Guessing Two Secrets with Small Queries

Daniele Micciancio and Nathan Segerlind
Department of Computer Science
University of California, San Diego
La Jolla, CA 92093
daniele@cs.ucsd.edu, nsegerli@cs.ucsd.edu

Abstract

We present an adaptive strategy to recover two secrets from an adversary which uses linear size queries and is computationally efficient, solving a problem left open in [2]. To obtain an intersecting family containing the two n-bit secrets, $4n + 3$ queries are are made, and time $O(n^2)$ and space $O(n)$ are consumed.

1 Introduction

The problem of recovering one of two n-bit secrets from an adversary was introduced in [2]. A malevolent but truthful adversary has two secret n-bit strings. A seeker repeatedly asks questions of the form "Does a string belong to a set X?". The adversary must give the correct answer for one of the strings, but can choose which string to use when answering. It is possible for the adversary to answer in such a way that the seeker cannot learn both secrets. However, there are strategies by which the seeker can learn either one of the secrets, or a triple of strings which contains both of the secrets. Such problems have arisen recently in connection with certain Internet traffic routing applications.

In theorem one of [2], an adaptive algorithm requiring at most $4n + 3$ queries was demonstrated. However, the queries made require 2^n bits of representation, so this recovery algorithm requires space $\Omega(2^n)$! A strategy making $O(n^3)$ many queries each of size $O(n)$, requiring $O(n^4)$ time was discussed. It was let open to provide a strategy which recovers the secrets efficiently with the optimal, $\Theta(n)$, number of queries. We answer this question in the affirmative by demonstrating an adaptive strategy which makes $4n + 3$ queries, each of size $O(n)$, and runs in time $O(n^2)$. Not only is this strategy efficient, but it makes use of the least number of queries currently known for *any* strategy.

Independently, Noga Alon, Venkatesan Guruswami, Tali Kaufman and Madhu Sudan have shown a $O(n^3)$ time, $O(n)$ size *oblivious* strategy which makes $\Theta(n)$ many queries in [1]. While our strategy makes fewer queries, by a linear factor, and recovers the secrets more efficiently, it makes great use of asking different queries at different stages, and is inherently adaptive.

2 The Two Secrets Problem

We adopt the formulation of the 2-secrets problem as a search for a hidden edge in a graph. Consider the complete graph on length n binary strings. The adversary has a hidden pair of strings, an edge. The goal of the seeker is to learn the hidden edge. The seeker makes a query by proposing a set of vertices,

[*] NSF Career Award CCR-0093029
[†] Partially supported by NSF grant DMS-9803515

X. The adversary must choose one of the secrets, s, and answer the question $x \in S$ correctly. If the adversary answers "YES", edges disjoint from X are deleted, if the adversary answers "NO", edges contained in X are deleted.

It can be shown, [2], [1], that the best the seeker can hope to do is to determine an intersecting family of edges so that the secret edge belongs to the family. Because the only intersecting families of edges are stars (sets of edges adjacent to a common vertex) and triangles, the seeker learns either one secret or a triple that contains both secrets. It is in this sense that the seeker is said to have a strategy for the 2-secrets problem.

We use the following notation from [2] to identify the intermediate graphs which arise during the seeker's secret search.

Definition 2.1 *Let A, B and C be disjoint set of n bit strings.*

- *$K(A, B)$ is the complete bipartite graph between vertex sets A and B.*
- *$\overline{K}(A, B)$ is the graph formed by joining every vertex in a clique on A to every vertex in an independent set on B.*
- *$K(A, B, C)$ is the complete tripartite graph with vertex sets A,B,C.*

Notice that the intersecting families are of the form $K(\{s\}, B)$ and $K(\{s_1\}, \{s_2\}, \{s_3\})$.

3 Using Prefixes to Guess Secrets

3.1 Representation of Sets

We will not work with arbitrary sets of strings, rather we will work with boolean combination of sets specified by prefixes. Indeed, all sets of strings used in the course of the strategy will be represtable by a certain simple combination of at most three prefixes. This is what keeps the seeker's memory requirements low.

