Rigorous Time/Space Tradeoffs for Inverting Functions[*]

Amos Fiat[†] Moni Naor [‡]

Abstract

We provide rigorous time/space tradeoffs for inverting any function. Given a function f, we give a time/space tradeoff of $TS^2 = N^3 q(f)$, where $q(f)$ is the probability that two random elements (taken with replacement) are mapped to the same image under f. We also give a more general tradeoff, $TS^3 = N^3$, that can invert *any* function at *any* point.

Key words. cryptography, cryptanalysis, one-way functions, randomized algorithms, random graphs, hashing data encryption standard

AMS subject classifications. 68M10, 68Q20, 68Q22, 68R05, 68R10

---

[*] A preliminary version of this paper appeared in the Proc. 23rd ACM Symp. on Theory of Computing, 1991, pp. 534–541.

[†] Department of Computer Science, Tel-Aviv University, Tel-Aviv, Israel. e-mail: fiat@math.tau.ac.il. Part of this work was done while the author was visiting the International Computer Science Institute and the IBM Almaden Research Center. Research supported by a grant from the Israel Science Foundation administered by the Israel Academy of Sciences. E-mail: fiat@math.tau.ac.il.

[‡] Incumbent of the Morris and Rose Goldman Career Development Chair, Dept. of Applied Mathematics and Computer Science, Weizmann Institute of Science, Rehovot 76100, Israel. Most of this work performed while at the IBM Almaden Research Center. Supported by a grant from the Israel Science Foundation administered by the Israel Academy of Sciences. E-mail: naor@wisdom.weizmann.ac.il.

1 Introduction

Time/space tradeoffs occur in many searching tasks. Typical examples are a $TS^2 = \tilde{O}(2^n)$ time/space tradeoff for knapsack-like problems [15, 17, 10], algorithms for solving the discrete log problem [16], etc. In this paper we investigate time/space tradeoffs for inverting functions.

Hellman [12] was the first to study this problem. He suggests a general time/space tradeoff to invert one-way functions. Let the domain be $D = \{1, \ldots, N\}$ and let $f : D \mapsto D$ be the function to be inverted. Suppose that f is given in a black box manner, i.e. on input x the value $f(x)$ appears as output. To invert $f(x)$, simple exhaustive search requires time $\tilde{O}(N)$ and constant space. (Throughout, we ignore low order polylogarithmic factors in time/space requirements and use the $\tilde{O}$ notation. We say that $f \in \tilde{O}(g)$ if there exists some c such that $f \in O(g \cdot \log^c(g))$ An extreme alternative is to pre-compute a table with N entries whose ith location contains the pre-image of i. Constructing such a table requires preprocessing time and space $\tilde{O}(N)$ but later requires only $O(1)$ time per function inversion.

Hellman's motivation is a chosen cleartext attack on block ciphers. Given a block encryption algorithm, $E_k(x)$, consider the function f that maps the encryption key k to the encryption of a fixed cleartext block B under k, $f(k) = E_k(B)$. Inverting such a function is equivalent to breaking the encryption scheme under a fixed cleartext attack. An eavesdropper that wants to listen to many communications may find that $\tilde{O}(N)$ time per transmission is too expensive or causes an unacceptable delay. Because of the differing costs of time and memory, a memory of size $O(N)$ might be absolutely too large, while a preprocessing time of $O(N)$ may be feasible. This motivates searching for time/space tradeoffs that require less than $\tilde{O}(N)$ time per function inversion and also require less than $\tilde{O}(N)$ space.

Hellman assumes in his analysis that the function f that is to be inverted is a random function. He also postulates that functions obtained by minor modifications to f (permuting the output bits) can be modeled as independently chosen random functions. Under these presuppositions, Hellman's scheme [12] requires $\tilde{O}(N)$ time for preprocessing and time/space T/S per function inversion, where T and S satisfy $TS^2 = \tilde{O}(N^2)$.

We find it difficult to give an exact characterization of the assumptions actually required for Hellman's scheme to work. We cannot characterize the set of functions for which the Hellman time/space tradeoff is applicable. While the construction may work for specific functions such as the DES key to ciphertext, it seems to be very difficult to prove so. We note that taking f to be polynomial time indistinguishable from a random function is insufficient.

It is possible to give functions for which Hellman's time/space construction will fail. In fact, it is relatively simple for a cryptanalyst to design a cryptographic scheme that causes the Hellman scheme to fail completely, paying for this through the inconvenience that the encryption scheme is not invertible for some negligible fraction of the keys. Our goal in this paper is to give a rigorous time/space tradeoff construction, that works for any function.

In this paper we give an algorithm for constructing time/space tradeoffs, denoted A. A consists of a preprocessing phase and an online inversion phase.

Our main result in this paper can be summarized in the following theorem:

Theorem 1.1 *For any function* $f : D \mapsto D$, $|D| = N$, *and for any choice of values* (S, T) *that obey* $TS^3 = N^3$, *the preprocessing phase of* A *requires time* $\tilde{O}(N)$ *and space* $\tilde{O}(S)$, *producing a time space data structure of space* $\tilde{O}(S)$.

With probability $1 - 1/N$, *over the coin tosses of the preprocessing phase, the data structure produced by the preprocessing phase allows the online inversion phase to invert* f *at any point* y *in the range of* f *in time* $\tilde{O}(T)$.

An interesting point in this tradeoff is $T = S = N^{3/4}$. We also show a that this time/space tradeoff can be improved for some functions where we can bound the average number of domain elements that map to the same range value. In particular, for functions where the maximum number of domain elements that map to any range value is polylogarithmic in N, we can give a tradeoff of $TS^2 = N^2$, an interesting point in this tradeoff is $T = S = N^{2/3}$.

The principle ideas used in this paper are as follows:

1. We build a table of high indegree points in the graph induced by f; this table is used to construct functions that bypass these points. This allows a better cover of the domain by random chains.
2. We use k-wise independent functions to substitute for the random functions assumed by the Hellman tradeoff. An appropriate choice of parameters allows us to do so without harm.
3. We choose a specific order of events during the inversion process, and allow dependencies between different k-wise independent functions. These changes are harmless and allow us to compute these k-wise independent functions in nearly constant *amortized* time.

