Guessing Secrets Efficiently via List Decoding

(Extended Abstract)

Noga Alon* Venkatesan Guruswami† Tali Kaufman‡ Madhu Sudan §

Abstract

We consider the guessing secrets problem defined by Chung, Graham, and Leighton [CGL01]. This is a variant of the standard 20 questions game where the player has a set of $k > 1$ secrets from a universe of N possible secrets. The player is asked Boolean questions about the secret for each question, the player picks one of the k secrets adversarially, and answers according to this secret.

We present an explicit set of $O(\log N)$ questions together with an efficient (i.e., poly$(\log N)$ time) algorithm to solve the guessing secrets problem for the case of 2 secrets. This answers the main algorithmic question left unanswered by [CGL01]. The main techniques we use are small *ε-biased spaces* and the notion of *list decoding*.

We also establish bounds on the number of questions needed to solve the k-secrets game for $k > 2$, and discuss how list decoding can be used to get partial information about the secrets.

1 Introduction

Under the familiar "20 questions" game a player, say **B**, tries to discover the identity of some unknown secret drawn by a second player, say **A**, from a large space of N secrets. **B** is allowed to ask binary (Yes/No) questions about the secret to **A** (cf. [I52]). The assumption is that **A** answers each question truthfully according to the secret he picked. The goal of **B** is to recover the secret by asking as few questions as possible. If the N secrets are associated with $\lceil \log N \rceil$-bit strings, then clearly $\lceil \log N \rceil$ questions are both necessary and sufficient to discover the secret. (Here, and in the rest of the paper, all logarithms are in base 2).

Now, consider the following variant of the above game. Under this variant, the player **A** picks not one, but a set of $k \geq 2$ secrets. For each question asked by **B**, **A** gets to adversarially choose which one of the k secrets to use in supplying the answer, but having made the choice must answer truthfully according to the chosen secret. (When questions have binary answers, this imposes the restriction that if all secrets give the same answer on a given question, then **A** must answer accordingly, else it can give any answer to the given question.) This variant was introduced by Chung, Graham and Leighton in [CGL01], and they called the problem "Guessing Secrets". What is the best strategy for **B** in this situation, and how many questions does it take in the worst-case for **B** to "find" the secrets? In addition to being an interesting "puzzle", secret guessing problems of this type have apparently arisen recently in connection with certain Internet traffic routing applications (cf. [CGL01]). Moreover, problems of a related nature have been studied in the literature under the label of separating systems (see [KS88, Seg94, CES01] and references therein), and have been applied in different areas of computer science such as technical diagnosis, constructions of hash functions, and authenticating ownership claims. The focus of much of this line of work has been combinatorial, and our work appears to be the first to present non-trivial *algorithms* to deal with (certain kinds of) separating systems.

Specifically, in this paper we present an algorithmic solution using list decoding to a problem left open by the work of [CGL01] dealing with the case of $k = 2$ secrets. We also obtain some new results concerning adaptive and oblivious schemes for guessing $k > 2$ secrets.

The paper assumes knowledge of some standard terminology and definitions concerning error-correcting codes, all of which can be found, for example, in [vL99].

1.1 Our results

*Schools of Mathematics and Computer Science, Tel Aviv University, Tel Aviv, Israel. Email: noga@math.tau.ac.il.
Supported in part by a USA Israeli BSF grant, by a grant from the Israel Science Foundation and by the Hermann Minkowski Minerva Center for Geometry at Tel Aviv University.

†University of California at Berkeley, Computer Science Division, Berkeley, CA 94720. Email: venkat@lcs.mit.edu. Work done while at MIT.

‡School of Computer Science, Tel Aviv University, Tel Aviv, Israel. Email: kaufmant@tau.ac.il

§Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139. Email: madhu@mit.edu.
Supported in part by NSF grants CCR-9875511, CCR-9912342 and by MIT-NTT award MIT2001-01.

We defer a formal definition of what it means to solve the 2-secrets problem to the next section. We now briefly state our main results and compare it to prior work.

Chung *et al.* [CGL01] prove that for the 2-secrets problem with a universe of N secrets there exist a set of $O(\log N)$ questions which suffice. They also present an *explicit* set of $O(\log^2 N)$ questions, the answers to which allows one to "find" the two secrets. In this paper, using a connection to ε-biased spaces, we present an explicit set of $O(\log N)$ questions for the 2-secrets problem. Note that up to a constant factor, this is the fewest possible number of questions. Moreover, the results of [CGL01] did not present an efficient algorithm for **B** to recover the secrets given **A**'s answers to the questions. They do briefly report a strategy using $O(\log^3 N)$ questions which can recover the secrets in $O(\log^4 N)$ time, but the number of questions used is much more than the optimal $O(\log N)$. For our explicit set of questions, we also present an $O(\log^3 N)$ time algorithm to "recover" the secrets. This gives a $\mathrm{poly}(\log N)$ time algorithm to recover the secrets that uses an optimal $O(\log N)$ number of explicitly specified questions, and answers one of the main open questions of [CGL01]. In a sense to be made precise, our algorithm finds the maximum possible information about the 2 secrets which **A** picked.

Using a straightforward connection to $2k$-universal sets, we also explicitly describe a set of $c_k \log N$ questions to solve the k-secrets problem, for any $k \geq 2$. The constant c_k depends exponentially on k. We also prove that this exponential dependence of c_k on k is necessary, *even if* **B** is allowed to pick the questions *adaptively*, i.e., depending upon **A**'s answers to previous questions. On the other hand, all our positive results yield a non-adaptive or oblivious strategy for **B**, i.e., **B** can fix the set of questions to be asked right at the outset.

1.2 Related work

Independent of our work, Micciancio and Segerlind [MS01] recently presented a different strategy with the optimal $O(\log N)$ questions together with an $O(\log^2 N)$ time algorithm to recover the secrets. The difference between their result and ours is that our questions are *oblivious* or *non-adaptive*, where as in the strategy of [MS01], **B** picks the questions *adaptively*, depending upon **A**'s answers to previous questions. Thus the results of [MS01] are incomparable to ours. We elaborate further on adaptive and oblivious strategies and motivate why oblivious strategies are interesting in the next section.

