



HAL
open science

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

Charles Bouillaguet, Claire Delaplace

► **To cite this version:**

Charles Bouillaguet, Claire Delaplace. Algorithms for Sparse Random 3XOR: The Low-Density Case. 2021. hal-02306917v3

HAL Id: hal-02306917

<https://hal.science/hal-02306917v3>

Preprint submitted on 1 Mar 2021 (v3), last revised 2 Oct 2021 (v4)

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.

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

Charles Bouillaguet 

Sorbonne Université, CNRS, LIP6, F-75005 Paris, France
charles.bouillaguet@lip6.fr

Claire Delaplace

MIS Laboratory, Université de Picardie Jules Verne, 14 quai de la Somme, 80080 Amiens, France
claire.delaplace@u-picardie.fr

Abstract

We present an algorithm for a variant of the 3XOR problem with lists consisting of n -bit vectors whose coefficients are drawn independently at random according to a Bernoulli distribution of parameter $p < 1/2$. We show that in this particular context the problem can be solved much more efficiently than in the general setting. This leads to a linear algorithm with overwhelming success probability for $p < 0.0957$. With slightly different parameters, this method succeeds deterministically. The expected runtime is also linear for $p < 0.02155$ and always sub-quadratic.

2012 ACM Subject Classification Theory of computation → Computational complexity and cryptography; Theory of computation

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

1 Introduction

Given three lists A , B and C of n -bit vectors, 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 represent 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 time $\mathcal{O}(N^{2-\epsilon})$ for any $\epsilon > 0$. Dietzfelbinger, Schlag and Walzer found an algorithm that gain a poly-logarithmic factor over the quadratic algorithm [4].

3XOR can be seen as a variant of the celebrated 3SUM problem, where this time the input list items are seen as integers and we must have $x + y + z = 0$. Many geometric problems can be reduced to 3SUM in sub-quadratic time, and those problems are said to be 3SUM hard [6]. 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 [7]. In the other direction, some conditional lower-bounds have been established based on the hypothesis that 3XOR is inherently quadratic: [4] 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 cryptographic applications, in which the input lists consist of uniformly *random* vectors. In particular, we can mention Nandi's attack [10] 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 [9]. May and Both have been considering a variant of the 3XOR problem, the *approximate 3-list birthday problem*: given three lists of N uniformly random elements of $\{0, 1\}^n$ the goal consist in finding 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 [2].

46 Studying the 3XOR problem with uniformly random inputs seemed natural to the
 47 cryptography community (which makes this assumption “by default”), and this led to
 48 several algorithms gaining a polylogarithmic factor, predating [4], and often tailored to
 49 specific input sizes [8, 11, 3]. The assumption that the input is *random* makes the problem
 50 simpler, but it has not yet been possible to obtain a $\mathcal{O}(N^{2-\epsilon})$ algorithm even in this simpler
 51 case.

52 **Contributions.** In this paper, we consider an even more favorable setting, where the input
 53 lists are both *random* and *sparse*, namely where the input bitstrings are biased towards
 54 zero in some way. More precisely, we assume that each input bit is drawn independently
 55 at random according to a Bernoulli distribution of parameter $0 < p < 1/2$ — the “dense”
 56 random case corresponds to $p = 1/2$.

57 We first give the probability that the input actually contains a 3XOR triplet for a given
 58 list size N and density p . To the best of our knowledge, this result was not readily available
 59 from the existing literature, not even in the simple “cryptographically relevant” case where
 60 $p = 1/2$. We then describe a new algorithm to solve the random low-density 3XOR problem
 61 (Section 3). The main idea is to discard useless input vectors (i.e. whose Hamming weight is
 62 above a well-chosen threshold), then search a solution using the simple quadratic algorithm.
 63 This algorithm returns a solution with overwhelming probability in time linear in N for
 64 small density (i.e. $p < 0.0957$). A slight variation of the same algorithm can also be used to
 65 deterministically return a solution if there is one in the input. In this case, the expected
 66 running time of the procedure is also linear in N when p is smaller than 0.02155. The
 67 algorithm has a time complexity of $\mathcal{O}(N + N^e)$, for some parameter $e < 2$ when $p < 1/2$.
 68 The evolution of this parameter e in function of p is shown in Figure 1.

69 Another relevant sparse distribution would be the “low-weight” distribution, where
 70 bitstrings from $\{0, 1\}^n$ of a given small weight are drawn uniformly at random. For lack of
 71 space, we do not consider this (quite different) problem in this paper. Our main contribution,
 72 the algorithm described in section 4, cannot be adapted to this other setting.

73 The algorithm presented in this paper has no concrete application that we are aware of,
 74 either in cryptography or elsewhere. We nevertheless demonstrate its practical efficiency by
 75 obtaining the —non-trivial and “sensible”— result shown in Section 5.