Section 2 of this paper gives an overview of the Hellman tradeoff, Section 3 introduces the rigorous tradeoff, Section 4 describes our scheme and its analysis, while restricting the function f to be not "trivial to invert in space S". (Intuitively, if a function f is trivial to invert in space S it implies that f is generally not useful for cryptographic applications in a scenario where the cryptanalyst has space S.) For completeness, we describe how to modify our construction to deal with trivial to invert functions in in Section 5, concluding the proof of Theorem 1.1, we also consider various extensions to the scheme. The last section gives some open problems associated with this work.

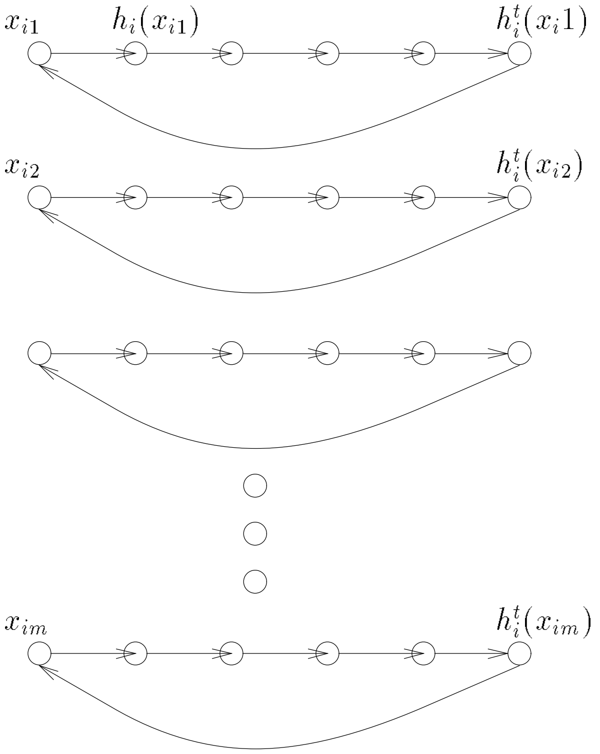

Figure 1: The chains for a specific function h_i

2 The Hellman Construction

We now review the Hellman construction which is the starting point of our scheme. Suppose first that the function $f : D \mapsto D$, $D = \{1, \ldots, N\}$ is a permutation. In this case there is an elegant time/space tradeoff of $T \cdot S = \tilde{O}(N)$: consider the graph induced by f, where vertices are points in the domain and directed edges describe f's operation on the point. Since f is a permutation, this graph splits into disjoint cycles. If a cycle is less than T in length, do nothing. If the cycle is longer then T, store a set of "shortcut elements" along the cycle so that no two are further apart than T vertices. Each one of these shortcut elements points to it's closest predecessor. I.e. if the shortcut elements are T apart, then z points to $f^{-T}(z)$. Now given $y = f(x)$, it can be inverted by following the directed graph; If a pointer element is found, then follow the pointer and continue as before; Stop when y is reached again. The immediate predecessor to y is it's inverse.

Unfortunately, this does not work for functions f that are not permutations and the time/space tradeoff is worse. Hellman's breakthrough was to consider many related functions, such that if any one of them can be inverted, then so can f. Hellman dealt with a random function $f : D \mapsto D$ and postulated that functions obtained by minor modifications to f (permuting the output bits) can be modeled as independently chosen random functions. We describe Hellman's functions differently, as the composition of f with a random function.

The Hellman tradeoff curve allows one to choose space S and an inversion time T subject to $TS^2 = N^2$, $|D| = N$. The values S and T are a function of three scheme parameters ℓ, m and t where $S = \ell m$, $T = \ell t$, and subject to the constraints $t^2 m \leq N$ and $mt\ell = N$.

For concreteness we'll assume $\ell = m = t = N^{1/3}$, giving the most interesting tradeoff point $T = N^{2/3}$, $S = N^{2/3}$. Choose $\ell = N^{1/3}$ different functions $h_i : D \mapsto D$, where $h_i(x)$ has the form $g_i(f(x))$ and g_i is a random function. For every h_i function, $1 \leq i \leq \ell$, Hellman suggests taking $m = N^{1/3}$ randomly selected points $x_{i1}, x_{i2}, \ldots, x_{i,m}$ and storing a table with m entries. Let $t = N^{1/3}$, every entry is a pair: x_{ij} and the tth iterate of h_i on x_{ij}, denoted $h_i^t(x_{ij})$ (see figure 1). The table is sorted (or hashed) by $h_i^t(x_{ij})$ so that table lookup of x_{ij}, given $h_i^t(x_{ij})$, takes no more than $O(\log N)$ time. This table requires space $\tilde{O}(m) = \tilde{O}(N^{1/3})$ per h_i function, and thus space $\tilde{O}(m \cdot \ell) = \tilde{O}(N^{2/3})$ for all $\ell = N^{1/3}$ tables.

To invert f at a point y, Hellman's tradeoff repeats the following process for all $1 \leq i \leq \ell$: Compute the values $h_i(y), h_i^2(y), \ldots$, if you find a value $h_i^j(y)$ in the table associated with function h_i, then follow the link (the appropriate table entry) and continue applying h_i. If during this process we find a value z such that $h_i(z) = g_i(y)$ then check if $f(z) = y$. It can be shown that this will indeed be the case with constant probability under the appropriate idealized assumptions.

If f and the functions g_i are all independent random functions, $O(1)$ time computable, then f can be inverted at a random point with $\Omega(1)$ probability in time $\tilde{O}(\ell \cdot t) = \tilde{O}(N^{2/3})$. Under the same set of assumptions, f can be inverted at a random point with constant probability and time/space requirements obeying $TS^2 = \tilde{O}(N^2)$.

The difficulty with Hellman's construction is that it requires completely random and independent g_i's, but it is not clear of course how to come up with such functions.

3 Outline of Results and Methodology

We do not want to assume that f has any particular structure; specifically, we do not want to assume that f is random. An adversary may devise f so as to cause the Hellman's basic cryptanalytic time/space tradeoff to fail. There exist cryptographically sound functions f, e.g., polynomial time indistinguishable from a truly random function, for which Hellman's time/space tradeoff fails. This occurs since the entire tradeoff scheme deals with *super-polynomial* values.