2 Formal problem description

We first restrict ourselves to the case $k = 2$ when there are only two secrets. This is already a non-trivial case and as we shall see one where a very satisfactory solution exists to the above problem. In this case, **A** has a set $X = \{x_1, x_2\}$ of two secrets, chosen from a universe U of N possible secrets. Next, we proceed to precisely formulate the algorithmic problem that **B** wishes to (and can hope to) solve (the reader familiar with the paper [CGL01] probably already knows the formal definition, and can skip the next few paragraphs).

Note that **A** can always choose to answer according to the secret x_1, and thus **B** can never hope to learn with certainty more than one of **A**'s secret. Moreover, a moment's thought reveals that **B** cannot even hope to pin down with certainty one secret and claim that it must be one of **A**'s secrets. This is because **A** could pick three secrets $\{x_1, x_2, x_3\}$ and answer each question according to the majority vote of the answers of x_1, x_2, x_3. For such a strategy, irrespective of the number of questions **B** asks, **A** can always justify *any* subset of two of these secrets as the set X he picked.

In light of these, it turns out that the best that **B** can hope for is the following: For every set of two *disjoint* pairs of secrets $X = \{x_1, x_2\}$ and $Y = \{x_3, x_4\}$ where the x_i's are all distinct, rule out one of X or Y as the set which **A** picked. An instructive way to visualize this requirement is in terms of graphs. Let K_N denote the complete graph on the universe U of N secrets. View a pair of secrets $X = \{x_1, x_2\}$ as an edge (x_1, x_2) of K_N. A question is simply a function $F : U \to \{0, 1\}$, and the answer to it naturally induces a partition $U = F^{-1}(0) \cup F^{-1}(1)$. If **A** answers question F with a bit $b \in \{0, 1\}$, then we know that the set X which A picked must satisfy $X \cap F^{-1}(b) \neq \emptyset$, and hence **B** can "eliminate" all edges within the subgraph of K_N spanned by $F^{-1}(1 - b)$. Stated in this language, the goal of **B** is to ask a series of questions by which he can eliminate all edges except those in a set W that contains *no pair of disjoint edges*.

Now, there are only two possibilities for such a surviving set W. Either W must be a "star", i.e., a set of edges all sharing a common x_0, or W must be a "triangle", i.e., the set of three edges amongst a set $\{x_1, x_2, x_3\}$ of three secrets. In the former case, **B** can assert that x_0 must be one of **A**'s secrets. In the latter case, **B** can assert that the secret pair of **A** is one of (x_1, x_2), (x_2, x_3), or (x_3, x_1). In the sequel, when we use the phrase "find the secrets" we implicitly assume that we mean finding the underlying star or triangle as the case may be. We also use the phrase "solve the 2-secrets problem" to refer to the task of finding the underlying star or triangle.

Oblivious vs Adaptive Strategies: There are two possible strategies that one can consider for $\mathbf{B}$: *adaptive* and *oblivious* (also called *non-adaptive*). For adaptive strategies each question of $\mathbf{B}$ can depend on $\mathbf{A}$'s answers to the previous questions. For oblivious strategies $\mathbf{B}$ must fix the set of questions to be asked right at the outset and be able to infer the secrets just based on $\mathbf{A}$'s answers to those fixed questions.

Definitely, adaptive strategies seem more natural for a "20 questions" kind of set-up, since that is what a human would use when playing such a game. But, oblivious strategies have the merit of being easy to play (and being more democratic in terms of different players' abilities), since one just has to read out a fixed predetermined set of questions. Moreover, as we shall see, it is possible to do surprisingly well just using oblivious strategies. In fact, it turns out that there exist oblivious strategies that find the secrets using just $O(\log N)$ questions, which is only a constant-factor off the obvious lower bound of $\log N$ on the number of necessary questions. Moreover, the quest for oblivious strategies runs into some intriguing combinatorial questions, and leads us, quite surprisingly, into list decodable codes! Hence, we focus mainly on oblivious strategies here. (See the work of [CGL01] for some lower and upper bounds on the number of questions required by adaptive strategies.)

A probabilistic construction shows that $O(\log N)$ questions are sufficient to solve the 2-secrets problem [CGL01]. But this only proves the existence of good strategies and the questions are not explicitly specified. In the next section, we discuss how certain binary codes give explicit oblivious strategies.

3 An explicit strategy with $O(\log N)$ questions

3.1 A characterization of oblivious strategies using separating codes

An oblivious strategy for $\mathbf{B}$ is simply a sequence $\mathcal{F}$ of n Boolean functions (questions) $f_i : [N] \to \{0,1\}$, $1 \le i \le n$. We say a strategy solves the 2-secrets guessing problem if the answers to the questions f_i bring down the possible pairs of secrets to a star or a triangle.

For each secret $x \in [N]$, we denote the sequence of answers to the questions f_i on x by $C(x) = \langle f_1(x), f_2(x), \ldots, f_n(x)\rangle$. We suggestively call the mapping $C : [N] \to \{0,1\}^n$ thus defined as the *code* used by the strategy. There is clearly a one-one correspondence between oblivious strategies $\mathcal{F}$ and such codes C (defined by $f_i(x) = C(x)_i$, where $C(x)_i$ is the i'th bit of $C(x)$). Hence we will from now on refer to a strategy $\mathcal{F}$ using its associated code C.

We say that a code C is $(2,2)$-*separating* (or simply, *separating*) if for every 4-tuple of distinct secrets $a, b, c, d \in [N]$, there exists at least one value of i, $1 \le i \le n$, called the *discriminating index*, for which $C(a)_i = C(b)_i \neq C(c)_i = C(d)_i$. Note that if $\mathbf{B}$ asks questions according to a separating code C, then for every two disjoint pairs of edges (a, b) and (c, d), $\mathbf{B}$ can rule out one of them based on the answer which $\mathbf{A}$ gives on the i'th question, where i is a discriminating index for the 4-tuple (a, b, c, d)). In fact it is easy to see that the $(2,2)$-separating property of C is also necessary for the corresponding strategy to solve the 2-secrets guessing game.

