Guessing More Secrets via List Decoding

Alexander A. Razborov *

August 10, 2004

Abstract

We consider the following game introduced by Chung, Graham and Leighton in [CGL01]. One player, **A** picks $k > 1$ secrets from a universe of N possible secrets, and another player, **B** tries to gain as much information about this set as possible by asking binary questions $f : [N] \longrightarrow \{0,1\}$. Upon receiving a question f, **A** adversarially chooses one of her k secrets, and answers f according to it.

In this paper we present an explicit set of $2^{O(k)}(\log N)$ questions, along with an $2^{O(k^2)}(\log^2 N)$ recovery algorithm that achieve **B**'s goal in this game. This, in particular, completely solves the problem for any constant number of secrets k.

Our strategy is based on the list decoding of Reed-Solomon codes, and it extends and generalizes ideas introduced by Alon, Guruswami, Kaufman and Sudan in [AGKS02].

1 Introduction

Motivated in part by certain Internet traffic routing applications (see [Pet02] for a popular account), Chung, Graham and Leighton introduced in [CGL01] the following neat combinatorial problem. Two players, **A** and **B** play the game defined as follows. **A** picks a k-element subset S of an N-element universe $[N]$ representing "secrets" (k is typically to be thought of as of a constant, and $[N]$ as of the set of binary strings of length $\log_2 N$). Then **B**

*Institute for Advanced Study, Princeton, US, on leave from Steklov Mathematical Institute, Moscow, Russia, razborov@ias.edu. Supported by the State of New Jersey, The Bell Companies Fellowship, The James D. Wolfensohn Fund, and The Ellentuck Fund.

tries to get as much information about this set as possible by asking binary questions $f : [N] \longrightarrow \{0, 1\}$. If $f|_S \equiv 0$ or $f|_S \equiv 1$, then **A** must give this unique answer, otherwise she may choose to answer 0 or 1 arbitrarily. What is the best strategy for **B** in this situation?

Obviously, we still need to make several important clarifications in this loose description of the game to make a rigorous mathematical problem out of it. The first question is how much information about S can the player **B** hope to get in principle, say, after he asks all possible questions whatsoever?

The answer to this question was given already in the original paper [CGL01]. Recall that a family $\mathcal{S} \subseteq [N]^k$ of k-element subsets of $[N]$ is *intersecting* if $S \cap S' \neq \emptyset$ for every $S, S' \in \mathcal{S}$. Then it turns out that for every intersecting family $\mathcal{S}$, **A** always has a strategy that allows her to "protect" every potential set of secrets $S \in \mathcal{S}$, so that **B** can not hope to learn anything more than the fact $S \in \mathcal{S}$. On the other hand, **B** can always end up with an intersecting family $\mathcal{S}$ that includes all sets of secrets S consistent with **A**'s answers. For these reasons, **B**'s goal in this game is *defined* as finding any such intersecting family $\mathcal{S}$.

Next, **B**'s strategy can be *oblivious* (the set of his questions is presented to **A** at once) or *adaptive* (questions may depend on **A**'s previous answers).

Finally, there are three levels of efficiency requirements "good" strategies for **B** may be asked to satisfy. The weakest of them simply asks that the overall number of questions is bounded by a polynomial in $\log N$ (for the time being we assume that k is a constant). On the second level ("constructive strategies" in the terminology of [CGL01]) we also demand that the queries $f : [N] \longrightarrow \{0, 1\}$ should themselves be computable in time $(\log N)^{O(1)}$. Finally, *invertible strategies* must in addition be able to recover, within the same time $(\log N)^{O(1)}$, the actual answer $\mathcal{S}$.[1]

A number of good strategies for **B** based on quite different ideas were presented in [CGL01] and in the subsequent papers [CGL02, AGKS02, MS04]. For the most well-studied case $k = 2$ an oblivious constructive strategy was given in the original paper [CGL01]. The first (oblivious) *invertible* strategy for $k = 2$ was also sketched in that paper, and it later appeared, with a complete proof, in [CGL02]. [AGKS02] presented another invertible strategy

[1] At the first sight, this requirement may look somewhat confusing since even the bit length of any description of $\mathcal{S}$ may a priori be super-polynomial in $\log N$. We will see below, however, that succinct answers $\mathcal{S}$ are in fact always possible, and an invertible strategy must necessarily produce one.

with slightly better performance: it asks $O(\log N)$ questions and recovers $\mathcal{S}$ within time $O(\log^3 N)$.

