Fast Two-Party Secure Computation with Minimal Assumptions

[Extended Abstract]*

abhi shelat
University of Virginia
85 Engineer's Way
Charlottesville, Virginia, U.S.A
abhi@virginia.edu

Chih-hao Shen
University of Virginia
85 Engineer's Way
Charlottesville, Virginia, U.S.A
shench@virginia.edu

ABSTRACT

Almost all existing protocols for secure two-party computation require a specific hardness assumption, such as DDH, discrete logarithm, or a random oracle, even after assuming oracle access to the oblivious transfer functionality for their correctness and/or efficiency. We propose and implement a Yao-based protocol that is secure against malicious adversaries and enjoys the following benefits:

1. it requires the minimal hardness assumption, i.e., OTs;
2. it uses 10 rounds of communication plus OT rounds;
3. it has the optimal overhead complexity (for an approach that uses the circuit-level cut-and-choose technique); and
4. it is embarrassingly parallelizable in the sense that each circuit can be processed in a pipelined manner, and all circuits can be processed in parallel.

To achieve these properties, we describe novel solutions for the three main obstacles for achieving security against malicious adversaries in a cut-and-choose garbled-circuit protocol. We propose an efficient proof to establish the *generator's output authenticity*; we suggest the use of an auxiliary circuit that computes a hash to ensure *the generator's input consistency*; and we advance the performance of Pinkas and Lindell's state-of-the-art approach for handling *the selective failure attack*.

Not only does our protocol require weaker cryptographic assumptions, but our implementation of this protocol also demonstrates a several factor improvement over the best prior work which relies on specific number-theoretic assumptions. Thus, we show that performance does not require specific algebraic assumptions.

*A full version is at `http://eprint.iacr.org/2013/196`.

CCS'13, November 4–8, 2013, Berlin, Germany.

ACM 978-1-4503-2477-9/13/11 ...$15.00.
http://dx.doi.org/10.1145/2508859.2516698.

Categories and Subject Descriptors

F.0 [**Theory of Computation**]: General

Keywords

The Yao protocol, Malicious Model, Cut-and-Choose

1. INTRODUCTION

Secure Two-Party Computation (2PC) aims to allow two parties to collaborate in a way that achieves maximal privacy of their inputs with simultaneous guarantees of correctness of outputs. The correctness property guarantees that when both parties follow the protocol honestly, the protocol output is indeed the output of the objective function. The privacy property ensures that during the protocol execution, neither party can learn more than that derivable from her own input and output.

The first generic solution for Secure 2PC in the *honest-but-curious* model was proposed by Yao [26]. In this protocol, both parties agree on an objective function and its boolean circuit format (called the *objective circuit*) in advance[1]. One party (denoted by Gen) constructs a garbled version of this objective circuit, and the other party (denoted by Eval) obliviously evaluates this garbled circuit to compute an output. By oblivious evaluation we mean that Eval does not learn any intermediate value of the computation. This protocol satisfies the two security properties if both participants follow the protocol instructions honestly.

This basic protocol must be hardened to handle the situation in which either party arbitrarily deviates from the given instructions, e.g. by constructing a faulty circuit that purposely reveals Eval's private input.

The *circuit-level cut-and-choose* technique is one of the most efficient methods that enforce honest circuit garbling [11, 15, 16, 19, 23]. This technique instructs Gen to prepare multiple copies of the garbled circuit, each with independent randomness, and instructs Eval 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, Eval aborts and Gen is caught cheating; otherwise, Eval starts to evaluate the remaining circuits (called the *evaluation circuits*) as instructed in the Yao protocol. Finally, Eval takes the majority of the evaluation outputs as the final output. As

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

a result, a malicious GEN constructs either too many faulty circuits and gets caught, or only a few and does not influence the final output at all. This approach is efficient because it can naturally be run in parallel.

Besides the threat of faulty circuits, there remain three other subtle but equally critical security issues that need to be addressed when dealing with malicious adversaries. The first two in fact result from the use of multiple garbled circuits (as instructed by the circuit-level cut-and-choose technique): *two-output function handling* and GEN*'s input consistency*. The third issue is also known as the *selective failure attack*. Prior solution to these three concerns either requires specific hardness assumptions or introduces large overheads in communication and computation.

1.1 Contributions

Our main contribution is to construct an *optimal* protocol in the circuit-level cut-and-choose-based category such that (1) it requires minimal hardness assumptions, namely, an oblivious transfer (OT) secure in the presence of malicious adversaries; (2) it introduces little computational and communicational overhead to solve the above three security issues. In particular, its complexity is linear (in terms of the security parameter) in the original Yao protocol, which is the best a circuit-level cut-and-choose-based solution could ever achieve. In other words, we show that *malicious security comes almost for free* both in terms of required hardness assumptions and various protocol performance metrics. Let n denote the input/output size, k denote the security parameter, and σ denote the number of garbled circuits needed[2]. The contributions of this work are as follows:

1. We propose a novel witness-indistinguishable proof to ensure GEN's output authenticity. This proof requires only standard primitives (a symmetric encryption scheme and commitment scheme to be precise) and incurs about the same amount of overhead as garbling and evaluating GEN's output gates. In other words, it requires only $O(\sigma n)$ symmetric operations; prior approaches require $O(\sigma^2 n)$ symmetric operations [15, 22] or $O(\sigma n)$ symmetric operations plus $O(\sigma)$ algebraic operations [11].

2. We suggest the use of an auxiliary circuit to achieve GEN's input consistency. This auxiliary circuit computes the (universal) hash of GEN's input. By utilizing an XOR-homomorphic hash, we are able to evaluate the auxiliary circuit almost for free. Our solution is much more efficient than the prior works which need $O(\sigma^2 n)$ symmetric operations [15, 18] or $O(\sigma n)$ algebraic operations [23].

3. Most importantly, the above two techniques allow us to handle issues of two-output functions and GEN's input consistency while entirely avoiding algebraic operations, except for those needed by OTs. Lindell and Pinkas' [15] and Woodruff's [25] approaches are the only prior works, to the best of our knowledge, that enjoy this property. Nevertheless, our approach works in a more efficient manner as shown in Table 1.

4. Lindell and Pinkas suggested the use of a k-probe-resistant matrix (cf. Definition 5) to defend against the selective failure attack [15]. This solution has little overhead while combining with the free-XOR technique. However, it increases the number of OTs needed at the same time. While XOR gates can be computed almost for free, the OTs can not. While there exist extension techniques for OTs, not all variants of OTs can be efficiently extended. We therefore design a probabilistic algorithm based on Reed-Solomon code such that the number of OTs needed can be as low as 25% of that in the original Lindell and Pinkas' solution.

5. We propose an optimization technique that can save communication overhead by up to 60% (when 60% of all the garbled circuits are check circuits) with the price of a slight increase of computation overhead. We stress that our technique compares favorably with the Random Seed Checking technique [14]. In particular, our approach is compatible with the pipelining technique [8].

6. Based on an open-source system [14], we experimentally verify our theories by developing some of the above techniques. The integrated system can process 650,000+ gates per second on Stampede [24]. This is the fastest maliciously secure 2PC system reported.

1.2 Related Work

Prior work on converting the Yao protocol into a maliciously secure one based on the cut-and-choose technique [11, 14–16, 18, 22, 23, 25] reports several completely different approaches. Jarecki and Shmatikov suggested an approach that needs only a single copy of the garbled circuit, but this approach requires expensive zero-knowledge proof of correctness for every single gate [10] that also rely on specific RSA-based hardness assumptions. Nielson et al. reported a solution that uses efficient OTs to generate a pool of authenticated primitives [21]. With these primitives, both parties are able to securely evaluate a boolean circuit based on the generic Goldreich, Micali, and Wigderson protocol [5]. However, this protocol requires interactive communication for every AND gate, and therefore the number of rounds of communication depends on the circuit. While suitable for small circuits, large complicated circuits will require thousands of back-and-forth messages. Damgård et al. proposed a solution that uses somewhat homomorphic encryption to precompute a bunch of triples that are later used to securely compute an arithmetic circuit [1].

Among other approaches in the cut-and-choose-based category, our approach demonstrates superior performance in terms of the number of symmetric or algebraic 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 the computation assumptions needed, as shown in Table 2. While Jarecki and Shmatikov's approach requires hundreds of expensive algebraic operations per gate, ours does not need any (given oracle access to OTs) [10]. Although Neilson et al.'s approach favorably compares to our approach in terms of computation complexity, ours requires constant communication rounds while theirs needs rounds linear to the circuit depth [21]. At last, since Damgård et al.'s approach works

[2]Typically, the number of garbled circuits needed is linear to the security parameter, that is, $\sigma = O(k)$.

