Fast Two-Party Secure Computation with Minimal Assumptions

abhi shelat and Chih-hao Shen

University of Virginia
Charlottesville, VA 22903

Abstract. All recent implementations of two-party secure computation protocols require specific complexity assumptions for their correctness and/or efficiency (e.g., DDH, homomorphic encryption, Sigma protocols for specific languages). We propose and implement a Yao-based protocol for secure two-party computation against malicious adversaries that enjoys the following benefits:

1. it assumes the minimal hardness assumption, that is, oblivious transfers;
2. it has constant round complexity;
3. its overhead is linear times (in terms of security parameter) of the Yao protocol's, which is the best one could hope for by using the circuit-level cut-and-choose technique to achieve malicious security; and
4. it is embarrassingly parallelizable in that its depth complexity is roughly the same as the Yao protocol.

To achieve these properties, we use the cut-and-choose paradigm, but solve the main three problems for achieving malicious security (input consistency, selective failure, and output authentication) in a novel and efficient manner. In particular, we propose an efficient witness-indistinguishable proof for output authentication; we suggest the use of an auxiliary 2-universal circuit to ensure the generator's input consistency; and we advance the performance of the state-of-the-art approach defending the selective failure attack.

Not only does our protocol require weaker complexity assumptions, but our implementation of this protocol also demonstrates a several factor improvement over the best prior two-party secure computation protocol which rely on specific number theoretic assumptions.

keywords: the Yao protocol, malicious model, circuit-level cut-and-choose, depth complexity

1 Introduction

Secure two-party computation is a well studied problem. A solution consists of a protocol that satisfies two security properties—*correctness* and *privacy*. 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, each participant cannot learn more than that derivable from her own input and output. An easy way to achieve these two properties is to use a trusted third party to do the computation. Both parties hand their private inputs to this third party and later retrieve the computation result. However, the solution that cryptographers desire needs to achieve the effect of a trusted party without one.

The first generic solution for secure two-party computation against honest-but-curious adversaries was proposed by Yao [Yao82]. In this protocol, both parties agree on an objective function and its boolean circuit format (called the *objective circuit*) in advance[1]. One party (called the *generator* and denoted by GEN) starts with generating a garbled version of the objective circuit. The other party (called the *evaluator* and denoted by EVAL) then obliviously evaluates this garbled circuit and gets the

[1] The equivalence between the objective function and the objective circuit is out of the scope of this paper.

output. By oblivious evaluation we mean that the evaluator does not get to learn any intermediate value of the computation. This protocol satisfies the two security properties if both participants follow the protocol instructions honestly.

To achieve the security in the malicious model, the standard approach is to force both parties to behave honestly in a protocol secure in the honest-but-curious model. Note that a solution secure in the malicious model ensures that either a cheater is caught or both parties get their outputs with two security properties satisfied. The Yao protocol becomes vulnerable in the malicious model. One of the biggest issues is that the generator could construct faulty circuits that break the security properties, for example, one that reveals the evaluator's private input purposely. Since the garbled circuit is "garbled", the evaluator could not tell whether a garbled circuit is faulty. The *circuit-level cut-and-choose* technique is one of the most efficient methods that enforces honest behavior and has been studied in several prior works [LP07, Kir08, LP11, SS11, MR13]. This technique instructs the generator to prepare multiple copies of the garbled circuit, each with independent randomness, and instructs the evaluator then to randomly pick a fraction of the circuits whose randomness is later revealed. If any of the chosen circuits (called the *check circuits*) is not consistent with the revealed randomness, the evaluator aborts; otherwise, the evaluator starts to evaluate the remaining circuits (called the *evaluation circuits*) as instructed in the Yao protocol. In the end, the evaluator picks the *majority circuit*, whose output matches the majority of the evaluation circuits, and takes the output of the majority circuit as her final output.

Unfortunately, the circuit-level cut-and-choose technique brings two new subtle issues that need to be addressed: (1) a malicious evaluator could either learn the generator's output or claim an arbitrary value to be the one, and (2) a malicious generator could potentially enter inconsistent inputs to different evaluation circuits and violate the evaluator's input privacy with non-negligible probability. A third problem, while irrelevant to the cut-and-choose technique, also needs to be addressed when dealing with malicious adversaries. This problem is that (3) a malicious generator could provide inconsistent random keys to the evaluator's input wire and its corresponding oblivious transfers, which is also known as the *selective failure attack*.

1.1 Contributions

The main contribution of this work is that we construct an optimal protocol in the circuit-level cut-and-choose-based category. Our protocol is optimal in the sense that it needs no more hardness assumptions than the honest-but-curious Yao protocol does, nor does it introduce any asymptotic overheads compared to the circuit-level cut-and-choose technique. In other words, we show that malicious security comes almost for free both in terms of required hardness assumptions and various protocol performance metrics. Our ideas are as follows (Note that n denotes the input/output size of the participants, k denotes the security parameter, and s denotes the number of garbled circuits needed. Typically, $s = O(k)$):

1. 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 $O(k^2n)$ symmetric operations [LP07,PSSW09] or $O(kn)$ symmetric operations plus $O(s) = O(k)$ number-theoretic operations [Kir08].

2. We suggest the use of an auxiliary circuit to achieve the generator's input consistency. We found that an auxiliary circuit that computes the hash of the generator's input can deter such a malicious behavior, more specifically, a hash that has the hiding property. We then suggest an instantiation that introduces only XOR-gates to each garbled circuit, $O(kn)$ XOR gates to be precise. By applying the free-XOR technique [KS08], the overhead is much less than those $O(s^2n) = O(k^2n)$ symmetric operations [MF06, LP07] or $O(sn) = O(kn)$ number-theoretic operations [SS11].
3. Most important, the above two techniques allow us to handle issues of two-output functions and the generator's input consistency while avoiding number-theoretic operations entirely. Lindell and Pinkas' [LP07] and Woodruff's [Woo07] approaches are the only prior works, to the best of our knowledge, that use no number-theoretic assumptions. However, our approach works in a more efficient manner as shown in Table 1.

	Gen. Input Consist.		**Gen. Output Auth.**		**Assumption**
	Symm. Op.	**Alge. Op.**	**Symm. Op.**	**Alge. Op.**	
[LP07]	$O(k^2n)$		$O(k^2n)$		Standard (OWF)
[Kir08]	$O(k^2n)$		$O(kn)$	$O(k)$	Discrete Log.
[LP11]	$O(kn)$	$O(kn)$	not mentioned		Decisional Diffie-Hellman
[SS11]	$O(kn)$	$O(kn)$	$O(kn)$	$O(kn)$	Discrete Log.
[KsS12]	$O(kn)$	$O(kn)$	$O(kn)$	$O(k)$	Discrete Log.
This Work	$O(k^2n)$ $O(kn)$		$O(kn)$ $O(kn)$		Standard (OWF) Free-XOR

Table 1: Complexity of various cut-and-choose-based approaches in terms of symmetric (or algebraic) operations.

4. Lindell and Pinkas suggested the use of a k-probe-resistant matrix (cf. Definition 1) to defend the selective failure attack [LP07]. This solution has little overhead while combining with the free-XOR technique. However, it increases the number of oblivious transfers needed at the same time. While XOR gates can be computed almost for free, the oblivious transfers can not. While there exists extension techniques for oblivious transfers, not all variants of oblivious transfers can be efficiently extended. We therefore come up with a probabilistic algorithm based on Reed-Solomon code such that the number of oblivious transfers needed is can be as low as 25% of that in the original solution.

Next, we explored an extremely important but often overlooked measure on secure computation—*the depth complexity*. We realized that secure computation is computation-intensive and highly parallelizable. It would be wasteful if a protocol does not utilize the modern architectures, which favor highly parallel processors. To work in the parallel environment, the depth complexity analysis helps us understand the fundamental limit of a protocol's real world performance such as wall-clock time. We give a detailed analysis on both our protocol and Nielsen et al.'s [NNOB12], which are two of the fastest protocols belonging to the constant-round and linear-round categories, respectively.

Finally, we experimentally verify our analysis and briefly report some interesting numbers here (see also Appendix I).

1. Building on the open-source system developed by Kreuter, shelat, and Shen [KsS12], we developed each of our new techniques and further optimized their system so that it can process up to 663,000 gates per second on the Stampede [Sta] as shown in Appendix I.2. This is the fastest maliciously secure 2-party system reported.
2. To verify that our performance in not entirely due to the cluster we used, we ported our system onto Amazon EC2 with the help of StarCluster [STA09]. Our results showed, in Appendix I.3, that it is not only feasible for the general public to do secure computation on EC2, it is also fairly inexpensive. For example, the amortized cost is 0.5¢ to run a block of AES (30 thousand gates) securely and $1.3 to run an instance RSA-64 (14 million gates).

1.2 Related Work

There has been fruitful progress in the secure two-party computation research in the malicious model. While substantial efforts have been spent on converting the Yao protocol based on the cut-and-choose technique [MF06, Woo07, LP07, Kir08, PSSW09, LP11, SS11, KsS12], several completely different approaches have been reported. Jarecki and Shmatikov suggested a approach that only a single copy of the garbled circuit is evaluated, but only after it is proven in zero knowledge to be correctly constructed [JS07]. Nielson et al. reported a solution that uses efficient oblivious transfers to generate a pool of authenticated primitives [NNOB12]. With these primitives, both parties are able to securely evaluate a boolean circuit based on the generic Goldreich, Micali, and Wigderson protocol [GMW87]. Damgård et al. proposed a solution that uses somewhat homomorphic encryption to precompute a bunch of triples that are then used to securely compute an arithmetic circuit [DPSZ12].

Our approach outperforms the rest in the cut-and-choose-based category in terms of the number of symmetric/number-theoretic operations needed, as shown in Table 1. Note that by "in the cut-and-choose-based category," we mean that the number of the garbled circuits needed is linear to the security parameter. So there is a hidden cost of $O(kC)$ of symmetric cryptographic operations in all these approaches, where C is circuit size. More details about Table 1 will be given in Section 3.

Our approach is also as competitive as any other in terms of computation complexity, round complexity, and computation assumptions, as shown in Table 2. Whereas Jarecki and Shmatikov's approach requires hundreds of expensive number-theoretic operations per gate, ours does not need any (given oracle access to oblivious transfers) [JS07]. Although Neilson et al.'s approach favorably compares to our approach in terms of computation complexity, ours requires constant communication rounds while theirs needs linear rounds to the circuit depth [NNOB12]. At last, since Damgård et al.'s approach works with arithmetic circuits, it is incomparable to our work and thus omitted in the table [DPSZ12]

We recently noticed an independent work by Mohassel and Riva that also proposes an optimal Yao-based protocol [MR13]. Their protocol indeed shares the same asymptotic complexity as ours and also relies on minimal assumptions. However, our approach favorably compares to their for two reasons:

1. One of the two solutions they proposed for solving two-output functions is problematic. Their idea is to assign the same random keys to the generator's output wire among all garbled circuits, and defer the opening of the check circuits *after* the evaluator finishes the evaluation and commits to the results. Since the random keys are the same, the evaluator can perform majority operation at the end without a problem. However, under this design, opening a check circuit will reveal both random

	Symmetric Op.	Algebraic Op.	Rounds	Assumption
[JS07]	$O(C)$	$O(C)$	$O(1)$	Composite Residuosity
[NO09]	$O(C \cdot \frac{k}{\lg(C)})$	$O(C \cdot \frac{k}{\lg(C)})$	$O(1)$	Discrete Log
[NNOB12]	$O(\frac{k}{\lg C} C)$		$O(D)$	Random Oracle
This Work	$O(kC)$		$O(1)$	Standard (OWF)

Table 2: Overall complexity comparison with prior works, where C is the size and D_f is the depth of the objective circuit.

keys associated with the generator's output wire in all the evaluation circuits, too. They claim this will not affect the soundness of the proof since the generator's output has been committed, which we agree. However, this disclosing of both random keys associated with the same wire violates the principle of the Yao protocol already, not to mention that in some objective circuit, the generator's output wire is an input to other gates, which will result in a chain effect of disclosures.

2. Their approach for checking the generator's input consistency uses a different instance of circuit garbling, in which the generator's and the evaluator's roles are reversed. While our approach introduces only XOR gates, theirs introduces oblivious transfers (two oblivious transfers for each of the generator's input bits to be exact), which is arguably the most expensive part of the Yao protocol.

Paper Organization: The rest of this paper is organized as follows. We first give background and notations in Section 2. We then show how the three attacks are handled by cryptographic primitives in Section 3. Next, we introduce our depth complexity analysis in Section 4. A detailed description of our full protocol is presented in Appendix E. Some experimental results are placed in Appendix I.

2 Preliminaries and Notations

Due to the limited space, we skip the introduction to the Yao protocol and assume the familiarity of this classic approach. Since our solution is based on the Yao protocol, the two participants are referred to as the generator and EVALfor the rest of this paper.

