Survey on Complexity of Secure Function Evaluation

Naman Agarwal
Department of Computer Science
University of Illinois, Urbana-Champaign
nagarwl3@illinois.edu

Done as part of the coursework in CS 598 under Prof. Manoj Prabhakaran

Abstract

In this survey we consider the problem of quantification of the complexity of secure two-party function evaluation in terms of the number of AND black boxes used by the protocols. We specifically prove explicit lower and upper bounds on the number of ANDs required in case of deterministic and randomized protocols.

1 Introduction

In this paper survey we will specifically be dealing with the question of quantifying the complexity of secure two party function evaluation. The ideas and theorems/proofs presented in this report have been borrowed from the paper A Quantitative Approach to Reductions in Secure Computation [BM04] on this subject by Amos Beimel and Tal Malkin.

So what is Secure Two-party Function Evaluation ? Though we define this notion formally in the next section but intuitively it means the problem of computing a function whose arguments are held partially by one party (Alice) and partially by another party (Bob) such that in the process of evaluation no extra information is leaked to either party except what would be observable from the output itself.

On the face of it its not even clear whether all functions are even computable under the following conditions. It is indeed proven [BOGW88] that in the tight perfect security information theoretic setting all functions are not computable. However as shown by Goldreich and Vainish [GV87] and Killian [Kill88], given some complete function (more specifically an AND or an Oblivious Transfer) all functions are in fact computable in the tight information theoretic setting. The previous result automatically suggests a way of reduction in which these complete functions are treated as black boxes and Secure Computation can be designed based on these black boxes. The black boxes could then be replaced by computationally secure implementation of them.

The nature of the reduction above leads to two possible forms of hierarchies in functions. Firstly a crude division in 'easy' and 'hard' functions where by easy we mean that these functions are computable without using these black boxes and 'hard' functions are those that necessarily require these black box. There has been extensive research on this characterization and a full list of papers dealing with this can be found in the introduction of the main paper [?].

The other more fine grained approach to quantifying the complexity of these reductions is considering the question of the number of such black boxes required to securely compute a function. Lets refer to this as the AND − complexity of a function $f$ and denote it by $AND(f)$. It is precisely this hierarchy that the paper we are reviewing tries to explore.

2 Results

The authors consider two kind of protocols DETERMINISTIC and RANDOMIZED PROTOCOL. The Deterministic setting is when both parties are allowed no randomness and follow a deterministic protocol given their inputs. The Randomized setting is one where in both parties have a random tape. Specifically the authors show the following things
2.1 Deterministic Protocols

- An upper bound on $\text{AND}(f)$ in terms of the size of the input space (it is exponential in the size of the input space). This shows that given enough ANDs any function can be computed.

- A matching lower bound in case of boolean functions. Therefore for some functions an exponential number of ANDs is indeed necessary.

2.2 Randomized Protocols

- They exhibit using a specific function an exponential gap between AND-complexity of a deterministic protocol and a randomized one. They also exhibit a trade-off between AND-complexity and amount of randomness used for this specific function.

- They prove a lower bound on the AND-complexity in Randomized protocols. More specifically they bound the gap between deterministic and non-deterministic protocols: For any randomized protocol with AND complexity $q$ there is a deterministic protocol using at most $2^q$ ANDs.

- They also consider the notion of non-perfect protocols where in correctness and security are parametrized by error factors and show that the AND-complexity can solely be characterized in terms of the error factors. More specifically any function can be computed with $k$ ANDs with a protocol achieving $1/2^k$ probability of error and statistical distance.

In this survey we formally define the setting considered by the authors and the notion of security. We provide proofs of some of the above results. Note that in interest of brevity the level of detail and rigour in the proofs have been kept low. Interested readers are referred to the detailed proofs presented in the original paper $[?]$.  

