Electronic Colloquium on Computational Complexity, Report No. 141 (2005)

Private Approximation of Search Problems[*]

Amos Beimel[†] Paz Carmi Kobbi Nissim Enav Weinreb

Department of Computer Science
Ben-Gurion University, Beer-Sheva, Israel
Email: {beimel,carmip,kobbi,weinrebe}@cs.bgu.ac.il

November 29, 2005

Abstract

Many approximation algorithms have been presented in the last decades for hard search problems. The focus of this paper is on cryptographic applications, where it is desired to design algorithms which do not leak unnecessary information. Specifically, we are interested in private approximation algorithms – efficient algorithms whose output does not leak information not implied by the optimal solutions to the search problems. Privacy requirements add constraints on the approximation algorithms; in particular, known approximation algorithms usually leak a lot of information.

For functions, [Feigenbaum et al., ICALP 2001] presented a natural requirement that a private algorithm should not leak information not implied by the original function. Generalizing this requirement to search problems is not straight forward as an input may have many different outputs. We present a new definition that captures a minimal privacy requirement from such algorithms – applied to an input instance, it should not leak any information that is not implied by its *collection of exact solutions*. Although our privacy requirement seems minimal, we show that for well studied problems, as vertex cover and maximum exact 3SAT, private approximation algorithms are unlikely to exist even for poor approximation ratios. Similar to [Halevi et al., STOC 2001], we define a relaxed notion of approximation algorithms that leak (little) information, and demonstrate the applicability of this notion by showing near optimal approximation algorithms for maximum exact 3SAT which leak little information.

Key words. Secure computation, Private approximation, Solution-list algorithms, Vertex cover.

[*]This work was partially supported by the Frankel Center for Computer Science.
[†]On sabbatical at the University of California, Davis.

ISSN 1433-8092

1 Introduction

Approximation algorithms are currently one of the main research fields in computer science. The design of algorithms for approximating computationally hard problems has attracted substantial attention in the last few decades, as did the research on proving hardness of approximation. Frequently, approximation algorithms are applied to sensitive data, as in the distributed cryptographic setup of secure computation. In this paper we study privacy issues related to approximation algorithms of search problems.

We consider an abstract client-server setting. This scenario, besides being interesting on its own sake, is important since the multiparty distributed setting can be reduced to this model using secure function evaluation protocols [21, 8]. In the client-server setting, the server $\mathcal{S}$ is willing to let the client $\mathcal{C}$ learn a specific functionality f of its input. The standard requirement in this setting is that no other information would be leaked to $\mathcal{C}$. For concreteness, assume that $\mathcal{S}$ holds a graph G and consider the following examples:

1. If $f(G)$ is the diameter of G then $\mathcal{S}$ can simply compute $f(G)$ and send it to $\mathcal{C}$. It is clear that $\mathcal{C}$ learns $f(G)$, but no other information about G.

2. If $f(G)$ is the number of perfect matchings in G, then $\mathcal{C}$ and $\mathcal{S}$ have to settle for an approximation $\hat{f}$. This raises the question of which approximation to use, as $\mathcal{S}$ is only willing to reveal $f(G)$, but $\hat{f}$ may leak some other information. This was the setting in the work by Feigenbaum et al. [6]. They defined the notion of *private approximation* that combines two requirements: (i) *approximation:* $\hat{f}$ is an approximation to f, and (ii) *functional privacy:* $\hat{f}(G)$ can be simulated given $f(G)$; In particular, functional privacy implies that if $f(x) = f(y)$ then $\hat{f}(x)$ and $\hat{f}(y)$ are indistinguishable. It turns out that an efficient private approximation algorithm exists for the number of perfect matchings in a graph [6].

3. A more general and more typical case is when $\mathcal{C}$ needs to learn a "solution" for an optimization problem, rather than its optimization objective. For example, one is usually interested in finding a vertex cover of minimum size in G, rather than just learning the size of the optimal vertex cover. However, even if $\mathcal{C}$ and $\mathcal{S}$ are willing to settle for an approximation (i.e. finding a cover which is not much larger than the minimum), it is not clear which cover $\mathcal{C}$ should learn – the framework of private approximations of functions does not address this problem as there may be many solutions to the search problem.

This more general case where one seeks not a computation of a function, but of a solution to a computational problem is the focus of this work.

The generalization of the definition of (functional) private approximation to search problems is not straight forward. As one input may have many different outputs, it is not clear in which cases we need $\hat{f}(x)$ and $\hat{f}(y)$ to be indistinguishable. For our example of vertex cover, it seems that a minimal requirement is that a private approximation algorithm should not distinguish between graphs G_1, G_2 that are equivalent in the sense that they have exactly the same set of solutions, i.e., every minimum cover for G_1 is also a minimum cover for G_2 and vice versa. Note that this is a rather weak requirement, as it does not restrict the approximation algorithm with respect to non-equivalent graphs. Furthermore, as the number of equivalence classes of graphs is exponential in $|V|$, there may be many equivalence classes that contain only a few graphs. This is in sharp contrast with functional privacy that divides the graphs to only $|V|$ equivalence classes – as there are $|V|$ possible answers to the functional problem.

Nevertheless, we show that vertex cover cannot be approximated privately even to an approximation ratio as poor as $n^{1-\epsilon}$. A similar result is shown for Max Exact 3 SAT. This means that although our notion of privacy is minimal and it seems that any reasonable notion of privacy for search problems should imply it, there are natural problems for which it is too strong.

Our proof techniques for the impossibility results are different from those used for obtaining the inapproximability results for the functional version of the problem [12]. All our lower bounds have the same structure, where a solution is constructed in an iterative manner. In each iteration a node is examined. If that node appears in some optimal solution, we add it to the solution and remove it and its neighbors from the graph. If there exists some optimal solution that does not include this node, we remove it from the graph. When both conditions are met, we chose one of them arbitrarily. The crux of the algorithm is that the private approximation algorithm is used for making the decision.

In view of the impossibility results, it is natural to seek for a relaxation of the definition. In the setting of functional privacy, Halevi et al. [12] defined the notion of *almost private algorithms* that are allowed to leak (little) information beyond what is leaked by the exact functionality. This notion falls elegantly within our definitional framework, where it is generalized to search problems. We say that an algorithm leaks at most k bits if it refines each equivalence class by dividing it to at most 2^k sub-classes. For Max Exact 3 SAT, this relaxation results in a tremendous improvement, and there exists an approximation algorithm for Max Exact 3 SAT with a near optimal approximation ratio of $7/8 - \epsilon$ that leaks only $\log \log n$ bits of information.

Interestingly, these algorithms for vertex cover and Max Exact 3 SAT fall into a class of approximation algorithms that we call *solution-list algorithms*, and provide a much stronger privacy guarantee. Intuitively, for every input size n, the solution-list algorithm deterministically computes a list of possible outcomes. Upon seeing the actual input, the algorithm is restricted to output a solution from the list. For Max Exact 3 SAT, our algorithm efficiently computes a list of length $O(\log n)$ assignments such that for every exact 3 CNF formula ϕ there exists an assignment in the list that is a good approximation for ϕ, hence a leakage of $O(\log \log n)$ bits. For vertex cover, we obtain a solution-list algorithm that is also significant, but not as dramatic as for Max Exact 3 SAT, and we obtain a tradeoff between the amount of leakage and approximation quality – the multiplication of these quantities is roughly n.

We also show an impossibility result for approximating vertex cover while leaking at most $O(\log n)$ bits. This suggests that the solution-list algorithm for the problem may be optimal. However, there is still an exponential gap between the lower and upper bounds. Finding algorithmic techniques, possibly stronger than solution-list, for private approximation algorithms is an open problem.

The notion of private solutions is applicable also to search problems in $\mathcal{P}$. For many problems in $\mathcal{P}$, there is an efficient private algorithm solving the search problem. One option is choosing the lexicographically first exact solution (e.g., for maximum matching and shortest path). Another option is constructing a randomized algorithm that chooses a random exact solution according to some distribution. However, finding a solution privately imposes additional constrains on the algorithm. We show that these constrains can make the problem much harder. We present a search problem which can be easily solved without privacy constrains, but cannot be efficiently and privately solved unless $\mathcal{NP} \subseteq \mathcal{P}/\text{poly}$.

1.1 Related Work

Feigenbaum et al. [6] initiated the discussion of private approximation of functions. They observed that combining approximation algorithms and secure function evaluation protocols might result in protocols that are not private as the result of the approximation algorithm might leak information. The definition of functional privacy put forward by [6] is a simulation based definition, where the simulator's input is the exact value $f(x)$ and its output distribution is computationally indistinguishable from $\hat{f}(x)$. Under this definition, they provided a protocol for approximating the Hamming distance of two n-bit strings with communication complexity $\tilde{O}(\sqrt{n})$, and polynomial solutions for approximating the permanent and other natural $\#\mathcal{P}$ problems. Other private approximation protocols published since include [16, 7, 14], the latter providing a polylogarithmic communication approximation for the Hamming distance.

Inapproximability results for computing the size of a minimum vertex cover within approximation $n^{1-\epsilon}$ were proved by Halevi et al. [12]. Their proof uses a special *sliding-window* reduction that translates a SAT instance ϕ to an instance G of vertex cover such that if ϕ is satisfiable then G has a vertex cover of size z, and otherwise any vertex cover for G is of size at least $z + 1$. These techniques do not apply in our setting, as the large number of equivalence classes does not allow a simple averaging argument, as used in [12].

The notion of almost private approximation was introduced in [12]. Their definition modifies that of [6] by allowing the simulator to consult a deterministic predicate of the input. They showed that by this slight compromise in privacy, one can get fairly good approximations for any problem that admits a good deterministic approximation. For the functional version of vertex cover this yields an approximation ratio 4 (more generally, there is a tradeoff between the leakage and approximation ratio). A similar relaxation of privacy is the notion of additional information in secure two-party protocols by Bar-Yehuda et al. [3]. Related ideas can be found in the study of knowledge complexity [11, 10, 4, 9, 20].

Organization. In Section 2 we define private algorithms for search problems. In Section 3 we provide impossibility results for the minimum vertex cover and the maximum exact 3SAT search problems. In Section 4 we discuss algorithms that leak (little) information and describe such algorithms for both search problems. Later, in Section 5, we prove our strongest impossibility result, showing vertex cover cannot be privately approximated even if a leakage of $O(\log n)$ bits is allowed. Finally, in Section 6, we present an impossibility result for a search problem in $\mathcal{P}$.

2 Definition of Private Algorithms with respect to Privacy Structure

There are two different aspects of private algorithms – the utility of the algorithm (what should be computed) and the privacy requirement (what should be protected, that is, what information should not be revealed by the computation). In computing functions this is quite straight forward, we want to compute (or approximate) a function and we want to protect inputs with the same output. For search algorithms, we know what should be computed. However, since these algorithms may output different outputs on the same input, it is less clear what should be protected. We, thus, separate the specification of what we want to protect from what we compute. In general, we require the output of the algorithm on certain pairs of inputs to be indistinguishable. In this section we define the pairs of inputs that should be protected by a private algorithm.

Definition 2.1 (privacy structure) *A privacy structure $\mathcal{R} \subseteq \{0,1\}^* \times \{0,1\}^*$ is an equivalence relation on instances. For $\langle x, y\rangle \in \mathcal{R}$, we use the notation $x \equiv_{\mathcal{R}} y$.*

We will only discuss privacy structures of the form $\mathcal{R} = \cup_{n\in\mathcal{N}}\mathcal{R}_n$, where $\mathcal{R}_n$ is an equivalence relation between instances of size n, such as graphs on n vertices or Boolean formulae over n variables. We say that an algorithm $\mathcal{A}$ is *private* with respect to a privacy structure $\mathcal{R}$ if the results of executing $\mathcal{A}$ on two $\mathcal{R}$-equivalent inputs are computationally indistinguishable.

Definition 2.2 (Private Algorithm) *Let $\mathcal{R}$ be a privacy structure. A probabilistic polynomial time algorithm $\mathcal{A}$ is* private *with respect to $\mathcal{R}$ if for every polynomial-time algorithm $\mathcal{D}$ and for every positive polynomial $p(\cdot)$, there exists some $n_0 \in \mathbb{N}$ such that for every $x, y \in \{0,1\}^*$ such that $x \equiv_{\mathcal{R}} y$ and $|x| = |y| \geq n_0$*

$$|\Pr[\mathcal{D}(\mathcal{A}(x), x, y) = 1] - \Pr[\mathcal{D}(\mathcal{A}(y), x, y) = 1]| \leq \frac{1}{p(|x|)}.$$

That is, when $x \equiv_{\mathcal{R}} y$, every algorithm $\mathcal{D}$ cannot distinguish if the input of $\mathcal{A}$ is x or y.

Example 2.3 Let $f : \{0,1\}^* \rightarrow \mathbb{N}$ be a function. Define $\mathcal{R}_f = \{\langle x, y\rangle : |x| = |y|, f(x) = f(y)\}$. The relation $\mathcal{R}_f$ is the relation implicitly considered when discussing private computation of functions.