Notations: We use $f(x,y) \mapsto (f_1(x,y), f_2(x,y))$ to denote a two-output objective function, where the generator with input x gets output $f_1(x,y)$ and the evaluator with 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 objective function, we use $f_1 = \perp$ or $f_2 = \perp$ to indicate that either the generator or the evaluator gets no output. We use use $\mathsf{com}(x;r)$ to denote the commitment to message x with randomness r. The randomness could be omitted for simplicity. We use $\mathsf{enc}_e(x)$ to denote the encryption of message x under encryption key e, and use $\mathsf{dec}_d(c)$ to denote the decryption of ciphertext c under decryption key d. Additionally, we use $x||y$ or sometimes (x,y) to denote the concatenation of x and y, and use $[n]$ to denote the set $\{1,2,\ldots,n\}$ for some $n \in \mathbb{N}$.

3 Achieving Security in the Malicious Model

3.1 Two-Output Functions

The objective function could be a *two-output function*, in which the generator also gets an output. For these functions, a secure protocol needs to fulfill two properties: the generator's *output privacy* and *output authenticity*. The former ensures that the evaluator does not learn the generator's output, and the latter ensures 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* since the evaluator always learns her output and may refuse to send the generator's output back.

Many protocols for two-output functions have been proposed. Goldreich suggested the use of an auxiliary circuit that encrypts the generator's output and computes the digital signature of the resulting ciphertext so that a malicious evaluator could neither learn the generator's output from the ciphertext nor forge an arbitrary signature [Gol04]. Lindell and Pinkas later proposed an improved approach that uses one-time-pad encryption and one-time MAC circuits instead, which incurs $O(kn)$ extra gates per circuit [LP07]. Kiraz presented a two-party protocol in which a zero-knowledge proof of size $O(s)$ is conducted at the end [Kir08]. shelat and Shen reported a signature-based solution, which adds $O(n)$ gates to each circuit, and requires a witness-indistinguishable proof of size $O(s+n)$ [SS11].

Our approach solves the generator's output privacy problem with a one-time-pad encryption circuit, which requires only $O(n)$ extra XOR-gates. The novel part of our approach is that we handle 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 random keys obtained from evaluating the generator's output gates are in fact sufficient for the evaluator to prove the output authenticity. Recall that the Yao protocol guarantees that the evaluator learns one and only one of the random keys associated with each wire. In other words, the knowledge of the random keys corresponding to the generator's output is more than enough for the evaluator to show the output authenticity. What remains is how the evaluator shows the knowledge without disclosing the index of the majority circuit, which has been shown to be exploitable in breaching the evaluator'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 the generator (plays as the verifier) has private input $U = (u^{(1)}, u^{(2)}, \ldots, u^{(s)})$ for some $s \in \mathbb{N}$, and the evaluator (plays as the prover) knows $u^{(m)}$ for some $m \in [s]$. In the honest-but-curious model, a simple witness-indistinguishable proof of the evaluator's knowledge $u^{(m)}$ can be done as follows:

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

Unfortunately, this proof is not witness-indistinguishable in the malicious model. In fact, a malicious generator may pick distinct $(r^{(1)}, r^{(2)}, \ldots, r^{(s)})$ and send $\mathsf{enc}_{u^{(1)}}(r^{(1)})||\mathsf{enc}_{u^{(2)}}(r^{(2)})||\ldots||\mathsf{enc}_{u^{(s)}}(r^{(s)})$ to the evaluator so that the evaluator's knowledge could be identified by locating r' in $(r^{(1)}, r^{(2)}, \ldots, r^{(s)})$.

To force the generator to behave honestly in the first step, we suggest that the generator discloses $U = (u^{(1)}, u^{(2)}, \ldots, u^{(s)})$ and r after receiving r' so that the evaluator could check if $c^{(1)}||c^{(2)}||\ldots||c^{(s)}$ is

constructed correctly. Our third observation is that this disclosure does not compromise the generator's input privacy. Indeed, the evaluator should have already learned the majority of U from the circuit evaluation, and r is a random nonce that has no information about the generator's input at all. So the generator's input is not leaked through U and r. Moreover, this disclosure does not compromise the soundness of the protocol since after the generator receives r', the evaluator has already delivered her proof of authenticity so that learning U and r afterwards will not change the proof retroactively. Nonetheless, we stress that the generator should not learn r' before the evaluator finishes the check, and nor should the evaluator be able to change r' after the check. Naturally, a commitment scheme comes into the picture. Our idea is that the evaluator commits to $\mathsf{dec}_{u^{(m)}}(c^{(m)})$ instead of giving it away in clear. After the generator reveals U and r, the evaluator checks if $c^{(1)}||c^{(2)}||\ldots||c^{(s)}$ is indeed correctly generated. If the check fails, the evaluator aborts; otherwise, the evaluator decommits to r'. Then the generator checks if $r' = r$ and responds as in the above honest-but-curious protocol.

One more issue is that a malicious generator could learn the evaluator's private input $u^{(m)}$ with non-negligible probability by faking her private inputs from the beginning. More specifically, a malicious generator 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 the evaluator's private input is indeed $u^{(m)}$, that is, the fact that the evaluator 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 the generator's private inputs in the first place. By the binding property of the commitment scheme, a malicious generator cannot change her private inputs at will. The correctness of these commitments will be guaranteed by the circuit-level cut-and-choose technique.

Lemma 3.1. *Let* $\{(com(u_0^{(j)}), com(u_1^{(j)}))\}_{j\in[s]}$ *be common input. Suppose the generator has private input* $\{(u_0^{(j)}, u_1^{(j)})\}_{j\in[s]}$, *where* $u_b^{(j)} \in \{0,1\}^k$. *Then the protocol presented in Appendix A satisfies the following properties:*

1. *If the evaluator knows* $u_b^{(j)}$ *for some* $j \in [s]$ *and* $b \in \{0,1\}$, *the generator always accepts.*
2. *If the evaluator does not know any of* $u_b^{(j)}$, *the generator rejects with probability at least* $1 - 2^{-k}$.
3. *If the evaluator 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 the generator's view during the protocol with the evaluator's private input being* m.

3.2 Generator's Input Consistency

According to the circuit-level cut-and-choose technique, multiple copies of the garbled circuit are constructed and then either checked or evaluated. It is conceivable that a malicious generator could provide inconsistent inputs to different evaluation circuits. Lindell and Pinkas showed that for some functions, it is not difficult for a malicious generator to extract information of the evaluator's input [LP07]. For instance, suppose both parties agree upon the objective function

$$f(a_1||a_2||a_3, b_1||b_2||b_3) \mapsto (a_1b_1 \oplus a_2b_2 \oplus a_3b_3, \perp),$$

where a_i and b_i is the generator's and the evaluator's i-th input bit, respectively. Instead of providing $a_1||a_2||a_3$ consistently, a malicious generator may send $1||0||0$, $0||1||0$, and $0||0||1$ to different evaluation circuits. At the end, the generator learns the majority bit of the evaluator's input, which is the extra information that the evaluator did not agree to reveal in the first place.

Several approaches have been proposed to deter this attack. Mohassel and Franklin proposed the equality-checker technique, which uses $O(s^2n)$ commitments to be computed and exchanged [MF06]. Lindell and Pinkas developed an approach that also requires $O(s^2n)$ commitments [LP07]. Later, they realized that $O(s^2n)$ commitments are too much of the communication overhead, and then further suggested a pseudorandom synthesizer that relies on efficient zero-knowledge proofs under specific hardness assumptions and requires $O(sn)$ number-theoretic operations [LP11]. shelat and Shen proposed the use of malleable claw-free collections, which also uses $O(sn)$ number-theoretic operations, but they showed that the witness-indistinguishability is sufficient [SS11]. Our approach gets the best of the both worlds that it requires only $O(sn)$ symmetric cryptographic operations.

We suggest to tackle this issue by using a 2-universal hash circuit in addition to the objective circuit. The insight is that the Yao protocol itself will ensure that this auxiliary circuit is correctly constructed and its output (the hash) will tell the evaluator that the generator's inputs are consistent among the garbled circuits but nothing about their exact values. For this technique to work, we need to endow a 2-universal hash of the generator's input with the collision-free and hiding properties. The collision-free property ensures that the consistency of the hashes implies the consistency of the generator's inputs, and the hiding property ensures that the hashes do not reveal information of the generator's inputs.

The collision-free property comes naturally with the 2-universal hashes. Recall that a hash function family $\mathscr{H} = \{h|h : A \to B\}$ is called *2-universal* if for any *fixed*, distinct $x, y \in A$, the probability that a random $h \in \mathscr{H}$ satisfies that $h(x) = h(y)$ is at most $1/|B|$. This tells us that if the generator commits to her inputs $x_1, x_2, \ldots, x_s$ before a universal hash function h is randomly picked (hence $x_1, x_2, \ldots, x_s$ are fixed before h is chosen), where x_i is the generator's input to the i-th evaluation circuit, the probability that $x_i \neq x_j$ but $h(x_i) = h(x_j)$ is at most $1/|B|$. In other words, the consistency of $h(x_1), h(x_2), \ldots, h(x_s)$ will imply the consistency of $x_1, x_2, \ldots, x_s$ with probability at least $1 - 1/|B|$. We stress that by "commits to her input," the generator commits to the corresponding random keys rather than the actual values so that her inputs are fixed but not revealed even after decommitted.

The deterministic hash function 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, the evaluator could exhaustively compute the hash of all possibilities in A and then learn x_i from $h(x_i)$. As a consequence, the hash function has to be randomized. In particular, if the hashes are pseudorandom, they reveal little information of the input, which is the hiding property we desire. The Leftover Hash Lemma tells us that the output of a universal hash function is pseudorandom if the output size is smaller than the input's min-entropy minus the entropy loss [IZ89]. In other words, if x has enough min-entropy, $h(x)$ is pseudorandom even if h is made public, assuming h is picked unbiased. As a result, we need to introduce entropy to the input of the hash functions in order to have pseudorandom output hashes. We therefore suggest the solution that objective function $f(x, y) \mapsto (f_1, f_2)$ is converted into $g(x||r, y) \mapsto (f_1, h(x||r)||f_2)$, where r is a fresh randomness picked by the generator at the onset of the protocol and h is a 2-universal hash function picked *after* the generator's inputs are committed.

Next, we need to find an efficient 2-universal hash family. We propose the use of

$$\mathscr{M} = \{h_M |\ M \in \{0,1\}^{m\times n} \wedge h_M(x) = M \cdot x \text{ for some } m,n \in \mathbb{N}\}.$$

A nice property of this hash family is that once a random $h_M \in \mathscr{M}$ is sampled, the corresponding matrix can be made public. A circuit that computes a vector-matrix multiplication over field $\mathbb{Z}/2\mathbb{Z}$ with a public matrix *needs no non-XOR gates*. For example, suppose

$$M = \begin{bmatrix} 0\,1\,0 \\ 1\,0\,1 \\ 0\,1\,1 \end{bmatrix}, \text{ then } M \cdot \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} b_2 \\ b_1 \oplus b_3 \\ b_2 \oplus b_3 \end{bmatrix},$$

which shows that the circuit needs only XOR gates. It is worth-mentioning that this approach introduces minimal overhead when the free-XOR technique is applied [KS08].

It remains to show the parameter selection for the desired security parameter k. By the definition of 2-universal hash function family, the generator 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 the generator'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 $\mathscr{M}$, we can do a little better than this, that is, we can reach the same goal with $|r| = 2k + \lg(k)$.

Lemma 3.2. *Let $\mathscr{M} = \{h_M |\ M \in \{0,1\}^{k\times(n+t)}$ 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 $h_M \in \mathscr{M}$ satisfies that $h_M(x||r)$ is uniformly distributed over $\{0,1\}^k$ is at least $1 - 2^{-k}$.*

3.3 Selective Failure Attack

A subtle attack—*selective failure attack*—possible in the malicious model has been pointed out by Mohassel and Franklin [MF06] and independently by Kiraz and Schoenmakers [KS06]. This attack happens when a malicious generator uses inconsistent random keys to construct the garbled gate and the corresponding oblivious transfer so that the evaluator's input can be inferred according to whether the protocol completes. More specifically, a malicious generator could assign (K_0, K_1) to an the evaluator's input wire in the garbled circuit while using (K_0, K_1^*) instead in the corresponding oblivious transfer, where $K_1 \neq K_1^*$. Consequently, if the evaluator's input is 0, she learns K_0 from the oblivious transfer and completes the evaluation without complaints; otherwise, she learns K_1^* and gets stuck during the evaluation, and hence, the generator learns that the evaluator's input would have been 1.