This implies the following characterization for the existence of oblivious strategies for the 2-secrets guessing game.

LEMMA 3.1. *There exists a $(2,2)$-separating code $C : [N] \to \{0,1\}^n$ if and only if there exists an oblivious strategy for $\mathbf{B}$ to solve the 2-secrets guessing problem for a universe size of N that uses n questions.*

Hence the problem of finding a small set of questions to solve the 2-secrets problem reduces to the task of finding a good $(2,2)$-separating code. There is a reason why we called these objects "codes" since the following result states that any error-correcting code with a certain property is also a $(2,2)$-separating code. We will assume without loss of generality that $N = 2^m$ so that we can view each secret as an m-bit binary string. The separating code C then encodes an m-bit string into an n-bit string.

LEMMA 3.2. *Let C be an $[n,m]_2$ binary linear code with minimum distance d and maximum distance (i.e., the maximum number of coordinates where two distinct codewords differ) equal to m_1. Assume further that d, m_1 satisfy the condition $d > \frac{3m_1}{4}$. Then, C is a $(2,2)$-separating code. If the constraint of linearity is removed, then an $(n,m)_2$ binary code C is $(2,2)$-separating if $d > \frac{m_1}{2} + \frac{n}{4}$.*

The above lemma is proved in the work of Cohen, Encheva, and Schaathun [CES01]. The result for linear codes had been previously proved by Segalovich [Seg94], and, as mentioned in [CGL01], it has also been discovered, later and independently, by the first author of the present paper. For the case of general codes, note that the condition $d > m_1/2 + n/4$ implies $d > n/2$, since $m_1 \ge d$. We obtain the following strengthening of the above lemma (for the non-linear case), by noting that the weaker condition $d > n/2$ itself suffices for a code to be $(2,2)$-separating. We omit the proof, since the result is not needed in the rest of the paper.

LEMMA 3.3. *Let C be an $[n,m]_2$ binary code with minimum distance $d > n/2$. Then C is a $(2,2)$-separating code. Moreover, this bound is tight, in that there are codes of blocklength n and distance $n/2$ which are* not *$(2,2)$-separating.*

There is a big advantage in using linear codes C for **B**'s strategy, since then each question simply asks for the inner product over GF(2) of the secret with a fixed m-bit string. Thus all questions have a succinct description, which is not the case for general non-linear codes. Hence, we focus exclusively on strategies based on linear separating codes in the rest of this section and in the one following it.

3.2 Construction of good linear separating codes

DEFINITION 1. (ε-BIASED CODES) *A binary linear code of blocklength n is defined to be* ε-biased *if every non-zero codeword in C has Hamming weight between $(1/2-\varepsilon)n$ and $(1/2+\varepsilon)n$.*

Now Lemma 3.2 implies the following separation property of ε-biased codes.

COROLLARY 3.1. *If a binary linear code C is ε-biased for some $\varepsilon < 1/14$, then C is $(2,2)$-separating.*

Thus, in order to get explicit $(2,2)$-separating codes (and hence, an explicit strategy for the secret guessing game), it suffices to explicitly construct an ε-biased code for $\varepsilon < 1/14$.

A simple explicit construction of ε-biased codes can be obtained by concatenating an outer Reed-Solomon code with relative distance $(1-2\varepsilon)$ with an inner binary Hadamard code. It is easy to see that all non-zero codewords have relative Hamming weight between $(1/2-\varepsilon)$ and $1/2$, and thus this gives an ε-biased space. However, this construction encodes m bits into $O(m^2/\varepsilon^2)$ bits. Other explicit constructions of ε-biased codes of dimension m and blocklength $O(m^2/\varepsilon^2)$ are also known (cf. [AGHP92]). In fact, the explicit construction of a secret guessing strategy with $O(\log^2 N)$ questions in [CGL01] is based on one of the ε-biased codes from [AGHP92]. All these constructions suffer from the drawback of needing $\Omega(\log^2 N)$ questions, and this means they provide a strategy with $O(\log^2 N)$ questions, while we would like to achieve the optimal $O(\log N)$ questions.

But, there are also known ways to achieve ε-biased codes with blocklength $O(m/\varepsilon^{O(1)})$. For example, one can use a concatenated scheme with outer code any explicitly specified code with relative distance $(1-O(\varepsilon))$ over a constant alphabet size (that depends on ε), and inner code itself being a Reed-Solomon concatenated with Hadamard code. Specifically, one can use for the outer code the construction from [ABN+92] that achieves rate $\Omega(\varepsilon)$ and alphabet size $2^{O(1/\varepsilon)}$. It is easy to check that this gives an explicit $[O(m/\varepsilon^4), m]_2$ ε-biased code. A better choice of inner code can be used to bring down the blocklength to $O(m/\varepsilon^3)$ [ABN+92], but this is not very important to us since this will only improve the number of questions by a constant factor.

We therefore have:

LEMMA 3.4. ([ABN+92]) *For any $\varepsilon > 0$, there exists an* explicitly specified *family of constant rate binary linear ε-biased codes.*

Applying the above with any $\varepsilon < 1/14$, and using the connection to separating codes from Corollary 3.1 and the result of Lemma 3.1, we get the following:

THEOREM 3.1. *There is an explicit oblivious strategy for the 2-secrets guessing game that uses $O(\log N)$ questions where N is the size of the universe from which the secrets are drawn.*