Consider a function f with the property that some set of $N^{1-\epsilon}$ domain elements map to the same image, $\epsilon < 1/3$. Such a function may be polynomial time indistinguishable from a random function. Even if one assumes that random $O(1)$ time computable g_i functions are available, the Hellman time/space scheme fails with overwhelming probability. A cryptographer might devise the cryptographic scheme so that only $N - N^{1-\epsilon}$ of the keys induce a permutation, the other keys map all cleartext values to zero. The Hellman attack on this scheme will fail.

Another assumption we wish to remove is that the random functions g_i are available. If every g_i function is described by a long table of truly random entries then the time/space tradeoff above becomes meaningless. We will use k-wise independence to limit the storage requirements of our g_i functions. We will show that an appropriate choice of k will ensure that the cryptanalytic scheme works. However, for any construction of k-wise independent functions, computing the function requires more than $\tilde{O}(1)$ time. We then show how to modify the scheme so that the FFT algorithm can be used to reduce the *amortized* computation cost of our k-wise independent g_i.

As f can be determined by an adversary, we cannot assume anything about the structure of f. Let $I(y)$ denote the number of pre-images for y under f, $I(y) = |\{x \in D | f(x) = y\}|$. We call $I(y)$ the *indegree* of y.

Images y with many pre-images under f imply images $g_i(y)$ with many pre-images under h_i.

Consider the sequence $I(0), I(1), \ldots, I(N)$, we are interested in the probability that two randomly chosen elements from the domain D have the same image. We denote this probability by

$$q(f) = \frac{\sum_{i=0}^{N} I(i)^2}{N^2}.$$

Under the assumptions of random g_i functions, and given $q(f)$, we can modify the parameters of the Hellman time/space tradeoff of Section 2 to obtain a generalized tradeoff. If the g_i's are random functions, then $q(h_i) = q(g_i \circ f) \leq q(f) + 1/N$ with high probability. Any point on the time/space tradeoff curve

$$T \cdot S^2 = N^3 \cdot q(f) \tag{1}$$

can be obtained[1].

We now sketch the argument as to why the above tradeoff is possible. Let ℓ denote the number of functions, m the number of chains per function, and t the length of each chain. Set m and t such that $m \cdot t^2 \cdot q(f) \leq 1$, we can argue that for all i, the union of all m chains

---

[1] This tradeoff is not the best possible at the endpoints $T = N$ or $S = N$ in which case we can do better.

contains $\Theta(m \cdot t)$ elements with high probability. Setting $\ell \cdot t \cdot m \geq N$ implies that the union of all $m \cdot \ell$ chains contains a constant fraction of all images, with high probability.

In Section 4 we show that we can obtain the tradeoff of (1) without recourse to unprovable assumptions on the functions g_i. Instead, we give explicit provable constructions.

We give an alternative tradeoff, which works for any function f, of arbitrary $q(f)$, such that any point on the time/space tradeoff curve

$$T \cdot S^3 = N^3 \tag{2}$$

can be obtained.

The tradeoff of (2) is better than the tradeoff of (1) whenever $S > 1/q(f)$. This tradeoff makes use of $O(S)$ memory so as to reduce the "effective probability of collision" to $1/S$ instead of $1/q(f)$. Thus, if we modify tradeoff (1) by replacing $q(f)$ with the "effective probability of collision", tradeoff (2) becomes a special case. As mentioned above, we will get a low "effective probability of collision" by designing the h_i functions so as to avoid high indegree nodes under f. We give a construction to a single data structure that obeys the above tradeoffs with respect to space and time, but each element can be inverted with constant probability. By repeating the construction we get that *all* the element can be inverted with high probability.

4 The Construction

4.1 Preliminaries

The function to be inverted, f, maps the domain D of size N onto itself. We may consider $D = \{1, \ldots, N\}$.

Definition 4.1 *A function $f : D \mapsto D$ is said to be trivial to invert in space S if storing the S highest indegree images in a table allows one to invert $f(x)$ by table lookup, for all $x \in D' \subset D$ such that $|D'| \geq N/2$.*

By definition, all functions are trivial to invert in space N. Throughout this section, we always assume that f is *not* trivial to invert in space S. If f is trivial to invert, table lookup takes care of most of the cases. We can handle trivial to invert functions as well. (See Section 5)

In the construction we use several tables in which pairs of the form (x, y) are stored. Given a value y we need to search whether a pair with y is stored in the table. This can be implemented by either binary search or by hashing (see e.g. [2, 6, 11, 9]). In any case this is a low order multiplicative factor that vanishes in the $\tilde{O}$ notation.

Definition 4.2 *Let G be a family of functions such that for all $g \in G$: $g : D \mapsto R$ (where D and R are some finite sets). Let g be chosen uniformly at random from G. We call G k-*wise independent *if for all* distinct $x_1, x_2, \ldots x_k \in D$*, the random variables $g(x_1), g(x_2), \ldots, g(x_k)$ are independent and uniformly distributed in R.*

Families of k-wise independent functions have been studied and applied extensively in recent years (cf. [3, 6, 7, 8, 14, 18]). A useful property is that if G is k-wise independent, then for all $1 \le i \le k$, for all $x_1, x_2, \ldots, x_i \in D$, and $v_1, v_2, \ldots v_{i-1} \in R$: the distribution of $g(x_i)$ given that $(g(x_1) = v_1, g(x_2) = v_2, \ldots, g(x_{i-1}) = v_{i-1})$ is uniform in R. For our application we will need a family of functions $g : \{1, \ldots, N\} \times \{1, \ldots, N\} \mapsto \{1, \ldots, N\}$.

4.2 The Scheme

Our scheme uses the parameters ℓ, t, and m to define a point on the time/space tradeoff curve: We have space $S = \tilde{O}(\ell \cdot m)$ and inversion time $T = \tilde{O}(\ell \cdot t)$. The restrictions on t, ℓ and m are that $t \le \ell$, $t \ge 8 \log(4m)$ and $t \cdot \ell \cdot m = N$. We also use another parameter k which is set $8 \cdot t$ in this section. This parameter one of the few things we have to change in order to handle trivial to invert functions as well in Section 5. In order to achieve the time/space tradeoff given in (2) we also use the constraint $mk^2 \le S$ and to achieve the time/space tradeoff given in (1) we use the constraint $mk^2 \le 1/q(f)$.