Lindell and Pinkas suggested an auxiliary circuit to deter the selective failure attack [LP07]. In their solution, the evaluator picks $M \in \{0,1\}^{n\times m}$ for some $m \in \mathbb{N}$ and computes her new input $\overline{y} \in \{0,1\}^m$ such that $M \cdot \overline{y} = y$. The auxiliary circuit will later convert $\overline{y}$ back to y to ensure correctness. The insight is that now the fact that the generator gets to probe some partial information of $\overline{y}$ should not

grant her non-negligible advantage to learn partial information of the evaluator's real input y. Formally, for this approach to work, the security property needed here is defined as follows:

Definition 1. *$M \in \{0,1\}^{n\times m}$ for some $n, m \in \mathbb{N}$ is called k-*probe-resistant *for some $k \in \mathbb{N}$ if for any $L \subset \{1,2,\ldots,n\}$, 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 generator will have to successfully probe k bits of $\overline{y}$ 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 [KS08]. 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.

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 the evaluator'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,

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

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

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

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

$$(2^t)^K - 1 = 2^{Kt} - 1 \geq n. \tag{3}$$

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 $\overline{y}$ such that $M \cdot \overline{y} = y$. On the other hand, if there indeed exists such a $\overline{y}$, a malicious generator can easily deduce that $y_3 = y_1 + y_2$, which is the partial information of y that the generator is not supposed to learn. Lemma 3.3 shows that with probability at least $1 - 2^{-k}$, no polynomial is a subset sum of the rest as long as

$$K \geq (\lg(n) + n + k)/t. \tag{4}$$

Lemma 3.3. *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 (1)(2)(4)[2], we know that

$$2^t \geq N + 1 \geq k + K \geq k + (\lg(n) + n + k)/t \tag{5}$$

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 (5).

Finally, we provide a probabilistic algorithm in Appendix B finding a k-probe-resistant M with high probability. The correctness of this algorithm is shown in Theorem 3.1.

Theorem 3.1. *With probability at least $1 - 2^{-k}$, the algorithm presented in Appendix B outputs a k-probe-resistant $M \in \{0,1\}^{n \times m}$ such that $m \in \mathbb{N}$ and $m \leq \lg(n) + n + k + k \cdot \max(\lg(4n), \lg(4k))$.*

Theorem 3.2. *Assume that the $\binom{2}{1}$-OT protocol is secure in the presence of malicious adversaries, and there exist a commitment scheme, a family of pseudo-random functions, the Main protocol $\Pi = (\textsc{Gen}, \textsc{Eval})$ presented in Appendix E securely computes $f : (x, y) \mapsto (f_1, f_2)$ in the presence of malicious adversaries.*

4 The Depth Complexity Analysis

A two-party secure computation protocol is a method for transforming a circuit for a given function f into a protocol that two parties execute to compute the same function f. A natural goal in cryptographic protocol design is to reduce some aspect of the overhead of this process; one can reduce the complexity assumptions, the round complexity, the communication complexity, or the computational complexity of the method.

[2] Constraint (3) is in fact weaker than Constraint (4) and thus not needed.

In this section, we suggest that the *circuit depth complexity* of the transformation is one of the most important (yet overlooked) measures of its efficiency. We believe this measure is important because modern computer designs provide more and more "independent cores" instead of faster single cores. Naturally, a protocol that has low circuit depth can exploit such hardware more effectively. Our empirical results in later sections confirm this intuition. To formalize this, we put forth the following:

Definition 2 (k-core model). *If k is the security parameter, each party of a protocol in the k-core model has k-independent processors and bandwidth $O(k)$ to each of the other parties.*

The assumption of easy access to a cluster can be justified as cloud services such as Amazon EC2, Microsoft Azure, or Google Cloud become common. With reasonable efforts, anyone can launch a cluster of 512 nodes for an hourly fee of roughly $30 from Amazon EC2.

We now introduce a circuit model that allows us to easily, yet realistically evaluate the circuit depth of various approaches to two-party secure computation. Naturally, the basic AND and XOR gates, and all other primitive operations such as compare are modeled with depth complexity 1. We use D_f to denote the depth of the objective circuit for function f.

Many protocols make use of cryptographic primitives such as pseudo-random functions, oblivious transfer, etc. Some protocols require specific number-theoretic assumptions. For simplicity, we treat these primitives as a black-box and use D_{ENC} to denote the depth of a PRF (also symmetric encryption scheme), D_{G} to denote the depth of a number-theoretic operation, D_{OT} to denote the depth of an (honest-but-curious secure) oblivious transfer, and D_{MOT} to denote the depth of a (maliciously secure) oblivious transfer. We stress that security parameter k is always an implicit parameter to the above cryptographic primitives (except for D_f) and omitted for clarity.

4.1 The Depth of the Yao Protocol

Lemma 4.4. *The depth complexity of honest-but-curious Yao is upper-bounded by $D_{OT} + O(D_f) \cdot D_{ENC}$.*

The classic Yao protocol consists of two major parts—oblivious transfers and the garbling. We first observe that in the honest-but-curious model, the security of oblivious transfers is not compromised under parallel execution. So the first part of the Yao protocol has depth D_{OT}, regardless of the evaluator's input size. Next, we realize that the depth for circuit garbling can be as small as $(1+2) \cdot D_{\text{ENC}}$ by using a symmetric cipher in the counter mode to pick the garbled keys associated with each wire and computing the doubly encrypted entries in parallel. As a result, the depth of garbling the whole circuit is the same as that of garbling a single gate, which is the same as that of garbling a single truth table entry. As for the other party, evaluating a garbled gate requires only one double decryption, and after exploiting the gate-level parallelism, there are only D_f level of garbled gates. So the depth for the evaluator is $D_f \cdot 2 \cdot D_{\text{ENC}}$. Overall, the depth for the Yao protocol is $D_{\text{OT}} + (3 + 2 \cdot D_f) \cdot D_{\text{ENC}} = D_{\text{OT}} + O(D_f) \cdot D_{\text{ENC}}$.

4.2 The Depth of Our Proposed Protocol

Lemma 4.5. *The protocol presented in Appendix E has depth complexity $D_{MOT} + O(\lg(n+k) + D_f) \cdot D_{ENC}$.*

The analysis of the protocol proposed in this work can be divided into the following five parts:

1. (**Oblivious Transfer**) As suggested by Pinkas et al. that n malicious oblivious transfers can be done in parallel under the Peikert, Vaikuntanathan, and Waters' construction [PSSW09, PVW08]. Thus, n oblivious transfers have the same depth as one, which is D_{MOT}. Moreover, all copies of the garbled circuit can share the same n oblivious transfers with the price of an encryption and a decryption of s garbled values[3], the depth of which is $2 \cdot D_{\text{ENC}}$ due to the counter mode usage of a symmetric cipher. So the depth of this part is $D_{\text{MOT}} + 2 \cdot D_{\text{ENC}}$.
2. (**The Generator's Input Consistency Check**) The main component here is to jointly sample a 2-universal hash function, which is basically a $k(n + 2k + \lg(k))$-bit random string. The coin flipping protocol consists of a committing operation, an opening operation, and a XOR in sequence. Since the committing and opening can be done in a block-wise manner, the depth for this part is $2 \cdot D_{\text{ENC}} + 1$.
3. (**Circuit Choosing**) This part could be merged into the previous part, and add no depth at all.
4. (**Circuit Garbling/Checking or Evaluating**) Similar to the Yao protocol, garbing a circuit has depth $3 \cdot D_{\text{ENC}}$. Checking a circuit is basically regenerating it (with the revealed randomness) followed by a comparison, the depth of which is therefore $3 \cdot D_{\text{ENC}} + 1$. Also similar to the Yao protocol, evaluating a circuit has depth $D_f \cdot D_{\text{ENC}}$. Note that the auxiliary circuit computing a 2-universal hash of the generator's input may contribute here. In fact, since this is auxiliary circuit, after fully parallelized, is merely a scalar product of $O(n + k)$-bit strings. The depth of this auxiliary circuit is thus $O(\lg(n + k)) \cdot D_{\text{ENC}}$. Since all circuits are garbled and then either checked or evaluated, the overall depth of this step is $(3 + O(\lg(n + k) + \max(3, D_f)) \cdot D_{\text{ENC}} = O(\lg(n + k) + D_f) \cdot D_{\text{ENC}}$.
5. (**The Generator's Output Authenticity Proof**) This proof first requires the generator to encrypt a random nonce with a key consisting of n garbled values, the depth of which is D_{ENC}[4]. Next, the proof instructs the evaluator to commit to a decryption of the ciphertext of her choice, the depth of which is $2 \cdot D_{\text{ENC}}$. After the evaluator verifies the correctness of all the ciphertexts, the depth of which is $D_{\text{ENC}} + 1$ (decrypting plus a comparison), the generator checks the commitment, the depth of which is $D_{\text{ENC}} + 1$ (decommitting plus a comparison). So the depth for this part is $5 \cdot D_{\text{ENC}} + 2$.

As a result, the overall depth complexity of our protocol is $D_{\text{MOT}} + O(\lg(n + k) + D_f) \cdot D_{\text{ENC}}$

4.3 The Depth of the Nielsen et al.'s Protocol [NNOB12]

In CRYTPO 2012, Nielsen et al. proposed a groundbreaking protocol that reduces the overall overhead by a factor of $\lg(C)$, where C is the size of the objective circuit. In particular, this protocol reduces the duplication factor from $O(k)$ to $O(k/\lg(C))$. Recall that the duplication factor for the circuit-level cut-and-choose-based Yao protocol is typically $s = O(k)$. We are interested in its depth complexity, but before we get any further, let us briefly introduce its base protocol—the Goldreich, Micali, and Wigderson protocol [GMW87].

Briefly speaking, the Goldreich, Micali, and Wigderson protocol instructs both participants to jointly compute the objective circuit in a way that the value for each wire is additively shared among two parties. In particular, this protocol requires an oblivious transfer for each AND gate. However, due to

[3] This is possible if after $\text{OT}((a_0, a_1), b) \mapsto (\perp, a_b)$, the generator uses a_0 and a_1 to encrypt the concatenation of the garbled values associated with the evaluator's input wires in all garbled circuit so that only those encrypted with a_b can be retrieved by the evaluator.

[4] If $\overline{\mathsf{enc}}_{K_1||K_2||\cdots||K_n}(r) \stackrel{\text{def}}{=} \mathsf{enc}_{K_1}(r)||\mathsf{enc}_{K_2}(r)||\cdots\mathsf{enc}_{K_n}(r)$, the depth of $\overline{\mathsf{enc}}$ is essentially the same as that of enc.

the technique of OT extensions, these oblivious transfers can be done in batch at the beginning so that each AND gate later requires only simple operation. So its depth complexity is $D_{\text{OT}} + O(D_f)$.

Conceptually, Nielsen et al.'s protocol is clever in the sense that it precomputes random oblivious transfers in the offline phase and later uses lightweight operations such as XORs to manipulate these precomputation results in the online phase. Note that by adopting the UC model, they can use oblivious transfers safe to run in parallel. In other words, not only is the offline phase highly parallelizable, the online phase requires only inexpensive operations. As a result, Neilsen et al.'s protocol has the depth complexity roughly $D_{\text{MOT}} + O(D_f)$ (but not quite this simple as details will be discussed later).

Here, we analyze Nielsen et al.'s protocol in five parts:

1. (**LaBit Protocol**) This protocol starts with $\frac{44}{3}k$ oblivious transfers, which has depth D_{MOT}, followed by $\frac{22}{3}k$ instances of the EQ protocol over $O(C \cdot \frac{k}{\lg(C)})$-bit strings, which compute a hash of the input bit-string and thus has depth $O(\lg(C) - \lg\lg(C)) \cdot D_{\text{ENC}} = O(\lg(C)) \cdot D_{\text{ENC}}$. So this part has depth $D_{\text{MOT}} + O(\lg(C)) \cdot D_{\text{ENC}}$.
2. (**WaBit Protocol**) This part is only a matrix transposition and thus has no depth at all.
3. (**aBit Protocol**) This part requires parallel matrix multiplications that convert many $\frac{22}{3}k$-bit strings into k-bit ones independently, which after fully parallelized, has depth complexity $\lg(\frac{22}{3}k) = O(\lg(k))$.
4. (**aOT and aAND Protocols**) An aOT protocol is a combination result of $O(\frac{k}{\lg(C)})$ instances of LaOT. Since each LaOT requires a sequential invocations of a constant number of the random oracle, LaOT has depth $O(1) \cdot D_{\text{ENC}}$. Also, since a combination of l instances of depth D_{ENC} circuit can have depth $\lg(l) \cdot D_{\text{ENC}}$ by combining in a binary-tree manner, we conclude that aOT has depth $O(\lg(k) - \lg\lg(C)) \cdot D_{\text{ENC}} = O(\lg(k)) \cdot D_{\text{ENC}}$. The same analysis applies to the aAND protocol.
5. (**The Main Protocol**) Finally, since the online phase requires a constant number of primitive operations such as XOR, its depth is $O(D_f)$.

