



**HAL**  
open science

# Algorithms for Sparse Random 3XOR: The Low-Density Case

Charles Bouillaguet, Claire Delaplace, Antoine Joux

► **To cite this version:**

Charles Bouillaguet, Claire Delaplace, Antoine Joux. Algorithms for Sparse Random 3XOR: The Low-Density Case. 2021. hal-02306917v4

**HAL Id: hal-02306917**

**<https://hal.science/hal-02306917v4>**

Preprint submitted on 2 Oct 2021

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# 1 Algorithms for the Sparse Random 3XOR Problem

2 **Charles Bouillaguet** 

3 Sorbonne Université, CNRS, LIP6, F-75005 Paris, France

4 charles.bouillaguet@lip6.fr

5 **Claire Delaplace** 

6 MIS Laboratory, Université de Picardie Jules Verne, 14 quai de la Somme, 80080 Amiens, France

7 claire.delaplace@u-picardie.fr

8 **Antoine Joux**

9 CISPA Helmholtz Center for Information Security, Saarbrücken, Germany

10 joux@cispa.de

---

## 11 — Abstract —

12 We present algorithms for variants of the 3XOR problem with lists consisting of random sparse  $n$ -bit  
13 vectors. We consider two notions of sparsity: low-density (each bit is independently biased towards  
14 zero) and low-weight (the Hamming weight of  $n$ -bit vectors is fixed).

15 We show that the random sparse 3XOR problem can be solved in strongly subquadratic time,  
16 namely less than  $\mathcal{O}(N^{2-\epsilon})$  operations for a constant  $\epsilon > 0$ . This stands in strong contrast with the  
17 regular case, where it has not been possible to have the exponent drop below  $2 - o(1)$ .

18 In the low-density setting, a very simple algorithm even runs in linear time with overwhelming  
19 success probability when the density is less than 0.0957. Our algorithms exploit the randomness  
20 (and sparsity) of the input in an essential way.

21 **2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Computational complexity and crypto-  
22 graphy; Theory of computation

23 **Keywords and phrases** Algorithms, 3-xor problem, random sparse 3-xor

## 1 Introduction

Given three lists  $A$ ,  $B$  and  $C$  of  $n$ -bit strings, the 3XOR problem consists in deciding the existence of (or even finding) a triplet  $(\mathbf{x}, \mathbf{y}, \mathbf{z}) \in A \times B \times C$  such that  $\mathbf{x} \oplus \mathbf{y} \oplus \mathbf{z}$  is equal to a given target, often assumed to be zero (here the  $\oplus$  symbol represents the exclusive-OR or XOR).

In the general setting where the input lists have size  $N$  and can be arbitrary, a simple algorithm decides the existence of a 3XOR triplet in time  $\mathcal{O}(N^2)$ , but there is no known way of doing it in  $\mathcal{O}(N^{2-\epsilon})$  operations for any constant  $\epsilon > 0$ . Dietzfelbinger, Schlag and Walzer found an algorithm that gains a poly-logarithmic factor over the quadratic algorithm [5].

3XOR can be seen as a variant of the celebrated 3SUM problem. In 3SUM, the input lists contain integers and we must have  $x + y + z = 0$  over  $\mathbb{Z}$ . Many geometric problems can be reduced to 3SUM in sub-quadratic time, and those problems are said to be 3SUM-hard [8]. Although the 3XOR problem has enjoyed less interest in the complexity theory field, there exists a few such reductions. For instance, it is a fact that any  $\mathcal{O}(N^{1+\epsilon})$  algorithm for the 3XOR problem with input lists of size  $N$  would imply faster-than-expected algorithms for listing triangles in a graph [9]. In the other direction, some conditional lower-bounds have been established based on the hypothesis that 3XOR is inherently quadratic: [5] shows that the offline SETDISJOINTNESS and SETINTERSECTION problems cannot be solved in time  $\mathcal{O}(N^{2-\epsilon})$  unless 3XOR can.

Besides being a natural extension of 3SUM, the 3XOR problem has some applications in the cryptanalysis of symmetric encryption schemes. Nandi’s attack [13] against the COPA mode of authenticated encryption, or the more recent attack against the two-round single-key Even-Mansour cipher by Leurent and Sibleyras [11] both reduce the problem to solving a large instance of the 3XOR problem.

In the context of cryptanalysis, the attacker has oracle access to three random functions  $f, g, h$  whose range is  $\{0, 1\}^n$ , and she must output a triplet  $(i, j, k)$  such that  $f(i) \oplus g(j) \oplus h(k) = 0$  as quickly as possible. In other terms, the attacker is free to *choose* a target size  $N$ , query the oracles and assemble input lists of size  $N$ , then solve this instance of the 3XOR problem. In order to minimize the total running time of the attack, the usual approach consists in assembling lists of size  $N \approx 2^{n/2}$ , which makes it possible to find a 3XOR triplet in time  $\tilde{\mathcal{O}}(N)$  by essentially sorting the lists. At this point, there is no known way to solve this “cryptanalytic version” of the 3XOR problem in  $2^{(0.5-\epsilon)n}$  operations, for any constant  $\epsilon > 0$ .

Cryptanalytic applications yield a family of 3XOR problems with a distinctive flavor: the input lists consist of *uniformly random* bit strings. Assuming random inputs is natural in the context of cryptanalysis, and this lead to several algorithms (predating [5]) gaining a polylogarithmic factor over  $N^2$  and often tailored to specific input sizes [10, 14, 4]. This randomness assumption makes the problem simpler, but even in this simpler case it has not yet been possible to obtain a  $\mathcal{O}(N^{2-\epsilon})$  algorithm.

Let us also mention that May and Both considered the “approximate 3-list birthday problem”: given three lists of  $N$  uniformly random  $n$ -bit strings, find triplets  $(\mathbf{x}, \mathbf{y}, \mathbf{z})$  in the list such that the hamming weight of  $\mathbf{x} \oplus \mathbf{y} \oplus \mathbf{z}$  is small [3].

In this paper, we consider an even more favorable setting for the 3XOR problem, where the input lists are both *random* and *sparse*. We consider two natural sparse distributions over  $\{0, 1\}^n$ , with a sparsity parameter  $0 < p < 1/2$ :

**Low-density**  $\mathcal{D}$  is the distribution where each bit is drawn independently from the Bernoulli distribution of parameter  $p$ .

71 **Low-weight**  $\mathcal{W}$  is the uniform distribution over  $n$ -bit strings of Hamming weight  $np$ .

72 We consider the problem of finding a 3XOR triplet in three lists  $A, B$  and  $C$  of size  $N$   
 73 containing  $n$ -bit strings drawn independently according to either  $\mathcal{D}$  or  $\mathcal{W}$ . The two settings  
 74 are quite different and we give an algorithm for each case. In both cases it is possible to find  
 75 a 3XOR triplet in  $\mathcal{O}(N^e)$  operations with  $e < 2$  when the input lists contain one.

76 **Motivation.** The two algorithms presented in this paper have no concrete application that  
 77 we are aware of, either in cryptography or elsewhere. Our initial motivation was the study  
 78 of the following strategy for the ‘‘cryptanalytic flavor’’ of the 3XOR problem: build three  
 79 lists of size  $2^{\mu n}$  with  $\mu = \frac{5}{24} \log_2 5 = 0.483\dots$ , then discard all vectors of weight  $\neq n/4$ . This  
 80 result in three smaller lists of expected size  $\approx 2^{\lambda n}$ , with  $\lambda = \frac{5}{24} \log_2 5 - \frac{3}{4} \log_2 3 + 1 = 0.295\dots$   
 81 They are expected to contain *one* 3XOR triplet, according to theorem 4 below.

82 If this instance of the random low-weight 3XOR problem could be solved in time  $\mathcal{O}(N^{1.69})$ ,  
 83 then the original problem would be solved in time  $\approx 2^{0.498n}$ , which would have been a  
 84 breakthrough. Unfortunately, algorithm 3 below is only capable of solving this problem in  
 85  $\mathcal{O}(N^{1.75})$  operations.

86 **Contributions.** In section 3, we first give the probability that the input actually contains a  
 87 3XOR triplet for a given list size  $N$  and parameter  $p$  in each setting, using a form of the  
 88 second-moment inequality. To the best of our knowledge, this result was not readily available  
 89 from the existing literature, not even in the dense case which is cryptographically relevant.

90 In section 4 we give an algorithm for the low-density 3XOR problem. It discards input  
 91 vectors whose Hamming weight is above a well-chosen threshold, then searches a solution  
 92 using the naive quadratic algorithm. This yields the

93 **► Theorem 1 (low-density).** *Write:*

$$94 \quad e = 2 + 6 \frac{D\left(\frac{2p^2}{1-2p+4p^2}, p\right)}{\ln(1-p)(1-2p+4p^2)}, \quad D(a, p) = a \ln \frac{a}{p} + (1-a) \ln \frac{1-a}{1-p}.$$

96  $D(a, p)$  is the Kullback-Leibler divergence between an  $a$ -coin and a  $p$ -coin. For all  $d > e$  there  
 97 is an algorithm for the random low-density 3XOR problem (with density  $p$ ) that runs in time  
 98  $\mathcal{O}(N + N^d)$ , where  $N$  denotes the size of the input list and fails to reveal a 3XOR triplet  
 99 present in the input with negligible probability (in  $n$ ).

100 This means that a solution can be found with overwhelming probability in time *linear* in  $N$   
 101 for small density (i.e.  $p < 0.0957$ , which makes  $e \leq 1$  in the theorem).

