j)})$.

The main goal of the above twist is to replace the garbled circuit that computes $h(x^{(j)})$ with actual hash $h(\pi^{(j)})$ that is locally computed while maintaining the main structure of the protocol, that is, letting Gen commit to its inputs before h is jointly picked and letting Eval learn the hash of the input to the evaluation circuits.

The complete protocol of the secure CCCT instantiated from 2-universal hash circuit is presented in Protocol 4.4.

Protocol 4.4. Secure CCCT via 2-Universal Circuit

Common Input: security parameter 1^k, statistical security parameter 1^σ, perfectly-hiding commitment scheme com, and commitments $\{\Theta_{i,0}^{(j)}, \Theta_{i,1}^{(j)}, \Pi_i^{(j)}\}_{j\in[\sigma],i\in[n]}$. Also, both parties have built the cut-and-choose communication channels. Let E denote the set of the evaluation instances.

Private Input: GEN has $x \in \{0,1\}^n$ and $\{K_{i,0}^{(j)}, K_{i,1}^{(j)}, \pi_i^{(j)}\}_{j\in[\sigma],i\in[n]}$.

Output: EVAL receives $\{W_i^{(j)}\}_{j\in E,i\in[n]}$ and out $\in \{0,1\}$.

1. **(Compute New Input)** GEN picks $r \overset{R}{\leftarrow} \{0,1\}^t$ for some $t \in \mathbb{N}$ and computes $\bar{x} \leftarrow (x,r)$ and $m \leftarrow n+t$.
2. For $j \in [\sigma], i \in [t]$, GEN picks $(K_{n+i,0}^{(j)}, K_{n+i,1}^{(j)}, \pi_{n+i}^{(j)}) \leftarrow \{0,1\}^{2k+1}$.
3. **(Fix GEN's Input)** For $j \in [\sigma], i \in [m]$, GEN commits to $W_{i,\bar{x}_i}^{(j)}$ by sending $\Psi_i^{(j)}$ to EVAL, where $W_{i,b}^{(j)} = (K_{i,b}^{(j)}, b \oplus \pi_i^{(j)})$ and $\Psi_i^{(j)} = \mathsf{com}(W_{i,\bar{x}_i}^{(j)})$.
4. **(Uniformly Pick a Hash)** Both parties jointly run a coin toss protocol to uniformly pick $M \in \{0,1\}^{k\times m}$.
5. **(Securely Compute GEN's Input Hashes)** Both parties jointly run the twisted version of the cut-and-choose-based Yao protocol as previously discussed. More specifically, both parties work as follows:
   (a) **(Commit to Input Label Pairs)** GEN commits to the label pairs assigned to all the input wires just like in the Main protocol. Recall that the commitments to the label pairs corresponding to the first n wires are already common inputs. It remains to commitment to the new t wires. Therefore, GEN commits to $\{W_{n+i,0}^{(j)}, W_{n+i,1}^{(j)}, \pi_{n+i}^{(j)}\}_{j\in[\sigma],i\in[t]}$ by sending $\{\Theta_{n+i,0}^{(j)}, \Theta_{n+i,1}^{(j)}, \Pi_{n+i}^{(j)}\}_{j\in[\sigma],i\in[t]}$ to EVAL,

where

$$\Theta^{(j)}_{n+i,b} = \mathsf{com}(W^{(j)}_{n+i,b\oplus\pi^{(j)}_{n+i}}) \text{ and } \Pi^{(j)}_{n+i} = \mathsf{com}(\pi^{(j)}_{n+i}).$$

(b) **(Retrieve the Twisted Garbled Circuit)** For $j \in [\sigma]$, GEN sends $h^{(j)}_\pi$ to EVAL, where $h^{(j)}_\pi = M \cdot (\pi^{(j)}_1 || \pi^{(j)}_2 || \cdots || \pi^{(j)}_m)$.

(c) **(Check Instances)**

- For all $j \in [\sigma]$, GEN sends decommitments $\{W^{(j)}_{i,0}, W^{(j)}_{i,1}, \pi^{(j)}_i\}_{i\in[m]}$ over the check channels.
- For all $j \in [\sigma] \backslash E$, EVAL checks the following:
  i. if $\theta^{(j)}_{i,b}$ successfully decommits $\Theta^{(j)}_{i,b}$ to $W^{(j)}_{i,b\oplus\pi^{(j)}_i}$ for $b \in \{0,1\}$ such that $W^{(j)}_{i,0} = (K^{(j)}_{i,0}, \pi^{(j)}_i)$ and $W^{(j)}_{i,1} = (K^{(j)}_{i,1}, 1 - \pi^{(j)})$ for some $K^{(j)}_{i,0}, K^{(j)}_{i,1} \in \{0,1\}^k$;
  ii.

(d) **(Evaluation Instances)**

- For all $j \in [\sigma]$, GEN sends decommitments $\{\psi^{(j)}_i, \theta^{(j)}_{i,\bar{x}_i\oplus\pi^{(j)}_i}\}_{i\in[m]}$ over the evaluation channels.
- For each $j \in E$, EVAL checks the following:
  i. if $\theta^{(j)}_{i,b}$ successfully reveals $\Theta^{(j)}_{i,b}$ to $W^{(j)}_{i,b\oplus\pi^{(j)}_i}$ for $i \in [m]$, $b \in \{0,1\}$;
  ii. if $\phi^{(j)}_{i,b}$ successfully reveals $\Phi^{(j)}_{i,b}$ to $U^{(j)}_{i,b\oplus\pi^{(j)}_i}$ for $i \in [m]$, $b \in \{0,1\}$;
  iii. if $U^{(j)}_{i,b} = W^{(j)}_{i,b}$ for $i \in [m]$, $b \in \{0,1\}$; and
  iv. if $G(C)^{(j)}$ is an honest garbled version of C such that $(W^{(j)}_{i,0}, W^{(j)}_{i,1})$ is indeed the label pair assigned to the i-th input wire.

  If any of the checks fails, EVAL computes out $\leftarrow 0$.

6. If out $\neq 0$, EVAL computes out $\leftarrow 1$.

7. Finally, EVAL outputs $(\{W^{(j)}_i\}_{j\in E, i\in[n]}, \mathsf{out})$.

1

Lemma 4.9. *Assume that the $\binom{2}{1}$-OT protocol is secure in the presence of malicious adversaries, and there exist a perfectly-hiding commitment scheme, and a semantically secure symmetric encryption scheme. Protocol 4.4 is a secure CCCT.*

It remains to show the parameter selection for the desired security parameter k. By the definition of 2-universal hash function family, GEN cannot pick distinct $x^{(i)}, x^{(j)} \in A$ whose hashes collide with probability better than $1/|B|$. So the size of B needs to be at least 2^k. Next, the Leftover Hash Lemma states that if the input's min-entropy minus the output hash's entropy is great than $2\lg(1/\varepsilon)$, then the output hash is distinguished from a truly random string with probability at most ε. In other words, the output hash is k-bit long and 2^{-k}-indistinguishable from truly random if the min-entropy of $x||r$ is at least $k + 2\lg(1/2^{-k}) = 3k$. To put the minimal assumption on GEN's input entropy, a simple approach suggested by the Leftover Hash Lemma is to append $3k$ random bits after x, that is, $|r| = 3k$. However, by exploiting the properties of the specific hash family $\mathbb{M}$, we can do a little better than this, that is, we can reach the same goal with $|r| = 2k + \lg(k)$.

Lemma 4.10. *Let $\mathbb{M} = \{h_M | \ M \in \{0,1\}^{k\times(n+t)}$ and $h_M(x) = M \cdot x$ for some $k, n, t \in \mathbb{N}\}$. For any $x \in \{0,1\}^n$, if r is uniformly distributed over $\{0,1\}^t$ such that $t \geq 2k + \lg(k)$, the probability that a randomly picked $h_M \in \mathbb{M}$ satisfies that $h_M(x||r)$ is uniformly distributed over $\{0,1\}^k$ is at least $1 - 2^{-k}$.*

Proof. Note that picking $h_M \in \mathbb{M}$ is equivalent to picking $M \in \{0,1\}^{k\times(n+t)}$. We first fix some $x \in \{0,1\}^n$. For a random $M \in \{0,1\}^{k\times(n+t)}$, let $M = [M_1, M_2]$, where $M_1 \in \{0,1\}^{k\times n}$ and $M_2 \in \{0,1\}^{k\times t}$. For any $r \in \{0,1\}^t$, we know that

$$h_M(x||r) = M \cdot \begin{bmatrix} x \\ r \end{bmatrix} = M_1 x + M_2 r.$$

For $h_M(x||r)$ to be uniformly distributed over $\{0,1\}^k$, it suffices that $M_2 r$ is uniformly distributed over $\{0,1\}^k$, which happens if and only if the rank of M_2 is k. So all we need to do is count the number of full rank matrices of size $k \times t$. Starting from the first row of M_2, there are $2^t - 1$ choices (all but the all-zero vector). The second row can be any vector outside the space spanned by the first row, and there are $2^t - 2$ choices. Following the same logic, the i-th row can be any vector outside the space spanned by the first $(i-1)$ rows, and there are $2^t - 2^{i-1}$ choices. Therefore, among all the 2^{kt} possibilities, the number of full rank matrices is

$$F = \prod_{i=0}^{k-1}(2^t - 2^{i-1}) \geq (2^t - 2^{k-1})^k = 2^{kt}(1 - 2^{k-1-t})^k \geq 2^{kt}(1 - k2^{k-1-t}).$$

As a result, the probability that $h_M(x||r)$ is uniformly distributed over $\{0,1\}^k$ is the same as the probability that M_2 has full rank, which is

$$\frac{F}{2^{kt}} \geq 1 - k \cdot 2^{k-1-t} \geq 1 - k \cdot 2^{k-1-2k-\lg(k)} = 1 - 2^{-1-k} > 1 - 2^{-k}.$$

$\square$

4.2.2 CCOT via Probe-Resistant-Matrix Circuit

We first give the formal definition of the *probe-resistant* matrices as follows:

Definition 4.4. *$M \in \{0,1\}^{n\times m}$ for some $n, m \in \mathbb{N}$ is called k-*probe-resistant *for some $k \in \mathbb{N}$ if for any $L \subset [n]$, the Hamming distance of $\bigoplus_{i\in L} M_i$ is at least k, where M_i denotes the i-th row of M.*

As long as M is k-probe-resistant, a malicious Gen will have to successfully probe k bits of $\bar{y}$ in order to gain any partial information of y, the probability of which is negligible. We stress that matrix M can even be made public so that the auxiliary circuit could consist of only XOR-gates, which requires no communication overhead and can be computed efficiently when combining with the free-XOR technique [KS08b]. In short, not only does this approach elegantly reduce the selective failure attack problem to the classic error correcting code construction, it also incurs very little overhead.