	Gen. Input Consist.		Gen. Output Auth.		Assumptions
	Symm. Op.	Alge. Op.	Symm. Op.	Alge. Op.	(besides OT)
[15]	$O(k^2n)$		$O(k^2n)$		OWF
[11]	$O(k^2n)$		$O(kn)$	$O(k)$	Discrete Log.
[16]	$O(kn)$	$O(kn)$	not mentioned		Decisional Diffie-Hellman
[23]	$O(kn)$	$O(kn)$	$O(kn)$	$O(kn)$	Discrete Log.
[14]	$O(kn)$	$O(kn)$	$O(kn)$	$O(k)$	Discrete Log.
This Work	$O(kn)$		$O(kn)$		OWF

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

with arithmetic circuits, it is incomparable to our work and thus omitted in the table [1]

	Symm. Op.	Alge. Op.	Rounds	Assumptions
[10]	$O(C)$	$O(C)$	$O(1)$	DCR + RSA
[20]	$O(\frac{k}{\lg C}C)$	$O(\frac{k}{\lg C}C)$	$O(1)$	Discrete Log
[21]	$O(\frac{k}{\lg C}C)$		$O(D_f)$	Random Oracle
[2]	$O(\frac{k}{\lg C}C)$		$O(1)$	Random Oracle
This work	$O(kC)$		$O(1)$	OWF

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

We recently noticed an independent work by Mohassel and Riva that also proposes an optimal Yao-based protocol [19]. Their protocol indeed shares the same asymptotic complexity as ours and also relies on minimal assumptions. The protocols for ensuring Gen's output authenticity in both works are essentially the same. Both protocols capture the idea that Eval provides a unique random key corresponding to each of Gen's output wires as the proof of authenticity.

However, our approach favorably compares to theirs for two reasons:

First, for each of Gen's output wires in each of the garbled circuits, Mahassel and Riva's protocol requires two possible random keys, which correspond to 0 or 1, to be encrypted and exchanged, whereas our protocol only needs the one that corresponds to Gen's output value to be encrypted and exchanged. In other words, although both solutions need $O(\sigma n)$ symmetric cryptographic operations, ours has a smaller constant factor.

Second, their approach for checking Gen's input consistency uses a different instance of circuit garbling, in which Gen's and Eval's roles are reversed. Hence, their solution requires $O(n)$ extra instances of OTs *in which the roles are reversed*, which is arguably more expensive than the extra $O(nk)$ cryptographic symmetric operations in our solution.

Paper Organization: We give background and notations in Section 2, show how the three attacks are handled by cryptographic primitives in Section 3, and provide a detailed description of our main protocol in Section 4. Finally, experimental results are reported in Section 5.

2. PRELIMINARIES

We denote by $f(x, y) \mapsto (f_1(x, y), f_2(x, y))$ a two-output objective function, where Gen with input x gets output $f_1(x, y)$ and Eval with input y gets output $f_2(x, y)$. For simplicity, $f_1(x, y)$ and $f_2(x, y)$ are often noted as f_1 and f_2, respectively. We use $f_1 = \perp$ or $f_2 = \perp$ to indicate that either Gen or Eval gets no output. We denote by $\mathsf{com}(x; r)$ the commitment to message x with randomness r. The randomness may be omitted for simplicity. We denote by $\mathsf{enc}_e(x)$ the encryption of message x under encryption key e and by $\mathsf{dec}_d(c)$ the decryption of ciphertext c under decryption key d. Additionally, we denote by $x||y$ or sometimes (x, y) the concatenation of x and y, and by $[n]$ the set $\{1, 2, \ldots, n\}$ for some $n \in \mathbb{N}$. We also use the notation that $x^{(j)}$'s superscript implies that this variable is related to the j-th circuit.

For the rest of this paper, we denote by k the security parameter, by σ the number of copies of the garbled circuit needed (also known as the statistical security parameter), and by n the size of a participant's input and output.

3. MALICIOUS SECURITY

Our protocols achieve security against malicious adversaries according to the standard ideal-real paradigm for defining security. In the full version of this paper, we present the standard definition of ideal-real security and prove that our protocols achieve this notion using proof techniques already highlighted in [15] and [23]. In this abstract, we focus on a high-level discussion of our contributions—namely, how we solve the three security issues faced by the protocol that transforms the Yao protocol into one that is secure in the malicious model via the cut-and-choose technique.

3.1 Two-Output Function Handling

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

Goldreich suggested the use of an auxiliary circuit that encrypts Gen's output and computes the digital signature of the resulting ciphertext so that a malicious Eval could neither learn Gen's output from the ciphertext nor forge an arbitrary signature [4]. Later, Lindell and Pinkas proposed an approach that uses one-time-pad encryption and one-time MAC circuits instead, which incurs $O(kn)$ extra gates per circuit [15]. Kiraz presented a two-party protocol in which a

zero-knowledge proof of size $O(\sigma)$ is executed at the end [11]. shelat and Shen reported a signature-based solution that adds $O(n)$ gates to each circuit, and requires a WI proof of size $O(\sigma + n)$ [23] under specific complexity assumptions.

Our approach solves Gen's output privacy problem with a one-time-pad encryption circuit, which requires only $O(n)$ extra XOR-gates per circuit. The novel part of our approach is that we tackle the output authenticity problem in a way that uses only $O(n)$ symmetric operations per circuit and no algebraic 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 retrieved from evaluating Gen's output gates can in fact serve as "message authentication codes" sufficient for Eval to prove Gen's output authenticity [3]. Recall that the Yao protocol ensures that Eval learns exactly one of the two random keys assigned to each wire. So the knowledge of the retrieved random key corresponding to Gen's output wire is more than enough for Eval to show the output authenticity. What remains is how Eval demonstrates this knowledge without revealing the index of the garbled circuit from which Eval retrieves the random key. This index has been shown to be exploitable in breaching Eval's input privacy [11].

The second observation we have, which is also pointed out in shelat and Shen's work [23], is that a WI proof suffices the purposes here. Let us consider the case in which Gen (plays as the verifier) has private input $U = \{u^{(j)}\}_{j\in[s]}$ for some $s \in \mathbb{N}$, and Eval (plays as the prover) knows $u^{(m)}$ for some $m \in [s]$. In the honest-but-curious model, a simple WI proof of Eval's knowledge $u^{(m)}$ can be done as follows:

1. Gen picks random nonce r, and sends Eval $\{\mathsf{enc}_u(r)\}_{u\in U}$.
2. Eval receives $C = \{c^{(j)}\}_{j\in[s]}$ and returns $\mathsf{dec}_{u^{(m)}}(c^{(m)})$.
3. Gen receives r' and accepts if $r' = r$, or aborts otherwise.

This proof is sound because an Eval with no knowledge of any $u^{(m)} \in U$ can only guess r with negligible probability. However, this proof is not WI in the malicious model. In fact, a malicious Gen may pick distinct $r^{(j)}$s and send Eval

$$(\mathsf{enc}_{u^{(1)}}(r^{(1)}), \mathsf{enc}_{u^{(2)}}(r^{(2)}), \ldots, \mathsf{enc}_{u^{(s)}}(r^{(s)}))$$

so that $u^{(m)}$ can later be deduced by locating r' in $\{r^{(j)}\}_{j\in[s]}$.

To force Gen to behave honestly in Step 1, we suggest that Gen discloses (U, r) after receiving r' so that Eval could check if C is constructed correctly. Our third observation is that this disclosure does not compromise Gen's input privacy. Indeed, Eval should have already learned the majority of U from the circuit evaluation, and r is a random nonce that has no information about Gen's input at all. So Gen's input is not leaked through (U, r). Moreover, this disclosure does not compromise the soundness of the protocol since after Gen receives r', Eval has already delivered its proof of authenticity so that learning (U, r) afterwards will not change the proof retroactively. Nonetheless, we stress that Gen should not learn r' *before* Eval finishes the check, and nor should Eval be able to change r' after the check. This property suggests the use of a commitment scheme. Our idea is that Eval commits to $\mathsf{dec}_{u^{(m)}}(c^{(m)})$ instead of giving it away in clear. After Gen reveals (U, r), Eval checks if C is indeed correctly generated. If the check fails, Eval aborts; otherwise, Eval decommits to r'. Then Gen checks if $r' = r$ and responds as in the honest-but-curious protocol.