4 An efficient algorithm to recover the secrets

The construction of an explicit strategy using $O(\log N)$ questions is not difficult, and follows rather easily once one realizes the connection to ε-biased spaces. However, a fairly basic and important point has been ignored so far in our description. We have only focused on strategies that "combinatorially" limit the possible pairs of secrets to a star or a triangle. But how can **B** figure out the star or triangle as the case may be, once he receives the answers to all the questions? One obvious method is to simply go over all pairs of secrets and check each one for consistency with the answers. By the combinatorial property of the strategy, we will be left with only a star or a triangle. The disadvantage of this approach, however, is that it requires $O(N^2)$ time. We would ideally like to have a strategy to recover the secrets that runs in poly$(\log N)$ time, since we are thinking of N as very large. Strategies with such an efficient secret recovery algorithm are called *invertible strategies* in [CGL01]. In [CGL01], the authors mention an invertible strategy for the 2-secrets guessing game, attributed to Lincoln Lu, which uses $O(\log^3 N)$ questions to find the star/triangle in $O(\log^4 N)$ time. Note, however, the number of questions is much larger than $O(\log N)$. The problem of finding an invertible strategy that uses only $O(\log N)$ questions was left unanswered in [CGL01]. Independent of our work, [MS01] answered this question by presenting an *adaptive* invertible strategy using only $O(\log N)$ questions.

In this section, we present a connection between list decoding and the 2-secrets guessing game. Using this connection, we are able to give an *oblivious* invertible strategy that uses only $O(\log N)$ questions. The time to recover the secrets (i.e. the triangle or a succinct representation of the star) is $O(\log^3 N)$. We stress that, unlike the result of [MS01], our strategy is *oblivious*, and therefore is incomparable to their result (it is not strictly better because the constants in front of the $\log N$ in the number of questions and the time needed to find the secrets are slightly worse in our construction).

This algorithmic result is our main contribution to the guessing secrets problem. Details on our construction and algorithm follow.

4.1 Connection to list decoding

LEMMA 4.1. *Suppose that C is a $[cm, m]_2$ binary linear code which is ε-biased for some constant $\varepsilon < 1/14$. Suppose further that there exists a list decoding algorithm for C that corrects up to a $(1/4 + \varepsilon/2)$ fraction of errors in time $O(T(m))$. Then, C is a $(2,2)$-separating code which gives a strategy to solve the 2-secrets guessing game for a universe size $N = 2^m$ in $O(T(\log N) + \log^3 N)$ time using $c \log N$ questions.*

Proof: Let C be a code as in the statement of the lemma and assume that $\mathbf{B}$ is using C for its strategy. Let $X = \{x_1, x_2\}$ be the set which $\mathbf{A}$ claims he picked after giving all the answers. Let the set of answers be $\mathbf{a} = (a_1, a_2, \ldots, a_n)$. Then for each i, we must have either $C(x_1)_i = a_i$ or $C(x_2)_i = a_i$ since $\mathbf{A}$ is supposed to answer each question according to one of x_1 or x_2. Now by the property of C, we have $C(x_1)_i = C(x_2)_i$ for all $i \in A$ for some set $A \subseteq [n]$ of size at least $(1/2-\varepsilon)n$. For each $i \in A$ we have $C(x_1)_i = C(x_2)_i = a_i$, and for each $i \notin A$, exactly one of $C(x_1)_i$ and $C(x_2)_i$ equals a_i. It follows that either $C(x_1)$ or $C(x_2)$ is within Hamming distance $(n - |A|)/2$ of $\mathbf{a}$; assume without loss of generality that it is $C(x_1)$. Then

$$\Delta(\mathbf{a}, C(x_1)) \leq (1/2 + \varepsilon)\frac{n}{2} = (\frac{1}{4} + \frac{\varepsilon}{2})n \ .$$

The algorithm for $\mathbf{B}$ to recover the secrets (i.e., the triangle or the star) after receiving the answer vector $\mathbf{a}$ is as follows.

1. Perform list decoding of the code C using the assumed algorithm to find the set, say S, of all $x \in \{0,1\}^m$ for which $\Delta(\mathbf{a}, C(x)) \leq (\frac{1}{4} + \frac{\varepsilon}{2})n$.
2. For each $x \in S$ returned by the list decoding algorithm in the previous step, do the following. Compute $A = \{i : C(x)_i = a_i\}$, and perform an *erasure list decoding* of the received word $\mathbf{a}$ when all of its symbols in positions in A are erased. In other words find (some representation of) the set S_x of all x' for which $C(x')_i = a_i$ for each $i \in [n] \setminus A$. If S_x is empty, then remove x from S.
3. Return the set of unordered pairs $\{(x, x') : x \in S, x' \in S_x\}$ as the final set of all possible feasible pairs.

We now argue the correctness of the algorithm. First note that any pair returned by the algorithm is a proper solution to the guessing secrets. This is because the set S_x consists of precisely those secrets that could form the other secret in a pair with x so that the resulting pair will be "consistent" with the answers $\mathbf{a}$. We next prove that any pair (x, x') which is a consistent solution to the 2-secrets problem for the answers $\mathbf{a}$, will be found by the algorithm. Appealing to the $(2,2)$-separation property of C (which is implied by Corollary 3.1 since C is ε-biased for some $\varepsilon < 1/14$), the above two facts imply that the final set of pairs *will* either be a triangle or a star.

If a pair (x, x') is consistent with $\mathbf{a}$, then we know by the initial arguments in this proof that $\min\{\Delta(\mathbf{a}, C(x)), \Delta(\mathbf{a}, C(x'))\} \leq (1/4 + \varepsilon/2)n$. Assume, without loss of generality, that $\Delta(\mathbf{a}, C(x)) \leq (1/4 + \varepsilon/2)n$. Then, x will be found as part of the set S in the first list decoding step of the above algorithm. Now for each i such that $C(x)_i \neq a_i$, we *must* have $C(x')_i = a_i$, or otherwise (x, x') would not be a consistent pair for the answers $\mathbf{a}$. Hence x' will be a solution to the erasure decoding performed in the second step. It follows that $x' \in S_x$ and that (x, x') will be output by the algorithm, as desired.