We next give the full description of the secure CCOT instantiated from probe-resistant matrix circuit in Protocol 4.5.

Protocol 4.5. Secure CCOT via Probe-Resistant Matrix Circuit

Common Input: security parameter 1^k, statistical security parameter 1^σ, perfectly-hiding commitment scheme com, and commitments $\{\Omega_{i,0}^{(j)}, \Omega_{i,1}^{(j)}\}_{j\in[\sigma],i\in[n]}$. Also, both parties have built the cut-and-choose communication channels. Let E denote the set of the evaluation instances.

Private Input: GEN has $\{W_{i,0}^{(j)}, W_{i,1}^{(j)}\}_{j\in[\sigma],i\in[n]}$ such that $W_{i,b}^{(j)} = (K_{i,b}^{(j)}, b \oplus \pi_i^{(j)})$ for some $\pi_i^{(j)} \in \{0,1\}$, and EVAL has $y \in \{0,1\}^n$ for some $n \in \mathbb{N}$.

Output: EVAL receives $\{W_i^{(j)}\}_{j\in[\sigma],i\in[n]}$ and out $\in \{0,1\}$.

1. EVAL randomly computes a k-probe-resistant matrix $M \in \{0,1\}^{m\times n}$.

2. EVAL computes $\bar{y} \in \{0,1\}^m$ such that $M \cdot \bar{y} = y$.

   Remark 4.8. *$\bar{y}$ can always be efficiently computed by Gaussian Elimination.*

3. EVAL reveals (m, M) so that both parties can agree upon the auxiliary circuit C that computes $(\perp, z) \mapsto (\perp, M \cdot z)$ for any $z \in \{0,1\}^m$.

4. Both parties jointly run σ copies of the Yao protocol over the auxiliary circuit in the way that the i-th output bit of the j-th circuit are assigned label pair $(W_{i,0}^{(j)}, W_{i,1}^{(j)})$, which is GEN's private input to this protocol. More specifically, both parties work as follows:

   (a) GEN garbles σ copies of circuit C. We denote by $G(C)^{(j)}$ the j-th circuit, by $(U_{i,0}^{(j)}, U_{i,1}^{(j)})$ the label pair assigned to the i-th input wire of $G(C)^{(j)}$, and by $(V_{i,0}^{(j)}, V_{i,1}^{(j)})$ the label pair assigned to the i-th output wire of $G(C)^{(j)}$. 1

(b) GEN commits to $(U_{i,0}^{(j)}, U_{i,1}^{(j)})$ by sending $(\Delta_0^{(j)}, \Delta_1^{(j)})$ to EVAL, where

$$\Delta_{i,b}^{(j)} = \mathsf{com}(U_{i,b}^{(j)}).$$

(c) GEN sends $\{G(C)^{(j)}\}_{j\in[\sigma]}$ to EVAL.

(d) Both parties jointly execute m instances of $\binom{2}{1}$-OT. In the i-th instance, both parties run

$$\mathsf{OT} : ((\vec{\delta}_{i,0}, \vec{\delta}_{i,1}), \bar{y}_i) \mapsto (\perp, \vec{\delta}_i),$$

where for $b \in \{0,1\}$, $\vec{\delta}_{i,b}$ denotes decommitments $\{\delta_{i,b}^{(j)}\}_{j\in[\sigma]}$ and $\vec{\delta}_i$ denotes the received decommitments $\{\delta_i^{(j)}\}_{j\in[\sigma]}$.

Remark 4.9. *If* GEN *is honest, decommitment $\delta_{i,b}^{(j)}$ can successfully reveal commitment $\Delta_{i,b}^{(j)}$ to input label $U_{i,b}^{(j)}$ for all $j \in [\sigma], i \in [n]$, and $b \in \{0,1\}$.*

(e) GEN sends $\{\delta_{i,b}^{(j)}, \omega_{i,b}^{(j)}\}_{j\in[\sigma],b\in\{0,1\}}$ over the check channel.

(f) For each $j \in [\sigma]\backslash E$, EVAL checks the following:

i. if $\delta_{i,b}^{(j)}$ successfully reveals $\Delta_{i,b}^{(j)}$ to $U_{i,b)}^{(j)}$ for $i \in [m]$, $b \in \{0,1\}$;

ii. if $\omega_{i,b}^{(j)}$ successfully reveals $\Omega_{i,b}^{(j)}$ to $W_{i,b}^{(j)}$ for $i \in [n]$, $b \in \{0,1\}$; and

iii. if $G(C)^{(j)}$ is an honest garbled version of C such that $(U_{i,0}^{(j)}, U_{i,1}^{(j)})$ is indeed the label pair assigned to the i-th input wire and $(W_{i,0}^{(j)}, W_{i,1}^{(j)})$ indeed the label pair assigned to the i-th output wire.

If any of the checks fails, EVAL computes out $\leftarrow 0$; otherwise, EVAL computes $W_i^{(j)} \leftarrow W_{i,y_i}^{(j)}$ for all $j \in [\sigma]\backslash E$ and $i \in [n]$.

(g) For each $j \in E$, EVAL

i. checks if $\delta_i^{(j)}$ successfully reveals $\Delta_{i,\bar{y}_i}^{(j)}$ to $U_{i,\bar{y}_i}^{(j)}$ for $i \in [m]$, and

ii. evaluates $G(C)^{(j)}$ with input labels $\{U^{(j)}\}_{j\in[m]}$.

If any of the checks fails, EVAL computes out $\leftarrow 0$; otherwise, let $\{W_i^{(j)}\}_{j\in E,i\in[n]}$ be the computed labels corresponding to all output wires. 1

5. If out $\neq 0$, Eval computes out $\leftarrow 1$.

6. Finally, Eval outputs $(\{W_i^{(j)}\}_{j\in[\sigma],i\in[n]}, \mathsf{out})$.

Lemma 4.11. *Assume that the $\binom{2}{1}$-OT protocol is secure in the presence of malicious adversaries, and there exist a perfectly-hiding commitment scheme, and a semantically secure symmetric encryption scheme. Protocol 4.5 is a secure CCOT.*

In this work, we would like to improve upon the randomized construction proposed by Lindell and Pinkas. Their construction works as follows: Let M be randomly picked from $\{0,1\}^{n\times m}$. As long as m is big enough, the probability that M fails to be k-probe-resistant will be negligible. In particular, they suggest that $m = \max(4n, 8k)$. This approach is intuitive and simple, but there is room for improvement. Indeed, our study shows that it is possible to reduce m to less than $2n$ when n is big enough, for example, when $n = 1024$, we are able to generate an 80-probe-resistant M such that $m = 1752$.

We believe this improvement is worthwhile for two reasons. First, the computation of M can be done locally by the evaluator alone. This local computation is insignificant compared with circuit garbling, while the benefits could be as big as 75% saving in computation and communication. Moreover, M can be even computed beforehand. Second, it improves the scalability. It is true that the auxiliary circuit incurs little overhead with the help of the free-XOR technique. However, while XOR gates can be computed almost for free, the oblivious transfers can not. In fact, oblivious transfers are arguably the most expensive part (on average) due to the need of number-theoretic operations. It is also true that there are extension techniques that provide the amortized complexity of $O(1)$ symmetric cryptographic operations per oblivious transfer. Nonetheless, not all variants of oblivious transfer have a corresponding extension technique. For example, it is unknown if the committing oblivious transfer [SS11] or the cut-and-choose oblivious transfer [LP11] could be extended. In these cases, reducing the number of the real OTs (as opposed to extended OTs) needed makes a difference.

Our idea to construct a k-probe-resistant M is to use a maximum distance separable code such as Reed-Solomon codes. We will work on the Reed-Solomon code over $\mathbb{F}_{2^t}$ for some $t \in \mathbb{N}$ with codeword length N and message length K. Basically, we start by randomly picking polynomials $P_1, P_2, \ldots, P_n \in \mathbb{F}_{2^t}[x]$ with degree at most $K-1$, where n is EVAL's input size. Then we evaluate these polynomials at points $x_1, x_2, \ldots, x_N \in \mathbb{F}_{2^t}$. The outputs for P_i over these points are concatenated as a Nt-bit vector and becomes the i-th row of M. The goal now is to minimize the goal function $m = Nt$ under the constraint that M is k-probe-resistant with high probability.