We next recall the definition of search problem and define the privacy structure associated with it.

Definition 2.4 *A bivariate relation Q is polynomially-bounded if there exists a constant c such that $|w| \leq |x|^c$ for every $\langle x, w\rangle \in Q$. The* decision problem *for Q is, given an input x, decide if there exists a w such that $\langle x, w\rangle \in Q$ or not. The* search problem *for Q is, given an input x,* find *a w such that $\langle x, w\rangle \in Q$ if such w exists.*

Search problems are the more common algorithmic task of finding a solution to a problem (rather than deciding whether the problem has a solution or not). To define private solution or private approximation of a search problem, one must determine the privacy structure the algorithm should respect. It is a *minimal* requirement to demand that if two inputs have the same set of answers to the search problem, the approximation algorithm should not enable to distinguish between them.

Definition 2.5 (Privacy Structure of a Search Problem) *The privacy structure $\mathcal{R}_Q$ related to a relation Q is defined as follows: $x \equiv_{\mathcal{R}_Q} y$ iff*

- *$|x| = |y|$, and*
- *$\langle x, w\rangle \in Q$ iff $\langle y, w\rangle \in Q$ for every w.*

That is, $x \equiv_{\mathcal{R}_Q} y$ if they have the same set of solutions.

We give two examples of privacy structures, for specific relations, that would be the focus of this paper.

Example 2.6 Let minVC be the minimum vertex cover relation, that is, $\langle G, C\rangle \in$ minVC if C is a minimum vertex cover in G. In this case, $\mathcal{R}_{\text{minVC}}$ contains all pairs of graphs $G_1 = \langle V, E_1\rangle, G_2 = \langle V, E_2\rangle$ for which $C \subseteq V$ is a minimum vertex cover for G_1 iff it is a minimum vertex cover for G_2.

Example 2.7 An exact 3CNF formula is a CNF formula that contains exactly three different literals in each clause. Let maxE3SAT be the maximum exact three SAT relation, that is, $\langle \phi, a\rangle \in$ maxE3SAT if ϕ is an exact 3CNF formula over n variables, and a is an assignment to the n variables that satisfies the maximum number of clauses in ϕ. In this case, the privacy structure $\mathcal{R}_{\text{maxE3SAT}}$ contains all pairs of exact 3CNF formulae ϕ_1, ϕ_2 over n variables for which an assignment a satisfies the maximum number of clauses in ϕ_1 iff it satisfies the maximum number of clauses in ϕ_2.

3 Impossibility Results for Private Approximation

3.1 Impossibility Results for Private Approximation of Vertex Cover

In this section we show that private approximation of the vertex cover search problem with respect to $\mathcal{R}_{\text{minVC}}$ (defined in Example 2.6) is a hard task. We start with defining private approximation of vertex cover, and then prove impossibility results for both the deterministic and the randomized settings.

Definition 3.1 (Private Approximation of Vertex Cover) *An algorithm $\mathcal{A}$ is a* private $c(n)$-approximation *algorithm for* minVC *if: (i) $\mathcal{A}$ runs in polynomial time. (ii) $\mathcal{A}$ is a $c(n)$-approximation algorithm for* minVC, *that is, for every graph G with n vertices, it returns a vertex cover whose size is at most $c(n)$ times the size of the smallest vertex cover of G. (iii) $\mathcal{A}$ is private with respect to $\mathcal{R}_{\text{minVC}}$.*

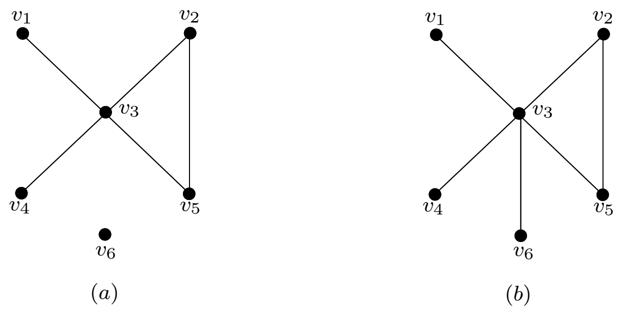

Figure 1: A pair of graphs equivalent under $\mathcal{R}_{\text{minVC}}$.

To illustrate our definitions, we present a private $(n/\log n)$-approximation algorithm for the vertex cover problem. This algorithm is based on the polynomial algorithm of [19] that returns a minimum vertex cover if the size of the vertex cover is at most $\log n$. Actually, in this case there are at most n such covers, and the algorithm can efficiently compute all of them. Thus, we can define any rule to choose one of them (e.g., the lexicographically first or uniform distribution). To approximate minVC we do the following:

If there is a cover of size at most $\log n$, return the lexicographically first minimum vertex cover. Otherwise, return the entire set of vertices.

We show impossibility results for privately approximating vertex cover in the deterministic and in the randomized setting.

Theorem 3.2 *Let $\epsilon > 0$ be a constant. If $\mathcal{P} \neq \mathcal{NP}$, then there is no deterministic private $n^{1-\epsilon}$-approximation algorithm for the search problem of* minVC.

Theorem 3.3 *Let $\epsilon > 0$ be a constant. If $\mathcal{RP} \neq \mathcal{NP}$, then there is no randomized private $n^{1-\epsilon}$-approximation algorithm for the search problem of* minVC.

3.1.1 Relevant And Critical Vertices

The framework for proving Theorems 3.2 and 3.3 is the following: We assume the existence of the appropriate private approximation algorithm and derive a greedy algorithm that solves vertex cover *exactly*. The following definitions are central for both the deterministic and the randomized case.

Definition 3.4 (Critical Vertices and Relevant Vertices) *Let $G = \langle V, E\rangle$ be a graph and $v \in V$ be a vertex of G. We say that v is* critical *for G if* every *minimal vertex cover of G contains v. We say that v is* relevant *for G if there* exists *a minimal vertex cover of G that contains v.*

Observation 3.5 *Every vertex is relevant or non-critical (or both).*

Example 3.6 To illustrate the relation $\mathcal{R}_{\text{minVC}}$ we show a pair of graphs that are equivalent under the relation. One way to create such a pair is to pick a graph and identify a vertex that is critical for this graph. For example, vertex v_3 in Figure 1(a) is a critical vertex. To get the second graph we connect the critical vertex to some other vertex. In Figure 1(b), vertex v_3 is connected to v_6. It is easy to verify that the set of minimum covers in both graphs is $\{\{v_3, v_2\}, \{v_3, v_5\}\}$. The equivalence can also be derived from Claim 3.9 below.

We reduce the design of a greedy algorithm for vertex cover, to solving the following problem.

Definition 3.7 (The Relevant / Non-Critical Problem) *Input: A graph $G = \langle V, E\rangle$ and a vertex $v \in V$. Output: One of the following: (i) v is relevant for G. (ii) v is non-critical for G.*

```
Algorithm Greedy Vertex Cover

INPUT: A graph $G = \langle V, E\rangle$.
OUTPUT: A minimal vertex cover of $G$.

  1. If $V = \emptyset$ return $\emptyset$.
  2. Pick a vertex $v \in V$ and execute Algorithm Relevant-Non-Critical on $G$ and $v$.
  3. If the answer is "RELEVANT":
     (a) Run Greedy Vertex Cover on the graph $G \setminus \{v\}$. Denote the answer by $C_v$.
     (b) Return $C_v \cup \{v\}$.
  4. If the answer is "NON-CRITICAL":
     (a) Let $N(v)$ be the neighbors of $v$ in $G$.
     (b) Run Greedy Vertex Cover on the graph $G\setminus(\{v\}\cup N(v))$. Denote the answer by $C_{N(v)}$.
     (c) Return $C_{N(v)} \cup N(v)$.
```

Figure 2: A greedy algorithm using Algorithm `Relevant-Non-Critical Vertex` to fi nd a minimum vertex cover.

In Figure 2, we describe a greedy algorithm for vertex cover given an access to an algorithm that solves the Relevant / Non-Critical problem. In subsequent sections, we solve the latter using oracle access to private approximation algorithms for vertex cover. The following claim asserts the correctness of the greedy algorithm.

Claim 3.8 *If Algorithm* `Relevant-Non-Critical` *is polynomial and correct then Algorithm* `Greedy Vertex Cover` *is polynomial and correct.*

Proof: The proof is by induction on $|V|$. The algorithm is trivially correct for $V = \emptyset$. Now suppose $V \neq \emptyset$, and we start from the case where v is relevant. In this case, there is a minimal cover C for G that contains v. Denote $d = |C|$, and note that the set $C\setminus\{v\}$ is a cover of size $d-1$ for the graph $G\setminus\{v\}$. By the induction hypothesis, the set C_v is a minimal cover for $G\setminus\{v\}$. We claim that $C_v \cup \{v\}$ is a minimal cover for G. First note that it is indeed a cover of G as any edge adjacent to v is covered by v, and all the rest are covered by C_v. Since C_v is a minimal cover of $G\setminus\{v\}$, it is of size at most $d-1$. Therefore, the size of $C_v \cup \{v\}$ is d, and it is a minimal cover of G.

In the other case, v is not critical for G. Thus, there is a minimal cover C' for G that does not contain v, and, therefore, contains all vertices in $N(v)$. Denote $d' = |C'|$ and $h = |N(v)|$. In this case the set $C'\setminus N(v)$ is a cover of size $d'-h$ of the graph $G\setminus(N(v)\cup\{v\})$. By the induction hypothesis, the set $C_{N(v)}$ is a minimal cover for $G\setminus N(v)$. We claim that $C_{N(v)} \cup N(v)$ is a minimal cover for G. First note that it is indeed a cover of G as any edge adjacent to $N(v)\cup\{v\}$ is covered by $N(v)$, and all other edges are covered by $C_{N(v)}$. Since $C_{N(v)}$ is a minimal cover of $G\setminus N(v)$, it is of size at most $d'-h$. Therefore, the size of $C_{N(v)} \cup N(v)$ is d', and it is a minimal cover of G.

It is straight forward to verify that if `Relevant-Non-Critical` is polynomial, then the greedy algorithm is polynomial. □

3.1.2 Combinatorial Claims

The following combinatorial claims will be helpful in designing algorithms for the Relevant / Non-Critical problem in both the deterministic and the randomized settings. Intuitively, a private approximation algorithm must be "sensitive" to small changes in the set of minimum vertex covers of its input graph. We study the connection between the $\mathcal{R}_{\text{minVC}}$ relation and the set of critical and relevant vertices in a graph.

Claim 3.9 *Let $G = \langle V, E\rangle$ be a graph, $u, v \in V$ such that $(u, v) \notin E$, and $G^* = \langle V, E^*\rangle$, where $E^* = E \cup (u, v)$. If u is critical for G, then $G \equiv_{\mathcal{R}_{\text{minVC}}} G^*$.*

Proof: We first show that every minimum vertex cover of G is a minimum vertex cover of G^*. Let C be a minimal cover of G. As u is critical for G, we get that $u \in C$. Therefore, C covers the edge (u, v) and thus it is a cover of G^*. Note that every cover of G^* is also a cover of G, and thus C is a minimal cover of G^*.

For the other direction, let C^* be a minimal cover of G^*. Let c be the size of a minimal cover of G. As u appears in at least one minimal cover of G, which is also a cover of G^*, the size of C^* is at most c. On the other hand, as $E \subseteq E^*$, the set C^* is also a cover of G and thus the size of C^* is exactly c. Therefore, C^* is a minimal cover of G. □

We will later see that if a vertex u is not in the result of the private approximation algorithm $\mathcal{A}$, then it is non-critical for the input graph G. However, if the vertex cover size of the input graph is large, the approximation algorithm may return the entire set V as its result. To avoid this, we add a large set of isolated vertices to G. (The size of this set is a function of the approximation ratio.) The mere fact that an isolated vertex is non-critical for G is of-course not helpful. Nevertheless, we gain information by connecting this isolated vertex to the vertex v and running $\mathcal{A}$ on the new graph. It will also be helpful to consider duplicating the graph G and connecting the isolated vertex to both copies of the original vertex v.

Definition 3.10 (The Graphs G_2 and $G_{\underset{\wedge}{i}}$) *Let $G = \langle V, E\rangle$ be a graph, $v \in V$ be a vertex, I be a set of vertices, and $i \in I$. The graph G_2 is defined as $G_2 = \langle V_2, E_2\rangle$ where $V_2 \stackrel{def}{=} (V \times \{1, 2\}) \cup I$ and $E_2 \stackrel{def}{=} \{(\langle u, j\rangle, \langle w, j\rangle) : (u, w) \in E, j \in \{1, 2\}\}$. The graph $G_{\underset{\wedge}{i}}$ is defined as $G_{\underset{\wedge}{i}} = \langle V_2, E_{\underset{\wedge}{i}}\rangle$ where $E_{\underset{\wedge}{i}} \stackrel{def}{=} E_2 \cup \{(\langle v, j\rangle, i) : j \in \{1, 2\}\}$.*

The graphs G_2 and $G_{\underset{\wedge}{i}}$ are illustrated in Figure 3. The following claims summarize the properties of G_2 and $G_{\underset{\wedge}{i}}$.