Note that the above analysis is in a bottom-up and sequential manner. In other words, the current part will not start until the previous part is completely finished. As a result, the overall depth complexity is the summation of all parts and thus $D_{\text{MOT}} + O(\lg(k) + \lg(C)) \cdot D_{\text{ENC}} + O(D_f)$.

The results of our analysis are summarized in Table 3.

Protocol	Depth Complexity	Assumptions	Threat Model
[GMW87]	$D_{\text{OT}} + O(D_f)$	Standard	HBC
[NNOB12]	$D_{\text{MOT}} + O(\lg(k) + \lg(C)) \cdot D_{\text{ENC}} + O(D_f)$	Random Oracle	Malicious
[Yao82]	$D_{\text{OT}} + O(D_f) \cdot D_{\text{ENC}}$	Standard	HBC
This Work	$D_{\text{MOT}} + O(\lg(n+k) + D_f) \cdot D_{\text{ENC}}$	Standard	Malicious

Table 3: The depth complexity of various existing secure computation protocols

References

DPSZ12. I. Damgård, V. Pastro, N.P. Smart, and S. Zakarias. Multiparty Computation from Somewhat Homomorphic Encryption. In *Proceedings of the 32th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '12, 2012. http://eprint.iacr.org/2011/535.

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.

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.

KS08. 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.

KS11. Benjamin Kreuter and Chih-hao Shen. BetterYao, 2011. http://crypto.cs.virginia.edu/index.php/Two_Party_Computation.

KsS12. 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.

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.

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 *USENIX Security*, volume 13, pages 287–302, 2004.

MR13. Payman Mohassel and Ben Riva. Garbled Circuits Checking Garbled Circuits: More Efficient and Secure Two-Party Computation, 2013. http://eprint.iacr.org/2013/051.

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.

NO09. Jesper Nielsen and Claudio Orlandi. LEGO for Two-Party Secure Computation. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 368–386. Springer, 2009.

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.

PVW08. Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In *Proceedings of the 28th Annual conference on Cryptology: Advances in Cryptology*, CRYPTO 2008, pages 554–571, Berlin, Heidelberg, 2008. Springer-Verlag.

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.

Sta. Stampede. Tacc stampede. http://www.tacc.utexas.edu/user-services/user-guides/stampede-user-guide.

STA09. STAR (The Software Team for Academic and Research) group at MIT. StarCluster, 2009. http://star.mit.edu/cluster/.

Woo07. David Woodruff. Revisiting the Efficiency of Malicious Two-Party Computation. In Moni Naor, editor, *EUROCRYPT 2007*, volume 4515 of *LNCS*, pages 79–96. Springer, 2007.

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.

A The Proof for the Generator's Output Authenticity

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 EVAL $\mathsf{enc}_{u_a^{(1)}}(r)||\mathsf{enc}_{u_a^{(2)}}(r)||\cdots||\mathsf{enc}_{u_a^{(s)}}(r)$.
2. After receiving $c^{(1)}||c^{(2)}||\cdots||c^{(s)}$, 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 $j \in [s]$?
   (b) if $\mathsf{dec}_{u^{(j)}}(c^{(j)})$ equals $\mathsf{dec}_{u^{(j+1)}}(c^{(j+1)})$ for $j = 1, 2, \ldots, s-1$?
   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.

B The Probabilistic Algorithm for Generating a k-Probe-Resistant Matrix

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 $m \in \mathbb{N}$

$t \leftarrow \lceil \max(\lg(4n), \lg(4k)) \rceil$; `// find the minimum t such that` $2^t \geq k + (\lg(n) + n + k)/t$
while $2^{t-1} > k + (\lg(n) + n + k)/(t-1)$ **do**
  $t \leftarrow t - 1$;
end
$K \leftarrow \lceil (\lg(n) + n + k)/t \rceil$;
$N \leftarrow K + k - 1$;
for $i \leftarrow 1$ ***to*** n **do**
  Pick $P(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{K-1}x^{K-1}$, where $a_i \leftarrow_R \mathbb{F}_{2^t}$;
  $M_i \leftarrow [P(1)_2||P(2)_2||\ldots||P(N)_2]$; `//` $P(i)_2$ `denotes a` t`-bit row vector`
end
return M; `//` $M \in \{0,1\}^{n\times m}$`, where` $m = Nt$

C Proofs

Proof to Lemma 3.1

Proof. Here is the proof sketch for the completeness, soundness, and witness-indistinguishability:

(**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 $r_1' = \mathsf{dec}_{v_1}(c^{(m_1)}) = \mathsf{dec}_{u^{(m_1)}}(c^{(m_1)}) = \mathsf{dec}_{u^{(m_2)}}(c^{(m_2)}) = \mathsf{dec}_{v_2}(c^{(m_2)}) = r_2'$. □

Proof to Lemma 3.2

Proof. Note that picking $h_M \in \mathscr{M}$ 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

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

□

Proof to Lemma 3.3

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 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

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

□

D The Yao Protocol

In this section, we present the classic Yao protocol. In particular, our presentation uses the permuted garbled truth table technique [MNPS04] (also known as the *point-and-permute* technique in literature). Briefly, this technique suggests to assign each wire w_i a random permutation bit π_i. The circuit garbling and evaluating is also slightly modified: for Gen, each garbled truth table is constructed in the way that its entries are permuted according to its input wires' permutation bits; and for Eval, each random key now comes with a *locator* that helps Eval identify the right entry in the garbled truth table. It is worth-mentioning that for each wire, Eval learns a key-locator pair from the circuit evaluation, and the locator is in fact the evaluation result of that wire one-time padded with the permutation bit. As a result, not only does this property ensure that Eval is oblivious to the intermediate result of the circuit evaluation, it also allows Eval to learn the output by revealing the permutation bits assigned to circuit-output wires.

Protocol 1: The Yao Protocol

Common Input: a security parameter 1^k, a symmetric encryption scheme $(\mathsf{enc}, \mathsf{dec})$ with semantic security, and boolean circuit C computing $f(x, y)$.
Private Input: Gen has $x = x_1 x_2 \cdots x_{m_1}$ and Eval has $y = y_1 y_2 \cdots y_{m_2}$.
Output: Both Gen and Eval receive $f(x, y)$ at the end.

1. **(Circuit Garbling)** Gen garbles C as follows:
   (a) For each wire w_i, Gen randomly picks $(K_{i,0}, K_{i,1}, \pi_i) \in \{0,1\}^{2k+1}$. 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 computes its garbled truth table

$$G(g) = (\langle \pi_a, \pi_b \rangle, \langle \pi_a, 1 \oplus \pi_b \rangle, \langle 1 \oplus \pi_a, \pi_b \rangle, \langle 1 \oplus \pi_a, 1 \oplus \pi_b \rangle),$$

   where $\langle b_1, b_2 \rangle = \mathsf{enc}_{K_{a,b_1}}(\mathsf{enc}_{K_{b,b_2}}(W_{c,g(b_1,b_2)}))$.
   Gen sends $G(C) = (\{G(g)\}_{g \in C}, \{\pi_i : w_i \text{ is a circuit-output wire}\})$ to Eval.
2. **(Input Labels Retrieving)** Let $\{w_i\}_{i \in [m_1]}$ be the circuit-input wires corresponding to Gen's input, and let $\{w_{m_1+i}\}_{i \in [m_2]}$ be the circuit-input wires corresponding to Eval's input.
   (a) **(Gen's Input Labels)** Gen sends Eval the labels corresponding to her input $\{W_{i,x_i}\}_{i \in [m_1]}$.
   (b) **(Eval's Input Labels)** For each $i \in [m_2]$, Gen and Eval execute a $\binom{2}{1}$-OT

$$((W_{m_1+i,0}, W_{m_1+i,1}), y_i) \mapsto (\perp, W_{m_1+i,y_i}).$$

   The above $\binom{2}{1}$-OTs could all be run in parallel.
3. **(Circuit Evaluating)** Eval has now obtained garbled circuit $G(C)$ and $(m_1 + m_2)$ labels corresponding to the $(m_1 + m_2)$ 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 E in $G(g)$ and computes the output label

$$W_c = (K_c, \delta_c) = \mathsf{dec}_{K_b}(\mathsf{dec}_{K_a}(E)).^{a}$$

(b) For each circuit-output wire w_i with corresponding label $W_i = (K_i, \delta_i)$, EVAL computes the wire value $b_i = \delta_i \oplus \pi_i$. Recall that π_i for circuit-output wire w_i comes with $G(C)$.

In the end, EVAL interprets $\{b_i\}$ as $f(x, y)$ and sends $f(x, y)$ to GEN. Both parties output $f(x, y)$.

[a] Note that the decryption order has to be in reverse of the encryption order.

E The Full Description of the Main Protocol

We have introduced the components tackling the three known issues in Section 3. We now give the full protocol that transforms the Yao protocol in Section D into the one secure in the malicious model.

Protocol 2: The Main Protocol

Common Input: a security parameter 1^k, a statistical security parameter 1^σ, a symmetric encryption scheme $(\mathsf{enc}, \mathsf{dec})$ with semantic security, a commitment scheme com, and objective function $f : (x, y) \mapsto (f_1, f_2)$.
Private Input: Gen has input x and Eval has input y.
Private Output: Gen receives output f_1 and Eval receives output f_2.

1. Gen randomly picks $r \in \{0,1\}^{2k+\lg(k)}$ (as her random input to the 2-universal circuit) and e (as a one-time pad for f_1). Eval computes a k-probe-resistant matrix M and $\bar{y}$ such that $M \cdot \bar{y} = y$. From now on, Gen's input refers to $\bar{x} = x||e||r = \bar{x}_1\bar{x}_2\ldots\bar{x}_{m_1}$ and Eval's input refers to $\bar{y} = \bar{y}_1\bar{y}_2\ldots\bar{y}_{m_2}$.
2. **(Pick the randomness)** Gen picks randomness $\{\rho^{(j)}\}_{j\in[\sigma]}$.
3. **(Commit to Gen's input)** Gen uses $\rho^{(j)}$ to pick $(K^{(j)}_{i,0}, K^{(j)}_{i,1}, \pi^{(j)}_i) \in \{0,1\}^{2k+1}$ for $j \in [\sigma]$ and $i \in [m_1]$. Let $W^{(j)}_{i,b}$ denote the key-locator pair $(K^{(j)}_{i,b}, b \oplus \pi^{(j)}_i)$. $W^{(j)}_{i,b}$ is called the *label* corresponding to wire w_i of value b in the j-th garbled circuit from now on. Gen commits to her input by sending $\{\Gamma^{(j)}\}_{j\in[\sigma]}$ (the randomness of which needs to be independent of $\rho^{(j)}$) to Eval, where
$$\Gamma^{(j)} = \{\mathsf{com}(W^{(j)}_{i,\bar{x}_i}; \gamma^{(j)}_i)\}_{i\in[m_1]}.$$
4. **(Determine the objective circuit)** Eval first announces matrix M, and then both parties jointly run a coin flipping protocol[a] to randomly pick $H \in \{0,1\}^{k\times m_1}$. Both parties now have determined the objective circuit C that computes $g : (\bar{x}, \bar{y}) \mapsto (\perp, (h, c, g_2))$, where $h = H \cdot \bar{x}$, $g_1 = f_1(x, M \cdot \bar{y})$, $c = g_1 \oplus e$, and $g_2 = f_2(x, M \cdot \bar{y})$.
5. **(Commit to input/output label pairs)** Let $\{w_i\}_{i\in[m_1]}$ be the circuit-input wires corresponding to Gen's input, $\{w_{m_1+i}\}_{i\in[m_2]}$ be those corresponding to Eval's input, and $\{w_i\}_{i\in O_2}$ be the circuit-output wires corresponding to Gen's output (output c in particular). Gen uses randomness $\rho^{(j)}$ to
   (a) pick $(K^{(j)}_{i,0}, K^{(j)}_{i,1}, \pi^{(j)}_i) \in \{0,1\}^{2k+1}$ for all but Gen's input wires[b] and
   (b) commit to the label pairs assigned to Gen's input wires, Eval's input wires, and Gen's output wires by sending $\{\Theta^{(j)}, \Omega^{(j)}, \Phi^{(j)}\}_{j\in[\sigma]}$[c] to Eval, where
$$\Theta^{(j)} = \{\mathsf{com}(W^{(j)}_{i,0\oplus\pi^{(j)}_i}; \theta^{(j)}_i), \mathsf{com}(W^{(j)}_{i,1\oplus\pi^{(j)}_i}; \theta^{(j)}_i)\}_{i\in[m_1]},$$
$$\Omega^{(j)} = \{\mathsf{com}(W^{(j)}_{m_1+i,0}; \omega^{(j)}_i), \mathsf{com}(W^{(j)}_{m_1+i,1}; \omega^{(j)}_i)\}_{i\in[m_2]},$$
$$\Phi^{(j)} = \{\mathsf{com}(W^{(j)}_{i,0}), \mathsf{com}(W^{(j)}_{i,1})\}_{i\in O_2}.$$