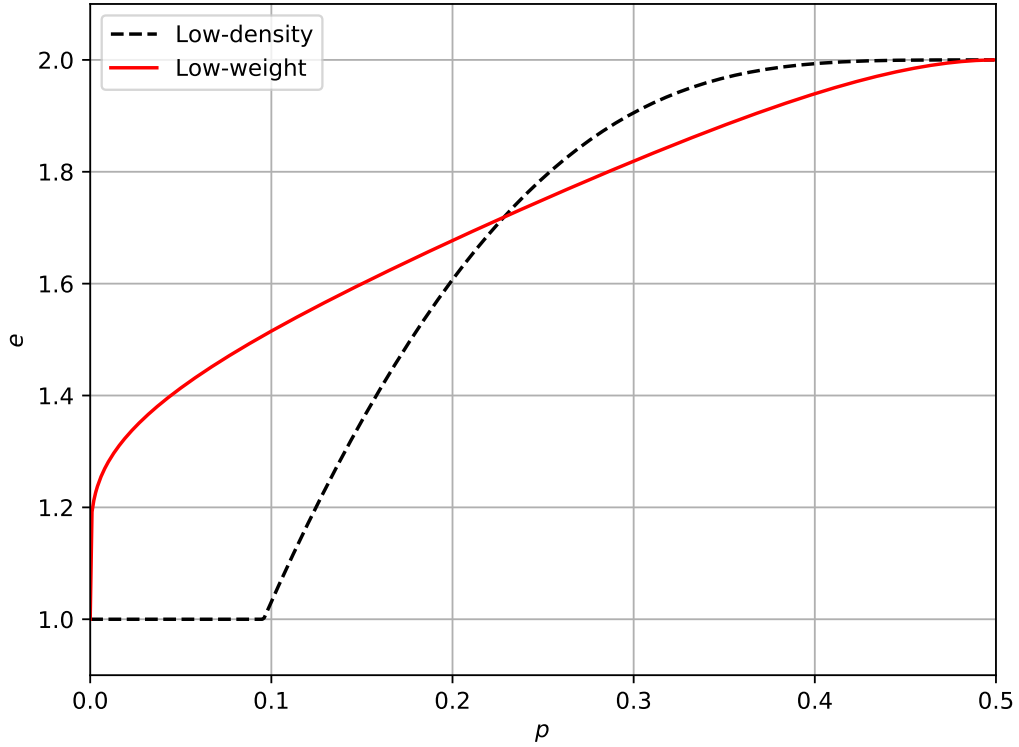
102 In section 5, we give an algorithm for the low-weight setting, inspired by an algorithm of  
 103 May and Ozerov [12] for Nearest-Neighbor search: it randomly guesses positions where bits  
 104 of the solution are 1, and uses this guess to reduce the size of the lists. This yields the other

105 **► Theorem 2 (low-weight).** *Write  $\underline{v} = p/2$  and*

$$106 \quad e = \frac{H(3\underline{v}) + 3\underline{v} \log_2 3}{(1 - 3\underline{v}) \log_2(1 - 3\underline{v}) - 3\underline{v} \log_2(4\underline{v}) - 3(1 - 2\underline{v}) \log_2(1 - 2\underline{v})}.$$

107 Assume that  $N$  is large enough (in a sense defined precisely in section 5). For any  $d > e$ ,  
 108 there is an algorithm that solves the low-weight random 3XOR problem (with weight  $np$ ) in  
 109 time  $\mathcal{O}(n^4 N^d)$  and fails to reveal a 3XOR triplet present in the input with exponentially  
 110 small probability as  $n \rightarrow \infty$ .

111 Figure 1 show the limiting exponents in theorems 1 and 2. We implemented these  
 112 algorithms ; appendix A presents a somewhat artificial application.



■ **Figure 1** The limit exponents of theorems 1 and 2: algorithms run in  $\mathcal{O}(N^d)$  for any  $d > e$ , where  $e$  is shown in the figure as a function of the sparsity of the input.  $p = 1/2$  corresponds to the random dense case.

## 113 2 Preliminaries

114 Let  $\mathbf{x} = x_1x_2 \dots x_n$  be an  $n$ -bit string (we use “bit string” and “vector” as synonyms). We  
 115 denote by  $\text{wt}(\mathbf{x})$  its Hamming weight. We identify a bit string  $\mathbf{x} \in \{0, 1\}^n$  with a subset  
 116  $S \subseteq \{1, 2, \dots, n\}$  in the obvious way ( $i \in S \Leftrightarrow x_i = 1$ ). For instance, with  $\mathbf{x}, \mathbf{y} \in \{0, 1\}^n$ , we  
 117 say that  $\mathbf{x} \subseteq \mathbf{y}$  (“ $\mathbf{x}$  is contained in  $\mathbf{y}$ ”) if  $(x_i = 1) \Rightarrow (y_i = 1)$ , for  $1 \leq i \leq n$ . In addition,  
 118 we define  $\mathbf{x}_{\downarrow \mathbf{y}}$  (“ $\mathbf{x}$  projected onto  $\mathbf{y}$ ”) to be the subword of  $\mathbf{x}$  formed by removing all letters  
 119 where  $\mathbf{y}$  is one. For instance  $0100101_{\downarrow 0011001} = 0110$  (bits 3, 4 and 7 are removed).

120 Let  $A$  be a list ;  $|A|$  denotes the number of elements in  $A$ . We say that  $(\mathbf{x}, \mathbf{y}, \mathbf{z})$  is a  
 121 *3XOR triplet* if  $\mathbf{x} \oplus \mathbf{y} \oplus \mathbf{z} = 0$ .

122 Let  $0 < p < 1/2$  be fixed and let  $\text{Ber}_p$  the Bernoulli distribution of parameter  $p$ . We also  
 123 denote by  $\text{Pois}(\lambda)$  the Poisson distribution with parameter  $\lambda$  and by  $\mathcal{B}(n, p)$  the binomial  
 124 distribution with parameters  $(n, p)$ .

125 **Bounds for Binomial Distributions.** Let  $H$  denote the binary entropy function, meaning  
 126 that  $H(x) = -x \log_2(x) - (1-x) \log_2(1-x)$ , for all  $0 < x < 1$ . The following standard  
 127 bounds for the binomial coefficient can be derived from Stirling’s formula:

$$128 \frac{2^{nH(x)}}{\sqrt{8nx(1-x)}} \leq \binom{n}{xn} \leq \frac{2^{nH(x)}}{\sqrt{2\pi nx(1-x)}}, \quad (0 < x < 1/2) \quad (1)$$

129 Let  $X \sim \mathcal{B}(n, p)$  be a binomial random variable. We make heavy use of tail bounds, notably  
 130 the Chernoff bound (2) and the tighter classical inequality (3), a proof of which can be found  
 131 in [1] amongst others. Here,  $D$  is again the Kullback-Leibler divergence as in theorem 1.

$$132 \quad \Pr(X \leq an) \leq \exp -\frac{n}{2p}(p-a)^2, \quad \text{if } a < p \quad (2)$$

$$133 \quad \Pr(X \leq an) \leq \exp -nD(a, p) \quad \text{if } a < p. \quad (3)$$

135 **Computational Model.** We consider a transdichotomous word Random Access Machine  
 136 (word-RAM) model. In this model, we have access to a machine in which each “memory  
 137 cell” contains an  $n$ -bit word (in other terms, we assume that the machine is large enough to  
 138 accommodate the instance of the problem at hand). We assume that the usual arithmetic  
 139 and bit-wise operations on  $n$ -bit words, as well as the comparison of two  $n$ -bit integers and  
 140 memory access with  $n$ -bit addresses can be done in constant time. We also assume that  
 141 obtaining the Hamming weight of an  $n$ -bit word is an elementary operation.

142 **The Quadratic Algorithm.** The simplest possible way to solve the 3XOR problem is the  
 143 quadratic algorithm (shown as algorithm 1). For all pairs  $(x, y) \in A \times B$ , check if  $x \oplus y$   
 144 belongs to  $C$ . Each test can be done in constant time if all entries of  $C$  have been stored in a  
 145 suitable data structure beforehand — for instance one could use the optimal static dictionary  
 146 of [7], or simply cuckoo hashing [15].

---

**Algorithm 1** The simple quadratic algorithm for 3XOR.

---

```

1: function QUADRATICSETUP( $C$ )
2:   Initialize a static dictionary  $\mathcal{C}$  containing all the bit strings in  $C$ 
3:   return  $\mathcal{C}$ 

4: function QUADRATIC3XOR( $A, B, C$ )
5:   for  $(x, y) \in A \times B$  do
6:     if  $x \oplus y \in C$  then
7:       return  $(x, y, x \oplus y)$ 
8:   return  $\perp$ 

```

---

147 The initialization of the dictionary holding  $C$  is separated from the rest, because we will  
 148 subsequently invoke QUADRATIC3XOR many times with the same  $C$ . We assume (using [7]  
 149 for instance) that QUADRATICSETUP takes time  $\mathcal{O}(|C|)$ . Then QUADRATIC3XOR( $A, B, C$ )  
 150 takes time  $\mathcal{O}(|A| \times |B|)$ . The quadratic algorithm works regardless of the sparsity of the  
 151 input, and as such it does not take advantage of it. It is the only solution, up to logarithmic  
 152 factors, when the input is dense.