Definition 3.1 *Let $\alpha \in \{0, 1\}^{\leq n}$ be given. Set $T_\alpha := \{x \in \{0, 1\}^n \mid \alpha \preceq x\}$.*

As the reader will see, all sets of vertices used in the course of the strategy are of the form T_α, T_α^c, or $T_\alpha \cup T_\beta$.

3.2 The Seeker's Strategy

Our strategy is the same as the one presented in [2], with a modification in the bookkeeping. Rather than abitrarily partition the strings so as to bisect the sets, we partition the trees so as to increase the length of the common prefix of each.

The first step of the strategy is to make the query "Does the secret begin with 0?". If the answer is "YES", let $\alpha =' 0'$. If the answer is "NO", let $\alpha =' 1'$. The seeker then reduces $\overline{K}(T_\alpha, T_\alpha^c)$.

3.2.1 Reducing $\overline{K}(T_\alpha, T_\alpha^c)$

At most two queries are made, $T_{\alpha 0}$ and $T_{\alpha 1}$. We proceed as diagrams 1 and 2 indicate, reducing to either $\overline{K}(T_{\alpha 0}, T_{\alpha 0}^c)$, $\overline{K}(T_{\alpha 1}, T_{\alpha 1}^c)$, or $K(T_{\alpha 0}, T_{\alpha 1}, T_\alpha^c)$.

We leave this phase either when the length of α becomes n, and we have a star centered at α, or we reduce to a graph of the form $K(T_{\alpha 0}, T_{\alpha 1}, T_\alpha^c)$.

If the graph has reduced to $K(T_{\alpha 0}, T_{\alpha 1}, T_\alpha^c)$, we proceed by reducing $K(T_{\alpha 0}, T_{\alpha 1}, \Omega)$. This is because in the usual case, when α is not very short, T_α^c is almost as large as Ω. It is of no help to use this et rather than the set of all strings. By reducing to $K(T_{\alpha 0}, T_{\alpha 1}, \Omega)$, we slightly reduce the complexity of our queries, and simplify the presentation of the strategy.

