Optimal Parameters For Efficient Two-Party Computation Protocols

Chaya Ganesh[1] and C. Pandu Rangan[2]

[1] Indian Institute of Technology, Madras, India
chaya.ganesh@gmail.com
[2] Indian Institute of Technology, Madras, India
prangan@iitm.ac.in

Abstract. We study the optimal parameters to minimize the cheating probability and communication complexity in protocols for two party computation secure against malicious adversaries. In cut-and-choose protocols for two party computation, we analyze the optimal parameters to keep the probability of undetected cheating minimum. We first study this for a constant number of circuits, and then generalize it to the case of constant bandwidth. More generally, the communication cost of opening a circuit is different from retaining the circuit for evaluation and we analyze the optimal parameters in this case, by fixing the total bits of communication. In the second part of our analysis, we minimize the communication complexity for a given probability of undetected cheating. We study, what should be the parameters to achieve a given cheating probability in minimum amount of communication in a given cut-and-choose protocol. While still keeping the security guarantees, that is, the cheating probability negligible, we achieve a concrete improvement in communication complexity by using optimal parameters in existing cut-and-choose protocols.
Keywords: secure computation, malicious adversaries, cheating probability, communication complexity.

1 Introduction

Secure two party computation. Secure two-party computation allows two parties with respective private inputs x and y to jointly compute a functionality $f(x, y) = (f_1(x, y), f_2(x, y))$, such that the first party receives $f_1(x, y)$ and the second party receives $f_2(x, y)$. The security requirements are privacy and correctness, that is, nothing should be learned from the protocol other than the output, and that the output should be distributed according to the prescribed functionality. The formal definition blends both the requirements and follows the simulation paradigm [2]. Security must be guaranteed even when one of the parties is adversarial. An adversary may be semi-honest, in which case it follows the specification of the protocol, but attempts to learn additional information by analyzing the transcript of messages of the protocol execution. In contrast, the adversary may even be malicious, in which case it can arbitrarily deviate from

the specifications of the protocol. Yao [11] presented the first general solutions for the problem of secure computation, with security against semi-honest adversaries for the two-party case, and Goldreich, Micali and Wigderson [1] gave the solution for the multi-party case with security even against malicious adversaries.

Yao's Protocol. Yao gave a constant-round protocol for the secure computation of any functionality in the presence of semi-honest adversaries. Let f be the functionality that the two parties agree to compute, and let x,y be their respective inputs. (for simplicity, assume that both parties wish to receive $f(x, y)$). In Yao's protocol, party P_1 first generates an encrypted (called "garbled") circuit computing $f(x, \cdot)$ and then sends it to P_2. The circuit is such that it reveals nothing in its garbled form and therefore reveals nothing to P_2. P_2 can, however, obtain the intended output $f(x, y)$ by "decrypting" the circuit. This decryption must ensure that it reveals nothing more than $f(x, y)$ to P_2. That is, P_2 should learn the value on the circuit output wire without learning the values on any of the internal wires. This is accomplished by P_2 obtaining a series of keys corresponding to its input y, such that given the garbled circuit and these keys, the output value $f(x, y)$, and only this value, may be obtained. Now, P_2 must somehow receive these keys from P_1 while making sure not to reveal anything about y to P_1. This is accomplished by running secure 1-out-of-2 Oblivious Transfer (OT) protocol [8]. A detailed description of Yao's protocol, and a proof of security can be found in [5]. Yao's generic protocol is known to be efficient, and even practical, for functionalities that have relatively small circuits.

Malicious behavior and cut-and-choose. Yao's protocol is secure only in the presence of relatively weak semi-honest adversaries. Goldreich, Micali and Wigderson gave the first positive results for general secure computation against malicious adversaries. The compiler of GMW [1] converts any protocol that is secure for semi-honest adversaries into one that is secure for malicious adversaries. The compiler, however, is based on reducing the statement that needs to be proved (the honesty of the parties' behavior in this case) to an NP-complete problem, and using generic zero-knowledge proofs to prove the statement. The secure protocol resulting from the compiler runs in polynomial time but is rather inefficient.

Lindell and Pinkas gave an efficient protocol secure against malicious adversaries, based on cut-and-choose methodology in [4]. Let us consider what happens when P_1 is malicious. It can construct a garbled circuit that computes a function which is different than the one P_1 and P_2 jointly agreed to compute. The "cut-and-choose" technique is a solution to this problem. P_1 first constructs many garbled circuits and sends them to P_2. Then, P_2 randomly chooses half of them and asks P_1 to "open" the chosen half, that is, reveal the decryption keys corresponding to the chosen circuits. P_1 opens the chosen half, and P_2 checks that they were constructed correctly. If they were indeed correct, then P_2 evaluates the remaining circuits and computes the output. The opened and checked circuits are called check circuits, and the rest are evaluation circuits. The idea behind the cut-and-choose methodology is that if a malicious P_1 constructs the circuits incorrectly, then it will be caught by P_2 with high probability. This so-

lution solves the problem of P_1 constructing the circuit incorrectly. Now, since the parties evaluate a number of circuits, some mechanism must be employed to force the parties to use the same input when evaluating each circuit; otherwise, an adversarial party could learn more information than allowed. This, and other requirements that are not met by just applying the cut-and-choose technique are handled by the protocol implementation given in [4], where the authors give a cut-and-choose based solution, and a simulation based proof of security.