### 153 **3** Bounds on the Existence of 3XOR Triplets

154 Suppose that  $(G, +)$  is a group (in additive notation) and assume that three lists  $A, B$  and  $C$ ,  
 155 each of size  $N$ , are made of group elements sampled independently at random according to  
 156 some distribution  $\Psi$  over  $\{0, 1\}^n$  (in this paper,  $\psi$  is either  $\mathcal{D}$  or  $\mathcal{W}$ ). Let  $Y$  be the random  
 157 variable that counts the number of triplets  $(x, y, z) \in A \times B \times C$  such that  $x + y + z = 0$   
 158 (taken over the random choice of  $A, B$  and  $C$ ). Assuming that the expected value of  $Y$   
 159 is bounded away from zero, a concentration inequality would yield a lower-bound on the  
 160 probability that the input lists contain at least one triplet that sums to zero. The problem is

## 6 Algorithms for the Sparse Random 3XOR Problem

161 that the  $N^3$  triplets in  $A \times B \times C$  are not independent and thus classical techniques such  
 162 as the Chernoff bound do not apply. A standard, but slightly less immediate argument  
 163 is necessary to take this dependence into account. Let  $x, y, z, u, v$  denote five independent  
 164 random variable distributed according to  $\Psi$ , and set:

$$\begin{aligned} 165 \quad \rho &= \Pr(x + y + z = 0) \\ 166 \quad \sigma &= \Pr(x + y + z = 0 \mid u + v + z = 0) \\ 167 \quad \tau &= \Pr(x + y + z = 0 \mid u + y + z = 0) \end{aligned}$$

169 Using a form of the second-moment inequality, we obtain:

170 **► Lemma 3.**  $EY = N^3\rho$  and  $1 - N^3\rho \leq \Pr(Y = 0) \leq \frac{1}{\rho} \left( \frac{3\sigma}{N} + \frac{3\tau}{N^2} + \frac{1}{N^3} \right)$ .

171 Specializing the result for the group of  $n$ -bit strings with the XOR operation and the two  
 172 distributions of interest to us yields the

173 **► Theorem 4.** *With the above notations, if the input distribution is  $\psi \in \{\mathcal{D}, \mathcal{W}\}$ , we have:*

$$174 \quad 1 - EY \leq \Pr(Y = 0) \leq \frac{3}{(EY)^{1/3}} + \frac{3}{(EY)^{2/3}} + \frac{1}{EY}. \quad (4)$$

175 *If  $\psi = \mathcal{D}$ , then  $\rho = (1 - p)^n(1 - 2p + 4p^2)^n$ .*

176 *If  $\psi = \mathcal{W}$ , then we have the following bound:*

$$177 \quad \left(\frac{\pi}{4}\right)^{3/2} \leq \frac{\rho}{\tilde{\rho}} \leq \left(\frac{4}{\pi}\right)^{3/2} \quad \text{with} \quad \tilde{\rho} = 8 \left(\frac{(1 - 2\underline{v})^3}{1 - 3\underline{v}}\right)^{1/2} 2^{n[3\underline{v} \log_2 3 + H(3\underline{v}) - 3H(2\underline{v})]}$$

178 Both proofs are in appendix B.

### 179 3.1 Implications

180 First of all, when the input is random, the 3XOR problem may potentially be easy to solve  
 181 with low error probability without even observing the input lists, depending on  $n$ , the size  
 182 and the distribution of the input. In light of theorem 4, we see that if  $N$  (the size of input  
 183 lists) is exponentially smaller than  $\rho^{-1/3}$ , then “**return**  $\perp$ ” is an algorithm that has an  
 184 exponentially small probability of yielding false negatives. Alternatively, in the low-density  
 185 case, if  $N$  is exponentially larger than  $(1 - p)^{-n}$ , then we expect the string  $000 \dots 0$  to be  
 186 present in  $A, B$  and  $C$  with overwhelming probability ; it follows that “**return**  $(0, 0, 0)$ ” is  
 187 a fast algorithm with exponentially low false positive probability. To avoid these pitfalls,  
 188 our main focus is on the hard case where  $N \approx \rho^{-1/3}$ , and where the expected number of  
 189 solutions in the random input lists is close to one.

190 In general, if the input lists are too long, we may use the following technique. Let  
 191  $N_0 := \rho^{-1/3}$  (input lists of this size contain a single 3XOR triplet in average) and suppose  
 192 that  $N > N_0 2^{\epsilon n}$ , where  $N$  denotes the size of the input lists. Assume that we have an  
 193 algorithm  $\mathcal{A}$  that solves the 3XOR problem with probability greater than some constant  
 194  $c > 0$  in expected time  $T$  for input lists of size  $N_0$ . The “exponent”  $e$  such that  $T = N_0^e$  can  
 195 be obtained for longer lists as well.

196 Slice the input lists in chunks of size  $4N_0$  and run algorithm  $\mathcal{A}$  on each successive chunk  
 197 until a solution is found. Theorem 4 tells us that each chunk contain 64 3XOR triplets on  
 198 average, and contain at least one 3XOR triplet with probability greater than  $3/64$ . Therefore,  
 199 a 3XOR triplet will be found in each chunk with probability greater than  $3c/64$ ; because the

200 chunks are completely independent parts of the random input, the events “ $\mathcal{A}$  finds a 3XOR  
 201 triplet in chunk  $i$ ” are independent. Therefore, the whole process fails to reveal a 3XOR  
 202 triplet with exponentially small probability, namely less than  $(1 - \frac{3\epsilon}{64})^{n\epsilon}$ . The running time  
 203 is  $\mathcal{O}(N \cdot N_0^{\epsilon-1})$ , which is less than  $N^\epsilon$ .

## 204 4 Random Low-Density 3XOR

205 In this section, we present a simple algorithm that solves the low-density 3XOR problem with  
 206 interesting theoretical guarantees when  $p$  is small. It is always subquadratic when  $p < 1/2$   
 207 but its most striking feature is that it succeeds with overwhelming probability in linear time  
 208 when  $p$  is small enough.

209 Let  $a, b$  and  $c$  be random bits drawn according to  $Ber_p$  conditioned to  $a \oplus b \oplus c = 0$ . The  
 210 possible combinations are 000, 011, 101 and 110. We find that

$$211 \quad u := \Pr(abc = 110 \mid a \oplus b \oplus c = 0) = p^2 / (1 - 2p + 4p^2)$$

213 The same result is attained for 101 and 011. Two out of the three non-zero options result  
 214 in a 1 bit, and therefore  $\Pr(a = 1 \mid a \oplus b \oplus c = 0) = 2u$ . It follows that if the  $(\mathbf{x}, \mathbf{y}, \mathbf{z})$  is a  
 215 triplet drawn from  $\mathcal{D}$  such that  $\mathbf{x} \oplus \mathbf{y} \oplus \mathbf{z} = 0$ , then the expected “density” of  $\mathbf{x}$ ,  $\mathbf{y}$  and  $\mathbf{z}$  is  
 216  $2u$ . This is always smaller than  $p$  when  $0 < p < 1/2$ . In other terms: *random triplets drawn*  
 217 *from  $\mathcal{D}$  have density  $p$ , but random 3XOR triplets drawn from  $\mathcal{D}$  have smaller density.*

218 This observation can be exploited in a simple way: we discard from the input lists all  
 219 vector above a given weight threshold then proceed using the naive quadratic algorithm.  
 220 This yields algorithm 2. It takes an additional argument  $w_{\max}$  controlling the maximum  
 221 allowed weight.  $w_{\max} = 2nu$  yields an algorithm that reveals a 3XOR triplet present in the  
 222 input with probability greater than  $1/4$ , because the median weight of both  $\mathbf{x}$  and  $\mathbf{y}$  is  $2nu$   
 223 if  $\mathbf{x} \oplus \mathbf{y} \oplus \mathbf{z} = 0$ . Setting  $w_{\max} = 2nu(1 + \epsilon)$  is enough to miss an existing solution with  
 224 exponentially small probability.

---

**Algorithm 2** Algorithm for the low-density 3XOR problem.

---

```

1: function LOW-DENSITY-3XOR( $A, B, C, w_{\max}$ )
2:    $A' \leftarrow \{\mathbf{x} \in A \mid \text{wt}(\mathbf{x}) \leq w_{\max}\}$ 
3:    $B' \leftarrow \{\mathbf{y} \in B \mid \text{wt}(\mathbf{y}) \leq w_{\max}\}$ 
4:    $\mathcal{C} \leftarrow \text{QUADRATICSETUP}(C)$ 
5:   return QUADRATIC3XOR( $A', B', \mathcal{C}$ )

```

---

225 The performance of algorithm 2 is summarized by theorem 1 in the introduction. It seems  
 226 at first glance that the limiting exponent satisfies  $e \leq 2p(1 - 2 \ln p)$ ; establishing this is a  
 227 fascinating endeavour that we must regrettably leave for future work. It is worth noting that  
 228 the not-so-friendly expression of  $e$  comes from the use of the tight binomial bound (3). It  
 229 would be greatly simplified had we instead used the simpler Chernoff bound (2). However,  
 230 doing so makes the result much worse for small  $p$ : it results in  $\lim e = 1$  when  $p$  goes to zero  
 231 (instead of  $\lim e = 0$ ).

232 The rest of this section is devoted to prove theorem 1. A sensible choice consists in picking  
 233  $w_{\max}$  in the range  $]2nu, np[$  — above the expected density of random 3XOR triplets so that  
 234 we do not discard them, and below the density of the input lists in order to actually discard  
 235 input vectors that are too heavy. In this case the algorithm succeeds with overwhelming  
 236 probability as long as the original input contains at least one solution.