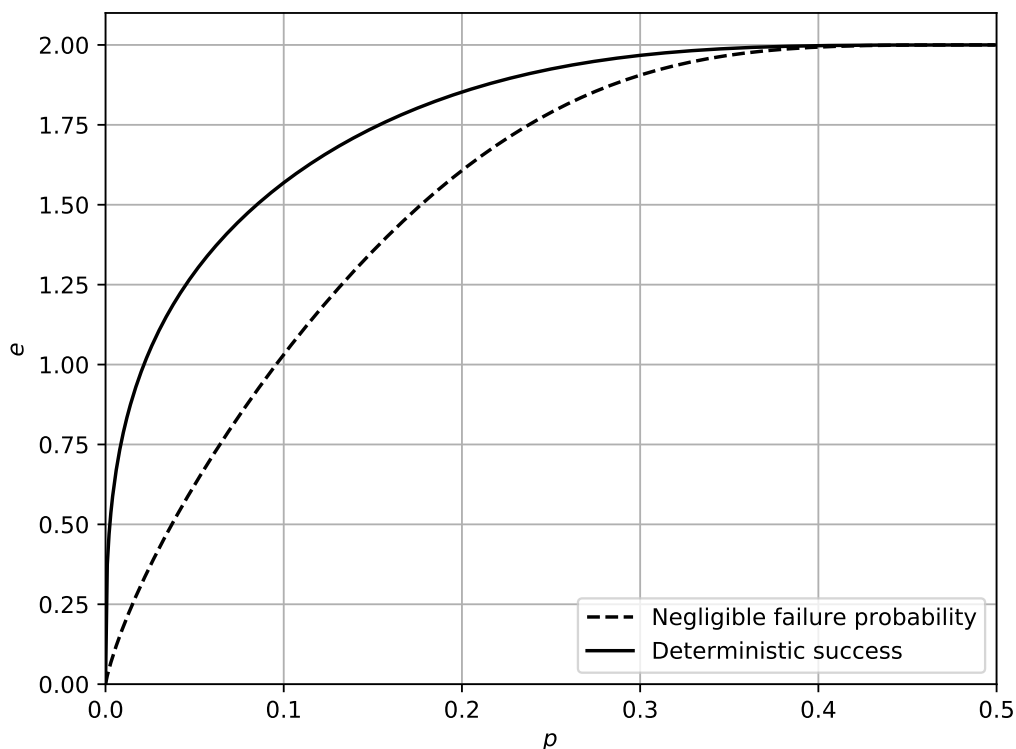
76 2 Preliminaries

77 Let $\mathbf{x} = x_0x_1 \dots x_{n-1}$ be an n -bit string (we use “bit string” and “vector” as synonyms). We
 78 denote by $\text{wt}(\mathbf{x})$ its Hamming weight. Let A be a list ; $|A|$ denotes the number of elements in
 79 A and $A[i]$ denotes the i -th element. We say that $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ is a *3XOR triplet* if $\mathbf{x} \oplus \mathbf{y} \oplus \mathbf{z} = \mathbf{0}$.

80 We consider random instances of the problem defined as follows. Let Ψ be some probability
 81 distribution over $\{0, 1\}^n$. Let A, B and C be three lists of N elements drawn independently at
 82 random according to Ψ . A *solution* to the instance of the 3XOR problem given by (A, B, C)
 83 is a 3XOR triplet $(\mathbf{x}, \mathbf{y}, \mathbf{z}) \in A \times B \times C$.

84 Let $0 < p < 1/2$ be fixed and let Ber_p the Bernoulli distribution of parameter p . In the
 85 sequel, we focus on the sparse “low-density” distributions over $\{0, 1\}^n$, denoted as \mathcal{D} , where
 86 each input bit is drawn independently from Ber_p .

87 **Bounds for Binomial Distributions.** Let $\mathcal{B}(n, p)$ denote the binomial distribution with
 88 parameters n, p . Let $X \sim \mathcal{B}(n, p)$ be a binomial random variable. We make heavy use of tail
 89 bounds, notably the Chernoff bound (1) and the tighter classical inequality (2), a proof of



■ **Figure 1** The INCREMENTAL3XOR algorithm of section 4 run in time $\mathcal{O}(N + N^e)$ where e is shown here. With $w_{\max} = 2nu$ (cf. theorem 3), this yields the “negligible failure probability” curve. With $w_{\max} = n$ (theorem 6), this yields the “deterministic success” curve.

90 which can be found in [1] amongst others.

$$91 \quad \Pr(X \leq an) \leq \exp -\frac{n}{2p}(p-a)^2, \quad \text{if } \frac{k}{n} < p \quad (1)$$

$$92 \quad \Pr(X \leq an) \leq \exp -nD(a,p) \quad \text{if } a < p, \quad (2)$$

$$93 \quad \Pr(X \geq an) \leq \exp -nD(a,p) \quad \text{if } a > p,$$

94 where $D(a,p) = a \ln \frac{a}{p} + (1-a) \ln \frac{1-a}{1-p}$ is the Kullback-Leibler divergence between an a -coin
95 and a p -coin.

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

104 **The Quadratic Algorithm.** The simplest possible way to solve the 3XOR problem is the
105 quadratic algorithm (shown as algorithm 1). For all pairs $(x,y) \in A \times B$, check if $x \oplus y$
106 belongs to C . Each test can be done in constant time if all entries of C have been stored in a

107 suitable data structure beforehand — for instance one could use the optimal static dictionary
 108 of [5], or simply cuckoo hashing [12].