For M to be k-probe-resistant, there are four constraints on parameters (N, K, t). The first two are inherited from Reed-Solomon code: the minimum distance has to be at least k, that is,

$$N - K + 1 \geq k, \tag{4.3}$$

and there needs to be enough non-zero points for the polynomials to be evaluated at, that is,

$$2^t - 1 \geq N. \tag{4.4}$$

The third constraint is that there needs to be enough distinct non-zero polynomials to choose from. Indeed, Reed-Solomon code guarantees that two codewords differ at k or more places only if the corresponding polynomials are *distinct*. Since each bit in y needs a corresponding polynomial, we have that

$$(2^t)^K - 1 = 2^{Kt} - 1 \geq n. \tag{4.5}$$

The last and the most important constraint is that each polynomial cannot be a subset sum of the rest. Let us consider the case that $P_3 = P_1 + P_2$. In this case, we know that M has rank at most $n-1$. The consequences of this case are twofold. On one hand, the range of the transformation induced by M has dimension at most $n-1$. As a result, for some $y \in \{0,1\}^n$, there exists no $\bar{y}$ such that $M \cdot \bar{y} = y$. On the other hand, if there indeed exists such a $\bar{y}$, a malicious GEN can easily deduce that $y_3 = y_1 + y_2$, which is the partial information of y that GEN is not supposed to learn. Lemma 4.12 shows that with probability at least $1 - 2^{-k}$, no polynomial is a subset sum of the rest as long as

$$K \geq (\lg(n) + n + k)/t. \tag{4.6}$$

Lemma 4.12. *If $K \geq (\lg(n) + n + k)/t$, the probability that for any $i \in \{1, 2, \ldots, n\}$ there exists no $L \subset \{1, 2, \ldots, n\} \backslash \{i\}$ such that $P_i = \sum_{j \in L} P_j$ is at least $1 - 2^{-k}$.*

From Constraint (4.3)(4.4)(4.6)[1], we know that

$$2^t \geq N + 1 \geq k + K \geq k + (\lg(n) + n + k)/t \tag{4.7}$$

Since the goal function $m = Nt \geq \lg(n) + n + k + (k-1)t$ increases as t increases, to minimize m, what remains is to find the minimum t satisfying Constraint (4.7).

Finally, we provide a probabilistic algorithm in Figure 4.1 finding a k-probe-resistant M with high probability. The correctness of this algorithm is shown in Theorem 4.13.

Theorem 4.13. *With probability at least $1 - 2^{-k}$, the algorithm presented in Figure 4.1 outputs a k-probe-resistant $M \in \{0, 1\}^{n \times m}$ such that $m \in \mathbb{N}$ and $m \leq \lg(n) + n + k + k \cdot \max(\lg(4n), \lg(4k))$.*

Proof. We want to randomly pick polynomials $P_1, P_2, \ldots, P_n \in \mathbb{F}_{2^t}[x]$ with degree at most $K - 1$ under the condition that with high probability, not only do they have to be nonzero, none of them can be a subset sum of the rest. We use the counting argument here. Starting from P_1, there are $2^{Kt} - 1$ choices, which is all but the zero polynomial. After P_1 is fixed, P_2 has $2^{Kt} - 2$ choices, which is all but zero polynomial nor P_1. After the first $P_1, \ldots, P_{i-1}$ are fixed, P_i has $2^{Kt} - 2^{i-1}$ choices, which is all but all the possible subset sum of $P_1, \ldots, P_{i-1}$. As a result, the

[1]Constraint (4.5) is in fact weaker than Constraint (4.6) and thus not needed.

number of possible choices for $(P_1, \ldots, P_n)$ is

$$F = \prod_{i=0}^{n-1} (2^{Kt} - 2^i) \geq (2^{Kt} - 2^{n-1})^n \geq (2^{Kt} - 2^n)^n = 2^{Ktn}(1 - 2^{n-Kt})^n \geq 2^{Ktn}(1 - n2^{n-Kt}).$$

For the success probability to be at least $1 - 2^{-k}$, we conclude that

$$\frac{F}{2^{Ktn}} > 1 - n2^{n-Kt} \geq 1 - 2^{-k} \quad \text{or} \quad K \geq (\lg(n) + n + k)/t.$$

□ 1

A probabilistic algorithm to generate a k-probe-resistant matrix $M \in \{0,1\}^{n\times m}$ for some $m \in \mathbb{N}$:

```
Input: EVAL's input size $n$ and a security parameter $1^k$
Output: A $k$-probe-resistant $M \in \{0,1\}^{n\times m}$ for some $m \in \mathbb{N}$
1  $t \leftarrow \lceil \max(\lg(4n), \lg(4k)) \rceil$;    // find the minimum $t$ such that $2^t \geq k + (\lg(n) + n + k)/t$
2  while $2^{t-1} > k + (\lg(n) + n + k)/(t-1)$ do
3      $t \leftarrow t - 1$;
4  end
5  $K \leftarrow \lceil (\lg(n) + n + k)/t \rceil$;
6  $N \leftarrow K + k - 1$;
7  for $i \leftarrow 1$ to $n$ do
8      Pick $P(x) = a_0 + a_1x + a_2x^2 + \cdots + a_{K-1}x^{K-1}$, where $a_i \leftarrow_R \mathbb{F}_{2^t}$;
9      $M_i \leftarrow [P(1)_2 || P(2)_2 || \ldots || P(N)_2]$; // $P(i)_2$ denotes a $t$-bit row vector
10 end
11 return $M$; // $M \in \{0,1\}^{n\times m}$, where $m = Nt$
```

Figure 4.1: A probabilistic algorithm to generate a k-probe-resistant matrix $M \in \{0,1\}^{n\times m}$ for some $m \in \mathbb{N}$

Chapter 5

Secure 2PC over Two-Output Functions

In most real-world applications, Gen also needs to receive an output from the computation, and this could be quite involved in the presence of malicious adversaries. Recall that Gen's output security needs both the privacy and correctness properties to be satisfied. The privacy property requires that Eval does not learn Gen's output, and the correctness property here requires that Eval *cannot trick* Gen *into accepting an arbitrary output*.

The privacy property is the easy part. Both parties could agree in advance to convert objective function $f(x,y) \mapsto (f_1, f_2)$ into new objective function $g((x,r),y) \mapsto (\perp, (f_1 \oplus r, f_2))$ (and the objective circuit changes accordingly). After the completion of evaluating the new objective circuit, Eval sends $f_1 \oplus r$ back to Gen for her to recover f_1 with one-time pad r that she enters as input in the first place. The one-time pad encryption's perfect privacy guarantee ensures that Eval does not learn f_1 from $f_1 \oplus r$.

However, the non-trivial part is to guarantee the correctness property, which is equivalent to asking Eval for a proof of Gen's output authenticity. Note that when in the presence of honest-but-curious adversaries, random key K_b corresponding to a circuit-output wire of bit value b suffices to be the proof. This is because if the circuit-output wire is of value b, the invariant ensures that Eval does not learn K_{1-b}, and thus Eval cannot provide a proof of output value $1-b$, except with negligible probability. Nevertheless, this idea does not automatically apply to the malicious case.

Let us consider a natural attempt and see why it does not work. Recall that in the malicious case, multiple garbled circuits are evaluated. In this attempt, Eval simply provides all random keys $\{K_b^{(i)}\}$ she learned to support output value b, where $K_b^{(i)}$ comes from the i-th garbled circuit and this circuit actually outputs b. We stress that the correctness of which the cut-and-choose technique guarantees is the majority output, not the circuit that produces the majority output. It is possible that the j-th garbled circuit $G(C)_j$ is designed to produce majority output under some condition, e.g., when the first bit of Eval's input is 0. There is a non-negligible chance that $G(C)_j$ would not get checked. Later, after the evaluation is completed, when Gen learns that $G(C)_j$ produced the majority output, she also learns the first bit of Eval's input. The same vulnerability remains even if Eval only provides the majority circuit's random key since $G(C)_j$ could be the majority circuit. In short, there is correctness guarantee over only $G(C)_j$'s output but $G(C)_j$ itself, and not its random keys, either. In other words, Eval needs to prove Gen's output authenticity without revealing which circuit the output comes from.

Gen's Output Authenticity Check via WI Proof with Garbled Values We propose a novel witness-indistinguishable proof to ensure the generator's output authenticity. The idea is novel in the sense that it requires only standard primitives, an encryption scheme and commitment scheme to be precise, and it has the overhead the same as garbling and evaluating the generator's output gates. In other words, it requires only $O(kn)$ symmetric cryptographic operations, compared with the previous state-of-the-art $O(k^2 n)$ symmetric operations [LP07, PSSW09] or $O(kn)$ symmetric operations plus $O(s) = O(k)$ number-theoretic operations [Kir08].

Our approach handles the output authenticity problem in a way that needs no number-theoretic operations at all (hence no number-theoretic intractability assumption is needed).

Our idea comes from the following three observations:

We first observe that the labels obtained from evaluating GEN's output gates are in fact sufficient for EVAL to prove the output authenticity. Recall that the Yao protocol guarantees that EVAL learns one and only one of the labels assigned with each wire. In other words, the knowledge of the labels corresponding to GEN's output is more than enough for EVAL to show the output authenticity. What remains is how EVAL shows the knowledge without disclosing the index of the majority circuit, which has been shown to be exploitable in breaching EVAL's input privacy [Kir08].