## 8 Algorithms for the Sparse Random 3XOR Problem

237 ► **Lemma 5.** *With  $w_{\max} = 2nu(1 + \epsilon)$ , if the input contains a 3XOR triplet, then algorithm 2*  
 238 *returns  $\perp$  with probability less than  $2 \exp(-nu\epsilon^2)$ .*

239 **Proof.** Assume that the input lists contain a 3XOR triplet  $(\mathbf{x}^*, \mathbf{y}^*, \mathbf{z}^*)$ . It will be discarded  
 240 if and only if the weight of either  $\mathbf{x}^*, \mathbf{y}^*$  is greater than  $w_{\max}$ .

241 We know that the expected weight of  $\mathbf{x}^*$  and  $\mathbf{y}^*$  (and  $\mathbf{z}^*$  as well but this is irrelevant) is  
 242  $2un$ , therefore the Chernoff bound (2) shows that either has weight greater than  $2un(1 + \epsilon)$   
 243 with probability less than  $\exp(-nu\epsilon^2)$ . A union bound (for  $\mathbf{x}^*$  and  $\mathbf{y}^*$ ) then ensures that  
 244 the solution is discarded with probability less than  $2 \exp(-nu\epsilon^2)$ . ◀

245 ► **Lemma 6.** *Let  $T$  denote the running time of algorithm 2 with  $w_{\max} = 2nu(1 + \epsilon)$ . Then*  
 246  *$\mathbb{E}T \leq N + N^2 \exp(-2nD(2u(1 + \epsilon), p))$ .*

247 **Proof.** In the sequel, all the stated complexities must be understood “up to a constant  
 248 factor”. Filtering the input lists and keeping only low-weight vectors can be done in linear  
 249 time. From the complexity of QUADRATIC3XOR on input  $(A', B', C)$ , we obtain the total  
 250 time complexity:  $T = N + |A'| \cdot |B'|$ .

251 Let  $X \sim \mathcal{B}(n, p)$  be a binomial random variable modeling the weight of an input vector  
 252 of density  $p$ . Such a vector belongs to  $A'$  or  $B'$  if its weight is less than or equal to  $w_{\max}$ ,  
 253 and this happens with probability  $s := \Pr(X \leq w_{\max})$ . Because  $w_{\max} < np$ , the binomial  
 254 tail bound (3) yields the tight upper-bound  $s \leq e^{-nD(2u(1+\epsilon), p)}$ .

255 The sizes of  $A'$  and  $B'$  are stochastically independent random variables following a  
 256 binomial distribution of parameters  $(N, s)$  with expectation  $Ns$ . The expected running time  
 257 of the quadratic algorithm on  $A'$  and  $B'$  is therefore  $\mathbb{E}(|A'| \times |B'|) = \mathbb{E}|A'| \times \mathbb{E}|B'| = N^2 s^2$ .  
 258 Combining this with the upper-bound on  $s$  gives the announced result. ◀

259 **proof of theorem 1.** Let  $d$  be a complexity exponent greater than the bound  $e$  given in  
 260 theorem 1. Let  $\epsilon > 0$  be such that

$$261 \quad d = 2 + 6 \frac{D(2u(1 + \epsilon), p)}{\ln(1 - p)(1 - 2p + 4p^2) - 3\epsilon \ln 2}.$$

262 (such an  $\epsilon$  always exist). Note that setting  $\epsilon = 0$  in this expression yields the lower-bound  
 263 exponent  $e$  of the theorem.

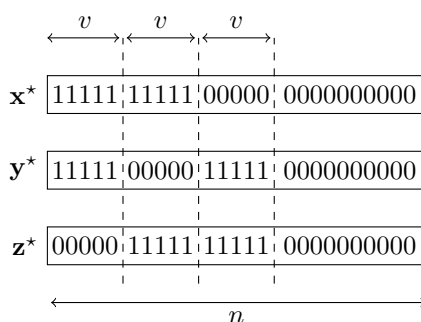
264 Let  $N_0 := \rho^{-1/3}$  (input lists of this size contain a single 3XOR triplet in average). Suppose  
 265 that  $N \leq N_0 2^{\epsilon n}$ , where  $N$  denotes the size of the input lists ; in this case run algorithm 2 with  
 266  $w_{\max} = 2un(1 + \epsilon)$ . Lemma 5 guarantees the exponentially small failure probability while  
 267 lemma 6 tells us that the expected running time  $T$  is less than  $N + N^2 \exp[-2nD(2u(1 + \epsilon), p)]$ .

268 Set  $d' := \log_N(\mathbb{E}T - N)$ , so that the algorithm runs in time  $\mathcal{O}(N + N^{d'})$ . A quick  
 269 calculation shows that  $d' \leq d$ , and the theorem is proved in this case.

270 If  $N > N_0 2^{\epsilon n}$ , then slice the input lists in chunks of size  $4N_0$  and run algorithm 2 with  
 271  $w_{\max} = 2un$  on each successive chunk until a solution is found. The reasoning in section 3.1  
 272 proves that this yields an algorithm that satisfies the conditions of theorem 1. ◀

### 5 Random Low-Weight 3XOR

274 Let us be given three lists of bit strings of weight  $w = pn$ , where  $w$  is an even integer (this  
 275 is a necessary condition for the existence of a 3XOR triplet). We assume that the input  
 276 lists contain a 3XOR triplet  $(\mathbf{x}^*, \mathbf{y}^*, \mathbf{z}^*)$ , and we want to find it. Following the reasoning  
 277 in section 3.1, we focus on input lists that are “long enough”. In particular, a different



■ **Figure 2** Shape of a 3XOR triplet  $(\mathbf{x}^*, \mathbf{y}^*, \mathbf{z}^*)$ , up to column permutation. Here,  $v = w/2$ .

278 algorithmic strategy would be required if lists were significantly shorter than necessary to  
 279 have a single solution in expectation. All the results in this section rely on this assumption.

280 A moment's reflection shows that each component of a 3XOR triplet has an intersection  
 281 of size  $w/2$  with the others (see fig. 2). This motivates the notation  $v := w/2$ . Our strategy  
 282 is the following: let  $s \leq v$  be a parameter to be determined later; guess a subset of size  $s$   
 283 of the intersection of  $\mathbf{x}^*$  and  $\mathbf{y}^*$  (this is a set of positions where  $\mathbf{x}^*$  and  $\mathbf{y}^*$  are both 1, thus on  
 284 the left segment of fig. 2); discard non-conformant vectors from  $A$  and  $B$ ; solve the smaller  
 285 resulting instance using the quadratic algorithm. However, instead of guessing random  
 286  $s$ -subsets of  $\{1, \dots, n\}$  *en bloc*, we proceed incrementally following an approach initiated by  
 287 May and Ozerov in [12] for nearest-neighbor search: guess *one* position in  $\mathbf{x}^* \cap \mathbf{y}^*$ , filter the  
 288 lists, solve the smaller subproblem recursively, repeat. The point is that each time the lists  
 289 are filtered, subsequent filtering steps start with smaller lists. This yields algorithm 3.

---

**Algorithm 3** An algorithm for the low-weight 3XOR problem.

---

```

1: function RECURSIVEFILTERING( $n, \alpha, A, B, C, k, s$ )
2:   if  $k = s$  then
3:     return QUADRATIC3XOR( $A, B, C$ )
4:   else
5:     Sample  $r$  according to the Poisson distribution with parameter  $n/\alpha$ .
6:     for  $r$  times do
7:        $\mathbf{m} \leftarrow$  random bit string of weight 1 in  $\{0, 1\}^n$     ▷ Correct with proba.  $\alpha/n$ 
8:        $A' \leftarrow \{\mathbf{x}_{\downarrow \mathbf{m}} \mid \mathbf{x} \in A \text{ and } \mathbf{m} \subseteq \mathbf{x}\}$     ▷ Keep elements that are 1 when  $\mathbf{m}$  is 1
9:        $B' \leftarrow \{\mathbf{y}_{\downarrow \mathbf{m}} \mid \mathbf{y} \in B \text{ and } \mathbf{m} \subseteq \mathbf{y}\}$ 
10:       $\zeta \leftarrow$  RECURSIVEFILTERING( $n - 1, \alpha - 1, A', B', C, k + 1, s$ )
11:      if  $\zeta \neq \perp$  then return  $\zeta$ 
12:    return  $\perp$ 
13: function LOW-WEIGHT-3XOR( $n, w, A, B, C, s$ )
14:    $C \leftarrow$  QUADRATICSETUP( $C$ )
15:   for  $n^2$  times do
16:      $\zeta \leftarrow$  RECURSIVEFILTERING( $n, w/2, A, B, C, 0, s$ )
17:     if  $\zeta \neq \perp$  then return  $\zeta$ 
18:   return  $\perp$ 

```

---

290 Here are a few comments. Assuming that  $A$  and  $B$  still contain a 3XOR triplet, each  
 291 random choice of  $\mathbf{m}$  is correct with a certain probability  $\alpha/n$  where  $\alpha = |\mathbf{x}^* \cap \mathbf{y}^*|$ . In order  
 292 to at least have one correct guess in expectation, the loop in RECURSIVEFILTERING should

do  $r \geq n/\alpha$  iterations. Naturally, this is not an integer, and rounding up is not an option because it would incur an exponential blow-up in the total number of iterations. Instead, we sample randomly the number of iterations, with expectation  $n/\alpha$ . Any easy-to-sample distribution could be used to choose  $r$ , but the choice of the Poisson distribution is necessary for us to prove a lower-bound on the success probability. The quadratic algorithm is run on sparse inputs but we do not know how to exploit this sparsity anymore. Guessing positions in segments of fig. 2 other than the left one leads to less efficient algorithms.