When $k > 2$, constructive oblivious strategies with $O(\log N)$ queries were presented in [AGKS02]. Much less, though, has been known about *invertible* strategies. The case $k = 3$ has been recently solved in [MS04] (an oblivious strategy with $O(\log^5 N)$ queries, $O(\log^5 N)$ recovery time, and an adaptive one with $O(\log N)$ queries, $O(\log^3 N)$ recovery time). What about $k > 3$, it seems that the only known partial result was a constructive (oblivious) strategy, equipped with a polynomial time randomized algorithm to produce a short list of candidates $L \subseteq [N]$ that is guaranteed to contain at least one of the k secrets [AGKS02].

In this paper we present an oblivious invertible strategy with $O(\log N)$ queries and $O(\log^2 N)$ recovery time allowing to guess *any* constant number of secrets k. The multiplicative constants assumed in the last sentence are $2^{O(k)}$ and $2^{O(k^2)}$, respectively, so our strategy in fact stays invertible up to $k \leq \sqrt{\log\log N}$ (and stays constructive up to $k \leq O(\log\log N)$).

Our techniques are based on the list-decoding algorithm for Reed-Solomon codes [GS99] and extend those from [AGKS02]. Perhaps, the main technical contribution of this paper consists in revealing the full potential of that powerful algorithm for the secret guessing.

2 Preliminaries and the main result

Throughout the paper, $[N] \stackrel{\text{def}}{=} \{1, 2, \ldots, N\}$, and for a finite set U, $[U]^{\leq k} \stackrel{\text{def}}{=} \{S \subseteq U \mid |S| \leq k\}$. $[[n]]^{\leq k}$ will be abbreviated to $[n]^{\leq k}$. $[U]^{\leq k}$ is partially ordered w.r.t. inclusion, and for $\mathcal{S} \subseteq [U]^{\leq k}$ we let

$$\lceil \mathcal{S} \rceil \stackrel{\text{def}}{=} \left\{ S \in [U]^{\leq k} \mid \exists S' \in \mathcal{S}(S' \subseteq S) \right\}$$

denote its upward closure w.r.t. this ordering (k will always be clear from the context). A family $\mathcal{S} \subseteq [U]^{\leq k}$ is *intersecting* if $S \cap S' \neq \emptyset$ for every $S, S' \in \mathcal{S}$. Clearly, $\mathcal{S}$ is intersecting if and only if $\lceil \mathcal{S} \rceil$ is so. For $\mathcal{S} \subseteq [U]^{\leq k}$, let $\mathcal{S}^\flat \subseteq \mathcal{S}$ denote the family of all minimal elements in $\mathcal{S}$, and let its *support* $Sup(\mathcal{S}) \subseteq U$ be defined as the union of all sets in $\mathcal{S}^\flat$. Clearly, $\mathcal{S}^\flat$ uniquely defines $\lceil \mathcal{S} \rceil$ (and vice versa), and every individual fact $S \in \lceil \mathcal{S} \rceil$ depends only on the intersection of S with the support $Sup(\mathcal{S})$. An intersecting family $\mathcal{S}$

is *critical* if for every $S \in \mathcal{S}^\flat$ and every proper subset $S' \subset S$, $\mathcal{S} \cup \{S'\}$ is no longer intersecting.

Proposition 2.1 ([Tuz85]) *Let $\mathcal{S} \subseteq [U]^{\leq k}$ be any critical intersecting family. Then $|Sup(\mathcal{S})| \leq \binom{2k}{k} \leq 4^k$ which, in turn, implies $|\mathcal{S}^\flat| \leq \exp(O(k^2))$.*

We now proceed to the description of the secret guessing game. Our main result holds in its slightly more general (and much more natural) version when $\mathbf{A}$ is allowed to hold a set of $\leq k$ (rather than exactly k) secrets, and we modify the definition accordingly.