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

```

109 The initialization of the dictionary holding C is separated from the rest, because we will
 110 subsequently invoke QUADRATIC3XOR many times with the same C . We assume (using [5]
 111 for instance) that QUADRATICSETUP takes time $\mathcal{O}(|C|)$. Then QUADRATIC3XOR(A, B, \mathcal{C})
 112 takes time $\mathcal{O}(|A| \times |B|)$. The quadratic algorithm works regardless of the sparsity of the
 113 input, and as such it does not take advantage of it. It is the only solution, up to logarithmic
 114 factors, when the input is dense. In our case, this is the baseline that must be improved
 115 upon.

116 **3** Bounds on the Existence of 3XOR Triplets

117 Suppose that $(G, +)$ is a group (in additive notation) and assume that three lists A, B and
 118 C , each of size N , are made of group elements sampled independently at random according
 119 to some distribution Ψ . Let Y be the random variable that counts the number of triplets
 120 $(x, y, z) \in A \times B \times C$ such that $x + y + z = 0$ (taken over the random choice of A, B and C).
 121 The expected value of Y is easy to compute, and because the average value is bounded away
 122 from zero, a concentration bound would yield the probability that the input lists contain at
 123 least a single “good triplet” (that sums to zero).

124 The problem is that the N^3 triplets in $A \times B \times C$ are not independent and thus classical
 125 techniques such as the Chernoff bound do not apply. This dependence has to be accounted
 126 for. Let x, y, z, u, v denote five independent random variable distributed according to Ψ , and
 127 set

$$\begin{aligned}
 128 \quad \rho &= \Pr(x + y + z = 0) \\
 129 \quad \sigma &= \Pr(x + y + z = 0 \mid u + v + z = 0) \\
 130 \quad \tau &= \Pr(x + y + z = 0 \mid u + y + z = 0) \\
 131
 \end{aligned}$$

132 The following result can be specialized for any group G and any distribution Ψ by computing
 133 ρ, σ and τ .

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

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

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

$$142 \quad \mathbb{E}Y = \mathbb{E} \sum X(i, j, k) = \sum \mathbb{E}X(i, j, k) = \sum \Pr(A[i] + B[j] + C[k] = 0) = N^3\rho.$$

143 The Markov bound yields $1 - \mathbb{E}Y \leq \Pr(Y = 0)$; the less immediate part consists in
 144 estimating $\Pr(Y > 0)$. Because Y is the sum of binary random variables, we are entitled to
 145 use Ross's conditional expectation inequality [13]:

$$146 \quad \Pr(Y > 0) \geq \sum \frac{\mathbb{E}(X(i, j, k))}{\mathbb{E}(Y \mid X(i, j, k) = 1)}.$$

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

$$150 \quad \mathbb{E}(Y \mid X(0, 0, 0) = 1) = \sum \Pr(A[i] + B[j] + C[k] = 0 \mid A[0] + B[0] + C[0] = 0)$$

152 We split this sum in 8 parts by considering separately the situation where $i = 0$,
 153 $j = 0$ and $k = 0$ (resp $\neq 0$ for each summation index). We introduce the shorthand
 154 $p_{ijk} = \Pr(A[i] + B[j] + C[k] = 0 \mid A[0] + B[0] + C[0] = 0)$ and we assume that $i, j, k > 0$.
 155 Then the two events are in fact independent and $p_{ijk} = \rho$. But when at least one index is
 156 zero, this is no longer the case; the extreme situation is $p_{000} = 1$. By symmetry between
 157 the three input lists, we find that $p_{i00} = p_{i0k} = p_{0jk}$ (this is the value we denote by σ) and
 158 $p_{i00} = p_{0j0} = p_{00k}$ (this is the value we denote by τ). We can now write:

$$159 \quad \begin{aligned} \mathbb{E}(Y \mid X(0, 0, 0) = 1) &= (N-1)^3\rho + 3(N-1)^2\sigma + 3(N-1)\tau + 1 \\ &= N^3\rho + 3N^2\sigma + 3N\tau + 1 - \Delta \\ 161 \quad \text{with} \quad \Delta &= (3N^2 - 3N + 1)\rho + 3(2N-1)\sigma + 3\tau \end{aligned}$$

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

$$164 \quad \Pr(Y > 0) \geq \frac{N^3\rho}{N^3\rho + 3N^2\sigma + 3N\tau + 1 - \Delta} \geq \frac{1}{1 + 3N^{-1}\sigma/\rho + 3N^{-2}\tau/\rho + N^{-3}/\rho}$$

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

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

170 ▶ **Theorem 2.** *Let Y denote the random variable counting the number of 3XOR triplets
 171 in $A \times B \times C$. The probability that a random triplet from $A \times B \times C$ XORs to zero is
 172 $\rho = (1-p)^n(1-2p+4p^2)^n$, and their expected number is $\mathbb{E}Y = N^3\rho$. Furthermore:*