The rest of this section proves theorem 2. The recursive calls to RECURSIVEFILTERING corresponds to nodes in a tree. At depth  $k$  in this tree, a problem instance is composed of two filtered lists  $A_k$  and  $B_k$  of expected size  $N_k$ , containing  $n - k$ -bit words of weight  $w - k$  (no longer necessarily even). Assuming that they contain a solution, then  $\alpha_k := |\mathbf{x}^* \cap \mathbf{y}^*|$  denotes the size of the intersection and  $\alpha_k = v - k$ .

A randomly chosen position belongs to the intersection  $\mathbf{x}^* \cap \mathbf{y}^*$  with probability  $(v - k)/(n - k)$ ; filtering the lists by only keeping vectors that are 1 on this position yields lists of expected size  $N_{k+1} = N_k w_k / n_k$ . We define  $R_k := (n - k)/(v - k)$  as well as  $F_k := (2v - k)/(n - k)$ , with the intention that  $R_k$  is the expected number of iterations of the **for** loop of line 6 at depth  $k$  while  $F_k$  is the “filtering factor” at depth  $k$ . Indeed, the size of the input lists at depth  $k$  is given by  $N_k = N \prod_{j=0}^{k-1} F_j = N \binom{n-k}{2v-k} / \binom{n}{2v}$ . Finally, the tree of recursive calls has (on average)  $\prod_{j=0}^{k-1} R_j = \binom{n}{k} / \binom{v}{k}$  nodes at depth  $k$ . It is sometimes useful to abstract  $n$  away, so we define  $\underline{s} = s/n$  and  $\underline{v} = v/n$ .

We begin by studying the success probability of algorithm 3.

► **Lemma 7.** *If the input lists contain a 3XOR triplet, then Algorithm 3 returns  $\perp$  with exponentially small probability as  $n \rightarrow +\infty$ .*

**Proof.** Each edge in the tree of recursive calls of algorithm 3 corresponds to a random choice of  $\mathbf{m}$ . The algorithm succeeds if and only if there is a branch that reaches depth  $s$  where all choices are correct. Take an arbitrary depth- $k$  node in the tree of recursive calls. Each choice of  $\mathbf{m}$  is correct with probability  $(v - k)/(n - k)$  and the number of trials ( $r$  in the algorithm) is also random. Let  $X_1, X_2, \dots$  denote a sequence of Bernoulli random variables of parameter  $(v - k)/(n - k)$ ; the distribution of the total number of correct choices at each node is given by  $G = \sum_{i=1}^r X_i$ , where  $r$  follows a Poisson distribution of parameter  $(n - k)/(v - k)$ . The sum  $G$  follows a compound Poisson distribution, and the special case where the  $X_i$  are Bernoulli is well-known:  $G$  again follows a Poisson distribution of parameter 1 (independently of  $n, v$  and  $k$ ).

This means that the (random) subtree formed by the edges corresponding to *correct guesses* is a Galton-Watson tree with offspring distribution  $\text{Pois}(1)$ . The expected offspring number is 1 and the variance is also 1. Let  $Z_k$  denote the size of the  $k$ -th generation. The algorithm succeeds if  $Z_s > 0$ . This Galton-Watson tree is critical and a well-known result of Kolmogorov states  $\Pr(Z_k > 0) \sim 2/k$  when  $k \rightarrow +\infty$  (see [6, 2] for details).

This implies that  $nZ_s \rightarrow 1/(2\underline{s})$  when  $n \rightarrow +\infty$ . Because  $\underline{s} \leq \underline{v} \leq 1/4$ , we know that for large enough  $n$ ,  $nZ_s$  becomes greater than 2, and therefore  $Z_n$  becomes greater than  $1/n$ . This implies that for large enough  $n$ , the probability that the  $n^2$  iterations of the main loop of algorithm 3 fail to disclose a 3XOR triplet in the input lists is less than  $(1 - 1/n)^{n^2} \leq e^{-n}$ . ◀

We now move on to upper-bound the time complexity of algorithm 3. We proceed in two steps: 1) we minimize the total time spent in the leaves of the recursion tree, then 2) we show that the time spent in internal nodes of the tree is not much larger. In order to simplify our analysis, we assume that the input lists are large enough so that  $N_s \geq 1$ . This is to avoid the degenerate situation where the filtering steps reduce the size of lists to either

340 0 or 1, which means that the quadratic algorithm has nothing to do anymore and the proof  
 341 strategy outlined above breaks down. Note that this condition is always satisfied if the input  
 342 lists are long enough to contain one solution in expectation (because we have  $N_v \geq 1$  in this  
 343 case). This is why we make this hypothesis.

344 ► **Lemma 8.** *If  $N_s \geq 1$ , then the expected time spent in QUADRATIC3XOR by algorithm 3*  
 345 *is minimized by  $s = v(n - 4v)/(n - 3v)$ . Note that this depends only on the input weight.*

346 **Proof.** Let  $R$  denote the total number of calls to QUADRATIC3XOR by algorithm 3 and let  
 347  $FN$  denote the expected size of inputs to QUADRATIC3XOR. According to the above, we  
 348 find that  $R = \prod_{j=0}^{s-1} R_j = \binom{n}{s} / \binom{v}{s}$  and  $F = \binom{n-s}{2v-s} / \binom{n}{2v}$ . To get rid of  $n$ , we write  $N = 2^{n\tilde{N}}$ ,  
 349  $R = 2^{n\tilde{R}}$  and  $F = 2^{n\tilde{F}}$ . Using (1), we find that up to small constant factors :

$$350 \quad \tilde{R} \approx H(\underline{s}) - vH(\underline{s}/v),$$

$$351 \quad \tilde{F} \approx (1 - \underline{s})H((2v - \underline{s})/(1 - \underline{s})) - H(2v)$$

353 And therefore, the expected total time spent in the quadratic algorithm is  $n2^{n\tilde{\sigma}}$ , with

$$354 \quad \tilde{\sigma} := \tilde{R} + 2\tilde{F} + 2\tilde{N} = H(\underline{s}) - vH(\underline{s}/v) + 2(\tilde{N} + (1 - \underline{s})H((2v - \underline{s})/(1 - \underline{s})) - H(2v))$$

355 We seek the value for which  $\tilde{\sigma}$  reaches a local minimum by computing its derivative:

$$356 \quad \frac{d\tilde{\sigma}}{ds} = \log_2 \frac{1 - \underline{s}}{\underline{s}} - \log_2 \frac{v - \underline{s}}{\underline{s}} + 2 \log_2 \frac{2v - \underline{s}}{1 - \underline{s}}$$

357 And we solve  $\frac{d\tilde{\sigma}}{ds} = 0$ , which translates to  $(1 - \underline{s})(v - \underline{s}) = (2v - \underline{s})^2$ . The announced value is  
 358 the only solution of this equation. ◀

359 We now argue that the time spent filtering the lists in internal nodes of the recursion  
 360 tree does not dominate the whole computation. Given an integer  $0 \leq s \leq v$ , we focus on the  
 361 sequence of numbers  $\theta_0, \dots, \theta_s$  defined by

$$362 \quad \theta_k \log N_k = \log R_k + \theta_{k+1}(\log N_k + \log F_k) \quad \text{and} \quad \theta_s = 2.$$

363 The point is that a node at depth  $k$  in the recursion tree performs  $\tilde{\mathcal{O}}\left(N_k^{\theta_k}\right)$  operations, a  
 364 fact that we prove below. It implies that the number of operations in the leaf nodes dominate  
 365 that of the inner nodes, and therefore the value of  $s$  given by lemma 8 actually minimizes  
 366 the total running time of algorithm 3.

367 ► **Lemma 9.** *Let  $T_k$  denote the expected total time spent in the leaf nodes (i.e. in QUAD-*  
 368 *RATIC3XOR) that are below a given node of depth  $k$  in the tree of recursive call. Then*  
 369  $T_k = \mathcal{O}\left(N_k^{\theta_k}\right)$ .