Claim 3.11 *If v is critical for G, then $G_2 \equiv_{\mathcal{R}_{\text{minVC}}} G_{\underset{\wedge}{i}}$.*

Proof: Since G_2 contains two separate copies of G and v is critical for G, the vertices $\langle v, 1\rangle$ and $\langle v, 2\rangle$ are critical for G_2. Hence, by Claim 3.9, adding the edges $(\langle v, 1\rangle, i)$ and $(\langle v, 2\rangle, i)$ does not change the set of minimal vertex covers of the graph. □

Claim 3.12 *If v is not relevant for G then i is critical for $G_{\underset{\wedge}{i}}$.*

Proof: Assume towards contradiction that i is non-critical for $G_{\underset{\wedge}{i}}$. Hence, there must be a minimal cover $C_{\underset{\wedge}{i}}$ of $G_{\underset{\wedge}{i}}$ that contains $\langle v, 1\rangle$ and $\langle v, 2\rangle$. The intersection of $C_{\underset{\wedge}{i}}$ with each copy of G contains a cover of G that contains the appropriate copy of v. As v is not relevant for G, these covers are not optimal. Let c be the size of a minimum vertex cover of G. Then $\left|C_{\underset{\wedge}{i}}\right| \geq 2(c + 1) = 2c + 2$. On the other hand, let C be a minimum cover of G of size c. Then the set $(C \times \{1, 2\}) \cup \{i\}$ is a cover of $G_{\underset{\wedge}{i}}$ of size $2c + 1$, in contradiction to the minimality of $C_{\underset{\wedge}{i}}$. Hence, i is critical for $G_{\underset{\wedge}{i}}$. □

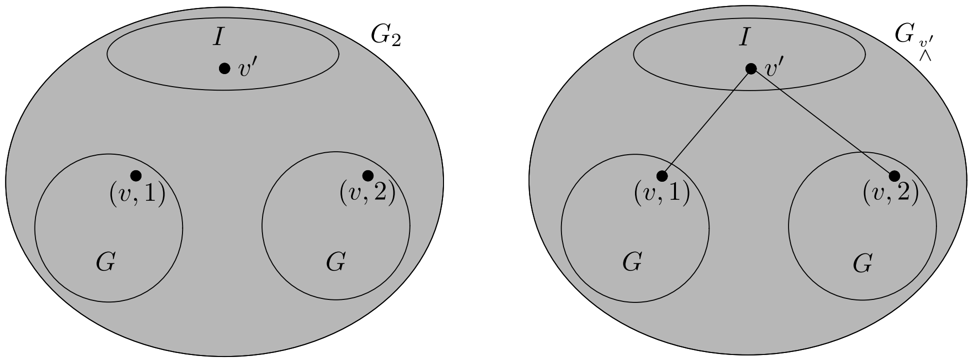

Figure 3: The graphs G_2 and $G_{\underset{\wedge}{v'}}$.

3.1.3 Impossibility Result for Deterministic Private Approximation

In this section we show an algorithm that solves the Relevant / Non-Critical problem, given an oracle access to a deterministic private approximation algorithm for minVC.

Claim 3.13 *Let $\mathcal{A}$ be a deterministic private approximation algorithm for* minVC, *let $G = \langle V, E\rangle$ be a graph, and denote $W = \mathcal{A}(G)$. Then for any two different vertices $v_1, v_2 \in V \backslash W$, the vertex v_1 is not critical for G (and, similarly, v_2 is not critical).*

Proof: As v_1 and v_2 are not in W, and W is a cover of G, we infer that $(v_1, v_2) \notin E$. Let $E^* = E \cup \{(v_1, v_2)\}$, and define $G^* = \langle V, E^*\rangle$. We now consider a hypothetical execution of the algorithm $\mathcal{A}$ on G^* and denote $W^* = \mathcal{A}(G^*)$. The set W^* must cover the edge (v_1, v_2) and thus $W^* \neq W$.

If v_1 is critical for G, then, by Claim 3.9, the sets of minimum-size vertex cover of G and G^* are equal. However, since $\mathcal{A}$ is deterministic and private, and $W \neq W^*$, the set of minimal vertex covers of G and G^* must be different. Therefore, v_1 is not critical for G. □

In Figure 4, we present the algorithm `Relevant-Non-Critical`. It takes a graph $G = \langle V, E\rangle$ and a vertex $v \in V$ as inputs, and uses a private $n^{1-\epsilon}$-approximation algorithm $\mathcal{A}$ to solve the Relevant / Non-Critical problem.

The correctness of the algorithm `Relevant-Non-Critical` steams from the following claims:

Claim 3.14 *If $W_2 \neq W_{\underset{\wedge}{v'}}$, then v is not critical for G.*

Proof: Assume towards contradiction that v is critical for G. By Claim 3.11, the graphs G_2 and $G_{\underset{\wedge}{v'}}$ have the same set of minimal vertex covers. Hence, from the privacy of $\mathcal{A}$, we get that $W_2 = \mathcal{A}(G_2) = \mathcal{A}(G_{\underset{\wedge}{v'}}) = W_{\underset{\wedge}{v'}}$, contradicting $W_2 \neq W_{\underset{\wedge}{v'}}$. □

Claim 3.15 *If $W_2 = W_{\underset{\wedge}{v'}}$, then v is relevant for G.*

Proof: As $W_2 = W_{\underset{\wedge}{v'}}$, and $v' \notin W_2$, we get $v' \notin W_{\underset{\wedge}{v'}}$. Hence, by Claim 3.13, the vertex v' is not critical for $G_{\underset{\wedge}{v'}}$. Applying Claim 3.12, we infer that v is relevant for G. □

To conclude the proof of Theorem 3.2, we show that there is a vertex we can choose in step (4) of the algorithm.

```
Algorithm Relevant-Non-Critical
INPUT AND OUTPUT: See Definition 3.7.

1. Let $I$ be a set of vertices of size $(4n)^{1/\epsilon} - 2n$.
2. Construct the graph $G_2$ from $G$ and $I$ as in Definition 3.10.
3. Execute $\mathcal{A}$ on $G_2$ and denote $W_2 = \mathcal{A}(G_2)$.
4. Choose any vertex $v' \in I \backslash W_2$ (there must be at least two such vertices as the approximation algo-
   rithm cannot return all the set $I$).
5. Construct $G_{\underset{\wedge}{v'}}$ from $G$, $I$, and $v'$ as in Definition 3.10.
6. Execute $\mathcal{A}$ on $G_{\underset{\wedge}{v'}}$ and denote $W_{\underset{\wedge}{v'}} = \mathcal{A}(G_{\underset{\wedge}{v'}})$.
7. If $W_2 \neq W_{\underset{\wedge}{v'}}$ return "NOT CRITICAL." Else return "RELEVANT."
```

Figure 4: Algorithm `Relevant-Non-Critical Vertex` for a private deterministic $\mathcal{A}$.

Claim 3.16 *Let $\epsilon > 0$. There is a vertex $v' \in I$ such that $v' \in I \setminus W_2$.*

Proof: Let $N = |I| + 2n = (4n)^{1/\epsilon}$ be the number of vertices in G_2. The size of the minimum vertex cover of G_2 is twice the size of the minimum vertex cover of G, thus, it is at most $2n$. Since $\mathcal{A}$ is an $N^{1-\epsilon}$-approximation algorithm for vertex cover, the size of $\mathcal{A}(G_2)$ is at most

$$2n \cdot N^{1-\epsilon} \;=\; 2n \cdot ((4n)^{1/\epsilon})^{1-\epsilon} \;=\; \frac{(4n)^{1/\epsilon}}{2} \;<\; (4n)^{1/\epsilon} - 2n \;=\; |I|.$$

Consequently, there is at least one vertex $v' \in I \setminus W$. □

In Appendix A, we prove Theorem 3.3, that is, we generalize the impossibility results to randomized private algorithms. We also extend the techniques described in this section to get an impossibility result with respect to a weaker notion of private approximation defi ned in Section 4 (see Theorem 4.9).

3.2 Impossibility Results for Private Approximation of Max E3SAT

Similar results are obtained for private approximation of Max E3SAT as stated in the following theorem:

Theorem 3.17 *Let $\epsilon > 0$ be a constant.*

1. *If $\mathcal{P} \neq \mathcal{NP}$, then there is no* deterministic *private $1/n^{1-\epsilon}$-approximation algorithm for the search problem of* maxE3SAT.
2. *If $\mathcal{RP} \neq \mathcal{NP}$, then there is no* randomized *private $1/n^{1-\epsilon}$-approximation algorithm for the search problem of* maxE3SAT.

See Appendix B for a proof of the deterministic case. The proof of the probabilistic case follows, with similar modifi cations as in Appendix A.

4 Algorithms that Leak Little Information

As demonstrated in the previous section, in some cases it is impossible to design an efficient algorithm which is private with respect to a privacy structure $\mathcal{R}$. However, letting $\mathcal{R}'$ be a refinement of $\mathcal{R}$, we view a private algorithm with respect to $\mathcal{R}'$ as a private algorithm with respect to $\mathcal{R}$ that *leaks* information. The amount of information leaked is quantified according to the relation between $\mathcal{R}$ and $\mathcal{R}'$.

Definition 4.1 (k-Refinement) *Let $\mathcal{R}$ and $\mathcal{R}'$ be two privacy structures over $\{0,1\}^*$. We say that $\mathcal{R}'$ is a k-refinement of $\mathcal{R}$ if $\mathcal{R}' \subseteq \mathcal{R}$ and every equivalence class of $\mathcal{R}$ is a union of at most 2^k equivalence classes of $\mathcal{R}'$.*

Definition 4.2 *Let $\mathcal{R}$ be a privacy structure. A probabilistic polynomial time algorithm $\mathcal{A}$* leaks at most k bits *with respect to $\mathcal{R}$ if there exists a privacy structure $\mathcal{R}'$ such that (i) $\mathcal{R}'$ is a k-refinement of $\mathcal{R}$, and (ii) $\mathcal{A}$ is private with respect to $\mathcal{R}'$.*

4.1 Solution-List Algorithms

Solution-list algorithms are algorithms whose outcome is always in a small predetermined set. With respect to privacy, solution-list algorithms are valuable as they leak only a few bits with respect to any privacy structure – at most logarithmic in the number of their possible outcomes.

Definition 4.3 (Solution-List Algorithm) *We say that a deterministic algorithm $\mathcal{A}$ is a $K(n)$*-solution-list *algorithm if for every $n \in \mathbb{N}$*

$$|\{\, y : \exists x \in \{0,1\}^n \text{ such that } \mathcal{A}(x) = y \,\}| \le K(n).$$

I.e. a solution-list algorithm is an algorithm such that for every input size n "chooses" its outputs from a set of at most $K(n)$ possible outcomes.

We define the universal relation, denoted $\mathcal{U}^* = \cup_{n \in \mathcal{N}} \mathcal{U}^*_n$, as the privacy structure where every two instances of the same size are equivalent. Note that any privacy structure is a refinement of $\mathcal{U}^*$, hence if an algorithm is private with respect to $\mathcal{U}^*$ it is also private with respect to any privacy structure,[1] and similarly, if an algorithm leaks at most k bits with respect to $\mathcal{U}^*$ then so is the case with respect to any privacy structure.

Observation 4.4 *Any $K(n)$-solution-list algorithm leaks at most $\log K(n)$ bits with respect to $\mathcal{U}^*$.*

4.2 Solution-List Algorithms for Max E3SAT

In this section we present a $(7/8 - \epsilon)$-approximation algorithm for maximum satisfiability on exact 3CNF formulae that leaks little information, i.e., $O(\log\log n)$ bits. The algorithm is a solution-lists algorithm as defined in Definition 4.3. The approximation in our algorithm is nearly optimal, as by the result of Håstad [13], there is no polynomial-time $(7/8 + \epsilon)$-approximation algorithm for this problem[2] (unless $\mathcal{P} = \mathcal{NP}$).

We start with a simple motivating example. Consider the following simple algorithm for the Max-SAT problem:

If 0^n satisfies at least half of the clauses in ϕ, then return 0^n. Otherwise, return 1^n.

[1] An algorithm $\mathcal{A}$ is private with respect to $\mathcal{U}^*$ if only the instance size may be learned from its outcome.

[2] Any $(7/8 + \epsilon)$-approximation solution-list algorithm that uses $\mathrm{poly}(n)$ solutions would imply $\mathcal{NP} \subseteq \mathcal{P}/\mathrm{poly}$ even if the list cannot be efficiently constructed.