Motivation. In a cut-and-choose protocol for two party computation, half of the total number of garbled circuits are check circuits in all protocols in literature. We ask, *what should be the number of check circuits, so as to minimize the probability of undetected cheating* by P_1? We study this question in two settings: same cost circuits, and cheaper check circuits. Then, we ask, *what should be the number of check circuits to achieve a given cheating probability in minimum amount of communication*? With increasing interest from both within and outside the cryptographic community in secure protocols that are efficient enough to be implemented in practice, efficiency issues are an important consideration. Communication complexity of interactive protocols is one of their most important complexity measures. Bandwidth optimization is interesting in settings where communication is expensive, e.g, for a mobile roaming user. This is the motivation of our work in this paper.

Our Results. We study the optimal parameters for cut-and-choose protocols for general secure computation. Consider a cut-and-choose protocol: the total number of garbled circuits constructed by P_1, the number of circuits opened out of the total (these are called check circuits and the rest are evaluation circuits) are the parameters of the protocol, and the protocol achieves some (negligible) probability of undetected cheating incurring a certain communication complexity. Each garbled circuit has a communication cost associated, which is the number of bits that needs to be communicated in order to send the circuit to the other party. In general, the communication cost of a check circuit need not be the same as the cost of an evaluation circuit. In this paper, we analyze the optimal parameters to achieve minimum probability of undetected cheating and minimum communication complexity. We first study the optimal fraction of the total number of circuits that should be check circuits, so as to minimize the probability of undetected cheating by P_1. We then generalize this to the case when the communication cost of a check circuit is cheaper than the cost of an evaluation circuit (as in [3]). We show the optimal number of check circuits to minimize the cheating probability in this setting.

Further, we minimize the communication complexity of a cut-and-choose protocol, while still keeping the probability of undetected cheating by P_1 negligible. We study the optimal number of check circuits to minimize the communication complexity while achieving a given cheating probability. Our analysis yields parameters which can be used in any of the existing cut-and-choose protocols and get a concrete improvement in the communication complexity.

Related work. Efficient protocols for secure two party computation based on the cut-and-choose technique have been studied in [7], [4], [10] and [6]. The optimization of parameters for secure computation protocols has been done earlier only in [9]. Our results are more general, we use a different technique to arrive at optimal parameters for minimum probability of undetected cheating, and our techniques further extend to a more general case of cheaper check circuits. Furthermore, we study the optimal parameters for minimum communication complexity, while achieving a given cheating probability.

2 Background

2.1 Cut-and-Choose Protocol

Lindell and Pinkas gave an efficient two party protocol secure against malicious adversaries [4]. Their construction is based on applying cut-and-choose techniques to the original Yao's circuit and inputs. Security is proved in the ideal/real simulation paradigm.

A malicious P_1 is forced to construct the garbled circuit correctly so that it indeed computes the desired function. P_1 constructs many independent copies of the garbled circuit and sends them to P_2. Party P_2 randomly chooses half of them, and asks P_1 to open the chosen circuits. Now, P_2 checks that the opened circuits are constructed correctly. If they are, then P_2 is convinced that most of the remaining garbled circuits are also constructed correctly. If there are many circuits that are incorrectly constructed, then with high probability, one of those circuits will be in the set that P_2 challenges P_1 to open. The parties then evaluate the remaining circuits as in the original Yao's protocol for semi-honest adversaries, and take the majority output. The protocol also has to force both P_1 and P_2 to use the same inputs in each circuit. Such consistency checks are necessary, because if the parties were allowed to use different inputs to different copies of the circuit, then they can learn information that is more than just the desired output of the function. P_2 can do so, since it observes the outputs of all circuits, but in fact even P_1, who only gets to see the majority output, can learn additional information: For example, if the protocol computes n invocations of a circuit computing the inner-product between n bit inputs. A malicious P_2 could provide the inputs $\langle 10\cdots 0\rangle, \langle 010\cdots 0\rangle, \cdots, \langle 0\cdots 01\rangle$ to the n different garbled circuits, and learn P_1's input completely. If P_1 is malicious, it could also provide the inputs $\langle 10\cdots 0\rangle, \langle 010\cdots 0\rangle, \cdots, \langle 0\cdots 01\rangle$. P_2 now sends P_1 the majority output value, which is equal to the majority value of P_2's input bits. A malicious P_1 could thus get additional information about P_2's input. The protocol enforces consistency checks by having P_1 commit to the garbled circuits and also to the garbled values corresponding to the input wires of the circuits. We give a high-level overview of the protocol here.