370 **Proof.** The proof is by decreasing induction. At  $k = s$  (leaf nodes), the quadratic algorithm  
 371 is invoked, therefore  $T_s = N_s^2$  and we have  $\theta_s = 2$ . Up in the tree, we have  $T_k = R_k T_{k+1}$  and  
 372 by induction hypothesis there is a constant  $c$  such that  $T_{k+1} \leq cN_{k+1}^{\theta_{k+1}}$ . This implies that  
 373  $T_k \leq cR_k(F_k N_k)^{\theta_{k+1}} = cN_k^{\theta_k}$ , which proves the lemma. ◀

374 ► **Lemma 10.** *If  $s = v(n - 4v)/(n - 3v)$  and  $N_s \geq 1$ , then  $1 < \theta_k \leq 2$  for all  $0 \leq k \leq s$ .*

375 **Proof.** The proof is by decreasing induction starting with  $k = k_s$ , where by definition  $\theta_s = 2$ .  
 376 Next, assume that  $1 < \theta_{k+1} \leq 2$ . Because both  $N_k$  and  $N_{k+1} = F_k N_k$  are greater than 1 by  
 377 hypothesis, it follows that:

$$378 \quad \log R_k + \log N_k + \log F_k < \theta_k \log N_k \leq \log R_k + 2 \log N_k + 2 \log F_k. \quad (5)$$

379 The left part of (5) implies that  $(\theta_k - 1) \log N_k > \log R_k F_k$ . Because  $R_k F_k = \frac{2v-k}{v-k} \geq 1$   
 380 and  $N_k \geq 1$ , we find that  $1 < \theta_k$ . On the other hand, the right part of (5) also implies  
 381 that  $(\theta_k - 2) \log N_k \leq \log R_k + 2 \log F_k$ . We now claim that  $R_k F_k^2 \leq 1$  for all considered  
 382 values of  $k$ , and the lemma follows. This claim results from the observation that a)  $R_k F_k^2$  is  
 383 an increasing function of  $k$  over the considered range and b)  $R_s F_s^2 = 1$  (indeed, this is the  
 384 equation satisfied by the “optimal” value of  $s$  in the proof of lemma 8). Let  $\underline{k} = k/n$ ; we find  
 385 that

$$386 \quad \frac{d}{dk}(R_k F_k^2) = \frac{(k[1-3v] + 2v^2)(2v-k)}{(v-k)^2(1-k)^2}.$$

387 All factors are obviously positive, which shows that  $R_k F_k^2$  increases to 1 when  $k = s$  (and  
 388 therefore is less than one otherwise). ◀

389 ▶ **Lemma 11.** *Assume that  $N_s \geq 1$ . In algorithm 3, denote by  $I$  the total time spent in  
 390 inner nodes (i.e. filtering the lists) and by  $L$  the total time spent leaf nodes (in the quadratic  
 391 algorithm). Then  $I = \mathcal{O}(n^2 L)$ .*

392 **Proof.** In an inner node of depth  $k$ , filtering the lists costs  $N_k$ , while a recursive call costs  
 393 more than  $(F_k N_k)^{\theta_{k+1}}$  — indeed, this value ignores the cost of inner nodes below the current  
 394 one. In all cases,  $F_k \geq 1/n$  and  $\theta_{k+1} \leq 2$ , so that a recursive call costs more than  $n^{-2} N_k^{\theta_{k+1}}$ .  
 395 Because  $\theta_{k+1} > 1$ , we conclude that a recursive call costs more than  $n^{-2} N_k$ . This holds true  
 396 for all inner nodes, and the lemma follows. ◀

397 We now have all the ingredients to prove theorem 2. The proof is very similar to that of  
 398 theorem 1.

399 **Proof of theorem 2.** Let  $N_0$  denote the size of input lists containing a single 3XOR triplet in  
 400 expectation, and write  $\tilde{N}_0 = 2^{n\tilde{N}_0}$ . From theorem 4, we know that  $\tilde{N}_0 = H(2v) - H(3v)/3 -$   
 401  $\underline{v} \log_2 3$ .

402 Write  $e = 2 + (\tilde{R} + 2\tilde{F})/\tilde{N}_0$ . The value given in the statement of the theorem is the result  
 403 of many simplifications of this expression, using the expressions of  $\tilde{R}$  and  $\tilde{F}$  from the proof  
 404 of lemma 8.

405 Let  $\epsilon$  be such that  $d = 2 + (\tilde{R} + 2\tilde{F})/(\tilde{N}_0 + \epsilon)$ . Such an  $\epsilon$  always exist, because this is an  
 406 increasing function of  $\epsilon$ .

407 Write  $N = 2^{n\tilde{N}}$  the size of the input lists. From lemma 11, we know that the running  
 408 time of algorithm 3 is upper bounded by  $n^4 2^{n(\tilde{R} + 2\tilde{F} + 2\tilde{N})}$ .

409 Assume that  $\tilde{N} \leq \tilde{N}_0 + \epsilon$ . Use algorithm 3 with  $s = v(n-4v)/(n-3v)$  as in lemma 8.  
 410 Lemma 7 guarantees the success probability. We claim that its expected running time is less  
 411 than  $n^4 N^d$ . Indeed, it is  $n^4 2^{n[\tilde{R} + 2\tilde{F} + 2\tilde{N}]}$  which we can rewrite as  $n^4 N^{2 + [\tilde{R} + 2\tilde{F}]/\tilde{N}}$ . Because  
 412  $\tilde{R} + 2\tilde{F}$  is negative, this is less than  $n^4 N^{2 + [\tilde{R} + 2\tilde{F}]/(\tilde{N}_0 + \epsilon)}$  and  $d$  was chosen so that this is  
 413  $n^4 N^d$ .

414 The case where  $\tilde{N} > \tilde{N}_0 + \epsilon$  can be dealt with using the technique highlighted in  
 415 section 3.1. ◀

416 **Acknowledgements** We thank Pierre-Alain Fouque, Anand Kumar Narayanan and Amandine  
 417 Véber for useful discussions. We are very grateful to 3 out of 9 anonymous reviewers (so far)  
 418 for rejecting two previous versions of this paper while providing extremely helpful feedback  
 419 and suggesting new ideas.

## 420 — References —

- 421 **1** R. Arratia and L. Gordon. Tutorial on large deviations for the binomial distribu-  
422 tion. *Bulletin of Mathematical Biology*, 51(1):125 – 131, 1989. URL: <http://www.sciencedirect.com/science/article/pii/S0092824089800527>, doi:[https://doi.org/10.1016/S0092-8240\(89\)80052-7](https://doi.org/10.1016/S0092-8240(89)80052-7).  
423  
424
- 425 **2** K.B. Athreya and PE Ney. *Branching processes*. Dover Publications, 2004.
- 426 **3** Leif Both and Alexander May. The approximate k-list problem. *IACR Transactions on*  
427 *Symmetric Cryptology*, 2017(1):380–397, 2017.
- 428 **4** Charles Bouillaguet, Claire Delaplace, and Pierre-Alain Fouque. Revisiting and improving  
429 algorithms for the 3xor problem. *IACR Transactions on Symmetric Cryptology*, 2018(1):254–  
430 276, 2018.
- 431 **5** Martin Dietzfelbinger, Philipp Schlag, and Stefan Walzer. A subquadratic algorithm for 3xor.  
432 In Igor Potapov, Paul G. Spirakis, and James Worrell, editors, *43rd International Symposium*  
433 *on Mathematical Foundations of Computer Science, MFCS 2018, August 27-31, 2018, Liverpool,*  
434 *UK*, volume 117 of *LIPICs*, pages 59:1–59:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik,  
435 2018. doi:10.4230/LIPICs.MFCS.2018.59.
- 436 **6** William Feller. *An introduction to probability theory and its applications. Vol. I*. Third edition.  
437 John Wiley & Sons Inc., New York, 1968.
- 438 **7** Michael L. Fredman, János Komlós, and Endre Szemerédi. Storing a sparse table with  $o(1)$   
439 worst case access time. *J. ACM*, 31(3):538–544, June 1984. URL: <http://doi.acm.org/10.1145/828.1884>, doi:10.1145/828.1884.  
440
- 441 **8** Anka Gajentaan and Mark Overmars. On a class of  $\mathcal{O}(n^2)$  problems in computational geometry.  
442 *Computational geometry*, 5(3):165–185, 1995.
- 443 **9** Zahra Jafargholi and Emanuele Viola. 3sum, 3xor, triangles. *Algorithmica*, 74(1):326–343,  
444 2016. doi:10.1007/s00453-014-9946-9.
- 445 **10** Antoine Joux. *Algorithmic cryptanalysis*. CRC Press, 2009.
- 446 **11** Gaëtan Leurent and Ferdinand Sibleyras. Low-memory attacks against two-round even-mansour  
447 using the 3XOR problem. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances*  
448 *in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa*  
449 *Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II*, volume 11693 of *Lecture Notes*  
450 *in Computer Science*, pages 210–235. Springer, 2019. doi:10.1007/978-3-030-26951-7\_8.
- 451 **12** Alexander May and Ilya Ozerov. On computing nearest neighbors with applications to  
452 decoding of binary linear codes. In Elisabeth Oswald and Marc Fischlin, editors, *Advances*  
453 *in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory*  
454 *and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings,*  
455 *Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 203–228. Springer, 2015.  
456 doi:10.1007/978-3-662-46800-5\_9.
- 457 **13** Mridul Nandi. Revisiting Security Claims of XLS and COPA. *IACR Cryptology ePrint Archive*,  
458 2015:444, 2015.
- 459 **14** Ivica Nikolić and Yu Sasaki. Refinements of the k-tree Algorithm for the Generalized Birthday  
460 Problem. In *ASIACRYPT*, pages 683–703. Springer, 2015.
- 461 **15** Rasmus Pagh and Flemming Friche Rodler. Cuckoo hashing. In *European Symposium on*  
462 *Algorithms*, pages 121–133. Springer, 2001.
- 463 **16** E. Prange. The use of information sets in decoding cyclic codes. *IRE Transactions on*  
464 *Information Theory*, 8(5):5–9, 1962. doi:10.1109/TIT.1962.1057777.
- 465 **17** S.M. Ross. *Probability Models for Computer Science*. Elsevier Science, 2002. URL: <https://books.google.fr/books?id=fG3iEZ8f3CcC>.  
466

## 467 **A** A “Sensible” Application

468 In order to put algorithm 2 to the test, we forged an artificial instance of the problem. We  
 469 downloaded an XML dump of the DBLP database, and extracted all articles published in a  
 470 few selected cryptography conferences (CRYPTO, EUROCRYPT, ASIACRYPT, FSE, PKC,  
 471 CHES, SAC) as well as two journals (TOSC, TCHES). This made more than 7700 articles.  
 472 For each article, we wrote down one line of text with the authors and title.