To simplify explanations, we define $\tilde{S}$ be $3S \cdot \log^2 N$, we actually use $\tilde{S}$ space, not S. For $S = N^\delta$, $0 < \delta < 1$, the difference between S and $\tilde{S}$ disappears in the $\tilde{O}$. The other values in the tradeoff follow trivially.

Set $k = 8 \cdot t$ (this is one of the few things we have to change in order to handle trivial to invert functions as well). The scheme uses a family G of k-wise independent functions $g : D \times \{1, \ldots, N\} \mapsto D$. In all ℓ functions $g_1, g_2, \ldots, g_\ell$ are applied, where each g_i is uniformly distributed in G. The scheme requires that for any $1 \le i < j \le \ell$ the random variables g_i and g_j be independent, i.e. given the value of g_i at all the points we learn nothing about g_j. The family G and the method by which the ℓ functions are chosen is described in Section 4.4. As we shall see, $g_1, g_2, \ldots, g_\ell$ are *not* completely independent.

The scheme described in the following section only yields that any element has a constant probability (over the coin flips of the preprocessing phase) of being inverted successfully. However, by repeating the full scheme (both preprocessing and the on-line inversion) $O(\log N)$ times we can get that with probability at least $1 - 1/N$ *all* range members can be inverted successfully in one of the schemes (this $O(\log N)$ term is absorbed in the $\tilde{O}$ notation).

4.2.1 Preprocessing for One Data Structure

The preprocessing phase consists of two stages, a choice of table A and the choice of the chains. The table A will contains pairs of values of the form $\langle x, f(x) \rangle$, sorted or hashed so that it is easy to search if $f(x)$ is in the table and find the corresponding entry x. The table A is constructed by a random process: choose $\tilde{S}$ random values $x_1, \ldots, x_{\tilde{S}} \in D$, and store pairs $\langle x_i, f(x_i) \rangle$ in A.

Remark: the table A is used to check if a point y has high indegree. If y has high indegree then there is high probability that $y = f(x)$ will appear in A. The preimage of y under f is only used to invert y itself but the main use of the table A is to avoid the high indegree points during the search. We use the notation $y \in A$ to mean that there is an entry of the form $\langle x, y \rangle$ in A.

The choice of A defines the functions g_i^* and h_i: For $1 \le i \le \ell$, define $g_i^*(x) = g_i(x, j)$, where $1 \le j \le k/2$ is the smallest value such that $f(g_i(x, j)) \notin A$ [2]. If there is not such

---

[2] Recall that each $g_i : D \times \{1, \ldots, N\} \mapsto D$

j, then $g_i^*(x)$ is undefined. Define $h_i(x) = g_i^*(f(x))$ (or undefined if $g_i^*(f(x))$ is undefined). Evaluating $h_i(x)$ requires j evaluations of f and g_i. Note that for any $x \in D$ the expected value of j (assuming that f is not trivial to invert) is $O(1)$.

The second preprocessing stage consists of choosing the chains that are suppose to cover all the range. It is done in the following way:

For all $1 \leq i \leq \ell$:

1. For all $1 \leq j \leq m$ pick x_{ij} at random from D.
2. Compute pairs of the form $(h_i^t(x_{ij}), x_{ij})$. If $h_i^t(x_{ij})$ is undefined or if computing $h_i^t(x_{ij})$ requires more than $k/2$ invocations of g_i then discard x_{ij}. (Recall that evaluation of h_i at different points may result in a different number of invocations of g_i. Here we are interested in the total number of invocations).
3. Store a table T_i, with m entries $(h_i^t(x_{ij}), x_{ij})$.

Define a *chain* rooted at x_{ij} to be the set $\{h_i^p(x_{ij}) | 1 \leq p \leq t\}$. Define the ith *cluster*, $1 \leq i \leq \ell$, to be the set

$$C_i = \{h_i^p(x_{ij}) | 1 \leq p \leq t, 1 \leq j \leq m\}.$$

Every cluster C_i is the union of the m chains of length t. The chains may overlap one another (two chains are said to overlap if they share one or more elements). We say that an image $y = f(x)$ is *contained* in the chain rooted at x_{ij} if for some $w \in \{h_i^p(x_{ij}) | 0 \leq p \leq t-1\}$ we have $f(w) = y$. We say that y is in the ith cluster if for some $1 \leq j \leq m$ it is contained in the chain rooted at x_{ij}.

4.2.2 On-line Inversion

We now explain how the tables constructed in the preprocessing phase are used to invert f. Given $y = f(x)$ we apply the following procedure to invert it:

- Search if $y \in A$. If found then we're done, the inverse is the x such that $\langle x, y \rangle$ is in A .
- Otherwise, for all $1 \leq i \leq \ell$:
  1. Set $u_i = g_i^*(y)$. and h_i (i.e. they do not participate in the on-lien inversion).
  2. Repeat for $p = 0$ to $t-1$:
     (a) Search for u_i in T_i. If found, ($u_i = h_i^t(x_{ij})$ for some j) then set z to be $h_i^{t-p}(x_{ij})$. If $f(z) = y$ then we have found a pre-image for y.
     (b) Set $u_i \leftarrow h_i(u_i)$ if defined. Otherwise, discard u_i.

This concludes the description of the scheme, except for how the g_i's are chosen, represented, and computed.

4.3 Analysis of the Scheme

We now turn to the analysis of the scheme. In this section, we use the assumption that computing the g_i functions can be done in $O(1)$ time. This will be justified in Section 4.4. Since f is arbitrary, all we know about $q(f)$ is that $1/N \leq q(f) \leq 1$. The net result of our construction of the table A is to reduce the effective $q(f)$ so that $q(h_i) \in O(1/S)$ with high probability. I.e. given a choice of the table A and given that x_1 and x_2 are chosen at random in D, we consider

$$Q_A = \text{Prob}[f(x_1) = f(x_2) \text{ and } f(x_1), f(x_2) \notin A] \ .$$

We know that $Q_A \leq q(f)$. However, we can treat Q_A as a random variable over the probability space defined by choices of A according to the preprocessing phase of the algorithm $\mathcal{A}$. What we show is that with high probability Q_A is less than $1/S$ over this probability space.