$$173 \quad 1 - \mathbb{E}Y \leq \Pr(Y = 0) \leq \frac{3}{(\mathbb{E}Y)^{1/3}} + \frac{3}{(\mathbb{E}Y)^{2/3}} + \frac{1}{\mathbb{E}Y}. \quad (3)$$

174 **Proof.** If a, b and c are random bits drawn according to Ber_p , then the probability that they
 175 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} ,
 176 then $\rho = \Pr(\mathbf{x} \oplus \mathbf{y} \oplus \mathbf{z} = 0) = (1-p)^n(1-2p+4p^2)^n$.

177 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
 178 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
 179 according to Ber_p XOR to zero with probability $p^2 + (1-p)^2$ and their XOR to one with
 180 probability $2p(1-p)$. This yields (using the binomial theorem):

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

186 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
 187 the hamming weight of $\mathbf{u} \oplus \mathbf{v}$. This yields (again using the binomial theorem):

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

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

$$195 \quad \Pr(Y = 0) \leq \frac{3}{(\mathbb{E}Y)^{1/3}} \frac{\sigma}{\rho^{2/3}} + \frac{3}{(\mathbb{E}Y)^{2/3}} \frac{\tau}{\rho^{1/3}} + \frac{1}{\mathbb{E}Y}.$$

196 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
 197 yields the desired result. This claim follows from the facts that both values are decreasing
 198 functions of p . This can be seen by computing their derivatives (all factors are easily seen to
 199 be positive when $0 \leq p \leq 1/2$):

$$\begin{aligned}
 200 \quad \frac{\partial}{\partial p} \frac{\sigma^{3/n}}{\rho^{2/n}} &= -6 \frac{(4p^3 - 8p^2 + 4p - 1)^2 (2p^2 - 6p + 3)(1-2p)^2 p}{(4p^2 - 2p + 1)^6 (1-p)^3} \\
 201 \quad \frac{\partial}{\partial p} \frac{\tau^{3/n}}{\rho^{1/n}} &= -6 \frac{(4p^2 - 3p + 1)^2 (4p^2 - 6p + 3)(1-2p)p}{(4p^2 - 2p + 1)^5 (1-p)^2}
 \end{aligned}$$

203

204 **4** An Algorithm for Random Low-Density 3XOR

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

209 First of all, because the input is random, the problem may potentially be easy to solve
 210 with low error probability without even observing the input lists, depending on n , the size and

211 the distribution of the input. In light of theorem 2, we see that if N (the size of input lists) is
 212 exponentially smaller than $\rho^{-1/3}$, then “**return** \perp ” is an algorithm that has an exponentially
 213 small probability of yielding false negatives. Alternatively, if N exponentially larger than
 214 $(1-p)^{-n}$, then we expect the string $000\dots 0$ to be present in A, B and C with overwhelming
 215 probability ; it follows that “**return** $(0, 0, 0)$ ” is a fast algorithm with exponentially low
 216 false positive probability. To avoid these pitfalls, our main focus is on the hard case where
 217 $N \approx \rho^{-1/3}$, and where the expected number of solutions in the random input lists is close to
 218 one.

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

$$221 \quad u := \Pr(abc = 110 \mid a \oplus b \oplus c = 0) = p^2 / (1 - 2p + 4p^2)$$

223 The same result is attained for 101 and 011. Two out of the three non-zero options result
 224 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
 225 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
 226 $2u$. This is always smaller than p when $0 < p < 1/2$. In other terms: *random triplets drawn*
 227 *from \mathcal{D} have density p , but random 3XOR triplets drawn from \mathcal{D} have smaller density.*

228 This observation can be exploited in a simple way: we iteratively search for 3XOR
 229 triplets of increasing maximum weight. This yields the INCREMENTAL3XOR function (shown
 230 as algorithm 2). It takes an additional argument w_{\max} controlling the maximum allowed
 231 weight. $w_{\max} = 2nu$ yields an algorithm that reveals a 3XOR triplet present in the input
 232 with probability greater than $1/4$, because the median weight of both \mathbf{x} and \mathbf{y} is $2nu$ if
 233 $\mathbf{x} \oplus \mathbf{y} \oplus \mathbf{z} = 0$. Setting $w_{\max} = 2nu(1 + \epsilon)$ is enough to miss an existing solution with
 234 exponentially small probability. With $w_{\max} = n$, the algorithm deterministically finds a
 235 solution if it is present in the input, but the complexity is much higher.

Algorithm 2 An “incremental” algorithm for the sparse 3XOR problem.