For every clause, either 0^n or 1^n satisfy the clause. Thus, 0^n or 1^n satisfy at least half of the clauses in ϕ, and this is a $1/2$-approximation of Max-SAT. Since there are only two possible answers, this algorithm leaks at most one bit.

Claim 4.5 *There is a $7/8$-approximation algorithm for* maxE3SAT *that leaks at most $O(\log n)$ bits with respect to $\mathcal{R}_{\text{maxE3SAT}}$. Furthermore, for every $\epsilon > 0$, there is a $(7/8 - \epsilon)$-approximation algorithm for* maxE3SAT *that leaks at most $O(\log \log n)$ bits with respect to $\mathcal{R}_{\text{maxE3SAT}}$.*

Proof: We first describe the $7/8$-approximation algorithm. Towards this goal, we construct for every n a list of $\text{poly}(n)$ assignments such that for every exact 3CNF formula with n variables there is an assignment in the list that satisfies at least $7/8$ of the clauses of the formula. Furthermore, there is an efficient algorithm that generates this list. Thus, the $7/8$-approximation algorithm, with input ϕ – a formula with n variables, constructs this list, and chooses the first assignment in the list that satisfies the most clauses in ϕ.

We next explain how to construct the list, using ideas of the randomized $7/8$-approximation algorithm of Johnson [15]. Fix a clause with three different literals. If we pick an assignment at random, then with probability at least $7/8$ it satisfies the clause. Now, fix any exact 3CNF formula. If we pick an assignment at random, then the expected fraction of satisfied clauses is at least $7/8$. Thus, there exists at least one assignment that satisfies a fraction of at least $7/8$ of the clauses in the formula. This is true even if we pick the assignments from a 3-wise independent space. As there is a 3-wise independent space of size $O(n^3)$, this implies the existence of the list. To generate the assignments we can use any of the constructions of 3-wise independent spaces, e.g., the construction based on polynomials (see, e.g., [17, 1, 5]).

We next describe the $(7/8 - \epsilon)$-approximation algorithm. As in the previous case, it suffices to show how to efficiently construct, for every n and $\epsilon > 0$, a list of $\text{poly}(\frac{\log n}{\epsilon})$ assignments such that for every exact 3CNF formula with n variables there is an assignment in the list that satisfies at least $7/8 - \epsilon$ of the clauses of the formula. To construct the list, notice that if we pick an assignment from an ϵ-biased 3-wise independent space, then the probability that a given clause is satisfied is at least $7/8 - \epsilon$. Thus, the expected fraction of satisfied clauses is at least $7/8 - \epsilon$, and there exists at least one assignment that satisfies a fraction of at least $7/8 - \epsilon$ of the clauses in the formula. There are ϵ-biased 3-wise independent spaces of size $\text{poly}(\frac{\log n}{\epsilon})$. To generate the assignments we can use any of the constructions of [18, 2]. $\square$

Claim 4.6 *Every solution-list algorithm for* maxE3SAT *that achieves approximation ratio better than $1/2$ uses at least $\log n - 1$ solutions.*

Proof: Assume a solution-list algorithm for maxE3SAT, and let $a_1, \ldots, a_t$, where $t < \log n - 1$, be its list of possible output assignments on formulae over n variables $x_1, \ldots, x_n$. To each variable x_i we assign a *label* that is the concatenation of the truth values assigned to x_i by the t assignments $\langle a_1(x_i), \ldots, a_t(x_i) \rangle$. As there are at most 2^t different labels and $n/2^t > 2$, there exist three distinct variables $x_{i_1}, x_{i_2}, x_{i_3}$ that share the same label. I.e. for all $1 \le j \le t$ it holds that $a_j(x_{i_1}) = a_j(x_{i_2}) = a_j(x_{i_3})$.

Consider the formula $\phi = (x_{i_1} \vee x_{i_2} \vee x_{i_3}) \wedge (\neg x_{i_1} \vee \neg x_{i_2} \vee \neg x_{i_3})$. It is easy to see ϕ is satisfied by exactly those assignment which do not assign the same truth value to all three variables $x_{i_1}, x_{i_2}, x_{i_3}$. However, as this is not the case for any of the t assignments, each of them satisfies exactly one clause in ϕ, achieving approximation factor at most $1/2$. $\square$

4.3 Solution-List Algorithms for Vertex Cover

In this section we present search algorithms for minimum vertex cover. These algorithms are significant, but not as dramatic as the algorithms for Max Exact 3 SAT. For any $0 < \epsilon < 1$, there is an $n^{1-\epsilon}$-approximation algorithm that leaks $O(n^{\epsilon})$ bits. The algorithms are solution-lists algorithms, and we will prove that no solution-list algorithm can do better for this problem.

Claim 4.7 *For every $0 < \epsilon < 1$, there is an $n^{1-\epsilon}$-approximation algorithm for the minimum vertex cover problem which leaks at most $2n^{\epsilon}$ bits.*

Proof: The algorithm proceeds as follows:

```
INPUT: A graph $G$ with $n$ vertices.
    1. Let $\ell \leftarrow 2n^{\epsilon}$.
    2. Partition the $n$ vertices into $\ell$ fixed sets, $V_1, \ldots, V_\ell$, each of size $n/\ell = n^{1-\epsilon}/2$.
    3. Use any 2-approximation algorithm for VC, and get a cover $C$.
    4. Let $C' \leftarrow \bigcup_{\{i : V_i \cap C \neq \emptyset\}} V_i$.
    5. Return $C'$.
```

The algorithm first finds a small cover. Then, if V_i contains at least one vertex in this cover, the algorithm returns the entire set V_i. This implies that the size of C' is at most $|C| n^{1-\epsilon}/2$, and since $|C|$ is at most twice the size of the minimum vertex cover, this algorithm is an $n^{1-\epsilon}$-approximation algorithm.

Notice that the algorithm has 2^{ℓ} possible outputs (it only chooses which of the sets V_i is in its output). That is, this is a solution-list algorithm with a list of size 2^{ℓ}, thus, it leaks at most $\ell = 2n^{\epsilon}$ bits. □

The private algorithms for maxE3SAT and for minVC that we presented are solution-list algorithms. For maxE3SAT, the algorithm generated the entire list, and chooses the best candidate in the list. For minVC this is not possible as the size of the list is big. The algorithm generates a "good" candidate from the list without generating the entire list.

We next claim that any solution-list algorithm for minVC cannot use a shorter list than the algorithm we presented (up-to a constant factor).

Claim 4.8 *Any solution-list algorithm that $n^{1-\epsilon}$-approximates* minVC, *uses at least $2^{n^{\epsilon}/6}$ solutions.*

Proof: Assume that there is a list of covers that $n^{1-\epsilon}$-approximates minVC, that is, for every graph with n vertices and minimum vertex cover of size d, there exists a cover of size at most $n^{1-\epsilon}d$ in the list. We construct a "big" family of graphs, and show that every cover in the list covers "few" graphs in the family, thus the size of the list must be "big."

Consider the following family of graphs. Each graph is defined by a subset I of size $d \overset{\text{def}}{=} n^{\epsilon}/6$. The graph G_I contains all edges between I and $V \setminus I$. Notice that the number of graphs in this family is

$$\binom{n}{d} \geq (n/d)^d. \tag{1}$$

The set I of size d is a cover of the graph G_I. Since we assume that the list $n^{1-\epsilon}$-approximates minVC, there is a cover in the list that covers G_I and contains at most $n^{1-\epsilon}d = n^{1-\epsilon}n^{\epsilon}/6 = n/6$ vertices. However, every vertex cover of G_I contains either all vertices in I (and possibly vertices from $V \setminus I$) or all vertices of $V \setminus I$ (and possibly vertices from I). Since $|V \setminus I| = n - d > n/6$, this cover must contain I. The number of graphs in the family that a given cover of size at most $n/6$ covers is at most

$$\binom{n/6}{d} \leq \left(\frac{en/6}{d}\right)^d. \tag{2}$$

Thus, by (1) and (2), the number of covers in the list is at least

$$\frac{(n/d)^d}{(en/6d)^d} \geq 2^d = 2^{n^\epsilon/6}.$$

$\square$

We emphasize that Claim 4.8 applies only to solution lists algorithms, and does not even imply that there is no polynomial $n^{1-\epsilon}$-approximation algorithms that leaks $o(n^\epsilon)$ bits.[3] The next theorem, however, introduces a general impossibility result, stating that *every* such algorithm must leak $\Omega(\log n)$ bits. The theorem is proves in Section 5.

Theorem 4.9 *Let $\epsilon > 0$ be a constant. If $\mathcal{RP} \neq \mathcal{NP}$, then there is no randomized $n^{1-\epsilon}$-approximation algorithm for the search problem of* minVC *that leaks at most $\frac{\epsilon}{8}\log n$ bits.*

5 Impossibility Result for Vertex Cover Approximation that Leaks $\log n$ Bits

In this section we show it is unlikely that there is an efficient approximation algorithm for minVC that leaks $\log n$ bits of information. Specifically, if there is such an $n^{1-\epsilon}$-approximation algorithm $\mathcal{A}$ that leaks at most $\frac{\epsilon}{8}\log n$ bits, then $\mathcal{RP} = \mathcal{NP}$. In this section we assume that $\mathcal{A}$ is deterministic. Dealing with a randomized $\mathcal{A}$ is done using similar ideas to the proof of Theorem 3.3 in Appendix A.

As in the proof of Theorem 3.2, we assume the existence of such an approximation algorithm and deduce an algorithm that solves vertex cover. Again, we do this by designing an algorithm that solves the Relevant / Non-Critical problem (see Definition 3.7), and thus, by Claim 3.8, solves vertex cover. This algorithm, given the input G and v, and an oracle access to an approximation algorithm $\mathcal{A}$ that leaks k bits, applies $\mathcal{A}$ on a set of inputs, and decides whether v is relevant or non-critical for G according to the results. Note that the `Relevant/Non Critical` algorithm for the (perfectly) private case is not applicable; here, even assuming $\mathcal{A}$ is deterministic, the fact that $\mathcal{A}(G_1) \neq \mathcal{A}(G_2)$ does not directly imply that G_1 and G_2 are not equal under $\mathcal{R}_{\text{minVC}}$. However, as $\mathcal{A}$ leaks at most k bits, if there are $2^k + 1$ graphs with different outputs, then at least two of them are not equivalent.

Our inputs for the `Relevant/Non Critical` algorithm are generally constructed from a number of copies of the original input graph G and big set of isolated vertices I. In each such graph we connect the different copies of the input vertex v with vertices from I, and sometimes connect two different vertices from I. We will use the following notation to address these graphs.

Definition 5.1 *Let $G = \langle V, E\rangle$ be a graph, I be a set of vertices, t, m be indices such that $0 \leq t \leq m$, and $i_1, \ldots, i_m, j_1, \ldots, j_{2t} \in I$ be distinct vertices. The graph*

$$G_{\substack{j_1 j_2\\ \vee\\ i_1} \cdots \substack{j_{2t-1} j_{2t}\\ \vee\\ i_t}\ \substack{i_{t+1}\\ \wedge} \cdots \substack{i_m\\ \wedge}}$$

is defined as follows: the vertices of the graph are $(V \times \{1, \ldots, 2m\}) \cup I$, and the edges of the graph are $E = E_m \cup E_\vee \cup E_\wedge$, where

$$E_m = \{(\langle u, \ell\rangle, \langle w, \ell\rangle) : (u, w) \in E, \ell \in \{1, \ldots, m\}\},$$

$$E_\vee = \{(i_\ell, j_{2\ell-1}), (i_\ell, j_{2\ell}) : \ell \in \{1, \ldots, t\}\}, \textit{and}$$

$$E_\wedge = \{(\langle v, 2\ell-1\rangle, i_\ell), (\langle v, 2\ell\rangle, i_\ell) : \ell \in \{t+1, \ldots, m\}\}.$$

Informally, the graph has $2m$ copies of G (see Figure 5). It has t "vees," where the ℓth vee is associated with copies $2\ell - 1$ and 2ℓ of G. It has $m - t$ "wedges," where the ℓth wedge connects i_ℓ to the copies of v in copies $2\ell - 1$ and 2ℓ of G, for $t < \ell \leq m$.

[3]It can be proved that the algorithm we presented leaks $\Omega(n^\epsilon)$ bits.