Now we move on to the runtime analysis of the algorithm. By the hypothesis of the lemma, the first list decoding step can be performed in $O(T(m))$ time. Moreover, the size of the list S returned will be bounded by an absolute constant. This follows from the Johnson bound (see, for example, [GS00, Theorem 1]), which for binary codes states that list decoding to a $\alpha/2$ fraction of errors in a code of relative distance $\delta/2$ when $\alpha < 1 - \sqrt{1-\delta}$, requires lists of size at most $(\alpha^2 - 2\alpha + \delta)^{-1}$. Applying this with $\alpha = 1/2 + \varepsilon$ and $\delta = 1 - 2\varepsilon$, gives that the list size will be at most $(\varepsilon^2 - 3\varepsilon + 1/4)^{-1}$, which is at most 24.5 for $\varepsilon < 1/14$. Hence we will have $|S| \leq 24$ and therefore the second erasure decoding step will only be performed for $O(1)$ choices of x.

For the second step we critically use the fact that C is a linear code, and hence erasure list decoding amounts to finding all solutions to a linear system. The set S_x, therefore, is either empty or the coset of a linear subspace, say W_x, of $\mathbb{F}_2^m$, and in the latter case can be represented by one solution together with a basis for W_x. Hence an $O(m^2)$ size representation of each non-empty S_x can be computed in the time needed to solve a linear system, which is certainly $O(m^3)$.

Hence the above algorithm finds either the triangle or the star of all pairs of secrets consistent with the answer vector $\mathbf{a}$ in $O(T(m) + m^3)$ time. Note that in the case when it outputs a star, the number of pairs could be quite large (as high as $(N-1)$ in case the answer vector $\mathbf{a}$ exactly matches $C(x)$ for some secret x). The algorithm exploits the fact that the non-hub vertices of the star, being the set of solutions to a linear system, can be described succinctly as the coset of a linear space. □

Remark: We stress here that the use of list decoding in the above result is critical and unique decoding does not suffice for the above application. This is because for any pair (x, x') which is consistent with $\mathbf{a}$, we are only guaranteed that one of x or x' is within Hamming distance $(1/4+\varepsilon/2)n$ from $\mathbf{a}$. Thus, we need to perform decoding to a relative radius of $(1/4+\varepsilon/2)$. Therefore, if we were to perform unique decoding, we would need a relative distance of $(1/2+\varepsilon)$, which is of course impossible for binary codes (unless they just have a constant number of codewords which is not very useful). Also, note that after the list decoding algorithm finds the set S of codewords close to $\mathbf{a}$, the application gives a *natural post-processing routine to prune the list* and actually zero down the possibilities to the true solutions.

4.2 The final result using specific list decodable codes

We now prove that explicit codes with the property needed in Lemma 4.1 exist, and thus conclude our main algorithmic result about the 2-secrets guessing game. The following result is quite standard and can be proved easily using known techniques on concatenated codes. The only new element is the requirement of an ε-biased code, but as we shall see this necessitates no significant change in the proof technique.

LEMMA 4.2. *For every positive constant $\alpha < 1/2$, the following holds. For all small enough $\varepsilon > 0$, there exists an explicit asymptotically good family of binary linear ε-biased codes which can be list decoded up to an α fraction of errors in $O(n^2(\frac{\log n}{\varepsilon})^{O(1)})$ time.*

Proof: (Sketch) We only sketch the proof since it is by now quite routine. Given $\alpha < 1/2$, we pick $\varepsilon = O((1/2-\alpha)^2)$. The code construction will be the concatenation of an outer Reed-Solomon code C_{RS} of rate smaller than ε with inner code C_{in} being an explicitly specified $\varepsilon/2$-biased binary linear code (such a code exists by Lemma 3.4). It is clear that the resulting concatenated code, say C, has minimum relative distance at least $(1-\varepsilon)(1/2-\varepsilon/2) > 1/2-\varepsilon$, and maximum relative distance at most $1\cdot(1/2+\varepsilon/2) < 1/2+\varepsilon$. Hence C is definitely an ε-biased code.

Let the Reed-Solomon code be defined over $\mathrm{GF}(2^\ell)$ and have blocklength $n_0 = 2^\ell$. Let the blocklength of C_{in} be n_1. The blocklength of C is then $N = n_0 n_1$. To list decode a received word $\mathbf{r} \in \mathbb{F}_2^n$, we first divide $\mathbf{r}$ into n_0 blocks $r_1, r_2, \ldots, r_{n_0}$ corresponding to the n_0 inner encodings, where each $r_i \in \mathbb{F}_2^{n_1}$. Each of the r_i's is decoded by brute-force to produce a list L_i of all $\zeta \in \mathrm{GF}(2^\ell)$ for which $\Delta(C_{\mathrm{in}}(\zeta), \mathbf{r}_i) \le \beta n_1$, for some β where $\alpha < \beta < 1/2$. Since $\delta(C_{\mathrm{in}}) \ge 1/2-\varepsilon$ and $\alpha = 1/2 - \Omega(\sqrt{\varepsilon})$, it follows using Johnson bounds for list decoding (see, for example, [GS00, Theorem 1]), that for each i, $|L_i| = O(1/\varepsilon)$. Now if x is such that $\Delta(C(x), \mathbf{r}) \le \alpha N$, then by an averaging argument for at least an α/β fraction of i, $1 \le i \le n_0$, we must have $C_{\mathrm{RS}}(x)_i \in L_i$. Therefore, to finish the list decoding, it suffices to list decode the outer Reed-Solomon code to find all x for which $C_{\mathrm{RS}}(x)$ has an element from L_i at the i'th position for at least $\alpha n_0/\beta$ values of i. If the rate of C_{RS} is at most $O(\alpha^2\varepsilon/\beta^2)$, this can be accomplished in $O((n_0/\varepsilon)^2 \log^2 n_0)$ time using the Reed-Solomon list decoding algorithms from [GS99]. (For easy reference we state the precise form of the result used from [GS99] at the end of the proof.) This completes the proof of the lemma. □