Parties P_1 and P_2 have respective inputs x and y, and wish to compute the agreed function $f(x, y)$.

1. The parties first decide on a circuit C that computes f. This circuit is then changed by replacing each input wire of P_2 by a gate whose input consists of

s new input wires of P_2 and whose output is the exclusive-or of these wires. The number of input wires of P_2 increases by a factor of s.

P_2's input is encoded in this way to prevent the following attack by P_1: A malicious P_1 may provide corrupt input to one of possible inputs of P_2 in an OT protocol. In case P_2 chooses to learn this input it will not be able to decrypt the garbled tables which use this value, and will have to abort. If on the other hand, P_2 chooses to learn the other input associated with this wire then it will never know that the first input is corrupt. P_1 can thus learn P_2's input by observing whether or not P_2 aborts. Checking that the circuit is correctly constructed will not help in preventing this attack, since the attack is based on changing P_1's input to the OT protocol. The attack is prevented by replacing the input bits of P_2 with s new input bits whose exclusive-or is used instead of the original input. P_2 can now encode a 0 input in 2^{s-1} ways, and encode a 1 in 2^{s-1} way. Given its input, P_2 chooses an encoding with uniform probability. The protocol is then executed with the new circuit, and P_2 uses oblivious transfer to learn the garbled values of its new inputs. As is shown in [4], if P_1 supplies incorrect values as garbled values that are associated with P_2's input, the probability of P_2 detecting this cheating is almost independent of P_2's actual input.

2. P_1 commits to s different garbled circuits, all of them computing f, where s is a statistical security parameter. Additionally, P_1 also commits to the garbled values corresponding to the input wires of the circuits.

   P_1 can prove consistency of its inputs the following way. The proof is based on a cut-and-choose test for the consistency of the commitment sets combined with the cut-and-choose test for the correctness of the construction of circuits. P_1 constructs s pairs of sets of commitments, for each of its input wires. One set in every pair contains commitments to the 0 values of this wire in all circuits, and the other set is the same with respect to value 1. The protocol now randomly chooses a subset of these pairs, and a subset of the circuits, and checks that these sets give consistent inputs for these circuits. The protocol then evaluates the remaining circuits, and asks P_1 to open, in each of the remaining pairs, and only in one set in each pair, its garbled values for all evaluated circuits. This way, nothing is revealed to P_2 about whether these garbled values correspond to a 0 or to a 1. For the committed sets and circuits to pass P_2's checks, there must be large subsets C and S, of the circuits and commitment sets, respectively, such that every choice of a circuit from C and a commitment set from S results in a circuit and garbled values which correctly compute f. P_2 accepts the verification stage only if all the circuits and sets it chooses to check are from the subsets C and S, respectively. If P_2 does not abort, then circuits which are not from C are in a minority of the evaluated circuits with high probability, and similarly for S. Therefore the majority result of the evaluation stage is correct.

3. Parties P_1 and P_2 run a 1-out-of-2 oblivious transfer protocol for every input bit of P_2 in which P_2 learns the garbled values of input wires corresponding to its input.

4. P_1 sends to P_2 the garbled circuits, as well as all the commitments that it prepared above.
5. P_1 and P_2 run a coin-tossing protocol to choose a random string. The resulting string defines which garbled circuits and commitments will be opened.
6. P_1 opens the garbled circuits and commitments which are chosen in the previous step. P_2 verifies that the opened circuits are correct and runs the consistency checks on the input values.
   P_2 checks the correctness of the check circuits. It verifies that each check circuit is a garbled version of the circuit C. The input tables are constructed by checking that the decommitments in the above step are valid, and associating the first value with the garbled value for 0 and the second value with 1. P_2 next checks the decommitments to P_1's inputs. Finally, given all the garbled values to the input wires and their associated binary values, P_2 now decrypts the garbled circuit and compares it to the circuit C. If any of the checks fail, P_2 aborts.
7. P_1 sends the garbled values corresponding to P_1's input wires in the unopened garbled circuits (evaluation circuits), P_2 runs consistency checks on these values, that is, for every evaluation circuit, all of the commitments that P_1 opened in evaluation sets commit to the same garbled value.
8. If all the checks pass, P_2 evaluates the unopened circuits, and takes the majority value as output.