Definition 2.2 An *oblivious strategy for* $\mathbf{B}$ *in the secret guessing game* is simply a sequence of Boolean functions (queries) $f_1, \ldots, f_t : [N] \longrightarrow \{0,1\}$. Given an integer k, this strategy is *k-separating* if for every sequence $a_1, \ldots, a_t \in \{0,1\}$ of answers the family

$$\mathcal{S}_{a_1,\ldots,a_t} \stackrel{\text{def}}{=} \left\{ S \in [N]^{\leq k} \mid \forall i \in [t](a_i \in f_i(S)) \right\}$$

(made of all sets of secrets consistent with $\mathbf{A}$'s answers) is intersecting.

[AGKS02] observed that the minimal number of queries t in a k-separating strategy is $2^{\theta(k)}(\log N)$. Accordingly, we call a k-separating strategy *constructive* if $f_1, \ldots, f_t$ can be computed within time $2^{O(k)}(\log N)^{O(1)}$, that is, polynomial in this quantity (if we are interested only in the case when k is a constant, the factor $2^{O(k)}$ becomes irrelevant, of course). Note that the number of queries in every constructive strategy is also bounded by $2^{O(k)}(\log N)^{O(1)}$. Constructive strategies with the optimal number of queries $2^{O(k)}(\log N)$ do exist [AGKS02].

The family $\mathcal{S}_{a_1,\ldots,a_t}$ itself can be extremely complex, therefore in general (i.e., for arbitrary k-separating strategies) we can not hope to output its description within reasonable time. Proposition 2.1, however, implies that there always exists a small intersecting $\mathcal{S}$ such that $\lceil \mathcal{S} \rceil \supseteq \mathcal{S}_{a_1,\ldots,a_t}$ (in fact, any critical extension of $\mathcal{S}_{a_1,\ldots,a_t}$ would do), and we define an *invertible strategy* as a k-separating constructive strategy equipped with a polynomial (also in $2^k \log N$) algorithm that for every $a_1, \ldots, a_t \in \{0,1\}$ outputs some intersecting $\mathcal{S} \subseteq [N]^{\leq k}$ such that $\lceil \mathcal{S} \rceil \supseteq \mathcal{S}_{a_1,\ldots,a_t}$.

Remark 1 The existing literature is rather vague on the issue whether an invertible strategy must necessarily output the exact family of possible sets of secrets $\mathcal{S}_{a_1,\ldots,a_t}$ or an intersecting upper bound $\mathcal{S}$ on it. This especially applies to the main case of interests to us $k > 2$, when intersecting families become extremely complex. At any rate, we think that the definition given above is more natural and better conforms to the stated goal "output as much information about $\mathcal{S}_{a_1,\ldots,a_t}$ as possible", and our reasoning is this. Suppose that the intersecting family $\mathcal{S}_{a_1,\ldots,a_t}$ is "far" from being maximal, and our algorithm is able to produce an intersecting $\mathcal{S}$ for which $\lceil \mathcal{S} \rceil$ is "much larger" than $\mathcal{S}_{a_1,\ldots,a_t}$. As was mentioned in Introduction, **A** then has another strategy $b_1, \ldots, b_t$ such that $\mathcal{S}_{b_1,\ldots,b_t} \supseteq \lceil \mathcal{S} \rceil \supset \mathcal{S}_{a_1,\ldots,a_t}$; in other words, this strategy allows her to protect "many more" sets of secrets than the original strategy $a_1, \ldots, a_t$. Thus, the only reason why we were able to produce some $\lceil \mathcal{S} \rceil$ significantly larger than $\mathcal{S}_{a_1,\ldots,a_t}$ stemmed from the fact that **A** played very badly: for any optimal (that is, when $\mathcal{S}_{a_1,\ldots,a_t}$ is maximal intersecting) strategy, we necessarily must have $\lceil \mathcal{S} \rceil = \mathcal{S}_{a_1,\ldots,a_t}$. We feel that requesting **B**'s goal family $\mathcal{S}$ to capture, among other things, how exactly bad his opponent **A** has been should not be a part of this definition.