The second observation we have, as also pointed out in shelat and Shen's work [SS11], is that a witness-indistinguishable proof suffices the purposes here. Let us consider the case in which GEN (plays as the verifier) has private input $U = \{u^{(j)}\}_{j\in[s]}$ for some $s \in \mathbb{N}$, and EVAL (plays as the prover) knows $u^{(m)}$ for some $m \in [s]$. In the honest-but-curious model, a simple witness-indistinguishable proof of EVAL's knowledge $u^{(m)}$ can be done as follows:

1. GEN picks a random nonce r, and then sends $\{\mathsf{enc}_{u^{(j)}}(r)\}_{j\in[s]}$ to EVAL;
2. EVAL receives $\{c^{(j)}\}_{j\in[s]}$ and sends $\mathsf{dec}_{u^{(m)}}(c^{(m)})$ back to GEN; and
3. GEN receives r' and accepts if $r' = r$, or aborts otherwise.

Unfortunately, this proof is not witness-indistinguishable in the presence of malicious adversaries. In fact, a malicious GEN may pick distinct $\{r^{(j)}\}_{j\in[s]}$ and send $\{\mathsf{enc}_{u^{(j)}}(r^{(j)})\}_{j\in[s]}$ to EVAL so that EVAL's knowledge could be identified by locating r' in $\{r^{(j)}\}_{j\in[s]}$.

To force GEN to behave honestly in the first step, we suggest that GEN discloses $U = \{u^{(j)}\}_{j\in[s]}$ and r after receiving r' so that EVAL could check if $\{c^{(j)}\}_{j\in[s]}$ is constructed correctly. Our third observation is that this disclosure does not compromise GEN's input privacy. Indeed, EVAL should have already learned the majority of U from the circuit evaluation, and r is a random nonce that has no information about GEN's input at all. So GEN's input is not leaked through

U and r. Moreover, this disclosure does not compromise the soundness of the protocol since after GEN receives r', EVAL has already delivered her proof of authenticity so that learning U and r afterwards will not change the proof retroactively. Nonetheless, we stress that GEN should not learn r' before EVAL finishes the check, and nor should EVAL be able to change r' after the check. Naturally, a commitment scheme comes into the picture. Our idea is that EVAL commits to $\mathsf{dec}_{u^{(m)}}(c^{(m)})$ instead of giving it away in clear. After GEN reveals U and r, EVAL checks if $\{c^{(j)}\}_{j\in[s]}$ is indeed correctly generated. If the check fails, EVAL aborts; otherwise, EVAL decommits to r'. Then GEN checks if $r' = r$ and responds as in the above honest-but-curious protocol.

One more issue is that a malicious GEN could learn EVAL's private input $u^{(m)}$ with non-negligible probability by faking her private inputs from the beginning. More specifically, a malicious GEN could guess $u^{(m)}$ with probability $1/s$ and then pretend her private input to be

$$\overline{U} = (\overline{u}^{(1)}, \ldots, \overline{u}^{(m-1)}, u^{(m)}, \overline{u}^{(m+1)}, \ldots, \overline{u}^{(s)})$$

rather than U, where $\overline{u}^{(i)}$ is randomly sampled. With this attack, a malicious generator is capable of providing checkable ciphertexts when EVAL's private input is indeed $u^{(m)}$, that is, the fact that EVAL can provide the correct nonce r reveals her private input $u^{(m)}$. A straightforward way to get around this issue is for the two parties to share the commitment to GEN's private inputs in the first place. By the binding property of the commitment scheme, a malicious GEN cannot change her private inputs at will. The correctness of these commitments will be guaranteed by the cut-and-choose technique.

Protocol 5.1. WI Proofs with Labels

Common Input: a security parameter 1^k, a statistical security parameter 1^s, the commitments to GEN's private input $\{(\mathsf{com}(u_0^{(j)}), \mathsf{com}(u_1^{(j)}))\}_{j\in[s]}$, and GEN's output a.

Private Input: GEN has $\{(u_0^{(j)}, u_1^{(j)})\}_{j\in[s]}$ and EVAL has the index $m \in [s]$ of the majority circuit and the random key v corresponding to GEN's output wire of value a.

1. GEN picks a nonce $r \in \{0,1\}^k$ and sends $\{\mathsf{enc}_{u_a^{(j)}}(r)\}_{j\in[s]}$ to EVAL.
2. After receiving $\{c^{(j)}\}_{j\in[s]}$, EVAL sends $\mathsf{com}(\mathsf{dec}_v(c^{(m)}))$ back to GEN.
3. After receiving $\mathsf{com}(r')$, GEN decommits to $\{u_a^{(j)}\}_{j\in[s]}$.
4. EVAL checks the decommited values $\{u^{(j)}\}_{j\in[s]}$ that
   (a) if $\mathsf{com}(u_a^{(j)})$ can be correctly opened to $u^{(j)}$ for all $j \in [s]$?
   (b) if $\mathsf{dec}_{u^{(j)}}(c^{(j)})$ equals $\mathsf{dec}_{u^{(j+1)}}(c^{(j+1)})$ for $j = 1, 2, \dots, s-1$?

   EVAL aborts if any of the checks fails; otherwise, she decommits to r'.
5. GEN accepts the proof if $\mathsf{com}(r')$ is opened correctly and $r' = r$; otherwise, she rejects.

Lemma 5.1. *Let $\{(com(u_0^{(j)}), com(u_1^{(j)}))\}_{j\in[s]}$ be common input. Suppose* GEN *has private input $\{(u_0^{(j)}, u_1^{(j)})\}_{j\in[s]}$, where $u_b^{(j)} \in \{0,1\}^k$. Protocol 5.1 satisfies the following properties:*

1. *If* GEN *knows $u_b^{(j)}$ for some $j \in [s]$ and $b \in \{0,1\}$,* GEN *always accepts.*
2. *If* EVAL *does not know any of $u_b^{(j)}$,* GEN *rejects with probability at least $1 - 2^{-k}$.*
3. *If* EVAL *knows the majority of $\{u_b^{(j)}\}_{j\in[s]}$ for some $b \in \{0,1\}$, say she knows M, then for any $m_1, m_2 \in M$,* VIEW_{m_1} *and* VIEW_{m_2} *are computationally indistinguishable, where* VIEW_m *is*

GEN*'s view during the protocol with* EVAL*'s private input being m.*

Proof. In order to prove the lemma, we show the completeness, soundness, and witness-indistinguishability as follows:

(**Completeness**) We first clarify that by saying "(a, v) is the evaluation result from the majority circuit (the evaluation circuit of index m)" in the protocol, we mean that the majority of the evaluation circuits agree with GEN's output a, and v is indeed a correct decommitment of $\mathsf{com}(u_a^{(m)})$. Since $v = u_a^{(m)}$ for some $m \in [s]$, the completeness of the proof follows.

(**Soundness**) On one hand, due to the hiding property of com, EVAL with private input (a, v) cannot learn $u_{1-a}^{(j)}$ for some $j \in [s]$ from $\{\mathsf{com}(u_{1-a}^{(j)})\}_{j\in[s]}$. On the other hand, if the proof is accepted, the binding property of com guarantees that EVAL knows r' at the moment she receives $c^{(1)}||c^{(2)}||\ldots||c^{(s)}$ from GEN, and the semantic security of $(\mathsf{enc}, \mathsf{dec})$ ensures that EVAL does not learn r from those ciphertexts. As a result, EVAL must have known $r = r'$ to begin with. Hence, the soundness property holds.

(**Witness-Indistinguishability**) Let $m_1, m_2 \in [s]$ be the indices of two distinct majority circuits, and let (a, v_1) and (a, v_2) be their evaluation results, respectively.

First, because the checking in Step 4 is independent of EVAL's private input, the probability that EVAL aborts in Step 4 is independent of m_1 and m_2.

Next, assuming that EVAL does not abort by the end of Step 4, to prove contrapositively, suppose that transcripts $\{\mathsf{com}(\mathsf{dec}_{v_1}(c^{(m_1)})), r_1'\}$ and $\{\mathsf{com}(\mathsf{dec}_{v_2}(c^{(m_2)})), r_2'\}$ are distinguishable. Because of the hiding property of com, commitments $\{\mathsf{com}(\mathsf{dec}_{v_1}(c^{(m_1)}))\}$ and $\{\mathsf{com}(\mathsf{dec}_{v_2}(c^{(m_2)})\}$ are indistinguishable. Thus, we know that $r_1' \neq r_2'$, which contradicts to the result of the checking in Step 4 that $\mathsf{com}(u_a^{(m_1)})$ and $\mathsf{com}(u_a^{(m_2)})$ are correctly opened to $u^{(m_1)} = u_a^{(m_2)} = v_1$ and $u^{(m_2)} = u_a^{(m_2)} = v_2$, respectively, and $r_1' = \mathsf{dec}_{v_1}(c^{(m_1)}) = \mathsf{dec}_{u^{(m_1)}}(c^{(m_1)}) = \mathsf{dec}_{u^{(m_2)}}(c^{(m_2)}) = \mathsf{dec}_{v_2}(c^{(m_2)}) = r_2'$. □

Chapter 6

Secure 2PC in Practice

The goal of this chapter is to present a high performance secure 2PC system that integrates state-of-the-art techniques for dealing with *malicious* adversaries efficiently. Although some of these techniques have been reported individually, we are not aware of any attempt to incorporate them all into one system, while ensuring that a security proof can still be written for that system. Even though some of the techniques are claimed to be compatible, it is not until everything is put together and someone has gone through all the details can a system as a whole be said to be provably secure.

System Framework This system is an implementation of our Main protocol (Protocol 3.2). In particular, it uses the CCCT instantiated from 2-universal hash circuits (Protocol 4.4) and the CCOT instantiated from probe-resistant matrix circuits(Protocol 4.5). Moreover, it uses the WI proof with labels to handle two-output functions (Protocol 5.1).