```

1: function INCREMENTAL3XOR-ONE-SIDED( $A, B, C, w_{\max}$ )
2:   for  $0 \leq j \leq w_{\max}$  do
3:     for  $0 \leq i \leq j$  do
4:        $\zeta \leftarrow$  QUADRATIC3XOR( $A_i, B_j, C$ )
5:       if  $\zeta \neq \perp$  then return  $\zeta$ 
6:   return  $\perp$ 
7: function INCREMENTAL3XOR( $A, B, C, w_{\max}$ )
8:   for  $0 \leq i \leq w_{\max}$  do ▷ Bucket sort by Hamming weight
9:      $A_i \leftarrow \{\mathbf{x} \in A \mid \text{wt}(\mathbf{x}) = i\}$ 
10:     $B_i \leftarrow \{\mathbf{y} \in B \mid \text{wt}(\mathbf{y}) = i\}$ 
11:     $\zeta_1 \leftarrow$  INCREMENTAL3XOR-ONE-SIDED( $A, B, C, w_{\max}$ )
12:     $\zeta_2 \leftarrow$  INCREMENTAL3XOR-ONE-SIDED( $B, A, C, w_{\max}$ )
13:    if both  $\zeta_1 = \perp$  and  $\zeta_2 = \perp$  then return  $\perp$  ; otherwise return the solution found.
```

236 It is fairly obvious that `Incremental3XOR-One-Sided` only finds a 3XOR triplet $(\mathbf{x}, \mathbf{y}, \mathbf{z})$
 237 in the input lists if $\text{wt}(\mathbf{x}) \leq \text{wt}(\mathbf{y}) \leq w_{\max}$. By calling it twice, we lift the restriction that \mathbf{x}
 238 must be sparser than \mathbf{y} .

239 4.1 Overwhelming Success Probability

240 Choosing a small value of w_{\max} leads to smaller “filtered” instances and thus to a smaller
 241 running time for the actual computation using the quadratic algorithm. However, if w_{\max} is

242 too small, then a potential solution present in the input might be discarded.

243 A sensible choice consists in picking w_{\max} in the range $]2nu, np[$ — above the expected
 244 density of random 3XOR triplets so that we do not discard them, and below the density of
 245 the input lists in order to actually discard input vectors that are too heavy. In this case the
 246 algorithm succeeds with overwhelming probability as long as the original input contains at
 247 least one solution. The main contribution of this paper is the following

248 ► **Theorem 3.** *Let $e := 2 + 6D(2p^2/(1 - 2p + 4p^2), p) / \ln(1 - p)(1 - 2p + 4p^2)$. For all
 249 $d > e$ there is an algorithm for the random low-density 3XOR problem that runs in time
 250 $\mathcal{O}(N + N^d)$, where N denotes the size of the input list and fails with negligible probability
 251 (in n).*

252 We first discuss some aspects of the result. The graph of the best possible exponent
 253 (e) is shown in Fig. 1. The exponent e increases from zero to 2 as p goes from zero to
 254 $1/2$. Using the bisection algorithm, we find that $e \leq 1$ when $p \leq 0.0957$. It follows that
 255 INCREMENTAL3XOR is *linear* in the size of the input for small p .

256 It seems that $e \leq 2p(1 - 2 \ln p)$; establishing this is a fascinating endeavour that we must
 257 regrettably leave for future work. It is worth noting that the not-so-friendly expression of
 258 e comes from the use of the tight binomial bound (2). It would be greatly simplified had
 259 we instead used the simpler Chernoff bound (1). However, doing so makes the result much
 260 worse for small p : it results in $\lim e = 1$ when p goes to zero (instead of $\lim e = 0$).

261 ► **Lemma 4.** *With $w_{\max} = 2nu(1 + \epsilon)$, if the input contains a 3XOR triplet, then INCRE-
 262 MENTAL3XOR returns \perp with probability less than $2 \exp(-nu\epsilon^2)$.*

263 **Proof.** Assume that the input lists contain a 3XOR triplet $(\mathbf{x}^*, \mathbf{y}^*, \mathbf{z}^*)$. It will be discarded
 264 if and only if the weight of either $\mathbf{x}^*, \mathbf{y}^*$ is greater than w_{\max} . We know that the expected
 265 weight of \mathbf{x}^* and \mathbf{y}^* (and \mathbf{z}^* as well but this is irrelevant) is $2un$, therefore the Chernoff
 266 bound (1) shows that either has weight greater than $2un(1 + \epsilon)$ with probability less than
 267 $\exp(-nu\epsilon^2)$. A union bound (for \mathbf{x}^* and \mathbf{y}^*) then ensures that the solution is discarded with
 268 probability less than $2 \exp(-nu\epsilon^2)$. ◀

269 ► **Lemma 5.** *Let T denote the running time of INCREMENTAL3XOR with $w_{\max} = 2nu(1 + \epsilon)$.
 270 Then $ET \leq N + N^2 \exp(-2nD(2u(1 + \epsilon), p))$.*

271 **Proof.** In the sequel, all the stated complexities must be understood “up to a constant
 272 factor”. Bucket-sorting the input lists by Hamming weight takes time N . Up to a constant
 273 factor, it is sufficient to upper-bound the running time of INCREMENTAL3XOR-ONE-SIDED.
 274 The running time of QUADRATIC3XOR on input (A_i, B_j, \mathcal{C}) is upper-bounded by $|A_i| \cdot |B_j|$.
 275 Therefore, the total running time of INCREMENTAL3XOR is then:

$$276 \quad T = N + \sum_{j=0}^{w_{\max}} \sum_{i=0}^j |A_i| \cdot |B_j| \leq N + \left(\sum_{j=0}^{w_{\max}} |B_j| \right) \left(\sum_{i=0}^{w_{\max}} |A_i| \right)$$

277 Let $\mathfrak{A} = \bigcup_{i=0}^{w_{\max}} A_i$ and $\mathfrak{B} = \bigcup_{j=0}^{w_{\max}} B_j$ denote the lists of all input vectors of weight less