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

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

instead of U, where $\bar{u}^{(j)}$ is randomly picked. With this attack, a malicious Gen is capable of providing checkable ciphertexts C when Eval's private input is indeed $u^{(m)}$. In particular, the fact that Eval can provide the correct nonce r confirms that its private input is indeed $u^{(m)}$. A straightforward way to get around this issue is for the two parties to share the commitments to Gen's private inputs in the first place. By the binding property of the commitment scheme, a malicious Gen cannot change its private inputs at will. The correctness of these commitments will be guaranteed by the circuit-level cut-and-choose technique.

The complete description of our Gen's output authenticity protocol is presented in Figure 1, and the security of this protocol is stated in Lemma 1.

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

Private Input: Gen has $\{(u_0^{(j)}, u_1^{(j)})\}_{j\in[s]}$ and Eval has random key v corresponding to Gen's output wire of value b in the m-th garbled circuit for some $m \in [s]$.

1. Gen picks random nounce $r \in \{0,1\}^k$ and sends Eval $\{\mathsf{enc}_u(r)\}_{u\in U}$, where $U = \{u_b^{(j)}\}_{j\in[s]}$.
2. Eval gets $\{c^{(j)}\}_{j\in[s]}$ and sends Gen $\mathsf{com}(\mathsf{dec}_v(c^{(m)}))$.
3. After getting $\mathsf{com}(r')$, Gen decommits to U.
4. Eval checks the decommitted values $\{u^{(j)}\}_{j\in[s]}$ that
   (a) if $\mathsf{com}(u_b^{(j)})$ is correctly opened to $u^{(j)}$ for all j?
   (b) if $\mathsf{dec}_{u^{(j)}}(c^{(j)}) \stackrel{?}{=} \mathsf{dec}_{u^{(s)}}(c^{(s)})$ for all $j \in [s-1]$?
   Eval aborts if any of the checks fails; otherwise, it decommits to r'.
5. Gen accepts the proof if $\mathsf{com}(r')$ is correctly opened and $r' = r$; otherwise, it rejects.

Figure 1: A WI proof for Gen's output authenticity with malicious security (where Eval plays the role of the prover)

Lemma 1. *Let $\{(u_0^{(j)}, u_1^{(j)})\}_{j\in[s]}$ be Gen's private input, and let commitments $\{(\mathsf{com}(u_0^{(j)}), \mathsf{com}(u_1^{(j)}))\}_{j\in[s]}$ and bit b be the common inputs. If all $u_b^{(j)}$s are uniformly distributed over $\{0,1\}^k$, then the protocol presented in Figure 1 satisfies the following properties:*

1. ***(Completeness)** If Eval knows $u_b^{(j)}$ for some $j \in [s]$, Gen always accepts.*
2. ***(Soundness)** If Eval does not know any of $u_b^{(j)}$s, Gen rejects with probability at least $1 - 2^{-k}$.*

3. ***(Witness-indistinguishability)*** *Let* VIEW_v *denote the view of* Gen *from running the protocol with* Eval *using input* v. *If* Eval *knows any* V *of* $\{u_b^{(j)}\}_{j\in[s]}$, *then for any* $v_1, v_2 \in V$, $\{\text{VIEW}_{v_1}\}_{k\in\mathbb{N}}$ *and* $\{\text{VIEW}_{v_2}\}_{k\in\mathbb{N}}$ *are computationally indistinguishable.*

3.2 Generator's Input Consistency

In the cut-and-choose technique, multiple copies of the garbled circuit are constructed and then either checked or evaluated. It is conceivable that a malicious Gen may provide inconsistent inputs to different evaluation circuits. Lindell and Pinkas showed that for some functions, it is not difficult for a malicious Gen to use inconsistent inputs to extract information of Eval's input [15]. 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 Gen's and Eval's i-th input bit, respectively. Instead of providing (a_1, a_2, a_3) consistently, a malicious Gen may send $(1, 0, 0)$, $(0, 1, 0)$, and $(0, 0, 1)$ to different evaluation circuits. In the end, Gen learns the majority bit of Eval's input, which is the extra information that Eval did not agree to reveal.

Several approaches have been proposed to defend this attack. Mohassel and Franklin proposed the equality-checker technique, which requires $O(\sigma^2 n)$ commitments to be computed and exchanged [18]. Lindell and Pinkas developed an approach that also requires $O(\sigma^2 n)$ commitments [15]. Later, they realized that $O(\sigma^2 n)$ commitments are too much of the communication overhead, and then further suggested a pseudo-random synthesizer that relies on efficient zero-knowledge proofs under specific hardness assumptions and requires $O(\sigma n)$ algebraic operations [16]. shelat and Shen proposed the use of malleable claw-free collections, which also uses $O(\sigma n)$ algebraic operations, but they showed that the witness-indistinguishability is sufficient [23]. Our approach gets the best of the both worlds, i.e., it requires only $O(\sigma n)$ symmetric cryptographic operations.

We tackle this issue with an auxiliary circuit, in addition to the objective circuit, that computes a function of Gen's input. At a high level, the circuit-level cut-and-choose technique ensures the correctness of this auxiliary circuit, and its output is used to ensure Gen's input consistency. In particular, for this idea to work, we need to endow the output of this auxiliary circuit with *collision-free* and *hiding* properties. The former ensures that the consistency of the auxiliary outputs implies the consistency of Gen's inputs, and the latter ensures that the auxiliary outputs do not reveal any information about Gen's inputs.

A natural candidate for this auxiliary circuit is a commitment circuit. The binding and hiding properties of a commitment scheme satisfy the two security properties needed here. This is a conceptually much simpler solution. We, however, failed to find a commitment circuit that introduces less overhead than the previous state-of-the-art solution does. Fortunately, we figured that a universal hash circuit is a sufficient and much more efficient alternative. We next give the definition of universal hash functions, and then we discuss how we achieve both collision-free and hiding properties with a universal hash circuit. Finally, we present an efficient instantiation with proper parameters.

Definition 2 (Universal Hash). *A collection of hash functions* $\mathcal{H} = \{h | h : A \to B\}$ *is called* universal *if for any distinct* $x, y \in A$, *the probability that a uniformly chosen* $h \in \mathcal{H}$ *satisfies that* $h(x) = h(y)$ *is at most* $1/|B|$.

3.2.1 Collision-Free Property

This property comes naturally with universal hash functions. Indeed, Definition 2 shows that for any distinct x, y, if they are *fixed* before h is *uniformly* chosen, their hashes are unlikely to collide. This suggests that the collision-free property can be achieved by letting Gen commit to (fix) its inputs before a hash function is jointly (uniformly) picked. Later, the consistency of Gen's inputs can be verified by checking the consistency of the auxiliary outputs (the hashes). Since both the objective circuit and the auxiliary circuit share the same input from Gen, Gen's input consistency to the auxiliary circuits implies the same to the objective circuits. Our protocol is outlined as follows:

1. Gen commits to its inputs $x^{(1)}, x^{(2)}, \ldots, x^{(\sigma)}$, where $x^{(j)}$ denotes its input to the j-th garbled circuit.
2. Gen and Eval jointly and uniformly pick $h \in \mathcal{H}$.
3. Gen constructs σ copies of the garbled circuit. Each circuit contains two parts: the objective circuit and the auxiliary circuit. While the first part computes the objective function, the j-th auxiliary circuit uses the input wires of the objective circuit to compute $h(x^{(j)})$.
4. Eval asks to check the correctness of a random fraction of the garbled circuits. If the check fails, Eval aborts; otherwise, Eval asks Gen to decommit to its inputs for the remaining (unchecked) circuits.
5. Eval first evaluates the remaining auxiliary circuits. If the evaluation outputs (the hashes) are not consistent, Eval aborts; otherwise, Eval proceeds to evaluating the remaining objective circuits.

Since the cut-and-choose technique employs a majority operation at the end, the final evaluation output will not be influenced by a few inconsistent inputs introduced by a malicious Gen. We next argue, at a high level, that with high probability, the protocol outlined above enjoys the desired security property that Gen*'s inputs to the majority of the remaining circuits are consistent.* Indeed, if a malicious Gen is able to pass the hash consistency check and provide inconsistent inputs to the majority of the remaining objective circuits, there are only three possibilities:

1. *The (auxiliary) circuits are faulty:* The cut-and-choose technique ensures that with high probability, this only happens to a minority of the remaining circuits.
2. Gen *is really lucky to have found a collision:* By Definition 2, this happens with probability at most $1/|B|$, which becomes negligible if B is properly chosen.
3. Gen *is able to break the binding property of the commitments:* Since universal hash functions do not even provide pre-image resistance, given h and $h(x^{(i)})$, it can be easy to find $x^{(j)}$ such that $h(x^{(i)}) = h(x^{(j)})$. So if Gen is able to open the commitment from Step 1 to some value computed after h is chosen in Step 2, it breaks the desired security property. However, this would imply that Gen is able to break the commitment scheme's binding property, which happens with negligible probability too.