6. Both parties jointly execute $(m_2+\sigma)$ instances of $\binom{2}{1}$-OTs. In particular,
   (a) **(Eval's Input OTs)** For each $i \in [m_2]$, both parties run a $\binom{2}{1}$-OT in which Gen's input equals
   $$\left(\{(W^{(j)}_{m_1+i,0}, \omega^{(j)}_i)\}_{j\in[\sigma]}, \{(W^{(j)}_{m_1+i,1}, \omega^{(j)}_i)\}_{j\in[\sigma]}\right)$$
   and Eval's input equals $\bar{y}_i$. Let $Y^{(j)}$ denote the set of decommitments Eval received for the j-th garbled circuit, that is, $Y^{(j)} = \{(W^{(j)}_{m_1+i,\bar{y}_i}, \omega^{(j)}_i)\}_{i\in[m_2]}$.[d]
   (b) **(Circuit OTs)** Eval randomly picks $S \subset [\sigma]$ such that $|S| = 2\sigma/5$. Let $s \in \{0,1\}^\sigma$ such that $s_j = 1$ if $j \in S$; and $s_j = 0$ otherwise. For each $j \in [\sigma]$, both parties run a $\binom{2}{1}$-OT in which Gen's input equals $(\rho^{(j)}, X^{(j)})$ and Eval's input equals s_j, where $X^{(j)} = X^{(j)}_1 \cup X^{(j)}_2$[e] and
   $$X^{(j)}_1 = \{(W^{(j)}_{i,\bar{x}_i}, \gamma^{(j)}_i)\}_{i\in[m_1]}, \qquad X^{(j)}_2 = \{(W^{(j)}_{i,\bar{x}_i}, \theta^{(j)}_i)\}_{i\in[m_1]}.$$
   In other words, if $s_j = 0$, Eval will learn the randomness and check the j-th circuit; otherwise, Eval will retrieve Gen's input labels and evaluate the j-th circuit.

   The above $\binom{2}{1}$-OTs could all be run in parallel. Either party aborts if any $\binom{2}{1}$-OT fails.
7. **(Circuit Garbling)** 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 computes its garbled truth table
   $$G(g)^{(j)} = (\langle \pi^{(j)}_a, \pi^{(j)}_b\rangle, \langle \pi^{(j)}_a, 1\oplus\pi^{(j)}_b\rangle, \langle 1\oplus\pi^{(j)}_a, \pi^{(j)}_b\rangle, \langle 1\oplus\pi^{(j)}_a, 1\oplus\pi^{(j)}_b\rangle),$$
   where $\langle b_1, b_2\rangle = \mathsf{enc}_{K^{(j)}_{a,b_1}}(\mathsf{enc}_{K^{(j)}_{b,b_2}}(W^{(j)}_{c,g(b_1,b_2)}))$.
8. **(Circuit Checking)** Gen then sends $\{G(C)^{(j)}\}_{j\in[\sigma]}$ to Eval, where
   $$G(C)^{(j)} = \left(\{G(g)^{(j)}\}_{g\in C}, \{\pi^{(j)}_i : w_i \text{ is a circuit-output wire}\}\right).$$
   - **(Check Circuits)** For each $j \in [\sigma]\backslash S$, Eval checks if $\rho^{(j)}$ received in Step 6b can be used to regenerate $\{\Theta^{(j)}, \Omega^{(j)}, \Phi^{(j)}\}$ received in Step 5b and reconstruct $G(C)^{(j)}$?
   - **(Evaluation Circuits)** For each $j \in S$, Eval checks:
     (a) if $X^{(j)} = X^{(j)}_1 \cup X^{(j)}_2$ received in Step 6b successfully decommits $\Gamma^{(j)}$ received in Step 3 and half of $\Theta^{(j)}$ received in Step 5b? and if the decommitted labels match? In particular,
       i. if the i-th entry of $X^{(j)}_1$ successfully decommits the i-th entry of $\Gamma^{(j)}$?
       ii. if the i-th entry of $X^{(j)}_2$ successfully decommits the $(2\cdot i + \bar{x}_i \oplus \pi^{(j)}_i)$-th entry of $\Theta^{(j)}$?
       iii. if the i-th decommitted labels from the above two steps coincide?
     (b) if $Y^{(j)}$ received in Step 6a successfully decommits half of the commitments in $\Omega^{(j)}$? In particular, if the i-th entry of $Y^{(j)}$ successfully decommits the $(2\cdot i + \bar{y}_i)$-th entry of $\Omega^{(j)}$?

     Eval aborts immediately as long as a failure occurs.
9. **(Circuit Evaluating)** Eval has now obtained garbled circuit $G(C)^{(j)}$ and (m_1+m_2) labels corresponding to the circuit-input wires of C. Eval then evaluates the circuit 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 E in $G(g)^{(j)}$ and computes its output label

$$W_c^{(j)} = (K_c^{(j)}, \delta_c^{(j)}) = \mathsf{dec}_{K_b^{(j)}}(\mathsf{dec}_{K_a^{(j)}}(E)).$$

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

Eval interprets $\{b_i^{(j)}\}$ as $(h^{(j)}, c^{(j)}, g_2^{(j)})$. Let $Z^{(j)}$ denote the labels that came with $c^{(j)}$.

10. **(Majority Operation)** Let (h, c, g_2) be the most occurred element in $\{(h^{(j)}, c^{(j)}, g_2^{(j)})\}$. Eval aborts
    (a) if $h^{(j)} \neq h$ for any $j \in S$, or
    (b) if (h, c, g_2) is not the majority in $\{(h^{(j)}, c^{(j)}, g_2^{(j)})\}$, that is,

$$\left|\left\{(h^{(j)}, c^{(j)}, g_2^{(j)}) : (h^{(j)}, c^{(j)}, g_2^{(j)}) = (h, c, g_2)\right\}\right| \leq \frac{|S|}{2} = \frac{\sigma}{5}.$$

    If Eval does not abort, she outputs g_2.

11. **(Gen's Output Authenticity Proof)** The two parties conduct the protocol presented in Appendix A. In particular, common inputs include security parameter 1^k, statistical security parameter $1^{|S|}$, commitments to label pairs assigned to Gen's output wires $\{\Phi^{(j)}\}_{j \in S}$, and Gen's to-be-proved output c. Also, Gen' private input equals $\{W_{i,0}^{(j)}, W_{i,1}^{(j)}\}_{i \in O, j \in S}$ and Eval's private input equals $(j, Z^{(j)})$ for some $j \in S$ such that $(h^{(j)}, c^{(j)}, g_2^{(j)}) = (h, c, g_2)$.
    If Gen accepts the proof, Gen outputs $c \oplus e$; otherwise, Gen aborts.

---

[a] Here we use a simple coin-flipping protocol that works as follows: (1) Eval sends to Gen the commitment to a random string ε; (2) Gen sends to Eval a random string ε'; (3) Eval opens ε; (4) Gen checks the correctness of Eval's commitment and outputs $\varepsilon \oplus \varepsilon'$ if the check passes; otherwise, aborts.

[b] The random keys assigned to Gen's input wires were already picked in Step 3.

[c] In this step, Gen commits to the label pairs assigned to Gen's input wires, Eval's input wires, and Gen's output wires (for authenticity proof). Later Eval will receive proper decommitments and be sure that the decommitted labels are valid.

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, and Eval will not learn Gen's input according the location. 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 know the semantics of her input labels and can verify that the retrieved labels actually match her input.

In particular, in $\Theta^{(j)}$, the commitment pairs 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 $\Omega^{(j)}$ and $\Phi^{(j)}$, the commitment to b-label of the i-th wire is the $(2 \cdot i + b)$-th entry.

[d] Decommitments $Y^{(j)}$ are meant to let Eval retrieve Eval's input labels from commitments $\Omega^{(j)}$ she received in Step 5b.

[e] Decommitments $X_1^{(j)}$ are meant to let Eval retrieve Gen's input labels from commitments $\Gamma^{(j)}$ she received in Step 3, and decommitments $X_2^{(j)}$ are meant to let Eval retrieve Gen's input labels from commitments $\Theta^{(j)}$ she received in Step 5b.

F Security Definition

Here we summarize a well-accepted definition for secure two-party computation presented in [Gol04] and based on [GMW87]. The security notion follows the "ideal/real" paradigm in which we compare what a malicious party can accomplish in the proposed protocol versus what a malicious party can accomplish in an idealized setting in which a trusted party aids in the computation. Loosely speaking, a protocol is secure if any adversary interacting in the real protocol (where no trusted third party exists) can do no more harm than if it was involved in the ideal computation described as follows.

Malicious Parties: 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, 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-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 $(f_1, f_2) = f(x', y')$. 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. An honest party always sends 1 to O if queried after receiving an input, and always outputs the message received from O. The variable $\mathbf{Ideal}_{f,\bar{S}}(x, y)$ represents the output of $S_1(x)$ and $S_2(y)$ from the above experiment. 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 strict probabilistic polynomial-time machines is executed without any oracle. In particular, if $\bar{P} = (P_1, P_2)$ is a pair of non-uniform strict 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. The variable $\mathbf{Real}_{\Pi,\bar{P}}(x, y)$ represents the final output of $P_1(x)$ and $P_2(y)$. Again, if at least one P_i is the same as the honest protocol Π_i, then $\bar{P}$ is called *admissible.*

Definition 3 (Secure two-party computation). *A protocol Π is said to* securely compute *function*

$$f : \{0,1\}^n \times \{0,1\}^n \mapsto \{0,1\}^* \times \{0,1\}^*$$

in the presence of malicious adversaries if for every pair of admissible non-uniform strict 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

$$\left\{\mathbf{Real}_{\Pi}(P_1(x), P_2(y), 1^k)\right\}_{x,y\in\{0,1\}^n, k\in\mathbb{N}} \approx_c \left\{\mathbf{Ideal}_{f}(S_1(x), S_2(y), 1^k)\right\}_{x,y\in\{0,1\}^n, k\in\mathbb{N}}$$

where $\approx_c$ denotes computational indistinguishability.

G Security Proof for Theorem 3.2

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 3.2, we need to construct two non-uniform probabilistic expected polynomial-time simulators S_1 and S_2 against P_1^* and P_2^*, respectively.

G.1 Malicious generator P_1^*

Intuition: S_1 first randomly picks a k-probe-resistant M and a fake input y'. Then S_1 finds a pre-image $\bar{y}'$ such that $M \cdot \bar{y}' = y'$. This $\bar{y}'$ is used as input to EVAL's input OTs. The OT's receiver security ensures that P_1^* cannot distinguish S_1 on input $\bar{y}'$ in the ideal model from P_2 on input $\bar{y}$ in the real model.

Next, for the circuit OTs, S_1 invokes the OT's simulator (the existence of which is guaranteed by OT's security) to extract both inputs from P_1^*, randomness $\rho^{(j)}$ and the decommitments $X^{(j)}$ to GEN's input keys. A garbled circuit is *bad* if the retrieved randomness cannot be used to regenerate the provided commitments and garbled circuit. S_1 aborts if more than $\sigma/5$ of the garbled circuits are bad. This step is indistinguishable from the real model due to the cut-and-choose technique. In particular, if there are more than $\sigma/5$ bad circuits, P_2 in the real model would abort with high probability too since the probability that none of them is chosen to be check circuit is negligible.

If S_1 does not abort, it learns the randomness of at least $4\sigma/5$ good garbled circuits. Note that after P_1^* passes the circuit checking, S_1 also learns the decommitments $X^{(j)}$ of the $2\sigma/5$ evaluation circuits. In other words, S_1 learns the randomness of at least $\sigma/5$ evaluation circuits, which are good circuits too. The binding property of the commitments ensures that S_1 learns the private inputs P_1^* provided to these good evaluation circuits. S_1 aborts if these private inputs are inconsistent. This step is indistinguishable from the real model due to GEN's input consistency check. In particular, the input consistency check ensures that the probability of good evaluation circuits having inconsistent inputs is negligible.

Let P_1^*'s private input extracted from above be $\bar{x}' = x'||r'||e'$. S_1 sends x' to the external trusted party and gets $f_1(x', y)$ in return, where $y = M \cdot \bar{y}$. If S_1 in the ideal model (resp. P_2 in the real model) has come to this far, with high probability, P_1^* must have provided majority good evaluation circuits with valid input labels corresponding to P_1^*'s input $\bar{x}'$ and S_2's input $\bar{y}'$ (resp. P_2's input $\bar{y}$). As a result, the majority of P_2's evaluation outputs are exactly $f_1(x', y) \oplus e'$. This implies that P_2^* cannot distinguish S_1 on input $\bar{y}'$ but providing $f_1(x', y) \oplus e'$ (from the external party) versus P_1 on input $\bar{y}$ and provding the evaluation output $f_1(x', y) \oplus e'$. Moreover, the k-probe-resistant matrix also helps to support the indistinguishability between S_1 on protocol input y' and P_1 on protocol input y. In particular, the k-probe-resistant matrix ensures that the difference between the probability that S_1 on fake input y' aborts (due to selective failure) and the probability that P_2 on real input y aborts is negligible.