THEOREM 4.1. (IMPLICIT IN [GS99]) *Let C be an $[n, k+1, n-k]_q$ Reed-Solomon code defined by the evaluation of degree k polynomials over $\mathrm{GF}(q)$ at n distinct elements of $\mathrm{GF}(q)$. Then, given lists $L_i \subseteq \mathrm{GF}(q)$ with each $|L_i| \le \ell$, there are at most $O(\sqrt{n\ell/k})$ codewords of C which agree with an element of L_i for at least αn values of i, provided $\alpha > \Omega(\sqrt{k\ell/n})$. Moreover, the list of all such codewords can be found in $O(n^2\ell^2\log^2 q)$ time.*[1]

Applying the result of Lemma 4.2 with $\alpha = 1/4 + 1/28 = 2/7$ and any $\varepsilon < 1/14$, gives an explicit construction of the codes which were needed in Lemma 4.1, with a quadratic list decoding algorithm. The $O(\log^3 N)$ time required to perform the "simple, clean-up" erasure decodings in Lemma 4.1 therefore dominates the overall time to recover the triangle or star of secrets. This gives our main result of this section, which achieves the optimal (up to constant factors) number of questions together with a $\mathrm{poly}(\log N)$ algorithm to recover the secrets.

[1]The claim about the runtime follows from fast implementations of the algorithm of [GS99] from [NH99, RR00].

THEOREM 4.2. [Main Result on Guessing Secrets, $k = 2$] *For the guessing secrets game between players* **B** *and* **A** *with 2 secrets picked out of a universe of size N, there exists an explicit oblivious strategy for* **B** *to discover the underlying star or triangle of possible pairs of secrets, that requires $O(\log^3 N)$ time and uses $O(\log N)$ questions.*

5 The case of more than two secrets

One can also consider the situation when the player **A** has $k > 2$ secrets. In this case, stated in the same graph-theoretic language that we used to describe the 2-secrets problem, the goal of **B** would be to find a k-uniform hypergraph H with vertex set being the N secrets with the property that every two hyperedges of H intersect. Let us call such a hypergraph an *intersecting hypergraph.*

5.1 Explicit oblivious strategies with $O(\log N)$ questions

Unlike the case of graphs, where there were only two classes of such graphs, namely a triangle or a star, the situation even for 3-regular hypergraphs is much more complicated. Nevertheless, there exist explicit strategies which will allow **B** to "combinatorially" reduce the possibilities to an intersecting hypergraph, even though we do not know any method for **B** to actually find some representation of this hypergraph short of trying out all k element subsets of secrets and pruning out all "inconsistent" ones. This follows from a connection of the k-secrets guessing problem to the study of $2k$-universal families of binary strings. The latter problem concerns finding a subset $S \subset \{0,1\}^N$ of as small size as possible with the property that for every subset of $2k$ indices $i_1, i_2, \ldots, i_{2k}$ and every $(a_1, a_2, \ldots, a_{2k}) \in \{0,1\}^{2k}$, there exists a string $x \in S$ such that $x_{i_j} = a_j$ for each $j = 1, 2, \ldots, 2k$. Explicit constructions of such universal families of very small size, namely at most $c_k \log N$, are known [NN93, ABN$^+$92], where c_k is a constant that depends exponentially on k.

We claim this implies the existence of explicit oblivious strategies using $c_k \log N$ questions for the k-secrets guessing game. Indeed let $\{y_1, y_2, \ldots, y_n\}$ be a $2k$-universal family of N-bit strings for some $n \le c_k \log N$. For $1 \le i \le n$, define the function $f_i : [N] \to \{0,1\}$ as follows: for each $x \in [N]$, $f_i(x)$ is simply the x'th bit of the string y_i. That is, the string y_i gives the truth table of the function f_i. Clearly if the y_i's are explicitly specified then so are the functions f_i. We claim the sequence of questions $f_1, f_2, \ldots, f_n$ is a valid oblivious strategy for the k-secrets guessing game. This is because, for every pair of disjoint sets of k secrets each, say $S_1 = \{i_1, i_2, \ldots, i_k\} \subset [N]$ and $S_2 = \{i_{k+1}, \ldots, i_{2k}\} \subset [N]$, by the $2k$-universality property there exists some i for which $f_i(x) = 0$ for each $x \in S_1$ and $f_i(z) = 1$ for each $z \in S_2$. This implies that the answer to question number i rules out one of the sets S_1 or S_2 as being a possible set of k secrets consistent with all answers to the questions $f_1, f_2, \ldots, f_n$. This is exactly what we wanted to show, and the k-sets of secrets consistent with any answer to the questions $f_1, f_2, \ldots, f_n$ therefore form an intersecting k-uniform hypergraph.

The best known construction of $2k$-universal families, due to [ABN$^+$92], achieves $c_k = ck2^{6k}$ where c is an absolute constant. We therefore have:

THEOREM 5.1. *For the k-secrets guessing game over a universe of size N, there exists an explicit oblivious strategy for* **B** *that uses at most $ck2^{6k} \log N$ questions, where c is an absolute constant independent of k.*

5.2 An efficient "partial solution" for the k-secrets game

For the case of $k > 2$ secrets, The question of whether there exists a strategy together with an efficient algorithm to actually find a representation of the intersecting hypergraph is wide open, and appears to be rather difficult to solve. We instead aim for the weaker goal of finding a *small* "core" of secrets such that any k-set which **A** might pick must intersect the core in at least one secret. This at least gives useful partial information about the set of secrets which **A** could have picked.

This version of the problem is quite easily solved if we could ask not binary questions, but questions over a larger alphabet $[q] = \{1, 2, \ldots, q\}$. That is, each of the n questions that **B** asks is now a function $F_i : [N] \to [q]$, $1 \le i \le n$. For any set of k secrets which **A** might pick, the sequence of answers $\mathbf{a} \in [q]^n$ must agree with the correct answers to one of the secrets for at least n/k values of i. If q is a prime power bigger than k, there are known explicit constructions of q-ary linear codes, say C, with N codewords and blocklength $O(\log N)$ which are efficiently list decodable from a $(1 - 1/k)$ fraction of errors [GS00]. Basing the questions F_i on the n positions of the code (as in the earlier binary case), the answer vector $\mathbf{a}$ of **A** will differ from at least one secret in **A**'s set in at most a $(1 - 1/k)$ fraction of positions. The list decoding algorithm, when run on input $\mathbf{a}$, will output a small list that includes that secret.