REMARK 3.1. *Since the above protocol is not the final version of our solution, we only provide the intuitions for now. A simpler and more elaborate protocol will be given in Figure 2, but the same outline and security argument will apply.*

3.2.2 Hiding Property

A deterministic universal hash $h : A \to B$ provides the collision-free property we need, yet it is insufficient for the purposes here due to the lack of the hiding property. Indeed, if the size of domain A is small, EVAL could exhaustively compute the hash of all possibilities in A and then deduce $x^{(j)} \in A$ from $h(x^{(j)})$. If the hashes are pseudo-random, they reveal little information about the input, which is the hiding property we desire. In particular, the celebrated left-over-hash lemma [9] (LHL) states that the output of a uniformly picked universal hash function h is pseudo-random (even if h is made public) as long as the input has enough (min-)entropy. Consequently, we suggest that function $f(x, y) \mapsto (f_1, f_2)$ is converted to $g(x||r, y) \mapsto (f_1, h(x||r)||f_2)$, where r is randomness properly picked by GEN at the beginning and h is a universal hash function uniformly picked *after* GEN commits to its new input $x||r$.

We argue that our approach does provide the hiding property even when h is public. Indeed, given $h(x||r)$ and h, if r is long enough (has enough entropy), for any x', there must exist r' such that $h(x||r) = h(x'||r')$. This shows that giving away hash $h(x||r)$ does not rule out any possibilities of x.

3.2.3 Efficient Instantiation

We suggest the use of the matrix universal hash family

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

for some $m \in \mathbb{N}$. A nice property of this hash family is that it is $\oplus$-homomorphic, that is, for any $x, y \in \{0,1\}^m$ and $h_M \in \mathcal{M}_{k,m}$, it holds that $h_M(x \oplus y) = h_M(x) \oplus h_M(y)$.

This homomorphism allows an efficient instantiation of the protocol outlined above. The main idea is to let EVAL learn GEN's input hash $h_M(x)$ not by circuit evaluation but by first giving away $h_M(\pi)$ and then revealing $x \oplus \pi$, where π is a randomly chosen one-time pad. The $\oplus$-homomorphism of h_M then allows EVAL to compute $h_M(x) = h_M(\pi) \oplus h_M(x \oplus \pi)$ without learning x. We stress that although using this technique allows hashes to be computed locally, we still have to maintain the main structure of the above protocol outline, that is, letting GEN commit to its inputs before h_M is jointly picked and letting EVAL learn the hash of the input to the evaluation circuits. The complete protocol of our GEN's input consistency check is presented in Figure 2, and its security is stated in Lemma 3.

LEMMA 3. *Let $k, \sigma \in \mathbb{N}$ be common inputs such that $\sigma = O(k)$. Suppose* GEN *has private inputs $\{x^{(1)}, x^{(2)}, \ldots, x^{(\sigma)}\}$, and* EVAL *has private input $S \subset [\sigma]$. If $|S| = c \cdot \sigma$ for some $1 > c > 0$ and* EVAL *accepts the proof presented in Figure 2, the probability that no $x^{(j)}$ appears more than $|S|/2$ times in multiset $\{x^{(j)}\}_{j\in S}$ is negligible in k.*

REMARK 3.2. *We stress that* GEN*'s private input $x^{(j)}$ in Figure 2 refers to the concatenation of its actual input and a random input as previously discussed. It is also worth-mentioning that k in Lemma 3 is also the security parameter for the commitment scheme used in Figure 2.*

Common Input: security parameter 1^k, statistical security parameter 1^σ, perfectly-hiding commitment scheme com, and matrix hash function family $\mathcal{M}_{k,m}$ for some $m \in \mathbb{N}$.

Private Input: GEN has $x^{(1)}, x^{(2)}, \ldots, x^{(\sigma)} \in \{0,1\}^m$, and EVAL has subset $S \subset [\sigma]$.

1. GEN randomly picks $\pi^{(1)}, \pi^{(2)}, \ldots, \pi^{(\sigma)} \in \{0,1\}^m$.
2. For all $j \in [\sigma]$, GEN commits to $x^{(j)}$ by sending $\mathsf{com}(\pi^{(j)})$ and $\mathsf{com}(x^{(j)} \oplus \pi^{(j)})$ to EVAL.
3. GEN and EVAL jointly pick a random $h_M \in \mathcal{M}_{k,m}$.
4. GEN sends $h_M(\pi^{(1)}), h_M(\pi^{(2)}), \ldots, h_M(\pi^{(\sigma)})$ to EVAL.
5. EVAL receives $h_\pi^{(1)}, h_\pi^{(2)}, \ldots, h_\pi^{(\sigma)}$. Next,
   - if $j \in [s]\backslash S$, GEN decommits $\mathsf{com}(\pi^{(j)})$ to $\pi'^{(j)}$, and EVAL checks if $h_\pi^{(j)} = h_M(\pi'^{(j)})$;
   - if $j \in S$, GEN decommits $\mathsf{com}(x^{(j)} \oplus \pi^{(j)})$ to $y^{(j)}$, and EVAL computes $h_x^{(j)} = h_M(y^{(j)}) \oplus h_\pi^{(j)}$.

   EVAL rejects if any of the commitments fails to open, any of the checks fails, or for any distinct $a, b \in S$, $h_x^{(a)} \neq h_x^{(b)}$; otherwise, EVAL accepts.

Figure 2: A proof for GEN's input consistency

REMARK 3.3. *We allow* EVAL *to select S, the set of circuits to evaluate. This saves the effort of jointly picking set S, but still guarantees that whatever S that* EVAL *picks will not compromise* GEN*'s privacy. Lemma 3 holds as long as S is a constant fraction of $[\sigma]$. The use of perfectly-hiding commitment schemes avoids the problems with the well-known* Selective Decommitment Attack.

Finally, we suggest the parameters for security level 2^{-k}. By Definition 2, GEN cannot find a collision with probability better than $1/|B|$. So $|B|$ needs to be at least 2^k. As to the size of random input r, the LHL lemma shows that if the min-entropy of $x||r$ is at least $k + 2k = 3k$ and the output hash is k-bits long, then the output is indistinguishable from a truly random k-bit string with probability at least $1 - 2^{-k}$. Since we make no assumptions about the input distribution of x, a simple approach is to uniformly pick r such that $|r| = 3k$. We can do better by exploiting specific properties of $\mathcal{M}_{k,m}$ and reach the same goal with $|r| = 2k + \lg(k)$ as shown in Lemma 4.

LEMMA 4. *Let X_n denote $\{0,1\}^n$ for some $n \in \mathbb{N}$ and $\mathcal{M}_{k,m}$ be defined as above. For any $x \in X_n$ and any $t \geq 2k + \lg(k)$, distributions $\{(h, h(x||r))\}$ and $\{(h, y)\}$ are statistically indistinguishable with probability at least $1 - 2^{-k}$, where h, r, and y are uniformly chosen from $\mathcal{M}_{k,n+t}, X_t$, and X_k, respectively.*

3.3 Selective Failure Attack

The *Selective Failure Attack* possible in the presence of malicious adversaries has been pointed out by Mohassel and Franklin [18] and by Kiraz and Schoenmakers [12]. This attack occurs when a malicious GEN assigns (K_0, K_1) to an EVAL's input wire in the garbled circuit while using (K_0, K_1^*) instead in the corresponding OT such that $K_1 \neq K_1^*$. Consequently, if EVAL has input 1, it learns K_1^*, gets stuck during the evaluation, and GEN eventually learns EVAL's input.