State-of-the-Art Optimization Techniques For garbled circuit generation and evaluation, we incorporate Kolesnikov and Schneider's free-XOR technique that minimizes the computation and communication cost for XOR gates in a circuit [KS08b]. We also adopt Pinkas et al.'s garbled-row-reduction technique that reduces the communication cost for k-fan-in non-XOR gates by $1/2^k$ [PSSW09], which means at least 25% communication saving in our system since we only have gates of 1-fan-in or 2-fan-in. Although these techniques exist individually, this project is the first system to incorporate all of these mutually compatible state-of-the-art techniques.

Circuit-Level Parallelism The most important new technique that we plan to use is to exploit the embarrassingly parallel nature of our Main protocol for achieving security in the presence of malicious adversaries. Exploiting this, however, requires careful engineering in order to achieve good performance while maintaining security. We will parallelize all computation-intensive operations such as OTs or circuit construction by splitting the GEN-EVAL pair into hundreds of slave pairs. Each of the pairs works on an independently generated copy of the circuit in a parallel but synchronized manner as synchronization is required for our Main protocol to be secure.

Large Circuits In the Fairplay system, a garbled circuit is fully constructed before being sent over a network for the other party to evaluate [MNPS04]. This approach is particularly problematic when hundreds of copies of the garbled circuit are needed against malicious adversaries. Huang et al. pointed out that keeping the whole garbled circuit in memory is unnecessary, and that instead, the generation and evaluation of garbled gates could be conducted in a “pipelined” manner [HEKM11]. Consequently, not only do both parties spend less time idling, only a small number of garbled gates need to reside in memory at one time, even when dealing with large circuits. However, this pipelining idea does not work trivially with other optimization techniques for the following two reasons:

- The cut-and-choose technique requires the generator to finish constructing circuits before the coin flipping (which is used to determine check circuits and evaluation circuits), but the evaluator cannot start checking or evaluating before the coin flipping. A naive approach would ask the evaluator to hold the circuits and wait for the results of the coin flipping before she proceeds to do her jobs. When the circuit is of large size, keeping hundreds of copies of such a circuit in memory is undesirable.

- Similarly, the random seed checking technique [GMS08] requires the generator to send the hash for each garbled circuit, and later on send the random seeds for check circuits so that the communication for check circuits is vastly reduced. Note that the hash for an evaluation circuit is given away before the garbled circuit itself. However, a hash is calculated only after the whole circuit is generated. So the generation-evaluation pipelining cannot be applied directly.

Our system, however, will integrate this pipelining idea with the optimization techniques mentioned above, and is capable of handling circuits of billions of gates.

6.1 Techniques Regarding Security

In our system, we incorporate the malleable claw-free collection approach because of its efficiency. Although the commitment-based approaches can be implemented using lightweight primitives such as collision-resistant hash functions, they incur high communication overhead for the extra complexity factor s, that is, the number of copies of the circuit. On the other hand, the group-based approach could be more computationally intensive, but this discrepancy is compensated again due to the parameter s.[1] Hence, with similar computation cost, group-based approaches enjoy lower communication overhead.

[1]To give concrete numbers, with an Intel Core i5 processor and 4GB DDR3 memory, a SHA-256 operation (from OpenSSL) requires $1,746$ cycles, while a group operation (160-bit elliptic curve from the PBC library with preprocessing) needs $322,332$ cycles. It is worth-mentioning that s is at least 256 in order to achieve security level 2^{-80}. The gap between a symmetric operation and an asymmetric one becomes even smaller when modern libraries such as RELIC are used instead of PBC.

Our system is based on the random input replacement approach due to its scalability. It is a fact that the committing OT or the cut-and-choose OT does not alter the circuit while the random input replacement approach inflates the circuit by $O(sn)$ additional gates. However, it has been shown that $\max(4n, 8s)$ additional gates suffice [PSSW09]. Moreover, both the committing OT and the cut-and-choose OT require $O(ns)$ group operations, while the random input replacement approach needs only $O(s)$ group operations. Furthermore, we observe that the random input replacement approach is in fact compatible with the OT extension technique. Therefore, we were able to build our system which has the group operation complexity independent of the evaluator's input size, and as a result, our system is particularly attractive when handling a circuit with a large evaluator input.

Lindell and Pinkas' approach, albeit straightforward, might introduce greater communication overhead than the description function itself. We therefore employ the approach that takes the advantages of the remaining two solutions. In particular, we implement Kiraz's approach (smaller proof size), but only a witness-indistinguishable proof is performed (weaker security property).

6.2 Techniques Regarding Performance

Yao's garbled circuit technique has been studied for decades. It has drawn significant attention for its simplicity, constant round complexity, and computational efficiency (since circuit evaluation only requires fast symmetric operations). The fact that it incurs high communication overhead has provoked interest that has led to the development of fruitful results.

In this section, we willdiscuss the optimization techniques that greatly reduce the communication cost while maintaining the security. These techniques include free-XOR, garbled row reduction, random seed checking, and large circuit pre-processing. In addition to these original ideas, practical concerns involving large circuits and parallelization will be addressed.

6.2.1 Free-XOR

Kolesnikov and Schneider [KS08b] proposed the free-XOR technique that aims for removing the communication cost and decreasing the computation cost for XOR gates.

The idea is that the generator first randomly picks a global key R, where $R = r||1$ and $r \in \{0,1\}^{k-1}$. This global key has to be hidden from the evaluator. Then for each wire w, instead of picking both W_0 and W_1 at random, only one is randomly chosen from $\{0,1\}^k$, and the other is determined by $W_b = W_{1\oplus b} \oplus R$. Note that π_w remains the rightmost bit of W_0. For an XOR gate having input wire x with $(X_0, X_0 \oplus R, \pi_x)$, input wire y with $(Y_0, Y_0 \oplus R, \pi_y)$, and output wire z, the generator lets $Z_0 = X_0 \oplus Y_0$ and $Z_1 = Z_0 \oplus R$. Observe that

$$X_0 \oplus Y_1 = X_1 \oplus Y_0 = X_0 \oplus Y_0 \oplus R = Z_0 \oplus R = Z_1$$
$$X_1 \oplus Y_1 = X_0 \oplus R \oplus Y_0 \oplus R = X_0 \oplus Y_0 = Z_0.$$

This means that while evaluating an XOR gate, XORing the labels for the two input wires will directly retrieve the label for the output wire. So no garbled truth table is needed, and the cost of evaluating an XOR gate is reduced from a decryption operation to a bitwise XOR.

This technique is only secure when the encryption scheme satisfies certain security properties. The solution provided by the authors is

$$\mathsf{enc}_X(\mathsf{enc}_Y(Z)) = H(X||Y) \oplus Z,$$

where $H : \{0,1\}^{2k} \mapsto \{0,1\}^k$ is a random oracle. Recently, Choi et al. [CKKZ12] have further shown that it is sufficient to instantiate $H(\cdot)$ with a weaker cryptographic primitive, *2-circular correlation robust functions*. Our system instantiates this primitive with

$H(X||Y) = \text{SHA-256}(X||Y)$. However, when AES-NI instructions are available, our system instantiates it with $H^k(X||Y) = \text{AES-256}(X||Y, k)$, where k is the gate index.

6.2.2 Garbled Row Reduction

The GRR (Garbled Row Reduction) technique suggested by Pinkas et al. [PSSW09] is used to reduce the communication overhead for non-XOR gates. In particular, it reduces the size of the garbled truth table for 2-fan-in gates by 25%.

Recall that in the baseline Yao's garbled circuit, both the 0-key and 1-key for each wire are randomly chosen. After the free-XOR technique is integrated, the 0-key and 1-key for an XOR gate's output wire depend on input key and R, but the 0-key for a non-XOR gate's output wire is still free. The GRR technique is to make a smart choice for this degree of freedom, and thus, reduce one entry in the garbled truth table to be communicated over network.

In particular, the generator picks (Z_0, Z_1, π_z) by letting $Z_{g(0\oplus\pi_x, 0\oplus\pi_y)} = H(X_{0\oplus\pi_x}||Y_{0\oplus\pi_y})$, that is, either Z_0 or Z_1 is assigned to the encryption mask for the 0-th entry of the GTT_g, and the other one is computed by the equation $Z_b = Z_{1\oplus b} \oplus R$. Therefore, when the evaluator gets $(X_{0\oplus\pi_x}, Y_{0\oplus\pi_y})$, both $X_{0\oplus\pi_x}$ and $Y_{0\oplus\pi_y}$ have rightmost bit 0, indicating that the 0-th entry needs to be decrypted. However, with GRR technique, she is able to retrieve $Z_{g(0\oplus\pi_x, 0\oplus\pi_y)}$ by running $H(\cdot)$ without inquiring GTT_g.

Pinkas et al. claimed that this technique is compatible with the free-XOR technique [PSSW09]. For rigorousness purposes, we carefully went through the details and came up with a security proof for our protocol that confirms this compatibility. The proof will be included in the full version of this paper.

6.2.3 Random Seed Checking

Recall that the cut-and-choose approach requires the generator to construct multiple copies of the garbled circuit, and more than half of these garbled circuits will be fully revealed, including the randomness used to construct the circuit. Goyal, Mohassel, and Smith [GMS08] therefore pointed out an insight that the evaluator could examine the correctness of those check circuits by receiving a hash of the garbled circuit first, acquiring the random seed, and reconstructing the circuit and hash by herself.