2.2 Efficient two party computation protocols - Cheaper Check circuits

In [3], the authors design an efficient multi party computation protocol in the covert adversary model. The techniques used in the two party case generalize to the case of two party computation protocols secure against standard malicious adversaries. To achieve an improvement in communication complexity, they take a different approach to constructing the garbled circuit. To construct a garbled circuit and commitments for the input keys, P_1 first generates a short random seed and feeds it to a pseudorandom generator. This generates the necessary randomness. P_1 then uses this randomness to construct the garbled circuit and the commitments. When the protocol begins, P_1 sends only a hash of each garbled circuit using a collision-resistant hash function to P_2. P_2 chooses half of the circuits at random. In order to expose the secrets of each of the chosen circuit later, P_1 sends the seeds corresponding to the circuit, and not the whole opened circuit. P_2 uses the pseudorandom generator to generate the randomness from the sent seed and constructs the check circuits, and checks that they are indeed garbled versions of the agreed circuit. Once the checks go through, P_1 sends the remaining circuits, called the evaluation circuits to P_2. The communication cost of a check circuit is therefore, not the whole circuit but only a hash as opposed to the cost of an evaluation circuit.

We briefly give the garbling procedure of [3]. Let G be the description of a pseudorandom generator, *seed* a seed of suitable length, C be the description of the circuit that computes the agreed function, which is to be garbled.

$Garble(G, seed, C, 1^s)$ denotes the garbling procedure where s is the security parameter. The randomness required for constructing the garbled circuits includes, the random keys corresponding to the circuit wires, the random permutation chosen for the garbled entries of each gate table, and the random string chosen for each encryption. A random string of length $O(s|C|)$ is sufficient for garbling. A pseudorandom generator $G : \{0,1\}^{n_1} \rightarrow \{0,1\}^{n_2}$, where n_1 is polynomial in s and $n_2 = O(s|C|)$ is used. The algorithm *Garble* runs G on *seed* to generate the randomness required for the circuit, and then computes the garbled circuit for C as described in [5].

3 Optimal number of check circuits

In the cut-and-choose protocol described in Section 2.1, we have the flexibility of choosing the number of check circuits. This is the subject of this section; we study what is the optimal number of circuits that should be check circuits to minimize the probability of undetected cheating by Party P_1. In Section 3.1, we keep the total number of garbled circuits a constant, and minimize the cheating probability. In the next subsection, we generalize this to the case where a constant amount of communication bits is allowed (but the total number of circuits is not fixed). We minimize the cheating probability in these settings by choosing the optimal number of check circuits, c. In [9], the authors show the optimal number of check circuits, but our techniques to get an approximation for the expression of cheating probability makes the analysis extendible to the more general case of cheaper check circuits. We discuss both the cases, of same cost circuits and cheaper check circuits as the latter is an extension of the computations done for the former case.

3.1 Same-cost circuits

We consider the case where both check circuits and evaluation circuits have the same communication cost. Let n be the number of garbled circuits constructed by P_1, and let c be the number of circuits checked.
Assume that i out of n circuits are constructed incorrectly by a cheating P_1. P_1's cheating is not caught if all the check circuits are constructed correctly, and P_2 does not abort after evaluation of the remaining circuits.
Now, the probability of P_1's cheating not caught is given by

$$\frac{\binom{n-i}{c}}{\binom{n}{c}}$$

If $i < \dfrac{n-c}{2}$, then the majority output is correct, and if $i > n - c$, corrupt P_1 is caught in one of the check circuits.

Therefore, the cheating probability is,

$$\begin{aligned} P &= \max_i \frac{\binom{n-i}{c}}{\binom{n}{c}} \\ &= \frac{1}{\binom{n}{c}} \max_i \binom{n-i}{c} \end{aligned}$$

The above is maximum for $i = \dfrac{n-c}{2}$. Thus,

$$\begin{aligned} P &= \frac{\binom{\frac{n+c}{2}}{c}}{\binom{n}{c}} \\ &= \frac{\left(\frac{n+c}{2}\right)!}{c!\left(\frac{n-c}{2}\right)!} \frac{(n-c)!c!}{n!} \\ &= \frac{\left(\frac{n+c}{2}\right)!}{\left(\frac{n-c}{2}\right)!} \frac{(n-c)!}{n!} \end{aligned}$$

By Stirling's approximation,

$$n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

$$P \approx \frac{(n+c)^{\frac{n+c+1}{2}}(n-c)^{\frac{n-c}{2}}}{2^c\, n^{n+\frac{1}{2}}} \tag{1}$$

To minimize P for a given n, we differentiate partially with respect to c and set the resulting expression to 0.