In this paper we prove the following.

Theorem 2.3 *For every parameter $k = k(N)$ there exists a k-separating constructive strategy $f_1, \ldots, f_t : [N] \longrightarrow \{0,1\}$ with $t \leq 2^{O(k)}(\log N)$, equipped with an algorithm that has running time $2^{O(k^2)}(\log^2 N)$ and for every $a_1, \ldots, a_t \in \{0,1\}$ outputs some intersecting family $\mathcal{S} \subseteq [N]^{\leq k}$ such that $\lceil \mathcal{S} \rceil \supseteq \mathcal{S}_{a_1,\ldots,a_t}$.*

In particular, this strategy is invertible as long as $k \leq \sqrt{\log\log N}$.

The main ingredient of our algorithm is borrowed from [AGKS02], and this is the list decoding of Reed-Solomon codes. More specifically, we need the following statement that was implicit in [GS99], and was extracted in an explicit form in [AGKS02]. We formulate it here only for the "basic" Reed-Solomon codes with $n = q$, as we do not need it in the generalized $(n < q)$ case.

Proposition 2.4 ([GS99], [AGKS02, Theorem 4.1]) *Let q be a power of a prime, and d, ℓ be arbitrary parameters. Given lists $L_x \subseteq \mathbb{F}_q$ $(x \in \mathbb{F}_q)$ with each $|L_x| \leq \ell$, there are at most $O(\sqrt{q\ell/d})$ polynomials $p \in \mathbb{F}_q[x]$ of degree $\leq d$ such that $p(x) \in L_x$ for at least αq elements of $\mathbb{F}_q$, provided $\alpha \geq C_0\sqrt{d\ell/q}$ (C_0 a sufficiently large constant). Moreover, the list of all such polynomials can be found in $O(q^2\ell^2 \log^2 q)$ time.*

3 The strategy

We first describe a k-separating constructive strategy that satisfies the required bound $t \leq 2^{O(k)}(\log N)$ on the number of queries, but whose recovery algorithm has slightly worse running time $2^{O(k^2)}(\log N)^{O(k)}$ (note, however, that already this strategy is invertible for any constant k). Then we show how to improve the running time of the recovery algorithm to $2^{O(k^2)}(\log^2 N)$ by composing our strategy with itself.

So, let k, N be given. Let

$$d(q) \stackrel{\text{def}}{=} \lfloor q/((4C_0k)^2 4^k) \rfloor, \tag{1}$$

where C_0 is the constant from Proposition 2.4, and q be the smallest prime such that

$$q^{d(q)+1} \geq N. \tag{2}$$

Note that by Chebyshev's theorem we have the bound

$$q \leq 2^{O(k)}(\log N/\log\log N). \tag{3}$$

Take the Reed-Solomon code C consisting of all polynomials $p(x)$ in $\mathbb{F}_q[x]$ of degree $\leq d(q)$. Then, by (2), $|C| \geq N$. We identify secrets $s \in [N]$ with codewords in C via an easily computable embedding, and assume from now on that the $\leq k$ secrets to be guessed are just polynomials from C.

Let $g_x : C \longrightarrow \mathbb{F}_q$ ($x \in \mathbb{F}_q$) be the evaluation functions; that is, $g_x(p) \stackrel{\text{def}}{=} p(x)$. Consider also any k-separating constructive strategy $h_1, \ldots, h_u : \mathbb{F}_q \longrightarrow \{0,1\}$ with the optimal number of queries $u \leq 2^{O(k)}(\log q)$. Our final strategy $f_1, \ldots, f_t : C \longrightarrow \{0,1\}$ is obtained simply by concatenating g and h; that is, it will consist of the Boolean queries $h_i \circ g_x : C \longrightarrow \{0,1\}$, where $x \in \mathbb{F}_q$ and $i \in [u]$. Note at once that, by (3), the number of queries made by this strategy is $uq \leq 2^{O(k)}(q \log q) \leq 2^{O(k)}(\log N)$, as desired. The queries $f_1, \ldots, f_t$ are obviously computable within polynomial time $2^{O(k)}(\log N)^{O(1)}$. It is also easy to see that the strategy $f_1, \ldots, f_t$ is in fact k-separating, but since this automatically follows from the existence of the recovery algorithm, we skip the proof and proceed immediately to that algorithm.