3 Relation between Circuit Complexity and AND-complexity

Goldreich and Killian $[GV87]$ proved that for any function that can be computed by a circuit of size $s$ there exists a randomized protocol that computes it using $4s$ ANDs. This shows that $\text{CIRCUITCOMPLEXITY} > \text{ANDCOMPLEXITY}$ modulo constant factors. This shows that any super-linear lower bound on $\text{AND} – \text{COMPLEXITY}$ proves a super linear lower bound on $\text{CIRCUIT} – \text{COMPLEXITY}$ which has been a notoriously hard open problem. Note that the above result automatically provides an upper bound on $\text{AND} – \text{complexity}$ in terms of circuit size. It should also be noted that the circuit complexity by itself gives no clear indication on the lower bound for $\text{AND} – \text{complexity}$. It may happen that the circuit for a certain function may be of large size but all inputs lie with Alice and therefore no AND boxes are needed.

4 Definitions, Notations and Preliminaries

**Definition 1** (Protocol). A protocol is defined on two parties Alice and Bob. Alice has an input $x$ and a random tape whose value is referred to as $r_A$. Bob has an input $y$ and a random tape whose value is denoted by $r_B$. The protocol has access to a black box $BB : D_1 \times D_2 \rightarrow D_3$. A round in the protocol is composed of three steps:

- **Alice’s message** Alice sends a message to Bob which is available to Bob at the start of the next round.

- **Bob’s message** Bob sends a message to Alice which is available to Alice at the start of the next round.

- **Black Box round** Alice inputs a value $\in D_1$ and Bob inputs a value $\in D_2$ in the black box and Alice receives the input $\in D_3$.  

A computation between Alice and Bob ends after a certain number of rounds at the end of which Alice computes a private output. We are only concerned with an AND black box in our setting where \( \text{AND}(a, b) = a \land b \).

Fix an execution \( E \) of the protocol where the inputs of Alice and Bob are \( x, y \) and their random tape values are \( r_A, r_B \).

**Definition 2 (Transcripts, Views).**

- The transcript of \( E \) refers to the sequence of messages exchanged by Alice and Bob and is denoted by \( \text{TRANS}(x, r_A, y, r_B) \).
- The Black Box outputs visible to Alice during the execution is referred to as \( \text{BLACK-BOX}(x, r_A, y, r_B) \).
- View of Alice is defined as \( \text{VIEW}_{Alice}(x, r_A, y, r_B) = (x, r_A, \text{TRANS}(x, r_A, y, r_B), \text{BLACK-BOX}(x, r_A, y, r_B)) \).
- View of Bob is defined as \( \text{VIEW}_{Bob}(x, r_A, y, r_B) = (y, r_B, \text{TRANS}(x, r_A, y, r_B)) \).

In a similar spirit we can define random variables \( \text{TRANS}(x, \ldots, y, r_B), \text{TRANS}(x, r_A, y, \ldots), \text{TRANS}(x, \ldots, y, \ldots) \) by choosing one or both of \( r_A, r_B \) randomly and outputting the value of \( \text{TRANS}(x, r_A, y, r_B) \).

We consider only the notion of security against an honest but curious adversary which is an adversary that follows the protocol scrupulously but is allowed to derive as much information as it can from its view in the protocol.

It is under these settings that we define the notion of secure computation. Note that we are defining only the notion of perfect correctness and security here (notions of statistical correctness and security can be similarly defined).

**Definition 3 (Secure Computation).** Let \( f: A \times B \to C \) be a function. A protocol \((Alice, Bob)\) securely computes \( f \), if the following conditions hold:

- **Correctness** For every \( x \in A \) and every \( y \in B \), with probability 1 Alice outputs \( f(x, y) \) where the probability is taken over \( r_A, r_B \).
- **Bob’s Privacy** \( \forall x \in A, \forall y_0, y_1 \in B, \forall r_A, \text{ if } f(x, y_0) = f(x, y_1) \text{ then distribution of } \text{VIEW}_{Alice}(x, r_A, y_0, \ldots) \text{ and } \text{VIEW}_{Alice}(x, r_A, y_1, \ldots) \text{ are the same} \).
- **Alice’s Privacy** \( \forall x_0, x_1 \in A, \forall y \in B, \forall r_B, \text{ distribution of } \text{VIEW}_{Bob}(x_0, \ldots, y, r_B) \text{ and } \text{VIEW}_{Alice}(x_0, \ldots, y, r_B) \) are the same.