Taking logarithm of (1), we get,

$$\log P = \left(\frac{n+c+1}{2}\right)\log(n+c) + \frac{n-c}{2}\log(n-c) - c\log 2 - \left(n+\frac{1}{2}\right)\log n$$

Now differentiating,

$$\frac{dP}{dc} = \frac{P}{2}\left(\log\frac{n+c}{4(n-c)} - \frac{n-c}{n-c} + \frac{n+c+1}{n+c}\right)$$

We now have,

$$\frac{dP}{dc} = 0$$

$$\log\frac{n+c}{4(n-c)} - \frac{n-c}{n-c} + \frac{n+c+1}{n+c} = 0$$

$\frac{n+c+1}{n+c} \approx 1$, Therefore, we have,

$$\log\frac{n+c}{4(n-c)} = 0$$

$$\frac{n+c}{4(n-c)} = 1$$

This yields,

$$c = \frac{3n}{5}$$

Therefore, for a given number of total circuits, n, minimum cheating probability is achieved when the number of check circuits is 3/5th of n.

Theorem 1. *For a given total number of garbled circuits n, constructed and sent by P_1 in the cut-and-choose protocol, P_2 should ask $\frac{3}{5}$th of them to be opened, to minimize the probability of undetected cheating by P_1.*

Thus, challenging P_1 to open 3/5 of the circuits is better for P_2 than challenging half as is done in existing cut-and-choose strategies. Independent of us, the above result is also obtained in [9], by counting the number of bad circuits that optimizes the cheating probability. They do not derive an explicit expression for the cheating probability as we do above, which we use in the generalization to cheaper check circuits which is the subject of the next section.

3.2 Cheaper check circuits

In this section, we consider the more general case of cheaper check circuits. A given number of communication bits does not fix the total number of circuits when the costs of a check circuit and an evaluation circuit are different. Consider the protocol of [3], i.e applying the ideas of [3] to the protocol of the previous section. In this protocol, P_1 uses a short seed and a pseudorandom generator to generate the required randomness for construction of the garbled circuits. A hash of the n garbled circuits are sent to P_2. P_2 asks for a random half of them to be opened, and P_1 sends only the short seeds of the selected half. P_2 verifies that they are constructed correctly. The evaluation circuits are now sent by P_1. The cost of a check circuit is therefore, less than the cost of an evaluation circuit; once the hashes of all n circuits are sent, only a short seed is communicated in case of a check circuit, whereas the whole garbled circuit is the communication cost for an evaluation circuit. In this case, we analyze the optimal number of check circuits for minimum cheating probability. In the previous case when the costs of a check circuit and an evaluation circuit are the same, fixing the communication bits k, also fixes the total number of circuits n. We now study the general case when a check circuit is cheaper than an evaluation circuit. Given a constant amount of communication bits, we minimize the cheating probability. Our analysis yields the total number of circuits to be constructed and the number of circuits to be challenged so as to achieve minimum probability of undetected cheating in a given amount of communication bandwidth.

Let k be the number of bits of communication allowed. Let c be the number of check circuits, e the number of evaluation circuits and n the total number of circuits.

Then,

$$c \cdot Cost_{check} + e \cdot Cost_{eval} = k$$

Let q be the ratio of the cost of check circuits to the cost of evaluation circuits.

$$cq + e = s$$

where,

$$q = \frac{Cost_{check}}{Cost_{eval}}, s = \frac{k}{Cost_{eval}}$$

Using $n = c + e$ and $e = s - cq$ in the cheating probability, we have,

$$\begin{aligned} P &\approx \frac{(n+c)^{\frac{n+c+1}{2}} (n-c)^{\frac{n-c}{2}}}{2^c \, n^{n+\frac{1}{2}}} \\ &= \frac{(2c+e)^{\frac{2c+e+1}{2}} e^{\frac{e}{2}}}{2^c \, (e+c)^{e+c+\frac{1}{2}}} \\ &= \frac{(2c+s-cq)^{\frac{2c+s-cq+1}{2}} (s-cq)^{\frac{s-cq}{2}}}{2^c \, (s-cq+c)^{s-cq+c+\frac{1}{2}}} \end{aligned} \tag{2}$$

We now differentiate with respect to c and equate the resulting expression to zero.