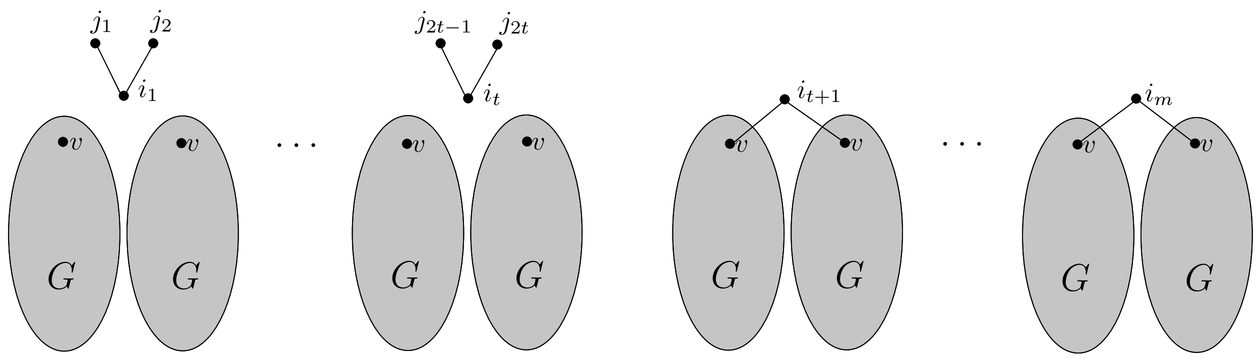

Figure 5: The graph $G_{\underset{i_1}{\overset{j_1j_2}{\vee}} \cdots \underset{i_t}{\overset{j_{2t-1}j_{2t}}{\vee}} \overset{i_{t+1}}{\wedge} \cdots \overset{i_m}{\wedge}}$.

We next present two combinatorial lemmas, whose role is similar to the Lemmas 3.11 and 3.12 in the case where $\mathcal{A}$ was perfectly private.

Claim 5.2 *Let $1 \le t \le m$, and $i_1, \ldots, i_m, j_1, \ldots, j_{2t}, i'_{t+1}, \ldots, i'_m \in I$ be distinct vertices. Furthermore, let*

$$H \overset{\text{def}}{=} G_{\underset{i_1}{\overset{j_1j_2}{\vee}} \cdots \underset{i_t}{\overset{j_{2t-1}j_{2t}}{\vee}} \overset{i_{t+1}}{\wedge} \cdots \overset{i_m}{\wedge}} \quad \text{and} \quad H' \overset{\text{def}}{=} G_{\underset{i_1}{\overset{j_1j_2}{\vee}} \cdots \underset{i_t}{\overset{j_{2t-1}j_{2t}}{\vee}} \overset{i'_{t+1}}{\wedge} \cdots \overset{i'_m}{\wedge}}.$$

If v is critical for G, then $H \equiv_{\mathcal{R}_{\text{minVC}}} H'$.

Proof: By Claim 3.11, the graphs H and H' are unions of graphs the are equivalent, thus, H and H' are equivalent. $\square$

Claim 5.3 *Let $1 \le t' < t \le m$, and $i_1, \ldots, i_m, j_1, \ldots, j_{2t} \in I$ be distinct vertices. Furthermore, let*

$$H \overset{\text{def}}{=} G_{\underset{i_1}{\overset{j_1j_2}{\vee}} \cdots \underset{i_{t'}}{\overset{j_{2t'-1}j_{2t'}}{\vee}} \cdots \underset{i_t}{\overset{j_{2t-1}j_{2t}}{\vee}} \overset{i_{t+1}}{\wedge} \cdots \overset{i_m}{\wedge}} \quad \text{and} \quad H' \overset{\text{def}}{=} G_{\underset{i_1}{\overset{j_1j_2}{\vee}} \cdots \underset{i_{t'}}{\overset{j_{2t'-1}j_{2t'}}{\vee}} \overset{i_{t'+1}}{\wedge} \cdots \overset{i_m}{\wedge}}.$$

If v is non-relevant for G, then $H \equiv_{\mathcal{R}_{\text{minVC}}} H'$.

Proof: Note that the vertices $i_1, \ldots, i_m$ are critical for both H and H': It is straight forward that i_ℓ is critical for $\underset{i_\ell}{\overset{j_{2\ell-1}j_{2\ell}}{\vee}}$. By Claim 3.12, vertex i_ℓ is critical for $\overset{i_\ell}{\wedge}$. Therefore, the two graphs have the same set of minimum vertex covers, hence they are equivalent under $\mathcal{R}_{\text{minVC}}$. $\square$

In both claims we have the same graph H. In the Claim 5.2, the graph H' has the same "vees" as H, however, the "wedges" have different vertices from I (namely, $i'_1, \ldots, i'_{t-1}$). In Claim 5.3, the sequence of i's is the same in H and H', however, there are "vees" in H' in some places where there are "wedges" in H.

5.1 Handling One Bit Leakage

To simplify our presentation, we first present `Algorithm Relevant/Non Critical – one bit leakage` in Figure 5.4 . This algorithm assumes that the deterministic algorithm $\mathcal{A}$ leaks at most one bit. This case is simpler and describes some of the ideas used for the $\log n$ bits leakage case.

```
Algorithm Relevant/Non Critical – one bit leakage

INPUT: A graph $G = \langle V, E \rangle$ and a vertex $v \in V$.
OUTPUT: One of the following: (i) The vertex $v$ is relevant for $G$. (ii) The vertex $v$ is not critical for $G$.

    1. Execute $\mathcal{A}$ on all the graphs of the form $G_{\underset{\wedge}{i_1}\ \underset{\wedge}{i_2}}$, where $i_1, i_2 \in I$.
        (a) If these executions result in more than two different answers, return "Non Critical,"
        (b) Else, remove from $I$ all the vertices that appeared in any of the above results.
    2. Pick $i_1, j_1, j_2 \in I$ and execute $\mathcal{A}$ on all the graphs of the form $G_{\underset{\underset{i_1}{\vee}}{j_1 j_2}\ \underset{\wedge}{i_2}}$, where $i_2 \in I \setminus \{i_1, j_1, j_2\}$.
        (a) If these executions result in more than two different answers, return "Non Critical."
        (b) Else, return "Relevant."
```

Figure 6: `Algorithm Relevant/Non Critical – one bit leakage`.

Claim 5.4 `Algorithm Relevant/Non Critical – one bit leakage` *is correct.*

Proof: By Claim 5.2, if v is critical for G, then all graphs considered in step (1) of the algorithm are equivalent. As $\mathcal{A}$ leaks at most 1 bit, there can be at most 2 different answers on equivalent graphs. Hence, if there are more than two different answers in step (1), vertex v is non critical for G. Similarly, if there are more than two different answers in step (2), vertex v is non critical for G. Thus, if the algorithm outputs "Non critical," vertex v is non critical for G, and the algorithm is correct.

Else, fix any $i_2, j_3, j_4 \in I$, and note that:

1. $\mathcal{A}(G_{\underset{\wedge}{i_1}\ \underset{\wedge}{i_2}})$ *does not contain* any of $\{i_1, j_1, j_2\}$, and *does not contain* any of $\{i_2, j_3, j_4\}$.
2. $\mathcal{A}(G_{\underset{\underset{i_1}{\vee}}{j_1 j_2}\ \underset{\wedge}{i_2}})$ *contains* at least one of $\{i_1, j_1, j_2\}$, and *does not contain* any of $\{i_2, j_3, j_4\}$.
3. $\mathcal{A}(G_{\underset{\underset{i_1}{\vee}}{j_1 j_2}\ \underset{\underset{i_2}{\vee}}{j_3 j_4}})$ *contains* at least one of $\{i_2, j_3, j_4\}$.

Thus, these 3 results of $\mathcal{A}$ are all different. However, by Claim 5.3, if vertex v is not relevant for G, then the three graphs are equivalent. Since $\mathcal{A}$ leaks at most one bit, there cannot be three different answers on three equivalent graphs. Thus, if the algorithm outputs "Relevant," vertex v is relevant for G, and the algorithm is correct. □

Similarly to the case where $\mathcal{A}$ is perfectly private, it is enough to set $|I| = O((4n)^{1/\epsilon} - 2n)$ to ensure there are enough vertices in I that are not returned by $\mathcal{A}$ (see Claim 3.16). As I is polynomial in n, the number of calls to $\mathcal{A}$ in the algorithm is polynomial.

Theorem 5.5 *Let $\epsilon > 0$. If $\mathcal{P} \neq \mathcal{NP}$, then there is no deterministic $n^{1-\epsilon}$-approximation algorithm for* vertex cover *that leaks at most* 1

5.2 Handling $\log n$-Bits Leakage

We next want to accommodate a deterministic algorithm $\mathcal{A}$ that leaks $\log n$ bits. Using a similar algorithm as in the one bit leakage case with graphs that contains more copies of G, we can handle Algorithms $\mathcal{A}$ that leaks more bits. However, if $\mathcal{A}$ leaks $\omega(1)$ bits, the number of executions of $\mathcal{A}$ will be too big. To reduce the number of calls to $\mathcal{A}$, we choose the vertices from I at random.

In Figure 7, we present `Algorithm Relevant/Non Critical` $-\log n$ `bits leakage`, which assumes that Algorithm $\mathcal{A}$ is a deterministic $n^{1-\epsilon}$-approximation algorithm that leaks at most $\frac{\epsilon}{8}\log n$ bits. `Algorithm Relevant/Non Critical` $-\log n$ `bits leakage` is a randomized algorithm which returns a correct answer with probability at least $1-\delta$, for some $0 < \delta < 1$. Given a graph G with n vertices, we construct the graphs from Definition 5.1. These graphs have $N = (18n/\delta)^{2/\epsilon}$ vertices, $2m$ copies of G, and a disjoint set of vertices I. We choose the number of copies to be $2m$, where

$$m \overset{\text{def}}{=} N^{\epsilon/8} = (18n/\delta)^{1/4}. \tag{3}$$

Therefore, the size of the set I is $|I| = N - 2mn$. In the proof of Claim 5.7 it would become clear why we made these choices.

```
Algorithm Relevant/Non Critical – $\log n$ bits leakage

INPUT: A graph $G = \langle V, E\rangle$, a vertex $v \in V$, and a number $0 < \delta < 1$.
OUTPUT: One of the following: (i) The vertex $v$ is relevant for $G$. (ii) The vertex $v$ is not critical for $G$.
The algorithms errs with probability at most $\delta$.

  1. Choose distinct $i_1, \ldots, i_m, j_1, \ldots, j_{2m}$ at random from $I$.
  2. For $t = 0$ to $m$ do:
      (a) $W_t = \mathcal{A}(G_{\substack{j_1 j_2\\ \vee\\ i_1} \cdots \substack{j_{2t-1} j_{2t}\\ \vee\\ i_t} \substack{i_{t+1}\\ \wedge} \cdots \substack{i_m\\ \wedge}})$.
      (b) If $W_t \cap \{i_{t+1}, \ldots, i_m, j_{2t+1}, \ldots, j_{2m}\} \neq \emptyset$ then return “Non Critical.”
  3. (* all $m+1$ sets $W_t$ do not contain the last $m-t$ vertices *)
     RETURN “Relevant.”
```

Figure 7: `Algorithm Relevant/Non Critical` $-\log n$ `bits leakage`.

In the following we assume that, on graphs with n vertices, $\mathcal{A}$ leaks at most $k(n) = \frac{\epsilon}{8}\log n$ bits. Recall that we execute $\mathcal{A}$ on graphs with N vertices, thus, the approximation ratio is $N^{1-\epsilon}$ and the leakage is bounded by $k(N) = \frac{\epsilon}{8}\log N = \log m$. The next sequence of claims asserts the correctness of `Algorithm Relevant/Non Critical` $-\log n$ `bits leakage`.

Claim 5.6 *If* `Algorithm Relevant/Non Critical` $-\log n$ `bits leakage` *returns “Relevant,” then v is relevant.*

Proof: If the algorithms returns “Relevant,” then the for loop finishes. In this case the sets $W_0, \ldots, W_m$ are $m+1 = 2^k + 1$ different sets; for $t' < t$ the set W_t must contain at least one of the vertices in $\{i_j, j_{2t-1}, j_{2t}\}$ while $W_{t'}$ contains none of them. Thus, there are 2^k+1 graphs with pair-wise different answers of $\mathcal{A}$, and, since $\mathcal{A}$ leaks at most k bits, at least two of the graphs are not equivalent according to $\mathcal{R}_{\text{minVC}}$. Thus, by Claim 5.3, the vertex v is relevant for G. □

Claim 5.7 *Let $0 \le t \le m$. For every $i_1, \ldots, i_t, j_1, \ldots, j_{2t} \in I$, consider the following experiment: choose $i_{t+1}, \ldots, i_m$ at random and with uniform distribution from I and return 1 if*

$$\mathcal{A}(G_{\substack{j_1 j_2\\ \vee\\ i_1} \cdots \substack{j_{2t-1} j_{2t}\\ \vee\\ i_t} \substack{i_{t+1}\\ \wedge} \cdots \substack{i_m\\ \wedge}}) \cap \{i_{t+1}, \ldots, i_m, j_{2t+1}, \ldots, j_{2m}\} \neq \emptyset.$$

If v is critical and $N = (18n/\delta)^{2/\epsilon}$, then the probability that this experiment returns 1 is at most δ/m.

Proof: We choose N – the number of vertices in the graphs we construct – such that the size of each cover $\mathcal{A}$ returns is at most $|I|/\alpha$, for some α to be fixed later in this proof. Thus, with probability at most $1/\alpha$, a random vertex from I is in a given answer.