Finally, with P_1^*'s encrypted output $f_1(x', y) \oplus e'$ and its corresponding random keys (since S_1 knows the randomness of many circuits), S_1 is able to do the GEN's output authenticity proof as P_1 does.

Common Input: a security parameter 1^k, a statistical security parameter 1^σ, a symmetric encryption scheme $(\mathsf{enc}, \mathsf{dec})$ with semantic security, a commitment scheme com, and objective function $f : (x, y) \mapsto (f_1, f_2)$.
Private Input: P_1^* has private input x but uses x' (which will be extracted) in the protocol instead.
Private Output: P_1^* receives output $f_1(x', y)$.

1. S_2 randomly picks a k-probe-resistant matrix M and fake input y'. For the rest of the protocol, S_2's input refers to $\bar{y}'$ such that $M \cdot \bar{y}' = y'$.
2. **(Pick the randomness)** S_2 does nothing in this step.
3. **(Commit to Gen's input)** S_2 receives commitments $\{\Gamma^{(j)}\}_{j\in[\sigma]}$ from P_1^*.
4. **(Determine the objective circuit)** S_2 sends M to P_1^*, and then both parties jointly run a coin flipping protocol to randomly pick $H \in \{0,1\}^{k\times m_1}$. If P_1^* aborts, S_1 aborts; otherwise, both parties now have determined the objective circuit C that computes $g : (\bar{x}, \bar{y}') \mapsto (\perp, (h, c, g_2))$, where $h = H \cdot \bar{x}$, $g_1 = f_1(x, M \cdot \bar{y}')$, $c = g_1 \oplus e$, and $g_2 = f_2(x, M \cdot \bar{y}')$ for some $\bar{x} = x||e||r$. (Since P_1^* has not entered her input yet, $\bar{x}$ is unclear at this point.)
5. **(Commit to input/output label pairs)** S_1 receives commitments $\{\Theta^{(j)}, \Omega^{(j)}, \Phi^{(j)}\}_{j\in[\sigma]}$ from P_1^*.
6. For each of the m_2 instances of input OTs, S_1 uses her fake input $\bar{y}'$. In contrast, for each of σ instances of circuit OTs, S_1 invokes OT's simulator S_1^{OT} in order to extract both of P_1^*' input, that is, randomness $\rho^{(j)}$ and decommitments $X^{(j)}$. S_1 aborts if simulator S_1^{OT} aborts.
7. **(Circuit Garbling)** S_1 waits for P_1^* to complete circuit garbling.
8. **(Circuit Checking)** S_1 receives garbled circuit $\{G(C)^{(j)}\}_{j\in[\sigma]}$ from P_1^*. S_1 randomly picks $S \subset [\sigma]$ such that $|S| = 2\sigma/5$. Let $s \in \{0,1\}^\sigma$ such that $s_j = 1$ if $j \in S$; and $s_j = 0$ otherwise.
   (Identify Bad Circuits) For each $j \in [\sigma]$, $G(C)^{(j)}$ is bad if $\rho^{(j)}$ extracted in Step 6 cannot regenerate $\{\Theta^{(j)}, \Omega^{(j)}, \Phi^{(j)}\}$ received in Step 5 and reconstruct $G(C)^{(j)}$? S_1 aborts if there are more than $\sigma/5$ bad circuits.

   (Evaluation Circuits) for each $j \in S$, S_1 checks:
   (a) if $X^{(j)} = X_1^{(j)} \cup X_2^{(j)}$ received in Step 6 successfully decommits $\Gamma^{(j)}$ received in Step 3 and half of $\Theta^{(j)}$ received in Step 5? and if the decommitted labels match? In particular,
      i. if the i-th entry of $X_1^{(j)}$ successfully decommits the i-th entry of $\Gamma^{(j)}$?
      ii. if the i-th entry of $X_2^{(j)}$ successfully decommits the $(2 \cdot i + \bar{x}_i \oplus \pi_i^{(j)})$-th entry of $\Theta^{(j)}$?
      iii. if the i-th decommitted labels from the above two steps coincide?
   (b) if $Y^{(j)}$ received in Step 6 successfully decommits half of the commitments in $\Omega^{(j)}$? In particular, if the i-th entry of $Y^{(j)}$ successfully decommits the $(2 \cdot i + \bar{y}_i)$-th entry of $\Omega^{(j)}$?

   S_1 aborts immediately as long as a failure occurs.
9. **(Circuit Evaluating)** S_1 does nothing in this step.
10. **(Majority Operation)** S_1 has now completely opened at least $4\sigma/5$ good circuits (denoted by set U), and S_1 also receives P_1^*'s input labels to $2\sigma/5$ evaluation circuits (denoted by set V). In other words, there are at least $\sigma/5$ circuits that S_1 has complete understanding and also has P_1^*'s input labels. S_1 translates those random labels back to their semantics. S_1 aborts if any of the translations fails or there are more than one translation results. If S_1 does not abort, S_1 invokes the external trusted party with input x', where $\bar{x}' = x'||e'||r'$ is the unique translation result, and gets $f_1(x', y)$.
11. **(Gen's Output Authenticity Proof)** Let $Z^{(j)}$ be the labels corresponding to P_1^*'s output wires of value $f_1(x', y) \oplus e'$ in circuit $G(C)^{(j)}$ for some $j \in S$. S_1 randomly picks $m \in U \cap V$, and use $(m, Z^{(m)})$ as its private input in the output authenticity proof presented in Appendix A.
    S_1 aborts if P_1^* rejects the proof. Otherwise, S_1 outputs whatever P_1^* outputs.

Fig. 1: The Simulator for an Arbitrary Malicious Generator P_1^*

Lemma G.6 (Security against a Malicious Generator). *Assume that the $\binom{2}{1}$-OT protocol is secure in the presence of malicious adversaries, and there exist a commitment scheme and a family of pseudo-random functions. For any function $f(x,y) \mapsto (f_1, f_2)$, and any non-uniform probabilistic strict polynomial-time machine P_1^*, simulator S_1 presented in Fig. 1 is a non-uniform probabilistic expected polynomial-time machine satisfying that*

$$\left\{\mathbf{Real}_{(\text{GEN},\text{EVAL})}(P_1^*(x), P_2(y), 1^k)\right\}_{x,y\in\{0,1\}^n,k\in\mathbb{N}} \approx_c \left\{\mathbf{Ideal}_f(S_1^{P_1^*}(x), S_2(y), 1^k)\right\}_{x,y\in\{0,1\}^n,k\in\mathbb{N}}.$$

Proof. Recall that in this case, P_2 always follows the Main protocol faithfully, and S_2 always provides y to the trusted party and responds with 1 after getting its result (so that the trusted party will give S_1 its output). To prove this lemma, consider the following hybrid experiments and lemmas:

$\mathbf{Hybrid}_1(P_1^*(x), P_2(y), 1^k)$: This is the same as $\mathbf{Real}_{(\text{GEN},\text{EVAL})}(P_1^*(x), P_2(y), 1^k)$ except that during the circuit OTs, this experiment invokes simulator S_1^{OT} (the existence of which is guaranteed by OT's security for the receiver) to extract both P_1^* inputs, randomness $\rho^{(j)}$ and decommitments $X^{(j)}$.

Lemma G.7. $\left\{\mathbf{Real}_{(\text{GEN},\text{EVAL})}(P_1^*(x), P_2(y), 1^k)\right\}_{x,y\in\{0,1\}^n,k\in\mathbb{N}} \approx_c \left\{\mathbf{Hybrid}_1(P_1^*(x), P_2(y), 1^k)\right\}_{x,y\in\{0,1\}^n,k\in\mathbb{N}}$

Proof. Intuitively, this lemma follows the OT's security for the receiver nicely. Formally, suppose there exists a non-uniform probabilistic polynomial-time machine D such that for infinitely many k, D can efficiently distinguish **Real** from $\mathbf{Hybrid}_1$. We will show another non-uniform probabilistic polynomial-time machine A that breaks the OT's security for the receiver with black-box access to D. More specifically, A internally runs the Main protocol with D such that D plays as P_1 and A plays as P_2. When it comes to the circuit OTs, A simply forwards the internal messages from D to the external OT receiver, and vise versa. After the circuit OTs, A and D continue to complete the Main protocol. In the end, A outputs whatever D outputs. Since if the external OT receiver is from the ideal model, D sees $\mathbf{Hybrid}_1$, and if the OT receiver is from the real model, D sees **Real**. In other words, the assumption that D can efficiently distinguish **Real** from $\mathbf{Hybrid}_1$ implies that A could efficiently distinguish S_1^{OT} and an OT's receiver from the real model, which contradicts the OT's security. □

$\mathbf{Hybrid}_2(P_1^*(x), P_2(y), 1^k)$: This is the same as $\mathbf{Hybrid}_1$ except this experiment aborts if more than $\sigma/5$ garbled circuits are *bad*. A garbled circuit $G(C)^{(j)}$ is bad if randomness $\rho^{(j)}$ cannot properly regenerate $\{\Theta^{(j)}, \Omega^{(j)}, \Phi^{(j)}\}$ and $G(C)^{(j)}$, where $\{\Theta^{(j)}, \Omega^{(j)}, \Phi^{(j)}\}$ are the commitments to the pair of keys assigned to GEN's input wires, EVAL's input wires, and GEN's output wires.

Lemma G.8. $\left\{\mathbf{Hybrid}_1(P_1^*(x), P_2(y), 1^k)\right\}_{x,y\in\{0,1\}^n,k\in\mathbb{N}} \approx_s \left\{\mathbf{Hybrid}_2(P_1^*(x), P_2(y), 1^k)\right\}_{x,y\in\{0,1\}^n,k\in\mathbb{N}}$

Proof. The only case that the two experiments differ is when there are more than $\sigma/5$ bad circuits, that is, when $\mathbf{Hybrid}_2$ aborts but $\mathbf{Hybrid}_1$ does not. We argue that this happens with negligible probability. Indeed, when there are more than $\sigma/5$ bad circuits, the probability that none of them is chosen is

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

Therefore, the probability the $\mathbf{Hybrid}_1$ and $\mathbf{Hybrid}_2$ differ is negligible. □

Hybrid$_3(P_1^*(x), P_2(y), 1^k)$**:** In **Hybrid**$_2$, there are at least $4\sigma/5$ good circuits (denoted by U), whose randomness is successfully retrieved and can regenerate the circuit. Once P_1^* passes the circuit checking, this experiment also learns decommitments $X^{(j)}$ for the $2\sigma/5$ evaluation circuits (denoted by V). **Hybrid**$_3$ is the same as **Hybrid**$_2$ except that this experiment aborts if for any $j \in U \cap V$, decommitments $X^{(j)}$ cannot reveal P_1^*'s private input $\bar{x}'^{(j)}$ to evaluation circuit $G(C)^{(j)}$.

Lemma G.9. $\left\{\textbf{Hybrid}_2(P_1^*(x), P_2(y), 1^k)\right\}_{x,y\in\{0,1\}^n, k\in\mathbb{N}} \approx_c \left\{\textbf{Hybrid}_3(P_1^*(x), P_2(y), 1^k)\right\}_{x,y\in\{0,1\}^n, k\in\mathbb{N}}$.

Proof. Suppose $\bar{x}_i'^{(j)}$ is unknown for some $j \in U \cap V$ and some $i \in [m_1]$. Let us consider the commitment $\Theta_i^{(j)}$ to the label corresponding to this bit. Since $j \in U$, commitment $\Theta_i^{(j)}$ has been checked according to randomeness $\rho^{(j)}$. Let d_1 denote the decommitment from $\rho^{(j)}$. Also, since $j \in V$, the same commitment has been checked by decommitments $X^{(j)}$. Let d_2 denote the decommitment from $X^{(j)}$. The fact that $\bar{x}_i'^{(j)}$ is unknows implies that $d_1 \neq d_2$. In other words, the fact that decommmitment $X^{(j)}$ cannot reveal P_1^*'s input $\bar{x}'^{(j)}$ to circuit $G(C)^{(j)}$ implies that P_1^* can break the binding property of the commitment scheme, which only happens with negligible probabilisty. So this lemma holds. □

Hybrid$_4(P_1^*(x), P_2(y), 1^k)$**:** This is the same as **Hybrid**$_3$ except that this experiment aborts if the extracted P_1^*'s private inputs $\{\bar{x}'^{(j)}\}_{j\in U\cap V}$ are inconsistent.