Taking logarithm of (2) we get,

$$\log P = \frac{(2-q)c+s+1}{2} \log\left((2-q)c+s\right) + \frac{s-cq}{2} \log\left(s-cq\right) - c\log 2 - \left(c(1-q)+s+\frac{1}{2}\right) \log\left(c(1-q)+s\right)$$

Differentiating with respect to c we have,

$$\frac{1}{P}\frac{dP}{dc} = \frac{2-q}{2} \log\left((2-q)c+s\right) - \frac{q}{2}\log(s-qc) - \log 2 - (1-q)\log\left((1-q)c+s\right)$$

Now setting the derivative to zero,

$$\frac{dP}{dc} = 0$$

$$\log\left(\frac{\left((2-q)c+s\right)^{\frac{2-q}{2}}}{2(s-qc)^{\frac{q}{2}} \left((1-q)c+s\right)^{1-q}}\right) = 0$$

$$\frac{((2-q)c+s)^{\frac{2-q}{2}}}{2(s-qc)^{\frac{q}{2}}((1-q)c+s)^{1-q}} = 1$$

This implies,

$$((2-q)c+s)^{\frac{2-q}{2}} = 2(s-qc)^{\frac{q}{2}}((1-q)c+s)^{1-q} \tag{3}$$

$$(2c+e)^{1-\frac{q}{2}} = 2\,e^{\frac{q}{2}}\,(c+e)^{1-q}$$

$$(n+c)^{1-\frac{q}{2}} = 2\,(n-c)^{\frac{q}{2}}\,n^{1-q}$$

$$\left(1+\frac{c}{n}\right)^{2-q} = 4\left(1-\frac{c}{n}\right)^{q} \tag{4}$$

Let r be the fraction of the circuits which are check circuits. i.e

$$r = \frac{c}{n}$$

Using this in (4) yields,

$$(1+r)^{2-q} = 4\,(1-r)^{q}$$

$$(1+r)^{2} = 4\,\left(1-r^2\right)^{q} \tag{5}$$

Given q, the ratio of costs, we can solve the above equation for r.

The total number of circuits to be sent n is then given by,

$$n = \frac{k}{(1-r)\,Cost_{eval} + r\,Cost_{check}}$$

Given that we are allowed a constant k bits of communication, the total number of circuits, and the number of check circuits can be set as in the above analysis to minimize the cheating probability by P_1. If the cost of check circuit is the same as the cost of the evaluation circuit,
i.e when,

$$Cost_{check} = Cost_{eval}, q = 1$$

Setting $q = 1$ in equation (5) yields,

$$r = \frac{3}{5}$$

and this agrees with our earlier conclusion when the costs are same.

Theorem 2. *Let q be the ratio of the communication cost of a check circuit to the cost of an evaluation circuit in a two party cut-and-choose protocol, and k, a constant amount of communication bits allowed. Then, probability of cheating by P_1 is minimized if the ratio of the number of check circuits to the number of evaluation circuits, r is as given by, $(1+r)^2 = 4\,\left(1-r^2\right)^q$, and the total number of circuits, $n = \frac{k}{(1-r)\,Cost_{eval} + r\,Cost_{check}}$.*

4 Communication Complexity

In this section, we minimize the number of communication bits for a given cheating probability. We show how to achieve a given negligible probability of undetected cheating in minimum communication. We show that, for our choice of parameters, the communication complexity of existing cut-and-choose protocols can be improved. The communication complexity of existing protocols are stated in the following theorems.

Theorem 3. *([10]) Let n be the number of circuits, and g the number of gates in the circuit. The protocol of [10] is secure in the malicious model with inverse exponential (in n) probability of undetected cheating. The communication complexity is $O(ng)$.*

Theorem 4. *([7]) The equality-checker protocol of [7] is secure in the malicious model with probability of undetected cheating ϵ. If g is the number of gates the circuit, and I the number of input bits, the communication complexity of the scheme is $O(ln\left(\frac{1}{\epsilon}\right) g + ln\left(\frac{1}{\epsilon}\right)^2 I)$.*

4.1 Minimize Communication Complexity

We now formulate the problem as minimizing the communication complexity which is a function of two variables, given a cheating probability. Given a cheating probability p, for what relation between the check circuits and the total number of circuits, is p achieved in minimum number of communication bits? Minimize the function,

$$f(c, n) = c \cdot Cost_{check} + (n - c) \cdot Cost_{eval}$$

That is, minimize,

$$k = c + (n - c)Q$$

subject to the constraint that,

$$p = \frac{1}{2^c} \frac{(n + c)^{\frac{n+c+1}{2}} (n - c)^{\frac{n-c}{2}}}{n^{n+\frac{1}{2}}}$$

where,

$$Q = \frac{Cost_{eval}}{Cost_{check}}$$

Since the constraint equation is exponential, we go back to the expression of cheating probability and try to get a more friendly approximation.

$$p \approx \frac{\binom{\frac{n+c}{2}}{c}}{\binom{n}{c}}$$

We know that,

$$\binom{x}{y} \geq \left(\frac{x}{y}\right)^y$$

$$\binom{x}{y} \leq \frac{x^y}{y!} \leq \frac{x^y}{2^{y-1}}$$

Therefore,

$$p \geq \frac{(\frac{n+c}{2c})^c}{\frac{n^c}{2^{c-1}}} = \left(\frac{n+c}{cn}\right)^c \frac{2^{c-1}}{2^c}$$

$$p \geq \frac{1}{2}\left(\frac{1}{c} + \frac{1}{n}\right)^c$$

$$(2p)^{\frac{1}{c}} \geq \frac{1}{c} + \frac{1}{n}$$

$$\frac{1}{n} \leq (2p)^{\frac{1}{c}} - \frac{1}{c}$$

$$n \geq \frac{c}{c\,(2p)^{\frac{1}{c}} - 1}$$

Since we do not get a closed form for c in terms of n and p, we investigate, of what order c should be in, so that k is minimized while keeping p negligible.