Also define \( M_f \) a matrix associated with a function which has \( |A| \) rows and \( |B| \) columns and \( M_f(x, y) = f(x, y) \). The column corresponding to \( y \in B \) would be referred to as \( C_y \) and the row corresponding to \( x \in A \) would be referred to as \( R_x \).

We further define a notion of equivalence between columns.

**Definition 4.** Two columns \( C_y \) and \( C'_y \) of a matrix \( M \) are related by a relation \( \sim_c \) if \( \exists x \in A \) such that \( M(x, y) = M(x, y') \). Define the equivalence relation \( \equiv_c \) on the columns as the transitive closure of \( \sim_c \), i.e. \( C_y \equiv_c C'_y \) if there are \( y_1 \ldots y_t \) such that \( y \sim_c y_1 \ldots \sim_c y_t \sim_c y' \).

We now decompose a function according to the equivalence classes on the above defined relation \( \equiv_c \) and show that evaluation can be considered on these classes themselves.

**Claim 1.** Let \( f: A \times B \to C \) be a function and \( B_1, \ldots, B_k \) be the equivalence classes of the relation \( \equiv_c \) define \( f_i: A \times B_i \to C \) as the restriction of \( f \) on equivalence class \( B_i \). The function can be securely computed with \( q \) ANDS if each \( f_i \) can be securely computed by \( q \) ANDS.

**Proof.** One direction of the claim is obvious because if \( f \) can be computed with \( q \) ANDS so can any restriction of \( f \). To see that the reverse implication holds consider the protocol in which Bob initially sends Alice the equivalence class its input is in and then they follow the protocol for \( f_i \). Since Alice can by looking at the output anyway figure out which equivalence is Bob’s input in this initial message gives no extra information to Alice.
We now prove a very important claim which informally asserts that if the columns of $M_f$ are equivalent then no information is disclosed by the communication and all the information required by Alice to compute the output is passed by the output of the black boxes only.

**Claim 2.** Let $f : A \times B \to C$ be a function such that all columns of $M_f$ are equivalent and let $c$ be any communication transcript that can be exchanged by the parties. Then in a protocol with perfect privacy for every $x, x' \in A$ and every $y, y' \in B$

$$\Pr[c = \text{TRANS}(x, y)] = \Pr[c = \text{TRANS}(x', y')],$$

where the probability is over the random inputs of Alice and Bob.

**Proof.** Note first that by Alice’s privacy we get that for every $x, x' \in A$ and every $y \in B$ and every $r_B$

$$\Pr[c = \text{TRANS}(x, y, r_B)] = \Pr[c = \text{TRANS}(x', y, r_B)]$$

(1)

If the above does not hold Bob can distinguish. Thus we have that for every $x, x' \in A$ and every $y \in B$ and every $r_B$

$$\Pr[c = \text{TRANS}(x, y)] = \Pr[c = \text{TRANS}(x', y)]$$

(2)

Also due to Bob’s privacy for every $x \in A$ and every $y, y' \in B$ such that $f(x, y) = f(x, y')$

$$\Pr[c = \text{TRANS}(x, y)] = \Pr[c = \text{TRANS}(x, y')]$$

(3)

Now note that due to equivalence we can find a path through the matrix from $x, y$ to $x', y'$ on the path $y \sim_c y_1 \ldots \sim_c y_i \sim_c y'$ such that in the first step we jump to the $x$ along the same column(vertically) such that $f(x, y_i) \sim_c f(x, y_{i+1})$ and then we jump horizontally from column $y_i$ to $y_{i+1}$. During vertical jumps we apply equation (2) and during horizontal jumps we apply equation (3) to get the desired result.

Note that the above claim in absence of randomness implies something very strong in case of deterministic protocols. It implies that in deterministic protocols the communication is in fact redundant i.e. for perfect privacy for all inputs the transcript exchanged would be the same and hence it can be discarded altogether which is summarized in the following claim.

**Claim 3.** Let $f : A \times B \to C$ be a function such that all columns of $M_f$ are equivalent. Then in every deterministic secure protocol there is exactly one communication transcript that is exchanged between Alice and Bob for all inputs $x, y$.