Lindell and Pinkas [15] suggested that EVAL picks matrix $M \in \{0,1\}^{n\times m}$ for some $m \in \mathbb{N}$ and computes its new input $\bar{y} \in \{0,1\}^m$ such that $M \cdot \bar{y} = y$. An auxiliary circuit will later convert $\bar{y}$ back to y to use as input to the original circuit. The insight is that although selective failures allow GEN to probe some partial information of $\bar{y}$, it is possible that this partial information does not reveal any information about EVAL's actual input y. For this approach to work, the security property needed is formulated as follows:

DEFINITION 5. *For some $m, n, k \in \mathbb{N}$, $M \in \{0,1\}^{n\times m}$ is called* k-probe-resistant *if for any non-empty $L \subset [n]$, the Hamming distance of $\bigoplus_{i\in L} M_i$ is at least k, where M_i denotes the i-th row of matrix M.*

As long as M is k-probe-resistant, a malicious GEN will have to successfully probe k bits of $\bar{y}$ in order to gain any partial information of y, the probability of which is negligible. [15] shows that when M is uniformly chosen from $\{0,1\}^{n\times m}$ where $m = \max(4n, 8k)$, the probability that M fails to be k-probe-resistant will be 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 trick [13]. In short, not only does this approach elegantly reduce the selective failure attack problem to the classic error correcting code construction, it also has the potential to incur only little overhead.

Our idea to construct a k-probe-resistant matrix is to use a maximum distance separable code such as Reed-Solomon codes.[3] We will work on the Reed-Solomon code over $\mathbb{F}_{2^t}$ for some $t \in \mathbb{N}$ with codeword size N and message size K.

By optimizing parameters, we prove the following in the full version of this paper:

LEMMA 6. *Let $P_1, P_2, \ldots, P_n$ be distinct, non-zero polynomials with degree at least $K-1$ uniformly picked from $F_{2^t}[x]$, where integer $K \geq (\lg(n)+n+k)/t$. The probability that there exist some $i \in [n]$ and some $L \subset [n]\backslash\{i\}$ such that $P_i = \sum_{j\in L} P_j$ is at most 2^{-k}.*

Finally, we describe and *implement* an algorithm to find a k-probe-resistant M with high probability and prove the correctness of this algorithm as follows:

THEOREM 7. *With probability at least $1-2^{-k}$, the algorithm presented in Figure 5 outputs a k-probe-resistant matrix $M \in \{0,1\}^{n\times m}$ such that $m \in \mathbb{N}$ and $m \leq \lg(n)+n+k+k\cdot\max(\lg(4n),\lg(4k))$.*

4. THE MAIN PROTOCOL

We now give the full description of our main protocol that is based on the Yao protocol presented in Appendix B.[4]

[3]This direction has indeed been mentioned by Lindell and Pinkas: "an explicit construction can be achieved using any explicit linear code. [15]" Here, we develop their idea further by describing an explicit solution that enjoys the optimal performance—optimal asymptotic complexity with a constant factor of one is worth reporting.

[4]We stress that although the Yao protocol is a well-known work, readers are encouraged to get familiarized with our notations and presentation, which serve as a steppingstone to better understanding our main protocol.

Common Input: security parameter 1^k, statistical security parameter 1^σ, symmetric cipher $(\mathsf{enc}, \mathsf{dec})$ with semantic security, perfectly-hiding 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 f_1, EVAL receives f_2.

1. **(New Inputs)** GEN uniformly picks $r \in \{0,1\}^{2k+\lg(k)}$ and $e \in \{0,1\}^{|f_1|}$, while EVAL samples k-probe-resistant matrix M and computes $\bar{y}$ 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 $\bar{y} = \bar{y}_1\bar{y}_2\ldots\bar{y}_{m_2}$, where $\bar{x}_i$ and $\bar{y}_i$ denote the i-th bit of $\bar{x}$ and $\bar{y}$, respectively.

   REMARK 4.1. *r is GEN's random input used to achieve the hiding property of the hash of GEN's input and protect GEN's input privacy, and e is the one-time pad used to encrypt f_1 and protect GEN's output privacy.*

   REMARK 4.2. *By definition, k-probe-resistant matrix M has to have full (row) rank. So for any y, there must exist some $\bar{y}$ such that $M \cdot \bar{y} = y$. Also, given y and M, $\bar{y}$ can be efficiently computed by Gaussian elimination.*

2. **(Pick the Randomness)** For all $j \in [\sigma]$, GEN picks randomness $\rho^{(j)}$ for the j-th garbled circuit and uses $\rho^{(j)}$ to compute $(K_{i,0}^{(j)}, K_{i,1}^{(j)}, \pi_i^{(j)}) \in \{0,1\}^{2k+1}$ for all wires. Let $W_{i,b}^{(j)}$ denote key-locator pair $(K_{i,b}^{(j)}, b \oplus \pi_i^{(j)})$. From now on, $W_{i,b}^{(j)}$ is called the *label* that corresponds to wire w_i of value b in the j-th garbled circuit.

   REMARK 4.3. *Randomness $\rho^{(j)}$ can be considered as either a pool of truly random bits or a truly random seed to a pseudo-random number generator. It has internal states that keep track of used and fresh random bits, and it always returns fresh random bits when it is used. When later requested, GEN needs to reveal $\rho^{(j)}$'s initial state so that EVAL will be able to regenerate the random bits that GEN used to construct the j-th garbled circuit.*

3. **(Commit to Input Label Pairs)** Let $\{w_i\}_{i\in[m_1]}$ be the wires corresponding to GEN's input and $\{w_{m_1+i}\}_{i\in[m_2]}$ be those corresponding to EVAL's input. For each $j \in [\sigma]$, GEN uses randomness $\rho^{(j)}$ to commit to the label pairs assigned to $\{w_i\}_{i\in[m_1+m_2]}$ by sending commitments $\Theta^{(j)}$ and $\Omega^{(j)}$ to EVAL, where

$$\Theta^{(j)} = \{\mathsf{com}(W^{(j)}_{i,0\oplus\pi_i^{(j)}}; \theta_i^{(j)}), \mathsf{com}(W^{(j)}_{i,1\oplus\pi_i^{(j)}}; \theta_i^{(j)})\}_{i\in[m_1]},$$

$$\Omega^{(j)} = \{\mathsf{com}(W^{(j)}_{m_1+i,0}), \mathsf{com}(W^{(j)}_{m_1+i,1})\}_{i\in[m_2]}.$$

   REMARK 4.4. *The statement "GEN uses randomness ρ to commit to something" means that the randomness, such as $\theta_i^{(j)}$ here, needed in the commitment is computed from ρ in a standard, verifiable way. Also, recall that some commitment may have its randomness omitted for presentation simplicity, for example, $\Omega^{(j)}$.*

   REMARK 4.5. *GEN commits to the label pairs assigned to GEN's and EVAL's input wires so that when EVAL later receives proper decommitments, it will know that the decommitted labels are valid. Moreover, the commitment*

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

4. **(Fix Gen's Input)** For each $i \in [m_1]$ and $j \in [\sigma]$, Gen commits to its input by sending commitments $\{\Gamma^{(j)}\}_{j\in[\sigma]}$ to Eval, where $\Gamma^{(j)} = \{\mathsf{com}(W_{i,\bar{x}_i}^{(j)}; \gamma_i^{(j)})\}_{i\in[m_1]}$.

   Remark 4.6. *It is crucial that commitments $\Gamma^{(j)}$ cannot use the randomness from $\rho^{(j)}$ like commitments $\Theta^{(j)}$ do. If so,* Eval *will learn* Gen*'s inputs to check circuits.*

   Remark 4.7. *Commitments $\Theta^{(j)}$ in the previous step along with commitments $\Gamma^{(j)}$ in this step play the role of Step 2 of our* Gen*'s input consistency check shown in Figure 2. In particular, $\Theta^{(j)}$ here is equivalent to $\mathsf{com}(\pi^{(j)})$ there since the swapping pattern in $\Theta^{(j)}$ is exactly $\pi^{(j)}$, and $\Gamma^{(j)}$ here is equivalent to $\mathsf{com}(x^{(j)} \oplus \pi^{(j)})$ there since the committed label $W_{i,\bar{x}_i}^{(j)} = (K_{i,\bar{x}_i}^{(j)}, \bar{x}_i \oplus \pi_i^{(j)})$.*

5. **(Determine the Objective Circuit)** Eval first reveals M, and then both parties jointly run a coin flipping protocol to uniformly pick $H \in \{0,1\}^{k\times m_1}$. Both parties now have determined the objective circuit C that computes $g : (\bar{x}, \bar{y}) \mapsto (\perp, (c, g_2))$, where $\bar{x} = x||e||r$, $y = M \cdot \bar{y}$, $g_1 = f_1(x, y)$, $c = g_1 \oplus e$, and $g_2 = f_2(x, y)$.