Let $n - c = n^\epsilon$, and, $\epsilon = \dfrac{\log\log n}{\log n}$
For this value of ϵ, $n^\epsilon = \log n$, and $n - n^\epsilon = c = n - \log n$.

For the case when check circuits are cheap as opposed to evaluation circuits, the total communication is dominated by the number of evaluation circuits. For the above value of c, p is still negligible. We now show that the cheating probability p is negligible when $c = n - \log n$.

$$p = \frac{1}{2^c} \frac{(n+c)^{\frac{n+c+1}{2}} (n-c)^{\frac{n-c}{2}}}{n^{n+\frac{1}{2}}}$$

For $c = n - \log n$,

$$p = \frac{1}{2^{n-\log n}} \frac{(2n - \log n)^{\frac{2n-\log n+1}{2}} (\log n)^{\frac{\log n}{2}}}{n^{n+\frac{1}{2}}}$$

For large n, and $c = n - n^\epsilon$, p is still negligible. That is, $p \approx \dfrac{1}{n^{\frac{1}{2}+\frac{\log n}{2}}}$.
We observe that, the cheating probability p remains negligible, for $\epsilon = \dfrac{\log\log n}{\log n}$, giving communication cost dominated by the cost of evaluation circuits. Therefore, we get constant factor improvement on the communication complexity by the new choice of parameters; by choosing the number of check circuits to be, $c = n - \log n$. Thus, the communication cost of existing cut-and-choose protocols can be improved, while still retaining the security guarantees, i.e. keeping the

cheating probability negligible. The communication complexity and the cheating probability achieved by the above optimal parameters in a cut-and-choose protocol are stated in the following theorem.

Theorem 5. *The number of check circuits to achieve near optimal communication cost is given by $c = n - \log n$, where n is the total number of circuits. The communication complexity is dominated by $O(\mathcal{C} \log n)$, and the probability of undetected cheating is $p \approx \frac{1}{n^{\frac{1}{2}+\frac{\log n}{2}}}$, where $\mathcal{C}$ is the communication cost of an evaluation circuit.*

The improvement by using the above parameters in the setting of cheaper check circuits [3], is discussed in the following section. Using the above choice of parameters for ϵ, the number of check circuits c and the technique of [3] as applied to cut-and-choose protocol for general secure two party computation, we get an improvement in the communication complexity in concrete terms compared to the protocols of [7], [10], [6].

4.2 Comparison of communication cost

In the previous section, we analyzed in detail the parameter values such as the number of circuits to be challenged and arrived at optimal values to achieve minimum communication complexity. We now show how the use of these optimal parameters in existing protocols significantly improve the communication complexity. In particular, we sketch how our optimal parameters along with the technique of [3], improve the efficiency of existing cut-and-choose constructions. Informally, in the setting of [3], P_1 sends only a hash of the check circuits, and the entire garbled circuit is sent only for an evaluation circuit. Check circuits are therefore cheaper than evaluation circuits. Let the total number of circuits be n. From the analysis of Section 4.1, we choose the number of evaluation circuits to be optimal for minimum communication cost which is $\log n$. We summarize our results below.

Let the size of a garbled circuit be, $|GC| = 4g|E|$, where $|E|$ is the size of the ciphertext of the encryption scheme, g the number of gates in the circuit and I the input size. Let $|E|$ also be the output size of the commitment scheme. If t is the number of garbled circuits in [7], the number of bits communicated is $t|GC| + t^2 I|E| = 4tg|E| + t^2 I|E|$ (Theorem 4). The same protocol, by using the parameters of our analysis communicates, roughly, $4g|E| \log n + (\log n)^2 I|E|$ bits, where n is the total number of circuits (Theorem 5). Consider a circuit with 32 gates and input wires, $g = I = 32$. The cheating probability and the communication complexity achieved by [7] and by incorporating the optimal choice from our analysis in [7] is shown in the table below.

For the purpose of our comparison, we fix the cheating probability. Setting the number of circuits t and n, in the two variants such that, the cheating probability is the same, say, 2^{-50}, we get $t = 196$ and $\log n = 10$. Substituting the values of t and n in the number of communication bits shown above, we see that the communication bits using our optimal parameters is around 20 times

Table 1.