When **B** is only allowed binary questions, we can still give such a "core finding" strategy as follows. Pick q to be a prime power larger than k^2, and C to be an explicit q-ary linear code that is list decodable from a $(1-1/k^2)$ fraction of errors [GS00]. As above, we first encode each secret by C to get a string of length $n = O(\log N)$ over $[q]$. We then encode each element of $[q]$ further using $2k$-universal family $\mathcal{F}$ of strings in $\{0,1\}^q$. That is, we encode $i \in [q]$, by the string comprising of the i'th entry from the set of strings in $\mathcal{F}$. In other words, we *concatenate* C with the $2k$-universal family $\mathcal{F}$ to get a binary code C'. Player $\mathbf{B}$ now asks $\mathbf{A}$ for the bits of the encoding of the secret as per the concatenated code C'.

Using the $2k$-universal property of $\mathcal{F}$, $\mathbf{B}$ can recover an intersecting k-hypergraph H_i on $[q]$ for the value of the i'th symbol of the encoding of the k secrets by C. $\mathbf{B}$ can do this by a brute-force search over all k-element subsets of q, since q, k are constants, this only takes constant time for each i. $\mathbf{B}$ then picks one of the hyperedges E_i from H_i arbitrarily, and then picks an element $a_i \in [q]$ from it at random.

Let $S = \{x_1, x_2, \ldots, x_k\}$ be the set of k-secrets that $\mathbf{A}$ picked. Note that E_i must intersect the set, say $S_i = \{C(x_1)_i, \ldots, C(x_k)_i\}$ consisting of the i'th symbols of the encoding of the secrets in S by C. Therefore, we will have $a_i \in S_i$ for an expected $1/k$ fraction of i's. An averaging argument that implies that there must exist a j, $1 \le j \le k$, for which $a_i = C(x_j)_i$ for at least a $1/k^2$ fraction of i's. Therefore, the assumed list decoding algorithm for C on input a will find a small list that includes the secret x_j.

This gives the following result, whose detailed proof will appear in the full version of the paper.

THEOREM 5.2. *For the k-secrets guessing game with a universe of N secrets, there exists an explicit oblivious strategy for $\mathbf{B}$ that uses $O(\log N)$ questions. Moreover, there is an efficient $\mathrm{poly}(\log N)$ time algorithm for $\mathbf{B}$ to find a small core of $\mathrm{poly}(k)$ secrets such that the k-set picked by $\mathbf{A}$ must contain at least one secret from the core.*

5.3 Lower bounds for the k-secrets game

As mentioned above, there is an oblivious algorithm for solving the k secrets problem by asking $c_k \log N$ questions. An easy probabilistic argument shows that the smallest possible c_k is $O(k2^{2k})$. This also follows from the result of [KS73] that shows (non-explicitly) that there is a $2k$-universal family of N-bit strings consisting of $O(k2^{2k} \log N)$ strings. The next result shows that no oblivious algorithm can be much better.

PROPOSITION 5.3. *Any oblivious strategy for solving the k-secrets problem requires at least $\Omega(2^{2k} \log N)$ questions.*

Proof: Suppose there is an oblivious strategy consisting of n questions. Then there are n functions $f_i : [N] \to \{0,1\}$ such that for every pair of disjoint sets of k secrets each, say $S_1 = \{i_1, i_2, \ldots, i_k\} \subset [N]$ and $S_2 = \{i_{k+1}, \ldots, i_{2k}\} \subset [N]$, there exists some i for which $f_i(x) = \varepsilon$ for each $x \in S_1$ and $f_i(z) = 1-\varepsilon$ for each $z \in S_2$, where $\varepsilon \in \{0,1\}$. Our objective is to show that n is large. To do so, pick, randomly and independently, two disjoint subsets S'_1 and S'_2 of $[N]$, where $|S'_1| = |S'_2| = k-1$. We say that f_i *separates* S'_1 and S'_2 if there is an $\varepsilon \in \{0,1\}$ such that $f_i(x) = \varepsilon$ for each $x \in S'_1$ and $f_i(z) = 1-\varepsilon$ for each $z \in S'_2$. A simple computation shows that for each fixed i, the probability that f_i separates the two random sets S'_1 and S'_2 is smaller than 2^{1-2k}. Therefore, by linearity of expectation, there are some two sets S'_1 and S'_2 as above which are separated by at most $n2^{1-2k}$ functions f_i. Fix such S'_1, S'_2 and let $I \subset \{1, 2, \ldots, n\}$ be the set of all functions separating them. We claim that for each two distinct $p, q \in [N] \setminus (S'_1 \cup S'_2)$, the two vectors $(f_i(p) : i \in I)$ and $(f_i(q) : i \in I)$ must differ. Indeed, otherwise, none of the functions f_i will separate $S'_1 \cup \{p\}$ and $S'_2 \cup \{q\}$, contradicting the fact that they form an oblivious strategy for solving the k-secrets problem. It thus follows that

$$N - (2k-2) \le 2^{n2^{1-2k}},$$

implying that

$$n \ge 2^{2k-1} \log(N - 2k + 2),$$

and completing the proof. □

We next show that even adaptive algorithms cannot solve the k-secrets problem in less than $\Omega(\frac{2^{2k}}{\sqrt{k}} \log N)$ questions.