### 5 Deterministic Protocols

In this section we prove an upper bound as well as a matching lower bound for boolean function in the deterministic protocol setting. Such a lower bound is possible because due to Claim 2 deterministic protocols have been constrained quite a bit as there is no communication that happens in them. We start by proving a trivial upper bound which is in fact tight for Boolean functions.

**Theorem 1.** Any function $f : A \times B \to C$ can be computed securely by a deterministic protocol using $|A| \log(|C|)$ ANDs

**Proof.** We can wlog assume that $f$ is boolean. We show that it can be computed using $|A|$ ANDs in this case. Note that if this is true we can compute each bit of the output separately and hence totally requiring $|A| \log(|C|)$ ANDs. So let $f$ be boolean. Consider the following protocol where Alice and Bob execute $|A|$ AND black boxes, one for each $a \in A$. Alice inputs 1 iff $x = a$ and 0 otherwise and Bob inputs $f(a, y)$ for each black box. Clearly in the case when the black box for $x$ is executed Alice gets the right answer which it can output. Note that since there is no communication Bob gets no information and also Alice only gets to know $f(x, y)$ which it is supposed to know (the rest of the AND boxes all result in 0).

$\square$
We now prove a matching lower bound in case of Boolean functions. Before stating the claim we state a small caveat that the claim holds for functions with at least one column with all 0’s, however this assumption is without loss of generality as we can rename all rows such that output corresponding to a column is 0, and since this is the renaming of the rows Alice knowing the input row can always ‘un-rename’ the output.

**Theorem 2.** Let \( f : A \times B \rightarrow C \) be a Boolean function such that all columns of \( M_f \) are equivalent and all rows are distinct and there is some \( y_0 \) such that all for all \( x \in A, f(x, y_0) = 0 \). Then every deterministic protocol computing \( f \) securely must use at least \(|A| \) ANDS.

**Proof.** For any deterministic protocol computing \( f \) securely we can assume that there is no communication and hence Alice’s view is simply the outputs of the black boxes and its own input. Since \( f \) is boolean for every \( x \) there are only two views Alice can see one for all \( y \) such that \( f(x, y) = 0 \) and the other for all \( y \) such that \( f(x, y) = 1 \). Now the only place where the view can differ is the black box call so there must exist one call where the output is different for these two views. Call that call the significant call as Alice can discern the output value by just looking at this call.

We will now prove that for two \( x, x' \) the significant call cannot be the same. Assume for contradiction that for \( x, x' \) the significant call is indeed the same. Note that by our assumptions \( f(x, y_0) = f(x', y_0) = 0 \) and there exists some \( y' \) such that \( (x, y') = 0, f(x', y') = 1 \). Now since Bob sees nothing it simply obliviously puts values in the black boxes depending upon its input. This implies that for the significant call for \( x, x' \) it must put the same value otherwise Alice would learn about Bob’s input when she is holding \( x \) but if Bob puts the same value then Alice cannot discern the function at this call when she is holding \( x' \) and hence there is a contradiction.

This implies that for all \( x \) the significant call is different and this implies that there are at least \(|A| \) black boxes used.

It’s interesting to note that the trivial upper bound was proved by using a protocol that was non-adaptive and in the claim above we prove that adaptiveness does not help in case of Boolean functions. However as the authors show via an example in the paper this is not the case with non-Boolean functions i.e. in case of non-Boolean functions we can do better than the trivial upper bound by using an adaptive strategy.

### 6 Randomized Protocols

As commented before we know that the circuit complexity of a function provides an upper bound for the number of ANDs, but in case the circuit is of a nice form we can hope to do much better.

The question that we wish to answer now is that how much can randomization help as compared to deterministic protocols. We see in two successive claims that the gap in the two could be exponential in certain cases however the gap is at max exponential. Therefore the two following claims provide a complete characterization of the gap between randomness and determinism.

**Claim 4.** There is a function such that any deterministic protocol requires at least \( 2^n \) ANDs to compute it and there is a deterministic protocol that does it in those many ANDs. Also the function has a randomized protocol for computing it which requires \( \leq 8*n \) ANDs.