We now analyze the probability that the experiment returns 1. If v is critical, then, by Claim 5.2, every two choices of $i_{t+1}, \ldots, i_m$ result in an equivalent graphs according to $\mathcal{R}_{\text{minVC}}$. Since $\mathcal{A}$ leaks at most $k(N) = \frac{\epsilon}{8} \log N = \log m$ bits, there are at most $2^{k(N)} = m$ different answers for all different choices. Thus, the size of the union of all the answers is at most $\frac{m|I|}{\alpha}$. The probability that any of the $3(m-t) \leq 3m$ vertices are in the union of the m answers is, by the union bound, at most $\frac{3m^2}{\alpha}$. We, thus, choose the number of vertices N, such that

$$\alpha = 3m^3/\delta, \tag{4}$$

and the probability of the experiment returning 1 is at most $\frac{3m^2}{\alpha} = \delta/m$, as required.

Finally, we show how to choose N. The first requirement we need is

$$|I| = N - 2mn \geq N/2. \tag{5}$$

That is, we need $N/2 \geq 2mn$. As $m = N^{\epsilon/8} \leq N^{1/8}$, it suffices to require $N \geq (4n)^{8/7}$. By the choice of N

$$N = (18n/\delta)^{2/\epsilon} \geq (18n)^2 \geq (4n)^{8/7}$$

(since $0 < \delta, \epsilon \leq 1$), thus (5) holds.

We next upper-bound the size of the covers that $\mathcal{A}$ outputs. The size of a minimum vertex cover of these graphs is at most $2 \cdot (n+1)m \leq 3nm$. Since $\mathcal{A}$ is a $N^{1-\epsilon}$-approximation algorithm for vertex cover, the size of its output is at most $3 \cdot nm \cdot N^{1-\epsilon}$. We choose $N = (\frac{18n}{\delta})^{2/\epsilon}$ and $m = N^{\epsilon/8}$, thus, by (3),

$$18nm^4 = \delta(18n/\delta) \cdot N^{\epsilon/2} = \delta((18n/\delta)^{2/\epsilon})^{\epsilon/2} \cdot N^{\epsilon/2} = \delta N^{\epsilon/2} N^{\epsilon/2} = \delta N^{\epsilon}. \tag{6}$$

Therefore, the size of the cover returned by $\mathcal{A}$ is at most

$$3 \cdot nm \cdot N^{1-\epsilon} = \frac{3nmN}{N^\epsilon} \leq \frac{6nm|I|}{N^\epsilon} = \frac{18nm^4}{\delta N^\epsilon} \cdot \frac{\delta|I|}{3m^3} = \frac{\delta|I|}{3m^3},$$

where the inequality above follows from (5) and the last equality follows (6). To conclude, taking $N = (18n/\delta)^{2/\epsilon}$ guarantees that the size of the cover is at most $|I|/\alpha$, for the α we needed in (4), which, in turn, implies that the probability of the experiment returning 1 is at most δ/m. □

Claim 5.8 *If* `Algorithm Relevant/Non Critical`$-\log n$ `bits leakage` *returns "Non Critical," then, with probability at least $1-\delta$, vertex v is non-critical.*

Proof: Since `Algorithm Relevant/Non Critical`$-\log n$ `bits leakage` repeats $2^k + 1$ the experiment of Claim 5.7, and for $t = m$ the experiment cannot fail, the probability that `Algorithm Relevant/Non Critical`$-\log n$ `bits leakage` errs when returning "Non Critical" is at most δ. □

`Algorithm Relevant/Non Critical`$-\log n$ `bits leakage` executes m times the polynomial algorithm $\mathcal{A}$ on graphs with N vertices, where $N = (18n/\delta)^{2/\epsilon}$. Thus,

Claim 5.9 *If $\mathcal{A}$ runs in polynomial time and $0 \leq \epsilon < 1$ is a constant, then the running time of* `Algorithm Relevant/Non Critical`$-\log n$ `bits leakage` *is* $\text{poly}(n/\delta)$.

Theorem 5.10 *Let $\epsilon > 0$ be a constant. If $\mathcal{RP} \neq \mathcal{NP}$, then there is no deterministic $n^{1-\epsilon}$-approximation algorithm for the search problem of* vertex cover *that leaks at most $\frac{\epsilon}{8}\log n$ bits.*

Proof: `Algorithm Greedy Vertex Cover`, which solves vertex cover, executes at most n times `Algorithm Relevant/Non Critical`$-\log n$ `bits leakage` with graphs of size at most n. We execute these calls with $\delta = \frac{1}{4n}$ (where n is the original number of vertices in G), thus, all together the error is at most $1/4$. By Claim 5.9, the running time of `Algorithm Greedy Vertex Cover` is polynomial.

Thus, if there is an $n^{1-\epsilon}$-approximation algorithm for the search problem of vertex cover that leaks at most $k(n) = \frac{\epsilon}{8}\log n$ bits, then there is a polynomial time randomized algorithm for minimum vertex cover that errs with probability $1/4$. This implies that $\mathcal{NP} \subseteq \mathcal{BPP}$. To contradict $\mathcal{RP} \neq \mathcal{NP}$ we construct a one-sided error algorithm for the decision problem of vertex cover as in the proof of Theorem 3.3. $\square$

6 Impossibility Result for Private Approximation of a Search Problem in $\mathcal{P}$

An algorithm solving a search problem can return any solution to the problem. An algorithm solving a search problem *privately* has additional requirements on the solutions that it returns. In this section, we show that these additional requirements can make the problem much harder. That is, we show that there is a polynomial relation Q whose search problem is in $\mathcal{P}$, however, unless $\mathcal{NP} \subseteq \mathcal{P}/\text{poly}$, there is no polynomial time private algorithm for Q with respect to $\mathcal{R}_Q$.

Definition 6.1 *Let G be a graph and $C_1, C_2 \subseteq V$. We define the relation Q as follows: $\langle G, C_1\rangle$ and C_2 are in Q (that is, $\langle\langle G, C_1\rangle, C_2\rangle \in Q$) if $|C_1| = |C_2|$ and C_1 and C_2 are cliques in G.*

Clearly, the search problem of Q is easy, given $\langle G, C_1\rangle$ return C_1. Assume that there is a private algorithm for $\mathcal{R}_Q$. That is, if C_1 and C_2 are two disjoint cliques of the same size in a graph G, then a private algorithm has to return the same output distribution on $\langle G, C_1\rangle$ and $\langle G, C_2\rangle$. Intuitively, this implies that given a clique in the graph, there is an effi cient algorithm that fi nds another clique. We will prove that this is impossible unless $\mathcal{NP} \subseteq \mathcal{P}/\text{poly}$.

Theorem 6.2 *If $\mathcal{NP} \not\subseteq \mathcal{P}/\text{poly}$, then there is no polynomial-time private algorithm for the search problem of Q with respect to the privacy structure $\mathcal{R}_Q$.*

Proof: We assume towards contradiction that there exists a polynomial-time private algorithm $\mathcal{A}$ for the search problem of Q with respect to the privacy structure $\mathcal{R}_Q$. We will use $\mathcal{A}$ to prove that the $\mathcal{NP}$-complete problem CLIQUE is in $\mathcal{P}/\text{poly}$. That is, we construct a sequence of polynomial-length advice strings $\langle a_n\rangle_{n\in\mathbb{N}}$ and a polynomial time algorithm $\mathcal{B}$ such that, given a graph G with n vertices, an integer k, and the advice a_n, Algorithm $\mathcal{B}$ decides if G contains a clique of size k.

Given two graph $G_0 = \langle V_0, E_0\rangle$ and $G_1 = \langle V_1, E_1\rangle$, where $V_1 \cap V_2 = \emptyset$, we defi ne their disjoint union $G_0 \cup G_1$ as the graph $G = \langle V, E\rangle$ where $V = V_0 \cup V_1$ and $E = E_0 \cup E_1$. Every clique in $G_0 \cup G_1$ is either a clique in G_0 or a clique in G_1. Assume that C_0 and C_1 are cliques of size k in G_0 and G_1 respectively. Then, $\langle G_0 \cup G_1, C_0\rangle$ and $\langle G_0 \cup G_1, C_1\rangle$ have the same set of cliques of size k, that is, they are in $\mathcal{R}_Q$.

As a fi rst step, we construct in Figure 8 an algorithm $\mathcal{A}'$ that will be used to construct Algorithm $\mathcal{B}$ and the advice strings. This algorithm gets two graphs G_0, G_1 and a clique C in one of them, executes $\mathcal{A}$ on their union, and checks in which graph $\mathcal{A}$ returns a clique. If only one graph G_i has a clique of size $|C|$, then, by the correctness requirement of $\mathcal{A}$, Algorithm $\mathcal{A}'$ must return i. However, if both graphs have a clique of size $|C|$, then, by the privacy requirement of $\mathcal{A}$, the output distribution of $\mathcal{A}'$ is approximately the same when C is a clique in G_0 and when C is a clique in G_1. That is, there is an integer n_0 such that for every pair of graphs with at least n_0 vertices, the difference between the probabilities that $\mathcal{A}'$ returns 0 in the two cases is small (for concreteness, at most $1/3$).

```
Algorithm $\mathcal{A}'$

INPUT: Two graphs $G_0, G_1$ with disjoint sets of vertices and a set $C$.
PROMISE: $C$ is a clique in $G_0 \cup G_1$.

    1. Execute $\mathcal{A}$ on $\langle G_0 \cup G_1, C\rangle$
        (a) Let $j$ be the index such that $\mathcal{A}$ returns a clique in $G_j$.
        (b) Return $j$.
```

Figure 8: Algorithm $\mathcal{A}'$ which gets two graphs and a clique, executes $\mathcal{A}$ on their union, and checks in which graph $\mathcal{A}$ returns a clique.

We construct the advice string a_n as the concatenation of advice strings $a_{n,1}, \dots, a_{n,n}$ where $a_{n,k}$ is used to decide if a graph G with n vertices has a clique of size k.[4] Fix an integer n and an integer k, where $n \geq n_0$ and $1 \leq k \leq n$. For every graph G with n vertices that has a clique of size k, we fi x some clique C_G of size k in G. Let G_0 and G_1 be two graphs that have a clique of size k.

Definition 6.3 *We say that G_0 loses to G_1 if Algorithm $\mathcal{A}'$ returns 1 on input $\langle G_0, G_1, C_{G_0}\rangle$ with probability at least $1/3$.*

That is, G_0 loses to G_1 if, when given a clique in G_0, Algorithm $\mathcal{A}$ "magically" manages to return with probability $1/3$ a clique in the graph G_1; thus, G_0, C_{G_0} can aid in fi nding a clique in G_1.

If G_0 does not lose to G_1, then $\mathcal{A}'$ returns 0 with probability at least $2/3$, and by the privacy requirement, with probability at least $1/3$ Algorithm $\mathcal{A}'$ returns 0 on input $\langle G_0, G_1, C_{G_1}\rangle$. That is,

Claim 6.4 *Let G_0 and G_1 be two graphs with n vertices that have a clique of size k. If G_0 does not lose to G_1, then G_1 loses to G_0*

(It is possible that G_0 loses to G_1 and G_1 loses to G_0.) Thus, for every set of graphs with n vertices and a clique of size k, there exists a graph G_0 in the set that loses to at least half the graphs in the set. This is the idea of constructing the advice string $a_{n,k}$.

```
Construction of $a_{n,k}$

    1. $a_{n,k} \leftarrow \emptyset$.
    2. Initialize $L$ as the set of all graphs with $n$ vertices that have a clique of size $k$.
    3. While $L \neq \emptyset$ do
        (a) Choose a graph $G$ in $L$ that loses to at least half of the graphs in $L$.
        (b) $a_{n,k} \leftarrow a_{n,k} \cup \{\langle G, C_G\rangle\}$.
        (c) $L \leftarrow L \setminus \{G_1 : G \text{ loses to } G_1\}$.
```

Since there are $2^{O(n^2)}$ graphs with n vertices, the advice $a_{n,k}$ contains $O(n^2)$ graphs. We are ready to describe the non-uniform algorithm $\mathcal{B}$ that, given a graph G with n vertices, an integer k, and the advice a_n, decides if G contains a clique of size k.