   Remark 4.8. *Choosing H plays the role of Step 3 of our* Gen*'s input consistency check presented in Figure 2. The coin-flipping protocol should guarantee unbiased output even in the presence of one malicious party; thus a three-round protocol suffices.*

6. **(Commit to Output Label Pairs)** Let $\{w_i\}_{i\in O}$ be the wires corresponding to Eval's output and $O = O_1 \cup O_2$, where O_1 contains all the wire indices corresponding to output c and O_2 has those corresponding to output g_2. For each $j \in [\sigma]$, Gen uses randomness $\rho^{(j)}$ to commit to the label pairs assigned to $\{w_i\}_{i\in O_1}$ by sending commitments $\Phi^{(j)} = \{\mathsf{com}(W_{i,0}^{(j)}), \mathsf{com}(W_{i,1}^{(j)})\}_{i\in O_1}$ to Eval.

   Remark 4.9. *Commitments $\{\Phi^{(j)}\}_{j\in[\sigma]}$ are part of the common input for our* Gen*'s output authenticity proof.*

7. **(Eval's Input OTs)** For each $i \in [m_2]$, both parties run a $\binom{2}{1}$-OT in which Gen's input equals

$$\left(\{(W_{m_1+i,0}^{(j)}\}_{j\in[\sigma]}, \{(W_{m_1+i,1}^{(j)})\}_{j\in[\sigma]}\right)$$

   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, $Y^{(j)} = \{(W_{m_1+i,\bar{y}_i}^{(j)})\}_{i\in[m_2]}$.

   Remark 4.10. *There are $\sigma{\cdot}m_2$ decommitments needed for* Eval *to retrieve its input labels from $\sigma \cdot m_2$ commitment pairs $\{\Omega^{(j)}\}_{j\in[\sigma]}$, which* Eval *received in Step 3. These decommitments are grouped according to* Eval*'s input index i as* Gen*'s input to the OTs and grouped according to circuit index j as decommitments $Y^{(j)}$.*

8. **(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 $j \in S$; or $s_j = 0$ otherwise. 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. More specifically, for each $j \in [\sigma]$, both parties run a $\binom{2}{1}$-OT in which Eval's input equals s_j and Gen's input equals $(\rho^{(j)}, (X_1^{(j)}, X_2^{(j)}, h_\pi^{(j)}))$, where

$$X_1^{(j)} = \{(W_{i,\bar{x}_i}^{(j)}, \gamma_i^{(j)})\}_{i\in[m_1]}, X_2^{(j)} = \{(W_{i,\bar{x}_i}^{(j)}, \theta_i^{(j)})\}_{i\in[m_1]},$$

   and $h_\pi^{(j)} = H \cdot (\pi_1^{(j)}||\pi_2^{(j)}||\cdots||\pi_{m_1}^{(j)})$.

   Remark 4.11. *The above $\binom{2}{1}$-OTs could run in parallel if they provide security for parallel execution.*

   Remark 4.12. *Decommitments $X_1^{(j)}$ and $X_2^{(j)}$ are used for* Eval *to retrieve* Gen*'s input labels from commitments $\Gamma^{(j)}$ received in Step 4 and $\Theta^{(j)}$ received in Step 3, respectively. In particular, each decommitment in $X_2^{(j)}$ opens one of a pair of commitments in $\Theta^{(j)}$.*

   Remark 4.13. *Revealing $h_\pi^{(j)}$s plays the role of Step 4 of our* Gen*'s input consistency check shown in Figure 2.*

9. **(Garble the Circuit)** 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

$$\begin{aligned} G(g)^{(j)} =& (\langle \pi_a^{(j)}, \pi_b^{(j)}\rangle, \langle \pi_a^{(j)}, 1 \oplus \pi_b^{(j)}\rangle, \\ & \langle 1 \oplus \pi_a^{(j)}, \pi_b^{(j)}\rangle, \langle 1 \oplus \pi_a^{(j)}, 1 \oplus \pi_b^{(j)}\rangle), \end{aligned}$$

   where $\langle b_1, b_2\rangle = \mathsf{enc}_{K_{a,b_1}^{(j)}}(\mathsf{enc}_{K_{b,b_2}^{(j)}}(W_{c,g(b_1,b_2)}^{(j)}))$.

   Remark 4.14. *Once $(K_{i,0}^{(j)}, K_{i,1}^{(j)}, \pi_i^{(j)})$s are chosen for all wires, no more randomness is needed for generating the j-th garbled circuit. So $\rho^{(j)}$ is not needed here.*

10. **(Check the Circuit)** Let $\{w_i\}_{i\in O}$ be the circuit-output wires. Gen then sends $\{G(C)^{(j)}\}_{j\in[\sigma]}$ to Eval, where

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

- **(Check Circuits)** For each $j \in [\sigma]\backslash S$, Eval checks:

(a) if randomness $\rho^{(j)}$ received in Step 8 can regenerate commitments $\{\Theta^{(j)}, \Omega^{(j)}, \Phi^{(j)}\}$ received in Step 3 and Step 6 and reconstruct garbled circuit $G(C)^{(j)}$?

(b) if hash $h_\pi^{(j)}$ received in Step 8 (via circuit-OT) is indeed equal to $H \cdot (\pi_1^{(j)}||\pi_2^{(j)}||\cdots||\pi_{m_1}^{(j)})$, in which permutation bit $\pi_i^{(j)}$ is computed from randomness $\rho^{(j)}$?

- **(Evaluation Circuits)** For each $j \in S$, Eval checks:

(a) if $X_1^{(j)}$ and $X_2^{(j)}$ received in Step 8 successfully opens $\Gamma^{(j)}$ and half of $\Theta^{(j)}$ received in Step 4, respectively? and if the decommitted labels match? Namely,

i. if the i-th entry of $X_1^{(j)}$ successfully opens the i-th entry of $\Gamma^{(j)}$?

ii. if the i-th entry of $X_2^{(j)}$ successfully opens 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 7 successfully opens half of the commitments in $\Omega^{(j)}$? Namely, if the i-th entry of $Y^{(j)}$ successfully opens the $(2 \cdot i + \bar{y}_i)$-th entry of $\Omega^{(j)}$?

EVAL aborts immediately if a failure occurs.

REMARK 4.15. *For evaluation circuits, the fact that decommitments $X_2^{(j)}$ (resp. $Y^{(j)}$) successfully open $\Theta^{(j)}$ (resp. $\Omega^{(j)}$) shows that the majority of the decommitted labels corresponding to* GEN*'s (resp.* EVAL*'s) input are valid. Furthermore, the fact that the decommitted labels from commitments $\Gamma^{(j)}$ coincide with those from commitments $\Theta^{(j)}$ shows that* GEN *indeed commits to its inputs before H is chosen, which is the necessary condition for our 2-universal hash idea to work.*

11. **(Evaluate the Circuit)** For each $j \in S$, EVAL has obtained garbled circuit $G(C)^{(j)}$ and $(m_1 + m_2)$ labels corresponding to the $(m_1 + m_2)$ circuit-input wires of C. EVAL then evaluates the circuit as follows:

(a) For each gate g with retrieved labels $W_a^{(j)} = (K_a^{(j)}, \delta_a^{(j)})$ and $W_b^{(j)} = (K_b^{(j)}, \delta_b^{(j)})$ corresponding to the gate-input wires, EVAL picks the $(2 \cdot \delta_a^{(j)} + \delta_b^{(j)})$-th entry E in $G(g)^{(j)}$ and computes label $W_c^{(j)} = \mathsf{dec}_{K_b^{(j)}}(\mathsf{dec}_{K_a^{(j)}}(E))$ corresponding to the gate-output wire.

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

EVAL interprets $\{b_i^{(j)}\}_{i \in O}$ as $(c^{(j)}, g_2^{(j)})$. Let $Z^{(j)}$ denote the labels that came with $c^{(j)}$.

12. **(Gen's Input Consistency Check)** For all $i \in [m_1]$ and $j \in S$, let $W_i^{(j)} = (K_i^{(j)}, \delta_i^{(j)})$ be the decommitted label corresponding to the i-th bit of GEN's input to the j-th garbled circuit. EVAL computes $h_{\bar{x}}^{(j)} = h_{\pi}^{(j)} \oplus H \cdot (\delta_1^{(j)} || \delta_2^{(j)} || \cdots || \delta_{m_1}^{(j)})$ for all $j \in S$, and then verifies GEN's input consistency by checking if for all $a, b \in S$, $h_{\bar{x}}^{(a)} = h_{\bar{x}}^{(b)}$. EVAL aborts if any of the checks fails.