 278 than of equal to w_{\max} . The above inequality shows that the running-time of the algorithm is
 279 less than that of running QUADRATIC3XOR directly on \mathfrak{A} and \mathfrak{B} .

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

284 The sizes of \mathfrak{A} and \mathfrak{B} are stochastically independent random variables following a
 285 binomial distribution of parameters N, s and their expected size is Ns . The expected
 286 running time of the quadratic algorithm on \mathfrak{A} and \mathfrak{B} (which is an upper-bound on that of
 287 INCREMENTAL3XOR) is therefore $\mathbb{E}|\mathfrak{A}| \times \mathbb{E}|\mathfrak{B}| = \mathbb{E}|\mathfrak{A}| \times \mathbb{E}|\mathfrak{B}| = N^2 s^2$. Combining this
 288 with the upper-bound on s gives the announced result. ◀

289 We are now ready to prove theorem 3.

290 **proof of theorem 3.** Let d be a complexity exponent greater than the bound e given in
 291 theorem 3. Let $\epsilon > 0$ be such that

$$292 \quad d = 2 + 6 \frac{D(2u(1 + \epsilon), p)}{\ln(1 - p)(1 - 2p + 4p^2) - 3\epsilon \ln 2}.$$

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

295 Let $N_0 := \rho^{-1/3}$ (input lists of this size contain a single 3XOR triplet in average).
 296 Suppose that $N \leq N_0 2^{\epsilon n}$, where N denotes the size of the input lists ; in this case run
 297 INCREMENTAL3XOR with $w_{\max} = 2un(1 + \epsilon)$. Lemma 4 guarantees the exponentially small
 298 failure probability while lemma 5 tells us that the expected running time T is less than
 299 $N + N^2 \exp -nD(2u(1 + \epsilon), p)$.

300 Set $d' := \log_N(ET - N)$, so that the algorithm runs in time $\mathcal{O}(N + N^{d'})$. A quick
 301 calculation shows that $d' = d$, and the theorem is proved in this case.

302 If $N > N_0 2^{n\epsilon}$, then do the following : slice the input lists in chunks of size $4N_0$ and run
 303 INCREMENTAL3XOR with $w_{\max} = 2un(1 + \epsilon)$ on each successive chunk until a solution is
 304 found. Theorem 2 tells us that each chunk contain 64 3XOR triplets on average, and contain
 305 at least one 3XOR triplet with probability greater than $3/64$. INCREMENTAL3XOR will
 306 reveal an existing 3XOR triplet present in a chunk with probability greater than $1/4$ (this is
 307 true regardless of the value of $\epsilon > 0$). Therefore, a 3XOR triplet will be found in each chunk
 308 with probability greater than $3/256$; because the the chunks are completely independent
 309 parts of the random input, the events “INCREMENTAL3XOR finds a 3XOR triplet in chunk i ”
 310 are independent. Therefore, the whole process fails to reveal a 3XOR triplet with probability
 311 less than $(1 - \frac{3}{256})^{n\epsilon}$. The running time is $4N \cdot N_0^{d-1}$, which is less than N^d . ◀

312 4.2 Deterministic Success

313 We now study the expected behavior of INCREMENTAL3XOR algorithm with $w_{\max} = n$.
 314 When $w \geq np$, then discarding vectors of weight greater than w does not reduce significantly
 315 the size of the lists. Thus, all iterations with “threshold weight” $w \geq np$ cost essentially
 316 $\Omega(N^2)$. It follows that if there is no 3XOR triplet in input lists (for instance, if they are
 317 too short), then the algorithm is essentially quadratic. On the other hand, if the input
 318 lists are so long that they contain many 3XOR triplets, then the probability that they
 319 contain a low-weight triplet (and thus that the algorithm stops early) increases. We therefore
 320 focus on the hardest relevant case, namely input lists of size $N = P(n)\rho^{-n/3}$, where P
 321 is a non-constant positive polynomial. The input lists contain on average $P(n)^3$ 3XOR
 322 triplets and thanks to theorem 2, they contain a 3XOR triplet with probability greater than
 323 $1 - 1/P(n)(1 + o(1))$. The crux of the analysis is that the w -th iteration is done if and only
 324 if the solution has Hamming weight greater than or equal to w . The expected weight of the
 325 solution is $2un < 2np$, and therefore the probability that the most expensive iterations take
 326 place is exponentially small.

327 ► **Theorem 6.** Consider input lists of size $N = P(n)\rho^{-n/3}$ and assume that they contain
 328 a 3XOR triplet. Then INCREMENTAL3XOR with $w_{\max} = n$ returns a valid solution and
 329 terminates in time $\mathcal{O}(N + N^e)$, where

$$330 \quad e = 2 + 3 \frac{2D\left(\frac{1}{1 + \sqrt[3]{\left(\frac{1}{p}-1\right)^2\left(\frac{1}{2u}-1\right)}}, p\right) + D\left(\frac{1}{1 + \sqrt[3]{\left(\frac{1}{p}-1\right)^2\left(\frac{1}{2u}-1\right)}}, 2u\right)}{\ln(1-p)(1-2p+4p^2)}.$$

331 The graph of e is also shown in Fig. 1. We find again that $e \leq 1$ when $p \leq 0.02155$. Thus,
 332 this unfailling algorithm is also linear for small p . It is worthwhile noting that the complexity
 333 is significantly higher than when $w_{\max} = 2un$. The proof is potentially less tight, but it
 334 seems plausible that there is a “heavy tail” effect. Theorem 6 follows from the following