This technique results in the communication overhead for check circuits independent of the circuit size. This technique has two phases that straddle the coin-flipping protocol. Before the coin flipping, the generator constructs multiple copies of the circuit as instructed by the cut-and-choose procedure. Then the generator sends to the evaluator the hash of each garbled circuit, rather than the circuit itself. After the coin flipping, when the evaluation circuits and the check circuits are determined, the generator sends to the evaluator the full description of the evaluation circuits and the random seed for the check circuits. The evaluator then computes the evaluation circuits and tests the check circuits by reconstructing the circuit and comparing its hash with the one received earlier. As a result, even for large circuits, the communication cost for each check circuit is simply a hash value plus the random seed. Our system provides that 60% of the garbled circuits are check circuits. Thus, this optimization significantly reduces communication overhead.

6.2.4 Interpolated Communication

In this section, we provide an effective method reducing the communication cost of our protocol by up to 60%. This is a simple trick yet it provides a big benefit.

Our main protocol is in fact compatible with the pipeline technique that vastly increases the scalability of secure two-party computation. In particular, the garbled circuits can be sent in batches of gates. Each batch consists of σ garbled gates corresponding to the same gate of the objective circuit. While the evaluator is still working on evaluating the current batch, the generator could start generating the next.

The observation is that the evaluator has already got the randomness for check circuits. It is inefficient for each gate in a check circuit to be sent over network while both parties are capable of generating it. However, the difficulty is that the generator does not get to know which is check circuit and which is evaluation circuit. She has to treat each gate in the same batch equally. The problem can be formulated as follows:

> the generator wants to send the evaluator σ numbers $\{g_0, g_1, \ldots, g_{\sigma-1}\}$, and in fact, the evaluator already knows (or is capable of generating) μ of them. How do they reduce the communication overhead while the generator does not learn which numbers the evaluator knows (or is capable of generating)?

An interesting trick is that they could treat $\{g_i\}$ as coefficients of a polynomial, that is,

$$P(x) = g_0 + g_1 x + \cdots g_{\sigma-1} x^{\sigma-1}.$$

Although the generator does not know which μ numbers out of $\{g_i\}$ that the evaluator knows, she does know that the evaluator only needs $(\sigma - \mu)$ points to fully recover the polynomial. the generator therefore sends Eval

$$P(1), P(2), \ldots, P(\sigma - \mu).$$

Clearly, there are σ unknowns in $P(x)$. With the μ equations the evaluator already knows plus the $(\sigma - \mu)$ new equations from Gen, she can easily recover all $\{g_i\}$ herself with simple polynomial interpolation.

Since our protocol suggests 60%-40% check circuit and evaluation circuit ratio, with a slight increase of the computation overhead, this trick helps to reduce the communication cost by 60%.

6.2.5 Large Circuits

A circuit for a reasonably complicated function can easily consist of billions of gates. For example, a 4095-bit edit distance circuit has 5.9 billion gates. When circuits grow to such a size, the task of achieving high performance secure computation becomes challenging.

Working Set Optimization Another problem encountered while dealing with large circuits is the *working set minimization problem*. Note that the *circuit value problem* is log-space complete for P. It is suspected that L≠P, that is, there exist some circuits that can be evaluated in polynomial time but require more than logarithmic space. This open problem captures the difficulty of handling large circuits during both the construction and evaluation, where at any moment there is a set of wires, called the *working set*, that are available and will be referenced in the future. For some circuits, the working set is inherently super-logarithmic. A naive approach is to keep the most recent D wires in the working set, where D is the upper bound of the input-output distance of all gates. However, there may be wires which are used as inputs to gates throughout the entire circuit, and so this technique could easily result in adding almost the whole circuit to the working set, which is especially problematic when there are hundreds of copies of a circuit of billions of gates. While reordering the circuit or adding identity gates to minimize D would mitigate this problem, doing so while maintaining the topological order of the circuit is known to be an NP-complete problem, the *graph bandwidth problem* [GGJK78].

Our solution to this difficulty is to pre-process the circuit so that each gate comes with a usage count. Our system has a compiler that converts a program in high-level language into a boolean circuit. Since the compiler is already using global optimization in order to reduce the circuit size, it is easy for the global optimizer to analyze the circuit and calculate the usage count for each gate. With this information, it is easy for the generator and evaluator to decrement the counter for each gate whenever it is being referenced and to toss away the gate whenever its counter becomes zero. In other words, we keep track of merely useful information and heuristically minimize the size of the working set, which is small compared with the original circuit size as shown in Table 6.1.

	AES	Dot_4^{64}	RSA-32	EDT-255
circuit size	49,912	460,018	1,750,787	15,540,196
wrk set size	323	711	235	2,829

Table 6.1: The size of the working set for various circuits (sizes include input gates)

6.3 Experimental Results

In this section, we give a detailed description of our system, upon which we have implemented various real world secure computation applications. The experimental environment is the Ranger cluster in the Texas Advanced Computing Center. Ranger is a blade-based system, where each node is a SunBlade x6240 blade running a Linux kernel and has four AMD Opteron quad-core 64-bit processors, as an SMP unit. Each node in the Ranger system has 2.3 GHz core frequency and 32 GB of memory, and the point-to-point bandwidth is 1 GB/sec. Although Ranger is a high-end machine, we use only a small fraction of its power for our system, only 512 out of 62,976 cores. Note that we use the PBC (Pairing-Based Cryptography) library [Lyn06] to implement the underlying cryptographic protocols such as oblivious transfers, witness-indistinguishable proofs, and so forth. However, moving to more modern libraries such as

RELIC [Rel] is likely to give even better results, especially to those circuits with large input and output size.

System Setup In our system, both the generator and the evaluator run an equal number of processes, including a root process and many slave processes. A root process is responsible for coordinating its own slave processes and the other root process, while the slave processes work together on repeated and independent tasks. There are three pieces of code in our system: the generator, the evaluator, and the IP exchanger. Both the generator's and evaluator's program are implemented with Message Passing Interface (MPI) library. The reason for the IP exchanger is that it is common to run jobs on a cluster with dynamic working node assignment. However, when the nodes are dynamically assigned, the generator running on one cluster and the evaluator running on another might have a hard time locating each other. Therefore, a fixed location IP exchanger helps the match-up process as described in Figure 6.1. Our system provides two modes—the user mode and the simulation mode. The former works as mentioned above, and the latter simply spawns an even number of processes, half for the generator and the other half for the evaluator. The network match-up process is omitted in the latter mode to simplify the testing of this system.

To achieve a security level of 2^{-80}, meaning that a malicious player cannot successfully cheat with probability better than 2^{-80}, requires at least 250 copies of the garbled circuit [SS11]. For simplicity, we used 256 copies in our experiments, that is, security parameters $k = 80$ and $s = 256$. Each experiment was run 30 times (unless stated otherwise), and in the following sections we report the average runtime of our experiments.

Timing methodology When there is more than one process on each side, care must be taken in measuring the timings of the system. The timings reported in this section are the time required by the root process at each stage of the system. This was chosen because the root

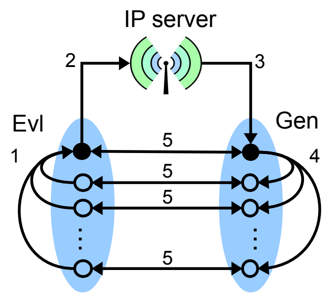


Figure 6.1: Both the generator and evaluator consist of a root process (solid circles) and a number of slave processes (hollow circles). The match-up works as follows: the slave evaluator processes send their IPs to the root evaluator process (Step 1), who then forwards them to the IP exchanger (Step 2). Next, the root generator process comes to acquire these IPs (Step 3) and dispatch them to its slaves (Step 4), who then proceed to pair up with one of the slave evaluator processes (Step 5) and start the main protocol. The arrows show the message flow.

process will always be the *longest* running process, as it must wait for each slave process to run to completion. Moreover, in addition to doing all the work that the slaves do, the root processes also perform the input consistency check and the coin tossing protocol.

Impacts of the Performance Optimization Techniques We have presented several performance optimization techniques in Section 6.2 with theoretical analyses, and here we demonstrate their empirical effectiveness in Table 6.2. As we have anticipated, the Random Seed Checking reduces the communication cost for the garbled circuits by 60%, and the Garbled Row Reduction further reduces by another 25%. In the RS and GRR columns, the small deviation from the theoretical fraction 40% and 30%, respectively, is due to certain implementation needs. Our compiler is designed to reduce the number of non-XOR gates. In these four circuits, the ratio of non-XOR gates is less than 43%. So after further applying the Free-XOR technique, the final communication is less than 13% of that in the baseline approach.

Performance Gain by AES-NI On a machine with 2.53 GHz Intel Core i5 processor and 4GB 1067 MHz DDR3 memory, it takes 784 clock cycles to run a single SHA-256 (with OpenSSL 1.0.0g), while it needs only 225 cycles for AES-256 (with AES-NI). To measure the benefits of

	non-XOR (%)	Baseline (MB)	RS (%)	GRR (%)	FX (%)
AES	30.81	509	39.97	30.03	9.09
Dot_4^{64}	29.55	4,707	39.86	29.91	8.88
RSA-32	34.44	17,928	39.84	29.88	10.29
EDT-255	41.36	159,129	39.84	29.87	12.36

Table 6.2: The impact of various optimization techniques: The Baseline shows the communication cost for 256 copies of the original Yao garbled circuit when $k = 80$; RS shows the remaining fraction after Random Seed technique is applied; GRR shows when Garbled Row Reduction is further applied; and FX shows when the previous two techniques and the Free-XOR are applied. (The communication costs here only include those in the generation and evaluation stages.)