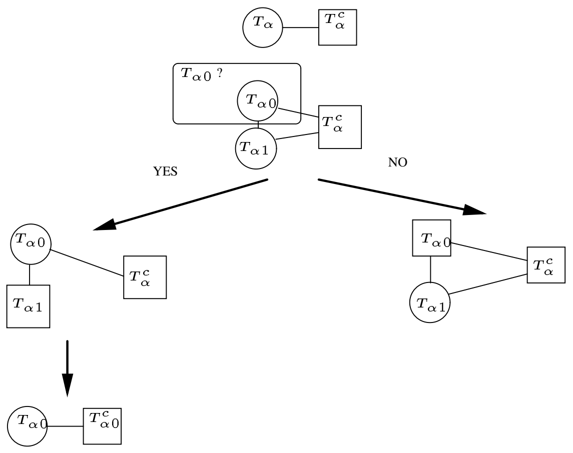

Figure 1. The "YES" branch repeats figre 1, the "NO" branch follows figure 2

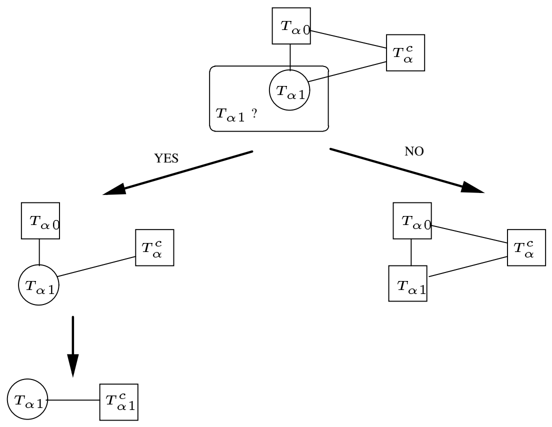

Figure 2. The "YES" branch follows figure 1, the "NO" branch follows figure 3

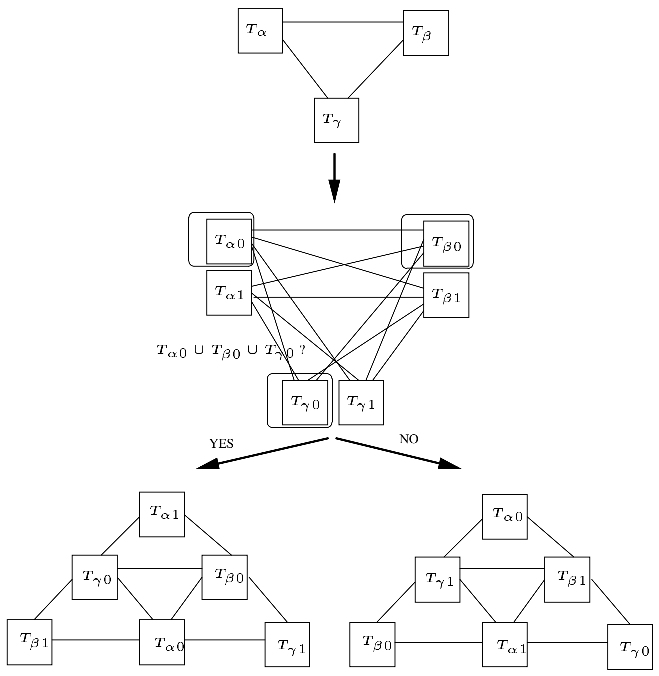

Figure 3. The "YES" branch follows diagram 4, the "NO" branch is handled symmetrically

3.2.2 Reducing $K(T_\alpha, T_\beta, T_\gamma)$

First we consider the case when $|\alpha|, |\beta|, |\gamma| < n$.

We split each of T_α,T_β and T_γ into two pieces, and make four queries.

First, the query $T_{\alpha 0} \cup T_{\beta 0} \cup T_{\gamma 0}$ is made, and we proceed as inicated by figure 3.

Notice that the resulting graphs in figure 3 are the same *up to swapping $T_{\alpha 0}$ and $T_{\alpha 1}$, $T_{\beta 0}$ and $T_{\beta 1}$, $T_{\gamma 0}$ and $T_{\gamma 1}$*. For this reason, diagrams 4, 5, 6, 7, 8. show only the behavior to be taken under the "YES" branch of figure 3. The strategy proceeds in much the same way under the "NO" branch of figure 3, with the exception that the "YES" and "NO" branches are transposed. This works because every query made in this phase is of the form