Lemma 4.1 *For any function f, with probability at least $1 - 1/\log N$ over the random choices of A,*

$$Q_A \leq 1/S.$$

Proof: Let $I(1), I(2), \ldots, I(N)$ be the sequence of indegrees under f, i.e. $I(j) = |\{x | f(x) = j\}|$. The probability that $f(x) \notin A$ is $(1 - I(f(x))/N)^{\tilde{S}}$. Thus,

$$\begin{aligned} E[Q_A] &= \text{Prob}[f(x_1) = f(x_2) \text{ and } f(x_1) \notin A] \\ &= \sum_{i=1}^{N} \frac{I(i)^2}{N^2} \cdot \left(1 - \frac{I(i)}{N}\right)^{\tilde{S}} \\ &= \sum_{\{j | I(j) > \frac{N}{S \log N}\}} \frac{I(j)^2}{N^2} \cdot \left(1 - \frac{I(j)}{N}\right)^{\tilde{S}} + \sum_{\{j | I(j) \leq \frac{N}{S \log N}\}} \frac{I(j)^2}{N^2} \cdot \left(1 - \frac{I(j)}{N}\right)^{\tilde{S}} . \end{aligned}$$

We deal with each term separately. Recall that $\tilde{S} = S \log^2 N$.

$$\begin{aligned} \sum_{\{j | I(j) > \frac{N}{S \log N}\}} \frac{I(j)^2}{N^2} \cdot \left(1 - \frac{I(j)}{N}\right)^{\tilde{S}} &\leq N \cdot 1 \cdot \left(1 - \frac{1}{S \log N}\right)^{\tilde{S}} \\ &\leq N \cdot 1/N^3 \\ &= 1/N^2. \end{aligned}$$

The sum $\sum_{i=i}^{j} x_i^2$ subject to the constraints $\sum_{i=1}^{j} x_i \leq c$ and $x_i \leq b$ is maximized by taking c/b x_i's equal to b and the rest zero. This is used in the following analysis:

$$\begin{aligned} \sum_{\{j | I(j) \leq \frac{N}{S \log N}\}} \frac{I(j)^2}{N^2} \cdot \left(1 - \frac{I(j)}{N}\right)^{\tilde{S}} &\leq \sum_{\{j | I(j) \leq \frac{N}{S \log N}\}} \frac{I(j)^2}{N^2} \\ &\leq S \log N \frac{N^2}{N^2 S^2 \log^2 N} \\ &= \frac{1}{S \log N} \end{aligned}$$

Thus, from Markov's inequality we get that Prob$[Q_A > 1/S]$ is at most $\frac{1}{\log N}$, which suffices for our purposes. By a more careful analysis we can get a much smaller probability. □

Fix $y = f(x)$, if $y \in A$ we can invert f on y via table lookup in A. Thus, we should handle only the case that $I(y) \le N/S$, since otherwise the probability that $y \notin A$ is at most $1/N$. We assume from now on that $y \notin A$. Let V_i be the event that $f(x)$ is in the ith cluster. The main lemma we require is as follows:

Lemma 4.2 *For all functions f, for all $y = f(x)$, and for all choices of m, t, and S such that $mt^2 < S/2$, assuming $Q_A \le 1/S$ then either $y \in A$ or*

$$Prob[V_i] \ge \frac{mt}{2N}.$$

To prove the lemma we need the following claims:

Claim 4.1 *Assuming $Q_A \le 1/S$, the probability that a chain rooted at x_{ij} contains less than t different elements is at most $1/2m$.*
Proof: We first compute that probability that the chain rooted at x_{ij} is discarded at Step 2 of the preprocessing phase and show that it is negligible. Then we bound the probability that the chain cycles on h.

For a chain to be discarded at Step 2 of the preprocessing phase it may be the case that more than $k/2$ applications of g_i were invoked. This occurs if through the attempted computation of h_i, $k/2$ invocations of g_i were performed altogether, while less than t of them gave images x such that $f(x)$ was not in the table A. Since g_i is $(k = 8 \cdot t)$-wise independent, it follows that the the probability that this occurs is at most the probability that a sequence of $k/2$ tosses of a coin contains at least $k/2 - t$ tails (the event g_i maps to a value x such that $f(x) \in A$) and less than t heads (the event g_i maps to a value x such that $f(x) \notin A$), with probabilities $p < 1/2$ for tails (by the assumption that f is not trivial to invert). By the Chernoff Bound (see Theorem A.1 in [4]) this probability is at most $e^{-t^2/k} = e - t/8 \le 1/4m$.

Even if the chain rooted at x_{ij} is not discarded, it may contain less than t different elements in case h_i cycles. This occurs if $f(z_1) = f(z_2) \notin A$ where z_1 and z_2 are two values encountered in the development of the chain either as x_{ij} or as the outcome of g_i. From the k-wise independence of g_i and the fact that g_i was invoked at most $k/2$ times in each chain we have that z_1 and z_2 are uniformly distributed at D. The probability that two such elements collide under f and not in A is at most $1/S$ by the assumption that $Q_A \le 1/S$ is at most $1/S$. There are at most $\binom{k/2}{2}$ candidates for such pairs, so the probability that this is the case is bounded by

$$\frac{\binom{k/2}{2}}{S} < \frac{k^2}{8S} \le \frac{1}{8m}.$$

Note that if $g_i(z_1, j_1) = g_i(z_2, j_2)$ but $f(g_i(z_1, j_1)) \in A$ where $z_1 \neq z_2$ are two values encountered in the development of the chain as the outcome of f and $1 \le j_1, j_2 \le N$ are the appropriate indices, then a cycle does *not* occur.

The probability that the chain is *not* discarded and that it does *not* cycle on h_i is thus at least $1 - /2m$. □

Let E^y_{ij} be the event that chain rooted at x_{ij} contains y and let $p_y = Prob[E^y_{ij}]$.

Claim 4.2 *For any $y = f(x)$ such that $I(y) \leq N/S$, and for parameters as defined in Lemma 4.2, either $y \in A$ or*

$$p_y \geq (1 - 1/2m) \cdot \frac{t}{N}$$

under the assumption that $Q_A \leq 1/S$.