PROPOSITION 5.4. *Any adaptive algorithm for solving the k-secrets problem requires at least $a_k \log N - a_k \log a_k$ questions, where*

$$a_k = 2k - 2 + \frac{1}{2}\binom{2k-2}{k-1}.$$

Proof: Let a_k be as in the statement of the proposition. We claim that for every $k \ge 2$ there is an intersecting k-uniform hypergraph H_k on a_k vertices which is a maximal intersecting hypergraph. That is, one cannot add to it any additional edge (of size k) and keep it intersecting. Such a hypergraph is described in [EL75]. Here is the construction; the set of vertices consists of two disjoint sets, X_k of size $2k-2$, and Y_k of size $\frac{1}{2}\binom{2k-2}{k-1}$, whose elements are indexed by the partitions of X_k into two disjoint parts of equal size. The edges are all k-subsets of X_k, together with $\binom{2k-2}{k-1}$ additional edges: for each partition of X_k into two equal parts

A and B, the sets $A \cup \{y\}$ and $B \cup \{y\}$ are edges of H_k, where y is the element of Y corresponding to the partition (A, B).

There are at least $\binom{N}{a_k}$ copies of H_k on subsets of the set of vertices $[N]$. For every question $F : [N] \mapsto \{0,1\}$ of **B**, and for each copy H of H_k, the player **A** can always save H (namely, ensure that each edge of it can still serve as a valid set of k secrets) by an appropriate answer. Indeed, as H is intersecting, there cannot be an edge of H in $F^{-1}(0)$ and another one in $F^{-1}(1)$, and hence either each edge of H intersects $F^{-1}(0)$ and then **A** can answer 0 and save H, or each edge intersects $F^{-1}(1)$, and then answering 1 will save H. This implies that the player **A** can save, in each question, at least half of the copies of H_k left after the previous questions. As **B** cannot finish the guessing game as long as more than one copy of H_k is still valid (because the maximality of H_k ensures that the union of every two copies is not intersecting) it follows that the number of questions required is at least

$$\log\left(\binom{N}{a_k}\right) \geq a_k \log N - a_k \log a_k,$$

as needed. $\square$

As mentioned in [CGL01], the question of guessing secrets may also be of interest when the questions have more than two possible answers. We conclude the paper by pointing out that several aspects of this variant have been studied by various researchers under the name *parent identifying codes*. For more details, see, e.g., [CFN94], [HLLT98], [BCE+01], [AFS01].

References

[ABN+92] Noga Alon, Jehoshua Bruck, Joseph Naor, Moni Naor, and Ronny Roth. Construction of asymptotically good low-rate error-correcting codes through pseudo-random graphs. *IEEE Transactions on Information Theory*, 38:509–516, 1992.

[AFS01] Noga Alon, Eldar Fischer and Mario Szegedy. Parent-identifying codes. *J. Combinatorial Theory, Series A*, 95:349–359, 2001.

[AGHP92] Noga Alon, Oded Goldreich, Johan Håstad, and Réne Peralta. Simple constructions of almost k-wise independent random variables. *Random Structures and Algorithms*, 3:289–304, 1992.

[BCE+01] Alexander Barg, Gerard Cohen, Sylvia Encheva, Gregory Kabatiansky and Gillés Zémor. A hypergraph approach to the identifying parent property: the case of multiple parents, *to appear.*

[CES01] Gérard D. Cohen, Sylvia B. Encheva, and Hans G. Schaathun. On separating codes. Technical Report, ENST, Paris, 2001.

[CFN94] Benny Chor, Amos Fiat and Moni Naor. Tracing traitors. *Proceedings of Crypto'94* LNCS 839 (1994), pp. 257–270.

[CGL01] Fan Chung, Ron Graham, and Tom Leighton. Guessing secrets. *The Electronic Journal of Combinatorics*, 8(1):R13, 2001.

[EL75] Paul Erdős and Láslzo Lovász, Problems and results on 3-chromatic hypergraphs and some related questions, *Infinite and Finite Sets (Colloq., Keszthely, 1973; dedicated to P. Erdős on his 60th Birthday)*, vol. II, pp. 609-627. *Colloq. Math. Soc. Janos Bolyai*, vol 10, North-Holland, Amsterdam, 1975.

[GS99] Venkatesan Guruswami and Madhu Sudan. Improved Decoding of Reed-Solomon and Algebraic-geometric codes. *IEEE Transactions on Information Theory*, 45 (1999), pp. 1757-1767.

[GS00] Venkatesan Guruswami and Madhu Sudan. List decoding algorithms for certain concatenated codes. *Proc. of the 32nd Annual ACM Symposium on Theory of Computing*, May 2000, pp. 181-190.

[HLLT98] H.D. Hollmann, J.H. van Lint, J.-P. Linnartz and L.M. Tolhuizen. On codes with the identifiable parent property. *J. Combinatorial Theory*, Series A, **82** (1998) pp. 121–133.

[I52] *I've Got a Secret.* A classic '50's and '60's television gameshow. See `http://www.timvp.com/ivegotse.html`.

[KS88] János Körner and Gábor Simonyi. Separating partition systems and locally different sequences. *SIAM J. Discrete Math.*, 1 (1988), pp. 355-359.

[KS73] Daniel J. Kleitman and Joel Spencer. Families of k-independent sets, *Discrete Mathematics*, 6 (1973), pp. 255-262.

[MS01] Daniele Micciancio and Nathan Segerlind. Using prefixes to efficiently guess two secrets. *Manuscript*, July 2001.

[NN93] Joseph Naor and Moni Naor. Small-bias probability spaces: Efficient constructions and applications. *SIAM Journal on Computing*, 22(4):838–856, 1993.

[NH99] Rasmus R. Nielsen and Tom Hϕholdt. Decoding Reed-Solomon codes beyond half the minimum distance. *Coding Theory, Cryptography and Related areas, (eds. Buchmann, Hoeholdt, Stichtenoth and H. tapia-Recillas)*, pages 221–236, 1999.

[RR00] Ronny Roth and Gitit Ruckenstein. Efficient decoding of Reed-Solomon codes beyond half the minimum distance. *IEEE Transactions on Information Theory*, 46(1):246–257, January 2000.

[Seg94] Yu L. Segalovich. Separating systems. *Problems of Information Transmission*, 30(2):105–123, 1994.

[vL99] J. H. van Lint. *Introduction to Coding Theory.* Graduate Texts in Mathematics **86**, (Third Edition) Springer-Verlag, Berlin, 1999.