REMARK 4.16. *Checking the correctness of $h_{\pi}^{(j)}$s received in Step 10 and the checks here play the role of Step 5 of our* GEN *'s input consistency check in Figure 2.*

13. **(Majority Operation)** Let (c, g_2) be the most common tuple in $\Pi = \{(c^{(j)}, g_2^{(j)})\}_{j \in S}$. EVAL aborts if (c, g_2) is not the majority in Π, that is, (c, g_2) does not appear more than $\frac{|\Pi|}{2} = \frac{\sigma}{5}$ times in Π; otherwise, EVAL outputs g_2.

14. **(Gen's Output Authenticity Proof)** The two parties conduct the protocol presented in Figure 1. In particular, the 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 alleged output c. Also, GEN's input equals $\{W_{i,0}^{(j)}, W_{i,1}^{(j)}\}_{i \in O, j \in S}$ and EVAL's input equals $Z^{(j)}$ for some $j \in S$ such that $c^{(j)} = c$. If the proof fails, GEN aborts; otherwise, GEN outputs $c \oplus e$.

THEOREM 8. *Assume that the $\binom{2}{1}$-OT protocol is secure in the presence of malicious adversaries, and there exist a perfectly-hidng commitment scheme, a family of pseudo-random functions, the main protocol* (GEN, EVAL) *presented above securely computes $f : (x, y) \mapsto (f_1, f_2)$ in the presence of malicious adversaries.*

The proof sketch is provided in Appendix C.

5. EXPERIMENTAL RESULTS

In this section, we report empirical evidence of our performance advantages over prior work. We implemented our work on top of the open-source project—KSS [14]. We ran our experiments on the grid Stampede hosted in Texas Advanced Computing Center. Each instance of the experiments invokes 32 computing nodes; each node has 32GB memory and two 2 Intel Xeon E5-2680 2.7G processors; and each processor has 8 cores.

5.1 Performance of our Proposed Technique

We show the performance of our GEN's input consistency check and k-probe-resistant matrix generating algorithm compared with the prior state-of-the-art in Figure 3.

We first compare the performance of the KSS system with that of the KSS system integrated with our GEN's input consistency check. This experiment is conducted by using both systems to evaluate circuits of various input sizes. These circuits compute 2^n blocks of AES128 encryption, where $n = 1, 2, \ldots, 10$ in which GEN provides inputs for 2^n blocks of AES128 (and thus has a 2^{n+7}-bit input), EVAL provides a 128-bit encryption key, and EVAL receives the 2^{n+7}-bit ciphertext. Figure 3a shows that when the input size increases only to a moderate level (2^{17}), the performance gap due to GEN's input consistency check has almost dominated the whole protocol execution. Specifically, the wall-clock running of the improved protocol is 48.8 seconds versus 92.1 seconds for KSS (which was the fastest published protocol whose results we could replicate on our setup).

Next, we show in Figure 3b that as GEN's input size increases, the ratio between the width and height of the k-probe-resistant matrices generated by our algorithm indeed approximates 1, while the ratio of those generated by Lindell and Pinkas's approach remains constant 4. This comparison suggests that for a circuit that has OTs as the dominant component, the overall protocol execution time could be reduced to 25% simply by replacing the k-probe-resistant matrix generated by the original work with the one generated by our algorithm.

5.2 Performance of the Main Protocol

We show in Figure 4 the overall execution time of our system securely evaluating circuits EDT-4095[5], RSA-256[6], and 1024-AES128. Overall, our system is able to handle $650,000+$ (or $\sim 200,000$ non-XOR) gates per second. We

[5]This circuit computes the edit distance of two 4,095-bit inputs.

[6]This circuit computes a 256-bit modular exponentiation.

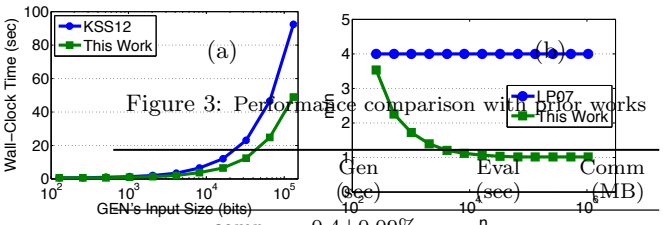

Figure 3: Performance comparison with prior works

		Gen (sec)	Eval (sec)	Comm (MB)
OT	comp	0.4±0.09%	–	6
	comm	0.1± 1%	0.3±0.6%	
cut-& chk	comp	–	–	9
	comm	–	–	
Inp. Chk	comp	0.8± 1%	0.3±0.2%	2,008
	comm	0.3± 1%	0.9± 1%	
Evl.	comp	11.4± 0.6%	28.0±0.4%	72,271
	comm	9.2± 1%	30.3±0.8%	
Total	comp	12.6± 0.3%	28.0±0.2%	74,294
	comm	9.6± 1%	31.5±0.4%	

Table 3: The 95% two-sided confidence intervals of the computation andcommunication time for each stage in the 1024-AES128 experiment $(x, y) \mapsto (\perp, \text{1024-AES128}_y(x))$.

also observe that for all three circuits that we evaluated, more than 60% of the execution time is spent on communicating the huge amount of data, the garbled circuits. If we consider only the circuit garbling, the rate that our system actually achieves could be as high as 1,600,000+ (or 500,000+ non-XOR) gates per second, with the help of various optimization techniques, including SSE2 and AESNI instruction sets, and the free-XOR technique.

circuit	gates	(non-XOR)	time (sec)	comm.
EDT-4095	5.9B	(2.4B)	9,042	18 TB
RSA-256	0.93B	(0.33B)	1,437	3 TB
1024-AES128	32M	(9.3M)	49	74 GB

Figure 4: The performance of our main protocol with $k = 80$ and $\sigma = 256$. All numbers in "time" column come from an average of 30 data points and have the 95% confidence interval $< 1\%$.

6. ACKNOWLEDGEMENTS

We thank Benny Pinkas and Ben Riva for their graciousness in correcting an earlier draft of this manuscript and the anonymous reviewers for their insightful comments.

This work is supported by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL) under contract FA8750-11-2-0211. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the US government.

7. REFERENCES

[1] I. Damgård, V. Pastro, N. Smart, and S. Zakarias. Multiparty Computation from Somewhat Homomorphic Encryption. CRYPTO '12. http://eprint.iacr.org/2011/535.

[2] T. K. Frederiksen, T. P. Jakobsen, J. B. Nielsen, P. S. Nordholt, and C. Orlandi. MiniLEGO: Efficient Secure Two-Party Computation From General Assumptions. EUROCRYPT '13. http://eprint.iacr.org/2013/155.

[3] R. Gennaro, C. Gentry, and B. Parno. Non-Interactive Verifiable Computing: Outsourcing Computation to Untrusted Workers. CRYPTO'10, pages 465–482.

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

[5] O. Goldreich, S. Micali, and A. Wigderson. How to Play any Mental Game. STOC '87, pp. 218–229.

[6] V. Goyal, P. Mohassel, and A. Smith. Efficient Two-Party and Multiparty Computation against Covert Adversaries. EUROCRYPT'08, pp. 289–306.

[7] D. Hofheinz. Possibility and Impossibility Results for Selective Decommitments. *J. Cryptol.*, 24(3):470–516, 2011.

[8] Y. Huang, D. Evans, J. Katz, and L. Malka. Faster Secure Two-Party Computation using Garbled Circuits. USENIX SEC'11, pp. 35–35.

[9] R. Impagliazzo and D. Zuckerman. How to Recycle Random Bits. SFCS '89.

[10] S. Jarecki and V. Shmatikov. Efficient Two-Party Secure Computation on Committed Inputs. EUROCRYPT '07, pp. 97–114.

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

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

[13] V. Kolesnikov and T. Schneider. Improved Garbled Circuit: Free XOR Gates and Applications. ICALP '08, pp. 486–498.

[14] B. Kreuter, a. shelat, and C. Shen. Billion-Gate Secure Computation with Malicious Adversaries. USENIX SEC'12, 2012.

[15] Y. Lindell and B. Pinkas. An Efficient Protocol for Secure Two-Party Computation in the Presence of Malicious Adversaries. EUROCRYPT '07.

[16] Y. Lindell and B. Pinkas. Secure Two-Party Computation via Cut-and-Choose Oblivious Transfer. TCC'11, pp. 329–346.

[17] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella. Fairplay: A Secure Two-Party Computation System. USENIX SEC'04, volume 13, pp. 287–302.

[18] P. Mohassel and M. Franklin. Efficiency Tradeoffs for Malicious Two-Party Computation. PKC'06, pp. 458–473.