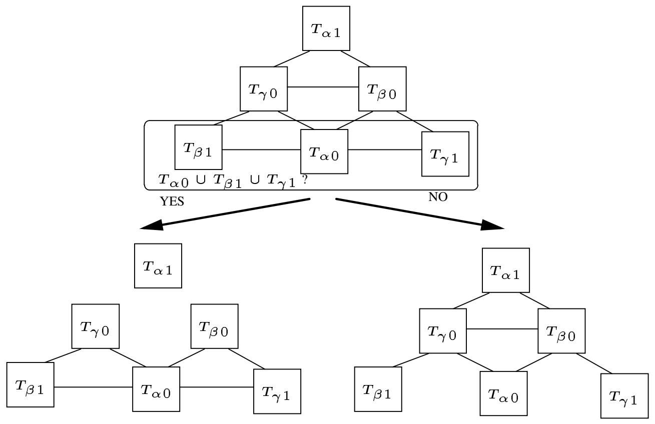

Figure 4. The "YES" branch follows diagram 5, the "NO" branch follows 7

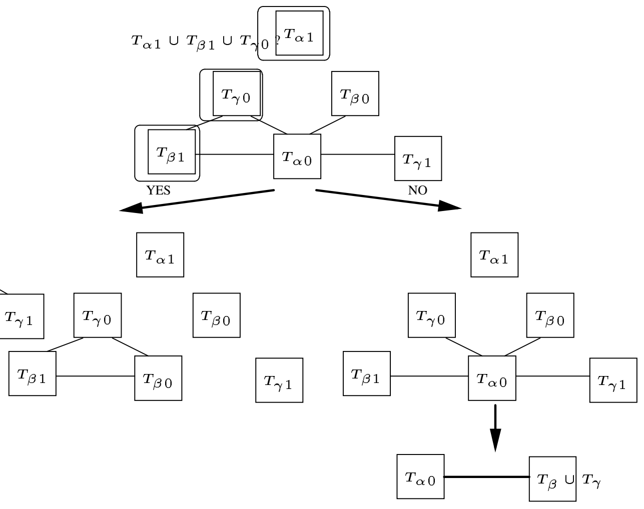

Figure 5. The "YES" branch follows 3, the "NO" branch follows 6

$T_{\alpha\epsilon_1} \cup T_{\beta\epsilon_2} \cup T_{\gamma\epsilon_3}$.

The second step is to make the query $T_{\alpha 0} \cup T_{\beta 1} \cup T_{\gamma 1}$. Figure 4 shows how we precede, *on the graph resulting when the answer to the first query is "YES"*.

Finally, we make the queries $T_{\alpha 1} \cup T_{\beta 0} \cup T_{\gamma 1}$ and $T_{\alpha 1} \cup T_{\beta 1} \cup T_{\gamma 0}$. Figures 5 and 6 show how we proceed in the case arising from the first possibility in the answer to the second query and figures 7 and 8 show how to proceed in the other case.

Next we consider the case when one of the trees consists of a single string. Because of how the procedure for reducing $\overline{K}(T_\alpha, T_\alpha^c)$ works, when we first obtain a graph of the form $K(T_\alpha, T_\beta, T_\gamma)$, we have that $T_\alpha = T_{\delta 0}$, $T_\beta = T_{\delta 1}$ and $T_\gamma = \Omega$, for some string δ. Each pass through this stage of the search increases the length of the prefix of each set by exactly one. Therefore, if there is one set which is a singleton, then there are two sets which are singletons. If all three sets are singletons, then we are finished. So we need only consider the case when there exactly two singleton sets.

To reduce such graphs, we split the non-singleton set and make two queries: $T_\alpha \cup T_{\gamma 0}$ and $T_\beta \cup T_{\gamma 0}$.