473 We considered the function (where  $\&$  denotes the bitwise AND):

$$474 \quad F(x_1, x_2, x_3, x_4) = \text{SHA-512}(x_1) \& \text{SHA-512}(x_2) \& \text{SHA-512}(x_3) \& \text{SHA-512}(x_4).$$

475 This yields 512-bit hashes with expected density  $1/16$ . SHA-512 is a secure cryptographic  
 476 hash function, so we assumed that there was no way to find inputs leading to correlated  
 477 outputs besides brute force. We looked for three quadruplets of articles such that

$$478 \quad F(x_1, x_2, x_3, x_4) \oplus F(y_1, y_2, y_3, y_4) \oplus F(z_1, z_2, z_3, z_4) = 0$$

479 With the additional constraint that all articles are distinct. There are 5775 ways to dispatch  
 480 12 items into 4 indistinguishable urns, so that with our 7700 articles, we can assemble  $2^{138.5}$   
 481 bundles of 12 publications having a chance to satisfy the above equation (of course the inputs  
 482 are correlated, and this deviates from the original problem formulation, but this is not a  
 483 serious issue). Alternatively, we may form  $2^{47}$  quadruplets of publication. With  $p = 1/16$   
 484 and  $n = 512$ , we could then expect about 40 solutions from our data set. This made us  
 485 confident that there would be at least one, but finding it does not seem so easy at first  
 486 glance.

487 Evaluating  $F$  on all the available quadruplets is not a problem (it takes 240 CPU-hours).  
 488 Trouble starts when we considered writing the list of  $2^{47}$  hashes to persistent storage: this  
 489 would require more than 9 petabytes — this is a lot, but some computing centers have that  
 490 much. However, finding the “golden” triplet of quadruplets using the quadratic algorithm  
 491 would then require  $2^{94}$  probes into a large hash table, and given mankind’s present computing  
 492 power, this does not seem feasible before the sun turns into a red giant.

493 Exploiting the sparsity of the input turns the problem into a walk in the park. The  
 494 expected weight of 3XOR triplets of density  $1/16$  is  $\approx 2.25$ . We evaluated  $F$  on all quadruplets,  
 495 but kept only the hashes with hamming weight less than or equal to 3. We thus kept 5091  
 496 candidate quadruplets, for a total storage size of 358KB. We then searched for solutions in  
 497 this restricted data set using the quadratic algorithm. This required 25 millions probes in a  
 498 hash table and was very fast.

499 We found six solutions, one of which is shown as algorithm 4. Amusingly, it contains the  
 500 name of one of the authors of the present article.

## 501 **B** Omitted Proofs

502 **Proof of lemma 3.** Let  $X(i, j, k)$  denote the binary random variable that takes the value 1  
 503 if and only if  $A[i] + B[j] + C[k] = 0$  (and zero otherwise), so that  $Y = \sum X(i, j, k)$ . Unless  
 504 mentioned otherwise, in this section all sums are taken over  $0 \leq i, j, k < N$ ; we omit the  
 505 indices to alleviate notations.

506 The expected value of  $Y$  is easy to determine. Because the elements of the lists are  
 507 identically distributed,  $\Pr(A[i] + B[j] + C[k] = 0)$  is independent of  $i, j$  and  $k$  and its value  
 508 is  $\rho$ . We get:

$$509 \quad \mathbb{E}Y = \mathbb{E} \sum X(i, j, k) = \sum \mathbb{E}X(i, j, k) = \sum \Pr(A[i] + B[j] + C[k] = 0) = N^3\rho.$$

**Algorithm 4** Demonstrating a sensible application of sparse 3XOR algorithms.

---

```

from hashlib import sha512

# FSE 2011
a = "Simon Knellwolf and Willi Meier: Cryptanalysis of the Knapsack Generator. (2011)"

# ASIACRYPT 2017
b = "Ran Canetti, Oxana Poburinnaya and Mariana Raykova: Optimal-Rate Non-Committing Encryption. (2017)"

# CRYPTO 2019
c = "Muhammed F. Esgin, Ron Steinfeld, Joseph K. Liu and Dongxi Liu: Lattice-Based Zero-Knowledge \
Proofs: New Techniques for Shorter and Faster Constructions and Applications. (2019)"

# FSE 2009
d = "Martijn Stam: Blockcipher-Based Hashing Revisited. (2009)"

# EUROCRYPT 1990
e = "Cees J. A. Jansen: On the Construction of Run Permuted Sequences. (1990)"

# EUROCRYPT 2013
f = "Charles Bouillaguet, Pierre-Alain Fouque and Amandine Véber: Graph-Theoretic Algorithms for the \
"Isomorphism of Polynomials" Problem. (2013)"

# CRYPTO 2017
g = "Prabhanjan Ananth, Arka Rai Choudhuri and Abhishek Jain: A New Approach to Round-Optimal Secure \
Multiparty Computation. (2017)"

# EUROCRYPT 2001
h = "William Aiello, Yuval Ishai and Omer Reingold: Priced Oblivious Transfer: How to Sell Digital \
Goods. (2001)"

# CRYPTO 2019
i = "Navid Alamati, Hart Montgomery and Sikhar Patranabis: Symmetric Primitives with Structured \
Secrets. (2019)"

# CRYPTO 2019
j = "Shweta Agrawal, Monosij Maitra and Shota Yamada: Attribute Based Encryption (and more) for \
Nondeterministic Finite Automata from LWE. (2019)"

# EUROCRYPT 1986
k = "Christoph G. Günther: On Some Properties of the Sum of Two Pseudorandom Sequences. (1986)"

# CRYPTO 2009
l = "Susan Hohenberger and Brent Waters: Short and Stateless Signatures from the RSA Assumption. (2009)"

def H(s : str) -> int:
    """Returns the hash (SHA-512) of the string s as a 512-bit integer."""
    return int.from_bytes(sha512(s.encode('utf8')).digest(), byteorder='big')

assert (H(a) & H(b) & H(c) & H(d)) ^ (H(e) & H(f) & H(g) & H(h)) ^ (H(i) & H(j) & H(k) & H(l)) == 0

```

---

510 The Markov bound yields  $1 - EY \leq \Pr(Y = 0)$ . Because  $Y$  is the sum of binary random  
511 variables, we are entitled to use Ross's conditional expectation inequality [17]:

$$512 \Pr(Y > 0) \geq \sum \frac{E(X(i, j, k))}{E(Y | X(i, j, k) = 1)}.$$

513 As argued above, the value of the term under the sum is independent of  $i, j$  and  $k$ , so this  
514 boils down to:  $\Pr(Y > 0) \geq EY/E(Y | X(0, 0, 0) = 1)$ . It remains to compute the expected  
515 number of solutions knowing that there is at least one. This yields:

$$516 E(Y | X(0, 0, 0) = 1) = \sum \Pr(A[i] + B[j] + C[k] = 0 | A[0] + B[0] + C[0] = 0)$$

518 We split this sum in 8 parts by considering separately the situation where  $i = 0$ ,  
519  $j = 0$  and  $k = 0$  (resp  $\neq 0$  for each summation index). We introduce the shorthand  
520  $p_{ijk} = \Pr(A[i] + B[j] + C[k] = 0 | A[0] + B[0] + C[0] = 0)$  and we assume that  $i, j, k > 0$ .  
521 Then the two events are in fact independent and  $p_{ijk} = \rho$ . But when at least one index is



522 zero, this is no longer the case ; the extreme situation is  $p_{000} = 1$ . By symmetry between  
 523 the three input lists, we find that  $p_{ij0} = p_{i0k} = p_{0jk}$  (this is the value we denote by  $\sigma$ ) and  
 524  $p_{i00} = p_{0j0} = p_{00k}$  (this is the value we denote by  $\tau$ ). We can now write:

$$\begin{aligned} 525 \quad \mathbb{E}(Y \mid X(0,0,0) = 1) &= (N-1)^3 \rho + 3(N-1)^2 \sigma + 3(N-1)\tau + 1 \\ 526 \quad &= N^3 \rho + 3N^2 \sigma + 3N\tau + 1 - \Delta \\ 527 \quad &\text{with} \quad \Delta = (3N^2 - 3N + 1)\rho + 3(2N-1)\sigma + 3\tau \\ 528 \end{aligned}$$

529 The “error term”  $\Delta$  is always positive for  $N \geq 1$ . Going back to the beginning, we have:

$$530 \quad \Pr(Y > 0) \geq \frac{N^3 \rho}{N^3 \rho + 3N^2 \sigma + 3N\tau + 1 - \Delta} \geq \frac{1}{1 + 3N^{-1} \sigma / \rho + 3N^{-2} \tau / \rho + N^{-3} / \rho}$$

532 Using the convexity of  $1/(1+x)$ , we obtain  $\Pr(Y = 0) \leq 3N^{-1} \sigma / \rho + 3N^{-2} \tau / \rho + N^{-3} / \rho$ . ◀

533 **Low-Density 3XOR.** We now specialize the result of lemma 3 to the group  $(\{0,1\}^n, \oplus)$  with  
 534 the low-density distribution  $\mathcal{D}$  (each bit is drawn independently at random according to the  
 535 Bernoulli distribution  $Ber_p$  of parameter  $p < 1/2$ ). This the low-density case of theorem 4.