AES-NI, we use two instantiations to construct various circuits, listed in Table 6.3, and observe a consistent 20% saving in circuit construction.[2]

	size (gate)	AES-NI (sec)	SHA-256 (sec)	Ratio (%)
AES	49,912	0.12± 1%	0.15± 1%	78.04
Dot_4^{64}	460,018	1.11±0.4%	1.41±0.5%	78.58
RSA-32	1,750,787	4.53±0.5%	5.9±0.8%	76.78
EDT-255	15,540,196	42.0±0.5%	57.6± 1%	72.92

Table 6.3: Circuit generation time (for a single copy) with different instantiations (AES-NI vs SHA-256) of the 2-circular correlation robust function.

6.3.1 Secure 2PC 128-bit AES

We used AES as a benchmark to compare our compiler to the Fairplay compiler, and as a test circuit for our system. We tested the full AES circuit, as specified in FIPS-197 [FIP01]. In the semi-honest model, it is possible to reduce the number of gates in an AES circuit by computing the key schedule offline; e.g. this is one of the optimizations employed by Huang et al. [HEKM11]. In the malicious model, however, such an optimization is not possible; the party holding the key could attempt to reduce the security level of the cipher by computing a

[2]The reason that saving 500+ cycles does not lead to more improvements is that this encryption operation is merely one of the contributing factors to generating a garbled gate. Other factors, for example, include GNU `hash_map` table insertion (∼1,200 cycles) and erase (∼600 cycles).

malicious key schedule. So in our experiments we compute the entire function, including the key schedule, online.

In this experiment, two parties collaboratively compute the function $(x, y) \mapsto (\perp, \mathrm{AES}_x(y))$, i.e., the circuit generator holds the encryption key x, while the evaluator has the message y to be encrypted. At the end, the generator will not receive any output, whereas the evaluator will receive the ciphertext $\mathrm{AES}_x(y)$.

In Table 6.4, both the computational and communication costs for each main stage are listed under the traditional setting, where there is only one process on each side. These main stages include oblivious transfer, garbled circuit construction, the generator's input consistency check, and the circuit evaluation. Each row includes both the computation and communication time used. Note that network conditions could vary from setting to setting. Our experiments run in a local area network, and the data can only give a rough idea on how fast the system could be in an ideal environment. However, the precise amount of data being exchanged is reported.

		Gen (sec)	Eval (sec)	Comm (KB)
OT	comp	45.8±0.09%	34.0±0.2%	5,516
	comm	0.1± 1%	11.9±0.6%	
Gen.	comp	35.6± 0.5%	–	3
	comm	–	35.6±0.5%	
Inp. Chk	comp	–	1.75±0.2%	266
	comm	–	–	
Evl.	comp	14.9± 0.6%	32.4±0.4%	28,781
	comm	18.2± 1%	3.2±0.8%	
Total	comp	96.3± 0.3%	68.0±0.2%	34,566
	comm	18.3± 1%	50.8±0.4%	

Table 6.4: The 95% two-sided confidence intervals of the computation and communication time for each stage in the experiment $(x, y) \mapsto (\perp, \mathrm{AES}_x(y))$.

node #	4		16	
	Gen	Evl	Gen	Evl
OT	12.56±0.1%	8.41±0.1%	4.06±0.1%	2.13±0.2%
Gen.	8.18±0.4%	–	1.92±0.7%	–
Inp. Chk	–	0.42± 4%	–	0.10± 10%
Evl.	3.3± 4%	7.08± 1%	0.80± 10%	1.58± 4%
Inter-com	4± 5%	13.2±0.3%	0.93± 10%	4.08±0.8%
Intra-com	0.17± 30%	0.23± 20%	0.18± 8%	0.25± 6%
Total time	28.3±0.3%	29.4±0.3%	7.90±0.5%	8.17±0.4%

(a)

node #	64		256	
	Gen	Evl	Gen	Evl
OT	1.96±0.1%	0.58±0.2%	0.64±0.1%	0.19±0.2%
Gen.	0.49±0.4%	–	0.14± 1%	–
Inp. Chk	–	–	–	–
Evl.	0.23± 17%	0.37± 7%	0.12±0.5%	0.05±0.6%
Inter-com	0.31± 20%	1.98± 1%	0.11± 40%	0.72±0.2%
Intra-com	0.45± 20%	0.48± 15%	0.34± 30%	0.34± 30%
Total time	3.45± 2%	3.44± 2%	1.4± 10%	1.3± 9%

(b)

Table 6.5: The average and error interval of the times (seconds) running AES circuit. The number of nodes represents the degree of parallelism on each side. "–" means that the time is smaller than 0.05 seconds. Inter-com refers to the communication between the two parties, and intra-com refers to communication between nodes for a single party.

We notice in Table 6.4 that the evaluator spends an unreasonable amount of time on communication with respect to the amount of data to be transmitted in both the oblivious transfer and circuit construction stages. This is because the evaluator spends that time waiting for the generator to finish computation-intensive tasks. The same reasoning explains why in the circuit evaluation stage the generator spends more time in communication than the evaluator. This waiting results from the fact that both parties need to run the protocol in a synchronized manner. A generator-evaluator pair cannot start next communication round while any other pair has not finished the current one. This synchronization is crucial since our protocol's security is guaranteed only when each communication round is performed sequentially. While the parallelization of the program introduces high performance execution, it does not and should not change this essential property. A stronger notion of security such as universal security will be required if asynchronous communication is allowed. By using TCP sockets in "blocking" mode, we enforce this communication round synchronization.

Note that the low communication during the circuit construction stage is due to the random seed checking technique. Also, the fact that the generator spends more time in the evaluation stage than she traditionally does comes from the second construction for evaluation circuits. Recall that only the evaluation circuits need to be sent to the evaluator. Since only 40% of the garbled circuits (102 out of 256) are evaluation-circuits, the ratio of the generator's computation time in the generation and evaluation stage is 35.63:14:92 $\simeq$ 5:2.

We were unfortunately unable to find a cluster of hundreds of nodes that all support AES-NI. Our experimental results, therefore, do not show the full potential of all the optimization techniques we have proposed. However, recall that for certain circuits the running time in the semi-honest setting is roughly half of that in the malicious setting. We estimate a 20% improvement in the performance of garbled circuit generation when the AES-NI instruction set becomes ubiquitous, based on the preliminary results presented above in Table 6.3.

Table 6.5 shows that the Yao protocol really benefits from the circuit-level parallelization. Starting from Table 6.4, where each side only has one process, all the way to when each side has 256 processes, as the degree of parallelism is multiplied by four, the total time reduces into a quarter. Note that the communication costs between the generator and evaluator remain the same, as shown in Table 6.4. It may seem odd that the communication costs are *reduced* as the number of processes increase. The real interpretation of this data is that as the number of processes increases, the "waiting time" decreases.

Notice that as the number of processes increases, the ratio of the time the generator spends in the construction and evaluation stage decreases from 5:2 to 1:1. The reason is that the number of garbled circuit each process handles is getting smaller and smaller. Eventually, we reach the limit of the benefits that the circuit-level parallelism could possibly bring. In this case, each process is dealing with merely a single copy of the garbled circuit, and the time spent in both the generation and evaluation stages is the time to construct a garbled circuit.

To the best of our knowledge, completing an execution of secure AES in the malicious model within 1.4 seconds is the best result that has ever been reported. The next best result from Nielsen et al. [NNOB12] is 1.6 seconds, and it is an amortized result (85 seconds for 54 blocks of AES encryption in parallel) in the random oracle model. This is only a crude comparison, however; our experimental setup uses a cluster computer while Nielsen et al. used only two desktops. A better comparison would be possible given a parallel implementation of Nielsen et al.'s system, and we are interested in seeing how much of an improvement such an implementation could achieve.

6.3.2 Secure 2PC 256-bit RSA

In this experiment, we run the 256-bit RSA encryption circuit, that is, $(x, y) \mapsto (\perp, x^y \mod m)$, where $x, y, m \in \{0, 1\}^{256}$ and m is a public odd number. The circuit generated by our compiler has 934 million gates, and 332 million of which are non-XOR. Roughly, our system can handle 38,000 (non-XOR) gates per second. The results are shown in Table 6.6.

6.3.3 Secure 2PC 4095-bit Edit-Distance

In this experiment, we run the 4095-bit edit distance circuit $(x, y) \mapsto (\perp, \mathrm{EDT}(x, y))$, where $x, y \in \{0, 1\}^{4095}$. Both the parties run the circuit generation and evaluation in a pipeline manner, where one party is generating and giving away garbled gates on one end, and the other party is

	Gen (sec)	Eval (sec)	Comm (MB)
OT	$1.43 \pm 0.0\%$ $0.07 \pm 10\%$	$0.47 \pm 2.9\%$ $1.02 \pm 1.8\%$	11
Gen.	$742.39 \pm 2.0\%$ –	$48.68 \pm 8.1\%$ $693.71 \pm 2.6\%$	<1
cut & chk	$44.44 \pm 8.7\%$ –	$44.44 \pm 8.7\%$ –	<1
Evl.	$723.48 \pm 0.9\%$ $230.17 \pm 1.5\%$	$348.40 \pm 0.6\%$ $605.41 \pm 1.5\%$	204,355
Total	$1742.05 \pm 1.0\%$	$1742.24 \pm 1.0\%$	204,368

Table 6.6: The result of $(x, y) \mapsto (\perp, x^y \bmod m)$, where $x, y, m \in \{0,1\}^{256}$ and m is public parameter. Each party is comprised of 256 cores in a cluster.

evaluating and checking the received gates at the other end at the same time. The results are shown in Table 6.7.