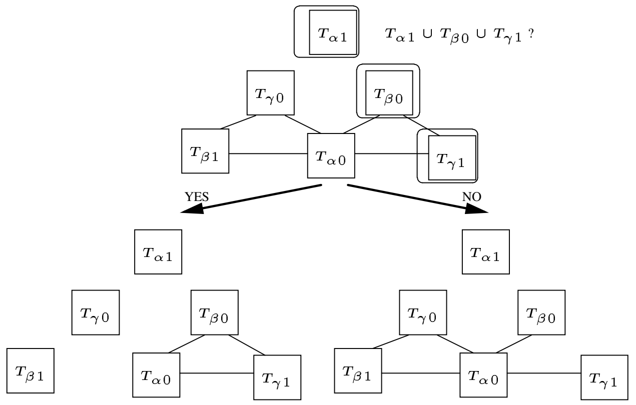

Figure 6. The "YES" branch follows 3, the "NO" branch follows 14

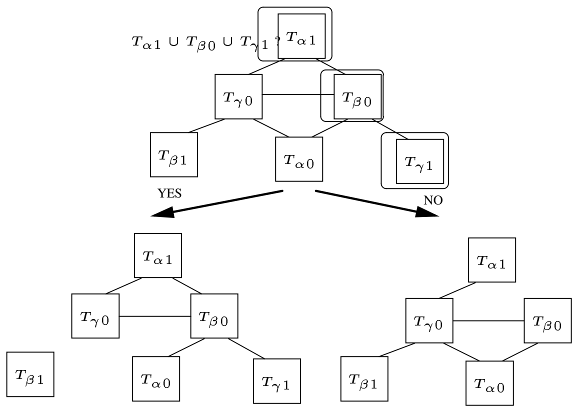

Figure 7. The "YES" branch follows diagram 8, the "NO" branch follows diagram 9

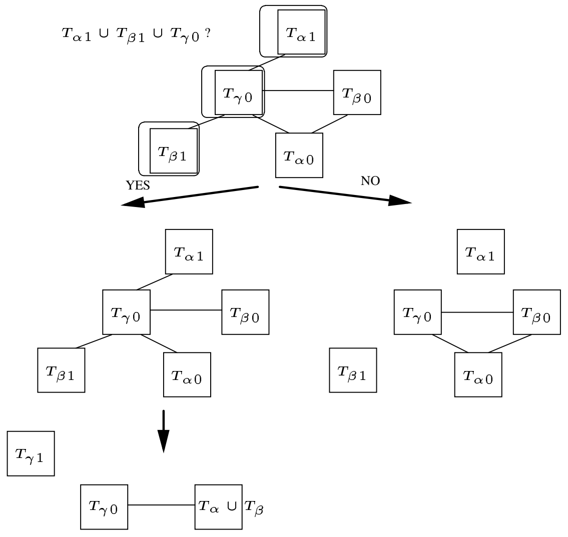

Figure 9. Both branches follow diagram 14

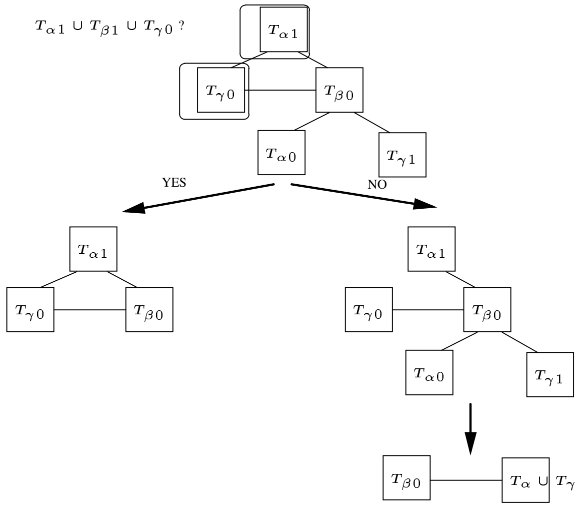

Figure 8. Both branches follow diagram 14

We proceed as in figures 10 and 11. Notice that in figure 10, the resulting graphs are identical up to the swapping of $T_{\gamma 0}$ and $T_{\gamma 1}$. For this reason, the graph resulting under the "NO" branch can be reduced as in figure 11, but with the query $T_\alpha \cup T_{\gamma 1}$.

Finally, we consider the form of the graphs resulting from this stage of the strategy. Intially, the graph is of the form $K(T_{\alpha 0}, T_{\alpha 1}, \Omega)$.

It is easily checked that as we repeat this stage, the graph will always have the form $K\,(T_\beta, T_\gamma, T_\delta)$ or $K\,(T_\alpha, T_\beta \cup T_\delta)$.

3.2.3 Reducing $K(T_\alpha, T_\beta \cup T_\gamma)$

If $|\alpha| = n$, then $K(T_\alpha, T_\beta \cup T_\gamma)$ is a star and we are done, so we consider the case when $|\alpha| < n$. We split T_α into two pieces, and make two queries $T_{\alpha 0} \cup T_\beta$ and $T_{\alpha 0} \cup T_\gamma$, following diagram 12.

It is easily verified that if the graph has the form $K(T_\alpha, T_\beta \cup T_\gamma)$ in the beginning of one of these passes, afterwards it will have the form $K(T_\alpha, T_\beta \cup$

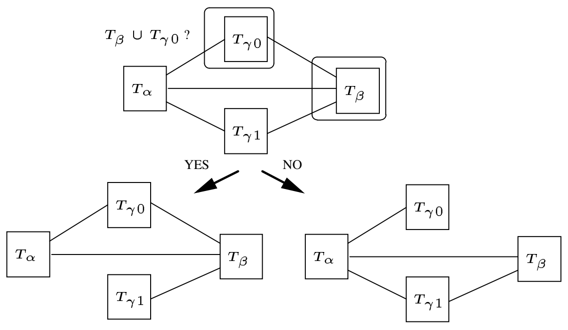

Figure 10. The "YES" branch follows diagram 11, the "NO" branch follows by symmetry.