536 If  $a, b$  and  $c$  are random bits drawn according to  $Ber_p$ , then the probability that they  
 537 XOR to zero is  $(1-p)(1-2p+4p^2)$ . It follows that if  $\mathbf{x}, \mathbf{y}$  and  $\mathbf{z}$  are drawn according to  $\mathcal{D}$ ,  
 538 then  $\rho = \Pr(\mathbf{x} \oplus \mathbf{y} \oplus \mathbf{z} = 0) = (1-p)^n (1-2p+4p^2)^n$ .

539 Let us compute  $\sigma = \Pr(\mathbf{x} \oplus \mathbf{y} = \mathbf{z} \mid \mathbf{u} \oplus \mathbf{v} = \mathbf{z})$ . What happens essentially depends on  
 540 the hamming weight of  $\mathbf{z}$ . Both  $\mathbf{x}, \mathbf{y}$  and  $\mathbf{u}, \mathbf{v}$  have to XOR to  $\mathbf{z}$ . Two random bits drawn  
 541 according to  $Ber_p$  XOR to zero with probability  $p^2 + (1-p)^2$  and their XOR to one with  
 542 probability  $2p(1-p)$ . This yields (using the binomial theorem):

$$\begin{aligned} 543 \quad \rho \sigma &= \Pr(\mathbf{x} \oplus \mathbf{y} = \mathbf{z} \wedge \mathbf{u} \oplus \mathbf{v} = \mathbf{z}) \\ 544 \quad &= \sum_{k=0}^n \Pr(wt(\mathbf{z}) = k) \Pr(\mathbf{x} \oplus \mathbf{y} = \mathbf{z} \wedge \mathbf{u} \oplus \mathbf{v} = \mathbf{z} \mid wt(\mathbf{z}) = k) \\ 545 \quad &= \sum_{k=0}^n \binom{n}{k} p^k (1-p)^{n-k} [2p(1-p)]^{2k} [p^2 + (1-p)^2]^{2(n-k)} \\ 546 \quad &= [(1-p)(1-4p+8p^2-4p^3)]^n \\ 547 \end{aligned}$$

548 We move on to  $\tau = \Pr(\mathbf{x} = \mathbf{u} \oplus \mathbf{v} \mid \mathbf{z} = \mathbf{u} \oplus \mathbf{v})$ . In this context, this mostly depends on  
 549 the hamming weight of  $\mathbf{u} \oplus \mathbf{v}$ . This yields (again using the binomial theorem):

$$\begin{aligned} 550 \quad \rho \tau &= \Pr(\mathbf{x} = \mathbf{u} \oplus \mathbf{v} \wedge \mathbf{z} = \mathbf{u} \oplus \mathbf{v}) \\ 551 \quad &= \sum_{k=0}^n \Pr(wt(\mathbf{u} \oplus \mathbf{v}) = k) \Pr(\mathbf{x} = \mathbf{u} \oplus \mathbf{v} \wedge \mathbf{z} = \mathbf{u} \oplus \mathbf{v} \mid wt(\mathbf{u} \oplus \mathbf{v}) = k) \\ 552 \quad &= \sum_{k=0}^n \binom{n}{k} [2p(1-p)]^k [p^2 + (1-p)^2]^{n-k} [p^k (1-p)^{n-k}]^2 \\ 553 \quad &= [(1-p)(1-3p+4p^2)]^n \\ 554 \end{aligned}$$

555 We now move on to establish (4). Because  $N = (\mathbb{E}Y)^{1/3} / \rho$ , the bound of lemma 3 can  
 556 be rewritten as:

$$557 \quad \Pr(Y = 0) \leq \frac{3}{(\mathbb{E}Y)^{1/3}} \frac{\sigma}{\rho^{2/3}} + \frac{3}{(\mathbb{E}Y)^{2/3}} \frac{\tau}{\rho^{1/3}} + \frac{1}{\mathbb{E}Y}.$$

558 We now claim that  $1/2 \leq \sigma^{3/n}/\rho^{2/n} \leq 1$  and  $1/4 \leq \tau^{3/n}/\rho^{1/n} \leq 1$  when  $0 \leq p \leq 1/2$ ; this  
 559 yields the desired result. This claim follows from the facts that both values are decreasing  
 560 functions of  $p$ . This can be seen by computing their derivatives (all factors are easily seen to  
 561 be positive when  $0 \leq p \leq 1/2$ ):

$$562 \quad \frac{\partial \sigma^{3/n}}{\partial p \rho^{2/n}} = -6 \frac{(4p^3 - 8p^2 + 4p - 1)^2 (2p^2 - 6p + 3)(1 - 2p)^2 p}{(4p^2 - 2p + 1)^6 (1 - p)^3}$$

$$563 \quad \frac{\partial \tau^{3/n}}{\partial p \rho^{1/n}} = -6 \frac{(4p^2 - 3p + 1)^2 (4p^2 - 6p + 3)(1 - 2p)p}{(4p^2 - 2p + 1)^5 (1 - p)^2}$$

565 **Low-Weight 3XOR** We finally focus on the case where the elements of the lists are sampled  
 566 uniformly at random amongst bit strings of hamming weight  $w = np$ . 3XOR triplets only exist  
 567 if  $w$  is even and less than  $2n/3$ , so that it seems fitting to define  $w = 2v$ , with  $0 \leq v \leq 1/3$ .

568 There are  $A := \binom{n}{3v} \binom{3v}{v} \binom{2v}{v}$  3XOR triplets of weight  $w$  and there are  $B := \binom{n}{2v}^3$  triplets  
 569 in total. It follows that :

$$570 \quad \rho = \frac{\binom{n}{3v} \binom{3v}{v} \binom{2v}{v}}{\binom{n}{2v}^3}$$

572 Using (1), we may simplify this expression. It turns out that using the relative weight  
 573  $\underline{v} = v/n$  is more convenient:

$$574 \quad \left(\frac{\pi}{4}\right)^{3/2} \leq \frac{\rho}{\tilde{\rho}} \leq \left(\frac{4}{\pi}\right)^{3/2} \quad \text{with} \quad \tilde{\rho} = 8 \left(\frac{(1-2\underline{v})^3}{1-3\underline{v}}\right)^{1/2} 2^{n[3\underline{v} \log_2 3 + H(3\underline{v}) - 3H(2\underline{v})]}$$

575 Next,  $\sigma = \Pr(x \oplus y = z \mid u \oplus v = z)$  can be determined by observing that this amounts to  
 576 count the number of pairs  $(x, y)$  such that  $x \oplus y$  gives a fixed weight- $2v$  bit string. It follows  
 577 that:

$$578 \quad \sigma = \frac{\binom{2v}{v} \binom{n-2v}{v}}{\binom{n}{2v}^2}$$

580 Using (1) again, we obtain:

$$581 \quad \frac{\pi}{4} \leq \frac{\sigma}{\tilde{\sigma}} \leq \frac{4}{\pi} \quad \text{with} \quad \tilde{\sigma} = 2\sqrt{2(1-2\underline{v})(1-\underline{v})} 2^{n[(1-2\underline{v})H(\underline{v}/(1-2\underline{v})) - 2H(2\underline{v}) + 2\underline{v}]}$$

582 Lastly,  $\tau = \Pr(x = y \oplus z \mid u = y \oplus z)$  is just the probability that sampling randomly  
 583 yields a fixed weight- $2v$  bit string. Therefore  $\tau = \binom{n}{2v}^{-1}$ , and

$$584 \quad \frac{\pi}{4} \leq \frac{\tau}{\tilde{\tau}} \leq \frac{4}{\pi} \quad \text{with} \quad \tilde{\tau} = 1/\sqrt{2n\underline{v}(1-2\underline{v})} 2^{n-H(2\underline{v})}$$

585 We again investigate  $\sigma^3/\rho^2$  and  $\tau/\rho^3$ , but this time we focus on the exponents. Define  
 586  $f(\underline{v}) \approx \frac{1}{n} \log_2(\sigma^3/\rho^2)$  and  $g(\underline{v}) \approx \frac{1}{n} \log_2(\tau^3/\rho)$ . We claim that both functions are negative  
 587 when  $0 \leq \underline{v} \leq 1/3$ , and this again entails (4).

588 We claim that  $f$  is decreasing over this range. To see why, compute its second derivative:

$$589 \quad \frac{\partial^2 f}{\partial \underline{v}^2} = \frac{3}{\ln 2} \frac{1-4\underline{v}}{(1-3\underline{v})(1-2\underline{v})v}.$$

590 It is positive when  $\underline{v} \leq 1/4$  and negative afterwards. This means that the first derivative of  
 591  $f$  reaches a maximum at  $\underline{v} = 1/4$ . It is then easy to check that  $\frac{\partial f}{\partial \underline{v}}(1/4) = 0$ .

## 18 Algorithms for the Sparse Random 3XOR Problem

592 We next claim that  $g$  is decreasing over  $[0; 1/4]$  and increasing over  $[1/4; 1/3]$ . To see  
593 why, just compute its derivative:

$$594 \quad \frac{\partial g}{\partial v} = -3 \log_2(1 - 3v)/v.$$

595 We then find that  $g$  reaches a limit of zero when  $v$  goes to zero, and that  $g(1/3) = -\log_2 3$ .  
596 This guarantees that  $g$  is negative over the full range.