335 ► **Lemma 7.** Let T denote the running time of INCREMENTAL3XOR with $w_{\max} = n$. Then

$$336 \quad \mathbb{E}T \leq N + 4nN^2 e^{-n[2D(x_0,p)+D(x_0,2u)]} \quad \text{where} \quad \frac{1}{x_0} - 1 = \sqrt[3]{\left(\frac{1}{p} - 1\right)^2 \left(\frac{1}{2u} - 1\right)}.$$

337 **Proof.** Let T_j denote the running time of the j -th iteration of the outer **for** loop and S the
 338 number of iterations done when the algorithm stops (i.e. the value of j in the algorithm). As in
 339 the proof of lemma 5, let $\mathfrak{A}_k = \bigcup_{i=0}^k A_i$ and $\mathfrak{B}_k = \bigcup_{j=0}^k B_j$ and we find that $T_j \leq |\mathfrak{A}_j| \times |\mathfrak{B}_j|$.

340 The total running time is then given by $T = \sum_{j=0}^n [S \geq j] T_j$. The two random variables
 341 $[S \geq j]$ and T_j are not independent, however we claim that $\mathbb{E}([S \geq j] T_j) \leq (\mathbb{E}[S \geq j]) T_j$
 342 — i.e. they are negatively correlated. The point is that is that the fact that input lists
 343 contain a “large weight” triplet t^* can only reduce the expected size of low-weight lists by at
 344 most one, and therefore reduce the expected time needed to process them. It follows that
 345 $\mathbb{E}T \leq \sum_{j=0}^n \Pr(S \geq j) \cdot (\mathbb{E}T_j)$.

346 Next, let us consider two random variables following binomial distributions $X_{2u} \sim \mathcal{B}(n, 2u)$
 347 and $X_p \sim \mathcal{B}(n, p)$. From the proof of lemma 5, we know that $\mathbb{E}T_j \leq N^2 s^2$, where $s =$
 348 $\Pr(X_p \leq j)$. In addition, following the same reasoning as in the proof of lemma 4, we have
 349 $\Pr(S \geq j) \leq 2\Pr(X_{2u} \geq j)$. This gives:

$$350 \quad \mathbb{E}T \leq 2N^2 \sum_{j=0}^n \Pr(X_p \leq j)^2 \Pr(X_{2u} \geq j)$$

352 Set $\mu_j := \Pr(X_p \leq j)^2 \Pr(X_{2u} \geq j)$. Our goal is to upper-bound the sum of the μ_j 's. To
 353 this end, we split the sum in three parts:

$$354 \quad \sum_{j=0}^n \mu_j \leq \sum_{j=0}^{2nu} \mu_j + \sum_{j=2nu}^{np} \mu_j + \sum_{j=np}^n \mu_j$$

355 First, we focus on the case where $0 \leq j \leq 2nu$. In this range, $\Pr(X_p \leq j)$ is increasing
 356 with j and $\Pr(X_{2u} \geq j)$ is greater than $\frac{1}{2}$, therefore we find that $\mu_j \leq 2\mu_{2nu}$ for $0 \leq j \leq 2nu$.
 357 A symmetric argument shows that $\mu_j \leq 2\mu_{np}$ for all $np \leq j \leq n$.

358 Let M denote the largest μ_j for $2nu \leq j \leq np$. It follows from the above discussion
 359 that this is the largest of all the μ_j , so that their is less than $2nM$. We use the binomial
 360 tail bound (2) to get an upper-bound on M . Set $f(x) = 2D(x, p) + D(x, 2u)$, so that
 361 $\mu_j \leq e^{-nf(\frac{j}{n})}$ when $2nu \leq j \leq np$. We next seek the maximum of f , and for this we compute
 362 its derivative:

$$363 \quad f'(x) = 2 \log \frac{x}{p} - 2 \log \frac{1-x}{1-p} + \log \frac{x}{2u} - \log \frac{1-x}{1-2u}$$

364 Solving $f'(x_0) = 0$ reveals only one possible real solution, that satisfies:

$$365 \quad \frac{1}{x_0} - 1 = \sqrt[3]{\left(\frac{1}{p} - 1\right)^2 \left(\frac{1}{2u} - 1\right)}$$

366 It appears that $1/x_0 - 1$ is the geometric mean of $1/p - 1$, $1/p - 1$ and $1/(2u) - 1$. This
 367 implies in particular $2u \leq x_0 \leq p$ as expected. ◀

368 **5 A “Sensible” Application**

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

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

$$375 \quad F(x_1, x_2, x_3, x_4) = \text{SHA-512}(x_1) \& \text{SHA-512}(x_2) \& \text{SHA-512}(x_3) \& \text{SHA-512}(x_4).$$

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

$$379 \quad F(x_1, x_2, x_3, x_4) \oplus F(y_1, y_2, y_3, y_4) \oplus F(z_1, z_2, z_3, z_4) = 0$$

380 With the additional constraint that all articles are distinct. There are 5775 ways to dispatch
 381 12 items into 4 indistinguishable urns, so that with our 7700 articles, we can assemble $2^{138.5}$
 382 bundles of 12 publications having a chance to satisfy the above equation (of course the inputs
 383 are correlated, and this deviates from the original problem formulation, but this is not a
 384 serious issue). Alternatively, we may form 2^{47} quadruplets of publication. With $p = 1/16$
 385 and $n = 512$, we could then expect about 40 solutions from our data set. This made us
 386 confident that there would be at least one, but finding it does not seem so easy at first
 387 glance.

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

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

400 We found six solutions, one of which is shown as algorithm 3. Amazingly, it contains the
 401 name of one of the authors of the present article.

402 **6 Conclusion**

403 We presented a simple algorithm for the random sparse “low-density” 3XOR problem, which is
 404 always subquadratic and can even be linear if the density is low enough. It works by reducing

Algorithm 3 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