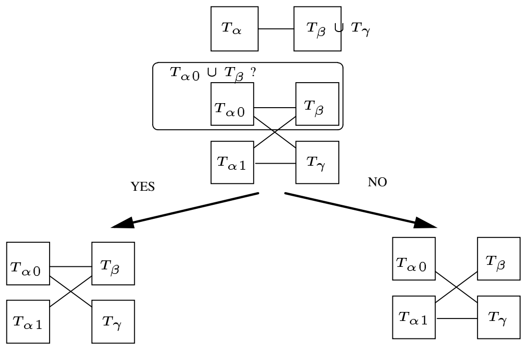

Figure 12. The "YES" branch follows figure 13, the "NO" branch follows symmetrically

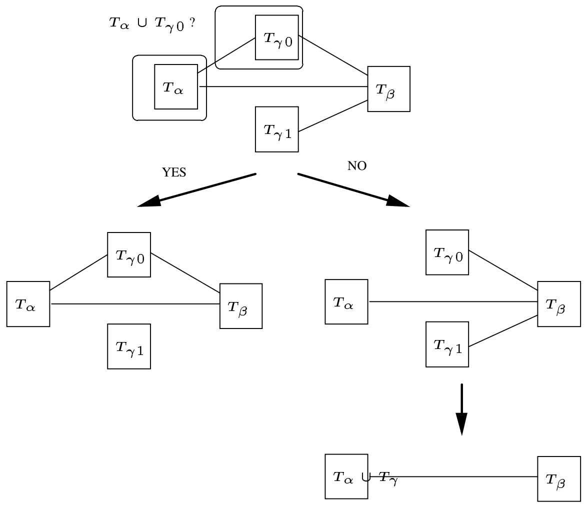

Figure 11. The "YES" branch follows diagram 10, the "NO" branch follows diagram 14

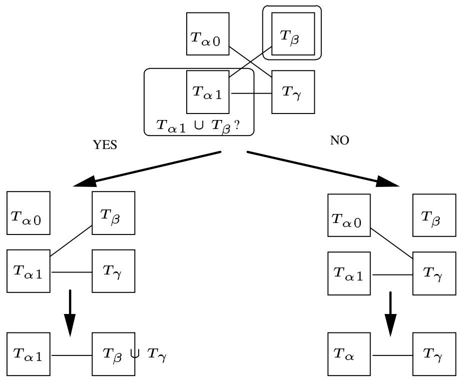

Figure 13. Both branches follow figure 14

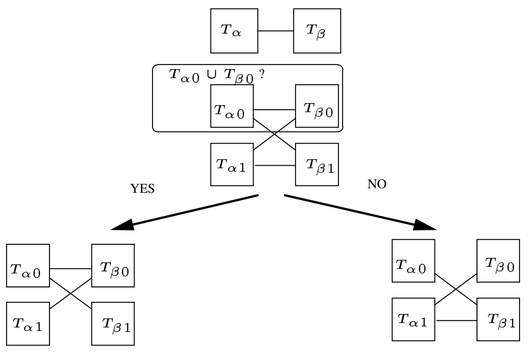

Figure 14. The "YES" branch follows figure 15, the "NO" branch follows symmetrically

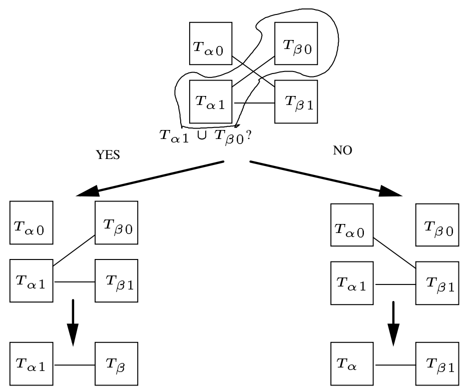

Figure 15. Both branches follow figure 14

T_γ) or $K(T_\alpha, T_\beta)$.