Lemma G.10. $\left\{\textbf{Hybrid}_3(P_1^*(x), P_2(y), 1^k)\right\}_{x,y\in\{0,1\}^n, k\in\mathbb{N}} \approx_s \left\{\textbf{Hybrid}_4(P_1^*(x), P_2(y), 1^k)\right\}_{x,y\in\{0,1\}^n, k\in\mathbb{N}}$.

Proof. At a high level, this lemma results from GEN's input consistency check, that is, the 2-universal hash design. Let E denote the event that **Hybrid**$_4$ aborts but **Hybrid**$_3$ does not. Observe that E happens when "P_1^* provides inconsistent inputs to good circuits." Note that in those good circuits, the hash circuits are good (checked via randomness $\rho^{(j)}$) and P_1^*'s input keys are valid (check via decommitments $X^{(j)}$). So the hashes are correctly computed. The fact that **Hybrid**$_3$ does not abort implies that E happens when inconsistent $\{\bar{x}'^{(j)}\}_{j\in U\cap V}$ result in consistent $\{H\cdot\bar{x}'^{(j)}\}_{j\in U\cap V}$. There are only two possbilities:

P_1^* chose his input adaptively: Given h, P_1^* could easily come up with $\bar{x}'$ such that $H\cdot\bar{x}' = h$. However, the labels corresponding to $\bar{x}'^{(j)}$ are committed to prior to when H is chosen. If P_1^* could choose $\bar{x}'$ adaptively, he could break the binding property of the commitment scheme, which only happens with negligible probability.

P_1^* found a collision: Since $\bar{x}'^{(j)}$ is committed to prior to when H is randomly chosen, the definition of the 2-universal hashes implies that this case only happens with probability 2^{-k}.

The above analysis shows that either case implies that $\Pr(E)$ is negligible. So this lemma holds. □

Hybrid$_5(P_1^*(x), P_2(y), 1^k)$**:** Let $\bar{x}' = x'||r'||e'$ be P_1^*'s input that is extracted from $\{G(C)^{(j)}\}_{j\in U\cap V}$. **Hybrid**$_5$ is the same as **Hybrid**$_4$ except that this experiment invokes the external trusted third party with input x'. Since S_2 is honest, the trusted party always returns $f(x', y)$. This experiment aborts if $f_1(x', y) \oplus e'$ does not coincide with the majority of the evaluation outputs

Lemma G.11. $\left\{\textbf{Hybrid}_4(P_1^*(x), P_2(y), 1^k)\right\}_{x,y\in\{0,1\}^n, k\in\mathbb{N}} \approx_s \left\{\textbf{Hybrid}_5(P_1^*(x), P_2(y), 1^k)\right\}_{x,y\in\{0,1\}^n, k\in\mathbb{N}}$.

Proof. This is a straightforward result because more than half (at least $\sigma/5$) of the $2\sigma/5$ evaluation circuits are good, which means that they are honestly constructed and their input keys are valid. Combined with the fact that P_1^*'s inputs $\bar{x}' = x'||r'||e'$ to these good evaluation circuits are consistent, we conclude that the majority of the outputs from garbled circuit evaluation is exactly $f(x', y) \oplus e'$. □

Hybrid$_6(P_1^*(x), P_2(y), 1^k)$**:** This is the same as **Hybrid**$_5$ except that this experiment randomly picks one of the good circuits, $G(C)^{(j)}$ for some $j \in U \cap V$, and uses its output keys corresponding to output $f(x', y) \oplus e'$ to conduct the GEN's output authenticity proof at the end.

Lemma G.12. $\left\{\textbf{Hybrid}_5(P_1^*(x), P_2(y), 1^k)\right\}_{x,y\in\{0,1\}^n, k\in\mathbb{N}} \approx_c \left\{\textbf{Hybrid}_6(P_1^*(x), P_2(y), 1^k)\right\}_{x,y\in\{0,1\}^n, k\in\mathbb{N}}$

Proof. At a high level, this Lemma follows the witness-indistinguishability of the GEN's output authentication proof. Formally, for contradiction proposes, suppose there exists a non-uniform probabilistic polynomial-time machine D that can distinguish **Hybrid**$_5$ from **Hybrid**$_6$ with non-negligible probability. We will construct a non-uniform probabilistic polynomial-time machine A that breaks the witness-indistinguishability of GEN's output authenticity proof by using black-box access to D. More specifically, A internally runs **Hybrid**$_5$ with D such that D runs as P_1^* and A runs as the experiment. When it comes to GEN's output proof, A forwards the statement from D to the external prover. A additionally sends witness $w^{(j)}$ to the external prover, where $w^{(j)}$ includes output $f_1(x', y)\oplus e'$, circuit index j, and the output keys corresponding to the output. A then receives the proof from the external prover and forwards it to D internally. Finally, A outputs whatever D outputs. If D distinguishes **Hybrid**$_5$ from **Hybrid**$_6$ with non-negligible probability, A breaks the witness-indistinguishability property of GEN's output proof. □

Hybrid$_7(P_1^*(x), P_2(y), 1^k)$**:** This is the same as **Hybrid**$_6$ except that this experiment does not evaluate any of the garbled circuits.

Lemma G.13. $\left\{\textbf{Hybrid}_6(P_1^*(x), P_2(y), 1^k)\right\}_{x,y\in\{0,1\}^n, k\in\mathbb{N}} \approx_s \left\{\textbf{Hybrid}_7(P_1^*(x), P_2(y), 1^k)\right\}_{x,y\in\{0,1\}^n, k\in\mathbb{N}}$

Proof. The only concern here is that **Hybrid**$_6$ might abort due to not evaluating circuits successfully (no output appears more than $\sigma/5$ times), whereas **Hybrid**$_7$ does not evaluate the circuits, and thus, never abort for the same reason. However, similarly to Lemma G.11, the majority of the evaluation circuits are good and have a consistent input from P_1^*. This concern happens only with negligible probability. □

Hybrid$_8(P_1^*(x), P_2(y), 1^k)$**:** This is the same as **Hybrid**$_7$ except that this experiment picks a random y' and computes $\bar{y}'$ such that $M \cdot \bar{y}' = y'$. Then this experiment uses $\bar{y}'$ as input to EVAL's input OTs.

Lemma G.14. $\left\{\textbf{Hybrid}_7(P_1^*(x), P_2(y), 1^k)\right\}_{x,y\in\{0,1\}^n, k\in\mathbb{N}} \approx_c \left\{\textbf{Hybrid}_8(P_1^*(x), P_2(y), 1^k)\right\}_{x,y\in\{0,1\}^n, k\in\mathbb{N}}$

Proof. Intuitively, this lemma comes from OT's security for the receiver and the effect of the k-probe-resistant matrix. Formally, let E be the event that **Hybrid**$_7$ on input y and **Hybrid**$_8$ on input y' respond differently in EVAL's input OTs, and let F be the event that **Hybrid**$_7$ and **Hybrid**$_8$ differ while checking the decommitments to the labels corresponding to EVAL's input. Recall that **Hybrid**$_7$'s input equals y, whereas its input to EVAL's input OT equals $\bar{y}$ such that $M \cdot \bar{y} = y$. Similarly, **Hybrid**$_8$'s input equals y', whereas its input to EVAL's input OT equals $\bar{y}$ such that $M \cdot \bar{y}' = y'$.

Claim. $\Pr[E]$ is negligible.

Proof. Suppose not. Then there must exist a non-uniform probabilistic polynomial-time machine D such that for infinitely many k, D distinguishes $\mathbf{Hybrid}_7$ from $\mathbf{Hybrid}_8$ with probability at least $1/p(k)$ for some polynomial $p(\cdot)$. We show that there must exist a non-uniform probabilistic polynomial-time machine A with black-box access to D that breaks the OT's security for the receiver. In particular, let $\mathbf{Hybrid}_{7,i}$ be the same as $\mathbf{Hybrid}_7$ except that for the first i instances of EVAL's input OT, this experiment uses input $\bar{y}_i'$ instead. Note that $\mathbf{Hybrid}_{7,0} = \mathbf{Hybrid}_7$ and $\mathbf{Hybrid}_{7,m_2} = \mathbf{Hybrid}_8$. Since D distinguishes $\mathbf{Hybrid}_7$ from $\mathbf{Hybrid}_8$ with probability at least $1/p(k)$, by averaging, there exists some i such that D distinguishes $\mathbf{Hybrid}_{7,i}$ and $\mathbf{Hybrid}_{7,i+1}$ with probability at least $1/(m_2 \cdot p(x))$. Let A work as follows: A internally runs $\mathbf{Hybrid}_{7,i}$ with D such that D plays as GEN and A plays as EVAL except upon the i-th intance of EVAL's input OT, A forwards D's message to an external OT receiver, and vice versa. After this OT, D and A continue to finish $\mathbf{Hybrid}_{7,i}$. In the end, A outputs whatever D outputs. Note that $\mathbf{Hybrid}_{7,i}$ and $\mathbf{Hybrid}_{7,i+1}$ differ implies that $\bar{y}_i \neq \bar{y}_i'$. Without loss of generality, let $(\bar{y}_i, \bar{y}_i') = (0,1)$. We know that if the external receiver has input 0, D sees $\mathbf{Hybrid}_{7,i}$; and if the external receiver has input 1, D sees $\mathbf{Hybrid}_{7,i+1}$. Therefore, the assumption that D distinguishes $\mathbf{Hybrid}_{7,i}$ from $\mathbf{Hybrid}_{7,i+1}$ wth probability at least $1/(m_2 \cdot p(k))$ implies that A distinguishes the external receiver on input 0 versus on input 1 with probability at least $1/(m_2 \cdot p(k))$, which contradicts the assumption of OT's security. □

Claim. $\Pr[F]$ is negligible.

Proof. Recall that $\mathbf{Hybrid}_7$'s input eqauls y and its input to EVAL's input OT equals $\bar{y}$ such that $M \cdot \bar{y} = y$, wherea $\mathbf{Hybrid}_8$'s input equals y' and its input to EVAL's input OT equals $\bar{y}'$ such that $M \cdot \bar{y}' = y'$. Intuitively, this claim is true because M is k-probe-resistant. Whatever P_1^* learns from $\bar{y}$ (or $\bar{y}'$) due to selective failure reveals little information about y (or y'). A bit more formally, this claim comes from a direct application of Lemma 10 in [LP07] slightly rephrased below:

Lemma G.15 (Lemma 10. of [LP07]). *If M is k-probe-resistant, then for any two different inputs y and y' of P_2 for objective function f, the difference between the probability that P_2 aborts the protocol as a result of corrupt OT values when its input is y and when its input is y' is at most 2^{-k+1}.*

□

This lemma holds due to the above two claims. □

Finally, since every step finishes in expected polynomial time, including σ invocations of simulator S_1^{OT}, S_1 runs in expected polynomial time. Observe that $\mathbf{Hybrid}_8(P_1^*(x), P_2(y), 1^k)$ and $\mathbf{Ideal}_f(S_1^{P_1^*}(x), S_2(y), 1^k)$ are syntactically similar. In particular, P_2's input is no longer used. By Lemma G.7–G.14, we prove the security/simulatability of the Main protocol when GEN is malicious. □

G.2 Malicious Evaluator P_2^* (Proof Sketch)

Simulator S_2 honestly follows the Main protocol all the way until the step of OTs except that S_2 picks a random $\bar{x}' \in \{0,1\}^{m_1}$ at the beginning and uses this fake input as input. In particular, S_2 commits to fake input $\bar{x}'$ in Step 3 by committing to the corresponding labels. The commitments are denoted

by $\{\Gamma^{(j)}\}_{j\in[\sigma]}$. In the step of OTs, S_2 invokes OT's simulator S_2^{OT} (whose existence is guaranteed by OT's security) to extract P_2^*'s input to OTs, that is, her private input $\bar{y}'$ to EVAL's input OTs and the choice string s' to the circuit OTs. Recall that s' determines the check circuits and evaluation circuits. Next, S_2 externally invokes the trusted third party with input $y' = M \cdot \bar{y}'$ and gets $f_2(x, y')$ in return. Note that M is the k-probe-resistant matrix received from P_2^* before OTs. After this, if $s'_j = 0$, the j-th garbled circuit is a check circuit and needs to be honestly constructed according to objective circuit C; otherwise, the j-th garbled circuit is an evaluation circuit and is constructed in a way that it always outputs $(h', c', f_2(x, y'))$, where h' and c' are randomly picked by S_2. From now on, S_2 follows the Main protocol faithfully. Finally, if S_2 accepts in the step of GEN's Output Authenticity Proof, it sends 1 to the external oracle so that S_1 gets $f_1(x, y')$; otherwise, S_2 sends 0 to the external oracle so that S_1 gets $\perp$.

Here we argue at a high level that P_2^* cannot distinguish S_2 using fake input $\bar{x}'$ in the ideal model versus P_1 using real input $\bar{x}$ in the real model.