[4]We assume that any two graphs with n vertices have disjoint sets of vertices. E.g., we can order all graphs with n vertices according to some order, and the vertices of the ith graph in this order are $\{1, \dots, n\} \times \{i\}$.

```
Algorithm $\mathcal{B}$

INPUT: $G$, $k$, and $a_{n,k}$.

  1. For every $G_0$ in $a_{n,k}$ execute $\mathcal{A}'$ on $\langle G_0, G, C_{G_0} \rangle$.
  2. If there exists an execution of $\mathcal{A}'$ which returns $1$, return "$G$ has a clique of size $k$,"
  3. Otherwise, return "FAIL."
```

If G does not have a clique of size k, then, by the correctness of $\mathcal{A}$, Algorithm $\mathcal{A}'$ always returns 0, and $\mathcal{B}$ always returns "FAIL." If G has a clique of size k, then there exists a graph G_0 in $a_{n,k}$ that loses to G, thus, with probability at least $1/3$, Algorithm $\mathcal{A}'$ returns 1 on input $\langle G_0, G, C_{G_0} \rangle$, and, thus, with probability at least $1/3$, Algorithm $\mathcal{B}$ returns "G has a clique of size k." That is, $\mathcal{B}$ is a randomized algorithm with advice strings that decides if $\langle G, k \rangle \in$ CLIQUE with a one-sided error of $2/3$. By a standard amplifi cation and union-bound arguments, we can get a deterministic polynomial-time non-uniform algorithm that decides if $\langle G, k \rangle \in$ CLIQUE without error. □

Acknowledgement. We thank Yinnon Haviv for helpful discussions and Robi Krauthgamer for pointing out that we can use ϵ-biased independent spaces.

References

[1] N. Alon, L. Babai, and A. Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *J. Algorithms*, 7(4):567 – 583, 1986.

[2] N. Alon, O. Goldreich, J. Håstad, and R. Peralta. Simple constructions of almost k-wise independent random variables. *Random Structures $\&$ Algorithms*, 3:289–304, 1992.

[3] R. Bar-Yehuda, B. Chor, E. Kushilevitz, and A. Orlitsky. Privacy, additional information, and communication. *IEEE Trans. on Information Theory*, 39(6):1930–1943, 1993.

[4] M. Bellare and E. Petrank. Making zero-knowledge provers effi cient. In *Proc. of the 24th ACM Symp. on the Theory of Computing*, pages 711–722, 1992.

[5] B. Chor, J. Friedmann, O. Goldreich, J. Hastad, S. Rudich, and R. Smolansky. The bit extraction problem or t-resilient functions. In *Proc. of the 26th IEEE Symp. on Foundations of Computer Science*, pages 396–407, 1985.

[6] J. Feigenbaum, Y. Ishai, T. Malkin, K. Nissim, M. J. Strauss, and R. N. Wright. Secure multiparty computation of approximations. In P. G. Spirakis and J. van Leeuven, editors, *Proc. of the 28th International Colloquium on Automata, Languages and Programming*, volume 2076 of *LNCS*, pages 927–938. Springer-Verlag, 2001.

[7] M. J. Freedman, K. Nissim, and B. Pinkas. Effi cient private matching and set intersection. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 1–19. Springer-Verlag, 2004.

[8] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proc. of the 19th ACM Symp. on the Theory of Computing*, pages 218–229, 1987.

[9] O. Goldreich, R. Ostrovsky, and E. Petrank. Computational complexity and knowledge complexity. *SIAM J. on Computing*, 27(4):1116–1141, 1998.

[10] O. Goldreich and E. Petrank. Quantifying knowledge complexity. *Computational Complexity*, 8(1):50–98, 1999. Preliminary version appeared in FOCS 91.

[11] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. on Computing*, 18(1):186–208, 1989.

[12] S. Halevi, R. Krauthgamer, E. Kushilevitz, and K. Nissim. Private approximation of NP-hard functions. In *Proc. of the 33th ACM Symp. on the Theory of Computing*, pages 550–559, 2001.

[13] J. Håstad. Some optimal inapproximability results. *J. of the ACM*, 48(4):798–859, 2001.

[14] P. Indyk and D. Woodruff. Polylogarithmic private approximations and efficient matching. Technical Report TR05-117, Electronic Colloquium on Computational Complexity, http://www.eccc.uni-trier.de/eccc/, 2005. To appear at TCC 2006.

[15] D. S. Johnson. Approximation algorithms for combinatorial problems. *J. of Computer and System Sciences*, 9:256–278, 1974.

[16] E. Kiltz, G. Leander, and J. Malone-Lee. Secure computation of the mean and related statistics. In J. Kilian, editor, *the Second Theory of Cryptography Conference – TCC 2005*, volume 3378 of *LNCS*, pages 283–302. Springer-Verlag, 2005.

[17] M. Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM J. on Computing*, 15(4):1036–1055, 1986.

[18] J. Naor and M. Naor. Small-bias probability spaces: Efficient constructions and applications. *SIAM J. on Computing*, 22(4):838–856, 1993.

[19] C. H. Papadimitriou and M. Yannakakis. On limited nondeterminism and the complexity of the V-C dimension. *J. of Computer and System Sciences*, 53(2):161–170, 1996.

[20] E. Petrank and G. Tardos. On the knowledge complexity of NP. *Combinatorica*, 22(1):83–121, 2002.

[21] A. C. Yao. Protocols for secure computations. In *Proc. of the 23th IEEE Symp. on Foundations of Computer Science*, pages 160–164, 1982.

A Impossibility Result for Randomized Private Approximation of Vertex Cover

In this section, we generalize the inapproximability results of Section 3.1 to *randomized* private protocols. here, a single execution of the approximation algorithm cannot yield information about v. However, from k executions of $\mathcal{A}$, we can decide if v is relevant or non-critical for G with probability $1 - \exp(-k)$. We start with a claim that is analogue to Claim 3.13 from the deterministic case.

Claim A.1 *Let $\mathcal{A}$ be a randomized private approximation algorithm for* vertex cover. *For every polynomial $p(\cdot)$, there exists some $n_0 \in \mathbb{N}$ such that for every $n \geq n_0$ and every graph $G = \langle V, E \rangle$, where $|V| = n$, if $\Pr[\{u, v\} \cap \mathcal{A}(G) = \emptyset] \geq 1/p(n)$, for some $u, v \in V$, then u is not critical for G.*

Proof: Let n be an integer, $p(\cdot)$ be a polynomial, $G = \langle V, E\rangle$ be a graph, where $|V| = n$, and $u, v \in V$. Assume that $\Pr[\{u, v\} \cap \mathcal{A}(G) = \emptyset] \geq 1/p(n)$. As u and v are both not in $\mathcal{A}(G)$ with probability at least $1/p(|n|)$, and $\mathcal{A}(G)$ is a cover of G, we infer that $(u, v) \notin E$. Let $E^* = E \cup \{(u, v)\}$, and define $G^* = \langle V, E^*\rangle$. We consider a (hypothetical) execution of the algorithm $\mathcal{A}$ on G^*. In every execution of $\mathcal{A}$, the set $\mathcal{A}(G^*)$ must cover the edge (u, v) and thus $\Pr[\{u, v\} \cap \mathcal{A}(G) = \emptyset] - \Pr[\{u, v\} \cap \mathcal{A}(G^*) \neq \emptyset] \geq 1/p(|n|)$.

Consider the following algorithm $\mathcal{D}$, whose input is two graphs and the output C of $\mathcal{A}$ on one of them. If the graphs differ in more than one edge, Algorithm $\mathcal{D}$ always returns 1. If the graphs differ in exactly one edge, say (w_1, w_2), Algorithm $\mathcal{D}$ returns 1 if $\{w_1, w_2\} \cap C \neq \emptyset$ and 0 otherwise. Now consider the execution of $\mathcal{D}$ where the graphs are G and G^* (which differ in exactly one edge). Since $\mathcal{A}$ is private, there exists some n_0 such that $\mathcal{D}$ cannot distinguish between equivalent graphs with more than n_0 vertices. Thus, if G has $n \geq n_0$ vertices and $\Pr[\{u, v\} \cap \mathcal{A}(G) = \emptyset] \geq 1/p(n)$, the graphs G and G^* are not equivalent. This implies, by Claim 3.9, that u is not critical for G. □

We present Algorithm `Randomized Relevant-Non-Critical Vertex` in Figure 9. We now prove its correctness. We start with asserting that in step (5) there exists at least one vertex v' the algorithm can choose.

```
Algorithm Randomized Relevant-Non-Critical Vertex

INPUT AND OUTPUT: See Definition 3.7.

  1. If $G$ contains less than $n_2$ vertices (where $n_2$ is a constant that will be determined later), then find if
     $v$ is relevant for $G$ or non-critical for $G$ using exhaustive search.
  2. Let $I$ be a set of size $(8n)^{1/\epsilon} - 2n$.
  3. Construct the graph $G_2$ from $G$ and $I$ as in Definition 3.10.
  4. Execute $k$ times Algorithm $\mathcal{A}$ on $G_2$.
  5. Choose a vertex $v' \in I$ such that $v'$ appeared at most $k/2$ times in $\mathcal{A}(G_2)$ in the $k$ executions.
  6. Construct $G_{\underset{\wedge}{v'}}$ from $G$, $I$, and $v'$ as in Definition 3.10.
  7. Execute $k$ times Algorithm $\mathcal{A}$ on $G_{\underset{\wedge}{v'}}$.
  8. If $v' \in \mathcal{A}(G_{\underset{\wedge}{v'}})$ in at least $0.75k$ of the $k$ executions, then return "NOT CRITICAL." Else return
     "RELEVANT."
```

Figure 9: An algorithm for finding a relevant or non-critical vertex with a blackbox access to a randomized algorithm $\mathcal{A}$.

Claim A.2 *There is a vertex $v' \in I$ such that v' appeared at most $k/2$ times in $\mathcal{A}(G_2)$ out of the k executions.*

Proof: Let $N = |I| + 2n = (8n)^{1/\epsilon}$ be the number of vertices in G_2. The size of the minimum vertex cover of G_2 is twice the size of the minimum vertex cover of G, thus, it is at most $2n$. Since $\mathcal{A}$ is an $N^{1-\epsilon}$-approximation algorithm for vertex cover, the size of $\mathcal{A}(G_2)$ is at most

$$2n \cdot N^{1-\epsilon} \;=\; 2n \cdot ((8n)^{1/\epsilon})^{1-\epsilon} = \frac{(8n)^{1/\epsilon}}{4} \leq \frac{(8n)^{1/\epsilon} - 2n}{2} = \frac{|I|}{2}.$$

Thus, in each execution, at least half of the vertices in I are not in $\mathcal{A}(G_2)$. Consequently, there is at least one vertex $v' \in I$ such that v' is not in $\mathcal{A}(G_2)$ in at least $k/2$ of the executions of $\mathcal{A}$ on G_2. □

In the following claims we use some constants (e.g., 0.55 and 0.6). These constants could be replaced by any constants greater than 0.5 provided the order between them is kept. The first claim is analogue to Claim 3.14 from the deterministic case.

Claim A.3 *There exits a constant n_1 such that if*

1. *G_2 contains at least n_1 vertices,*
2. *$\Pr[v' \in \mathcal{A}(G_2)] < 0.55$, and*
3. *$\Pr[v' \in \mathcal{A}(\underset{\wedge}{G_{v'}})] > 0.6$,*

then v is not critical for G.

Proof: Consider the following algorithm $\mathcal{D}$, whose input is two graphs and the output C of $\mathcal{A}$ on one of them. If the sets of isolated vertices in both graphs are equal, Algorithm $\mathcal{D}$ always returns 1. Otherwise, Algorithm $\mathcal{D}$ chooses the first vertex that is isolated in exactly one of the graphs and returns 1 if this vertex is in C and 0 otherwise. Now consider the execution of $\mathcal{D}$ where the graphs are G_2 and $\underset{\wedge}{G_{v'}}$ (in which v' is the only vertex that is isolated in exactly one of them). Since $\mathcal{A}$ is private, there exists some $n_1 \in \mathbb{N}$ such that $\mathcal{D}$ cannot distinguish between equivalent graphs with more than n_1 vertices. Thus, if G_2 and $\underset{\wedge}{G_{v'}}$ have more than n_1 vertices and $\Pr[v' \in \mathcal{A}(\underset{\wedge}{G_{v'}})] - \Pr[v' \in \mathcal{A}(G_2)] \geq 0.05$, the graphs G_2 and $\underset{\wedge}{G_{v'}}$ are not equivalent. This implies, by Claim 3.11, that $\underset{\wedge}{v}$ is non-critical for G. $\square$

The following claim is the randomized analogue of Claim 3.15.

Claim A.4 *If $\Pr[v' \in \mathcal{A}(\underset{\wedge}{G_{v'}})] \leq 0.8$, then v is relevant for G.*

Proof: Since $|I| = (4n)^{1/\epsilon}$, there must be some $v'' \in I$ such that $\Pr[\{v', v''\} \cap \mathcal{A}(G) = \emptyset]$ is non-negligible. By Claim A.1, the vertex v' is non-critical for $\underset{\wedge}{G_{v'}}$. Hence, by Claim 3.12, the vertex v is relevant for G. $\square$

We are ready to prove the correctness of Algorithm `Randomized Relevant-Non-Critical Vertex`.

Lemma A.5 *Let $k > \Omega((\log n)/\epsilon)$. Algorithm* `Randomized Relevant-Non-Critical Vertex` *returns the correct answer with probability $1 - 2^{-O(k)}$.*

Proof: Let n_0 and n_1 be the constants guaranteed in Claims A.3 and A.4 respectively, and define $n_2 = \max\{n_0, n_1\}$. If G contains less than n_2 vertices, then the correctness is obvious.