3.2.4 Reducing $K(T_\alpha, T_\beta)$

If either $|\alpha| = n$ or $|\beta| = n$, then $K(T_\alpha, T_\beta)$ is a star and we are done, so we consider the case when $|\alpha|, |\beta| < n$. We split T_α and T_β, and make two queries $T_{\alpha 0} \cup T_{\beta 0}$ and $T_{\alpha 0} \cup T_{\beta 1}$, following diagram 14.

4 Complexity Analysis

Clearly, every query can be specified with $O(n)$ many bits.

To bound the number of iterations, and hence the time complexity, we first bound the number of iterations for each of the sub-tasks: reducing graphs of the form $K(T_\alpha, T_\beta)$, $K(T_\alpha, T_\beta \cup T_\gamma)$, $K(T_\alpha, T_\beta, T_\gamma)$ and finally $\overline{K}(T_\alpha, T_\alpha^c)$.

Let the *unnamed length* of a string α be $n - |\alpha|$.

Let $f(n_1, n_2)$ denote the number of queries needed to reduce $K(T_\alpha, T_\beta)$ where α and β have unnamed lengths n_1, n_2 respectively. Clearly, $f(n_1, n_2) \leq 2n_1 + 2n_2$.

Let $g(n_1, n_2, n_3)$ denote the number of queries needed to reduce $K(T_\alpha, T_\beta \cup T_\gamma)$, where α, β and γ have unnamed lenghts n_1, n_2, n_3, respectively. WLOG, assume $n_2 \geq n_3$. The strategy of subsubsection 3.2.3 will either extend α to a length n string, or make i passes (lengthening α by i bits) before removing one of T_β or T_γ and moving to reducing a graph of the form $K(T_\alpha, T_\beta)$. Therefore, $g(n_1, n_2, n_3) \leq \max(2n_1, 2 + 2i + f(n_1 - i, n_2)) = \max(2n_1, 2 + 2i + 2n_1 - 2i + 2n_2) = 2 + 2n_1 + 2n_2$.

Let $h(n_1, n_2)$ denote the number of queries needed to reduce $K(T_\alpha, T_\beta, T_\gamma)$, where γ has unnamed length n_1, and α, β have unnamed length n_2. In the strategy of subsubsection 3.2.2, we either reduce to a triangle, or, we reduce to a graph of the form $K(T_\alpha, T_\beta \cup T_\gamma)$. If we reduce to a triangle, a total of $4n_2 + 2(n_1 - n_2) = 2n_1 + 2n_2$ queries are made. If we reduce to a bipartite graph, the number of queries is bounded by $\max_i(4i + 4 + g(n_1 - i - 1, n_2 - i)) = \max_i(4i + 4 + 2n_1 - 2i - 2 + 2n_2 - 2i + 2) = 2n_1 + 2n_2 + 4$.

Reducing $\overline{K}(T_\alpha, T_\alpha^c)$, when α has unnamed length n_1 therefore takes at most $\max(2n_1, \max_i(2i + h(n, n_1 - i))) =$

$\max(2n_1, 2i + 2n + 2n_1 - 2i + 2) = 2n + 2n_1 + 4$.

Therefore, because the strategy starts by reducing $\overline{K}(\Omega, \emptyset)$ by one query, it makes at most $1 + 2n + 2(n-1) + 4 = 4n + 3$ many queries before arriving at an intersecting family.

Because there are $O(n)$ many queries made, each query requires $O(n)$ many bits, and the computation between received answers is an analysis of a constant number of cases, the time complexity of this strategy is at most $O(n^2)$.

References

[1] Noga Alon, Tali Kaufman, Venkat Guruswami, and Madhu Sudan. Guessing secrets efficiently via list decoding, 2001. To appear in SODA 2002.

[2] Fan Chung, Ronald Graham, and Tom Leighton. Guessing secrets. *The Electronic Journal of Combinatorics*, 8, 2001.