For check circuits, the only difference between S_2 in the ideal model and P_1 in the real model is the committed message in $\Gamma^{(j)}$. Note that this commitment is never opened for check circuits. Therefore, if P_1^* is able to distinguish S_2 committing to fake input $\bar{x}'$ from P_1 committing to real input $\bar{x}$ in any check circuit, P_1^* is able to break the hiding property of the commitment scheme.

For evaluation circuits, the information P_1^* learned related the other party's input is the location of the decommitted labels within each pair of commitments and the evaluation output:

1. Recall that the pairs of commitments to the labels assigned to GEN's input wires are randomly swapped by permutation bits $\{\pi_i^{(j)}\}_{i\in[m_1],j\in[\sigma]}$. In particular, if $\pi_i^{(j)} = 0$, the pair of commitments are in the format of $(\mathsf{com}(W_{i,1}^{(j)}), \mathsf{com}(W_{i,0}^{(j)}))$; otherwise, they are swapped and in the format of $(\mathsf{com}(W_{i,1}^{(j)}), \mathsf{com}(W_{i,0}^{(j)}))$. This random swapping implies that the location learned in the ideal model is $\bar{x}' \oplus \pi'^{(j)}$, where $\pi'^{(j)} = \pi'^{(j)}_1 || \pi'^{(j)}_2 || \ldots || \pi'^{(j)}_{m_1}$, while that learned in the real model is $\bar{x} \oplus \pi^{(j)}$, where $\pi^{(j)} = \pi_1^{(j)} || \pi_2^{(j)} || \ldots \pi_{m_1}^{(j)}$. Since $\pi'^{(j)}$ and $\pi^{(j)}$ are independently chosen from the uniform distribution, the location information is statistically indistinguishable.
2. The other message that might give S_2 away is the evaluation output. First, we claim that P_2^* always gets consistent output among different garbled circuits. Indeed, in the real model, since P_1 is assumed to be honest, the outputs from the evaluation circuits are identical, and similarly, in the ideal model, S_2 also generates evaluation circuits that have a fixed output. So it remains to argue that (h, c, f_2) in the real model is indistinguishable from (h', c', f_2') in the ideal model. Intuitively, h and h' are indistinguishable due to the hiding property of our 2-universal hash scheme, c and c' are indistinguishable due to the perfect secrecy of the one-time pad encryption, and f_2 and f_2' are indistinguishable due to the simulation security of OTs.

G.3 Malicious Evaluator P_2^* (The Simulator S_2)

H Optimization Communication Overhead

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.

Common Input: a security parameter 1^k, a statistical security parameter 1^σ, a symmetric encryption scheme $(\mathsf{enc}, \mathsf{dec})$ with semantic security, a commitment scheme com, and objective function $f : (x, y) \mapsto (f_1, f_2)$.
Private Input: Both S_2 and P_2^* know private input y, but P_2^* uses y' as input in the protocol.
Private Output: P_2^* receives output $f_2(x, y')$.

1. S_2 randomly picks $\bar{x}' \in \{0,1\}^{m_1}$. From now on, S_2's input refers to $\bar{x}' = x'||e'||r' = \bar{x}'_1\bar{x}'_2\ldots\bar{x}'_{m_1}$.
2. **(Commit to randomness)** S_2 picks randomness $\{\rho^{(j)}\}_{j\in[\sigma]}$ and commits to them by sending $\{\mathsf{com}(\rho^{(j)})\}_{j\in[\sigma]}$ to P_2^*.
3. **(Commit to Gen's input)** S_2 uses $\rho^{(j)}$ to pick $(K_{i,0}^{(j)}, K_{i,1}^{(j)}, \pi_i^{(j)}) \in \{0,1\}^{2k+1}$ for $j \in [\sigma]$ and $i \in [m_1]$. 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 from now on.
   S_2 commits to her input by sending $\{\Gamma^{(j)}\}_{j\in[\sigma]}$ to P_2^*, where $\Gamma^{(j)} = \{\mathsf{com}(W_{i,\bar{x}'_i}^{(j)}; \gamma_i^{(j)})\}_{i\in[m_1]}$.
4. **(Determine the objective circuit)** S_2 receives M from P_2^*, and then both parties jointly run a coin flipping protocol to randomly pick $H \in \{0,1\}^{k\times m_1}$. If the coin flipping fails, S_2 aborts; otherwise, both parties now have determined the objective circuit C that computes $g : (\bar{x}', \bar{y}) \mapsto (\perp, (h, c, g_2))$, where $h = H \cdot \bar{x}'$, $g_1 = f_1(x', M \cdot \bar{y})$, $c = g_1 \oplus e'$, and $g_2 = f_2(x', M \cdot \bar{y})$ for some $\bar{y}$.[a]
5. **(Commit to input/output label pairs)** Let $\{w_i\}_{i\in[m_1]}$ be the circuit-input wires corresponding to S_2's input, $\{w_{m_1+i}\}_{i\in[m_2]}$ be those corresponding to P_2^*'s input, and $\{w_i\}_{i\in O_2}$ be the circuit-output wires corresponding to S_2's output (output c in particular). S_2 uses randomness $\rho^{(j)}$ to
   (a) pick $(K_{i,0}^{(j)}, K_{i,1}^{(j)}, \pi_i^{(j)}) \in \{0,1\}^{2k+1}$ for all but S_2's input wires and
   (b) commit to the label pairs assigned to S_2's input wires, P_2^*'s input wires, and S_2's output wires by sending $\{\Theta^{(j)}, \Omega^{(j)}, \Phi^{(j)}\}_{j\in[\sigma]}$ to P_2^*, where

$$\Theta^{(j)} = \{\mathsf{com}(W^{(j)}_{i,0\oplus\pi_i^{(j)}}; \theta_i^{(j)}), \mathsf{com}(W^{(j)}_{i,1\oplus\pi_i^{(j)}}; \theta_i^{(j)})\}_{i\in[m_1]},$$
$$\Omega^{(j)} = \{\mathsf{com}(W^{(j)}_{m_1+i,0}; \omega_i^{(j)}), \mathsf{com}(W^{(j)}_{m_1+i,1}; \omega_i^{(j)})\}_{i\in[m_2]},$$
$$\Phi^{(j)} = \{\mathsf{com}(W_{i,0}^{(j)}), \mathsf{com}(W_{i,1}^{(j)})\}_{i\in O_2}.$$

6. For each of the $(m_2 + \sigma)$ instances of $\binom{2}{1}$-OTs, S_2 instead invokes OT's simulator S_2^{OT} in order to extract P_2^*'s input $\bar{y}'$ to the input OTs and P_2^*'s input s' to the circuit OTs. Additionally,
   (a) **(P_2^*'s Input OTs)** upon the query $\bar{y}'_i$ from S_2^{OT}, S_2 responds with $\{(W^{(j)}_{m_1+i,\bar{y}'_i}, \omega_i^{(j)})\}_{j\in[\sigma]}$;
   (b) **(Circuit OTs)** upon the query s'_i from S_2^{OT}, S_2 responds with
      – decommitment $\mathsf{dc}(\rho^{(j)})$, if $s'_j = 0$; or
      – decommitments $X^{(j)} = (X_1^{(j)}, X_2^{(j)})$, where $X_1^{(j)} = \{(W^{(j)}_{i,\bar{x}'_i}, \gamma_i^{(j)})\}_{i\in[m_1]}$ and $X_2^{(j)} = \{(W^{(j)}_{i,\bar{x}'_i}, \theta_i^{(j)})\}_{i\in[m_1]}$, if $s'_j = 1$.

   S_2 outputs $\perp$ (abort) whenever S_2^{OT} outputs $\perp$ (abort).
7. **(Circuit Checking)** S_2 aborts whenever P_2^* aborts.
8. **(Circuit Evaluating)** S_2 waits for P_2^* until the circuit evaluation is completed.
9. **(Majority Operation)** S_2 aborts whenever P_2^* aborts.
10. **(Gen's Output Authenticity Proof)** S_2 and P_2^* jointly conduct the protocol presented in Appendix A. If P_2^* provides an accepting proof, S_2 sends 1 to the external trusted party so that S_1 receives output $f_1(x, y')$; otherwise, S_2 sends 0 to the external party, and S_1 gets $\perp$.

[a] Since P_2^* may change its input, $\bar{y}$ is not clear at this point.

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%.

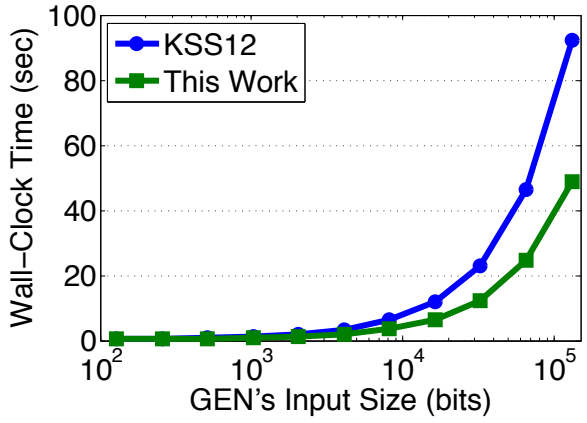
I Experimental Results

I.1 Performance of Our Proposed Techniques

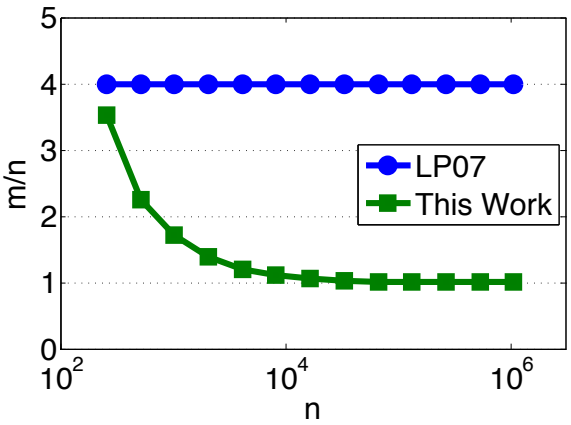
I.2 Performance of the Full Protocol Over Stampede

Improvements over Kreuter, shelat, and Shen [KsS12] include the employment of Intel SSE2 and AESNI instruction sets. While the former provides the 128-bit primitive operations, the latter provides the hardware optimized instructions computing AES encryption.

The EDT-4095 circuit has 5.9 billion gates (2.4 billion of which are non-XOR). It takes 9,042 seconds to run an instance with security parameter $k = 80$ (which requires 256 copies of the garbled circuit). In other words, our system on Stampede is able to handle 663,000 gates (or 265,000 non-XOR gates) per second. We observe that around 60% of the time is spent on communication. In particular, an instance of this computation requires 18TB of data to be exchanged.



(a) Performance of Our Consistency Check



(b) Size of Our k-Probe-Resistant Matrices

Input Size	2^7	2^8	2^9	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}
[KsS12]	0.57	0.65	0.83	1.18	1.88	3.29	6.22	11.87	23.10	46.30	92.11
This Work	0.52	0.57	0.66	0.84	1.22	1.96	3.45	6.45	12.37	24.68	48.83

Table 4: The performance of the generator's input consistency check with circuits of various input size. In particular, we pick circuits consisting of only copies of AES128, ranging from 1 to 1,024 copies. The security parameter is $k = 80$, which requires 256 copies of the garbled circuit. Each number comes from an average of 30 data points, and the 95% confidence interval is all within 1%.

The 1024-AES128 circuit (consisting of 1,024 copies of AES128) has 31.7 million gates (9.3 million of which are non-XOR). It takes 48.83 seconds to run an instance with security parameter $k = 80$. In this case, our system is handling 640,000 gates (or 191,000 non-XOR gates) per second.

The RSA256 circuit has 933 million gates (332 million of which are non-XOR). It takes 1,437 seconds to run an instance with security parameter $k = 80$. In this case, our system is handling 649,000 gates (or 232,000 non-XOR gates) per second.

I.3 Performance of the Full Protocol Over Amazon EC2

StarCluster [STA09] provides an easy, configurable way to set up a cluster on top of Amazon EC2 infrastructure. BetterYao [KS11] is the system based on which we implemented our protocol and built our system. In particular, we built on top of their $I + C$ approach.

n	2^8	2^9	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}
[LP07]	4	4	4	4	4	4	4	4	4	4	4	4	4
This Work	3.527	2.256	1.711	1.393	1.198	1.108	1.059	1.032	1.017	1.009	1.005	1.003	1.001

Table 5: The m/n ratio with security parameter $k = 80$ and n ranging from 2^8 to 2^{20}.

# of EC2 instances per party	1	2	4	8
Hourly Price (USD)	0.06	0.12	0.24	0.48
Time (sec/AES block)	321.0	154.0	78.6	40.6
AES blocks per hour	11.2	23.4	45.8	88.7
Amortized Price (USD/AES block)	0.0054	0.0051	0.0052	0.0054

Table 6: The price of running secure AES on Amazon EC2 with security parameter $k = 80$.