[19] P. Mohassel and B. Riva. Garbled Circuits Checking Garbled Circuits: More Efficient and Secure Two-Party Computation, 2013. http://eprint.iacr.org/2013/051.

[20] J. Nielsen and C. Orlandi. LEGO for Two-Party Secure Computation. TCC'09, volume 5444 of *LNCS*, pages 368–386.

[21] J. B. Nielsen, P. S. Nordholt, C. Orlandi, and S. S. Burra. A New Approach to Practical Active-Secure Two-Party Computation. CRYPTO '12, 2012.

[22] B. Pinkas, T. Schneider, N. P. Smart, and S. C. Williams. Secure Two-Party Computation is Practical. ASIACRYPT '09, pp. 250–267.

[23] a. shelat and C.-H. Shen. Two-Output Secure Computation with Malicious Adversaries. EUROCRYPT'11, pp 386–405.

[24] Stampede. Tacc stampede. `http://www.tacc.utexas.edu/resources/hpc/stampede`.

[25] D. Woodruff. Revisiting the Efficiency of Malicious Two-Party Computation. EUROCRYPT'07, volume 4515 of *LNCS*, pp. 79–96.

[26] A. C. Yao. Protocols for Secure Computations. SFCS '82, pp. 160–164.

APPENDIX

A. ALGORITHM

Input: Eval's input size n and a security parameter 1^k
Output: A k-probe-resistant $M \in \{0,1\}^{n\times m}$ for some $m \in \mathbb{N}$

1 **begin**
2 $t \leftarrow \lceil \max(\lg(4n), \lg(4k)) \rceil$
3 **while** $2^{t-1} > k + (\lg(n) + n + k)/(t-1)$ **do**
4 $t \leftarrow t - 1$;
5 **end**
6 $K \leftarrow \lceil (\lg(n) + n + k)/t \rceil$;
7 $N \leftarrow K + k - 1$;
8 **for** $i \leftarrow 1$ ***to*** n **do**
9 Pick $P(x) = \sum_{i=0}^{K-1} a_i x^i$, where $a_i \leftarrow_R \mathbb{F}_{2^t}$;
10 $M_i \leftarrow [P(1)_2 || P(2)_2 || \ldots || P(N)_2]$
11 **end**
12 **return** M; // $M \in \{0,1\}^{n\times m}$, `where` $m = Nt$
13 **end**

Figure 5: A probabilistic algorithm to generate a k-probe-resistant matrix $M \in \{0,1\}^{n\times m}$ for some $m \in \mathbb{N}$. Line 2-5 is to find the minimum t such that $2^t \geq k + (\lg(n) + n + k)/t$, and $P(i)_2$ denotes a $t \times 1$ row vector over $\{0,1\}$.

B. THE YAO PROTOCOL

Our presentation uses the permuted garbled truth table technique [17] (also known as the *point-and-permute* technique in the literature). Briefly, this technique suggests to assign each wire w_i an extra random permutation bit π_i. The circuit garbling and evaluating is then 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 to decrypt.

For each wire, Eval learns exactly one key-locator pair out of the two assigned to that wire from the circuit evaluation, and the learned locator is in fact the evaluation result of that wire one-time padded with the permutation bit. This property ensures that Eval is oblivious to the intermediate result of the circuit evaluation and allows Eval to learn the output by revealing the permutation bits assigned to circuit-output wires.

Common Input: security parameter 1^k, boolean circuit C that computes $f(x,y)$, and symmetric encryption scheme $(\mathsf{enc}, \mathsf{dec})$ with semantic security.

Private Input: Gen has private input $x = x_1 x_2 \cdots x_{m_1}$ and Eval has private input $y = y_1 y_2 \cdots y_{m_2}$, where x_i and y_i denote the i-th bit of x and y, respectively.

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

1. **(Pick the Randomness)** Gen picks $(K_{i,0}, K_{i,1}, \pi_i) \in \{0,1\}^{2k+1}$ at random for each wire w_i. 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 hereafter.
2. **(Retrieve Input Labels)** Let $\{w_i\}_{i\in[m_1]}$ be the wires corresponding to Gen's input, and let $\{w_{m_1+i}\}_{i\in[m_2]}$ be those corresponding to Eval's input.

(a) **(Gen's Input Labels)** Gen sends Eval the labels corresponding to its 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 in which Gen's input equals $(W_{m_1+i,0}, W_{m_1+i,1})$ and Eval's input equals y_i.

3. **(Garble the Circuit)** For each gate $g : \{0,1\}\times\{0,1\} \to \{0,1\}$ 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 $\langle b_1, b_2 \rangle = \mathsf{enc}_{K_{a,b_1}}(\mathsf{enc}_{K_{b,b_2}}(W_{c,g(b_1,b_2)}))$.

Let $\{w_i\}_{i\in O}$ be the circuit-output wires. Gen sends Eval

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

4. **(Evaluate the Circuit)** Eval has obtained circuit $G(C)$ and $(m_1 + m_2)$ labels corresponding to the $(m_1 + m_2)$ circuit-input wires. Eval 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 $W_c = (K_c, \delta_c) = \mathsf{dec}_{K_b}(\mathsf{dec}_{K_a}(E))$.

(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 wire w_i comes with $G(C)$.

Finally, Eval interprets $\{b_i\}_{i\in O}$ as $f(x,y)$ and sends it to Gen. Both parties output $f(x,y)$.

C. MAIN THEOREM PROOF SKETCH

We sketch the construction of simulators S_1 and S_2 such that any malicious Gen* (resp. Eval*) cannot tell whether it is working with S_1 (resp. S_2) in the ideal model or with honest Eval (resp. Gen) in the real model.

Malicious Gen*

Simulator S_1 acts as an honext Eval in the main protocol all the way until the step of OTs except that S_1 uses a fake input $\bar{y}'$. In particular, S_1 first fake input y' and then finds

a pre-image $\bar{y}'$ such that $M \cdot \bar{y}' = y'$, where M is the k-probe-resitant matrix sampled at the beginning.

This fake input $\bar{y}'$ is used as input to Eval's input OTs. The OT's receiver security ensures that Gen* (as the OT sender) cannot distinguish the OT receiver's input being $\bar{y}'$ provided by S_1 in the ideal model from being $\bar{y}$ provided by Eval 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 Gen*, including randomness $\rho^{(j)}$, decommitments $(X_1^{(j)}, X_2^{(j)})$ to Gen's input labels, and hash $h_\pi^{(j)}$. 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 more than $\sigma/5$ circuits are bad, Eval in the real model would abort with high probability too since the probability that none of the bad circuits is checked is negligible.

If S_1 does not abort, it learns the randomness of at least $4\sigma/5$ good garbled circuits. Note that after Gen* passes the circuit checking, S_1 also learns decommitments $(X_1^{(j)}, X_2^{(j)})$ for 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 that Gen* provided to those 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 because the check ensures that the probability of good evaluation circuits having inconsistent inputs is negligible.

Let Gen*'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. Eval in the real model) has come to this far, with high probability, Gen* must have provided majority good evaluation circuits with valid input labels corresponding to Gen*'s input $\bar{x}'$ and S_2's input $\bar{y}'$ (resp. Eval's input $\bar{y}$). So the majority of Eval's evaluation outputs are exactly $f_1(x', y) \oplus e'$. This implies that Gen* cannot distinguish S_1 on input $\bar{y}'$ but providing $f_1(x', y) \oplus e'$ (from the external party) versus Eval on input $\bar{y}$ and provding the evaluation output $f_1(x', y) \oplus e'$. Furthermore, 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 Eval on real input y aborts is negligible.

Finally, since S_1 knows the randomness of many circuits, it also knows the random keys corresponding to Gen*'s encrypted output $f_1(x', y) \oplus e'$. S_1 is able to complete the Gen's Output Authenticity Proof as Eval does.

Malicious Eval*

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 4 by committing to the corresponding labels. The commitments are denoted by $\{\Gamma^{(j)}\}_{j\in[\sigma]}$. Then S_2 invokes OT's simulator S_2^{OT} (whose existence is guaranteed by OT's security) to extract Eval*'s input to OTs, that is, its private input $\bar{y}'$ to Eval's Input OTs and the choice string s' to 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 Eval* 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 Eval* cannot distinguish S_2 using fake input $\bar{x}'$ in the ideal model versus Gen using real input $\bar{x}$ in the real model.

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

For evaluation circuits, the information Eval* 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 $\{\pi_i^{(j)}\}_{i\in[m_1],j\in[\sigma]}$. 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 Eval* always gets consistent output among different garbled circuits. Indeed, in the real model, since Gen 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.