This circuit generated by our compiler has 5.9 billion gates, and 2.4 billion of those are non-XOR. It is worth mentioning that, without the random-seed technique, the communication cost shown in Table 6.7 can also be estimated by $256 \times 2.4 \times 10^9 \times 3 \times 10 = 1.8 \times 10^{13}$, since 256 copies of the garbled circuits need to be transferred, each copy has 2.4 billion non-free gates, each non-free gate has three entries, and each entry has $k = 80$ bits.

In additional to showing that our system is capable of handling the largest circuits ever reported, we also have shown a speed in the malicious setting that is comparable to those in the semi-honest setting. In particular, we were able to complete an single execution of 4095-bit edit distance circuit in less than 8.2 hours with a rate of 82,000 (non-XOR) gates per second. Note that Huang et al.'s system is the only one, to the best of our knowledge, that is capable of handling such large circuits [HEKM11]; they reported a rate of over 96,000 (non-XOR) gates per second for an edit-distance circuit in the semi-honest setting.

	Gen (sec)	Eval (sec)	Comm (Byte)
OT	19.73±0.5% 1.1± 6%	5.26±0.4% 15.6±0.6%	1.7×10^8
Cut-& Choose	1.1±0.8% –	– 1.5± 2%	6.5×10^7
Gen./Evl.	24,400± 1% 4,900± 1%	14,600± 3% 14,700± 2%	1.8×10^{13}
Inp. Chk	0.6± 20% 0.4± 40%	– 0.60± 20%	8.5×10^6
Total	24,400± 1% 4,900± 1%	14,600± 3% 14,700± 2%	1.8×10^{13}

Table 6.7: The result of $(x, y) \mapsto (\perp, \text{EDT-4095}(x, y))$. Each party is comprised of 256 cores in a cluster. This table comes from 6 invocations of the system. Similarly, the upper row in each stage is the computation time, while the lower is the communication time.

Bibliography

[BM89] Mihir Bellare and Silvio Micali. Non-Interactive Oblivious Transfer and Applications. In *Proceedings on Advances in cryptology*, CRYPTO '89, pages 547–557, New York, NY, USA, 1989. Springer-Verlag New York, Inc.

[CDM00] Ronald Cramer, Ivan Damgård, and Philip MacKenzie. Efficient Zero-Knowledge Proofs of Knowledge without Intractability Assumptions. In Hideki Imai and Yuliang Zheng, editors, *Public Key Cryptography*, volume 1751 of *Lecture Notes in Computer Science*, pages 354–372. Springer Berlin Heidelberg, 2000.

[CKKZ12] Seung Geol Choi, Jonathan Katz, Ranjit Kumaresan, and Hong-Sheng Zhou. On the Security of the "Free-XOR" Technique. In *Proceedings of the 9th international conference on Theory of Cryptography*, TCC'12, pages 39–53, Berlin, Heidelberg, 2012. Springer-Verlag.

[DH76] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Trans. Inf. Theor.*, 22(6):644–654, November 1976.

[FIP01] FIPS. *Advanced Encryption Standard (AES)*, 2001.

[GGJK78] M. Garey, R. Graham, D. Johnson, and D. Knuth. Complexity Results for Bandwidth Minimization. *SIAM Journal on Applied Mathematics*, 34(3):477–495, 1978.

[GK96] Oded Goldreich and Ariel Kahan. How to Construct Constant-Round Zero-Knowledge Proof Systems for NP. *Journal of Cryptology*, 1996.

[GMS08] Vipul Goyal, Payman Mohassel, and Adam Smith. Efficient two party and multi party computation against covert adversaries. In *Proceedings of the theory and applications of cryptographic techniques 27th annual international conference on Advances in cryptology*, EUROCRYPT'08, pages 289–306, Berlin, Heidelberg, 2008. Springer-Verlag.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, STOC '87, pages 218–229, New York, NY, USA, 1987. ACM.

[Gol04] Oded Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, 2004.

[HEKM11] Yan Huang, David Evans, Jonathan Katz, and Lior Malka. Faster secure two-party computation using garbled circuits. In *Proceedings of the 20th USENIX conference on Security*, SEC'11, pages 35–35, Berkeley, CA, USA, 2011. USENIX Association.

[Hof11] Dennis Hofheinz. Possibility and impossibility results for selective decommitments. *J. Cryptol.*, 24(3):470–516, July 2011.

[IKNP03] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending Oblivious Transfers Efficiently. In *CRYPTO'03*, volume 2729 of *LNCS*, pages 145–161. Springer Berlin / Heidelberg, 2003.

[IZ89] R. Impagliazzo and D. Zuckerman. How to recycle random bits. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, SFCS '89, pages 248–253. IEEE Computer Society, 1989.

[JS07] Stanisław Jarecki and Vitaly Shmatikov. Efficient two-party secure computation on committed inputs. In *Proceedings of the 26th annual international conference on Advances in Cryptology*, EUROCRYPT '07, pages 97–114, Berlin, Heidelberg, 2007. Springer-Verlag.

[Kir08] Mehmet Kiraz. *Secure and Fair Two-Party Computation*. PhD thesis, Technische Universiteit Eindhoven, 2008.

[KS06] Mehmet Kiraz and Berry Schoenmakers. A Protocol Issue for The Malicious Case of Yao's Garbled Circuit Construction. In *27th Symposium on Information Theory in the Benelux*, 2006.

[KS08a] Mehmet Kiraz and Berry Schoenmakers. An Efficient Protocol for Fair Secure Two-Party Computation. In Tal Malkin, editor, *CT-RSA 2008*, volume 4964 of *LNCS*, pages 88–105. Springer, 2008.

[KS08b] Vladimir Kolesnikov and Thomas Schneider. Improved garbled circuit: Free xor gates and applications. In *Proceedings of the 35th international colloquium on Automata, Languages and Programming, Part II*, ICALP '08, pages 486–498, Berlin, Heidelberg, 2008. Springer-Verlag.

[KShS12] Benjamin Kreuter, Abhi Shelat, and Chih hao Shen. Billion-Gate Secure Computation with Malicious Adversaries. In *Proceedings of the 21th USENIX conference on Security*, pages 285–35. USENIX Association, 2012.

[LP07] Yehuda Lindell and Benny Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. In *Proceedings of the 26th annual international conference on Advances in Cryptology*, EUROCRYPT '07, pages 52–78, Berlin, Heidelberg, 2007. Springer-Verlag.

[LP09] Yehuda Lindell and Benny Pinkas. A Proof of Security of Yao's Protocol for Two-Party Computation. *Journal of Cryptology*, 22(2):161–188, 2009.

[LP11] Yehuda Lindell and Benny Pinkas. Secure two-party computation via cut-and-choose oblivious transfer. In *Proceedings of the 8th conference on Theory of cryptography*, TCC'11, pages 329–346, Berlin, Heidelberg, 2011. Springer-Verlag.

[LPS08] Yehuda Lindell, Benny Pinkas, and Nigel Smart. Implementing Two-Party Computation Efficiently with Security Against Malicious Adversaries. In Rafail Ostrovsky, Roberto De Prisco, and Ivan Visconti, editors, *SCN 2008*, volume 5229 of *LNCS*, pages 2–20. Springer, 2008.

[Lyn06] Ben Lynn. Pairing-Based Cryptography Library, 2006. http://crypto.stanford.edu/pbc/.

[MF06] Payman Mohassel and Matthew Franklin. Efficiency Tradeoffs for Malicious Two-Party Computation. In *Proceedings of the 9th international conference on Theory and Practice of Public-Key Cryptography*, PKC'06, pages 458–473, Berlin, Heidelberg, 2006. Springer-Verlag.

[MNPS04] Dahlia Malkhi, Noam Nisan, Benny Pinkas, and Yaron Sella. Fairplay: A Secure Two-Party Computation System. In *Proceedings of the 13th conference on USENIX Security Symposium - Volume 13*, SSYM'04, pages 20–20, Berkeley, CA, USA, 2004. USENIX Association.

[NNOB12] Jesper Buus Nielsen, Peter Sebastian Nordholt, Claudio Orlandi, and Sai Sheshank Burra. A New Approach to Practical Active-Secure Two-Party Computation. In *Proceedings of the 32th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '12, 2012.

[NP01] Moni Naor and Benny Pinkas. Efficient Oblivious Transfer Protocols. In *12th Annual ACM-SIAM Symposium on Discrete Algorithm*, pages 448–457. Society for Industrial and Applied Mathematics, 2001.

[PSSW09] Benny Pinkas, Thomas Schneider, Nigel P. Smart, and Stephen C. Williams. Secure two-party computation is practical. In *Proceedings of the 15th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*, ASIACRYPT '09, pages 250–267, Berlin, Heidelberg, 2009. Springer-Verlag.

[Rel] Relic. http://code.google.com/p/relic-toolkit/.

[SS11] Abhi Shelat and Chih-Hao Shen. Two-output secure computation with malicious adversaries. In *Proceedings of the 30th Annual international conference on Theory and applications of cryptographic techniques: advances in cryptology*, EUROCRYPT'11, pages 386–405, Berlin, Heidelberg, 2011. Springer-Verlag.

[Yao82] Andrew C. Yao. Protocols for secure computations. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, SFCS '82, pages 160–164, Washington, DC, USA, 1982. IEEE Computer Society.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets. In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*, SFCS '86, pages 162–167, Washington, DC, USA, 1986. IEEE Computer Society.