```

405 an instance of the “low-density” problem to several smaller instances of a “low-weight”
 406 problem, whose sparsity is not exploited at this stage. This begs for finding new algorithms
 407 for the “low-weight” 3XOR problem, which requires completely different techniques. This
 408 other sparse problem is not only interesting in itself, but better algorithms would yield
 409 even faster algorithms for the “low-density” case. This is the subject of ongoing and future
 410 research.

411 **Acknowledgements** We thank Pierre-Alain Fouque, Antoine Joux and Anand Kumar Naray-
 412 anan for useful discussions. We are very grateful to 3 out of 6 anonymous reviewers (so far)
 413 for rejecting two previous versions of this paper while providing extremely helpful feedback
 414 and suggesting new ideas.

415 — **References** —

- 416 **1** R. Arratia and L. Gordon. Tutorial on large deviations for the binomial distribu-
417 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).
418
- 419 **2** Leif Both and Alexander May. The approximate k-list problem. *IACR Transactions on*
420 *Symmetric Cryptology*, 2017(1):380–397, 2017.
421
- 422 **3** Charles Bouillaguet, Claire Delaplace, and Pierre-Alain Fouque. Revisiting and improving
423 algorithms for the 3xor problem. *IACR Transactions on Symmetric Cryptology*, 2018(1):254–
424 276, 2018.
- 425 **4** Martin Dietzfelbinger, Philipp Schlag, and Stefan Walzer. A subquadratic algorithm for 3xor.
426 In Igor Potapov, Paul G. Spirakis, and James Worrell, editors, *43rd International Symposium*
427 *on Mathematical Foundations of Computer Science, MFCS 2018, August 27-31, 2018, Liverpool,*
428 *UK*, volume 117 of *LIPICs*, pages 59:1–59:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik,
429 2018. doi:10.4230/LIPICs.MFCS.2018.59.
- 430 **5** Michael L. Fredman, János Komlós, and Endre Szemerédi. Storing a sparse table with $o(1)$
431 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.
432
- 433 **6** Anka Gajentaan and Mark Overmars. On a class of $\mathcal{O}(n^2)$ problems in computational geometry.
434 *Computational geometry*, 5(3):165–185, 1995.
- 435 **7** Zahra Jafargholi and Emanuele Viola. 3sum, 3xor, triangles. *Algorithmica*, 74(1):326–343,
436 2016. doi:10.1007/s00453-014-9946-9.
- 437 **8** Antoine Joux. *Algorithmic cryptanalysis*. CRC Press, 2009.
- 438 **9** Gaëtan Leurent and Ferdinand Sibleyras. Low-memory attacks against two-round even-mansour
439 using the 3XOR problem. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances*
440 *in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa*
441 *Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II*, volume 11693 of *Lecture Notes*
442 *in Computer Science*, pages 210–235. Springer, 2019. doi:10.1007/978-3-030-26951-7_8.
- 443 **10** Mridul Nandi. Revisiting Security Claims of XLS and COPA. *IACR Cryptology ePrint Archive*,
444 2015:444, 2015.
- 445 **11** Ivica Nikolić and Yu Sasaki. Refinements of the k-tree Algorithm for the Generalized Birthday
446 Problem. In *ASIACRYPT*, pages 683–703. Springer, 2015.
- 447 **12** Rasmus Pagh and Flemming Friche Rodler. Cuckoo hashing. In *European Symposium on*
448 *Algorithms*, pages 121–133. Springer, 2001.
- 449 **13** S.M. Ross. *Probability Models for Computer Science*. Elsevier Science, 2002. URL: <https://books.google.fr/books?id=fG3iEZ8f3CcC>.
450