**Proof.** Consider the equality function \( EQ_n : \{0,1\}^n \times \{0,1\}^n \rightarrow 1 \) which outputs 1 iff the inputs are equal. Since \( M_{EQ} \) is the identity matrix we see that by Theorem 2 and Theorem 1 deterministic protocol requires at least \( 2^n \) ANDs to compute it and there is a deterministic protocol that does it in those many ANDs.

Now to prove the upper bound on randomized protocols consider the circuit for equality which is essentially a binary tree with \( 2^n \) leaves. Alternate leaves being inputs \( x_i, y_i \) and each internal node an AND gate. This circuit computes equality and has size \( 2n \). Therefore the number of ANDs required to evaluate this circuit by a randomized protocol is \( \leq 8n \)

The above claim establishes an exponential lower bound on the gap between deterministic and randomized protocols. We now prove that this is in fact tight for Boolean functions.
Theorem 3. Let \( f : A \times B \to C \) be a Boolean function such that all columns of \( M_f \) are equivalent and all rows are distinct and there is some \( y_0 \) such that for all \( x \in A, f(x, y_0) = 0 \). Then the number of calls to the AND black box in any perfectly-secure randomized protocol computing \( f \) is at least \( \log(|A|) \).

Proof. Fix a communication transcript \( c \) which has non-zero probability for some \( x', y' \). Note that by claim 2 we have that for any \( x, y \) this transcript has non-zero probability. Now fix Bob’s input to be \( y_0 \) where \( y_0 \) is as defined above. There will exist some \( r_B \) which agrees with \( y_0, c \). Pick any such \( r_B \). Now for any input \( x \) of Alice, there must exist some \( r_{A,x} \) which agrees with \( x, c \) and the answers of the black boxes when Bob puts his bits according to \( y_0, r_B \). The above is true because of Alice’s privacy.

Now suppose that Alice holds \( x, r_{A,x} \) and Bob holds \( y, r_B \). Let \( a_x \) be the sequence of values that Alice puts in the black boxes in this execution. We claim that for any \( x, x' \), \( a_x \neq a_{x'} \). Assume for contradiction there is some \( x, x' \) such that \( a_x = a_{x'} \).

Note that by our assumptions \( f(x, y_0) = f(x', y_0) = 1 \) and there exists some \( y' \) such that \( (x, y') = 0, f(x', y') = 1 \). Now by Bob’s privacy since the output of \( f(x, y_0) = f(x, y') = 0 \) there must be some string \( r_B' \) such that the view of Alice is the same in the case when Bob holds \( y_0, r_B \) or when he holds \( y', r_B' \).

Now consider the execution where Alice holds \( x' \) and \( r_{A,x'} \) and Bob holds \( y_1 \) and \( r_B' \). We claim that in this execution Alice sees exactly the same information as in the execution in which she holds \( x' \) and \( r_{A,x'} \) and Bob holds \( y_0 \) and \( r_B \). This is true since Alice has the same inputs in both the cases and Bob behaves the same in both executions since he cannot distinguish these executions from the executions in which Alice holds \( x \) and \( r_{A,x} \). As \( a_x = a_{x'} \) and in the calls where Alice puts 1 into the AND box, Bob puts the same values for \( y_0 \) and \( y' \). Thus Alice cannot distinguish between the two cases although \( f(x', y_0) \neq f(x', y') \) contradicting the perfect correctness of the protocol.

This implies that for every \( x \) there is a unique \( a_x \) which implies that the number of ANDs more than \( \log(A) \).

7 Conclusion

In this survey we dealt with the problem of quantifying the complexity of secure computation of functions in terms of the AND black boxes required. We specifically showed that any function can be computed with deterministic protocols in exponential number of ANDs and for Boolean functions this is infact the minimum required. We then dealt with the case of Randomized protocols and showed there exists a function such that the gap between deterministic versus randomized secure evaluation have an exponential gap in them. We further showed that the above is in fact a tight example for Boolean functions since in the case of Boolean functions the gap can at best be exponential. The last result also provides a linear lower bound on the \( \text{AND} \) complexity of boolean functions. Constraining this with \( \text{CIRCUIT} \) complexity we see that improving this bound even slightly would give us super-linear circuit bounds.

References


---

1I have copied this paragraph of the proof verbatim from the paper because I think I have a doubt whether this is correct or not. It would be great if you could discuss it with me.