Proof: Let $w_1, w_2, \ldots w_{k/2}$ be independent random variables uniformly distributed in D. These random variables define the random choices of chain: each time g_i is invoked at a new point we treat its value as the next w_a (here we are using the k-wise independence of g_i). In case the chain cycles, the rest of the variables are never used, as is the case following t successful hits in the set outside A. We are interested in the event *the chain defined by $w_1, w_2, \ldots w_{k/2}$ is not discarded* and *$f(w_a) = y$ for $1 \leq a \leq k/2$ which is among the first t elements of $w_1, w_2, \ldots w_{k/2}$ such that $f(w_i) \notin A$.* There are t possible places where y can appear in the chain. For each such location the probability that y is hit is at least $1/N$. Given that y is hit in the jth position, the probability that the chain is discarded or contains less than t elements is at most $1/2m$, from the analysis of Claim 4.1 (using the facts that $Q_A \leq 1/S$ and $I(y) \leq N/S$). Therefore with probability at least $(1 - 1/2m) \cdot \frac{t}{N}$ we have that the chain contains y and is not discarded. □

From the inclusion/exclusion principle,

$$\text{Prob}[V_i] \quad \geq \quad \sum_{j=1}^{m} \text{Prob}[E^y_{ij}] - \sum_{j<j' \in \{1,\ldots,m\}} \text{Prob}[E^y_{ij} \cap E^y_{ij'}]. \tag{3}$$

The next claim says that though E^y_{ij} and $E^y_{ij'}$ are *not* independent, we can bound their correlation:

Claim 4.3 *Assuming $Q_A \leq 1/S$ and $I(y) \leq N/S$, $Prob[E^y_{ij'} \cap E^y_{ij}] \leq \frac{1}{2m-1} \cdot p_y$*

Proof: The probability of the event $E^y_{ij'} \cap E^y_{ij}$ is dominated by the following: let $w_1, w_2, \ldots w_{k/2}$ and $w'_1, w'_2, \ldots w'_{k/2}$ be random variables uniformly distributed in D. Let q_y be the probability that following two conditions holds:

1. There exists $1 \leq a \leq k/2$ such that $f(w_a) = y$ and w_a is among there first t elements of $w_1, w_2, \ldots w_{k/2}$ such that $f(w_i) \notin A$.
2. There are $1 \leq b, c \leq k/2$ such that $f(w_b) = f(w'_c) \notin A$.

That is we are adding to $E^y_{ij'} \cap E^y_{ij}$ all the cases that chain x_{ij} hits y but cycles and the cases where the two chains collide after y is hit. By Claim 4.2 the probability that condition 1 is satisfied is at most $p_y/(1 - 1/2m) = p_y \frac{2m}{2m-1}$. Given that condition 1 is satisfied, the probability that condition 2 holds is at most $(k/2)^2/S \leq 1/4m$. Therefore

$$\text{Prob}[E^y_{ij'} \cap E^y_{ij}] \leq q_y \leq \frac{2m}{4m(2m-1)} \cdot p_y < \frac{1}{2m-1} \cdot p_y.$$

□

From the last claim, it follows that (3) is bounded from below by

$$m \cdot p_y - \binom{m}{2} \cdot p_y \cdot \frac{1}{2m-1} \geq mp_y \left(1 - \frac{m-1}{2m-1}\right) \geq \frac{mt}{2N},$$

where the last inequality holds from Claim 4.2. It follows that $\mathrm{Prob}[V_i] \geq \frac{mt}{2N}$. □

Lemma 4.3 *Assuming* $Q_A \leq 1/S$, *Prob*$[\cup_{i=1}^{\ell} V_i] \geq \frac{1}{4}$.
Proof: Given the table A, the events V_i and V_j are independent since the functions g_i and g_j are independent (note that prior to the choice of A the events may not be independent, since V_i's implies a successful choice of A which effects $Prob[V_j]$). We therefore have ℓ identical experiments where each is successful with probability $p \geq \frac{mt}{2N}$ and we are interested in the probability that at least one of them is successful. This is minimized when $p = \frac{mt}{2N}$ and from inclusion/exclusion

$$\begin{aligned}
\mathrm{Prob}[\cup_{i=1}^{\ell} V_i] &\geq \sum_{i=1}^{\ell} \mathrm{Prob}[V_i] - \sum_{1 \leq i < j \leq \ell} \mathrm{Prob}[V_i \cap V_j] \\
&\geq \frac{\ell \cdot m \cdot t}{2N} - \binom{\ell}{2} \cdot \frac{m^2 t^2}{4N^2} \\
&\geq \frac{1}{2} \cdot \frac{\ell \cdot m \cdot t}{N} - \frac{1}{8} \cdot \frac{\ell^2 m^2 t^2}{N^2} \\
&\geq \frac{1}{4}.
\end{aligned}$$

□

We now turn to the analysis of the run-time of the scheme. For each cluster $1 \leq i \leq \ell$ we have to apply g_i and f at most $k/2$ times at step 2b. In addition we must consider the complexity of step 2a, i.e. the number of times a chain is traced. Fix $y = f(x)$. Consider stage 2(a) of the inversion procedure above: for cluster i, F_i is defined to be a random variable that counts the number of times the test "$f(C) = y$" fails. This failure is called a false alarm.

Lemma 4.4 $E[F_i] \leq 1$.
Proof: Any y and every h_i induces a chain $u_i = g_i^*(y), h_i(u_i), h_i^2(u_i) \ldots, h_i^t(u_i)$. A false alarm at the ith cluster occurs if for some $1 \leq j \leq m$ the chain rooted at y and one of the chains rooted at x_{ij} overlap. There are m such chains, and from Claim 4.3 the probability that two chains merge is bounded by $(k/2)^2/S \leq 1/4m$. Hence the expected number of false alarms is bounded by 1. □

Thus, false alarms increase the time required per cluster by at most a factor of two. Therefore the total expected complexity of on-line inversion is $O(k \cdot \ell)$ and the following precursor to the main theorem holds:

Theorem 4.5 *For any function* f *not trivial to invert in space* S, *and for any image* $y = f(x)$, *the probability that* $f^{-1}(y)$ *is found is* $\Omega(1)$, *the expected number of evaluations of* f *and the* g_i*'s required to invert* $f(x)$ *is* $\tilde{O}(t \cdot \ell)$. *The space* $S = \tilde{O}(\ell \cdot m)$ *and the preprocessing time is* $\tilde{O}(N)$. □

4.4 Construction of the g_i's

We now specify how to choose the g_i's so that they can be computed efficiently. We do not know how to construct a family of k-wise independent functions so that evaluating a function in the family at a given point requires only $\tilde{O}(1)$ time. Instead we show how to construct a family so that the total computation can be amortized to $\tilde{O}(1)$ time. We change the order of operations in the basic time/space scheme. We do not process cluster by cluster but rather advance one step in every cluster simultaneously. In order to simplify the definition of the g_i's encode their domain (which is $D \times \{1, \ldots N\}$) as $\{1, \ldots N^2\}$ and the range as $\{1, \ldots N\}$. We actually construct functions $g_i : \{1, \ldots N^2\} \mapsto \{1, \ldots N^2\}$, but by simply chopping part of the output we can obtain $g_i : \{1, \ldots N^2\} \mapsto \{1, \ldots N\}$ and then consider it as an element of D.

Construct $g_1, g_2, \ldots g_\ell$ as follows: We work in the finite field $GF[N^2]$ (N is a power of 2), and all computations are in this field. For $0 \le j \le k-1$, pick a_j and b_j randomly and independently in $GF[N^2]$. Encode i as a element of $GF[N^2]$ in some arbitrary manner and let g_i be the polynomial of degree $k-1$ whose jth coefficient is $a_j \cdot i + b_j$. *I.e.*,

$$g_i(x) = \sum_{j=0}^{k-1} (a_j \cdot i + b_j) x^j.$$

Proposition 4.6 *A function g_i chosen in this manner is k-wise independent.*

Proof: As each g_i is a random polynomial of degree $k-1$, this follows from [3]. □

From the pair-wise independence of polynomials of degree 1, and the random choice of the a and b values above, the set of the $2k$ coefficients of g_i and g_j are independent and we have the following (which was used in Lemma 4.3):

Proposition 4.7 *For all $1 \le i < j \le \ell$, the polynomials g_i and g_j are independent as defined in Section 4.2.*

To compute $g_1(x_1), g_2(x_2) \ldots g_\ell(x_\ell)$ note the following:

$$\begin{aligned} g_i(x) &= \sum_{j=0}^{k-1} (a_j \cdot i + b_j) x^j \\ &= i \cdot \sum_{j=0}^{k-1} a_j x^j + \sum_{j=0}^{k-1} b_j x^j. \end{aligned}$$

Define

$$A(x) = \sum_{j=0}^{k-1} a_j x^j,$$

and

$$B(x) = \sum_{j=0}^{k-1} b_j x^j.$$

Thus, $g_i(x) = i \cdot A(x) + B(x)$. Using FFT we can amortize the cost of computing A and B. We can compute $A(x_1)$, $A(x_2) \ldots A(x_\ell)$ and $B(x_1)$, $B(x_2)$,..., $B(x_\ell)$ at a cost of $O(k \log \ell +$

ℓ). (See [1] Chapter 7). Given $A(x_1), A(x_2) \ldots A(x_\ell)$ and $B(x_1), B(x_2), \ldots, B(x_\ell)$, computing $g_i(x_i) = i \cdot A(x_i) + B(x_i)$ for all $1 \leq i \leq \ell$ can be done at a cost of ℓ.

Thus the total number of operations required to advance one function computation in all g_i is $O(k \log \ell + \ell) = O(t \log \ell + \ell)$.

Lemma 4.8 *Computing $g_1, g_2, \ldots g_\ell$ at points $x_1, x_2, \ldots x_\ell$ can be done in time $O(t \log \ell + \ell)$.*

In order to achieve the time/space tradeoff (2), given any T and S set $t = N/S$, $m = N/T$ and $\ell = TS/N$. If $TS^3 = N^3$, then the two constraints of Section 4.2, $t \leq \ell$ and $mk^2 \leq S$, are satisfied and since we may assume that $S < N/8logN$ we also have that $t \geq 8 \log(4m)$. Therefore by Theorem 4.5 we have the desired time and space.

In order to achieve the time/space tradeoff (1), given any T and S such that $T \cdot S^2 = N^3 q(f)$, set $t = N/S$, $m = N/T$ and $\ell = TS/N$. Note that we have $t \leq \ell$, $mk^2 \leq 1/q(f)$ and $k/S \leq k^2 q(f)$. The conditions $mk^2 \leq 1/q(f)$ and $k/S \leq k^2 q(f)$ are sufficient to replace the constraint $mk^2 \leq S$ in the proofs of Claims 4.1, 4.2 and 4.3. Therefore we have

Corollary 4.9 *Any function f which is not trivial to invert in space S can be inverted at any point with high probability, using either time/space tradeoffs:*

$$\begin{aligned} T \cdot S^2 &= N^3 \cdot q(f), \text{ or} \\ T \cdot S^3 &= N^3. \end{aligned}$$

5 Extensions

For simplicity we have assumed in Section 4 that the function f is not trivial to invert in space S. To complete the proof of Theorem 1.1 we must also deal with this case.

If the S highest indegree images cover a substantially large fraction of the domain, then our scheme may suffer degradation in performance, since when we cannot claim that when evaluating $h_i(x)$ the expected number of evaluations of f and g_i is $O(1)$. On the other hand, in this case the table A covers a larger fraction of images and thus the effective domain size is smaller. We will see how to exploit this fact.

Suppose that following the first stage of the preprocessing phase it turns out that the number of elements in $D' = \{x \in D | f(x) \notin A\}$ is $N' < N/2$ (i.e. the function is trivial to invert in space S). Since all $x \notin D'$ are handled by A, we need to construct a scheme that covers only D', thus raising the possibility of only improving the complexity. Our problem is that we do not know how to construct g_i's that map into D' and not D, so we may need more applications of the g_i's in order to assure that we land in D'. For S and T set as before $m = N/T$ and $\ell = TS/N$, but set $t = N'/S$ and $k = 8N/N' \cdot t = 8N/S$. Note that $t \cdot \ell \cdot m = N'$ and $mk^2 \leq S$ and given that $TS^3 \geq N^3$ then $k \leq \ell$ (this last point is significant for the amortization of computing the g_i's).