So, assume we are given a set of answers $a_{xi} \in \{0,1\}$ ($x \in \mathbb{F}_q, i \in [u]$) to our queries $h_i \circ g_x$. Since the strategy $h_1, \ldots, h_u$ is k-separating, for every particular $x \in \mathbb{F}_q$ the family $\mathcal{S}_{a_{x1},\ldots,a_{xu}} \subseteq [\mathbb{F}_q]^{\leq k}$ defined by the related answers is intersecting, and we recover it simply by trying out all elements in $[\mathbb{F}_q]^{\leq k}$.

Next, we construct *critical* intersecting families $\mathcal{S}_x$ with $\mathcal{S}_x \supseteq \mathcal{S}_{a_{x1},\ldots,a_{xu}}$. This is done by the straightforward algorithm which first sets $\mathcal{S}_x := \mathcal{S}_{a_{x1},\ldots,a_{xu}}$, and then consecutively tries to append to $\mathcal{S}_x$ new sets $S \in [\mathbb{F}_q]^k$ such that $\exists S' \in \mathcal{S}_x^\flat (S \subset S')$, while keeping the intersecting property, and until no new set can be added.

It follows from definitions that for every set of original secrets $S \in \mathcal{S}_{(\, a_{xi} \mid x \in \mathbb{F}_q, i \in [u])} \subseteq [C]^{\leq k}$ compatible with all the given answers, and for every $x \in \mathbb{F}_q$, we have $g_x(S) \in \lceil \mathcal{S}_x \rceil$ (cf. a similar argument in [AGKS02, Section 5.2]). This is the only property of S we need, and our problem can be now formulated as the recovery of an intersecting family $\mathcal{S} \subseteq [C]^{\leq k}$ such that

$$\forall S \in [C]^{\leq k}((\forall x \in \mathbb{F}_q(g_x(S) \in \lceil \mathcal{S}_x \rceil)) \Longrightarrow S \in \lceil \mathcal{S} \rceil). \tag{4}$$

Let us first try to convey some intuition as to how we are planning to do this. [AGKS02] called $L \subseteq [C]$ a *core* if every $S \in [C]^{\leq k}$ such that $\forall x \in \mathbb{F}_q(g_x(S) \in \lceil \mathcal{S}_x \rceil)$ has a non-empty intersection with L, and they showed how to recover a small core in a situation very similar to ours. We strengthen this notion by requiring that for every *pair* $S, S' \in [C]^{\leq k}$ with the above property, $S \cap S' \cap L \neq \emptyset$; call such an L *supercore*. First we will show how to construct a small supercore L. This will almost bring us to the end since the set $\left\{ S \cap L \,\middle|\, S \in [C]^{\leq k} \ \&\ \forall x \in \mathbb{F}_q(g_x(S) \in \lceil \mathcal{S}_x \rceil) \right\}$ will have small size and can be taken as the desired (intersecting) $\mathcal{S}$. One remaining problem is that even if this $\mathcal{S}$ is small, it does not seem to be efficiently recognizable: given $\tilde{S} \in [L]^{\leq k}$, there is no easy way to tell the *existence* of its extension $S \in [C]^{\leq k}$ with the required property $\forall x \in \mathbb{F}_q(g_x(S) \in \lceil \mathcal{S}_x \rceil)$. We will remedy this by designing an *efficient* test for $\tilde{S}$ which is passed by all desired $S \cap L$ and such that, on the other hand, the family of all sets passing this test is still intersecting.