Scheme	Communication Complexity	Cheating Probability
[7]	$128t\|E\| + 32t^2\|E\|$	$2.(1/2)^{t/4}$
This paper	$128\|E\|\log n + 32(\log n)^2\|E\|$	$1/n^{\frac{1}{2}+\frac{\log n}{2}}$

less than the communication bits of [7]. It is also important to note that this factor of improvement increases as the number of gates and input wires in the circuit increase.

We now consider the protocol of [10]. If t is the number of garbled circuits in [10], the number of bits communicated is roughly $4tg|E|+2tI|E|+tg+tI|E|$. The number of bits communicated by using our techniques is, roughly, $4g|E|\log n + 3\log nI|E|$ (Theorem 5), where n is the total number of circuits. If we consider a circuit with 32 gates and input wires, $g = I = 32$, and set the parameters t and n such that the cheating probability is the same, then, $t = \frac{\log n}{2} + \frac{(\log n)^2}{2}$. Now, for a cheating probability of 2^{-50}, we get $t = 50$ and $\log n = 10$. Substituting the values of t and n in the number of communication bits shown above, we see that the communication bits using our optimal parameters is a factor of 9 less than the scheme of [10].

More recently, in [6], Lindell and Pinkas presented a protocol using the cut-and-choose methodology relying on DDH assumption, and significantly improved the efficiency. We give the improvement in communication cost by using optimal parameters in [6]. The communication complexity of [6] is the exchange of $5tI+14I+7t+5$ group elements and t copies of the garbled circuit. The cheating probability is $1/2^{t/4}$. The communication cost is dominated by the garbled circuits. Using the techniques in this paper, the communication cost of sending the garbled circuits is $4\log ng|E|$, for a cheating probability of $\frac{1}{n^{\frac{1}{2}+\frac{\log n}{2}}}$. To achieve

Table 2.

Scheme	Cheating Probability
[6]	$1/2^{t/4}$
This paper	$1/n^{\frac{1}{2}+\frac{\log n}{2}}$

the same cheating probability, say 2^{-50} we set $1/2^{t/4} = 2^{-50}$ and compute the number of GC's required as $t = 200$. We now set $\frac{1}{n^{\frac{1}{2}+\frac{\log n}{2}}} = 2^{-50}$, and solve the quadratic equation to get $\log n = 10$. This gives a factor of 20 improvement in the communication cost. Thus, for the same cheating probability, our techniques lead to a factor of 20 less communication between the parties in the protocol of [6]. We also remark that the increase in computational complexity is feasible. $\log n = 10 \Rightarrow n = 2^{10}$ implies that the total number of GC's the parties need is 1024.

5 Conclusion

In this paper we gave an analysis for optimal parameters in cut-and-choose protocols to achieve: (a) minimum probability of undetected cheating for same-cost circuits and cheaper check circuits, and (b) minimum communication bits in which a given cheating probability can be achieved. It would be interesting from a practical point of view, to carry out such optimization as part of an actual implementation of a specific protocol.

References

1. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. *In proceedings of 19th Annual ACM Symposium on Theory of Computing*, pages 218–229, 1987.
2. Oded Goldreich. *Foundations of Cryptography, Volume II: Basic Applications*. Cambridge University Press, 2004.
3. Vipul Goyal, Payman Mohassel, and Adam Smith. Efficient two party and multi party computation against covert adversaries. In *EUROCRYPT 2008*, pages 289–306, 2008.
4. Yehuda Lindell and Benny Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. In *EUROCRYPT*, pages 52–78, 2007.
5. Yehuda Lindell and Benny Pinkas. A proof of yao's protocol for secure two-party computation. *Journal of Cryptology, 22(2)*, pages 161–188, 2009.
6. Yehuda Lindell and Benny Pinkas. Secure two-party computation via cut-and-choose oblivious transfer. In *8th TCC*, pages 329–346, 2011.
7. Payman Mohassel and Mathew Franklin. Efficiency tradeoffs for malicious two party computation. In *Public Key Cryptography Conference*, pages 458–473, 2006.
8. M. Rabin. How to exhange secrets by oblivious transfer. Technical Memo, TR-81, Aiken computation laboratory, Harvard U., 1981.
9. Abhi Shelat and Chih-Hao Shen. Two-output secure computation with malicious adversaries. In *Advances in Cryptology - Eurocrypt 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 386–405. Springer, 2011.
10. David P. Woodruff. Revisiting the efficiency of malicious two party computation. In *EUROCRYPT 2007*, pages 79–96, 2007.
11. A. C. Yao. How to generate and exchange secrets. In *FOCS '86: Proceedings of 27th Annual Symposium on Foundations of Computer Science*, pages 162–167, 1986.