Therefore almost all the analysis of Sections 4.3 and 4.4 is applicable and in particular the space is $\tilde{O}(S)$ and the time is $\tilde{O}(T)$. The only point we should change is in the proof of Claim 4.1. We must recompute the probability that a chain rooted at x_{ij} is discarded since g_i was invoked more than $k/2$ times. However this means that in order to find t members of D' more than $k/2 = 4 \cdot N/N' \cdot t$ trials were needed, where in each time the probability of hitting a member of D' is N'/N. The probability that this occurs is at most the probability that a sequence of $k/2$ tosses of a coin contains at least $k/2-t$ tails (i.e. the event g_i maps to a value x such that $f(x) \in A$) and less than t heads (the event g_i maps to a value x such that $f(x) \notin A$), with probability $p = N'/N$ for heads (by the assumption that $|D'| = N'$). By the Chernoff Bound (See Theorem A.13 of [4]) this value is at most $e^{-9t^2/4t}$ which is smaller than $1/m$. This concludes the proof of Theorem 1.1.

Finally, we note that $q(f)$ need not be given explicitly in order to choose the best time/space tradeoff. We can consider an additional "proposal" stage in which a function f is studied to determine what time/space tradeoffs can be used to invert it. By estimating $q(f)$ we can check which tradeoff to use: if $S > 1/q(f)$ we use the tradeoff $T \cdot S^3 = N^3$, otherwise we use $T \cdot S^2 = N^3 \cdot q(f)$. Note that estimating the ith bit of $q(f)$ requires 2^i time. We can obtain a slightly pessimistic tradeoff by taking a slightly higher value for $q(f)$.

The "proposal" stage should also consider how $q(h_i)$ drops as a function of S. We can estimate $q(h_i)$ by estimating the probability that two elements collide under f, given that they *do not* map to any of the S images with largest indegrees. Even if $q(f)$ is big, $q(h_i)$ might drop very quickly as a function of S, becoming much smaller than $1/S$.

6 Open Problems

The most challenging problem is this area of investigation is whether the time/space tradeoff for inverting functions can be reduced to $T \cdot S = N$. For permutations this is true and Yao [19] has shown that this is tight for permutations. A. Shamir and J. Spencer (unpublished manuscript) consider schemes of the basic Hellman form, where functions are represented as a collection of chains indicated by their start and end locations, and show that $TS^2 = N^2$ is optimal for these algorithms.

Acknowledgments

We thank Noga Alon and Uri Feige for fruitful discussions. We thank the diligent referees for their many helpful comments.

References

[1] A. Aho, J. Hopcroft and J. D. Ullman, **Design and analysis of computer algorithms**, Addison-Wesley, 1974.

[2] A. Aho and D. Lee, *Storing a dynamic sparse table*, Proc. 27th IEEE Symp. on Foundations of Computer Science, 1986, pp. 55-60.

[3] N. Alon, L. Babai and A. Itai, *A fast and simple randomized algorithm for the maximal independent set problem*, Journal of Algorithms 7, pp. 567-583, 1987.

[4] N. Alon and J. Spencer, **The Probabilistic Method**, Wiley, 1992.

[5] H. R. Amirazizi and M. E. Hellman, *Time-memory-processor trade-off*, IEEE Trans. Infor. Theory 34, 1988, pp. 505-512.

[6] J. L. Carter and M. N. Wegman, *Universal classes of hash functions*, Journal of Computer and Systems Sciences, 18 (1979), pp. 143-154.

[7] B. Chor and O. Goldreich, *On the power of two-point based sampling*, Journal of Complexity, 5 (1989), pp. 96-106.

[8] B. Chor, O. Goldreich, J. Hastad, J. Friedman, S. Rudich and R. Smolensky, *The bit extraction problem or t-resilient functions*, Proc. of the 26th IEEE Symposium on Foundations of Computer Science, (1985) pp. 396-407.

[9] M. Dietzfelbinger, A. Karlin, K. Mehlhorn, F. Meyer auf der Heide and R. E. Tarjan, *Dynamic perfect hashing: upper and lower bounds*, SIAM Journal of Computing, Vol. 23, No. 4, pp. 738-761, 1994.

[10] A. Fiat, S. Moses, A. Shamir, I. Shimshoni, and G. Tardos, *Planning and learning in permutation groups*, Proc. of 30th IEEE Symposium on Foundations of Computer Science, 1989, pp. 274 – 279.

[11] M.L. Fredman, J. Komlós and E. Szemerédi, *Storing a Sparse Table with* $O(1)$ *Worst Case Access Time*, Journal of the Association for Computing Machinery, Vol 31, 1984, pp. 538–544.

[12] M. E. Hellman, *A cryptanalytic time memory trade-off*, IEEE Trans. Infor. Theory 26, 1980, pp. 401-406.

[13] M. E. Hellman, and J. M. Reyneri, *Drainage and the DES, Summary*, Advances in Cryptology - Proceedings of Crypto '82, pp. 129 – 131, Plenum press.

[14] M. Luby, *A simple parallel algorithm for the maximal independent set*, Siam J. on Computing, 15 (1986), pp. 1036-1053.

[15] R. C. Merkle and M. E. Hellman, *Hiding information and signatures in trapdoor functions*, IEEE Trans. Infor. Theory 24, 1978, pp. 525-530.

[16] S. C. Pohlig and M. E. Hellman, *An improved algorithm for computing logarithms over* $GF[p]$ *and its cryptographic significance*, IEEE Trans. Infor. Theory 24, 1978, pp. 106-110.

[17] R. Schroepel and A. Shamir, *A* $T = O(2^{n/2})$, $S = O(2^{n/4})$ *algorithm for certain NP-complete problems*, SIAM J. Computing, 10 (1981), pp. 456-464.

[18] A. Siegel, *On universal classes of fast high performance hash functions, their time-space tradeoff and their applications*, Proc. of 30th IEEE Symposium on Foundations of Computer Science, 1989, pp. 20-25.

[19] A. C. Yao *Coherent functions and program checkers.* Proc. 22nd ACM Symposium on Theory of Computing, 1990, pp. 84-94.