Returning to the formal argument, let $L_x \stackrel{\text{def}}{=} Sup(\mathcal{S}_x)$; then $|L_x| \leq 4^k$ by Proposition 2.1. We apply Proposition 2.4 with d given by (1), $\ell := 4^k$, and $\alpha := \frac{1}{4k}$, and get a list $L \subseteq C$ of size $|L| \leq 2^{O(k)}$ such that

$$\forall p \notin L \left(|\, \{ x \in \mathbb{F}_q \mid p(x) \in L_x \} \,| \leq \frac{q}{4k} \right).$$

And now let

$$\mathcal{S} \stackrel{\text{def}}{=} \left\{ S \in [L]^{\leq k} \,\middle|\, |\, \{ x \in \mathbb{F}_q \mid g_x(S) \in \lceil \mathcal{S}_x \rceil \} \,| \geq \frac{3q}{4} \right\}.$$

We are left to do three more things: check (4), show that $\mathcal{S}$ is intersecting, and analyze the running time of the entire algorithm.

Checking (4)**.** Assume that $S \in [C]^{\leq k}$ has the property $\forall x \in \mathbb{F}_q (g_x(S) \in \lceil \mathcal{S}_x \rceil)$. It suffices to show that this implies $S \cap L \in \mathcal{S}$.

Due to the definition of the list L, for every "irrelevant" secret $p \in S \setminus L$ there are at most $\frac{q}{4k}$ values $x \in \mathbb{F}_q$ such that $p(x) \in L_x$. Altogether there at most $\frac{q}{4}$ bad x, and the remaining $\frac{3q}{4}$ values $x \in \mathbb{F}_q$ are good in the sense $g_x(S \setminus L) \cap L_x = \emptyset$. For every particular good x, $g_x(S) = g_x(S \cap L) \cup g_x(S \setminus L) \in \lceil \mathcal{S}_x \rceil$, and thus it contains some set in $\mathcal{S}_x^\flat$. Since, however, $g_x(S \setminus L) \cap Sup(\mathcal{S}_x) = \emptyset$, this set must be entirely contained in $g_x(S \cap L)$. Which proves $g_x(S \cap L) \in \lceil \mathcal{S}_x \rceil$ for every good x and, therefore, $S \cap L \in \mathcal{S}$.

$\mathcal{S}$ is intersecting. Suppose $S, S' \in \mathcal{S}$. Then for at least $q/2$ values x we have $g_x(S) \in \lceil \mathcal{S}_x \rceil$ *and* $g_x(S') \in \lceil \mathcal{S}_x \rceil$. Assume, for the sake of contradiction that $S \cap S' = \emptyset$. Since all elements in S, S' are polynomials of degree at most $d(q)$, the polynomials $p(x) - p'(x)$ $(p \in S,\ p' \in S')$ have altogether at most $k^2 d(q) < q/2$ roots. For any x which is not a root of one of these polynomials, $g_x(S) \cap g_x(S') = \emptyset$. Picking any such x for which also $g_x(S), g_x(S') \in \lceil \mathcal{S}_x \rceil$, we get a contradiction with the fact that $\mathcal{S}_x$ is intersecting.

Time analysis. Choosing an efficient encoding $[N] \longrightarrow C$ is easy. $[\mathbb{F}_q]^{\leq k}$ has cardinality $\leq q^k \leq 2^{O(k^2)}(\log N)^k$, and for any particular $x \in \mathbb{F}_q$ the membership in $\mathcal{S}_{a_{x1},\ldots,a_{xu}} \subseteq [\mathbb{F}_q]^{\leq k}$ can be tested in time $2^{O(k)}(\log q)^{O(1)}$ (recall that the strategy h is constructive). Therefore, all q families $\mathcal{S}_{a_{x1},\ldots,a_{xu}} \subseteq [\mathbb{F}_q]^{\leq k}$ can be constructed within time $2^{O(k^2)}(\log N)^{O(k)}$, and constructing their critical extensions $\mathcal{S}_x$ takes additional time which is also polynomial in q^k.

The bound $2^{O(k)}(\log^2 N)$ on the time needed to construct the list L comes directly from Proposition 2.4 and (3). Finally, $[L]^{\leq k}$ contains $2^{O(k^2)}$ elements, and computing every individual value $p(x)$ with $p \in L$ and $x \in \mathbb{F}_q$ requires time $O(d(q) \log^2 q)$. This implies the overall bound $2^{O(k^2)} d(q) q \log^2 q \leq 2^{O(k^2)}(\log^2 N)$ on the time needed to construct the family $\mathcal{S}$ from the supercore L.