Fix any vertex $w \in I$. If $\Pr[w \in \mathcal{A}(G_2)] < 0.55$, then, by Chernoff bound, the probability that $w \in \mathcal{A}(\underset{\wedge}{G_{v'}})$ in less than $0.5k$ of the k executions of Algorithm $\mathcal{A}$ on G_2 is $2^{-O(k)}$. Thus, the probability that there is a vertex $w \in I$ such that $\Pr[w \in \mathcal{A}(G_2)] < 0.55$ and $w \in \mathcal{A}(\underset{\wedge}{G_{v'}})$ in less than $0.5k$ of the k executions of Algorithm $\mathcal{A}$ on G_2 is $|I|2^{-O(k)} = (4n)^{1/\epsilon}2^{-O(k)} = 2^{-O(k)}$ (since $k > \Omega((\log n)/\epsilon)$). Therefore, with probability at least $1 - 2^{-O(k)}$, the vertex v' chosen in `Randomized Relevant-Non-Critical Vertex` satisfies $\Pr[v' \in \mathcal{A}(G_2)] < 0.55$, and in the rest of the proof we assume that $\Pr[v' \in \mathcal{A}(G_2)] < 0.55$.

Let $p = \Pr[v' \in \mathcal{A}(\underset{\wedge}{G_{v'}})]$. If $0.6 < p < 0.8$, then by Claims A.3 and A.4, vertex v is both relevant and non-critical for G, thus the algorithm cannot err in this case. If $p > 0.8$ then, by Claim A.3, vertex v is not critical. By Chernoff bound, the probability that $v' \in \mathcal{A}(G_{\overset{\wedge}{v'}})$ in less than $0.75k$ of the k executions is $2^{-O(k)}$, thus the algorithm errs with probability at most $2^{-O(k)}$ in this case. If $p < 0.6$ then, by Claim A.4, vertex v is not critical. By Chernoff bound, the probability that $v' \in \mathcal{A}(G_{\overset{\wedge}{v'}})$ in more than $0.75k$ of the executions is $2^{-O(k)}$, thus the algorithm errs with probability at most $2^{-O(k)}$ in this case. □

Theorem 3.3 Let $\epsilon > 0$ be a constant. If $\mathcal{RP} \neq \mathcal{NP}$, then there is no randomized private $n^{1-\epsilon}$-approximation algorithm for vertex cover.

Proof: Lemma A.5 and Claim 3.8 imply that if there is a randomized private $n^{1-\epsilon}$-approximation algorithm for vertex cover, then there is a randomized algorithm for the exact search problem of minVC. However this algorithm has a two-sided error.

To contradict $\mathcal{RP} \neq \mathcal{NP}$, this algorithm is transformed to a one-sided error algorithm for the decision problem of vertex cover: Given $\langle G, s\rangle$ decide if G has a vertex cover of size at most s. The transformation is simple; execute the algorithm for the search problem of vertex cover. If this algorithm returns a set that covers G and its size is at most s return "yes." □

B Negative Result for Private Approximation of Max Exact 3 SAT

Recall that the privacy structure $\mathcal{R}_{\text{maxE3SAT}}$ contains all pairs of exact 3 CNF formulae ϕ_1, ϕ_2 over n variables for which an assignment a satisfies the maximum number of clauses in ϕ_1 iff it satisfies the maximum number of clauses in ϕ_2.

Definition B.1 (Private Approximation of Max Exact 3 SAT**)** *An algorithm $\mathcal{A}$ is a* private $c(n)$ approximation *algorithm for* Max Exact 3 SAT *if: (i) $\mathcal{A}$ runs in polynomial time. (ii) $\mathcal{A}$ is a $c(n)$-approximation algorithm for* Max Exact 3 SAT, *that is, for every exact 3 CNF formula ϕ over n variables it returns an assignment that satisfies at least $c(n)$ times the maximum number of clauses that are simultaneously satisfiable in ϕ. (iii) $\mathcal{A}$ is private with respect to privacy structure $\mathcal{R}_{\text{maxE3SAT}}$.*

Theorem B.2 *Let $\epsilon > 0$ be a constant. If $\mathcal{P} \neq \mathcal{NP}$, then there is no deterministic private $1/n^{1-\epsilon}$-approximation algorithm for the search problem of* maxE3SAT.

Proof: Similarly to the proof of Theorem 3.2, we will assume the existence of a deterministic private $1/n^{1-\epsilon}$-approximation algorithm $\mathcal{A}$, and use $\mathcal{A}$ to construct a deterministic algorithm for deciding the satisfiability of exact 3 CNF formulae. We emphasize that we are solving the decision problem of satisfiability and not the optimization problem of maximum satisfiability.

Definition B.3 (Relevant Assignment to a Variable) *Let ϕ be an exact 3 CNF formula over Boolean variables $x_1, \ldots, x_n$. We say that variable x_i is σ-relevant (where $\sigma \in \{0,1\}$) if there exist a maximum assignment a for ϕ such that $a(x_i) = \sigma$.*

Note that for every satisfiable formula, every variable is 0-relevant or 1-relevant (or both). Furthermore, if a variable is not σ-relevant, then, in each assignment that satisfies the formula, its value is $\neg\sigma$, that is, the variable is "$\neg\sigma$-critical."

We will first assume an algorithm `Relevant Variable-Assignment` that given an exact 3 CNF formula ϕ over variables $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$ outputs an index i and a bit σ such that x_i is σ-relevant. The variables $y_1, \ldots, y_n$ are auxiliary variables that we add to the formula to ensure that the formula remains exact 3 CNF. Algorithm `Relevant Variable-Assignment` returns an index of a variable x_i (it never returns a variable y_i). For technical reasons, `Relevant Variable-Assignment` needs that ϕ includes at least 3 variables from $x_1, \ldots, x_n$ (negated or non-negated). Algorithm `Greedy E3SAT`, described in Figure 10, uses algorithm `Relevant Variable-Assignment` to decide E3SAT.

```
Algorithm Greedy E3SAT

INPUT: An exact 3 CNF formula $\phi$ on $n$ variables $x_1, \ldots, x_n$.
OUTPUT: 1 if $\phi$ is satisfiable, 0 otherwise.

  1. Let $\hat{\phi} = \phi$.
  2. While $\hat{\phi}$ contains at least three variables in $x_1, \ldots, x_n$ do:
     (a) Execute Algorithm Relevant Variable-Assignment on $\hat{\phi}$. Denote the answer $\langle i, \sigma \rangle$.
     (b) Assign $a_i \leftarrow \sigma$.
     (c) Modify $\hat{\phi}$ as follows:
         i. Leave in $\hat{\phi}$ every clause that does not include $x_i$ or $\neg x_i$.
         ii. If $\sigma = 0$ then replace every clause $(x_i \vee \ell_j \vee \ell_k)$ in $\hat{\phi}$ by the clauses $(y_i \vee \ell_j \vee \ell_k)$ and
             $(\neg y_i \vee \ell_j \vee \ell_k)$. Remove all clauses that include $\neg x_i$.
         iii. If $\sigma = 1$ then replace every clause $(\neg x_i \vee \ell_j \vee \ell_k)$ in $\hat{\phi}$ by the clauses $(y_i \vee \ell_j \vee \ell_k)$ and
              $(\neg y_i \vee \ell_j \vee \ell_k)$. Remove all clauses that include $x_i$.
  3. For every variable $x_i$ that is not in $\hat{\phi}$ and was not assigned a value in step (2b), assign $a_i \leftarrow 0$.
  4. Exhaustively check all 4 assignments $a$ that agree with the assignments made in step (2b) and
     step (3). If at least one of these assignments satisfies $\phi$ return 1, otherwise return 0.
```

Figure 10: Algorithm `Greedy E3SAT`.

Note that the algorithm terminates in at most $n - 2$ iterations, as every iteration reduces the number of x variables in $\hat{\phi}$ by one. The final length of $\hat{\phi}$ is at most eight times that of ϕ as the replacement steps 2(c)ii and 2(c)iii are applied only to clauses including x variables, resulting in one less x variable in each of the two replacement clauses. Hence, if algorithm `Relevant Variable-Assignment` runs in polynomial time, so does `Greedy E3SAT`.

We now prove the correctness of algorithm `Greedy E3SAT`. Clearly, the only way `Greedy E3SAT` can err is by outputting 0 on a satisfiable formula ϕ, hence we assume from now on that ϕ is satisfiable, and show that executing `Greedy E3SAT` results in outputting 1. Before proving the correctness of the algorithm, we will try to explain what is going on. In each step of the algorithm we have a partial assignment to ϕ that can be extended to an assignment that satisfies ϕ. On one hand, this partial assignment already satisfies some clauses in ϕ and therefore we deleted them from $\hat{\phi}$. On the other hand, some literals in clauses are not satisfied by the partial assignment. We would have liked to delete them from the clauses that they appear in, and continue. However, this might result in a 3 CNF formula that is not an exact 3 CNF formula. We, therefore, replace the literal by an auxiliary variable y_i. By replacing each clause $(x_i \vee \ell_j \vee \ell_k)$ with two clauses, one with y_i and one with $\neg y_i$, we ensure that a satisfying assignment to $\hat{\phi}$ must satisfy $\ell_j \vee \ell_k$.

The formal proof is by induction, on the number of iterations in `Greedy E3SAT`: After executing the main iteration in algorithm `Greedy E3SAT` for k times (i) $\hat{\phi}$ is satisfiable; (ii) k variables are assigned values in Step 2b; (iii) $\hat{\phi}$ does not contain these variables; and (iv) any assignment that extends the k assigned variables satisfies ϕ iff it satisfies $\hat{\phi}$. $\square$

To conclude the proof, we now present algorithm `Relevant Variable-Assignment`. On input ϕ, `Relevant Variable-Assignment` uses a private $1/n^{1-\epsilon}$-approximation algorithm $\mathcal{A}$ to produce an index i and a bit σ such that x_i is σ-relevant.

```
Algorithm Relevant Variable-Assignment

INPUT: An exact 3 CNF formula $\phi$ over Boolean variables $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$. Without loss of
generality, variables $x_1, x_2, x_3$ all appear in $\phi$. Let $m$ denote the number of clauses in $\phi$.
OUTPUT: an index $i \in \{1, 2, 3\}$ and a bit $\sigma$ such that $x_i$ is $\sigma$-relevant.

  1. Execute $\mathcal{A}$ on $\phi$ and denote $a = \mathcal{A}(\phi)$.
  2. For $i \in \{1, 2, 3\}$, let $\ell_i = x_i$ if $a(x_i) = 0$ and $\ell_i = \neg x_i$ otherwise.
  3. Set $\phi' = \phi \wedge (\ell_1 \vee \ell_2 \vee \ell_3) \wedge \cdots \wedge (\ell_1 \vee \ell_2 \vee \ell_3)$, where the clause $(\ell_1 \vee \ell_2 \vee \ell_3)$ is added $n^{1-\epsilon} \cdot (m+1)$
     times.
  4. Execute $\mathcal{A}$ on $\phi'$ and denote $a' = \mathcal{A}(\phi')$.
  5. Choose $i \in \{1, 2, 3\}$ such that $a'(x_i) \neq a(x_i)$. Let $\sigma = a(x_i)$.
  6. Return $\langle i, \sigma \rangle$.
```

By our choice of ℓ_1, ℓ_2, ℓ_3, the clause $(\ell_1 \vee \ell_2 \vee \ell_3)$ is not satisfied by the assignment a. Hence, as $\mathcal{A}$ is a $1/n^{1-\epsilon}$ approximation algorithm, at least one of x_1, x_2, x_3 changes assignments between a and a'. Hence, as $\mathcal{A}$ respects the privacy structure $\mathcal{R}_{\text{maxE3SAT}}$, we conclude that the formulae ϕ and ϕ' differ on their sets of maximum assignments. The following claim implies the correctness of algorithm `Relevant Variable-Assignment`:

Claim B.4 *If $a'(x_i) \neq a(x_i)$ for some $i \in \{1, 2, 3\}$, then x_i is $a(x_i)$-relevant.*

Proof: Assume the contrary, i.e. that for every maximum assignment for ϕ assigns $a'(x_i)$ to x_i. It is easy to see that every maximum assignment of ϕ is also a maximum assignment for ϕ' as it satisfies the added clauses $(\ell_1 \vee \ell_2 \vee \ell_3)$. In the reverse direction, note that every maximum assignment for ϕ' is also maximum for ϕ as the assignment to the two other variables from x_1, x_2, x_3 does not affect the satisfiability of the clause $(\ell_1 \vee \ell_2 \vee \ell_3)$. We get that $\langle \phi, \phi' \rangle \in \mathcal{R}_{\text{maxE3SAT}}$, contradicting $\mathcal{A}(\phi) \neq \mathcal{A}(\phi')$. □