We have constructed a constructive k-separating strategy $f_1, \ldots, f_t : [N] \longrightarrow \{0,1\}$ with the optimal number of queries $2^{O(k)}(\log N)$, and a recovery algorithm for this strategy with running time $2^{O(k^2)}(\log N)^{O(k)}$. Now we show how to improve the latter to $2^{O(k^2)}(\log^2 N)$.

Examining the time analysis given above, we see that the only step that really required that much time was the construction of the auxiliary intersecting families $\mathcal{S}_x$: all other steps were done within the prescribed time $2^{O(k^2)}(\log^2 N)$. We were not too picky about the choice of our "inner" strategy $h_1, \ldots, h_u : \mathbb{F}_q \longrightarrow \{0,1\}$, as long as it was constructive and asked the optimal number of queries $2^{O(k)}(\log q)$. Now, however, we can do much better and use as $h_1, \ldots, h_u$ the strategy we have just constructed, as well as the recovery algorithm for this strategy. In other words, we iterate our previous construction with itself *once* (if we attempt to iterate, in an obvious way, all the $(\log^* N)$ levels down, this will result in an $2^{O(k \log^* N)}$ factor in the number of queries, so we have to be careful). Then for every $x \in \mathbb{F}_q$ we will construct, in time $2^{O(k^2)}(\log q)^{O(k)}$, an intersecting $\tilde{\mathcal{S}}_x \subseteq [\mathbb{F}_q]^{\leq k}$ with $\lceil \tilde{\mathcal{S}}_x \rceil \supseteq \mathcal{S}_{a_{x1}, \ldots, a_{xu}}$. In particular, $|\tilde{\mathcal{S}}_x| \leq 2^{O(k^2)}(\log q)^{O(k)}$, and only sets from its downward closure may appear in its critical extension $\mathcal{S}_x$ constructed from $\tilde{\mathcal{S}}_x$ by the same straightforward algorithm as above. Since every element $S \in \tilde{\mathcal{S}}_x$ leads to at most $2^{|S|} \leq 2^k$ sets in its downward closure, the size of $\mathcal{S}_x$ is still $2^{O(k^2)}(\log q)^{O(k)}$. This implies that every individual $\mathcal{S}_x$ can be also constructed within time $2^{O(k^2)}(\log q)^{O(k)}$ and, combined with the above remark, this gives the bound $2^{O(k^2)}(q(\log q)^{O(k)} + (\log^2 N))$ on the overall performance of the recovery algorithm. It only remains to note that (3) implies $q(\log q)^{O(k)} \leq 2^{O(k^2)}(\log^2 N)$, that is, the first additive term in this bound is subsumed by the second.

Acknowledgement

I am grateful to an anonymous referee for several useful remarks.

References

[AGKS02] N. Alon, V. Guruswami, T. Kaufman, and M. Sudan. Guessing secrets efficiently via list-decoding. In *Proceedings of the* 13*th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 254–262, 2002.

[CGL01] F. Chung, R. Graham, and T. Leighton. Guessing secrets. *Electronic Journal of Combinatorics*, 8(1):R13, 2001.

[CGL02] F. Chung, R. Graham, and L. Lu. Guessing secrets with inner product questions. In *Proceedings of the* 13*th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 247–253, 2002.

[GS99] V. Guruswami and M. Sudan. Improved decoding of Reed-Solomon codes and algebraic-geometry codes. *IEEE Transactions on Information Theory*, 45(6):1757–1767, 1999.

[MS04] D. Micciancio and N. Segerlind. Using hypergraph homomorphisms to guess three secrets. Manuscript, 2004.

[Pet02] I. Peterson. Guessing secrets. *Science News*, 161(14), 2002.

[Tuz85] Z. Tuza. Critical hypergraphs and intersecting set-pair systems. *Journal of Combinatorial Theory, Ser. B*, 39:134–145, 1985.